aculab

ACULAB.COM

Aculab SS7

Installation and administration guide

MAN1202 Revision 6.16.1

PROPRIETARY INFORMATION

Document Revision

Rev	Date	By	Detail
6.2.2	15.09.04	DJL	Full release
6.2.3	03.10.04	DJL	Clarification on signalling link management added.
6.2.4	09.11.04	DJL	Small changes including omitted –cIMP75 option
6.2.5	10.12.04	DJL	Small changes including `rlc_include_cause`
6.3.0	08.02.05	DJL	Introduction of Solaris install information
6.4.0	25.05.05	DJL	Addition of ANSI ISUP information
6.5.0	01.12.05	DSL	Corrections to rev 6.4.0 Distributed ISUP and dual MTP3 Flexible ISUP Move OPC instances to appendix B
6.5.1	21.12.05	DJL	Additional supported parameters/features for China
6.6.0	28.02.06	DSL	Add definition of ISUP and MTP3 variants
6.6.1	04.03.06	DJL	Minor corrections.
6.6.2	19.04.06	DJL	Review updates.
6.7.0	03.07.06	NW	New features for ETSI ISUP and UKISUP
6.7.1	19.07.06	BN	New mp_compat option
6.8.2	02.03.07	NW	New feature Distributed TCAP and SCCP
6.8.3	22.03.07	NW	New ANSI MTP3 parameter sltm_si
6.10.0b1	27.06.08	NW DSL	New feature M3UA, (Sigtran MTP3 User Adaptation Layer). Addition of SCCP parameter ludt. Correct ISUP parameter slc_shift to sls_shift. Removal of AIT examples. New block_circuits option.
6.10.1	12.09.08	WM	Clarified description of parameters affecting ISUP continuity tests.
6.10.2	07.10.08	NW	Addition of new ss7maint commands. Addition of information about SCTP.
6.10.3	30.10.08	NW	Updated after review.
6.10.4	15.12.08	NW/ DSL	Updated SCTP section, addition of MTP2 PCR.
6.10.5	11.03.09	DSL	Added SCCP loadshare, ISUP ACC changes
6.10.7	22.04.09	DSL	Additional parameters added.
6.11.0	03.10.10	DSL	Connection mode SCCP. Allow CICs to multiple destinations. Configuration file `[spdefault]` section.
6.11.3	18.01.11	DSL	Support for `fwdspldr -reconfig` and displaying the current configuration.
6.11.16	09.02.11	DSL	Updated to match 6.11.16 software release. Clarify configuration procedures.
6.12.2	28.06.13	DSL	Add support for IPv6, MTP3 STP and the specification of 24bit pointcodes in 8-8-8 format. Remove Solaris support. Enhancements to ISUP continuity test.

6.13.0	24.10.14	DSL	Typographical corrections, updated to match the 6.13.0 software release.
6.14.0	19.06.17	DSL	Updated to match the 6.14.0 software release. Remove references to the Prosody X rev 1 card.
6.15.0	25.06.18	DSL	Add M2PA signalling links.
6.15.1	31.08.18	DSL	Minor updates to match the 6.15.1 software release.
6.16.0	16.02.22	DSL	Add PX Evo for 6.16 ss7 release and call control 6.8 stripe release.
6.16.1	13.06.23	DSL	Update title page. Minor corrections.

CONTENTS

1 Introduction

This document is designed to help application developers and system integrators install, understand and use Aculab SS7 with Aculab V6 telephony and Aculab TCAP software.

In order to use Aculab SS7, you must also install Aculab V6 telephony software (call and switch drivers) as the product connects to the network using E1/T1 trunks that are under the control of these drivers.

If you wish to develop or run TCAP applications, you will also need to install the Aculab TCAP API. Support for ISUP call control is provided by Aculab's generic call control API, as documented in the ***Aculab call control API guide***.

In various places within this document, you may see references to a tool called `ss7maint`. This is a command-line tool provided with the Aculab SS7 product, which allows you to perform product-related maintenance functions such as manually installing and uninstalling drivers, starting and configuring the protocol stack and obtaining diagnostic information to aid troubleshooting. See section 4.

1.1 System pre-requisites

The software is available for Microsoft windows and Linux. Only 64bit versions of the operating systems are supported, 32bit windows applications are supported.
Please contact Aculab for details of the currently supported versions.

Note **The development of Aculab products and services is continuous; please check the current status of supported operating systems with Aculab support. (email support@aculab.com)**

2 Product Overview

The Aculab SS7 product allows applications to be written that use SS7 signalling protocols for call-related (ISUP) applications handling telephony or data calls, as well as non call-related applications (TCAP) handling aspects such as text messaging or Intelligent Network features.

This section of the document provides a brief overview of the product features and APIs. Further details of these features, and how to configure them are provided elsewhere in this document. The APIs are described in separate documents.

The ***Aculab SS7 Developers Guide*** contains additional guidance for software developers.

TCP/IP connections are used to carry data between chassis (e.g.: distributed ISUP and dual MTP3 configurations) and by TCAP applications. These connections are expected to be over local, reliable, low-latency networks. Both IPv4 and IPv6 connections are supported.

2.1 ISUP

ISUP is the SS7 protocol used for call setup, for both ISDN and non-ISDN calls. ISUP Applications use the generic Aculab Call Control API, in the same way as it is used for other Aculab protocols. The generic API provides all the features necessary for basic call setup and clearing, as well as some more advanced uses, such as some cases of call redirection. This allows application writers to treat Aculab ISUP as “just another protocol”, and many applications should be easily portable between ISUP and other protocols.

Some networks may place more complex demands on the ISUP protocol, requiring an application to encode or decode messages or parameters that are not available through the generic API, or that are a national extension to the protocol. This can be achieved using the flexible ISUP API extensions, as described in section 2.1.1.

The ISUP protocol allows signalling for a large numbers of voice/bearer circuits to be provided by a small number of signalling links. Voice/bearer circuits can be distributed among a number of different chassis, this is described in 2.1.2

2.1.1 Flexible ISUP

The message formats used by ISUP are usually selected based on the ISUP protocol variant chosen for a given signalling relation.

Application developers wishing to send or receive ISUP parameters that are not directly supported by the generic Call Control API can use the `FEATURE_RAW_MSG` functionality provided in the Call Control API. Parameters in outbound messages can be edited (added, replaced or deleted) by an application, with Aculab SS7 automatically managing any changes to ISUP message pointers.

Where non-standard parameters or messages are required, ISUP message formats can be specified via codec (coder/decoder) extensions. They may be applied on a per-point code or per-relation basis. Once configured, Aculab SS7 will automatically receive and transmit messages in the required format, and handle message/parameter compatibility actions, as applicable.

For message types that do not directly affect internal state machines, an entire ISUP message may be transmitted by an application.

2.1.2 Distributed ISUP

The Aculab SS7 product lets you install network access cards that will be used for bearers in a different chassis to that where the signalling links terminate. The Aculab SS7 and V6 telephony software must be installed on all of the systems, only the configuration options differ.

This allows large numbers of bearer circuits to use a small number of signalling links and a single SS7 point code without requiring a large powerful single computer. It also gives a measure of resilience to system failure because the failure of a single ISUP application node won’t affect the operation of the rest of the system.

The distributed ISUP clients communicate with the main system (where MTP3 runs and the

signalling links are terminated) using a proprietary message format over TCP/IP. The number of distributed ISUP clients is only limited by system resources, and the distributed systems do not need to run the same operating system as the main system or as each other.

Note **A system can be configured to accept connections from Distributed ISUP only systems, or to connect to systems running MTP3 with signalling links, but not both.**

2.2 TCAP

TCAP applications are supported by an API library, almost all of the TCAP protocol is handled by the library. Some applications may of course use ISUP as well as TCAP, in which case both the generic call control API and the TCAP API would be used together.

The Aculab TCAP allows application to reside on the same chassis or on a number of different chassis from where the SCCP and MTP3 reside. Distributing the TCAP applications over a number of different chassis gives a measure of resilience to system failures, since the loss of one TCAP application does not affect the operation of the rest of the system.

The distributed TCAP clients communicate with the main system (where SCCP and MTP3 reside) using a proprietary message format over TCP/IP. The number of TCAP applications is effectively limited only by system resources. Distributed TCAP applications can run on a different operating system to each other and from the main system.

TCAP messages received from the network are distributed to the TCAP applications based on SCCP SSN and the first12 bits of the TCAP transaction identifier.

2.3 SCCP

The driver resident SCCP supports applications using the Aculab TCAP and SCCP API libraries.

The SCCP code supports global title translation of the destination address. This is used to determine the destination pointcode (which may be the local system), and to rewrite the global title itself if the message is sent to a remote system. Two destination pointcodes can be specified, which can be used as primary/backup or loadshare.

The SCCP API library supports both connectionless and connection-oriented SCCP. Applications can be distributed across multiple systems (in the same way as the TCAP API library). Many connection-oriented applications can use the same SSN (the first 12 bits of the connection reference are assigned to each application). Only one application can use the connectionless API for each SSN (which must then not be used by TCAP).

2.4 MTP3

The Aculab SS7 drivers automatically handle MTP3 services, as required by TCAP and/or ISUP applications. All the user has to do is configure the network parameters and the drivers will take care of all aspects of link activation and traffic routing.

The MTP3 supports both SEP (signalling end point) and STP (signalling transfer point) functionality.

In many applications, the SS7 network will provide resilience by having a number of replicated applications, each using a different signalling point code. Sometimes however, it may be advantageous to build an application that has no single point of failure. Aculab supports this by the “Dual MTP3” feature, described in section 2.4.1.

2.4.1 Dual resilient MTP3

For additional resilience to system failure you can configure two systems as a dual with the signalling links split between the two chassis. MTP3 will use STP-like functionality to pass messages over a TCP/IP signalling link between the two systems so that they appear as a single network node to the rest of the network.

A single linkset may be split between the two MTP3 systems, or separate linksets used to connect each system to a different STP.

ISUP can be run on the MTP3 systems, inbound traffic will be transferred by MTP3 to the correct system.

Distributed ISUP and distributed TCAP clients can connect to both MTP3 systems (loadsharing based on CIC value). If one of the MTP3 systems fail, then outbound traffic will all be routed via the working system.

Note **Correct operation of a dual MTP3 system requires that the two systems can pass messages to each other. Resilience is achieved by configuring the connections established by each system to use a different physical IP network.**

2.5 MTP2

The Aculab MTP2 runs on dedicated processors on Prosody cards. The cards support (and can saturate) at least 64 signalling links. The signalling links can be connected to any of the external TDM timeslots. Raw hdlc is also supported and used by the ss7 monitor.

High speed signaling links (see Q.703 Annex A) using unchannelised E1 (or T1) to get data rates of 2.0 (or 1.5) Mbit/sec are supported, but only with the normal frame format. Extended sequence numbers are only supported by the SS7 monitor.

See 5.1.8 for MTP2 configuration parameters and 5.2 for firmware download (trunk) parameters.

2.6 SIGTRAN protocols

The SIGTRAN protocols carry parts of the SS7 protocol stack over the IP network. They are mostly used to carry TCAP and SCCP traffic (which can be high volume) but M3UA and M2PA can also carry ISUP signalling messages.

2.6.1 M3UA

The M3UA protocol (MTP3 User Adaptation layer, defined in RFC 4666) carries the MTP3 service interface (i.e. that to SCCP and ISUP) to a remote system using SCTP over IP.

Although designed to carry the MTP3 user interface, it can also be used in a peer to peer fashion to directly connect two SCCP (or ISUP) applications.

M3UA can be used in conjunction with dual MTP3.

M3UA is a separately licensed product. A valid licence is required in order to convey non-trivial amounts of data.

M3UA is implemented within the SS7 driver, so its use is transparent to both ISUP and TCAP applications and is controlled by the ss7 protocol stack configuration file. For a single local point code, remote point codes may be accessible via M3UA or MTP3 and TDM signalling links.

2.6.2 M2PA

The M2PA protocol (MTP2 User Peer-to-Peer Adaption layer, defined in RFC 4165) connects two MTP3 systems using SCTP over IP. It directly replaces an MTP2 signalling link.

M2PA signalling links are created from information in the ss7 driver configuration file and will take precedence over MTP2 links (defined during firmware download) for the same pointcodes and slc.

M2PA is software licensed using the same licence as M3UA.

3 Installing SS7 and related software products

This section explains how to obtain and install Aculab SS7 and related products. After installation, you must configure the product for use as explained in section 5.

3.1 Hardware installation

Aculab SS7 requires the use of Aculab digital network access cards for two purposes:

- To provide voice/bearer circuits when using ISUP.
- To provide signalling links that connect to the network.

Note **The procedure for installing Aculab digital network access cards is explained in the corresponding card Installation guides**

You may install hardware either before or after you have installed the V6 telephony software, however, if you install the hardware before you have installed the V6 telephony software, you may need to **Cancel** the Found New Hardware Wizard dialogue during system startup. Once you have installed the V6 telephony software, you can then select either *Scan for Hardware Changes* by right clicking on the first entry in the Device Manager dialogue, or restart your system as preferred.

3.2 Installing Aculab V6 telephony software with SS7

This software provides the SS7 protocol stack required by Aculab V6 telephony software for ISUP (call control) applications, and Aculab TCAP. You download and install the software package using the Aculab Install Tool (AIT). Generic instructions for installing and using AIT are detailed in the ***Aculab installation tool - FTP downloads utility guide***.

As well as the main ss7 package, you'll need to download the correct MTP2 package for the card you are using from the SS7 Firmware subsection.

4 The ss7maint program

The `ss7maint` program is used for all the ss7 specific configuration, maintenance and status requests.

The command line has the following standard form:

```
ss7maint global_options subcommand subcommand_options
```

The *`global_options`* are:

`-8` Invert decimal/8-8-8 format for 24bit pointcodes.
`-f` Force the action, ignores some minor errors and range checks.
`-v` Verbose output, `-vv` more verbose (etc).
Many of the subcommands have a `-v` option that has the same effect.
`-q` Quiet, suppresses some error messages and dialogue boxes.
`-l count` Limit status requests to `count` items, default 10000.

The valid `subcommand` are listed below; see references for the *`subcommand_options`*:

Subcommand	Description	See sections
`apiaction`	Exercises the ss7maint api library action functions	7.1.1, 7.1.2, 7.2 and 7.3
`apievent`	Exercises the ss7maint api library event functions	8.2.8
`apistatus`	Exercises the ss7maint api library status functions	8.2.7
`configure`	Change driver configuration	6.2.1.2
`decode`	Decode trace file	8.1.2
`filtertrace`	Change driver trace level	8.1.3
`generate`	Transmit MTP3 test data	9.2
`install`	Install driver (windows only)	10.1.4
`ipstatus`	Status of TCP/IP and SCTP/IP connections	8.2.9
`isupstatus`	Status of ISUP bearers and accessibility	8.2.4
`licence`	Software licence (for M3UA)	8.2.1
`linkactivate`	Signalling link activation	7.1.1
`linkinhibit`	Signalling link inhibition	7.1.2
`linkstatus`	Signalling link status	8.2.6
`load`	Load driver	10.1.1 and 10.2.2
`mtp3status`	MTP3 route and loadshare states	8.2.5
`osstatus`	Statistics from the ss7 driver infrastructure.	8.2.10
`prottrace`	Changes MTP2 trace level	8.1.4
`reflect`	Reflect MTP3 test data	9.1
`sccpstatus`	SCCP information	8.2.3
`start`	Load configuration from file and start driver	6.1.1 and 6.2.1.1
`tcapconfig`	Changes tcap application configuration	6.2.3
`tcapstatus`	tcap status from driver or application	8.2.2
`trace`	Read out driver trace	8.1.1
`uninstall`	Uninstall driver (windows only)	10.1.3
`unload`	Unload driver	10.1.2 and 10.2.3

The `ss7maint apievent`, `apistatus` and `apiaction` subcommands use the functions from maintenance API library. These requests will fail if the maintenance API library cannot be loaded.

Note **Do not parse the output from any of the ss7maint commands. It is subject to change without notice at any time.**

5 Configuration

Having installed the software and hardware components as explained in section 3, you now need to configure these products as required for your application. There are several key areas to consider when configuring Aculab SS7. These are:

1. Configuration of the SS7 driver protocol stack.

   The configuration parameters that are common to more than one application, and more than one E1/T1 trunk are kept in a text stack configuration file, as described in section 5.1. This information is usually loaded into the SS7 driver during system startup, see section 6.

2. Configuration of the E1/T1 trunks that host ISUP bearer circuits and signalling links.

   E1/T1 trunks are controlled by firmware files downloaded and configured using the V6 telephony software tools, as described in ***Call, switch and speech telephony software installation guide*** and summarised in section 5.2.

3. Configuration of optional distributed TCAP applications.

   TCAP application parameters are individually configured, as described in the ***Aculab SS7 Distributed TCAP API Guide***.

4. Configuration of optional SCCP applications.

   SCCP application parameters are individually configured, as described in the ***Aculab SS7 SCCP API Guide***.

5. Add any software licences needed for Sigtran M3UA and M2PA.

5.1 The SS7 stack configuration file

The parameters configured here may be common to more than one signalling link, and/or SS7 application.

5.1.1 File format

The stack configuration file is in plain text and may be created or maintained using any ASCII text editor. The file contains a series of section names, parameters for each section may be declared along with subsections as required. Section names are identified as text within parentheses ([...]), each section must be terminated by a matching [End...].

Most sections have a mandatory parameter that acts as a key to the section. All other parameters are optional – however some 'optional' parameters are required in order to make the configuration useful.

Note **Text following a # to the end of the line is a comment and will be ignored.**

Configuration files may include other files, however a section must start and end in the same file. To include a file specify:

```
include "filename"
```

to read `filename` from the current directory, or:

```
include <filename>
```

to read `filename` from `$ACULAB_ROOT/ss7`

The `include <filename>` form is useful for including the Aculab supplied configuration files that define protocol sub-variants.

Note **C/C++ programmers should note that the text `#include` is considered a comment in a stack configuration file.**

Example

```
[SP]
  NI = 2
  LocalPC = 1234
  [ISUP]
  [EndISUP]
  [MTP3]
    [DESTINATION]
      RemotePC = 5678
    [EndDESTINATION]
  [EndMTP3]
[EndSP]
```

Within any section, parameters and nested subsections can usually be declared in any order. White space and blank lines are ignored and text is not case-sensitive.

When you have finished editing your stack configuration file, you start the protocol stack using `ss7maint`, as described in section 6.

Various examples of the stack configuration file are provided in Appendix A: and it is suggested that you refer to these examples whilst reading the following sections describing the file contents. In many instances you may be able to create your configuration by modifying one of these examples. The rest of this section describes the structure of these files.

5.1.1.1 Section Hierarchy

The sections `[SPDEFAULT]`, `[SP]`, `[ISUPCODEC]`, `[GT_TABLE]` and `[MTP2]` appear at the outer level of the configuration file. All the other sections are always nested inside a `[SP]` (or `[SPDEFAULT]`) section. The `[ISUPCODEC]`, `[GT_TABLE]` and `[MTP2]` sections define global databases that other parameters reference by name.

The majority of the sections have a 'mandatory parameter'; this is the key by which the section is referenced.

5.1.1.2 Numbers

Unless otherwise specified, all numeric values are input in decimal. They may be specified in hexadecimal by preceding the value with **`0x`**, decimal by **`0t`** (useful if the default is hexadecimal), octal by **`0o`**, or binary by **`0b`** (unless the default is hexadecimal).

Note **Leading zeros will not cause an octal conversion**

5.1.1.3 Named items

Many parts of the configuration are named so that they can be referenced by other configuration fields and used by status commands.

While the characters in the names are not normally validated, non-alphanumeric characters might get treated as separators. The configuration file text also gets converted to lower case, so it is best to restrict names to lower case letters, digits and underscore.

The maximum length of names does vary, but is at least 15 characters, sometimes 31.

5.1.1.4 SS7 Pointcodes

14bit ITU pointcodes must be specified as numbers.

24bit ANSI or China pointcodes may either be specified as numbers, or in 8-8-8 format. So 15010346 and 229-10-42 are alternate encodings for the same value.

If LocalPC is specified in 8-8-8 format then the status displays and trace will output pointcodes in 8-8-8 format. Pointcodes less than 65536 are always displayed in decimal.

5.1.1.5 Timeouts

All timeouts default to integer milliseconds but may be specified in seconds by including a decimal point or a trailing **`s`**, or in minutes and seconds by using an **`m`**. Thus a two second timeout can be represented by any of `2000`, `2000ms`, `2.0`, `2.`, `2.0s`, `2s`, `0m2` or `0m2s`.

Out of range value timeout values are set to the relevant limit.

5.1.1.6 Host names and IP addresses

The SS7 stack configuration file allows multiple IP addresses (IPv4 and IPv6) to be specified for a remote host, the addresses being tried in turn until the connection succeeds.

Two parameters are used to configure the addresses:

```
host = [name,]hostname
host = [name,]ip_address[,ip_address[,...]]
ipaddresses = hostname
ipaddresses = ip_address[,ip_address[,...]]
```

The **name** is used to identify the address in trace and status commands, if absent the **hostname** or first **ip_address** is used.

If a **hostname** is specified (ie an invalid numeric address) then ss7maint start will call getaddrinfo() to obtain a list of IP addresses.

The **ipaddresses** parameter is used when the section's mandatory parameter is **host** (eg ISUP [REMOTEMTP3]).

TCP (or SCTP) port numbers can normally be specified by a separate `port=n` parameter. Address specific ports can be specified by appending a colon (:) and the port number to the name/address. For numeric IPv6 addresses (which are colon separated) add an interface id of zero before the port number e.g. `::1%0:1234`.

When the section's mandatory parameter isn't **host** (e.g. M3UA client section) ss7maint actually converts a **host** parameter into a **host** subsection containing an **ipaddresses** parameter.

Note **The hostname lookup is done when the configuration file is read, not prior to establishing the connection.**

5.1.1.7 TCP/IP connection passwords

For additional security (and also to make accidental cross connections less likely) the TCP/IP connections made between the ss7 components do an initial 3-way handshake that includes some password-encrypted values.

The password is used to generate a 64bit number used in the encryption key by rotating the value left 5 bits and adding the next password byte. This means that over 13% of all the 64bit values are generated by 13 lowercase letters or underscores.

To avoid parsing problems, restrict passwords to alphanumerics and underscore.

5.1.1.8 Linux network namespaces

Network namespaces allow the Linux IP stack have different routing tables (etc) on different interfaces. While the namespaces have character string names it is difficult for the SS7 kernel code to look them up by name, instead one of three values can be selected for each connection:

`net_namespace=init`
This selects the namespace used by the 'init' process (pid 0).
It is the system-wide default namespace.

`net_namespace=daemon`
This selects the namespace that the ss7maint daemon was started in.
ie: the one that **`ss7maint load`** was run in.

`net_namespace=start`
This selects the namespace that the **ss7maint start** that is loading the configuration record was started in.

Note **Different parts of the configuration can be loaded by different runs of `ss7maint start` in different network namespaces.**

The default is **`net_namespace=start`**.

The connections to cards for MTP2 are always made in the namespace of the process making the firmware download request.

Note **The SS7 driver always holds a reference to the namespace of the daemon and any needed for connections. This will stop the namespaces being deleted.**

5.1.2 [SPDEFAULT] section

The optional `[SPDEFAULT]` section can contain any of the subsections and optional parameters of an `[SP]` section. The configuration defined in the `[SPDEFAULT]` section is applied to all of the following SS7 signalling points before that in their own `[SP]` section.

The `[SPDEFAULT]` could be used to separate application specific parameters (eg ISUP's `auto_inf` parameter) from the network specific parameters, or to define network parameters that apply to all local pointcodes. If appropriate it could be defined in a separate included configuration file.

If an `ss7maint start -o local_pointcode` request specifies a local pointcode that isn't mentioned in the configuration file and an `[SPDEFAULT]` section has been defined then a local pointcode will be created using the parameters from the `[SPDEFAULT]` section.

Note **[SPDEFAULT] sections are not cumulative. A second one will discard the defaults from the first.**

5.1.3 [SP] section

Each `[SP]` section contains the configuration parameters for a single local SS7 Signalling Point.

The protocol stacks and loaded configurations for each local signalling point are completely separate. Nothing internal to the protocol stack will transfer messages between local pointcodes.

Subsections:

The subsections within an `[SP]` section are `[ISUP]`, `[TCAP]`, `[SCCP]`, `[MTP3]` and `[M3UA]`.

Mandatory parameter:

`localPC=`*`nnnnn`*

Where *`nnnnn`* is the local SS7 point code of the Aculab Signalling Point. You must obtain the value for `localPC` from the administrators of the network to which you will be connecting.

Optional parameters:

`ni=`*`n`*

Where *`n`* is a number from 0 to 3 that identifies the SS7 network indicator.

This indicator will become the default used in all MTP3 and ISUP messages generated by this SP. You must obtain the correct value of `ni` from the administrators of the network to which you are connecting.

Typically it will be 2 (national network), it might be 0 (international network), 1 and 3 are very uncommon.

If the parameter is not present, a default value of zero will be assumed.

`variant=ITU`
`variant=ANSI`
`variant=CHINA`

This indicates a protocol variant, which will be inherited by the inner sections. The precise version of each variant may change slightly between major product releases. If you wish to configure an explicit version of the protocol variant, you can do so as described in sections 5.1.4, 5.1.6 and 5.1.7.

If the parameter is not present, a default value of "`ITU`" will be assumed.

`local_hostname=`*`name`*

This sets a name that will be passed to distributed ISUP and dual MTP3 peer systems during TCP/IP connection establishment and used in status and trace records on the remote system.

If the parameter is not present, the system's actual hostname is used.

5.1.4 [ISUP] section

This section indicates that ISUP signalling is in use. You must specify it if you are using ISUP, even if it is an empty section with no parameters. If you are only using TCAP signalling, you do not need an `[ISUP]` section.

Subsections:

Within an `[ISUP]` section, you may declare any number of `[DESTINATION]` subsections, and zero, one or two `[REMOTEMTP3]` subsections.

Optional parameters:

When any of the following parameters are supplied in an `[ISUP]` section, these parameters become the defaults for all destinations served by the local point code. You can also specify any ISUP parameter (except **`mtp3_listen`** and **`password`**) within an ISUP `[DESTINATION]` subsection, in which case they apply only to that destination.

`ni=`*`n`*

Where *`n`* is a number from 0 to 3 that identifies the SS7 network indicator.

This indicator will become the default in all ISUP messages generated by this SP. You must obtain the correct value of `ni` from the administrators of the network you are connecting to.

If the parameter is not present, the value of `ni` will be inherited from the outer `[SP]` section.

`variant=ITU`
`variant=ANSI`
`variant=CHINA`
`variant=ITU_1999`
`variant=ITU_1997`
`variant=ITU_Q767_1991`
`variant=ITU_ETSI_2001`
`variant=ITU_UKISUP_2001`
`variant=ANSI_2005`
`variant=ANSI_1995`
`variant=CHINA_1997`

This parameter specifies the protocol variant used for processing ISUP messages.

The `ITU`, `ANSI` and `CHINA` variants define the message encoding and protocol procedures used by ISUP. The other variants (e.g.: those referring to a specific release of an ISUP protocol specification) modify one of the `ITU`, `ANSI` or `CHINA` variants using the parameters defined later in this section.

This device driver has the default values for the `ITU`, `ANSI` and `CHINA` variants (which are given as the defaults in this document), the other variants are defined in the files `itu_variants.cfg`, `ansi_variants.cfg`, `china_variants.cfg`, `etsi_variants.cfg` and `ukisup_variants.cfg`. The relevant file must be included at the top of the configuration file by specifying (for example):

```
include <itu_variants.cfg>
```

These files should also be consulted to determine the differences between the variants.

It is recommended that if the specific variant is known, and is already available from Aculab, it should be explicitly configured.

The definitions of the variants may change slightly between releases as additional features are fully supported. The default variant is defined for maximum network interoperability.

If the parameter is not present, the value of `variant` will be inherited from the outer `[SP]` section.

Note **To fully comply with an ISUP variant's signalling requirements, application-level usage of Flexible ISUP may be required.**

application_ref=*nn*

This defines a value between 0 and 255 that is returned in `ts_info[]` when `call_maint_port_status()` is called with `rqst` set to `ACU_MAINT_PORT_GET_APPL_REF` (value 3). The application can use the value to identify circuits to a specific destination.

If not specified, the value depends on the ISUP variant, 0 for ITU, 1 for ANSI and 2 for CHINA. Refer to the variant configuration files for the default for other variants.

Although the default value depends on the variant, the parameter can be used whenever the application needs to know which ISUP configuration applies to a timeslot. The SS7 driver code does not look at this value.

Note Values below 8 are reserved for default variants and cannot be set.

codecext=*name*

This allows the user to alter the format of ISUP messages in accordance with a codec extension of ***name***. This parameter may be repeated to allow the application of multiple codec extensions. See section A.1.8 for further information.

mtp3_listen=[y|*port*]

Accept messages from a distributed ISUP client (i.e.: one that is configured with a `[REMOTEMTP3]` section specifying this host) over TCP/IP.

If `y` is specified the default port of 8256 is assumed.

password=*string*

Where *`string`* consists of the characters 'a-z', 'A-Z', '0-9' and '_'. This parameter provides additional security in a distributed ISUP configuration by requiring the client ISUPs to specify the password.

If the parameter is not present, the distributed ISUP clients do not need to specify a password to use this system.

net_namespace=start/init/daemon

The network namespace for distributed ISUP clients.

sls_shift=*n*

This parameter selects which bits of the CIC number are used to generate the signalling link selection code (`sls`) that is used by the MTP3 for load sharing. The least significant ***n*** bits of the CIC number are ignored.

The default is zero so that the least significant bits of the CIC are used for the sls.

Note See section 8.3.2. for information about ISUP load sharing.

cic_width=*nn*

This parameter defines the number of bits in the CIC field of received ISUP messages.

If the parameter is not present, a default of 12 is used for ITU and CHINA, and 14 for ANSI, maximum value 16.

lowest_cic=*nnnnn*
highest_cic=*nnnnn*
cic_mask=*0xnnnnnnnn*
cics_per_trunk=*nn*

These parameters define a range of CIC numbers that are expected to be valid. Messages received for unknown CICs within the specified range will be responded to as if the circuit were locally hardware blocked. CIC numbers specified during firmware download do not have to be within the range. The `cic_mask` defines (modulo `cics_per_trunk`) which CIC numbers within the specified range are valid.

Defining CICs to be valid is particularly useful in a dual MTP3 or distributed ISUP configuration when it is likely the bearers associated with some CICs are attached to a system that is currently inaccessible.

Defaults are lowest_cic=1, highest_cic=0 (i.e. an empty range).
ITU and China cic_mask=0x7fffffff and cics_per_trunk=32.
ANSI cic_mask=0xffffff and cics_per_trunk=24.

More specifically, a CIC is valid if `cic >= lowest_cic && cic <= highest_cic && (cic_mask & (1 << (cic - lowest_cic) / cics_per_trunk))`. CIC numbers that refer to known signalling links are invalid.

`reset_circuits=y/n`

If a value of y is supplied, then a group reset will be done on circuits when a remote point code becomes accessible following a firmware download or after a distributed ISUP system reconnects to MTP3.

If the parameter is not present, a default value of *`y`* will be assumed.

Note The group reset is required to remove the effects of UCIC and blocking messages sent before the firmware download as well as ensuring all old calls are cleared.

`block_circuits=y/n`

If a value of y is supplied, then a hardware group block message will be sent when any circuits become unusable because the SS7 stack is being stopped, or because the connection to a distributed ISUP system has failed.

Circuits that are not locally hardware blocked will be unblocked (to ensure the remote system doesn't believe they are blocked) when the remote system becomes accessible following firmware download or when a distributed ISUP system reconnects to MTP3.

If `reset_circuits=y` is also specified, then a reset is sent after firmware download, and a hardware group unblock at other times.

If the parameter is not present, a default value of *`n`* will be assumed.

Note Any hardware group block message is sent only once, no timers are started.

`send_group_reset_messages=once/twice/never`
`send_group_block_messages=once/twice/never`

These parameters modify the behaviour when processing port blocking and reset requests.

If set to `once` (the default for ITU and China) the group message are sent once.
If set to `twice` (the default for ANSI) the group message is sent twice, this is required by ATIS-1000113.4.2005 for compatibility with T1.113-1988.
If set to `never` then single circuit blocking and reset messages are always used.

Note If `send_group_block_messages=never` then hardware blocks are simulated by resetting active circuits and then maintenance blocking the circuits.

`selection_method=highest`
`selection_method=lowest`
`selection_method=method2`

This modifies the behaviour of the driver when making outgoing calls with a timeslot of `-1` (don't care). Depending upon the selection method chosen, either the highest, or lowest numbered available timeslot within the corresponding E1/T1 trunk will be used.

If "`method2`" is chosen, the selection method will be the method as described in ITU Q.764 §2.9.1.3. (the exchange with the higher pointcode controls the even numbered circuits).

If the parameter is not present, a default value of "`lowest`" will be assumed.

Note The selection algorithms only operate within the E1/T1 corresponding to the user's API call. The driver will not select a timeslot from an E1/T1 other than that specified.

Note See section 8.3.2. for information about ISUP load sharing.

`location=`*`n`*

Where *`n`* is a number from 0 to15 that will be used in the *`Cause`* parameter of a REL message when it needs to be generated within the driver. It will also be used if the user does not specify a "`location`" in a corresponding API call such as `call_disconnect()`.

If the parameter is not specified, a default value of 0 is assumed.

sus,rel=*y/n*

If a value of *y* is supplied then, if a SUS (suspend) message is received from the network and a call is on progress, the Aculab software will immediately release the call.

If *n* is specified, SUS messages will be ignored.

If the parameter is not present, a default value of *n* will be assumed.

sus_incoming=*y/n*

If a value of *y* is supplied then for incoming calls if a SUS (suspend) message is received timer Q767_T6 (ITU and China) or ANSI_ISUP_T6 (ANSI) will be started. On time-out of T6, the call is released.

If *n* is specified, then for incoming calls the SUS message will be ignored.

If the parameter is not present, a default value of *n* will be assumed.

ucic,rel=*y/n*

If a value of *y* is supplied then, if a UCIC (Unassigned Circuit) message is received from the network during call setup, the Aculab software will release the call. If UCIC is received in response to a synchronous maintenance activity from the Call Control API such as timeslot blocking, then the maintenance activity will be aborted immediately with an error.

If *n* is specified, UCIC messages will be ignored, and any synchronous maintenance actions from the Call Control API such as timeslot blocking will block indefinitely whilst awaiting the expected acknowledgement (BLA/UBA/CGBA/CGUA/GRA).

If the parameter is not present, a default value of *y* will be assumed.

send_ucic=y/n

If a value of *y* is supplied then the Aculab software will respond with a UCIC message when a message for an unequipped CIC is received.

If *n* is specified, any messages received for unequipped CICs are silently discarded.

If the parameter is not present, a default value of *n* will be assumed.

abort_sync_maint_request=y/n

If a value of *y* is supplied then the Aculab software will abort a synchronous maintenance request (blocking/unblocking/reset) from the Call Control API upon expiry of a protection timer that requires maintenance personnel to be alerted (T13/T15/T17/T19/T21/T23).

If *n* is specified, synchronous maintenance requests from the Call Control API will block indefinitely whilst awaiting the expected acknowledgement (BLA/UBA/CGBA/CGUA/GRA).

If the parameter is not present, a default value of *y* will be assumed.

olm,rel=*y/n*

If a value of *y* is supplied then, if an OLM (Overload) message is received from the network during call setup the Aculab software will stop the CIC being reused for a period defined by Q764_T3 (TTB). The application will see an immediate release with cause 42 (congestion).

If *n* is specified, OLM messages will be ignored.

If the parameter is not present, a default value of *y* will be assumed. This parameter is invalid for ANSI variants (OLM and TTB are not supported).

ccl,rel=*y/n*

If Chinese ISUP is in use, this parameter allows the driver to automatically release a call if CCL is received from the network.

If *n* is specified, CCL messages will be ignored.

If the parameter is not present, a default value of *n* will be assumed.

rnr_value=*n*

Where *n* is 0 or 1.

If present, this parameter indicates that when the address complete message (ACM) is sent for a call that has been redirected, an RNR (Redirection Number Restriction) parameter will be included in the ACM message.'0' indicates "presentation allowed" and '1' indicates, "presentation restricted".

If the parameter is not present, no RNR parameter will be sent.

nci_value=*nn*

Where *n* is a number from 0 to 255.

If present, this parameter indicates a bit mask that will be logically ORed with the "Nature of Connection Indicators" when that parameter is included in an outgoing IAM message.

send_con=*y/n*

Setting this parameter to *y* enables sending of the CON (Connect) message. The CON message reduces traffic by eliminating the need for an ACM message in some circumstances but it may not be supported in all networks.

If the parameter is not present, a default value of *n* will be assumed. This parameter is invalid for ANSI variants.

auto_acm=y/n

If this parameter is set to *y* and the driver determines that all Called Party Address signals have been received, the driver will send an ACM (Address Complete Message) without any interaction with the user application. The application writer may nevertheless prefer the driver to delay sending of ACM, so that the application can influence the parameter contents through API calls; in this case, the parameter should be set to *n*.

If the parameter is not present, a default value of *n* will be assumed.

auto_inf=y/n

The driver can automatically respond to a solicited Information Request (INR message) that requests the Calling Party Address by transmitting an Information (INF message) without any interaction with the user application. To inhibit automatic transmission of the Calling Party Address in an INF message, set this parameter to *n*.

If the parameter is not present, a default value of *y* will be assumed.

Note The behaviour of this option is affected by the setting of the CNF_RAW_MSG bit on a per-call basis. When CNF_RAW_MSG is not set, the driver will always generate an INF message response. It can only contain a Calling Party Address if requested by the INR.

Note Some ISUP variants (e.g.: UK ISUP) require the application use flexible ISUP to respond to INR requests.

send_msg_cfn=y/n

Note This parameter is only applicable to ANSI ISUP, and will be ignored by other ISUP variants.

When this option is specified (send_msg_cfn=y) for a given signalling relation, CFN will be sent whenever an unrecognised message is received.

If the parameter is not present, a default value of *n* will be assumed.

Note This option only applies to messages; any unrecognised parameters will not cause CFN to be transmitted. Refer to the mp_compat option for details on the handling of unrecognised parameters.

short_redirection_info=*y/n*

This parameter specifies whether a single byte redirection information parameter will be generated when the redirection counter value is 1.

Defaults to y for ITU and China, fixed at n for ANSI.

charge_ind=SFS
charge_ind=GAB

ITU ISUP protocol recommendations do not specify the format for CRG (Charge) messages; the format of these messages is a national option. Aculab have implemented the ability to send these messages for two specific national variants; Finnish ISUP and Gabon ISUP.

If you specify "SFS", then the API call "call_put_charge()" will cause a Finnish ISUP CRG message to be sent.

If you specify "GAB", then the API call "call_put_charge()" will cause a Gabon ISUP CRG message to be sent.

If the parameter is not present, then no CRG messages will be sent.

l2nb=*y/n*

This parameter modifies the behaviour of the API call "call_l2_state()" when it is used for circuits that have been blocked by ISUP protocol Blocking messages.

If *y* is specified, then the blocking state will be ignored and call_l2_state() will report the corresponding timeslot as being "available" as long as the destination node is accessible.

If *n* is specified, then call_l2_state() will report blocked timeslots as being "unavailable" regardless of destination accessibility.

If the parameter is not present, a default value of *n* will be assumed.

report_non_q931_cause=y/*n*

This parameter affects the raw cause value reported by xcall_getcause() when the received cause parameter has a non-zero 'coding standard' (e.g.: ANSI) or any continued octets (which shouldn't happen for ISUP).

If *n* is specified, the raw cause value will be converted to 127 (Interworking Unspecified).

If *y* is specified, the 32bit raw cause will be encoded as follows:

Bits	0-6	Cause value
Bit	7	1 if the 'Cause value' was continued into another octet
Bits	8-12	Always zero
Bits	13-14	Coding standard
Bit	15	1 if the 'Location & Coding standard' octet was continued.
Bits	16-22	Recommendation
Bit	23	1 If the 'Recommendation' octet was continued.

The bit values match those of the received parameter bytes except that the 'continuation' bits are inverted.

If the parameter is not present, a default value of *n* will be assumed.

Note The same encoding can be used to send cause parameters with a non-zero coding scheme.

ss5_compatible_digits=y/*n*

This parameter controls the characters passed to the application when received address fields contain codes 11 or 12.

If set to *n* 11 is reported as b and 12 as c, if set to *y* then 11 is reported as g and 12 as h for compatibility with ss5.

If the parameter is not present, a default value of *y* will be assumed (compatible with older versions of Aculab's ss7).

Note Requests from the application can always contain either set of characters.

rlc_includes_cause=y/*n*

If this parameter is set to y, the SS7 driver may automatically include a Cause parameter in the Release Complete message if a Release message containing an unexpected parameter is received.

If the parameter is not present, a default value of *n* will be assumed.

rlc_cause_value=nn

This specifies the cause value included in `rlc` messages `by rlc_includes_cause=y`.

If the parameter is not present, a default value of 99 (Discard parameter non existent or not implemented) will be assumed

default_clearing_cause=*n*

If present, this parameter will be used as a Cause value in a REL message when an incoming call is rejected. Incoming calls may be rejected for the following reasons:

1. No call handle is available.
2. The IAM contains a Collect Call Request parameter indicating Collect Call Requested, but the `CNF_CALL_COLLECT` bit was not set in the cnf member of `in_xparms`.

If the parameter is not present, a default value of 44 (Requested circuit/channel not available) will be assumed.

t6_clearing_cause=*n*

If present, this parameter will be used as a Cause value in a REL message when ISUP timer T6 expires.

If the parameter is not present, a default value of 102 (Recovery on Timer Expiry) will be assumed.

t7_clearing_cause=*n*

If present, this parameter will be used as a Cause value in a REL message when ISUP timer T7 expires.

If the parameter is not present, a default value of 31 (Normal, unspecified) will be assumed.

t33_clearing_cause=*n*

If present, this parameter will be used as a Cause value in a REL message when ISUP timer T33 expires.

If the parameter is not present, a default value of 16 (Normal call clearing) will be assumed.

default_tmr=*n*

This parameter specifies the value used for the transmission media requirement of an IAM when a value cannot be derived from the `service_octet` and `add_info_octet` provided by the API user.

If the parameter is not present, a default of 2 (64kb unrestricted) will be assumed.

default_voice_encoding=*n*

This parameter specifies the value used for the Octet 3 of the USI parameter of an IAM, when it needs to be set to G.711 A-law or μ-law, and the correct value is not determinable from the API parameters. The parameter may be set to 0xa2 (μ-law), or 0xa3 (A-law).

If the parameter is not present, a default of A-law for will be assumed for ITU and China, or μ-law for ANSI.

default_cpc=*n*

This parameter specifies the value used for the calling party category of an IAM when a value of zero was provided by the API user.

If the parameter is not present, a default of 10 (ordinary subscriber) will be assumed.

default_noa=*n*

This parameter specifies the value used for the nature of address of an IAM when a value of zero was provided by the API user.

If the parameter is not present, a default of 4 (international) will be assumed.

segment=sgm/y/*n*

This parameter specifies whether the SS7 driver should use simple segmentation to convey certain parameters of an overlength message in a segmentation message.

For ANSI variants the additional parameters are passed in a segment message if `segment=sgm` is set (ANSI 2005), and in an unsolicited information response if `segment=y` (ANSI 1995).

If the parameter is not present, a default of *`y`* will be assumed,

dual_seizure_control=Q764
dual_seizure_control=all
dual_seizure_control=none

These options modify the algorithm determining the control exchange during dual seizure situations. The default `Q764` option allocates control based upon CIC and signalling point code values. The exchange with the higher signalling point code controls all even-numbered circuits and the other exchange the odd-numbered circuits.

Some national ISUP variants also allow the determination of control exchange for all circuits in a group by mutual agreement.

`dual_seizure_control=all` selects the local signalling point as the control exchange for all circuits in the relation.

`dual_seizure_control=none` selects the local signalling point as the non-control exchange for all circuits in the relation.

cgsmti2=without_release
cgsmti2=immediate_release

These options are only applicable to ANSI ISUP, and will be ignored by other ISUP variants.
The CGSMTI (Circuit Group Supervision Message Type Indicator) value 0x02 is reserved for national use, and may be received from ANSI T1.113-1988 exchanges. The meaning of this value must be determined by mutual agreement in a multi-vendor situation.

The coding of 0x02 can be configured to mean either:

"block without release"
"block with immediate release"

Aculab ISUP will never initiate a (un)blocking operation using a CGSMTI of 0x02, but will respond to and acknowledge a received value of 0x02 as configured by this option.

mp_compat=none/auto/application

This configuration option controls the handling of message and parameter compatibility information in outgoing messages, and the message and parameter compatibility actions taken for received messages.

If a value of *`none`* is supplied, unrecognised incoming messages and parameters will be discarded in accordance with Q.767. The driver will not insert message or parameter compatibility information in outbound messages.

If a value of *`auto`* is supplied, the driver will assume that “pass on” is not possible during processing of incoming messages. Unrecognised incoming messages and parameters will be handled by the driver in accordance with Q.764, as required by the current ISUP Exchange Type and compatibility information included in the message. The driver will insert message and parameter compatibility information in outgoing messages as required by the selected ISUP variant.

If a value of *`application`* is supplied, the driver will assume that the application will perform transparent passing of incoming messages and parameters using Flexible ISUP. Unrecognised incoming messages and parameters will be handled by the driver in accordance with Q.764, as required by the current ISUP Exchange Type and compatibility information included in the message. The driver will insert message and parameter compatibility information in outgoing messages as required by the selected ISUP variant.

In some circumstances, the driver is able to detect on a call-by-call basis when an application is definitely not using Flexible ISUP. The driver will then assume “pass on” is not possible for this call.

prm_compatibility=*code,compat*

This option is used to define default white book parameter compatibility information for a particular parameter. The parameter code must be in the range 1-255. The compatibility information can be specified as one or two octets. A value of zero deletes the current compatibility information. If one octet is specified, then bit H must be set to a ‘1’. If two octets are specified, then bit P must be set to a ‘1’ and bit H set to a ‘0’. Hence valid ranges are 0x0, 0x80-0xff, 0x8000-0x807f, 0x8100-0x817f, …, 0xff00-0xff7f.
Default values are already defined for ITU-T and China parameters as defined in ETSI EN 300 356-1 V4.2.1 (2001-07).

Parameter compatibility values supplied using Flexible ISUP will override the default values.

default_incoming_exchange_type=typea
default_incoming_exchange_type=typeb
default_outgoing_exchange_type=typea
default_outgoing_exchange_type=typeb

These options specify the default ISUP Exchange Type (relevant during White Book message and parameter compatibility procedures). Type A (terminal exchange) is the default, Type B (transit exchange) may be more appropriate for gateway applications. ISUP Exchange Type may be overridden from the Call Control API during `call_openin()` or `call_openout()`.

apply_continuity_loop=*y/n*

If a value of *y* is supplied, a loopback will be applied to the indicated circuit when requested by a received IAM or CCR message.

If *n* is specified, the application must provide the loopback when requested. The driver will still apply the loopback for CCR messages unless they are being reported to the application.

If the parameter is not present, a default value of *y* will be assumed.

Regardless of this parameter, the driver will run the appropriate protocol procedures and timers, and will release the call if failure of a continuity test is indicated.

Note The SS7 driver has no support for sending or detecting the continuity test tones.

Note Prosody X Evo systems have hardware support for sending and detecting tones. These could be used by the application,

continuity_defer_event=*y/n*

If a value of *y* is supplied, the `EV_INCOMING_CALL_DETECTED` (and `EV_RAW_MSG`) events for an IAM that requests continuity check will be deferred until the continuity test succeeds.

If *n* is specified, the events are generated immediately and the application must wait until the test has completed before removing the audio loopback.

If the parameter is not present, a default value of *y* will be assumed.

Note The events are always generated if the application has to apply the loopback

continuity_check_output_value=default
continuity_check_output_value=none
continuity_check_output_value=*value*

This setting controls the output value (0 to 255) placed on the outgoing circuit upon completion of an incoming continuity test if the driver had earlier applied a loopback.

If *none* is specified, the loopback is left in place.

If the parameter is not present (or set to **default**) then the output is set to A-law silence (213) for ITU/China and µ-law silence (255) for ANSI.

Note Due to timing issues, the output value isn't changed if a continuity test on an IAM succeeds.

ccr_application=*y/n*

If a value of *y* is supplied, a received continuity recheck request (CCR) will be passed to the application as an incoming call. The call details will have `continuity_check_ind` set to `CCI_CCR_TEST`.

If a value of *n* is supplied, received CCR are processed by the driver.

If the parameter is not present, a default value of *n* is assumed.

Setting `ccr_application=y` and `continuity_apply_loop=n` allows the application to apply a 2-wire transponder (tone detect and tone generate).

ccr_auto_lpa=*y/n*

(ANSI only) If a value of *y* is supplied, the driver will send a loopback acknowledge (LPA) in response to a received CCR.

If a value of *n* is supplied and `ccr_application=y` the application is responsible for sending the LPA..

If the parameter is not present, a default value of *y* is assumed.

congestion_steps=*nn*

This option specifies the number of steps (maximum 20) used when applying congestion due to receipt of congestion indications (TFC) from MTP3 (see ITU Q.764 section 2.10.2).

If the parameter is not present, a default value of *3* will be assumed.

Congestion is reduced linearly over the time specified by `Q764_T30` (ITU/CHINA) or `ANSI_ISUP_TACCr` (ANSI).

Note Congestion based on MTP3 TFC is disabled by default, to enable set timer T29 non-zero.

congestion_min_calls=*nn*

This option specifies the number of calls/second that are sent at 100% congestion (maximum 100).

If the parameter is not present, a default value of *5* will be assumed.

acc_target_rejects=*nn*

During automatic congestion control (ACC), the number of call attempts will be adjusted in an attempt to get this number of calls rejected each second with cause 42 and the automatic congestion parameter in the release message.

Set to zero to disable automatic congestion control.

If the parameter is not present, a default value of *2* will be assumed.

Note In a distributed ISUP configuration the ACC algorithm runs on MTP3 system(s) so that the combined call attempts of all the ISUP systems are controlled.

acc_initial_congestion=*nn*

This option specifies the initial level of congestion when automatic congestion control starts. This is set relatively high in order to respond quickly to overload conditions. The initial congestion will not be applied within a configurable interval (`isup_timer_acc_initial`) of a previous congestion indication from the same destination.

The default is 80, giving an immediate 80% reduction in the number of calls placed.

isup_timer_acc_initial=*sssss*

Specifies the minimum interval from the last received REL with cause 42 and ACL1 (or ACL2) or the last outward call that was not placed due to automatic congestion control (ACC) and the application of the high initial restriction when congestion is detected again.

Range 10-300s, default 60s.

isup_timer_acc_filter=*millisecs*

This option specifies the duration (in milliseconds) of the IRF filter bucket used to determine the average call and reject rates for ACC. Increasing this value will reduce the responsiveness of the ACC algorithm, but will also reduce any oscillations caused by network delays.

Range 50ms-2s, default 125ms.

uui_max_receive=*nn*

Specifies the maximum number of received UUI messages that will be queued for an application. If more are received then the oldest ones are discarded.

If the parameter is not present, a default value of *32* will be assumed

upt_on_resume=*y/n*

If this parameter is set to *y*, then the driver will send a UPT (User Part Test) message on MTP resume and wait for a response, usually a UPA (User Part Available) message from the remote ISUP. Local circuits remain unavailable for new calls until a response is received.

If the parameter is not present, a default value of *n* will be assumed.

Note The User Part Test will not be performed if timer Q764_T4 (ITU/CHINA) is configured to zero.

Note For ANSI networks the same procedures are followed using CVT/CVR and timer ANSI_ISUP_T37.

system_x_throttle_rate=rate
system_x_throttle_limit=limit

These parameters are used to restrict the rate at which ISUP sends non call related messages (eg: resets and blocking) in order to avoid overloading some old systems. ***rate*** is the minimum average interval (in milliseconds) between such messages, and ***limit*** is the interval (in milliseconds) over which the average applies.

The defaults are ***rate*** 0 (throttle disabled) and ***limit*** 200ms. Setting ***rate*** to 50 will stop more than 4 messages being sent in each 200ms period.

isup_timer_remote_keepalive=sssss

Specifies the interval between keepalive messages on the TCP links between distributed ISUP systems and the MTP3 server.

Range 2-120s, default 10s, may be set to zero to disable the keepalives.

isup_timer_openin=*millisecs*

Specifies the interval which the driver will permit to elapse between the clearing of a call, and an application calling `call_openin()` to handle a subsequent inbound call on the same CIC.

Range Zero or 50-900ms, default Zero.

Note The default value of zero will disable the timer causing any subsequent inbound call to be immediately cleared.

ISUP Protocol Timers

Because there are some differences between the timer numbers between ITU-T Q.764 and ANSI T1.113.4 two names are defined for common timers. Timers that are common to both definitions have a range than encompasses that of all the definitions - even when the actual timer numbers differ. Either name can be used.

Some of the timers can be configured with a value of zero. This either disables the timer, or causes it to expire almost immediately. See referenced notes (below the ANSI table) for details.

The default values in the tables below are those in the driver. The variants `itu_1997`, `itu_1999`, `itu_etsi_2001` and `itu_ukisup_2001` change the short and long timers from the Q.767 (international ISUP) values below (typically `4s` and `60s`) to the Q.764 values of `15s` and `300s`.

See section 5.1.1.5 for a definition of valid time formats.

`Q764_Tnn=sssss`

Where *nn* is a timer id as specified in ITU-T Q.764 annex A.

Timer	Default	Min	Max	Note	Brief description
Q764_T1	4s	4s	60s		Sent REL (short timer)
Q764_T3	120s	60s	240s	1	Receipt of OLM, TTB interval
Q764_T4	300s	30s	900s	2	User part test (sending UPT)
Q764_T5	60s	60s	900s		Sent REL (long timer)
Q764_T6	90s	10s	120s	3	Receipt of SUS
Q764_T7	20s	20s	30s	3	Sent IAM, waiting for ACM or CON
Q764_T8	10s	10s	15s		Waiting for continuity test to complete
Q764_T9	90s	60s	240s	3	Received ACM, waiting for ANM
Q764_T12	4s	4s	60s		Sent BLO (short timer)
Q764_T13	60s	60s	900s		Sent BLO (long timer)
Q764_T14	4s	4s	60s		Sent UBL (short timer)
Q764_T15	60s	60s	900s		Sent UBL (long timer)
Q764_T16	10s	4s	60s		Sent RSC (short timer)
Q764_T17	60s	60s	900s		Sent RSC (long timer)
Q764_T18	4s	4s	60s		Sent CGB (short timer)
Q764_T19	60s	60s	900s		Sent CGB (long timer)
Q764_T20	4s	4s	60s		Sent CGU (short timer)
Q764_T21	60s	60s	900s		Sent CGU (long timer)
Q764_T22	4s	4s	60s		Sent GRS (short timer)
Q764_T23	60s	60s	900s		Sent GRS (long timer)
Q764_T24	2s	0s	2s		Continuity test tone duration (not used)
Q764_T27	240s	10s	480s		Continuity check failure
Q764_T29	0	0.3s	0.6s	4	Congestion onset
Q764_T30	5s	0.5s	10s		Congestion relief
Q764_T33	12s	12s	15s	5	Sent INR, waiting for INF
Q764_T34	2s	2s	4s	6	Waiting for SGM
Q764_T36	10s	10s	15s	6	Continuity test call

`ANSI_ISUP_Tnn=sssss`

Where *nn* is a timer id as specified in ANSI T1.113.4 Table 3.

Timer	Default	Min	Max	Note	Brief description
ANSI_ISUP_T1	4s	4s	60s		Sent REL (short timer)
ANSI_ISUP_T5	60s	60s	900s		Sent REL (long timer)
ANSI_ISUP_T6	90s	10s	120s	3	Receipt of SUS
ANSI_ISUP_T7	20s	20s	30s	3	Sent IAM, waiting for ACM or ANM
ANSI_ISUP_T8	10s	10s	15s		Waiting for continuity test to complete
ANSI_ISUP_T9	120s	60s	240s	3	Received ACM, waiting for ANM
ANSI_ISUP_T12	4s	4s	60s		Sent BLO (short timer)
ANSI_ISUP_T13	60s	60s	900s		Sent BLO (long timer)
ANSI_ISUP_T14	4s	4s	60s		Sent UBL (short timer)

Timer	Default	Min	Max	Note	Brief description
ANSI_ISUP_T15	60s	60s	900s		Sent UBL (long timer)
ANSI_ISUP_T16	4s	4s	60s		Sent RSC (short timer)
ANSI_ISUP_T17	60s	60s	900s		Sent RSC (long timer)
ANSI_ISUP_T18	4s	4s	60s		Sent CGB (short timer)
ANSI_ISUP_T19	60s	60s	900s		Sent CGB (long timer)
ANSI_ISUP_T20	4s	4s	60s		Sent CGU (short timer)
ANSI_ISUP_T21	60s	60s	900s		Sent CGU (long timer)
ANSI_ISUP_T22	4s	4s	60s		Sent GRS (short timer)
ANSI_ISUP_T23	60s	60s	900s		Sent GRS (long timer)
ANSI_ISUP_T27	240s	10s	480s		Continuity check failure
ANSI_ISUP_T28	10s	10s	10s		Sent CQM, waiting for CQR (not used)
ANSI_ISUP_T33	12s	12s	15s	5	Sent INR, waiting for INF
ANSI_ISUP_T34	10s	10s	15s	6	Continuity test call
ANSI_ISUP_T36	2s	2s	4s	6	Waiting for SGM
ANSI_ISUP_T37	30s	30s	900s	2	User part test (sending CVR)
ANSI_ISUP_Taccr	5s	0.5s	10s		Congestion relief
ANSI_ISUP_Tccrr	20s	10s	40s		Waiting for continuity retest

Note 1 OLM messages are ignored unless `olm,rel=y`.

Note 2 A value of `0` (zero) will disable user part test. `Q764_T4` and `ANSI_ISUP_T37` set the same timer.

Note 3 If your application is serving purely as a Transit Exchange, timers T6, T7, and T9 may not always be required. If you are certain that it is the case, you should set the configured timeout value to `0` (zero) to indicate non-use of these timers. If you are uncertain whether the timers are required, it is safest to assume that they are.

Note 4 A value of `0` (zero) will disable congestion control.

Note 5 A value of `0` (zero) will disable sending of INR.

Note 6 ITU timer T34 is ANSI timer T36 (and v.v.).

5.1.4.1 ISUP [DESTINATION] section

This section appears within an `[ISUP]` section and describes a node within the network which serves as a destination for ISUP traffic.

You do not usually need to configure any `ISUP [DESTINATION]` sections as they are created automatically (with parameters inherited from the outer `[ISUP]` section) when a firmware download specifies ISUP bearer circuits.

An `MTP3 [DESTINATION]` (or M3UA connection) section is also needed for each ISUP destination. An `MTP3 [DESTINATION]` section will be automatically created if there is no M3UA configuration, or if `-cSIGMTP3` is specified on the firmware download command.

Mandatory parameter:

`remotePC=`*`nnnnn`*

Where *`nnnnn`* is the SS7 point code of the network Signalling Point.

You must obtain the value for `remotePC` from the administrators of the network to which you will be connecting.

Optional parameters:

By default, ISUP optional parameters will be inherited from the outer `[ISUP]` section. It may sometimes be useful to supply a different setting of one or more parameters that will apply to this destination only. You may do this using any of the optional parameters listed in Section 5.1.4 (except `mtp3_listen` and `password`).

If an `ISUP [DESTINATION]` section is created by firmware download, and has not been explicitly changed by `ss7maint configure` (see 6.2.1.2), then changes to the server's configuration will also change the destination's configuration.

5.1.4.2 ISUP [REMOTEMTP3] section

This section may appear once or twice within an [ISUP] section. If it is present then ISUP will use signalling links on the specified remote host(s) instead of using the MTP3 on the local system.

ISUP always controls bearers on cards in the local system. Any such ports have to have the relevant ss7 firmware downloaded to them specifying the CIC values but no signalling links.

If specified twice the two remote systems MUST be configured as a dual MTP3 system.

Note **It is invalid to include a [REMOTEMTP3] section and to specify `mtp3_listen=y`.**

Mandatory parameter:

`host=hostname`

Where *hostname* is the name of the remote system that has the SS7 signalling links. If the ipaddresses parameter is absent this must be either a numeric IPv4 or IPv6 address, or a hostname that can be resolved to such an address (see section 5.1.1.6).

Optional parameters:

`ipaddresses=ipaddress[,ipaddress]`

This parameter specifies a list of numeric IPv4 or IPv6 addresses that will be tried in turn when attempting to establish the TCP connection to the mtp3 system.

If the parameter is not present the addresses will be obtained by translating the ***hostname***.

Note **The hostname will be resolved by `ss7maint start`, which will fail if the name cannot be resolved.**

`net_select=n`

This parameter specifies which bit of the CIC number is used to select between the two remote MTP3 systems. Zero would indicate that message for even CICs be sent to the first MTP3 system, and those for odd CICs to the other.

If not specified a default of 3 is used, so that CICs 0-7 are sent to the first MTP3 system, and 8-15 to the second etc.

Note **See section 8.3.2. for information about ISUP load sharing.**

`port=nnnnn`

Specifies the remote TCP/IP port number to use for the connection.

If the parameter is not present, a default value of *8256* will be assumed.

`password=string`

Where ***string*** consists of the characters ‘a-z’, ‘A-Z’, ‘0-9’ and ‘_’. Using a password provides additional security in a distributed ISUP configuration.

If the parameter is not present, this client ISUP will not use a password when connecting to the MTP3 system.

`net_namespace=start/init/daemon`

The network namespace for connection to the mtp3 system.

`trace_remote=y/n`

If set to y then ISUP messages received from the MTP3 system are included in the ss7 trace.

If the parameter is not present, a default value of *y* will be assumed.

5.1.4.3 [ISUPCodec] section

An `[ISUPCodec]` section is an outer level section that contains the configuration parameters for a single ISUP codec extension. A codec extension defines the structure of one or more ISUP messages.

When an `[ISUPCodec]` section is referenced by a `codecext` parameter in an `[ISUP]` (or ISUP `[DESTINATION]` section) the message parameter definitions from the codec extension always completely replace the existing definition.

For examples see A.1.8 and the variant configuration files released in `$ACULAB_ROOT/ss7`.

Subsections:

The only valid subsection within an `[ISUPCodec]` section is `[Msg]`.

Mandatory parameter:

`name=`*`name`*

Where *`name`* is a series of alphanumeric characters. This parameter names the codec extension, so that it can be referenced by a `codecext=`*`name`* parameter in an `[ISUP]` section.

Optional parameters:

None.

5.1.4.4 ISUPCodec [Msg] section

Each `[Msg]` section describes the complete format of an ISUP message. Multiple `[Msg]` sections may occur within a single `[ISUPCodec]` section.

Subsections:

None.

Mandatory parameter:

`code=`*`nn`*

Where *`nn`* is an ISUP message type code (derived from Table 4/Q.763 or equivalent). This parameter specifies the ISUP message type being configured by the current `[Msg]` section.

Optional parameters:

`msg_base=itu/ansi/china/`*`codec_extension_name`*

This optional parameter defines the initial parameters for the message to be those (for the same message code) from one of the driver's built-in variants, or from another codec extension that has already been defined.

Note This must be the first optional parameter in the section.

`mode=rmmsg`

This optional parameter requests that the indicated ISUP message code be removed when the codec extension is applied. Aculab ISUP will treat a message type removed by a codec extension as an invalid message for both reception and transmission.

Note This must be the only optional parameter in the section.

`mode=add`

This optional parameter causes any parameters to be appended to those already defined for the message in this codec extension. When the codec extension is used, the message definition always replaces the existing definition for the ISUP server or destination.

This must be the first optional parameter. Adding parameters is only likely to be useful during dynamic reconfiguration.

`priority=`*`nn`*

Where *`nn`* is a number from 0 to 3 (default 0) representing the ISUP message priority. This parameter is only useful for ANSI ISUP variants.

prm=nn,type,min_len[-max_len]

Where *nn* is an ISUP parameter name code (derived from Table 5/Q.763 or equivalent). *type* is one of D, F, V, O, R and marks the parameter as "Deleted", "Fixed", "Mandatory Variable", "Optional" or "Optional, may be repeated" respectively. *min_len* and *max_len* specify the valid minimum and maximum lengths of the parameter. For Fixed parameters and Optional parameters of constant length, the *max_len* field may be omitted.

The need for an "End of Optional parameters" octet can be inferred if other optional parameters have been specified in a [Msg] section. In this situation, a prm= line for parameter code 0 may be omitted. If the last Optional parameter is deleted it is always replaced with an "End of optional parameter", this must also be deleted in order to remove the 'pointer to optionals' from the final message definition.

Note The message and parameter name codes may be specified in hexadecimal by preceding the digits with 0x, or binary by 0b (see section 5.1.1.2).

Note Fixed and Mandatory variable parameters are added after any existing parameters of the same type.

To simplify the construction of codec extensions, the *min_len* and *max_len* fields follow the semantics of Q.763:

The *min_len* and *max_len* values include:

- For type F parameters: the length in octets of the parameter content.
- For type V parameters: the length in octets of the length indicator and of the parameter content.
- For type O parameters: the length in octets of the parameter name, length indicator and parameter content.

The message definitions can be read back from the driver by ss7maint configure. The scripts ss7_dump_cfg.sh and ss7_dump_cfg.bat (see 8.4.1) will display all the information.

For example:

To list the codec extensions, and the messages they modify:

```
ss7_dump_cfg.sh -c
```

To also show all the parameters:

```
ss7_dump_cfg.sh -c -x-
```

To display the configuration of local pointcode 2020 including the message definitions:

```
ss7_dump_cfg.sh -o 2020 -p isup -x- -x destination
```

To display the parameters of message 1 (IAM);

```
ss7_dump_cfg.sh -o 2020 -p isup -x- -s msg:1
```

The output always contains the full list of parameters that are valid for each message - i.e. it always includes those defined in the driver code and not modified by the codec extensions. Parameters that codec extensions delete will be absent (rather than being marked as deleted).

Note The parameters are formatted with the prm=nn,type,min_len,max_len converted into a [prm] subsection. The length fields are those of the parameter data (they always exclude the parameter name and length indicator bytes)

5.1.5 [TCAP] section

This section is mandatory if distributed TCAP is required. You must specify it and an [SCCP] section if you are using distributed TCAP even if it is an empty section with no parameters.

Subsections:

None.

Mandatory parameters:

There are no mandatory parameters for this section.

Optional parameters:

tcap_listen=*port*

Where *port* is a TCP/IP port number. This parameter allows the default TCP/IP port number to be changed.

If not specified, a default value of 8256 is used.

password=*string*

Where *string* consists of the characters 'a-z', 'A-Z', '0-9' and '_'. This parameter provides additional security in a distributed TCAP configuration by requiring the distributed TCAP clients specify the password when connecting to this system.

If the parameter is not present, then the TCAP clients do not require a password to connect to this system.

net_namespace=start/init/daemon

The network namespace for connections from TCAP applications.

rx_qlen=nnn

Specifies the number of messages that will be queued in the driver for an application before new messages get discarded.

Default 250, range 100 to 20000.

Note Receive messages are also queued in userpace by theTCAP library.

TCAP timers

tcap_timer_remote_keepalive=*sssss*

Specifies the interval between keepalive messages on the TCP/IP links between distributed TCAP systems and the driver. Range 2-120s, default 10s. This timer may be set to zero to disable the keepalives.

5.1.6 [SCCP] section

This section indicates that SCCP is in use. You must specify it if you are using TCAP even if it is an empty section with no parameters.

Subsections:

Within an [SCCP] section, you may declare any number of [CONCERNED] subsections.

Mandatory parameters:

There are no mandatory parameters for this section.

Optional parameters:

variant=ITU
variant=ANSI
variant=CHINA
variant=ITU_1996
variant=ANSI_1996
variant=CHINA_1995

The `ITU`, `ANSI` and `CHINA` variants define the message encoding and protocol procedures used by SCCP. The other variants (e.g.: those referring to a specific release of an SCCP protocol specification) may modify one of the `ITU`, `ANSI` or `CHINA` variants using the parameters defined later in this section.

Aculab's SS7 device driver contains the default values for the `ITU`, `ANSI` and `CHINA` variants (which are given as the defaults in this document). Other variants are defined in the files `itu_variants.cfg`, `ansi_variants.cfg` and `china_variants.cfg`. To use one of the variants in these files, include the relevant file at the top of the configuration file by specifying (for example):

```
include <itu_variants.cfg>
```

These files should be consulted to determine the differences between the variants.

It is recommended that if the required variant is known, and is already available from Aculab, it should be explicitly configured.

The definitions of the variants may change slightly between releases as additional features are fully supported. Default variants are defined for maximum network interoperability.

If the parameter is not present, the value of `variant` will be inherited from the outer `[SP]` section.

ni=*n*

Where *n* is a number from 0 to 3 that identifies the SS7 network indicator.

This parameter is normally absent and the value for ni is inherited from the outer [SP] section.

gt_table=*name*

Where *name* is that assigned to a previously defined [GT_TABLE] section. This parameter includes the named global title translation table into this configuration.
This parameter may be repeated in order to include multiple global title tables.

master=*y/n*

This parameter specifies whether the system should be the master or the slave in a dual mtp3 configuration. Configuring a slave prevents duplicated SCCP management messages being sent into the network. A slave system (master=*n*) will not send SCCP management messages provided the master is accessible to the slave.

If the parameter is not present, a default value of *y* is used.

sccp_listen=y/*port*
This parameter specifies the TCP/IP port number that applications using the SCCP API library should use to connect to this system. If y is specified then the default TCP/IP port number (8256) is used.

If not specified then applications using the SCCP API library cannot use the system.

password=*string*
Where *string* consists of the characters 'a-z', 'A-Z', '0-9' and '_'. This parameter provides additional security by requiring the SCCP clients to specify the password when connecting to this system.

If the parameter is not present, then the SCCP clients require no password to use this system.

net_namespace=start/init/daemon
The network namespace for connections from SCCP applications.

spc_in_addr=calling
This parameter controls if a signalling point code should be placed into the calling party address if the calling party address is routed on ssn. Normally, this parameter should only be set if problems are encountered with a remote system that expects a point code to be present.

If not set, the Aculab SCCP will place a point code into the calling party address in accordance with ITU-T recommendations Q.714.

xudt=required
xudt=never
xudt=always
xudt=mgmt
This parameter determines under what circumstances an SCCP Extended Unit Data (XUDT) message will be sent in preference to an SCCP Unit Data (UDT) message. Received XUDT messages are not affected by this parameter.

If the parameter is set to `never`, then XUDT messages will never be sent. This also disables SCCP segmentation.

If the parameter is set to `required`, then XUDT messages will be used if segmentation is required.

If the parameter is set to `always`, then XUDT messages will be used for all SCCP messages except management messages.

If the parameter is set to `mgmt`, then XUDT will be used for all SCCP messages including management messages. This value is intended for testing purposes.

If the parameter is not present, then a default value of `required` will be assumed.

segment_size=*n*
Where *n* is a number from 160 to 255. This parameter determines the data size that can be included into an outgoing UDT or XUDT message before the message is segmented. The driver will attempt to limit the size of the data included in a UDT or segmented XUDT message to this value until the maximum number of permitted segments for SCCP is reached.

In certain networks, a subsequent SCCP relay point may need to perform global title translation that results in an overall increase in the message size. Limiting the size of the data allows room for the relay point to carry out this operation.

If not specified, a default value of 200 is used.

ludt=never
ludt=required
This parameter determines under what circumstances an SCCP Long Unit Data (LUDT) message will be sent in preference to SCCP segmented Extended Unit Data (XUDT) messages. Received LUDT messages are not affected by this parameter.

If the parameter is set to `never`, then LUDT messages will never be sent.

If the parameter is set to `required`, then LUDT messages will be sent in preference to sending segmented XUDT messages provided that the underlying transport supports sending of messages greater than 272 SIF octets.

If the parameter is not present, then a default value of `never` will be assumed.

ludt_interwork=*y/n*
When there is a possibility that an LUDT message may encounter a node in the network where it is necessary to convert the LUDT to segmented XUDT messages for interworking with narrowband networks, then the LUDT must include the segmentation parameter. This parameter controls whether the segmentation parameter should be included in outgoing LUDT messages.

If not specified, a default value of *y* is used, LUDT messages include the segmentation parameter.

return_all_segments=*y/n*
If a segmented XUDT message is received with the return on error option and an error is encountered, then this parameter determines if the driver should return only the first or all segments.

If not specified, a default value of *y* is used, return all segments.

hopcounter=*n*
Where n is a number from 1 to 15. This parameter specifies the hop counter value to include in outgoing XUDT and LUDT messages.

If not specified, a default of 8 is used.

broadcast=*y/n*
This parameter determines if the driver should use the SCCP broadcast method for informing concerned signalling points of any SCCP or subsystem status changes. If `broadcast=`*n*, the SCCP will only use the SCCP response method.

If not specified, a default value of *y* is used.

uos_on_resume=*y/n*
This parameter determines if the driver should set concerned remote subsystems to user out-of-service and initiate sending of SCCP subsystem status test (SST) messages on MTP resume.

If not specified, a default value of *n* is used.

congestion_max_steps=*n*
Where *n* is a number from 0 to 50. In the Aculab SCCP implementation, the cost of transmitting an outbound message ranges from 0 (no congestion) to 1000ms (a maximum of one transmitted message every second). The congestion_max_steps parameter divides this range of transmission costs into the specified number of discrete congestion steps. During congestion onset, the cost of transmitting an outbound message increases in integral steps up to a configurable maximum (refer to the congestion_min_msgs parameter).

A value of 0 disables SCCP congestion control.

If not specified, a default value of 25 is used.

congestion_min_msgs=*n*
Where *n* is a number from 1 to 100. This parameter limits the maximum cost of transmitting an outbound message to ensure a minimum level of traffic throughput.

If not specified, a default value of 5 is used, equating to a maximum message transmission cost of 200ms.

SCCP timers

If you attempt to set a timer to a value outside the indicated range, it will take the nearest maximum or minimum value of the range. Timers that are not present in the configuration file will be set to the indicated default value.

Tnn=sssss

Where *Tnn* is a timer id as follows:

Connectionless timers:

Timer	Default	Min	Max	Description
Treassembly	20s	5s	50s	Timer waiting to receive all the remaining segments of a segmented message after receiving the first segment.
Tstat_info	5s	5s	20m	Delay between sending SCCP subsystem status test (SST) messages requesting subsystem status information. The timer is started at the indicated value and then doubles upon each expiry up to a maximum time of 20 minutes (1200 seconds).
Ta	0.3s	0.06s	0.6s	Attack timer. Time period during which MTP-STATUS primitives are ignored for setting the congestion level of a remote SCCP node.
Td	5s	1s	10s	Delay timer. Period for decrementing the congestion level after congestion at a remote SCCP has abated.
Tresponse	0.8s	0s	5s	Inhibition timer to limit the number of SCCP subsystem prohibited (SSP) messages sent in response to SCCP receiving messages for a prohibited subsystem.

Connection oriented SCCP timers:

Timer	Default	Min	Max	Description
Tconn_est	1m30	1m	2m	Time waiting for a connection confirm message.
Tias	7m30	5m	10m	Delay before sending an IT (inactivity test) when no response message is expected.
Tiar	16m	11m	21m	Time waiting for any receive message on an idle connection.
Trel	15s	10s	20s	Time waiting for a release complete message, retransmit on timeout.
Trepeat_rel	15s	10s	20s	Time waiting for a release complete following retransmission of a release message, retransmit on timeout.
Tint	60s	1s	60s	Time waiting for release complete following the first retransmission, assumed disconnected on expiry.

The connection oriented timers are passed to the SCCP API library.

The T(guard) timer from Q.724 isn't implemented. Connection references are not allocated in strict sequence so that it is unlikely that a value will be reused following restart. Disabling new SCCP connections for 25 minutes (the standard value for T(guard)) following system restart is likely to be more troublesome!

5.1.6.1 SCCP [CONCERNED] section

This section appears within a [SCCP] section and describes a concerned (remote) SCCP node within a network.

This section is only needed if SCCP management is required. An MTP3 [DESTINATION] section, (or M3UA routing key) must be configured for each concerned destination.

If a destination is not defined as a concerned destination, then SCCP will only provide a basic management service for that destination. Only MTP-PAUSE and MTP-RESUME primitives will be taken into account. MTP-STATUS primitives and all SCCP management messages received will be ignored.

If a destination is defined as concerned, then SCCP will provide an intermediate form of management for that destination. All MTP3 primitives and SCCP management messages that refer to the SCCP as a whole (SSN=1) are acted upon. Management messages that refer to a particular SSN other than SSN=1 are still ignored.

If an SSN is also specified in the [CONCERNED] section for a particular destination, then SCCP will provide a more comprehensive management service, acting upon all received Subsystem Prohibited (SSP), Subsystem Allowed (SSA), Subsystem Test (SST) and Subsystem Congested (SSC) management messages for that SSN.

Subsections:

None.

Mandatory parameter:

`destination=`*`nnnnn`*

Where `nnnnn` is the SS7 point code of a remote network signalling point.

Optional parameters:

By default, SCCP parameters and timers that apply to a particular destination will be inherited from the outer [SCCP] section. Sometimes it is useful to use a different setting for a destination and for that reason, timers `Tstat_info`, `Ta`, `Td`, `Tresponse` and parameters `uos_on_resume`, `congestion_max_steps`, `congestion_min_msgs` may also be specified within this section.

`ssn=`*`n`*
`ssn=`*`n,p,q`*
`ssn=`*`n-q`*

Where `n` is a number from 2 to 255 and denotes a subsystem number. This parameter defines the remote subsystem as being a concerned subsystem. Values may also be specified as a comma separated list of value ranges.

5.1.6.2 SCCP [GT_TABLE] section

This section is used to define a global title translation table for SCCP. This section is only needed if routing on global title is required.

The translation table will be used for incoming messages that are routed on global title and for outgoing messages which are routed on global title but do not contain a valid destination point code.

The translated global title can define two destination point codes. These can be configured as primary and backup or loadshare. The `loadshare` and `Tsls_hold` parameters can be placed in the enclosing [GT_TABLE] section, and will then apply to all rules.

If the first destination pointcode is the local pointcode, then the message will be delivered locally. In this case the translated address must contain an SSN which is used to select the application. The application is passed the untranslated address.

If neither destination pointcode is accessible, then SCCP will forward the message over the dual link before reporting non-delivery. This is particularly useful when M3UA is used as M3UA cannot pass messages to the dual.

The current global title tables, including translation counts and loadshare details can be displayed by running `ss7maint sccpstatus -g`.

Subsections:

Any number of [RULE] subsections may appear within a [GT_TABLE] section.

Mandatory parameter:

`name=`*`name`*

Where *`name`* is a series of alphanumeric characters. Up to 31 characters can be specified. This parameter defines the name by which this global title translation table is referenced.

Optional parameters:

The parameters `tt`, `nai`, `np` and `es` define the selection criteria for this global title translation table. The driver will automatically determine the correct global title indicator from the SCCP variant and the `tt`, `nai`, `np` and `es` parameters supplied.

`tt=`*`n`*

Where *`n`* is a number from 0 to 255. This parameter defines the translation type by which this table is selected.

`nai=`*`n`*

Where *`n`* is a number from 0 to 127. This parameter defines the nature of address by which this table is selected.

`np=`*`n`*

Where *`n`* is a number from 0 to 15. This parameter defines the numbering plan by which this table is selected.

`es=`*`n`*
`es=`*`bcd`*

Where *n* is a number from 0 to 15. This parameter defines the encoding scheme by which this table is selected. Specifying `es=1` or `es=2` is equivalent to specifying `es=`*`bcd`*; a binary coded decimal encoding scheme.

Note **If a numbering plan is specified without an encoding scheme, then an encoding scheme of *`bcd`* is used.**

5.1.6.3 GT_TABLE [RULE] section

This section appears within a [GT_TABLE] section and describes a global title translation rule.

Subsections:

The [RULE] section must contain a [ROUTE_SSN], a [ROUTE_GT] subsection, or a `route=ssn/gt` parameter.

Mandatory parameter:

`gtdigits=`*`nnnn`*
`gtdigits=default`

Where *`nnnn`* is a series of hexadecimal digits '`0`' to '`f`'. Up to 124 digits can be specified. This parameter defines the global title digits to match on for selecting this specific rule. Provided the global title contains the same number or more digits than the `gtdigits` parameter and those digits match then this rule will be selected.

If more than one [RULE] section matches, then the longest match is used.
Specifying `gtdigits=default` is equivalent to specifying a zero length digit string which will match every address.

Optional parameters:

`route=gt/ssn`

This parameter can be specified instead of a `[ROUTE_SSN]` or `[ROUTE_GT]` section.

Follow with all the relevant parameters from the sections below.

5.1.6.4 GT_TABLE [ROUTE_SSN] section

This section appears within a [RULE] section and describes a translation from routing on global title to routing on ssn.

The optional parameters from the [ROUTE_GT] section can also be specified in order to include a global title in a transmitted message that is marked 'route on ssn'.

Subsections:

None.

Mandatory parameter:

destination=*nnnnn*

Where *nnnnn* is the SS7 point code of a network signalling point code.

If the destination is the same as the local signalling point code, then the message will be delivered to a local SCCP user.

Optional parameters:

ssn=*n*

Where *n* is a number from 2 to 255. This parameter defines the subsystem number to be used in the translated global title.

If this parameter is not present, then the ssn from the original global title will be used.

forward_as_xudt=*y/n*

If set to *y* then any received UDT messages that get forwarded will be converted to XUDT. In particular this adds a 'hop counter' to the message stopping indefinite forwarding.

If not specified a default of *n* is assumed, and no conversion takes place.

backup_destination=*nnnnn*

Where *nnnnn* is the SS7 point code of the backup (or loadshare) network signalling point code.

backup_ssn=*n*

Where *n* is a number from 2 to 255. This parameter defines the subsystem number to be used in the translated global title sent to the backup destination.

If this parameter is not present, then the same ssn is used for both the primary and backup destinations.

loadshare=auto
loadshare=no
loadshare=sls*n*

Where *n* is a number from 0 to 3. These parameters define how the traffic is distributed between the 2 destinations. If `auto` is specified then traffic will be split evenly between the destinations. If no is specified then traffic will be sent to the backup if the primary destination is unavailable. If sls*n* is specified then traffic is sent to the backup if bit n (ie: value 1 << n) is set in the sls.

If both destinations are available, then the assignment for class 1 (sequential delivery) is 'sticky' and will not change within the `Tsls_hold` interval of sending a class 1 message.

If the parameter is not present, a default of `auto` is will be assumed.

Tsls_hold=*sssss*

This parameter defines the interval following the transmission of a class 1 message during which further class 1 messages for the same sls will be sent to the same destination.

Range 0-10s, default 200ms.

Note For ANSI SCCP only the four least significant bits of the sls are used for load balancing.

5.1.6.5 GT_TABLE [ROUTE_GT] section

This section appears within a [RULE] section and describes a translation from one global title to another global title.

Subsections:

None.

Mandatory parameter:

`destination=`*`nnnnn`

Where *nnnnn* is the SS7 point code of a network signalling point.

If the destination is the same as the local signalling point code, then the message will be delivered to a local SCCP user. The local user will see the untranslated global title, the translated global title will be sent to the `backup_destination` if the local user is unavailable.

Optional parameters:

The driver will automatically determine the correct global title indicator for the SCCP variant and the `tt`, `nai`, `np` and `es` parameters supplied.

If any of the following optional parameters are not present, then the same value from the original global title will be used.

Any of the parameters `forward_as_xudt`, `backup_destination`, `backup_ssn`, `loadshare` or `Tsls_hold` defined for the `[ROUTE_SSN]` section above may be specified here.

`gtformat=`*`nnnn`
`gtformat=all`
`gtformat=default`

Where ***nnnn*** consists of hexadecimal digits **`0`** to **`f`** and the characters **`?`**, **`-`**, **`*`**, or **`/`**. Up to 124 format characters can be specified. This parameter defines the format string for translating the global title digits as follows:

`0` to **`f`**	Insert this digit into the global title.
`?`	Leave this digit unchanged.
`-`	Delete this digit from the global title.
`*`	Leave any remaining digits unchanged (must be the last character).
`/`	Separates the translation rules for the main destination from those for the backup destination.

Note **Specify `gtformat=all` not `gtformat=*` as the latter will be interpreted as a request to read the current value of the parameter.**

Examples:
To convert the digits 273800 to 01908273800
Use `gtformat = 01908*`
To convert the digits 273800 to 923800
Use `gtformat = --92*`
To convert the digits 273800 to 273802
Use `gtformat = ?????2`

If a `gtformat` is specified (with a non-default value) with route on ssn then a global title is added to the generated message.

If route on GT is specified with the default `gtformat` then the global title digits are added unchanged.

`ssn=`*`n`

Where *n* is a number from 0 to 255. This parameter defines the subsystem number to be used in the translated global title.

A value of 1 (the default) requests that the value from the original message be preserved. If the original message did not contain an ssn 0 will be used.

tt=*n*
tt=none
Where *n* is a number from 0 to 255. This parameter defines the translation type to be used in the translated global title. Specifying tt=*none* removes the translation type from the global title, thereby changing the global title indicator value.

nai=*n*
nai=*none*
Where *n* is a number from 0 to 127. This parameter defines the nature of address to be used in the translated global title. Specifying nai=*none* removes the nature of address from the global title.

np=*n*
np=none
Where *n* is a number from 0 to 15. This parameter defines the numbering plan to be used in the translated global title. Specifying np=*none* removes the numbering plan from the global title.

es=*n*
es=*bcd*
es=none
Where *n* is a number from 0 to 15. This parameter defines the encoding scheme to be used in the translated global title. Specifying es=*1* or es=*2* is equivalent to specifying es=*bcd*; a binary coded decimal encoding scheme. Specifying es=*none* removes the encoding scheme from the global title.

Note **If a numbering plan is specified without an encoding scheme, then an encoding scheme of *bcd* is used.**

5.1.7 [MTP3] section

This section appears within an [SP] section and provides the MTP3 protocol parameters for the signalling point.

An [MTP3] section is not required if the system is only using M3UA to connect to remote systems unless the system is in a dual configuration.

Subsections:

Within an [MTP3] section, you may declare any number of [DESTINATION] subsections, any number of [STP] subsections, and, optionally, a single [DUAL] section.

Mandatory parameters:

There are no mandatory parameters for this section, however the section would have no effect unless at least one [DESTINATION] section or the [DUAL] section is declared.
A [DESTINATION] section will be implicitly created if a firmware download creates a signalling link or ISUP circuits (if no M3UA) to a previously unknown remote pointcode.

Optional parameters:

It is likely that many of the optional parameters defined for [STP] and [DESTINATION] subsections will be included here so that they apply to all the remote pointcodes.

variant=ITU
variant=ANSI
variant=CHINA
variant=ITU_1996
variant=ANSI_1996
variant=CHINA_1990

The ITU, ANSI and CHINA variants define the message encoding and protocol procedures used by MTP3. The other variants (e.g.: those referring to a specific release of an MTP3 protocol specification) modify one of the ITU, ANSI or CHINA variants using the parameters defined later in this section.

This device driver has the default values for the ITU, ANSI and CHINA variants (which are given as the defaults in this document), the other variants are defined in the files itu_variants.cfg, ansi_variants.cfg and china_variants.cfg. The relevant file must be included at the top of the configuration file by specifying (for example):

```
include <itu_variants.cfg>
```

These files should also be consulted to determine the differences between the variants.

It is recommended that if the specific variant is known, and is already available from Aculab, it should be explicitly configured.

The definitions of the variants may change slightly between releases as additional features are fully supported. The default variant is defined for maximum network interoperability.

If the parameter is not present, the value of variant will be inherited from the outer [SP] section.

ni=*n*

Where *n* is a number from 0 to 3 that identifies the SS7 network indicator.

This indicator will become the default used in MTP3 messages generated by this SP. You must obtain the correct value of ni from the administrators of the network to which you are connecting.

This parameter will normally be absent and the value for ni inherited from the outer [SP] section.

`stp=y/n`

If set to `y` then MTP3 will act as a Signalling Transfer Point (STP), otherwise it acts as a Signalling End point (SEP).

If not specified, defaults to `n`.

Note The STP code has not been conformance tested, but is useful for setting up small, or test, networks.

Note An STP system must not also be a dual, use two separate STPs.

`restart=ITU`
`restart=ITU_white`
`restart=ITU_white_v1`

This parameter specifies the signalling point restart behaviour. It is important to set the value expected by the network, normally this will be `ITU_white`.

Setting `restart=ITU_white` requests that the signalling point restart procedures of ITU-T Q.704 White Book be used (T.111.4 section 9 for ANSI networks).

Setting `restart=ITU` modifies the behaviour so that traffic to adjacent pointcodes can be sent as soon as the direct linkset is up (i.e. without waiting for any TRA). This allows interworking with ITU-T Q.704 Blue Book systems. It is also useful when connecting two SEP together as traffic can be sent before `Q704_T20` expires.

Setting `restart=ITU_white_v1` (variant 1) modifies the behaviour such that routes via adjacents are assumed to be inaccessible unless a TRA is received during local pointcode restart. This is required for certain markets.

Defaults to `restart=ITU`, except for ANSI networks where only `restart=ITU_white` is valid.

Section 8.6.4 contains some additional information about MTP3 restart.

MTP3 Restart Timers

`Q704_T20=`*`sssss`*

ITU and China: Specifies the overall MTP restart timer for local pointcode restart.
Range `0s` to `61s`, default `60s`.

Note The stp doesn't separate T18 from T20. When T20 expires any TFP and all the TRA are sent (they will be queued by MTP2). Local pointcode restart then terminates.

`ANSI_MTP3_T22=`*`sssss`*

ANSI only: Time to wait for 'sufficient links' during local MTP restart.
Range `0s` to `60s`, default `5s`.

`ANSI_MTP3_T23=`*`sssss`*

ANSI only: Time to wait for TRA during local MTP restart.
Range `0s` to `60s`, default `20s`.

`ANSI_MTP3_T27=`*`sssss`*

ANSI only: Enforced minimum unavailability for full MTP restart.
Range `0s` to `5s`, default `2s`.

ITU timers T19 and T21 and ANSI timers T25, T26 and T28 are settable for each adjacent pointcode.

5.1.7.1 MTP3 [STP] and [DESTINATION] sections

These sections appear within an `[MTP3]` section and describe a remote node within the network.

The only differences between the two section types is that userpart traffic cannot terminate at a node defined by an `[STP]` section, and that an `[STP]` section always defines an adjacent pointcode.

A `[DESTINATION]` section will be automatically created when you define a signalling link, or when ISUP circuits are added (unless M3UA is also configured and `-cSIGMTP3` isn't specified).

Most of the remote pointcodes will require the same values for most of these parameters; this can be achieved by specifying them in the outer [MTP3] section.

Subsections:

Within a `[STP]` section, you may declare any number of `[ROUTE]` subsections.

Mandatory parameter:

adjacentPC=*nnnnn* ([STP] section)
remotePC=*nnnnn* ([DESTINATION] section)

Where *nnnnn* is the SS7 point code of the remote Signalling Transfer Point. You must obtain the value from the administrators of the network to which you will be connecting.

Optional parameters:

variant=ITU
variant=ANSI
variant=CHINA
variant=ITU_1996
variant=ANSI_1996
variant=CHINA_1990

Under unusual circumstances different remote nodes can use different variant-defined parameter defaults, however the main variant (`ITU`, `ANSI` and `CHINA`) must match that of the `[MTP3]` section.

ni=*n*

Where *n* is a number from 0 to 3 that identifies the SS7 network indicator for this remote pointcode.

This parameter will normally be absent and the value for `ni` inherited from the outer `[MTP3]` section.

MTP2=*name*

Where *name* is that assigned to a previously defined MTP2 section. Any signalling links created to this signalling point will use the MTP2 parameters from the named MTP2 section.

If the parameter is not present, the default MTP2 parameters are used.

Note **The referenced MTP2 section must exist when this is configured, but the MTP2 configuration parameters can be changed later.**

adjacent=y/n

If set to `y`, MTP3 will treat the remote pointcode of a `[DESTINATION]` section as an 'adjacent pointcode'.

Remote pointcodes defined by `[STP]` sections, those referenced by the `adjacentPC` parameter of `[ROUTE]` sections, and those with a direct signalling link are always marked as adjacent.

During MTP restart a TRA message is expected from every adjacent pointcode. Setting `adjacent=y` may be necessary in a dual configuration when all the configured links in the linkset are connected to the dual system.

If not specified, defaults to n.

Note **Once a remote pointcode has been marked as an adjacent, it cannot be unmarked.**

tra_needed=y/n

If set to n, during local pointcode restart, MTP3 will decree that 'Enough TRA messages have been received' without receiving one from this pointcode.

If not specified, defaults to y. Only applicable to adjacent pointcodes.

default_route_priority=*nn*

This parameter indicates that messages for any remote pointcode can be sent to this destination (or stp). It is equivalent to adding a [ROUTE] section that specifies adjacentPC=*thispc* to all the other [DESTINATION]/[STP] sections.

If not specified then a default route will not be added.

prefer_adjacent_via_dual=y/n

If set to y, traffic to an adjacent pointcode that cannot be sent on the direct linkset will be sent to the dual if the direct linkset from the dual is usable (regardless of the route priorities).

If set to n, such traffic will use the normal route priorities (typically being sent to an adjacent STP in preference to the dual).

If not specified, defaults to y.

Note **The default differs from the behaviour of pre version 6.12 systems to match that expected by the remote system.**

sls_rotation_bits=x
sls_rotation_count=y

Where *x and y* are numbers. These parameters, as a pair, specify that SLS bit rotation should be applied to traffic sent by MTP3, which is a common requirement of ANSI protocols.

'x' is the number of bits in the SLS that are to be rotated, starting at the least significant bit.

'y' is the number of positions each bit will be rotated (right shifted with carry over to most significant bit within 'x').

If not specified, a default value of zero is used for both parameters and SLS rotation is not done.

Note **In general, the lower (least significant) bits of SLS will be used to select a linkset, and then the next lower bits will select a link, before applying any SLS rotation. This behaviour is only completely predictable if each linkset has the same number of links, and the number of links, and of linksets, are both powers of 2.**

Note **For ANSI networks configure sls_rotation_bits=5 and sls_rotation_count=1 on everything except C-links and the link to the dual.**

balance_routes=y/n

This parameter modifies the behaviour of MTP3 load balancing rules in the case where some linksets have more links than others.

If set to 'y' then the traffic will be evenly distributed between all links in all linksets.

If set to 'n', traffic will be distributed evenly between available linksets, and then within each linkset the traffic of that linkset will be shared evenly among the links.

If not specified, a default value of 'n' will be assumed for ITU variants, and 'y' for ANSI variant.

si=*lo_si*[-*hi_si*][,…]

This parameter specifies the si values on which MTP3 will carry userpart traffic for each destination. It is necessary if, for example, ISUP traffic to a destination should use MTP3 and SCCP traffic to the same destination use M3UA.

If not specified the default is 2-15 and all traffic will use MTP3.

sltm_si=*si*

Where si is either 1 or 2. This parameter is only valid for ANSI variants. The parameter changes the Service Indicator value for transmitted Signalling Link Test (SLTM) messages. It has no effect on transmitted Signalling Link Test Acknowledgment SLTA messages, which always assume the same `si` value as the received SLTM.

If not specified, a default value of 2 will be assumed.

enable_tfc=y/n

This parameter modifies the behaviour of MTP3 congestion conditions, such as may occur if traffic is received at a faster rate than the application can accept it.

If TFC is not enabled, received traffic will be silently discarded when the traffic rate exceeds the system's capacity to process it. This is permissible, since ISUP and TCAP user parts can recover from such message loss, but it is inefficient and can lead to call failures, as well as adversely affecting overall performance owing to re-transmissions.

If TFC is enabled then as soon as congestion is detected by MTP3, but before the congestion is so severe as to require traffic discard, sending of TFC (Transfer Controlled) messages will commence to the originator(s) of the traffic. Since TFC is sent before any traffic has been discarded, the node that receives the TFC may be able to reduce the traffic rate sufficiently to avoid the need for discard.

MTP3 Timers

Most of the timers exist in both ITU (ITU-T Q.704/707) and ANSI (ANSI T1.111.4/7) although some of the timer numbers differ. The two names will refer to the same configurable value with variant dependant default values.

The local pointcode restart times (`Q704_T20`, `ANSI_MTP3_T22`, `ANSI_MTP3_T23` and `ANSI_MTP3_T27`) are configured in the `[MTP3]` section, see 5.1.7.

Refer to section 5.1.1.5 for valid timeout formats.

Q707_T1=*sssss*
ANSI_SLT_T1=*sssss*

Specifies the time to wait for the `SLTA` response to an `SLTM`.
Range `4s` to `12s`, default `10s`.

Q707_T2=*sssss*
ANSI_SLT_T2=*sssss*

Specifies the rate at which `SLTM` are sent while a link is active.
Range `30s` to `90s`, default `30s`.
May also be set to `0`, `SLTM` will then only be sent during link activation.

Q704_T*nn*=*sssss*

Where *nn* is a timer id as specified in ITU-T Q.704.

Timer	Default	Min	Max	Note	Brief description
Q704_T1	0.8s	0.5s	1.2s	1	Changeover delay
Q704_T2	1.4s	0.7s	2s		Waiting for changeover ack (COA/ECA)
Q704_T3	0.8s	0.5s	1.2s	1	Time controlled diversion delay
Q704_T4	0.8s	0.5s	1.2s		Waiting for changeback ack (CBA) (first attempt)
Q704_T5	0.8s	0.5s	1.2s		Waiting for changeback ack (CBA) (second attempt)
Q704_T6	0.8s	0.5s	1.2s	1	Controlled rerouting delay
Q704_T10	30s	30s	60s	2	Interval between sending routeset test (RST)
Q704_T12	0.8s	0.8s	1.5s		Waiting for uninhibit ack (LUA)
Q704_T13	0.8s	0.8s	1.5s		Waiting for force uninhibit (LFU)
Q704_T14	2s	2s	3s		Waiting for inhibit ack (LIA)
Q704_T15	2s	2s	3s		Waiting to start routeset congestion test
Q704_T16	1.4s	1.4s	2s		Interval between sending routeset congestion test (RST)
Q704_T17	1s	0.8s	1.5s		Delay between link failure and link restart
Q704_T19	67s	60s	69s	3	Guard time at the end of pointcode restart
Q704_T21	64s	0s	65s		Overall timer for adjacent pointcode restart
Q704_T22	200s	90s	360s		Local inhibit test (sending LLT)
Q704_T23	200s	90s	360s		Remote inhibit test (sending LRT)

ANSI_MTP3_T*nn=sssss*

Where *nn* is a timer id as specified in ANSI T1.111.4.

Timer	Default	Min	Max	Note	Brief description
ANSI_MTP3_T1	0.8s	0.5s	1.2s	1	Changeover delay
ANSI_MTP3_T2	1.4s	0.7s	2s		Waiting for changeover ack (COA/ECA)
ANSI_MTP3_T3	0.8s	0.5s	1.2s	1	Time controlled diversion delay
ANSI_MTP3_T4	0.8s	0.5s	1.2s		Waiting for changeback ack (CBA) (first attempt)
ANSI_MTP3_T5	0.8s	0.5s	1.2s		Waiting for changeback ack (CBA) (second attempt)
ANSI_MTP3_T6	0.8s	0.5s	1.2s	1	Controlled rerouting delay
ANSI_MTP3_T10	30s	30s	60s	2	Interval between sending routeset test (RST)
ANSI_MTP3_T12	0.8s	0.8s	1.5s		Waiting for uninhibit ack (LUA)
ANSI_MTP3_T13	0.8s	0.8s	1.5s		Waiting for force uninhibit (LFU)
ANSI_MTP3_T14	2s	2s	3s		Waiting for inhibit ack (LIA)
ANSI_MTP3_T15	2s	2s	3s		Waiting to start routeset congestion test
ANSI_MTP3_T16	1.4s	1.4s	2s		Interval between sending routeset congestion test (RST)
ANSI_MTP3_T17	1s	0.8s	1.5s		Delay between link failure and link restart
ANSI_MTP3_T20	120s	90s	360s		Local inhibit test (sending LLT)
ANSI_MTP3_T21	120s	90s	360s		Remote inhibit test (sending LRT)
ANSI_MTP3_T25	30s	0s	35s	4	Waiting for TRA during adjacent pointcode restart
ANSI_MTP3_T26	12s	12s	15s		Interval between sending TRW
ANSI_MTP3_T28	3s	0s	35s	5	Waiting for TRW during local pointcode restart
ANSI_MTP3_T29	65s	60s	69s	3	Guard time at the end of pointcode restart

Note 1 Delay to avoid message mis-sequencing.

Note 2 Values above 60s are allowed so that network load can be reduced when a lot of systems are inaccessible or due to long term inaccessibility. The actual limit is 1 day.

Note 3 Period during which unexpected TRA and TRW are ignored.

Note 4 Uses the same timer as Q704_T21. It is restarted whenever a TRW is received.

Note 5 This applies at the beginning of local pointcode restart, T28 will expire if two SEP are connected together - both systems will then exit local pointcode restart relatively quickly.

5.1.7.2 MTP3 [ROUTE] section

This section appears within MTP3 `[DESTINATION]` or `[STP]` sections and defines a signalling route towards the network signalling point. A route specifies the adjacent STP (or combined destination/STP) through which traffic will be sent.

A route declared within a destination section defines routing for all traffic, including user part messages.

A route declared within an STP section only defines routing for MTP management messages.

If another destination (or STP) has a `default_route_priority`, then a route via that destination (or STP) is automatically added.

Mandatory parameter:

`adjacentPC=`*`nnnnn`*

Where *`nnnnn`* is the point code of the adjacent network Signalling Point through which traffic is to be sent.

If an `[STP]` or `[DESTINATION]` hasn't been configured for the `adjacentPC` then a `[DESTINATION]` will be automatically created and marked as an adjacent pointcode.

In order for the route to be used the firmware downloads must create one or more signalling links to the `adjacentPC`.

Optional parameters:

`priority=n`
`priority=disabled`

Where *`n`* is a number that indicates the relative priority of this route (0 is the highest priority and 127 the lowest). Setting the priority to `disabled` stops the route being used.

If there are two alternative routes with the same priority, then traffic load will be shared between them when both routes are available, or transferred to the surviving route if one becomes unavailable.

If there are two alternative routes with different priorities, then the lower priority route will only be used if the higher priority route is unavailable.

If the parameter is absent, a default value of 0 is assumed.

Note **You do not always need to explicitly configure [ROUTE] sections. When you configure firmware parameters for a signalling link that connects directly to a destination, it is assumed that a route for traffic using the direct signalling link will be available. If you do not declare such a route in your stack configuration file, one will be created automatically with a priority of 0.**

5.1.7.3 MTP3 [M2PA] section

This section appears within MTP3 `[DESTINATION]` or `[STP]` sections and defines an M2PA signalling link to the adjacent pointcode.

This section can be repeated in order to create multiple M2PA signalling links to the adjacent system. Multiple connections will need different SCTP port numbers.

Note **Processor outage is not fully supported, remote processor outage will immediately take the link down.**

Mandatory parameter:

`slc=`*`n`
Where *`n`* is the signalling link code (0 to 15) for the link.

Optional parameters:

`host=[name,]hostname`
`host=[name,]ip_address,ip_address...]`
The **`name`** is used to identify the SCTP connection in trace and status commands, if absent the **`hostname`** or first **`ip_address`** is used.
If a **`hostname`** is supplied it is resolved to a list of ip addresses by ss7maint start.
Each **`ip_address`** is the numeric IP address of the system to connect to, or allow connections from. The list can contain both IPv4 and IPv6 addresses. See section 5.1.1.6.

Note **The hostname will be resolved by 'ss7maint start'. If the name cannot be resolved 'ss7maint start' will fail.**

`port=`*`nnnnn`
Specifies the `SCTP` (or `TCP`) port number to use for this connection. This is the remote port for an outward connection and the local port for a listener.

If the parameter is not present, then the standard M2PA port number of *`3565`* will be assumed.

`local_ipaddresses=n.n.n.n[:ppp][,n.n.n.n]...`
`local_ipaddresses=n:n::n:n[%0:ppp][,n:n::n:n]...`
Specifies the local IP addresses that will be bound to the socket. This determines the local addresses placed in the SCTP INIT chunk (INIT_ACK for a server). If *`ppp`* is specified it will be used as a source port number for an outward connection.

If not specified all local addresses are placed in the INIT chunks, and an unallocated local port number is assigned by the operating system.

`password=`*`string`
Where *`string`* consists of the characters 'a-z', 'A-Z', '0-9' and '_'. This parameter provides additional security for M2PA when connecting to other Aculab equipment.

If the parameter is not present, a password is not needed to use this system.

As well as giving additional security, using password sequence allows the same SCTP/TCP port to be used by more than one connection.

Note **The use of passwords is not part of the M2PA standard and this parameter should only be set if connecting to other Aculab equipment that has also been configured for passwords.**

`net_namespace=start/init/daemon`
The network namespace for the M2PA connection.

`listen=n/y`
This parameter defines if the SCTP layer should act as a client and make the outgoing connection or act as a server and listen for incoming connections.

If not specified, a default value of *`n`* will be used, and an outward connection attempted.

max_retx_qlen=*nnn*

This parameter defines the maximum number of unacknowledged messages that will be held in the retransmit queue. If the messages are 'retrieved' by MTP3 they are still counted until they are transmitted on a different signalling link.

When the limit is reached messages are queued applying back pressure flow control on the sender.

If not specified a default of 127 is used, range 30 to 65535.

max_rx_msu=*nnnn*

This parameter defines the number of received messages that will be passed to MTP3 before the link is treated as busy. Messages are no longer counted once they have been processed by a userpart or queued for an application.

If not specified a default of 2048 is used, range 30 to 65535.

proving_length=*nnn*

This parameter defines the number of bytes of 'pad' added to proving messages. The pad is filled with increasing byte values.

If not specified a default of 200 is used, range 0 to 255.

data_ack_timer=default/*sssss*
data_ack_count=*nnn*

These two parameters control how long M2PA waits before sending an acknowledgement message for received data.

An explicit acknowledgement message (an empty data message) is sent if there are either **data_ack_count** messages to acknowledge or the **data_ack_timer** expires.

If **data_ack_timer=default** then an acknowledgment message is sent on the next 'tick' of the 50ms timer used by the SS7 protocol stack. All other values get rounded up to ensure the specified interval has actually elapsed.

If **data_ack_count=0** an acknowledgement is sent immediately. This is the behaviour required by RFC 4165 but causes an acknowledgement message be sent for every received message – even ones that solicit a response from MTP3. The acknowledgement and response will also almost always end up in separate ethernet frames.
If not specified a value of 16 is used for **data_ack_count**.

M2PA Timers

M2PA uses the same timers as MTP2 (but with slightly different semantics). RFC 4165 (section 4) does not give defaults or ranges deferring to the 'applicable' MTP2 standard. This would only make any sense if M2PA were implemented (somehow) as an alternate physical layer for an existing MTP2 implementation.

The default timer values below mostly match those of Q.703 (ITU) and T1.111 (ANSI) but the valid ranges have been widened.

T*n*=*sssss*

Where *n* is the timer id.

Timer	ITU	ANSI	Min	Max	Brief description
T1	40s	13s	1s	100s	Alignment ready
T2	30s	30s	1s	140s	Not aligned
T3	1.5s	1.5s	1s	100s	Aligned
T4	8.2s	8.2s	1s	100s	Normal proving period
T4e	0.5s	0.6s	0.2s	100s	Emergency proving period
T5	50ms	50ms	0s	500ms	Status retransmission (see note)
T6	3s	3s	1s	10s	Remote congestion
T7	2s	2s	0.5s	10s	Excessive delay of acknowledgement

Note **Timer T5 is used for retransmitting all status messages (not just busy), if set to zero status messages will only be sent once.**

Note **The timeouts are guaranteed to be at least as long as the configured value. Since a 50ms 'tick' is used setting T5=50ms gives retransmissions (about) every 100ms.**

5.1.7.4 MTP3 [DUAL] section

This section appears within an `[MTP3]` section and indicates that the system is part of a dual `MTP3` configuration.

Mandatory parameter:

`host=hostname`
Where *`hostname`* is either the name of the dual host or a numeric IPv4/IPv6 address of the host (see section 5.1.1.6).

Note At least one `listen` or `connect` parameter must be given.

Optional parameters:

`ipaddresses=ipaddress[,ipaddress]`
This parameter lists the numeric IP addresses that should be tried in order to connect to the dual system.

If not specified ss7maint start will resolve the hostname into a list of addresses.

Note The 'ss7maint start' will fail if it cannot resolve the hostname.

`listen=slc[:port]`
`connect=slc[:port]`
These options create a signalling link on the specified **`slc`** (0 to 15) to the peer of the dual system. `connect` requests an outward call be made, and `listen` waits for an inward call. The optional *`:port`* specifies the TCP port number, if it is not specified then a default value of 8256 will be used. The TCP port number can be shared with other dual links and distributed ISUP and TCAP.

These parameters may be repeated to create additional signalling links, however there is usually no reason to specify more than one of each.

Defining one `listen` and one `connect` signalling link will cause the linkset between the dual systems to be established as soon as the second system starts.

Note The dual system should be configured with the same slc and port numbers, but with `listen` and `connect` swapped over.

`password=`*`string`*
Where *`string`* consists of the characters 'a-z', 'A-Z', '0-9' and '_'. This parameter provides additional security in a dual MTP3 configuration by requiring the two MTP3s to specify the password specified before the interconnecting signalling links can be established.

If the parameter is not present, then no password is required for the dual MTP3 systems.

`net_namespace=start/init/daemon`
The network namespace for the connection to the MTP3 dual.

`master=`*`y/n`*
This parameter specifies whether the system should be the master or slave of a dual configuration. One system should be configured as master and the other as slave.

The slave (`master=n`) will send Signalling Route Set Test and Traffic Restart Allowed messages (and ANSI Traffic Restart Waiting and Signalling Route Set Congestion Test messages) via the master which will filter them so that external systems don't see duplicate requests.

If the parameter is not present, a default value of *`y`* will be assumed.

`config_routes=`*`y/n`*
This parameter specifies whether the system should automatically define secondary routes of priority 64 over the dual system to each destination specified in the configuration file. If `config_routes=`*`n`* then routes over the dual system should be defined using a standard `[ROUTE]` section with `adjacentPC=dual`.

If the parameter is not present, a default value of *`y`* will be assumed.

Note The routes via the dual should normally be the lowest priority routes to any destination.

5.1.8 [MTP2] section

Each [MTP2] section defines a named set of MTP2 parameters that can later be applied to all the signalling links that connect to a specific STP. An `[MTP2]` section must be defined before it is referenced from within one or more MTP3 `[DESTINATION]` or `[STP]` sections, the values are passed to MTP2 whenever MTP3 requests the link be established.

Note **Unknown parameters are not detected until they are passed to MTP2 during link establishment. The errors will then be traced. The rest of the parameters may be ignored.**

The MTP2 parameters may also be set as firmware download parameters (`-cMTP2_`*`xxx=yyy`*), or temporarily using `ss7maint configure -p mtp2` *`xxx=yyy`*.

Mandatory Parameter:

`name=xxxx`
This defines the name by which this set of `MTP2` parameters is referenced by an `[MTP3]` section.

Optional Parameters:

`aerm_tin=nn`
The Errors limit for normal proving, see Q.703 10.3. The default is 4.

`aerm_tie=`*`nn`*
The Errors limit for emergency proving, see Q.703 10.3. The default is 1.

`iac_m=`*`nn`*
The number of proving attempts, see Q.703 10.3. The default is 5.

`suerm_t=`*`nn`*
The number of receive signal unit errors that will cause an error indication to MTP3, see Q.703 10.2. The default is 64.

`suerm_d=`*`nn`*
The number of receive signal units required to decrement the count of receive signal unit errors, see Q.703 10.2. The default is 256.

`tx_n1=`*`nn`*
The limit of the number of unacknowledged MSUs. The default value is `127` (unlimited).

`tx_n2=`*`nn`*
The limit of the number of unacknowledged MSU bytes. The default value is `65535` (unlimited).

`trace_fw=[`*`y/n`*`]`
Enable (`y`) or disable (`n`) the firmware function call trace. The default value is `y`, enabled.

`trace_rx=[flmv]`
`trace_tx=[flmv]`
Controls the tracing of receive and transmit message units.
`f` – trace `FISU`
`l` – trace `LSSU`
`m` – trace `MSU`
`mv` – trace all `MSU` (including out of sequence `MSU` when using `PCR`)
The default is **`flm`** so that all transmit and receive messages units are traced. Repeated `FISU` and `LSSU` are never traced.

The rest of the parameters are not valid on Prosody 1U enterprise cards.

`rx_cong_thresh=`*`nn`*
Sets he receive buffer threshold before congestion is set and SIB sent.. The default is 96, range 24 to 128.

`set_misc_flags=`*`nn`*
`clr_misc_flags=`*`nn`*
Set or clear one of the global `-f` flags to `ss7_mtp2_pmxrev3`, see Prosody X rev 3 MTP2.

speed=[*8/16/24/32/40/48/56/64*]

Specifies the speed of the signalling link in k bits/sec.
This overrides the value specified by the `-cSPEEDnnk` firmware download parameter and also allows the speed to be set through the monitor API.

timeslot=*nn*

Explicitly specifies the TDM timeslot used to carry the MTP2 data between the HDLC logic and the TDM switch.

Fixing the timeslot makes it easier to arrange to monitor the data, or to use the TDM switch matrix to route the data to/from somewhere non-standard.

The requested timeslot can be between 0 and 126. Timeslots are normally allocated from 0 upwards, so 64 to 126 are not used (timeslot 127 generates continuous SIOS).

If more than one signalling link tries to use the same timeslot, all the transmit data will come from one of the links.

This parameter is most useful when specified as the firmware download parameter `-cMTP2_timeslot=nn` along with `-cNOSW` when the links are going to be monitored. See 5.2.2. and the monitor API guide.

Timeslots 0 to 31 are on stream 48, 32 to 63 on stream 49, 64 to 95 on stream 50 and 96 to 126 on stream 51.

Note **Only the MTP2 code knows about this change, as far as the MTP3 knows, the signalling link is still associated with the port and timeslot from the firmware download parameters. Specify `-cNOSW` to stop the driver from trying to modify the TDM switch.**

monitor_tx_ts=*nn*

SS7 monitor only. Monitor the traffic being transmitted on the specified timeslot (between the HDLC logic and the TDM switch).

MTP2 protocol timers

T*nn*=*sssss*

Where *nn* is a timer id as specified in ITU-T Q.703.

The default values depend on whether the `-cT1` switch is specified when the firmware is downloaded, see section 5.2. The T1 defaults are those for an ANSI network.

Timer	E1 Default	T1 Default	Min	Max	Description
T1	40.1s	13.0s	6.5s	65.5s	Aligned ready (waiting for first FISU)
T2	30.0s	30.0s	2.5s	65.5s	Not aligned (waiting for SIO)
T3	1.49s	11.5s	0.5s	28.0s	Aligned (waiting for first SIN/SIE)
T4	8.2s	2.3s	1.2s	19.0s	Normal proving period (sending SIN)
T4e	0.5s	0.6s	0.23s	19.0s	Emergency proving period (sending SIN/SIE)
T5	0.08s	0.08s	0.04s	0.24s	Interval between sent SIB
T6	3.01s	3.0s	0.5s	12.0s	Remote congestion
T7	1.99s	1.99s	0.25s	4.0s	Excessive delay of acknowledgement

Out of range timers will be set to the nearest limit value.

Note **The Octet counter is fixed at one error per 16 byte times (effectively 2ms).**

5.1.9 [M3UA] section

This section appears within an `[SP]` section and provides the Sigtran M3UA protocol parameters for the signalling point.

Subsections:

Within an `[M3UA]` section, you may declare any number of `[ROUTING_KEY]` subsections and any number of `[CLIENT]`, `[SERVER]`, `[IPSP_CLIENT]` and `[IPSP_SERVER]` subsections.

The Aculab M3UA code requires that IPSP single exchange (SE) endpoints be explicitly configured to initiate `[IPSP_CLIENT]` or respond to `[IPSP_SERVER]` ASPTM and ASPSM messages. Double exchange (DE) mode is not supported.

Mandatory parameters:

There are no mandatory parameters for this section. However this section would have no effect unless at least one `[ROUTING_KEY]` subsection is declared and at least one of the `[CLIENT]`, `[SERVER]`, `[IPSP_CLIENT]` or `[IPSP_SERVER]` subsections is declared.

Optional parameters:

`variant=ITU`
`variant=ANSI`
`variant=CHINA`

This parameter is used by the system to determine the length of the MTP routing label for an SS7 network.

This parameter should normally be absent and the value for `variant` inherited from the outer [SP] section.

`ni=`*`n`*

Where *`n`* is a number from 0 to 3 that identifies the SS7 network indicator.

This parameter is normally absent and the value for `ni` is inherited from the outer [SP] section.

`msg_burst=`*`y/n`*

If set to *`y`*, then the transmit message rate limit can be exceeded for 10 seconds. The limit is then halved until the average is below the licensed limit. This allows short throughput measurements be taken without requiring an unrestricted license.

Note This parameter is global and applies to all local pointcodes.

5.1.9.1 M3UA [ROUTING_KEY] section

This section appears within an `[M3UA]` section and defines an M3UA routing key. A routing key is a label for a set of remote addresses.

A single routing key can only be referenced from one type of node. That is, a routing key can be referenced either from a number of `[CLIENT]` sections or `[SERVER]` sections or `[IPSP_CLIENT]` sections or `[IPSP_SERVER]` sections, but not mixed.

Subsections:

Within an `[ROUTING_KEY]` section, you should declare one or more `[ADDRESS]` subsections.

Mandatory Parameter:

`routing_context=n`

Where *`n`* is a 32 bit unsigned integer. This parameter defines the routing context for this routing key. It is also used as the Local-RK-Identifier when sending registration requests.

Optional Parameters:

`traffic_mode=loadshare`
`traffic_mode=override`
`traffic_mode=broadcast`

This parameter defines traffic mode type for this routing key.

If the parameter is set to *`loadshare`*, then traffic will be shared between all the connections that use this routing key.

If the parameter is set to *`override`*, then traffic is sent to the last connection to use this routing key.

If the parameter is set to *`broadcast`*, then traffic is sent to all of the connections that use this routing key (each connection receives a copy of every packet).

Note The Aculab M3UA uses the SLS value in the Data message to control loadsharing. Each connection will get traffic for an equal number of the active SLS values.

Normally the traffic mode is defined by the referencing client or server section, or by a received ASP Active request.

`min_asp=n`

Where *`n`* is a number in the range 1 to 65535. If the traffic mode used is loadshare the routing context will not be notified as `AS-ACTIVE` until `min_asp` client systems have activated it.

If not specified, a default value of 1 is used.

`auto_register=y/n`

If set to *`y`*, or if the `CLIENT`/`IPSP_CLIENT` section's `auto_register` parameter is set to *`y`*, then routing keys will be registered when the client connection establishes. If neither is enabled registration must be requested through the maintenance api. The default is *`n`*.

`auto_activate=y/n`

If set to *`y`*, or if the `CLIENT`/`IPSP_CLIENT` section's `auto_activate` parameter is set to *`y`*, then routing keys will be activated following registration. If neither is enabled activation must be requested through the maintenance api. The default is *`n`*.

Note The CLIENT sections default to enabling registration and activation.

M3UA Routing Key Timers

`t_recovery=sssss`

Time interval messages are buffered waiting for the routing key to become active (e.g. reconnecting after some failure). After timeout of `t_recovery` messages are discarded.

Range 0.1s to 100s, default 2s

t_loadshare=sssss

This parameter determines the idle time interval after which data for an sls may be reassigned to a different SCTP connection in order to loadshare traffic evenly.

Range 0.1s to 100s, default 1s

5.1.9.2 M3UA [ADDRESS] section

This section appears within an M3UA `[ROUTING_KEY]` section and describes the address information for remotes node. Traffic from the identified nodes is thus associated with the routing key.

If routing key management is used, then each address section generates an address parameter triplet within a routing key. The local point code from the SS7 stack configuration file is always used as the 'Destination Point Code' in an M3UA registration message.

Mandatory Parameter:

There are no mandatory parameters for this subsection, however at least one `remotePC` should be defined.

Optional Parameters:

si=n
***si*=n,p,q,**

Where *n* is a number from 2 to 15 and denotes an MTP service indicator that this address supports. More than one value may be specified as a comma separated list.

SCCP uses `si 3` and ISUP `si 5`.

If not specified all si values are enabled.

remotePC=nnnnn

Where *nnnnn* is a signalling point code in the remote network or at the remote end to which traffic is to be sent.

Multiple `remotePC` parameters may be specified.

broadband=y/n

This parameter defines if the remote end can support a broadband message length of 4096 SIF octets.

If not specified, a default value of *n* will be used, maximum SIF octet length of 272 octets.

Note SCCP will not send ludt messages unless they are also enabled in the SCCP configuration.

5.1.9.3 M3UA [CLIENT] section

This section appears within an [M3UA] section and indicates that the M3UA is an application server process (ASP) that wishes to use the services of a signalling gateway process (SGP) to send and receive messages from an MTP network.

An [MTP3] section need not be defined in an M3UA client configuration unless the configuration is also a dual redundant one, when an [MTP3] section is needed in order to configure the link between the dual pair. The link between the dual pair cannot carry outbound traffic that would be sent using M3UA, traffic that is locally undeliverable will be transferred over the dual link before being rejected (or discarded).

Note **SCCP global title translation will forward to the dual peer if the requested destination(s) are not accessible.**

Mandatory Parameter:

name=*name*

Where *name* is a series of alphanumeric characters. Up to 15 characters can be specified. This parameter defines the name by which this M3UA Client can be referenced in ss7maint commands.

Optional Parameters:

Any of the parameters below (except **host**) may be placed in the outer [M3UA] section, in which case they apply to all client and server sections.

host=[name,]hostname
host=[name,]ip_address,ip_address...]

The **name** is used to identify the SCTP connection, if absent the **hostname** or first **ip_address** is used.
If a **hostname** is supplied it is resolved to a list of ip addresses by ss7maint start.
Each **ip_address** is the numeric IP address of the system to connect to, or allow connections from. The list can contain both IPv4 and IPv6 addresses. See section 5.1.1.6.

If multiple IP addresses are specified, then each address will be tried in turn in order to establish the connection.

If this parameter is repeated, then separate connections will be established to each host (this only makes any sense if the traffic mode is loadshare).

Note **The hostname will be resolved by 'ss7maint start'. If the name cannot be resolved 'ss7maint start' will fail.**

port=*nnnnn*

Specifies the SCTP (or TCP) port number to use for this connection. This is the remote port for an outward connection and the local port for a listener.

If the parameter is not present, then the standard M3UA port number of *2905* will be assumed.

local_ipaddresses=n.n.n.n[:ppp][,n.n.n.n]...
local_ipaddresses=n:n::n:n[%0:ppp][,n:n::n:n]...

Specifies the local IP addresses that will be bound to the socket. This determines the local addresses placed in the SCTP INIT chunk (INIT_ACK for a server). If *ppp* is specified it will be used as a source port number for an outward connection.

If not specified all local addresses are placed in the INIT chunks, and an unallocated port number is assigned by the operating system.

routing_context=*n*

Where *n* is a routing_context that identifies one of the [ROUTING_KEY] subsection.

The routing key defines a set of remote addresses to be associated with this connection. Multiple routing contexts can be specified. At least one routing context must be specified, otherwise no traffic will be associated with the connection.

Note **The referenced [ROUTING_KEY] section must be defined earlier in the configuration file.**

sctp_mode=client
sctp_mode=server
This parameter defines if the SCTP (or TCP/IP) layer should act as a client and make the outgoing connection or act as a server and listen for incoming connections.

If not specified, then the default matches the m3ua section type.

If set to `server` the `host` parameter defines the remote addresses from which connections are allowed.

sctp_nodelay=y/n
This parameter controls any Nagle-like algorithm that defers data transmission in order to merge data chunks into a single ethernet frame.

If not specified, a default value of *y* will be used and data chunks will be transmitted immediately.

use_tcp=y/n
This parameter changes the underlying transport from the default of SCTP to TCP/IP.

If not specified, a default value of *n* will be used, and SCTP is used as the transport.

heartbeat=y/n
This parameter controls the sending of M3UA ASPSM heartbeat messages. The default is to send them on TCP connections but not on SCTP connections. Received heartbeat messages are always responded to.

password=*string*
Where *string* consists of the characters 'a-z', 'A-Z', '0-9' and '_'. This parameter provides additional security for M3UA when connecting to other Aculab equipment by requiring the Client or IPSP_Client to connect using a password before this system can be used.

If the parameter is not present, the M3UA Clients or IPSP_Clients do not need to specify a password to use this system.

As well as giving additional security, using password sequence allows the same SCTP/TCP port to be used by more than one service (this includes the TCP port used by the Aculab distributed ISUP and TCAP products).

Note **The use of passwords is not part of the M3UA standard and this parameter should only be set if connecting to other Aculab equipment that has also been configured for passwords.**

net_namespace=start/init/daemon
The network namespace for the M3UA connection.

max_data_streams=n
Where *n* is a number from 1 to 16. The value will be rounded down to the nearest power of 2, giving values of 1,2,4,8 and 16. The parameter defines the maximum number of outbound data streams (not including stream 0) for the SCTP connection. Since SCTP negotiates the number of streams with the remote end, the number of streams actually used may be less than requested.

The least significant bits of the `sls` are used for the SCTP stream of data messages.

If not specified, a default value of 4 will be used.

traffic_mode=loadshare
traffic_mode=override
traffic_mode=broadcast
This parameter defines the traffic mode type for this client, it controls the local behaviour (if multiple connections use the associated routing keys) and the traffic mode requested from the remote system.

See Section 5.1.9.1 for a definition of the traffic modes.

If not specified, the value from a referenced routing key is used, if none are defined there, then a default value of *loadshare* will be used.

If a routing key also defines a traffic mode it must be the same as that defined here.

loadshare_master=y/n

If specified implies **traffic_mode=loadshare**. Connections with **loadshare_master=y** take precedence over those with **loadshare_master=n** this gives master/slave functionality.

If not specified the **traffic_mode** setting is used.

key_management=disabled
key_management=enabled
key_management=dynamic

This parameter controls the M3UA routing key management procedures.

If the parameter is set to *disabled*, then all routing key management procedures are disabled. Received registration and deregistration messages will be responded to with a management error message indicating code "Unsupported Class".

If the parameter is set to *enabled*, then routing key management procedures are enabled. A Client or IPSP_Client will use M3UA registration messages to register the routing keys defined with the routing_key parameter. A Server or IPSP_Server will check incoming registration requests against its list of routing keys defined with the routing_key parameter and reject any registration requests that do no match.

If the parameter is set to *dynamic*, then routing key management procedures are enabled. Routing keys and routing contexts are dynamically allocated for Servers and IPSP_Servers. They do not need to be defined with a separate [ROUTING_KEY] section. For Clients and IPSP_Clients, this value has the same effect as *enabled*.

If not specified, a default value of *disabled* will be used, disable routing key management.

encode_route_ctx=y/n

This parameter determines if a routing context parameter should be included in outgoing M3UA messages.

If not specified, a routing context will only be included if routing key management is enabled (set to *enabled* or *dynamic*), or if messages received from the server contain a routing context.

asp_id=n

Where *n* is a 32 bit unsigned integer. This defines the optional application server process (ASP) identifier parameter that identifies a unique ASP.

If not specified, an ASP identifier will not be included in outgoing messages.

net_appear=n

Where *n* is a 32 bit unsigned integer. This defines the optional network appearance parameter that may be used to identify the SS7 network context.

If not specified, a network appearance will not be included in outgoing messages.

auto_connect=y/n

This parameter determines whether the connection will be established by 'ss7maint start' or whether it will only be established on request through the maintenance api. The default is *y*.

auto_register=y/n

This parameter determines whether the routing keys will be registered when the connection establishes, or whether registration must be requested through the maintenance api. If *n* is specified here, the auto_register parameter of the routing key is checked. The default is *y*.

auto_activate=y/n

This parameter determines whether the routing keys will be activated when registration completes, or whether activation must be requested through the maintenance api. If *n* is specified here, the auto_activate parameter of the routing key is checked. The default is *y*.

initial_daud=y/n
This parameter determines whether M3UA DAUD (destination state audit) messages will be sent following route key activation to request the accessibility of remote point codes. If set to *n* (the default) it is initially assumed that all remote addresses are accessible.

rfc3332_compat=*y/n*
This parameter improves interworking with M3UA server implementations that comply with RFC3332 (which was obsoleted by RFC4666). In particular if the AS state is unknown when an ASPAC ACK is received the AS state is assumed to be AS-Active. The default is `n`.

tx_stuck_tmo=sssss
This parameter specifies the time interval (default 2 seconds) after which it is assumed that an SCTP connection that has not accepted any new data is not actually sending it to the remote system. Messages queued for the connection are discarded and further messages will be sent using other connections in the 'loadshare' group.

Without this algorithm all the messages that are accepted from a TCAP application get queued on the broken connection and the flow control with the application stops it sending any additional messages.

max_tx_msgs=nn
This parameter limits the number of messages from all applications that can be queued for a single M3UA connection. If exceeded messages will be sent by other connections in the loadshare group.

Normally the per-application limit on the number of transmit messages (100 per TCAP ssap) stops this limit being reached.

Default 2000.

min_tx_msgs=nn
This specifies the number of transmit messages than can remain allocated when it is determined that an M3UA connection is no longer 'stuck'.

M3UA Connection Timers

t_nn=*sssss*
Where *nn* is a timer id as follows:

Timer	Default	Min	Description
t_connect	10s	1s	Time to wait for an SCTP/TCP connection to establish. If this timer expires the request is aborted and retried. Applicable if `sctp_mode=client`.
t_ack	2s	1s	Time to wait for an M3UA acknowledgement message. If this timer expires the connection will be disconnected. Applicable to Clients and IPSP_Clients.
t_beat	10s	1s	Time interval between sending M3UA Heartbeat messages (if enabled).
t_retry	20s	1s	Minimum time interval between attempts to establish an SCTP/TCP connection to a specific host.
t_daud	30s	30s	Time interval before sending an M3UA DAUD message to verify the unavailability of a remote point code (similar to MTP3's T10). Applicable to Clients.
t_wait_notify	0.2s	0s	If `encode_route_ctx` is defaulted. Time interval to wait for a notify message from the server that contains the correct routing context before activating a routing key. Applicable to Clients.

The maximum value for all these M3UA timers is `1000s`.

5.1.9.4 M3UA [SERVER] section

This section appears within an `[M3UA]` section and indicates that the system is a signalling gateway process (SGP) that can offer a gateway service between an application server process (ASP) and the MTP network. When M3UA is configured as a server, an `[MTP3]` section must also be defined to have any effect.

The routing contexts referenced are configured with the same values as for the client, that is they contain the remote point code of the peer application, not that of the M3UA client.

Note **The Aculab SS7 MTP3 only passes traffic for its own local pointcode to userparts so an ASP wishing to connect to this SGP must have the same local point code.**

Parameters:

The parameters listed in Section 5.1.9.3 are all valid, with the following changes:

`host`
Specifies the remote hosts that can connect to this server. If not specified then any remote host can connect.

`routing_key` and **`key_management`**
If **`key_management`**=**`disabled`**, then the addresses corresponding to all the specified routing keys are enabled on every connection.

If **`key_management`**=**`enabled`**, then each routing key received in a registration request must exactly match one of the configured keys specified for the server (or any defined key if none are specified).

Additionally, if **`key_management`**=**`dynamic`** then a received routing key that doesn't match any part of any existing key will cause a new key to be created.

`sctp_mode`
If not specified, then the default of `server` is used.

`auto_register`, **`auto_activate`** and **`asp_id`**
These parameters are not used.

`net_appear`
If specified then clients must be configured to use the same value. If not specified, then clients should not include a network appearance.

Note **The network appearance cannot be used to select between different ss7 protocol stacks.**

5.1.9.5 M3UA [IPSP_CLIENT] section

This section appears within an `[M3UA]` section and indicates that the system is an IP server process (IPSP) that wishes to directly send and receive messages with another peer IPSP without using the services of an MTP3 network. The Aculab M3UA uses the single exchange method to establish a connection with a peer IPSP. Configuring an IPSP_Client indicates that this IPSP will initiate the M3UA ASPSM and ASPTM procedures.

Parameters:

The parameters listed in Section 5.1.9.3 are all valid, with the following changes:

`routing_key`
The routing key should only refer to the single remote point code of the IPSP_Server system.

5.1.9.6 M3UA [IPSP_SERVER] section

This section appears within an `[M3UA]` section and indicates that the system is an IP server process (IPSP) that wishes to directly send and receive messages with another peer IPSP without using the services of an MTP3 network. Configuring an IPSP_Server indicates that this IPSP expects the remote end to initiate the M3UA procedures.

Parameters:

The parameters listed in Section 5.1.9.3 are all valid, with the changes from Section 5.1.9.4 additionally:

`routing_key`
The routing key should only refer to the single remote point code of the IPSP_Client system.

5.2 SS7 firmware parameters

The E1/T1 trunk layer 1 parameters and the assignment of the TDM timeslots to MTP2 signalling links and ISUP circuits are specified when the ss7 firmware is downloaded to the port. See section 6

The parameters will typically be different for each installation and trunk, you must determine the correct values required for the network to which you are connecting.

The parameters are all specified in the same configuration string, but are described in here is separate groups for clarity.

Parameters that set flags can be negated by adding a hyphen after the **c** e.g.: **-c-PCR** will cause further signalling links to use the default basic error correction method.

The parameters are shown here in upper case, the parsing is actually case-independent.

5.2.1 Layer 1 parameters

These parameters apply to the E1/T1 trunk itself, they should be specified only once.

-cNCRC This instructs the firmware not to use layer 1 CRC4 framing (E1 links) or CRC6 (T1 ESF links). CRC4 framing isn't normally used for ISUP.

-cIMP75 This modifies the line impedance to 75 Ohms for coax cables.

-cRXMON This modifies the line receiver characteristics to those appropriate for use with a resistive (or other passive tap), for use with the ss7 monitor.

-cT1 Initialise the physical layer for T1 operation at 1.536Mb/s (rather than E1 operation at 2Mb/s). The 24 timeslots are numbered 0 to 23. The MTP2 parameters will be set to the ANSI defaults instead of the ITU defaults.

-cAMI When used with `-cT1`, this instructs the firmware to use Alternate Mark Inversion (AMI) line encoding and 56kb/s signalling (rather than 64kb/s and B8ZS).

-cD4 When used with `-cT1`, this instructs the firmware to use D4 framing (rather than Extended Super Frame (ESF)).

5.2.2 Signalling link and ISUP circuit parameters

These parameters set the values that will be used when signalling links or ISUP circuits are added. If specified only once they apply to all the signalling links or ISUP circuits. They may be repeated in order to use different values for different entities. If repeated, the value used is the last one before the **-cSLC** or **-cCIC** parameter that adds the signalling link or circuits.

-cOPC*nnnnn* Where *nnnnn* is the user defined unique Originating Point Code used to represent the local point code. This must coincide with the point code (`LocalPC`) of an SS7 signalling point `[SP]` section in the stack configuration file, as described in section 5.1.
For ANSI networks the point code can be specified in 8-8-8 format.
If the local pointcode doesn't currently exist the signalling links or ISUP circuits will be added if the local pointcode is later created by an `ss7maint start` request.

-cDPC*nnnnn* Where *nnnnn* is the user defined unique Destination Point Code for a remote point code. The network component may be a Signalling Transfer Point or a Signalling End Point.
nnnnn should coincide with the point code of an `[STP]` (`adjacentPC`), or of a `[DESTINATION]` (`remotePC`) in the stack configuration file as described in section 5.1.

-cSIGMTP3 This requests that an MTP3 `[DESTINATION]` be created for the ISUP circuits. This is the default if M3UA isn't configured.

-cTS*nn* Where *nn* is the number of the timeslot that will be used for the signalling link

on the line connected to the trunk.
If not specified, then timeslot 16 is used.
The timeslot must be specified if more than one signalling link is created.

-cTS*nn-nn*[,*nn-nn*]
Where *nn-nn* is an inclusive range of timeslots for a high speed signalling link. Specifying **-cTS1-15,17-31** will use 30 timeslots excluding timeslot 16.

-cH100STR*nn* Where nn is an H.100 stream number between 0 and 31 specifies the H.100 stream number to use to retime receive data on high speed links on PXv3 cards. If not specified the external trunk/port number is used.

-cPCR This instructs the firmware MTP2 to use PCR (preventive cycle retransmission) instead of the basic error correction method.

-cSPEED*nn*k Where *nn* is a multiple of 8 less than, or equal to, 64. This sets the speed of the MTP2 signalling links (i.e.: defines the number of bits of the TDM timeslot used for signalling). The unused bits in the TDM timeslot are set to ones.
If not specified the default is 64k unless **-cAMI** is specified when it is 56k (to avoid 'zero code suppression').

-cNOSW This stops the ss7 driver setting the TDM switch matrix connections between MTP2 and the physical port. It also stops the timeslots used for signalling being removed from the ISUP `cct_map`.
This allows a signalling link to be connected to a non-standard place - e.g.: the H.100 bus or looped back onto another MTP2 link.

-cMTP2_*x=y* This parameter allows any of the MTP2 protocol parameters defined in section 5.1.8 to be specified as a firmware download parameter.
In particular `-cMTP2_timeslot=n` makes the Prosody card use a specific timeslot to the TDM switch. The timeslot number is incremented for subsequent signalling links.

See A.1.6 for an example that uses the `-cNOSW` and `-cMTP2_timeslot` parameters.

5.2.3 Adding signalling links

Multiple signalling links between any pair of pointcodes can be added to single trunk.

Each `-cSLC` parameter adds signalling links to an adjacent group of timeslots using the parameters set previously (see section 5.2.2). In order to add more than one signalling link to a trunk the timeslot must be specified (with `-cTSnn`) prior to each **-cSLC** parameter.

-cSLC*nn*[-*nn*] Where *nn* is the number (between 0 and 15) of the MTP3 Signalling Link Code for the signalling link.

Sixteen SS7 signalling links are allowed between each pair of pointcodes (an SS7 protocol limit).

To simplify the configuration of multiple signalling links on adjacent timeslots, a list of ranges of Signalling Link Codes can be specified. For example:

```
-cTS14 -cSLC1-3
```

has the same effect as:

```
-cTS14 -cSLC1 -cTS15 -cSLC2 -cTS16 -cSCL3
```

The default values of the MTP2 timers depend on whether a signalling link is being added to an E1 or T1 trunk.

If the remote pointcode (specified by **-cDPC*nnnn***) doesn't have an associated MTP3 `[DESTINATION]` or `[STP]` section, then a `[DESTINATION]` section will be created.

5.2.4 Adding ISUP circuits

Multiple groups of ISUP circuits can be added to a single trunk, although it is usual to only add a single group.

Each `-cCIC` parameter add circuits between a pair of pointcodes, not all the timeslots need be

assigned CIC numbers, and not all of those CIC numbers need be usable by ISUP.

-cCIC*nnnn*[,*cic_map*[,*circuit_map*]] Where *nnnn* is the number of the CIC assigned to the first bearer timeslot, with subsequent CICs allocated sequentially to higher timeslots.

The optional `cic_map` defines which timeslots have CIC numbers assigned. It may be specified as a list of timeslot ranges separated by `:` (eg: `1-15:17-31`) or as a hexadecimal number where bit zero (value 2^0) corresponds to timeslot zero etc (eg: fffefffe). This allows timeslots used for signalling to be skipped.

If not specified, a `cic_map` of `1-31` is used for E1 and `0-23` for T1.

The optional `circuit_map` defines which timeslots are used for ISUP calls. Timeslots that are not also present in the `cic_map`, those used for signalling, and timeslot zero for E1 trunks are automatically removed from the `circuit_map`.

If not specified, the `circuit_map` defaults to the `cic_map`.

Note **If `-cCICnnnn` is not specified, ISUP will not be available for call setup on the timeslots of that port.**

Examples

Specifying `-cCIC1,1-31,1-15:17-31` (or having signalling on timeslot 16 and specifying `-cCIC1`) puts CICs 1-15 on timeslots 1-15 and CICs 17-31 on timeslots 17-31. In this case CIC 16 isn't assigned to ISUP.

Specifying `-cCIC1,fffefffe` (or `-cCIC1-15:17-31`) puts CICs 1-15 on timeslots 1-15 and CICs 16-30 on timeslots 17-31. This gives a contiguous range of CICs with no CIC number assigned to timeslot 16 so not used by ISUP.

The following example shows an E1 trunk (port zero) that is using timeslots one and sixteen as signalling channels between point codes OPC99 and DPC29, and the remainder of the timeslots (excluding timeslot zero) as ISUP bearers between OPC99 and DPC30:

```
fwdspldr 1234567 0 ss7.pmx -cOPC99 -cDPC29 -cTS16 -cSLC0 -cTS1 -cSLC1
   -cDPC30 -cCIC1,fffffffe,fffefffc -cNCRC
```

Note **Further examples can be found in Appendix A:**

5.2.5 Removing signalling links and ISUP circuits

It is possible to remove signalling links and ISUP circuits from some timeslots on a trunk without affecting other timeslots (including active ISUP calls). These parameters would normally be used during dynamic reconfiguration (see section 6.2.2)

The signalling links/ISUP circuits to remove are identified by the timeslot specified by the last **-cTS** parameter for the trunk. In this case a range of timeslots can be specified as **-cTS*lo-hi***.

-c-SLC Remove any signalling links using the specified timeslots.

-c-CIC Remove any ISUP circuits using the specified timeslots.
This will remove all of the circuits that were added with a single **-cCIC** parameter if any of the circuits use one of the specified timeslots.

5.3 Configuring parameters for SCTP

In systems running Linux, SCTP is part of the operating systems kernel. To configure parameters for SCTP such as the Maximum Transmission Unit (MTU) please refer to the Linux kernel documentation.

For Windows, the SCTP implementation is from the FreeBSD Foundation (Appendix C:) and is linked to the Aculab SS7 driver. It makes use of the Windows registry for configuration data. Editing of the Windows registry is required to change the MTU for a network interface. The MTU can be set to a number in the range 576 to 1500, default 1500. Values outside of the range will be set to the nearest in-range value.

To change the MTU:

1) Click Start, click Run, type `regedit`, and then click OK.
2) Locate the following key in the registry:
   `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\<ID for network interface>`
3) On the Edit menu, point to New, and then click DWORD Value.
4) Type **`Acu_sctp_mtu`**, and then press ENTER.
5) On the Edit menu, click Modify.
6) Ensure the base is correct (decimal) and in the Value data box, enter the value of the MTU size (576 to 1500), and then click OK.
7) Restart the SS7 driver.

No other values can be changed.

Note **The list of local IP addresses is taken from the registry keys above. It may be necessary to delete stale entries for unused network interface by hand.**

6 Starting and Restarting Aculab SS7

Before you can load an SS7 configuration, the SS7 driver must be installed and loaded and the daemon/server process running (described in section 10).

6.1 Loading a complete configuration

The SS7 driver configuration and ss7 firmware downloads must both be done before applications can use ss7. These requests can be done in either order. If an existing configuration is being replaced it may be desirable to remove the old configuration completely before loading the new configuration.

Section 6.2 describes how an existing, working, configuration can be modified.

6.1.1 Loading the driver configuration

After creating your Aculab SS7 stack configuration file as described in section 5.1, before you can start your TCAP and/or ISUP application, you must perform the following steps:

Start the SS7 protocol stack by typing `ss7maint start <options>` at a command line prompt. Options:

`-f` *`filename`*	Take stack configuration from *`filename`* (default `ss7.cfg`).
`-v`	Verbose parameter listing to `stdout`, **`-vv`** do not truncate long lines.
`-o` *`opc`*	Only configure the specified pointcode, all other pointcodes are unaffected.

If *`filename`* cannot be opened, and doesn't contain a `/` or `\` then `$ACULAB_ROOT/cfg/`*`filename`* will be tried before an error is reported.

Refer to section 6.2.1.1 for a full list of the options to `ss7maint start`.

6.1.2 Downloading SS7 firmwares

Firmware files that provide the layer 1 and layer 2 interfaces must be downloaded to the cards and trunks you are going to use. The downloaded parameters define any MTP2 signalling links or ISUP bearer channels the card is to provide.

The ss7 firmware itself may be downloaded using the `fwdspldr` command line utility (as in the examples here), the `call_restart()` API function or by the Aculab Configuration Tool (ACT). Refer to the ***Call, switch and speech telephony software installation guide*** and the ***Call control API guide*** for further general information.

The ss7 firmware file (eg `Firmware/ss7.pmx`) only contains the layer 1 support. The MTP2 support for Prosody 1U enterprise cards is in `Firmware/pmx/pmx_ss7.eld` and that for v3 cards in `Firmware/pmx_v3/ss7_mtp2_pmxrev3`. These files are automatically copied to card and executed during firmware download.

To download using the `fwdspldr` utility use the command:

```
fwdspldr board_serial trunk_number ss7.pmx [parameters]
```

See section 5.2 for a list of the ss7 firmware parameters and some simple examples. Appendix A (page 109) contains further examples.

The MTP2 data on the signalling timeslots is routed through the card's TDM switch matrix. These switch connections are established automatically during link activation. Resetting the switch matrix will not normally remove them, but if they are changed the signalling links will fail and then reconnect.

6.1.3 Set the trunk clocking parameters

The source of the TDM clocks for the ports must be set correctly as described in the ***Call, switch and speech telephony software installation guide***. If you are using the ACT, this step may be performed automatically.

This may be achieved using the `swcmd` utility as defined in the ***Software for Aculab digital network access cards – V6 switch API guide*** Appendix D.

Note **If there is any TDM clock slip it is extremely unlikely that MTP2 signalling links will come into service.**

6.2 Dynamic reconfiguration

There are times when it becomes necessary to change the parameters on a running system. Maybe the network topology changes slightly (e.g.: additional ISUP circuits added to a possibly new remote pointcode), or, perhaps, one of the parameters is found to have the wrong value (or needs its value changed to aid testing).

While the best way to change the configuration is to edit the configuration file and/or firmware download parameters and perform a full restart of the ss7 protocol start as described in section 6.1 there are times when a full restart has undesirable effects (like disconnecting all existing calls).

Note **Do not attempt to repeatedly change the configuration in order to affect API calls.**

A firmware can be safely downloaded again without restarting the driver, and without affecting any other trunks, see section 6.1.2. Additionally MTP2 signalling links and groups of ISUP circuits can be added/removed from a trunk without affecting other timeslots, see section 6.2.2.

If your configuration has multiple local pointcodes, then the driver configuration for one local pointcode can be removed and replaced without affecting the other local pointcodes, see section 6.1.1 and, in particular, the `-o` parameter.

With certain constraints the parameters for a local pointcode can also be dynamically changed, see section 6.2.1.

The library configuration of TCAP applications (and SCCP applications) can also be changed from the driver, see section 6.2.3.

Note **If you dynamically change a parameter, make sure you also change the configuration loaded after system startup, otherwise the change will not persist.**

6.2.1 Driver reconfiguration

The driver configuration parameters are typically held as simple values (mainly numeric) in tables indexed by the 'mandatory parameter'. Changes made to this data (including adding new entries) will have an immediate effect.

Changes can be made either by requesting part of the configuration file to be reapplied (see section 6.2.1.1), or by explicitly requesting a single parameter be changed (see section 6.2.1.2).

There are a few parameters that cannot be changed, or where the change is likely to be ineffectual or have unwanted side effects:

- Do not attempt to change the major variant (e.g.: from ITU to ANSI).
- Any changes to the variant are likely to return other parameters to their defaults.
- The `listen` and `connect` parameters in the mtp3 dual section are commands, subsequent requests for the same `slc` are silently ignored.
- When ISUP codec extensions and SCCP global title tables are applied, the protocol entity will take copies of some values and references to others. This means that some changes are disallowed (e.g.: changing the ISUP mandatory parameters), and others may, or may not have an immediate effect. If you change an `[ISUPCODEC]` or `[GT_TABLE]` section you should reapply the changes to the relevant server section.

The current driver configuration along with the valid parameter names and values can be obtained from the driver, see section 8.4.

6.2.1.1 Reapplying part of the configuration file

When the ss7 stack configuration file is loaded, it normally removes all the existing protocol entities and then recreates them with the new configuration. Instead of doing this, it is possible to overlay all, or part, of the configuration onto the existing system.

To reapply the configuration use `ss7maint start -r`.

Note **Only use this feature after minor changes to the configuration file.**

The full list of options to `ss7maint start` is:

`-r`	Do not remove the old protocol entities. This is the default if **`-p`** specifies one of the subsections of an [SP].
`-v`	Verbose, lists parameter strings to `stdout`, **`-vv`** do not truncate long lines.
`-f` *`filename`*	Take stack configuration from *`filename`* (default `ss7.cfg`). If *`filename`* cannot be opened, and doesn't contain a `/` or `\` then `$ACULAB_ROOT/cfg/`*`filename`* will be tried before an error is reported. A *`filename`* of `/dev/null` is treated as an empty file (even on windows) and causes all pointcode server information to be deleted.
`-o` *`opc`*	Only configure the specified pointcode, all other pointcodes are unaffected.
`-p` *`section`* `[,`*`section`*`]…`	Only configure the requested sections. Valid sections are those than appear at the outer level of the configuration file or immediately inside an `[SP]` section, i.e.: `sp`, `mtp2`, `isupcodec`, `gt_table`, `mtp3`, `m3ua`, `isup`, `sccp` and `tcap`. Multiple sections may be specified by repeating the parameter or specifying a comma separated list of sections.

6.2.1.2 Changing a single parameter

The configuration parameters within a local point code can be changed without affecting operation using the `ss7maint configure` command which has the following options:

`-o` ***`nnnnn`***	Specifies the local point code to which the configuration change applies.
`-p isup` `-p` **`m3ua`** **`-p mtp2`** `-p` **`mtp3`** **`-p sccp -p tcap`**	Selects the part of the configuration to update.
**`-t mtp2:`*`name`*	Change text configuration records (e.g.: MTP2 configuration).
`-a`	Allows new configuration records to be created.
`-d` ***`nnnnn`***	Specifies that the configuration for the destination with point code ***`nnnnn`*** is to be changed. Equivalent to **`destination:`***`nnnnn`* (**`concerned:`***`nnnnn`* for SCCP).
`-S` ***`slc`***	Specifies the signalling link for `-p mtp2`. These changes will be lost when MTP3 restarts the signalling link.
`-v`	verbose parameter listing to `stdout`.
`subsection:name`	Specifies that parameters in the named subsection should be changed. This may be repeated, but must come before any `parameter=value` parameters.
`parameter=value`	Specifies the actual parameter to change, and its new value. This may be repeated in order to change multiple items in a single request.

Either **`-p`** or **`-t`** must be specified.

For example, `ss7maint configure -p isup -o2020 -d7070 sus,rel=y` is equivalent to having `sus,rel=y` in the `[ISUP]` section of the stack config file.

The parameter names, current values and their valid ranges can also be displayed, see 8.4.

Note **The subsections and parameters are as displayed by `ss7maint start -vv`.**

Note **The m2pa and m3ua `host=name,ip_address` parameters need to be specified as `host:name ipaddresses=ip_address`.**

6.2.2 Adding and removing signalling links and ISUP circuits

It is possible to use `fwdspldr -reconfig` to change the signalling links and ISUP circuits on a trunk without affecting other timeslots and active ISUP calls.

To add signalling links or ISUP circuits just specify the `-cOPC`, `-cDPC`, `-cTS`, `-cSLC` and `-cCIC` parameters as for a normal firmware download.

To remove signalling links specify a single timeslot, or range of timeslots with `-cTS`*`lo-hi`*, then `-c-SLC` to remove signalling links from the selected timeslots.

To remove ISUP circuits select the timeslot(s) with `-cTS`*`lo`*`[-`*`hi`*`]` then use `-c-CIC` to remove the circuits from the specified timeslots. All of the circuits defined by a single `-cCIC` parameter are removed if the timeslot for any of them is specified.

The parameters are processed individually in sequence; `-cOPC`, `-cDPC`, `-cTS` (etc) change the saved values that are used later. These need not be specified if the saved values are correct.

For example:

```
fwdspldr -reconfig 1234567 0 -cTS17 -cSLC1-15
```

adds 15 more signalling links on timeslots 17 to 31 between the last pointcodes specified (typically those used for slc 0 on timeslot 16).

Note **It isn't possible to change the layer 1 parameters.**

6.2.3 TCAP application reconfiguration

Changes to the TCAP application configuration can be made using `ss7maint tcapconfig`. These requests are first passed to the driver and hence to the application. So can only affect applications that have successfully connected to the driver on the system where the command is run.

`ss7maint tcapconfig` has the following options:

`-o` ***`nnnnn`***	Specifies the local point code to which the configuration change applies.
`-h` ***`hostname`***	Specifies the hostname (or numeric IP address) of the system where the application is running.
`-s` ***`ssn`***	Specifies the SCCP subsystem number the application is using.
`-t` ***`transaction_id`***	Specifies the TCAP transaction id of the application (in hexadecimal). If a full 32bit number is given then the change is applied to the configuration of that specific transaction. If only 12 bits are given (or if the least significant 20 bits are all zero) then the transaction_id is used to select the application using that block of transaction id values and affects the library 'ssap' configuration.
`parameter=`*`value`	Specifies the actual parameter to change, and its new value. This may be repeated in order to change multiple items in a single request

The `-o`, `-h`, `-s` and `-t` parameters act as filters. If not specified then all TCAP applications are changed.

Note **If a hostname is specified only the first IP address is used.**

7 SS7 management

The SS7 driver can be managed from the command line using the ss7maint command described here, and also through the maintenance API, see ***Aculab SS7 Maintenance API guide***.

7.1 Signalling link management

Link activation and link inhibition using `SS7maint` is detailed below. For link status display using `SS7maint`, please refer to section 8.2.

7.1.1 Link activation

By default, Aculab SS7 will attempt to keep all configured signalling links activated so that they are available for MTP3 and user part traffic. You can manually override this behaviour using `ss7maint linkactivate`, which allows you to enable or disable activation of individual links or linksets using the options `-y`, `-n`, `-o`, `-d` and `-S`, as described below:

`-y`	Active the selected links
`-n`	Deactive the selected links
`-o` ***`nnnnn`***	Only apply the command to links with the specified originating (local) point code.
`-d` ***`nnnnn`***	Only apply the command to links with the specified destination (remote) point code.
`-S` ***`nn`***	Only apply the command to links with the specified signalling link code (SLC).

Example 1:
To deactivate all links in the linkset between `1234` and `5678` manually:

```
ss7maint linkactivate -n -o1234 -d5678
```

Example 2:
To activate a single link (SLC1) in the linkset between `1234` and `5678` manually:

```
ss7maint linkactivate -y -o1234 -d5678 -S1
```

Links can also be activated and inactivated using the maintenance API library or `ss7maint apiaction mtp [in]activate`.

7.1.2 Link inhibition

Link inhibition is a procedure used to make a link unavailable to user part traffic without deactivating it. Signalling links may be either locally inhibited (using ss7maint) or remotely inhibited by management action at the adjacent MTP, or both. The MTP3 protocol includes procedures to ensure that link inhibition is not allowed if it would cause any destinations to become inaccessible. It also automatically removes inhibition if it would restore accessibility to a destination that becomes inaccessible for some other reason.

You can use `ss7maint` to request or remove local link inhibition, as defined in ITU-T Q.704. Before allowing the request to proceed, the local routing tables will be examined to see whether any destinations would become inaccessible. If not, then link inhibition will be attempted and a LIN (Link Inhibit) message will be sent. The adjacent MTP3 will then perform a similar check, and may then allow or deny the inhibition.

The `ss7maint` interface will not indicate whether the inhibition was allowed or denied. The current inhibition status of a link can be displayed as described in section 8.2.

To manually request or remove local inhibition, use `ss7maint linkinhibit` with the options `-y`, `-n`, `-o`, `-d` and `-S`, as described below:

`-y`	Inhibit the selected links.
`-n`	Uninhibit the selected links.

-o *nnnnn*	Only apply the command to links with the specified originating (local) point code.
-d *nnnnn*	Only apply the command to links with the specified destination (remote) point code.
-S *nn*	Only apply the command to links with the specified signalling link code (SLC).

Example 1:

To inhibit all links in the linkset between `1234` and `5678` manually:

```
ss7maint linkinhibit -y -o1234 -d5678
```

Example 2:

To uninhibit a single link (SLC1) in the linkset between `1234` and `5678` manually:

```
ss7maint linkinhibit -n -o1234 -d5678 -S1
```

Links can also be inhibited and uninhibited using the maintenance API library or `ss7maint apiaction mtp [un]inhibit`.

7.2 ISUP circuit management

ISUP circuits can be reset or management blocked/unblocked through the maintenance API, or from the command line using:

```
ss7maint apiaction [search_opts] isup reset|block|unblock|hwblock|hwunblock
```

The search options `[search_opts]` can be any of the following:

-s any	Select all systems (the default).
-s local	Select only the local machine.
-s dual	Select only the other half of a dual MTP3 system.
-s distrib	Select only distributed systems.
-o *nnnnn*	Only apply the command to circuits with the specified local point code.
-d *nnnnn*	Only apply the command to circuits with the specified remote point code.
-c *cic_low* *[-cic_high]*	Only apply the command to circuits with the specified CIC or within the specified CIC range.

Note **If the command is issued from a distributed ISUP system, then only local circuits can be affected.**

Note **This functionality is implemented by calling functions in the maintenance API library. It will fail if the library cannot be opened.**

7.3 M3UA connection management

By default the M3UA connections are established and the routing keys registered and activated when the SS7 protocol stack starts. This can be disabled from the configuration file and then enabled on demand either using the `ss7maint` commands below, or through the maintenance API. For use from a program or script please refer to the ***Aculab SS7 Maintenance API Guide***.

The only action that can be requested on a server (or an IPSP server) connection is disconnect.

`ss7maint apiaction [search_opts] m3ua connect`
Attempt to establish the SCTP connection for the selected M3UA clients (and IPSP clients) that are not connected. If `auto_register` and `auto_activate` are enabled the routing keys will be registered and activated once the connection is established.

`ss7maint apiaction [search_opts] m3ua disconnect`
Disconnect the selected SCTP connections. Client (and IPSP client) connections will be inactivated and deregistered first.

`ss7maint apiaction [search_opts] m3ua register`
Registers the selected routing keys. If `auto_register` is disabled and `auto_activate` is enabled the routing keys will be activated when the registration completes.

`ss7maint apiaction [search_opts] m3ua deregister`
Deregisters the selected routing keys. If the routing keys are active, they will be made inactive first. If routing key management isn't configured this has no effect.

`ss7maint apiaction [search_opts] m3ua activate`
Activates the selected routing keys. If the routing keys are not registered, they will be registered first. It is not necessary to wait for a registration request to complete before requesting activation.

`ss7maint apiaction [search_opts] m3ua inactivate`
Inactivate the selected routing keys.

The search options `[search_opts]` can be any of the following:

`-s any`	Select all systems.
`-s local`	Select only the local machine.
`-s dual`	Select only the other half of a dual MTP3 system.
`-o` ***`nnnnn`***	Only select entities with the specified local point code.
`-m` ***`name`***	Only select connections for the named client/server section.
`-p` ***`hostname`***	Only select entities with the specified remote host (IPv4 address in 'dotted quad' notation or a hostname that can be converted to one).
`-x` ***`route_ctx`***	Only select entities with the specified routing context.

8 Tracing and troubleshooting

The `ss7maint` tool provides facilities for setting system trace parameters, capturing trace information to a disk file and post-analysis of trace files to obtain protocol decodes. It also provides link and protocol status information that can be useful in diagnosing problems. This section of the document explains how to use these features.

In addition to the features described in this section of the document, Aculab SS7 may occasionally write diagnostic information to the native logging mechanism for the Operating System. In the case of Windows, this would be the Event Manager, which can be read from Windows control panel. When problems are encountered, check to see if any error messages have been generated.

8.1 SS7 driver trace

Aculab SS7 operates with system tracing permanently enabled, you do not need to take any pre-emptive action to enable tracing prior to a problem occurring. The trace is held in system memory in a history buffer, which is only copied to a disk file when requested by the user. The number of trace entries held in the buffer may be set using `ss7maint` as described in section 8.1.3.

Collecting in the history buffer has a smaller impact on system performance than writing continuously to a disk file. In typical use, the trace collected during some time critical protocol test or error condition should be written do disk as soon after the failure as reasonably possible.

Note **If something has gone wrong, you should write the trace history to file before attempting further diagnosis or recovery. This maximises the chances of the trace containing the required information.**

It is also possible, using `ss7maint`, to trace continuously to a disk file. This mode of operation avoids using large system memory buffers but, under conditions of heavy load, it can have an adverse effect on system performance. By requesting a continuous trace, but also limiting the number of trace entries read, the trace can be written to a series of disk files. See section 8.1.1 for further details.

If Aculab SS7 detects an irrecoverable internal error or inconsistency, the driver may generate additional trace information, even though your applications (or other system components) may not detect any errors. This trace information can be collected using `ss7maint`, but you must do so before unloading SS7 or restarting your system. Therefore anytime you think the system is malfunctioning, you should collect the trace information as described in section 8.1.1 before rebooting your system to restore normal operation. The trace file can then be submitted to Aculab support.

8.1.1 Trace collection

To read the trace history buffer, use `ss7maint trace`.

Usually, you would do this soon after a problem or event of some interest has occurred, so that you can decode it (see 8.1.2) and analyse the file yourself. Alternatively if you suspect the Aculab product is malfunctioning, you can submit the file to Aculab support.

`ss7maint trace` will normally terminate automatically when it has processed the entire contents of the history buffer. If trace is being generated more quickly than it can be read then `ss7maint trace` may need to be terminated by breaking in with `Ctrl+C`.

You can specify any of the options described below.

`-c`	Continuously read the trace, waiting for further events to be logged.
`-n` *`count`*	Stop after processing *`count`* trace entries, default infinite. This can be used in a script with **`-c`** to trace to files and perform log file rotation.
`-w` *`logfile`*	Write decoded trace to the specified file, default `stdout`

The first line of the trace is always the driver version string.

Note **Reading trace is relatively processor-intensive and may adversely affect system performance**

Note **Once you have read the history buffer as described above, it is deleted. You will not be able to read the same trace information again. Therefore it is strongly recommended that the output from `ss7maint trace` always be written to a disk file.**

Note **The first line of the trace always contains the version number of the ss7 driver and daemon/ss7sxc. The command `ss7maint trace -n1` will just display this information and can be used to check the correct driver version is loaded.**

8.1.2 Decoding Aculab SS7 trace files

The trace files that are produced by the Aculab SS7 trace mechanism are primarily intended to be of use to Aculab support. Whilst they are human readable ASCII text files, the information they contain is not generally informative to most Aculab customers, and is liable to change between product releases. In order to provide trace information that is of use to customers, a second level of analysis is required which produces a decode of the MTP2, MTP3, ISUP, SCCP and TCAP protocol messages and parameters, and includes certain error messages.

Note **If you send a trace file to Aculab support, always send the raw trace file, not the decoded one.**

The protocol decode files that are produced are plain text and human readable, and are intended to be self-explanatory. However, to make use of them you will need to be familiar with the SS7 protocols and messages as described in the ITU Q.7xx series of recommendations. Please note however that Aculab may change the format of the decoded information as the SS7 product is further enhanced.

Message formats, and protocol decode, may be different depending upon which national protocol variant is in use. ss7maint will perform decode based on the protocol variant that was configured when the traffic was captured, or it can be forced using the `-v` option to decode according to an alternative variant. This may be useful if there is any suspicion that traffic for an unexpected variant is being encountered.

To obtain a protocol decode, you must first collect the trace information to a disk file using `ss7maint trace` as explained in section 8.1.1, and then use `ss7maint decode` to analyse that file and produce a new file containing the protocol decode.

You can specify any of the options described below.

`-r` *`logfile`*	Read raw trace from the specified file, default `stdin`.
`-w` *`trace`*	Write decoded trace to the specified file, default `stdout`.
`-p` *`level`*	Pass raw trace entries with trace level less than, or equal to level through to the output file. The default level is 4, which includes most trace entries of abnormal events.
`-o` *`nnnnn`*	Only decode trace entries for the specified local point code.
`-d` *`nnnnn`*	Only decode trace entries for the specified destination endpoint.
`-S` *`slc`*	Only decode trace entries for the specified slc.
`-n` *`ni`*	Only decode trace entries for the specified ni.
`-s` *`si`* **`-s`** *`-si`*	Only decode trace entries for the specified si, may be repeated. Do not decode trace entries for the specified si, may be repeated.
`-f`	Do not decode trace entries for FISUs (implied by **`-n`** and **`-s`**).
`-l`	Do not decode trace entries for LSSUs (implied by **`-n`** and **`-s`**).
`-V` *`variant`*	Decode for a particular variant e.g. ansi, china, itu or ukisup.
`-F flags`	Explicilty specify the trace format flags, useful for decoding non-standard trace files. See 8.1.5.
`source=level`	Only decode trace entries from `source` that have level less than or equal to

`level`. Specifying an invalid `source` or `level` will display the valid values.

Note **The decode can be restricted to the MTP2 line trace by specifying** `-p 0 all=0 tx_data=normal rx_data=normal`

Note **Only the first occurrence of MTP2 FISU and LSSU messages (sent repeatedly by MTP2) are traced, so they will only appear once in the protocol decode.**

Note **The `-o`, `-d` and `-s` options do text comparisons of the trace text.**

8.1.3 Setting trace parameters

You can set the trace filter parameters using `ss7maint filtertrace` with any combination of the options described below:

`-a`	Includes some additional trace points such as those associated with MTP3 changeover. Equivalent to specifying **`all=8`**.
`-v`	Enables additional traces that include data buffers. This will generate more information for some of the events. Equivalent to specifying **`all=12`**.
`-d`	Requests the trace history be deleted. This can be useful if you are about to perform some test procedure and want to be sure that the trace buffer contains no information from earlier events.
`-h` ***`size`***	Allows you to reset the size of the history buffer. Setting a larger history buffer can be useful when tracking down problems that occur at unpredictable times. The history buffer consumes system memory so, when using large history buffers, overall system performance may be affected if the amount of memory remaining is insufficient for the system and applications you are running. The default number of entries is currently 10000.
`-n`	Specify with `-d` or `-h` to stop the trace level being reset.
`source=level`	Only trace events from source which have trace level less than or equal to level. The default is all=5, and the maximum level 14. The valid sources and level are displayed in the help text output if an invalid source is specified.

Note **You should normally only change the trace levels if instructed to do so by Aculab support.**

8.1.4 Changing the MTP2 protocol trace

The MTP2 protocol trace is generated by the MTP2 firmware code. Normally all received and transmitted message units and the first of each LSSU and FISU are traced. Under exceptional circumstances this may need to be changed.

The trace level can be set using the `trace_xx` options of the `MTP2` section of the stack configuration file, firmware download parameters, or changed dynamically using the `ss7maint prottrace` command using the following options:

`-o nnnnn` Only change tracing for links with the specified originating (local) point code.

`-d nnnnn` Only change tracing for links with the specified destination (remote) point code.

`-S nn` Only change tracing for links with the specified signalling link code, (SLC).

`-s+/-` Enable or disable the firmware function call trace.

`-t` ***`flags`***
`-r` ***`flags`***
Change the transmit (**`-t`**), or receive (**`-r`**) protocol trace. The valid flags are:

- **`+`** Set trace flags.
- **`-`** Clear trace flags.
- **`6`** Trace to call driver trace as well as ss7 trace.
- **`f`** Trace FISUs.
- **`l`** Trace LSSUs.
- **`m`** Trace MSUs.
- **`mv`** Trace out of sequence MSUs on PCR links.
- **`t`** Change transmit flags as well as receive flags.
- **`r`** Change receive flags as well as transmit flags.

If + (or –) is the last flag, then all the flags are set (cleared).

Note **Changes made using `ss7maint prottrace`.are discarded when the link fails.**

Examples:

```
ss7maint prottrace –rt-f
```

Will disable tracing of FISUs on all signalling links.

```
ss7maint prottrace –S0 –t+
```

Will enable tracing of all transmit message units on signalling links with `slc` 0.

```
ss7maint prottrace –rt+mv
```

Will enable tracing of retransmitted MSUs on a PCR link, this would be needed if the operation of PCR itself needs to be verified.

8.1.5 SS7 Trace file format

A typical ss7 trace entry looks like:

```
[30/1/13 16:34:02.552]:520e MTP2 4-2 slc 1 TX card 222872(2) ts 5 21m15.705
  81 82 13 01 02 00 01 10 11 c0 41 63 75 6c 61 62    ..........Aculab
  5f 53 53 37 00 02                                  _SS7..
```

The trace entry starts `[date]:flags text`. The `flags` controls how `ss7maint decode` processes the trace entry.

Flags

`....xx`	Trace source, identifies MTP3, ISUP etc, see `$ACULAB_ROOT/include/ss7_lib_decode.h`. The tcap library trace uses identifiers `0xc0` through `0xff`.
`...x..`	Data buffer format: 0 => No buffer or unknown format. 1 => SIGTRAN format (used for all TCP buffers). 2 => MTP2 data. 3 => MTP3 data. 4 => ISUP data. 5 => SCCP data. 6 => TCAP data (ASN.1 BER encoded). 7 => Other ASN.1 BER encoded data. 8 => TCAP/SCCP address buffer (as defined by the tcap API library).
`..x...`	Trace level. Lower values are more serious.
`.x....`	Major variant for userpart.
`x.....`	Width of pointcodes. 0 => 14 (ITU), 2 => 24 (ANSI/CHINA).

The driver generated trace will usually specify the correct data format.

If `-F flags` is specified then ss7maint decode will use the specified value instead of parsing the initial line (which can then have any format). This is useful for decoding hexdumps that have come from other sources. The lines of hex bytes must start with a few spaces, and may contain an offset terminated by a colon (`:`).

8.2 Status displays

The ss7maint program contains several subcommands that display system status and the contents of some internal data structures.

The output from these commands is intended to be understandable by the people able to make use of the information. In some cases this is only really the Aculab development group, although Aculab support may ask for the information in order to address specific problems.

Note **The information presented by these commands is plain text and human readable, but the format and exact wording of the text may change between different releases of the product.**

The following options are supported by all the status functions:

`-h` This causes the cursor to be positioned at top left ("home") of screen before displaying status. Use with `-i` to provide a continuously updated status display.

`-H` As `-h` but uses ANSI terminal escape sequences even on windows systems. This reduces flicker and system load but doesn't work in a window's command prompt window.

`-i` ***count*** This causes the display to be refreshed every ***count*** milliseconds. To terminate `ss7maint` in this mode, break in using `Ctrl+C`.

`-v` This may increase the amount of detail displayed.
Repeating the option, e.g. `-vvv`, may further increase the level of detail.

On windows systems `-h` usually causes `system("cls")` be executed. However, if `_isatty()` fails, then ANSI escapes are used instead (for recent CYGWIN). Specifying -hh forces the use of `system("cls")`.

8.2.1 ss7maint licence

This command displays SS7 licence information. Licence administration is normally done using the Aculab Configuration Tool (ACT). For more information, please refer to the documentation for this tool.

Note **It may take up to one minute after a licence has been installed before it is registered by the SS7.**

`-r` Restarts the licence manager.

`-p` Prints a summary of the currently installed SS7 licences.

`-c` Changes the TCP/IP port number used by the ACT.

`-l` Prints the current TCP/IP port number configured for use by the ACT.

`-t` Changes the licence manager trace level. You should normally only modify this value if instructed to do so by Aculab support.

Note **It is normal for all the licences to be shown as 'used'.**

Note **If the licence manager cannot be accessed, the feature names will not be displayed. This might be because the licence manager shared library cannot be opened.**

8.2.2 ss7maint tcapstatus

This command is only applicable when using distributed TCAP.

Displays information about each connected TCAP application including the point code, subsystem number and transaction identifier ranges. Information can also be obtained from the library code in the application itself.

-o ***nnnnn***	Only display information for the specified originating (local) point code.
-s ***ssn***	Only display information for the specified subsystem number.
-t ***tran_id***	Only display information for transactions with the specified transaction identifier (specified in hexadecimal). To select a specific application specify the most significant 3 digits only, for details of a specific transaction specify all 8 digits.
-a	Display information from the library about its connection to 'host a'.
-b	Display information from the library about its connection to 'host b'.
-d	Display information from the device driver (default).
-l	Display general information from the library.
-c	Display information about connections from TCAP applications.
-f	Display flow control counts.
-k	Display receive and transmit packet counts (default output).
-A	All information from driver, equivalent to **-cfk**.
-T	Display per-transaction information from the TCAP library. With **-v** display the SCCP address information.
-O	Display per-operation information from the TCAP library. Implies **-T**.

8.2.3 ss7maint sccpstatus

Displays information about SCCP. When used without any options, a summary of the remote MTP3 and SCCP states is displayed. A remote MTP3 can take one of the following states:

MTP3 State	Description
Paused	An MTP-PAUSE primitive has been received. The destination cannot be reached.
Resumed	An MTP-RESUME primitive has been received. The destination can be reached and SCCP messages can be sent to and received from the remote SCCP.

A remote SCCP can take one of the following states:

SCCP State	Description
Prohibited	An MTP-PAUSE primitive has been received. The destination cannot be reached.
Unavailable	An MTP-STATUS primitive has been received with cause unknown. The remote SCCP is unavailable.
Unequipped	An MTP-STATUS primitive has been received with cause unequipped remote user. The remote SCCP is unavailable.
Inaccessible	An MTP-STATUS primitive has been received with cause inaccessible remote user. The remote SCCP is unavailable.
Congested	An MTP-STATUS primitive has been received indicating congestion at the remote user. The remote SCCP is available but traffic to the remote SCCP is restricted.
Available	An MTP-RESUME primitive has been received. The destination can be reached and SCCP messages can be sent to and received from the remote SCCP.

You can display other more detailed information using the options `-o`, `-d`, `-g`, `-G`, `-l` and `-r` as described below:

`-o nnnnn` Only display information about the specified originating (local) point code.

`-d nnnnn` Only display information about the specified destination (remote) point code.

`-g` Display information about the global title tables, including loadshare information and packet counts.

`-G` Display information about current congestion levels.

`-l` Display information about local subsystems.

`-r` Include information about remote subsystems.

A remote subsystem can take one of the following states:

User State	Description
In-Service	The remote subsystem is available and traffic can be sent to and received from the remote subsystem.
Out-of-Service	The remote subsystem is unavailable. Either the remote MTP3 or SCCP is unavailable or an SCCP subsystem prohibited (SSP) message has been received for this subsystem or the uos_on_resume option has been set.

8.2.4 ss7maint isupstatus

Displays information about ISUP.

-s Show status of ISUP connections. With **-v** include idle connections. With -vv show which systems the CIC is registered with, 1 => local, 2 => dual, 10 => distrib 'a' and 20 => distrib 'b'.

-p Show status of ISUP connections with additional information about the ISUP protocol state engines. With **-v** include idle connections.

-t Show status of ISUP connections in a tabular form with one line per trunk and one character per timeslot. See table below.

-m Show status of destinations and remote ISUPs as reported by the MTP3. See table below.

-o *nnnnn* Only display information about the specified originating (local) point code.

-d *nnnnn* Only display information about the specified destination (remote) point code.

-c *min_cic* [-*max_cic*] Only display information for cic numbers between ***min_cic*** and ***max_cic***.

ISUP connection states (for **-t** display).

Character	Circuit state
<space>	Idle circuit.
.	No CIC number assigned to timeslot.
-	CIC assigned, but not used by ISUP.
i	Incoming call in progress.
o	Outgoing call being made.
c	Stable connected call.
d	Call being disconnected.
f	Transitory state, call released but data areas not yet freed.
u	State reported by driver is not known (software version mismatch).
I, O, C, D, F or U	As above but connection block or reset also in progress.
x	No access to remote MTP3 system.
X	No access to local MTP3 system.
b	Circuit blocked.
B	Circuit block in progress.
r	Circuit being outward reset.
R	Circuit being inward reset (awaiting response from the local application).

Destination states (for **-m** display).

State	Description
Unavailable	In a system with a remote MTP3, the MTP3 is not available.
MTP paused	An MTP-PAUSE primitive has been received. The destination cannot be reached.
ISUP restarting	Optional. An MTP-RESUME primitive has been received. An ISUP UPT (User Part Test) message has been sent to the remote ISUP. Local circuits remain unavailable for new calls until a response is received from the remote ISUP.
MTP resumed	An MTP-RESUME primitive has been received. The destination can be reached and ISUP messages can be sent to and received from the remote ISUP.
UP inaccessible	An MTP-STATUS primitive has been received with cause inaccessible remote user. The remote ISUP is unavailable.
UP unequipped	An MTP-STATUS primitive has been received with cause unequipped remote user. The remote ISUP is unavailable.
UP unavailable	An MTP-STATUS primitive has been received with an unknown cause. The remote ISUP is unavailable.

8.2.5 ss7maint mtp3status

This displays the MTP3 routing information. This allows you to view destination accessibility, and identify which links are in use for each destination, and for different values of the Signalling Link Selector (SLS) field of transmitted messages.

When used without any options, a summary of destinations and their current accessibility status is displayed. A destination can take one of the following states:

Destination State	Description
Accessible	The destination can be reached and traffic can be sent to and received from the destination.
Restricted	The destination can be reached, but only using restricted routes. Transfer Restricted (TFR) messages have been received for all routes used to this destination.
Inaccessible	The destination cannot be reached.

You can display other more detailed information using the options -o, -d, -a, -l, -I, -A, -S and -r as described below:

-o *nnnnn* Only display information about the specified originating (local) point code.

-d *nnnnn* Only display information about the specified destination (remote) point code.

-a *nnnnn* Only display information about routes via the specified adjacent point code.

-l Display per-link routing details, including the SLS masks currently associated with each link and the total number of SLS values assigned to each link.

-I Only display information relating to inaccessible destinations.

-A Only display information relating to accessible or restricted destinations.

-S *min* [-*max*] Display per-SLS routing information, including the current status of traffic routing for each SLS, which may be either normal TX, one of several temporary buffers (e.g. during changeover/changeback), or discarding.
Information about SLS ranging from min to max is displayed.

-r Display route states. See table below.

A route can take one of the following states:

Route State	Description
Working	The route is used to carry traffic to the destination.
Standby	Higher priority Working or Working/Restricted routes exist to the destination.
Working/Restricted	The route is used to carry traffic to the destination but is restricted. A Transfer Restricted (TFR) message has been received for this route.
Standby/Restricted	Equal priority Working routes or higher priority Working or Working/Restricted routes exist to the destination.
Prohibited	A Transfer Prohibited (TFP) message has been received for this route.
Unavailable	The linkset over which the route is defined is unavailable. The route will recover to Working or Standby on linkset recovery.
Unavailable/Restricted	The linkset over which the route is defined is unavailable. The route will recover to Working/Restricted or Standby/Restricted on linkset recovery.
Unavailable/Prohibited	The linkset over which the route is defined is unavailable. The route will recover to Prohibited on linkset recovery.

8.2.6 ss7maint linkstatus

This displays the status of signalling links. This should assist you in diagnosing problems with link activation.

When used without any options, a summary of all configured signalling links is displayed. You can change the display information using the options described below:

-o ***nnnnn***	Only display information about links with the specified originating (local) point code.
-d ***nnnnn***	Only display information about links with the specified destination (remote) point code.
-s ***nn***	Only display information about links with the specified signalling link code (SLC).
-b	Display information about board serial number, port number, timeslot number and firmware version.
-2	Display detailed information about MTP2 status. With **-v** display packet counts, with **-vv** display timers and sequence numbers.
-3	Display detailed information about MTP3 status. The information provided identifies whether links are available for MTP3 traffic. If a link is not available for MTP3 traffic, the reason for unavailability is displayed.
-t	Format output as a table.
-C	Display the MTP2 statistics that are available through the maintenance API. The statistics are described in section B.1.4 of the Maintenance API guide.

Example:

```
ss7maint linkstatus -d1234 -2 -3 -v -h -i1000
```

Note **You may sometimes see the text "not configured" in the link status display, which means that the signalling link has not been added to, or has been removed from, the MTP3 configuration. This can occur if you have not started the protocol stack using a stack config file containing an [MTP3] section and an [SP] section with `LocalPC` corresponding to the link's OPC (See sections 5.1 and 5.2), or if you have two links with the same pointcodes and slc.**

8.2.7 ss7maint apistatus

This uses the maintenance API functions to retrieve information and display it in a human readable form. It is not intended that programs or scripts parse the output. Please refer to the ***Aculab SS7 Maintenance API Guide*** for this purpose.

`ss7maint apistatus [-C] [search_opts] server item`

The **`server`** and **`item`** determine which information is requested:

Server	Item	Returned Information
`mtp`	`destination`	MTP3 and MTP2 signalling link states, similar to `ss7maint mtp3status -d`.
`mtp`	`link`	MTP3 and MTP2 signalling link states, similar to `ss7maint linkstatus -2`.
`mtp`	`route`	MTP3 signalling route states, similar to `ss7maint mtp3status -r`.
`m3ua`	`connection`	M3UA connection routing key.
`m3ua`	`address`	M3UA routing key addresses (pointcode + si pairs).
`m3ua`	`con_stats`	M3UA connection (summed across all route keys).
`m3ua`	`route_key`	M3UA route key (summed across all connections).
`isup`	`cic`	ISUP circuits.
`tcap`	`application`	Connected TCAP (& SCCP) applications.

Options:

`-C` Display any non-zero statistics counts.

The search options [search_opts] select the displayed entities and can be any of the following:

`-s any` Select all systems (the default).
`-s local` Select only the local machine.
`-s dual` Select only the other part of a dual MTP3 system.
`-s distrib` Select only distributed systems (ISUP only).

`-o nnnnn` Only select entities with the specified local point code.

`-d nnnnn` Only select entities with the specified remote point code.

`-a nnnnn` Only select entities with the specified adjacent point code (MTP routes only).

`-S slc` Only select entities with the specified slc (MTP links only).

`-m name` Only select connections for the named client/server section (M3UA only).

`-p hostname` Only select entities with the specified remote host (IP address in numeric form or a hostname that can be converted to one) (M3UA only).

`-x route_ctx` Only select entities with the specified routing context (M3UA only).

`-c cic_low [-cic_high]` Only select entities with the specified circuit identification codes (ISUP only).

8.2.8 ss7maint apievent

This displays the SS7 event indications that are made available through the SS7 maintenance API. It is most useful for verifying that specific events are generated in response to external stimuli when writing applications that use the maintenance API.

The enabled events are selected using the `-e`, `-o` and `-s` options, and the output further restricted by the `-i` and `-r` options.

`-e` *`nnnnn`*	Defines a bitmask of events to enable/disable (use 0xnnn for hexadecimal input). Default: all events.
`-o` *`nnnnn`*	Enable/disable events for the originating (local) point code. Default: all local point codes.
`-s mtp` **`-s isup`** **`-s tcap`** **`-s m3ua`**	Enable/disable events from the specified server. Default all servers.
`-a`	Enable (add) events selected by the previous the `-e`, `-o` and `-s` options.
`-d`	Disable events selected by the previous the `-e`, `-o` and `-s` options.
`-i` *`nnnnn`* **`[-`*`nnnnn`*`]`**	Only display events for the specified range of event '`ame_item`' values (e.g. CICs).
`-r` *`nnnnn`* **`[-`*`nnnnn`*`]`**	Only display events for the specified range of event '`ame_remotepc`' values.
`-n` *`nnnnn`*	Exit after displaying the specified number of events.
`-t` *`nnnnn`*	Exit after waiting the specified number of milliseconds without finding an event.

The `-a` and `-d` options may be repeated (after respecifying the `-e`, `-o` and `-s` options) in order to get finer control over the received events.

If `-a` isn't specified, then events are enabled based on the last `-e`, `-o` and `-s` options.

Note **The standard options listed in section 8.2 don't apply.**

8.2.9 ss7maint ipstatus

This displays information about all the TCP and SCTP connections.

Options:

`-l`	Display information about listening endpoints, not active connections.

8.2.10 ss7maint osstatus

This displays information relating to buffer allocation and queues. The information is not intended to be of direct use to customers, but it may provide some useful diagnostic for Aculab support if problems are encountered. You should only use this feature if asked to do so by Aculab support.

Options:

`-b`	Include buffer details.
`-B`	Include buffer contents.
`-l first [-last]`	Only show specified line numbers.
`-d`	Data buffer pools (not used in SS7 6.15.0 and later).
`-k`	Kernel memory allocation.
`-m`	Message pools.
`-q`	Scheduler message queues.
`-s`	Schedulers.
`-t`	Threads.

Without the global option `-v` only 'interesting' pools and scheduler message queues are output.

8.3 Loadsharing traffic

Complex configurations need careful configuration to ensure that outbound traffic is shared as evenly as possible over the available links. The default parameters are usually fine for systems with a small number of signalling links, but large configurations (probably over 4 signalling links to a single destination) will need careful configuration.

In order to set the configuration parameters it is necessary to understand how the various components of the Aculab ss7 protocol stack loadshare traffic.

8.3.1 MTP3 loadshare

MTP3 uses the sls value provided by the userpart (ISUP or SCCP) to determine the signalling link to use to transmit a packet. All packets sent with the same sls are sent on the same signalling link (to ensure they don't get reordered). The number of the sls depends on the network variant, for ITU it is 16, and for ANSI 256.

The MTP3 loadshare tries to assign the same number of sls to each linkset (only if `balance_routes=n`) and then to each link. There is no dependency on which sls are actually being used.

Each signalling link has a set of preferred sls which will always be assigned to it. The preferred sls assignments assume that the number of configured routes is a power of 2, and that all the linksets have the same number of links which is also a power of two. The least significant bits of the sls select the route, and the next bits the link (the bits in the sls and slc match), any higher bits are ignored.

Removing a signalling link doesn't cause the preferred sls be reassigned, the linkset is just treated as if the removed link has failed. This stops large numbers of changeback requests being issued if all the links are removed one by one.

Any sls that aren't the preferred sls for any link, and those whose preferred link are unavailable, are assigned to links to make (as much as is possible) the number of sls assigned to each route (only if `balance_routes=n`) and link the same.

In an ITU network with a single linkset of 16 signalling links each link will take 1/16th of the traffic. If one link fails then the sls from the failed link will be assigned to one of the other links which will now carry twice as much traffic as any of the others.

If the original load was above 50% then the one link will now be overloaded.

This also means that it is pointless (from a traffic load point of view) to provision anything other than 1, 2, 4, 8 or 16 signalling links in a linkset, or, for ITU, to have more than 16 signalling links in all the routes to any destination

If a single link fails on an ANSI network then there are 16 sls values to reassign to 15 working signalling links - so the load is still reasonably shared.

ANSI specifies that the low 5 bits of the sls value be rotated right when a message is sent (See T1.111.5). This means that the least significant bit of the sls (used to select the linkset) changes at each STP so traffic tends to use all the linksets between STP (assuming paired STP).

Note **For ANSI networks the 256 sls are evenly distributed, no attempt is made to evenly distribute any subset of the sls (e.g. for interworking with exchanges that only support 5 bit sls, or assuming that bit 5 is likely to be constant).**

8.3.2 Load sharing of ISUP messages

ISUP has to ensure that messages for a given circuit aren't reordered by MTP3. This is done by generating the sls from the CIC number. However any of the following may lead to the low bits of the CIC number not having the required distribution:

- A distributed ISUP, dual MTP3 system uses a bit (default `net_select=3`) to select between the MTP3 systems.
- Mostly odd (or even) circuits are in use because `selection_method=method2` and most

of the calls are either incoming or outgoing.
- E1 timeslot 16 is assigned a CIC number but is not used by ISUP and the base CIC for every trunk is a multiple of 32. So one of the 16 sls values never appears.

The sls defaults to the low bits of the CIC number, this can be changed by configuring `sls_shift=n` to a non-zero value (the lowest *n* bits of the CIC number are ignored when generating the sls).

Configuring the following might give better loadsharing for a large dual configuration.
- Avoid using CIC bit 0.
- `net_select=1` so that CIC bit 1 selects between the two dual systems.
- `sls_shift=2` to remove the bits that have already been used.

8.3.3 TCAP and SCCP loadshare

Systems using TCAP and SCCP are much more likely (than ISUP systems) to have significant numbers of signalling links, and to be loading the links to much higher levels. This makes it more important to understand how things work.

If the TCAP application has specified `QOS_SEQ_CTRL` (SCCP class-1 requesting that messages not be reordered by the network) the least significant bit of the TCAP transaction id is used to select the MTP3 system (in a dual configuration), and the sls value passed to MTP3 is the TCAP transaction id shifted right 1.

If SCCP class-1 delivery is not requested (probably the common case) the TCAP library attempts to equalise the total length of the messages sent to each MTP3 host and SCCP uses a (per local pointcode) counter to allocate the next sls value to each transmitted message.

This means that MTP3 or M3UA should see the full range of sls values.

In a dual MTP3 configuration, with two linksets each having 16 signalling links, it should be possible to loadshare traffic to a single destination over all 32 signalling links.

SCCP global title translation can loadshare between two destinations. This attempts to equalise the number of messages sent to each destination. The assignment for class-1 messages is 'sticky' (period set by `tsls_hold=sss`).

A TCAP application can explicitly set the 'next hop' mtp3 pointcode for a message that is routed on global title. This bypasses the SCCP global title translation code and allows distribution between a larger number of SCCP STP.

It is also possible to configure SCCP to deliver messages to a local application, and for that application to retransmit them to a specific pointcode (with route on GT still set) and having optionally rewritten both global titles. This can be done using either the TCAP or SCCP API libraries - use the SCCP API if you don't need to inspect the TCAP data.

8.3.4 M3UA loadshare

M3UA can loadshare data between multiple SCTP (or TCP) connections. If there has been no traffic on an sls for an interval of `t_loadshare` (default 1 second) then the sls is assigned to the connection with the fewest active sls.

If a local TCAP application requests SCCP class-0 delivery then M3UA ignores the sls and loadshares in strict rotation. This lets traffic be assigned to a new connection without needing a `t_loadshare` gap in the traffic.

Only the low 4 bits of the sls are used, so, even on ANSI networks, it is impossible to loadshare between more than 16 connections. However, given that connections are likely to be added for resilience (not throughput) that is unlikely to be an issue.

8.4 Displaying driver configuration parameters

It is possible to use the `ss7maint configure` command to display the current driver configuration and the names and values of all valid parameters.

The configuration section is selected (as the section 6.2.1.2), but instead of specifying `parameter=value` specify one (and only one) of the following:

*****	Displays the names of all valid subsections and parameters. Parameters which cannot be set in the current variant aren't shown, neither are parameter aliases for other variants.
subsection*:	Displays a comma separated list of all the known subsection names. The first name displayed might be `<ss7_pointcode>` (or similar) to indicate that new subsections can be created.
parameter*=	Displays a comma separated list of the valid values for this parameter. The first value is the variant dependant default.
?	Display **parameter=value** for all parameters that have a non-default value.
??	Display **parameter=value** for all parameters.
???	As **??** but include parameter name aliases (e.g.: names for other variants).

Note **Be sure to quote the ? and * parameters to avoid filename expansion.**

8.4.1 Displaying the driver configuration

The unix shell script `$ACULAB_ROOT/ss7/scripts/ss7_dump_cfg.sh`, and the windows `%ACULAB_ROOT%\ss7\scripts\ss7_dump_cfg.bat` display the current driver configuration.

Options (common to both scripts):

`-c`	Include ISUP codec extensions.
`-g`	Include SCCP global title tables.
`-o local_pointcode`	Display data for specified local pointcode. May be repeated.
`-a`	All local pointcodes. Default if `-c`, `-g` and `-o` all absent.
`-v`	Include parameters that have default values and the ISUP message definitions.
`-p protocol`	Display `mtp3`, `m3ua`, `isup`, `tcap` or `sccp` server configuration. May be repeated. Default all servers.
`-s section:value`	Base of configuration tree to display.
`-x section`	Skip specified subsection. May be repeated. Default `-x msg -x parameter -x prm` (excludes ISUP message definitions). Specify `-x-` or `-v` to remove the default.

The output matches the way the data is stored by the driver; this is different from the `ss7.cfg` file format.

The ISUP message definitions are normally suppressed because they are very wordy. There are some examples of displaying the ISUP message definitions in section 5.1.4.4.

Note **Due to `cmd.exe`'s quoting and parameter splitting rules the `.sh` version is much more robust.**

8.5 Obtaining the entire status and trace

Sometimes it is useful to get the driver trace, and all the status requests output from a single command. The unix shell script `$ACULAB_ROOT/ss7/scripts/ss7_dump_all.sh`, and the windows `%ACULAB_ROOT%\ss7\scripts\ss7_dump_all.bat` obtain all the available information (driver trace, status commands, driver configuration and card memory dump).

The `ss7_dump_all.sh` script has the following options:

`-b` Exclude Prosody card memory dump.
`-c` Exclude driver configuration.
`-s` Include SDRAM from Prosody X v3 card (large and slow).
`-t` Exclude driver trace.

There are no options to the `ss7_dump_all.bat` script, it acts as if none of the above options are supplied.

Note **Remember to redirect the output of the script to a file.**

8.6 Diagnosing faults

The ss7 protocol is necessarily complicated and it can sometimes be difficult to determine what is going wrong. It is generally best to verify that the lower layers are working correctly before looking at the higher layers - there is little point changing ISUP parameters if the MTP2 signalling links are unstable.

The ss7 driver trace will contain details of any errors, most of them will be shown by filtering the trace file through `ss7maint decode all=4`, although not all the displayed lines are necessarily important errors.

Ways of identifying some of the common problems are given in the following sections.

8.6.1 MTP2 signalling links

An MTP2 signalling link requires a stable TDM link. If there are any layer 1 line errors or clock slips then the signalling links are very unlikely to come into service.

The easiest way to check the signalling links is to run:

```
ss7maint links -2 -3 -v -h -i200
```

If you have a lot of signalling links, use the `-o`, `-d` and `-S` options to select the failing link.

This should give output looking something like:

```
opc 2020, dpc 7070, slc 0, MTP3 available, MTP2 in service
    tx:  active, fisu  1883, lssu 25420 (SIN), msu  1022, retx   0, qlen 0
    rx:  active, fisu  1882, lssu 24024 (SIN), msu  1023, bad    3, qlen 0
    t :
```

If `ss7maint` reports `Object not found`, check that the correct ss7 firmware has been downloaded. If it reports `MTP2 information not available` then the TCP connection from the ss7 driver to the Prosody card can't be established.

What is most interesting is not really the displayed values, but the values that are changing.

If the link is in service (and idle) then the counts of transmit and receive fisu (values `1883` and `1882` above) will be changing very rapidly, the other values should hardly change at all.

If the link won't come into service then the transmit count of lssu should be changing quickly and the type (`SIN` above) alternating between `SIOS` and `SIO` (possibly also showing `SIN` or `SIE`). If they are not doing so, then there is an ss7 driver problem, report to aculab support.

The receive counts should also be increasing, if none of them are changing check that the TDM cable is actually plugged in, that layer 1 is synchronised (use `sysdiag`), that the correct timeslot is being used, and that the remote system has actually provisioned an MTP2 link at all!

If the `bad` count (value `3` above) is incrementing (there might be some receive lssu - typically `SIO` and `SIOS`) then the most likely cause is layer 1 frame slip caused by incorrect receive TDM clocking. For ANSI T1 links the signalling might be at 56k, not 64k.

If the error rate is low enough, then MTP2 may start proving (sending SIN or SIE) before taking the link down.

If there are no MTP2 errors, but the signalling links are taken out of service after about a minute, it might be an MTP3 issue see 8.6.3.

The tests below might be useful:

8.6.1.1 Receive timeslot data check

If MTP2 isn't receiving any valid MTP2 frames, it can be worth checking that the receive data is actually changing. Run `swcmd -a serial 32+port timeslot` a few times and check that the value is changing, or `swcmd -a serial 32+port 1 -n 31` to look at all the timeslots.

This might show that the signalling is actually on an unexpected timeslot (eg timeslot 1, not timeslot 16).

On Prosody X Evo systems it is also possible to check the transmit data by specifying `-at` instead of `-a`.

8.6.1.2 MTP2 local loopback test

Confirmation that the ss7 software is functioning can be obtained by running a local loopback test.

The MTP2 data is routed to the correct trunk and timeslot by the TDM switch. This means it is possible to use the TDM switch to loop the MTP2 transmit traffic back on itself. The MTP2 link should establish.

To apply the loopback you need to know the stream and timeslot number used by MTP2. Either run `swcmd -q serial 32+port timeslot`, or `ss7maint linkstatus -2vv` and divide/remainder the `MTP2 timeslot` value by 32 and add 48 to the stream number. This should give you a stream between 48 and 55 and a timeslot between 0 and 31. Run:

```
swcmd -c serial stream timeslot stream timeslot
```

to apply the loopback.

Watch the MTP2 counters, the receive and transmit counts/states should now follow each other and the link establish.

When MTP3 fails to receive the correct SLTM the link will be stopped and restarted. This will reset the TDM switch (unless `-cNOSW` was specified in the firmware download parameters).

In the same way you can loop the TDM received data onto the TDM transmit timeslot.

8.6.1.3 Maintenance api library link status

The maintenance API library link status command contains some additional statistics that aren't normally displayed by `ss7maint linkstatus`. These values can be displayed by running:

```
ss7maint apistatus -C mtp link
```

or:

```
ss7maint linkstatus -C
```

As well as byte and packet counts, the output includes a count of `errors` and `failures` (failures are more serious and cause the link to go down), the type of the last `failure` and `error` and how long ago they happened - anything that happened just after the firmware download is really not interesting at all. The low 8 bits of the error/failure types are the MTP2 state, the upper 8 the error:

MTP2 states:

0x..00	Idle (repeated errors are suppressed).
0x..01	Not aligned.
0x..02	Aligned.
0x..03	Proving.
0x..04	Emergency proving.
0x..05	Aligned ready.
0x..06	Data transfer.

HDLC framing errors (etc):

0x01..	Receive abort.
0x02..	Receive too long.
0x03..	Receive not multiple of 8 bits.
0x04..	Receive CRC error.
0x05..	Receive frame too short.
0x06..	LI field incorrect for frame length.
0x07..	No receive buffers available.

FISU/MSU discards:

0x11..	First invalid BSN.
0x12..	Frame following bad BSN.
0x13..	First invalid FIB.
0x14..	Frame following bad FIB.
0x15..	Incorrect FIB, waiting retransmission.

0x16..	Not in data transfer.
0x17..	Bad FSN, retransmit requested.
0x18..	Duplicate FSN.
0x19..	Congestion discard.

Disconnect reasons:

0x81..	Protocol timer expired.
0x82..	LSSU invalid for current state.
0x83..	Invalid BSN received.
0x84..	Invalid FIBR received.
0x85..	Error rate monitor tripped.
0x8d..	Connection from driver lost.
0x8e..	Disconnected by MTP3.
0x8f..	Firmware stopped.

The most likely reason for any of the HDLC framing errors is TDM frame slip caused by using an incorrect TDM clock.

The same error information is available from the ss7 driver trace.

8.6.1.4 Prosody 1U enterprise MTP2 issues

On a Prosody X1U enterprise card the MTP2 code runs on a DSP. A simple ppc Linux program (`hdirelay`) transfers messages between the TCP connection to the ss7 driver and the DSP.

The DSP MTP2 code (`pmx_ss7.eld`) is copied to the card and the `hdirelay` program started when the aculab resource manager processes the `ss7.manifest` file during firmware download (both files are in the `$ACULAB_ROOT/Firmware/pmx` directory). Run `trace_mode resmgr all` to get any diagnostics. The DSP itself is loaded indirectly by request from the `ss7.pmx` firmware. It is difficult to directly determine whether this succeeded. To force a new version to be loaded reset the card.

8.6.1.5 Prosody X rev 3 MTP2 issues

On Prosody X rev 3 cards the MTP2 is controlled by the `ss7_mtp2_pmxrev3` program (running on the card's ppc Linux). A single copy handles all the signalling links on the card. The actual protocol code runs on two soft-core processors on the card's TDM FPGA.

The `ss7_mtp2_pmxrev3` program is executed by every download of `ss7.pmx` (even ones that don't specify signalling links), if a copy is already running the new one will exit unless it is a newer version (compiled later) in which case the old version will exit instead.

To verify that it is running look at the output of `board_cmd serial 0 run ps`. If it isn't running, especially if the file `/firmware/ss7_mtp2_pmxrev3` file is absent, it might just be that the `SS7_MTP2_PMXV3` AIT package hasn't been installed!

`ss7_mtp2_pmxrev3` is copied to the card and run when the aculab resource manager processes the `ss7.manifest` file during firmware download (both files are in the `$ACULAB_ROOT/Firmware/pmx_v3` directory). Run `trace_mode resmgr all` to get any diagnostics.

If `ss7_mtp2_pmxrev3` detects a fatal error it writes some diagnostic output to the ss7 driver trace before exiting. The diagnostics can also be requested by sending the program a SIGINT (e.g.: `board_cmd` *`serial`* `0 run kill -INT` *`pid`*). The board memory dump (see 8.5) will contain useful diagnostics (include the sdram contents if possible).

If `ss7_mtp2_pmxv3` seems to be exiting immediately, it might be worth executing it directly. Without console access `board_cmd -i` *`serial`* `0 run /firmware/ss7_mtp2_pmxrev3 -vv` to get the program output relayed to the local terminal. With console access the program can be run in the background tracing to the terminal.
The full options are:

`-v`	Verbose: directs diagnostics, trace and errors to `stdout` (rather than `syslog()`), may be repeated. The trace is very low level.
`-f 1`	Ignore some FPGA dual port memory errors (fixed in fpga version 97 51).

`-f 2`	Re execute after fatal error.
`-f 4`	Don't delay indications to the host in order to merge requests (useful for some types of performance measurement).
`-f 8`	Don't exit if FPGA version is older than 97 51.
`-f n`	Bitwise or of the above.
`-p port`	Use specified TCP port, default 8240. The ss7 driver always connects to port 8240.
`-d driver`	Driver path, default `/dev/arriaIIdev`.

While the program is running, `SIGUSR1` will decrement the trace level and `SIGUSR2` increase it.

Typing CTRL+C generates `SIGINT` and dumps some state to the ss7 driver trace. CTRL+\ will make the program exit.

The `-f` options can also be set/cleared by the `set_misc_flags`/`clr_misc_flags` MTP2 firmware parameters.

8.6.1.6 Prosody X Evo MTP2 issues

The architecture of the Prosody X Evo is much the same as that of the Prosody X rev 3 card except that the embedded Linux runs on an x86 motherboard rather than a small ppc processor.

The main differences are that the program is called `ss7_mtp2_prosody_t`, is downloaded from `$ACULAB_ROOT/Firmware/pxi` and is in AIT package `SS7_MTP2_PT_LINUX`.

The two physical Prosody T cards in a sixteen port Prosody X Evo system are handled by a single copy of `ss7_mtp2_prosody_t`.

8.6.2 M2PA signalling links

The link status commands will show information for M2PA signaling links as well as MTP2 ones. The most obvious difference is that lssu (eg SIOs) and fisu are not transmitted repeatedly (lssu might only be transmitted once) – so the counts do not increase the same way.

The fisu counters increment whenever an empty data frame (ie one that is just an acknowledgment) is transmitted or received.

8.6.3 MTP3 SLTA errors

If the signalling links are coming into service at MTP2, but MTP3 is reporting an SLTA error (after which it takes the links down and tries again), then it is likely that one of the fundamental MTP3 parameters is wrong. Most likely one of:

slc (signalling link code)	Check cables in correct sockets.
pointcodes	Check both local and remote values.
ni (network indicator)	Check ss7.cfg, the default is `ni=0` but you probably want `ni=2`.
major variant	Check that ITU/ANSI is correct in the ss7 stack configuration file (`ss7.cfg`).

Take the driver trace, decode and compare the received and sent SLTM for the signalling link. The parameters in the sent SLTM need to match those received (with the pointcodes swapped over of course).

If only the slc is wrong (e.g.: if two cables are swapped) then both MTP3 will send SLTA responses on the signalling link associated with the slc, not the one the SLTM was received on. A quick glance would indicate that all the packets are present - but careful inspection shows the error.

8.6.4 MTP3 local pointcode restart

When MTP3 connects to the SS7 network it expects adjacent pointcodes to send it routing updates (e.g. indications that remotes pointcodes are inaccessible) followed by a TRA. Until this has happened, or timer ITU T20 (ANSI T23) has expired, userparts (ISUP and SCCP)

cannot send messages to any destinations.

Configuring `tra_needed=n` for an adjacent pointcode stops MTP3 from waiting for a TRA from the specific adjacent before deciding that there are 'sufficient TRA' to continue the pointcode restart procedures.

If an SS7 network only has 2 systems (e.g. two SEP connected together) then both systems will wait for T20 (60 seconds) before traffic can be sent. (ANSI has a 3 second T28 which will expire before T23 is started.)

This 60 second delay can make it appear that the network is broken. During this period `ss7maint mtp3status` will show what is happening. In the example below pointcodes 1 and 2 are STPs, pointcode 1111 doesn't exist (but is marked as an adjacent to 2020), and pointcode 2020 has just started (or been temporarily isolated with `ss7maint linkactivate`).

```
  opc                 dpc
    1                   2 Accessible
    1                1111 Inaccessible
    1                2020 Inaccessible Restarting
    1                7070 Accessible
    2                   1 Accessible
    2                1111 Inaccessible
    2                2020 Inaccessible Restarting
    2                7070 Accessible
 2020    Restarting     1 Inaccessible
 2020    Restarting     2 Inaccessible
 2020    Restarting  1111 Inaccessible
 2020    Restarting  7070 Inaccessible
```

Once T20 expires everywhere (except 1111) becomes accessible. If this is a problem, configure T20 to a smaller value.

If the overall pointcode restart type is configured to `restart=ITU` (rather than `restart=ITU_white`) pointcodes 1 and 2 would be accessible from 2020 (because the direct linkset is up), but 7070 would still be inaccessible. (The STPs 1 and 2 would need configuring the same way for this to be at all useful.)

Note **A remote pointcode is considered to be 'adjacent' if the direct linkset data structure has been created. This happens if it is configured in an `[STP]` section, has an actual signalling link, or is explicitly requested with `adjacent=y`. The direct linkset data structure is never deleted.**

8.6.5 ISUP CIC mismatches

ISUP is normally configured with `reset_circuits=y` so the circuits are reset after a firmware download. Until the reset completes the circuits are unavailable for calls and `sysdiag` will report the 'L2 State' as 0 (for ISUP the L2 State means 'available for calls').

The circuit reset is done with one or more 'group reset' messages - each resets a block of consecutive CIC numbers. If any of these CIC numbers is invalid, then the remote system won't respond and the reset request will be retried forever.

A CIC mismatch can easily arise because E1 timeslot 16 is often used for a signalling link (or is reserved for one). So if timeslot 1 is CIC 1, timeslot 15 will be CIC 15, timeslot 16 might be allocated a CIC number (not used by ISUP) so timeslot 17 might be CIC 16 or CIC 17 - depending on how the network has decided to allocate CIC numbers, with timeslot 31 being either CIC 30 or 31.

The timeslot to CIC mapping is defined by the `-cCIC` firmware download parameter, specify `-cCIC`*`cic_base`*`,1-31,1-15:17-31` if timeslot 16 is allocated a CIC number and `-cCIC`*`cic_base`*`,1-15:17-31` if it isn't. See 5.2.4.

9 Laboratory testing features

Many customers will find that, at some point, network operators need to subject their application to conformance testing under laboratory conditions. Aculab SS7 includes certain features that are intended to simplify and expedite this process.

For MTP2 and MTP3, the vast majority of such testing is carried out in accordance with test recommendations published by ITU-T (Q.781, Q.782 series). Some of these tests require the system to generate test traffic and the traffic `reflect` and `generate` functions of `ss7maint` are compatible with these test traffic requirements.

Appendix A: includes some example configuration files that can be used for running these tests.

9.1 ss7maint reflect

In this mode, `ss7maint` simply reflects all received traffic, for a given service indicator and network indicator, back towards the network. This can be useful under test conditions, as it enables the tester to confirm that test traffic reaches the application, and that traffic generated by the application is correctly routed back to the network.

Accessibility status of remote destinations is displayed on screen while this command is running. The command will run until an error is encountered, or until it is interrupted with `Ctrl+C`.

Available options for `ss7maint reflect` are as follows:

`-o` *`nnnnn`* Identifies the local point code. This parameter must be specified.

`-n` *`ni`* Identifies the network indicator for which traffic will be reflected. By default, traffic for all values of network indicator (*`ni`*) will be reflected.

`-s` *`si`* Identifies the service indicator for which traffic will be reflected. By default, traffic for *`si`* `8` will be reflected.

`-t` *`time`* Statistics including the count of messages received on each `sls`, output every `time` seconds.

`-D` Send data for sls 8 through 15 via the dual system (ignores the higher bits of the sls on ANSI networks). Gives better sls distribution when conformance testing a dual MTP3 system.

`-d` Received data is discarded (not reflected).

`-v` Verbose, every message is reported.

Example:

```
ss7maint reflect -o1234 -t10
```

Will reflect back all data received at pointcode 1234.

9.2 ss7maint generate

This can be useful where there is a requirement for the equipment under test to generate traffic automatically, which the tester can monitor to evaluate link utilisation, load sharing, and routing.

-o *nnnnn*	Local point code.
-n *ni*	Identifies the Network Indicator for which traffic will be generated. By default, traffic will be generated for *ni* 0.
-s *si*	Identifies the Service Indicator for which traffic will be generated. By default, traffic will be generated for *si* 8.
-c *count*	Identifies the number (*count*) of messages that will be generated before the program terminates. By default, traffic will be generated continuously until interrupted using Ctrl+C.
-l *min* [-*max*]	Determines the length of generated messages. If both min and max are given then the size will be a random number from min to max. Default 100 bytes.
-i *min* [-*max*]	Determines the interval between messages in milliseconds (subject to the resolution of the system timer). If both min and max are given, then a random interval from min to max will be used. Default 0, no delay between messages.
-S *min* [-*max*]	Determines the sls values used when transmitting messages, sls values from min to max are used in turn.
-t *time*	Statistics, including the number of messages sent on each SLS, output every *time* seconds.
-r	Reflect data after sending the initial *count* messages. Received data is verified and a new message generated.
-v	Verbose, every received message is reported.
-N *number*	Provides the initial identification *number* for messages, the value is incremented each time a message is sent on a particular sls to a specific destination.
-z	Causes zero-filled data to be sent. This is required for strict conformance with ITU-T Q.782. By default, random data will be sent.

A list of destination DPC values must follow the options. A message is sent to each destination in turn.

Example:

```
ss7maint generate –o1234 -z 5678
```

9.3 MTP3 performance measurements

With a single trunk connecting two systems you can use `ss7maint reflect` and `ss7maint generate` for some simple MTP3 performance measurements. The following commands setup two reflectors and then insert 16 messages (one with each sls value), each 64 bytes long, into the reflect loop.

```
ss7maint reflect -o4321 -t10
ss7maint reflect -o1234
ss7maint generate -o1234 -c16 -l64 4321
```

After 10 seconds, the first reflect program will output some statistics:

```
time  <---------- number of packets processed on each sls ----------> total overall
  10:  70  70  70  70  70  70  70  70  70  70  70  70  70  70  69  1119    1119
  20:  66  66  66  66  66  66  66  66  66  65  65  65  65  65  66  1050    2169
```

All the traffic will be using a single signalling link on timeslot 16 - which will be 100% loaded. We can add another 15 signalling links:

```
fwdspldr -reconfig serial 0 -cTS1 -cslc1-15
fwdspldr -reconfig serial 1 -cTS1 -cslc1-15
```

Once the new signalling links have finished MTP2 proving the message rate increases substantially.

```
110: 361 360 360 360 361 360 360 360 360 360 360 360 360 360 360 360  5762   22708
```

However we now only have a single message going back and forward on each signalling link. So add some more messages, another 48 gives 4 on each link.

```
ss7maint generate -o1234 -c48 -l64 4321
```

```
350: 1051 1048 1046 1048 1046 1051 1054 ... 1053 1050 1048 1050 1049 16787  196072
```

All 16 signalling links are now 100% loaded (look at the output of `ss7maint linkstatus -S0 -2v` and note that the receive and transmit fisu counts are not changing).

With smaller messages it needs more than 4 messages on each sls to saturate the signalling links.

The TCAP package contains the source for two TCAP applications that can be used in a similar way to measure the performance of the TCAP interface.

10 Manual driver administration using ss7maint

In addition to the features of `ss7maint` already described, there are some additional features that allow you to administer some aspects of driver installation and service status manually. These features may be different, and may have different effects, for each OS platform. Please refer to the section that corresponds to the platform you are using.

The Aculab installation tools usually handle driver administration automatically. Manual administration would not normally be required, so you should only use the commands described in this section if instructed to do so by Aculab support.

10.1 Windows driver administration

10.1.1 ss7maint load

This loads the SS7 driver into system memory.

10.1.2 ss7maint unload

This unloads the SS7 driver from memory. When followed by `ss7maint load`, this can be useful as a means of restoring a system without the inconvenience of a full system reboot.

10.1.3 ss7maint uninstall

This completely unloads and uninstalls the SS7 driver.

10.1.4 ss7maint install

This installs and loads the SS7 driver. By default, `ss7maint` will expect the kernel driver file `aculabss7.sys` to be present in your current directory but, if preferred, you can specify a different directory and filename using the `ss7maint install -f<filename>`.

10.1.5 Aculab SS7 TS Manager service

This service requests TDM switch matrix connections and controls ss7 software licensing. It is normally installed and started by `ss7maint install`, and removed by `ss7maint uninstall`. It can be also be installed by `ss7sxc -i` and removed by `ss7sxc -u`.

10.1.6 Windows driver diagnostics

The windows `windbg` program can look at the live system as well as kernel dumps. To look at a live system **`-debug on`** has to be specified when the system is booted.

The symbol path needs setting to something like:

```
symsrv*symsrv.dll*c:\localsymbols*http://msdl.microsoft.com/download/symbols;
c:\Program Files\Aculab\v6\drivers
```

The kernel stacks for all programs **`foo.exe`** can be obtained by typing (after the kd> prompt):

```
!process 0 7 foo.exe
```

The stacks for the ss7 kernel threads can be displayed by (as a single line):

```
!list "-x \"da /c36 @$extret+6*@$ptrsize+4; dw @$extret+@$ptrsize L1;
!thread -t @$p;\" poi(aculabss7_64!ss7_threads)"
```

The following command will extract the ss7 driver trace:

```
!list "-x \"r $t0 = @$extret+3*@$ptrsize; dd /c100 @$t0 L3; da /c100
@$t0+1c+@$ptrsize L120; db poi(@$t0+18)+dwo(@$t0+10)
poi(@$t0+18)+dwo(@$t0+14)-1 ;\" -m 11111
poi(aculabss7_64!ss7_trace_qq_pending)"
```

The output can be written to a file by bracketing the commands between **`.logopen c:\file`** and **`.logclose`**.

Further information is in %ACULAB_ROOT%\ss7\scripts\windbg_cmds.txt

10.2 Linux driver administration

10.2.1 Building the driver

After installing the ss7 AIT package part of the driver must be compiled with the header files for the installed kernel. The `aculab_dacp` script does this for ss7 as well as the main call control driver.

10.2.2 ss7maint load

This loads the SS7 driver into system memory.

When this command is run, a daemon process is started to administer threads within the Linux kernel, to request TDM switch matrix connections, and for software licensing.

This process may fork additional child processes whenever the ss7 driver needs another thread. These are all reported as `ss7maint load` by `ps(1)`.

The following options are available to modify the behaviour of this process:

`-f <module-pathname>`
: By default, `ss7maint` will expect the SS7 driver to be present in the current directory. You can use this option to specify a different directory and filename.

`-c`
: Causes the process not to change its current directory to root.

`-d`
: Causes the process not to be run as a daemon. A daemon process typically takes several steps to insulate itself from all other processes and devices: puts itself into the background, ignores certain signals and starts a new session (makes itself the group leader of a new process group).

`-r`
: Causes the process not to redirect its `stdin`, `stdout`, and `stderr` streams to `/dev/null`.

In normal use, only the **`-f`** option is used. If problems are observed when the daemon process is starting or running, turning off some or all of these features may aid debugging.

10.2.3 ss7maint unload

This requests the SS7 driver be unloaded from memory.

This can be useful as a means of restoring a system without the inconvenience of a full system reboot when followed by `ss7maint load`.

The unload may not complete until all applications (especially TCAP ones) have terminated.

10.2.4 Linux driver diagnostics

A few very important errors get written to the kernel message buffer (view with `dmesg`) and, via syslog, to the system log files and, possibly, all terminals. If the ss7 driver seems to have locked up it is worth looking for such system errors.

If a process is 'stuck' in the kernel, then run **`cat /proc/<pid>/stack`** to get its kernel stack traceback. Hopefully this will give a clue to the problem. The pid values for the ss7 kernel threads can be found by running **`ss7maint osstatus -t`**.

Note **The symbol names reported may include all code addresses found on the stack not just those of the current call chain.**

Appendix A: Example configurations

These examples illustrate various common configurations. Each example includes the stack configuration file and firmware download parameters, and is accompanied by a network diagram.

The firmware downloads all specify the `ss7.pmx` file for Prosody X rev 3 cards. For Prosody X Evo systems you should specify `ss7.pxi` but the library code will change the filename suffix for you.

You may well find that one of these examples, with a few modifications, can be used to configure your installation. You will have to substitute the correct values for signalling point codes, and also take care to check what additional changes are necessary. Such as the correct values for Network Indicator (`ni`) in the stack configuration file and the Signalling Link Code (SLC) in the firmware parameters.

Text in the example stack configuration files has been indented to indicate nesting levels within sections but this is purely cosmetic as all white-space is ignored.

- In the example stack configuration files and firmware parameters:

  aaaaa indicates the point code for SP A (this represents the Aculab endpoint being configured)
  bbbbb indicates the point code for SP B
  ccccc indicates the point code for SP C
  ddddd indicates the point code for SP D
  eeeee indicates the point code for SP E

- In the example firmware parameters:

  sssss is used to represent the serial number of the Aculab digital access card.

The diagram components in the following examples are defined as follows:

Signalling relation (voice path)
Single signalling link link-set.
Multiple signalling link link-set.


Signalling end point.

Aculab Signalling end point


Signalling transfer point.

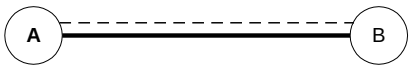

Joint signalling end point and signalling transfer point.

A.1 Typical configuration examples.

A.1.1 Fully associated mode to a single network destination.

This example shows the simplest form of configuration, where the network connection takes the form of a single, fully-associated signalling link set. ISUP and TCAP signalling are in use.

In the firmware parameters, observe how the `-cDPC` parameter is specified only once, before either `-cSLC` or `-cCIC`, to indicate that the same value applies to the signalling link and ISUP bearer circuits.



Stack configuration file

```
[SP]
  localPC = aaaaa
  ni = 2
  [ISUP]
  [EndISUP]
  [TCAP]
  [EndTCAP]
  [SCCP]
  [EndSCCP]
  [MTP3]
  [EndMTP3]
[EndSP]
```

Firmware parameters for single link system

```
fwdspldr ssssss 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC0 -cCIC1
```

Firmware parameters for quad link system

```
fwdspldr ssssss 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC0 -cCIC1
fwdspldr ssssss 1 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC1 -cCIC33
fwdspldr ssssss 2 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC2 -cCIC65
fwdspldr ssssss 3 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC3 -cCIC97
```

Firmware parameters for dual link signalling on ts1 and ts16 of one E1 trunk

```
fwdspldr ssssss 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC0
         -cTS1 -cSLC1 -cCIC1
```

Firmware parameters for quad link signalling on ts1, 2, 3 and ts16 of one E1 trunk on a PMX module.

```
fwdspldr ssssss 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS1 -cSLC1-3
         -cTS16 -cSLC0 -cCIC1
```

Firmware parameters for a high speed signalling link using all the timeslots of port 0

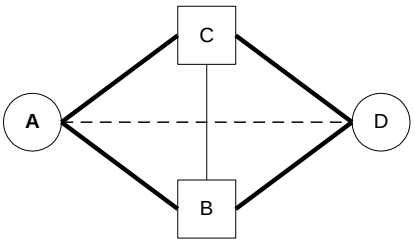
```
fwdspldr ssssss 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS1-31 -cSLC0
```

Note that this probably won't interwork with a remote system because that will expect extended sequence number be used. But it can be used for testing monitoring of high speed signaling links.

A.1.2 Quasi associated mode with two load-sharing STPs, to a single destination

In this example, traffic may be routed via two load-sharing STPs, to a destination Signalling point elsewhere within the network. ISUP signalling is in use, and it is assumed that all ISUP bearers circuits are connected between A and D.

In the firmware parameters, observe that the `-cDPC` parameter is specified before the `-cSLC` parameter, and then repeated before the `-cCIC` parameter, to indicate that a different value applies to the signalling link and ISUP bearer circuits.



Stack configuration file

```
[SP]
  localPC = aaaaa
  ni = 2
  [ISUP]
  [EndISUP]
  [MTP3]
    restart=ITU_White

    # Declare the two STPs...
    [STP]
      adjacentPC = bbbbb
      # Alternative route to this STP via the other STP
      [ROUTE]
        priority = 1
        adjacentPC = ccccc
      [EndROUTE]
    [EndSTP]

    [STP]
      adjacentPC = ccccc
      # Alternative route to this STP via the other STP
      [ROUTE]
        priority = 1
        adjacentPC = bbbbb
      [EndROUTE]
    [EndSTP]

    [DESTINATION]
      remotePC = ddddd
      # Load-sharing routes via both STPs
      [ROUTE]
        adjacentPC = bbbbb
      [EndROUTE]
      [ROUTE]
        adjacentPC = ccccc
      [EndROUTE]
    [EndDESTINATION]
  [EndMTP3]
[EndSP]
```

Firmware parameters for dual link system

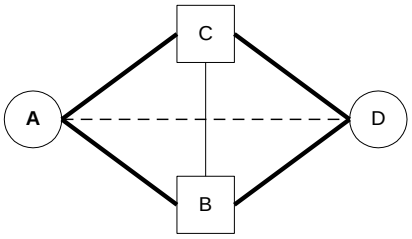
```
fwdspldr ssssss 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC0
         -cDPCddddd -cCIC1
fwdspldr ssssss 1 ss7.pmx -cOPCaaaaa -cDPCccccc -cTS16 -cSLC0
         -cDPCddddd -cCIC33
```

Firmware parameters for quad link system

```
fwdspldr ssssss 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb –cTS16 -cSLC0
         –cDPCddddd –cCIC1
fwdspldr ssssss 1 ss7.pmx -cOPCaaaaa -cDPCbbbbb –cTS16 -cSLC1
         –cDPCddddd –cCIC33
fwdspldr ssssss 2 ss7.pmx -cOPCaaaaa -cDPCccccc –cTS16 -cSLC0
         –cDPCddddd –cCIC65
fwdspldr ssssss 3 ss7.pmx -cOPCaaaaa -cDPCccccc –cTS16 -cSLC1
         –cDPCddddd –cCIC97
```

A.1.3 Normal and alternative STPs to a single destination.

This example is similar to the previous one but, instead of load-sharing traffic between the two STPs, the traffic has a "`normal`" route via STP B over which all traffic will normally be sent – it will not be load-shared. If, however, the normal route is unavailable, then traffic will be sent via STP C. ISUP parameters are not shown, but they could easily be added following the model illustrated in the first example.



Stack configuration file

```
[SP]
  localPC = aaaaa
  ni = 2
  [MTP3]
    Restart=ITU_White
    # Create priority 1 routes to everywhere via all adjacents
    Default_route_priority = 1

    # Declare the two STPs...
    [STP]
      adjacentPC = bbbbb
    [EndSTP]
    [STP]
      adjacentPC = ccccc
    [EndSTP]

    [DESTINATION]
      remotePC = ddddd
      [ROUTE]
        # Explicitly specify a lower priority route
        priority = 2
        adjacentPC = ccccc
      [EndROUTE]
    [EndDESTINATION]
  [EndMTP3]
[EndSP]
```

Firmware parameters for dual link system

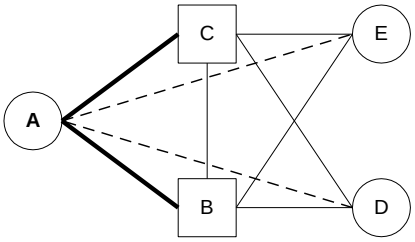
```
fwdspldr ssssss 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC0
fwdspldr ssssss 1 ss7.pmx -cOPCaaaaa -cDPCccccc -cTS16 -cSLC0
```

Firmware parameters for quad link system

```
fwdspldr ssssss 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC0
fwdspldr ssssss 1 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC1
fwdspldr ssssss 2 ss7.pmx -cOPCaaaaa -cDPCccccc -cTS16 -cSLC0
fwdspldr ssssss 3 ss7.pmx -cOPCaaaaa -cDPCccccc -cTS16 -cSLC1
```

A.1.4 Two load-sharing STPs, to two different destinations.

This example is similar to example A.1.2, but two different network destinations are declared, so application traffic could be sent to either of them. ISUP parameters are not shown, but they could easily be added following the model illustrated in example A.1.1.



Stack configuration file

```
[SP]
  localPC = aaaaa
  ni = 2
  [MTP3]
    restart=ITU_White
    default_route_priority = 1

    # Declare the two STPs...
    [STP]
      adjacentPC = bbbbb
    [EndSTP]
    [STP]
      adjacentPC = ccccc
    [EndSTP]

    # First destination
    [DESTINATION]
      remotePC = ddddd
    [EndDESTINATION]

    # Second destination
    [DESTINATION]
      remotePC = eeeee
    [EndDESTINATION]
  [EndMTP3]
[EndSP]
```

For ISUP (but not SCCP) you don't actually need to specify the `[DESTINATION]` sections they will be created when the ISUP circuits are added.

Firmware parameters for dual link system

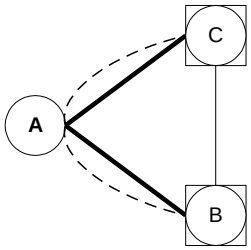
```
fwdspldr ssssss 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC0
fwdspldr ssssss 1 ss7.pmx -cOPCaaaaa -cDPCccccc -cTS16 -cSLC0
```

Firmware parameters for quad link system

```
fwdspldr ssssss 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC0
fwdspldr ssssss 1 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC1
fwdspldr ssssss 2 ss7.pmx -cOPCaaaaa -cDPCccccc -cTS16 -cSLC0
fwdspldr ssssss 3 ss7.pmx -cOPCaaaaa -cDPCccccc -cTS16 -cSLC1
```

A.1.5 Normal and alternative routes, to combined STPs and destinations.

In this example, there are two network destinations. Both are connected via direct signalling linksets and traffic is normally sent in fully-associated mode on the direct linkset. If, however, the direct linkset is unavailable, each destination can also function as an STP allowing traffic to be routed to the other destination via a cross-link. ISUP parameters are not shown, but they could easily be added following the model illustrated in example A.1.1.



Stack configuration file

```
[SP]
  localPC = aaaaa
  ni = 2
  [MTP3]
    restart=ITU_White

    # First destination
    [DESTINATION]
      remotePC = bbbbb
      # Alternative route via other destination acting as STP
      [ROUTE]
        adjacentPC = ccccc
        priority = 1
      [EndROUTE]
    [EndDESTINATION]

    # Second destination
    [DESTINATION]
      remotePC = ccccc
      # Alternative route via other destination acting as STP
      [ROUTE]
        adjacentPC = bbbbb
        priority = 1
      [EndROUTE]
    [EndDESTINATION]
  [EndMTP3]
[EndSP]
```

Here the `[DESTINATION]` sections will be created when the signalling links are added, so it is enough to specify:

```
[SP]
  localPC = aaaaa
  ni = 2
  [MTP3]
    restart=ITU_White
    default_route_priority = 1
  [EndMTP3]
[EndSP]
```

Firmware parameters for dual link system

```
fwdspldr ssssss 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC0
fwdspldr ssssss 1 ss7.pmx -cOPCaaaaa -cDPCccccc -cTS16 -cSLC0
```

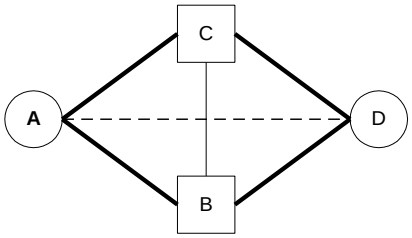
Firmware parameters for quad link system

```
fwdspldr ssssss 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC0
fwdspldr ssssss 1 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC1
fwdspldr ssssss 2 ss7.pmx -cOPCaaaaa -cDPCccccc -cTS16 -cSLC0
fwdspldr ssssss 3 ss7.pmx -cOPCaaaaa -cDPCccccc -cTS16 -cSLC1
```

A.1.6 Test network STP

This example shows two STP (on a single computer) that could be used with the earlier SEP configurations. Using two STP ensures that the STP systems only do 'adjacent pointcode restart' when the SEP connect.

In this case the configuration is for systems B and C.



Stack configuration file

```
[SPDEFAULT]
  ni = 2
  [MTP3]
    restart = itu_white
  [endMTP3]
[endSPDEFAULT]

[SP]
  localPC = bbbbb
  [MTP3]
    stp = y
    q704_t20 = 5s    # Speed up local pointcode restart
    [STP]
        adjacentPC = ccccc
        default_route_priority = 32
    [endSTP]
  [endMTP3]
[endSP]

[SP]
  localPC = ccccc
  [MTP3]
    stp = y
    [STP]
        adjacentPC = bbbbb
        default_route_priority = 32
    [endSTP]
  [endMTP3]
[endSP]
```

Firmware parameters

With a 4 port card, port 0 can be used to connect both STP systems to SEP A, port 1 to SEP D, and ports 2 and 3 connected using a cross over cable for the links between the STP.

With 2 signalling links in each linkset:

```
fwdspldr ssssss 0 ss7.pmx -cOPCbbbbb -cDPCaaaaa -cTS1 -cSLC0-1
        -cOPCccccc -cTS16 -cSLC0-1
fwdspldr ssssss 1 ss7.pmx -cOPCbbbbb -cDPCddddd -cTS1 -cSLC0-1
        -cOPCccccc -cTS16 -cSLC0-1

fwdspldr ssssss 2 ss7.pmx -cOPCbbbbb -cDPCccccc -cTS1 -cSLC0-1
fwdspldr ssssss 3 ss7.pmx -cOPCccccc -cDPCbbbbb -cTS1 -cSLC0-1
```

It is also possible to configure the inter-STP signalling links without using two more ports by connecting the TDM timeslots that connect MTP2 to the TDM switch to each other rather than to the external E1/T1 trunk.

The MTP2 code uses a timeslot number between 0 and 127 (126 for the Prosody X rev 3

card), these correspond to TDM switch streams 48 to 51 (each having 32 timeslots).

First generate the signalling links. Here they are associated with timeslots 30 and 31 but the data won't appear on those timeslots; so they can still be used for voice.

A Prosody 1U enterprise system will almost certainly use a timeslot of `port * 32 + trunk_timeslot` for the TDM data. The Prosody X rev 3 card allows the signalling to be forced onto specific timeslots eg 64 to 67.

```
fwdspldr -reconfig ssssss 0 -cMTP2_timeslot=64
fwdspldr -reconfig ssssss 1 -cMTP2_timeslot=66

fwdspldr -reconfig ssssss 0 -cNOSW -cOPCbbbbb -cDPCccccc -cTS30 -cSLC0-1
fwdspldr -reconfig ssssss 1 -cNOSW -cOPCccccc -cDPCbbbbb -cTS30 -cSLC0-1
```

Verify the allocated timeslots by running `ss7maint linkstatus -vv` and looking at the `MTP2 timeslot` values. They should be 64 to 67 on a rev 3 card and 30, 31, 62 and 63 on a 1U enterprise system. Make the correct switch matrix connections:

For a 1U enterprise system:

```
swcmd -c sssss 48 30 49 30 -n 2
swcmd -c sssss 49 30 48 30 -n 2
```

For a PX rev 3 card (using timeslots 64 to 67):

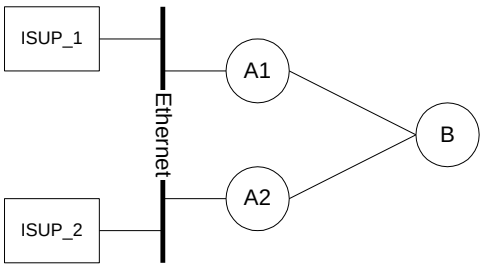
```
swcmd -c sssss 50 0 50 2 -n 2
swcmd -c sssss 50 2 50 0 -n 2
```

Run ss7maint **`linkstatus -2 -3 -v -h -i 200`** and check that all the signalling links are transmitting and receiving traffic. Wait for MTP3 to indicate the links are available.

A.1.7 Distributed ISUP and Dual MTP3.

This example shows a configuration with two systems running ISUP connected to a dual MTP3 setup. For simplicity a single network destination is shown, in practise the MTP3 sections will need to describe a more complex network.

Only ISUP signalling is supported by dual MTP3 configurations.



Stack configuration file for ISUP_1 and ISUP_2

```
[SP]
  localPC = aaaaa
  ni = 2
  [ISUP]
    slc_shift = 2
    [REMOTEMTP3]
      host = mtp3_a1
      net_select = 1
    [EndREMOTEMTP3]
    [REMOTEMTP3]
      host = mtp3_a2
    [EndREMOTEMTP3]
  [EndISUP]
[EndSP]
```

Stack configuration file for MTP3_A1

```
[SP]
  localPC = aaaaa
  ni = 2
  [ISUP]
    mtp3_listen = y
    lowest_cic = 1
    highest_cic = 63  # For fwdspldr commands below
  [EndISUP]
  [MTP3]
    [DUAL]
       host = mtp3_a2
       master = y
       listen = 14
       connect = 15
    [EndDUAL]
    [DESTINATION]
      remotePC = bbbbb
    [EndDESTINATION]
  [EndMTP3]
[EndSP]
```

Stack configuration file for MTP3_A2

```
[SP]
  localPC = aaaaa
  ni = 2
  [ISUP]
    mtp3_listen = y
    lowest_cic = 1
    highest_cic = 63
  [EndISUP]
  [MTP3]
```

```
    [DUAL]
       host = mtp3_a1
       master = n
       listen = 15
       connect = 14
    [EndDUAL]
    [DESTINATION]
      remotePC = bbbbb
    [EndDESTINATION]
  [EndMTP3]
[EndSP]
```

Firmware parameters for dual link system

```
isup_1: fwdspldr sssss0 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cCIC1
isup_2: fwdspldr sssss1 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cCIC33
mtp3_a: fwdspldr sssss2 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC0
mtp3_b: fwdspldr sssss3 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC1
```

CICs could be defined for ports on systems mtp3_a and mtp3_b. Traffic from ISUP on mtp3_a/b will always be sent over the local signalling link provided that link is active (if inactive it will be sent using TCP to the other MTP3 system). Received traffic is always forwarded to the correct system.

A.1.8 ISUP codec message definition

This example illustrates an ISUP codec extension to support the use of non-standard parameters in ISUP messages. In this case adding a single extra parameter to FAR message so that an application can use flexible ISUP to support Nortel RLT (release link trunk).

Codec extension file rlt_example.cfg

```
[ISUPCodec]
   name = rlt

   # FAR
   [Msg]
        code = 0x1f
        # Start with the base ITU definition in the driver
        msg_base = itu
        # Add extra parameters
        prm = 0x1e, O, 4     # Signalling point code
   [EndMsg]

[EndISUPCodec]
```

Stack configuration file

```
include "rlt_example.cfg"
[SP]
  localPC = aaaaa
  ni = 2
  [ISUP]
    codecext = rlt
  [EndISUP]
  [MTP3]
    [DESTINATION]
      remotePC = bbbbb
    [EndDESTINATION]
  [EndMTP3]
[EndSP]
```

A.1.9 SCCP global Title Translation

This example illustrates SCCP global title translation. The first rule shows distributing a message to a local TCAP user. The second rule demonstrates changing routing on global title to routing on ssn and sending onto a remote point code. The third rule demonstrates translating from one global title to another global title.

Global title file gt_example.cfg

```
[GT_Table]
  name = exampleTable
  tt  = 11
  nai = 44
  np  = 6
  es  = bcd                  # binary coded decimal, match on es=1 and es=2
  [Rule]
    gtdigits = 01908273800
    [Route_SSN]
      ssn = 27
      destination = aaaaa    # send to local TCAP user with ssn = 27
    [EndRoute_SSN]
  [EndRule]
  [Rule]
    gtdigits = 01908273801
    [Route_SSN]              # change routing to route on ssn
      ssn = 123              # change ssn to 123
      destination = bbbbb    # send to RemotePC bbbbb
    [EndRoute_SSN]
  [EndRule]
  [Rule]
    gtdigits = 01908273802
    [Route_GT]
      gtformat = ?----800*   # translate digits to 0800273802
      tt  = 23               # change tt to 23, np and es remain unchanged
      nai = none             # remove nai, changing gt indicator from 4 to 3
      ssn = 143              # change ssn to 143
      destination = bbbbb    # send to RemotePC bbbbb
    [EndRoute_GT]
  [EndRule]
[EndGT_Table]
```

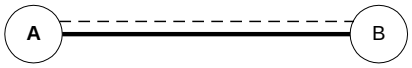
Stack configuration file

```
include "gt_example.cfg"
[SP]
  localPC = aaaaa
  ni = 2
  [TCAP]
  [EndTCAP]
  [SCCP]
    gt_table = exampleTable
  [EndSCCP]
  [MTP3]
    [DESTINATION]
      remotePC = bbbbb
    [EndDESTINATION]
  [EndMTP3]
[EndSP]
```

A.1.10 Sigtran M3UA Peer-to-peer

This example demonstrates how to configure M3UA nodes for peer-to-peer connection. In a peer-to-peer configuration, there are no SS7 signalling links defined. All transfer of messages is done using M3UA over SCTP/IP.

TCAP signalling is shown in this example, however ISUP may also be used.



Stack configuration file at A, IPSP Client node

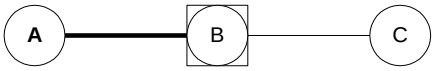
```
[SP]
  localPC = aaaaa
  ni = 2
  [TCAP]
  [EndTCAP]
  [SCCP]
  [EndSCCP]
  [M3UA]
    [ROUTING_KEY]
      routing_context = 38
      [ADDRESS]
        remotePC = bbbbb          # Point code of node B
        si = 3                    # Allow only SCCP traffic
      [EndAddress]
    [EndROUTING_KEY]
    [IPSP_CLIENT]                 # Client IPSP node
      name = clientA
      host = bbb.bbb.bbb.bbb      # or hostname of IPSP server at node B
      ASP_id = xxxxx
      routing_key = 38            # From a previously defined routing context
    [EndIPSP_CLIENT]
  [EndM3UA]
[EndSP]
```

Stack configuration file at B, IPSP Server node

```
[SP]
  localPC = bbbbb
  ni = 2
  [TCAP]
  [EndTCAP]
  [SCCP]
  [EndSCCP]
  [M3UA]
    [ROUTING_KEY]
      routing_context = 97
      [ADDRESS]
        remotePC = aaaaa          # Point code of node A
        si = 3                    # Allow only SCCP traffic
      [EndAddress]
    [EndROUTING_KEY]
    [IPSP_SERVER]                 # Server IPSP node
      name = serverB
      ASP_id = yyyyy
      routing_key = 97            # From a previously defined routing context
    [EndIPSP_SERVER]
  [EndM3UA]
[EndSP]
```

A.1.11 Sigtran M3UA Application Server and Signalling Gateway

In this example an application server is shown using a signalling gateway to a reach a remote node in a traditional SS7 signalling network.



Stack configuration file at A, Application Server

```
[SP]
  localPC = aaaaa      # Local point code must be same as the gateway B
  ni = 2
  [TCAP]
  [EndTCAP]
  [SCCP]
  [EndSCCP]
  [M3UA]
    [ROUTING_KEY]
      routing_context = 38
      [ADDRESS]
        remotePC = ccccc        # Point code at C in the ss7 network
      [EndAddress]
    [EndROUTING_KEY]
    [CLIENT]                    # Client application server node
      name = client
      host = bbb.bbb.bbb.bbb    # or hostname of gateway node B
      ASP_id = xxxxx
      routing_key = 38          # From a previously defined routing context
      rfc3332_compat = y        # Interwork with earlier versions of M3UA
    [EndCLIENT]
  [EndM3UA]
[EndSP]
```

Stack configuration file at B, Signalling Gateway

```
[SP]
  localPC = aaaaa       # Local point code must be same as the client A
  ni = 2
  [M3UA]
    [ROUTING_KEY]
      routing_context = 97
      [ADDRESS]
        remotePC = ccccc        # Point code of node C
      [EndAddress]
    [EndROUTING_KEY]
    [SERVER]                    # Server gateway node
      name = gateway
      ASP_id = yyyyy
      routing_key = 97          # From a previously defined routing context
    [EndSERVER]
  [EndM3UA]
  [MTP3]
    [DESTINATION]
      remotePC = ccccc          # Point code at C in the ss7 network
    [EndDESTINATION]
  [EndMTP3]
[EndSP]
```

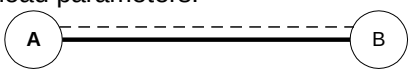
Firmware parameters for MTP3 at B, Signalling Gateway

```
fwdspldr ssssss 0 ss7.pmx -cOPCaaaaa -cDPCccccc -cTS16 -cSLC0
```

A.1.12 M2PA signalling link to a single network destination.

This example shows a single M2PA signalling link to destination system.

M2PA signalling link can be used in any of the earlier examples to replace the MTP2 links defined by the firmware download parameters.



Stack configuration file

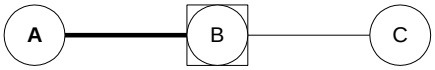
```
[SP]
  localPC = aaaaa
  ni = 2
  [TCAP]
  [EndTCAP]
  [SCCP]
  [EndSCCP]
  [MTP3]
    [DESTINATION]
      RemotePC = bbbbb
      [M2PA]
        slc = 0
        host=hostname[,nnn.nnn.nnn.nnn]
      [EndM2PA]
    [EndDESTINATION]
  [EndMTP3]
[EndSP]
```

A similar file with `listen=y` can be used for the destination system.

A.2 Q.782 configuration examples

A.2.1 Q.782 test configuration A.

This example is based upon `Test Configuration A` as described in ITU-T Q.782.



Stack configuration file

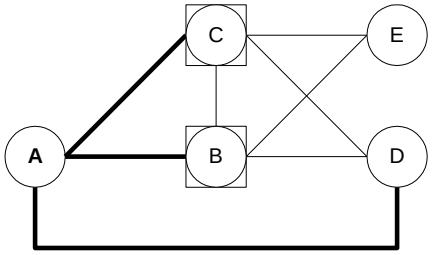
```
[SP]
  localPC = aaaaa
  ni = 2
  [MTP3]
    restart=ITU_White

    [DESTINATION]
      # SP B from Q.782
      remotePC = bbbbb

      # Note, since we declared at least one direct link to B,
      # A route from A->B is set up automatically...
    [EndDESTINATION]

    [DESTINATION]
      # SP C from Q.782
      remotePC = ccccc
      [ROUTE]
        # Indirect route from A -> C via B
        adjacentPC = bbbbb
        priority = 0
      [EndROUTE]
    [EndDESTINATION]
  [EndMTP3]
[EndSP]
```

Firmware parameters

```
fwdspldr ssssss 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC0
fwdspldr ssssss 1 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC1
fwdspldr ssssss 2 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC2
fwdspldr ssssss 3 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC3
```

A.2.2 Q.782 test configuration B.

This example is based upon `Test Configuration B` as described in ITU-T Q.782.



Stack configuration file

```
[SP]
  # SP A from Q.782
  localpc = aaaaa
  ni = 2
  [MTP3]
    restart=ITU_White

    [DESTINATION]

      # SP B from Q.782
      remotePC = bbbbb

      [ROUTE]
        # Direct route from A -> B
        # This one is always used if available, so priority is 0
        # This route would in fact be created by
        # default but is included for clarity...
        adjacentPC = bbbbb
        priority = 0
      [EndROUTE]

      [ROUTE]
        # Indirect route from A -> B via C
        # Used only if direct route unavailable, so priority is 1
        adjacentPC = ccccc
        priority = 1
      [EndROUTE]

      # End SP B
    [EndDESTINATION]

    [DESTINATION]
      # SP C from Q.782
      RemotePC = ccccc

      [ROUTE]
        # Direct route from A -> C
        # This one is always used if available, so priority is 0
        # This route would in fact be created by
        # default but is included for clarity...
        adjacentPC = ccccc
        priority = 0
      [EndROUTE]

      [ROUTE]
        # Indirect route from A -> C via B
        # used only if direct route unavailable, so priority is 1
        adjacentPC = bbbbb
        priority = 1
      [EndROUTE]

      # End SP C
    [EndDESTINATION]
```

```
    [DESTINATION]
      # SP D from Q.782
      remotePC = ddddd

      [ROUTE]
        # Direct route from A -> D
        # All routes to D load-share, so priority is 0
        # This route would in fact be created by
        # default but is included for clarity...
        adjacentPC = ddddd
        priority = 0
      [EndROUTE]

      [ROUTE]
        # Indirect route from A -> D via B
        # All routes to D load-share, so priority is 0
        adjacentPC = bbbbb
        priority = 0
      [EndROUTE]

      [ROUTE]
        # Indirect route from A -> D via C
        # All routes to D load-share, so priority is 0
        adjacentPC = ccccc
        priority = 0
      [EndROUTE]
    [EndDESTINATION]

    [DESTINATION]
      # SP E from Q.782
      RemotePC = eeeee

      [ROUTE]
        # Indirect route from A -> E via B
        # All routes to D load-share, so priority is 0
        adjacentPC = bbbbb
        priority = 0
      [EndROUTE]

      [ROUTE]
        # Indirect route from A -> E via C
        # All routes to D load-share, so priority is 0
        adjacentPC = ccccc
        priority = 0
      [EndROUTE]
    [EndDESTINATION]
  [EndMTP3]
[EndSP]
```

Firmware parameters

```
fwdspldr ssssss 0 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC0
fwdspldr ssssss 1 ss7.pmx -cOPCaaaaa -cDPCbbbbb -cTS16 -cSLC1
fwdspldr ssssss 2 ss7.pmx -cOPCaaaaa -cDPCccccc -cTS16 -cSLC0
fwdspldr ssssss 3 ss7.pmx -cOPCaaaaa -cDPCccccc -cTS16 -cSLC1
fwdspldr ssssss 4 ss7.pmx -cOPCaaaaa -cDPCddddd -cTS16 -cSLC0
fwdspldr ssssss 5 ss7.pmx -cOPCaaaaa -cDPCddddd -cTS16 -cSLC1
```

Appendix B: Local point code instances

There are a very few circumstances where it is necessary to configure two local signalling points with the same SS7 point code. For example a system gatewaying between two networks where both network providers happened to issue the same point code to the Aculab system (extremely unlikely). It is also necessary when testing Distributed ISUP, Dual MTP3 and some M3UA configurations on a single system.

Such systems can be configured by using local point code instances to differentiate between the signalling endpoints.

The local point code instance is configured by specifying **`instance=`*`n`*** in the `[SP]` section of the stack configuration file. ***`n`*** should be a small integer, the default being zero. The instance must also be specified on any `fwdspldr` and `ss7maint` commands that reference the point code, this is done by adding ***`,n`*** after the point code number, e.g: **`-cOPC2020,1`** or **`-o2020,1`**.

Note **To avoid confusion these parameters are not shown elsewhere in this document.**

Output from `ss7maint` usually displays the instance number in parenthesis following the point code number, normally suppressing the value when it is zero.

The messages sent over the TCP/IP connections when establishing the dual link, by TCAP (or SCCP) applications, or by distributed ISUP systems do not contain any information about point code instances. Specifying different passwords will allow these connections to connect in the desired manner.

Appendix C: SCTP Copyright Notices

The SCTP code in the windows driver is subject to the copyrights below.

C.1 The FreeBSD Copyright

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1) Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE FREEBSD PROJECT “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE FREEBSD PROJECT OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The views and conclusions contained in the software and documentation are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the FreeBSD Project.

C.2 Cisco Systems Copyright

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

a) Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
b) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
c) Neither the name of Cisco Systems, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN, CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

C.3 The Wide Project Copyright

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

a) Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
b) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
c) Neither the name of the project nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE PROJECT AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE PROJECT OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

C.4 RSA Data Security Copyright

License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

Contact us

Phone

+44 (0)1908 273800 (UK)
+1(781) 352 3550 (USA)

Email

Info@aculab.com

Sales@aculab.com

Support@aculab.com

Socials

Certificate number IS 722024
ISO 27001:2013

Certificate number FS722030
ISO 9001:2015