SIGTRAN UA Data Model

Published by:
OpenSS7 Corporation
Suite #424
4701 Preston Park Boulevard
Plano, TX 75093

NOTICE

OpenSS7 Corporation is making this documentation available as a reference point for the industry. While OpenSS7 Corporation believes that these interfaces are well defined in this release of the document, minor changes may be made prior to products conforming to the interfaces being made available.

TRADEMARKS

Abstract

This document provides a description and documentation of the OpenSS7 SIGTRAN Data Model that is used in the OpenSS7 SIGTRAN stack. It provides a general purpose data model that permits a single device to represent multiple Signalling Gateway Processes (SGPs) and multiple Application Server Processes (ASPs) in a single implementation.

Preface

Abstract

This document specifies a Signalling Link Manager for the SS7 Signalling Links supporting the OpenSS7 Signalling Link Interface. It provides abstraction from device-specific Physical Point of Attachment addresses and provides a single Signalling Link Identifier management domain for the identification, allocation and activation of signalling links.

Intent

This document is intended to provide information for writers of OpenSS7 Signalling Link Interface drivers as well as writers of OpenSS7 Signalling Link Users. It also provides documentation for the Signalling Link Manager and the Signalling Link Management Daemon.

Target Audience

The target audience is developers and users of the OpenSS7 SS7 stack.

Disclaimer

Although the author has attempted to ensure that the information in this document is complete and correct, neither the Author nor OpenSS7 Corporation will take any responsibility in it.

Revision History

Take care that you are working with a current version of this document: you will not be informed of updates. For a current version, please see the source documentation at http://www.openss7.org/.

$Log: datamodel.me,v $
Revision 0.8.2.1 2003/02/23 00:52:29 brian
Some idref and bullet changes.

Revision 0.8 2002/04/02 08:10:53 brian
Started Linux 2.4 development branch.

Revision 0.7.8.1 2002/01/11 00:57:10 brian
Checkin.

Revision 0.7 2001/12/20 21:14:09 brian
Working on datamodel document.

Revision 0.7.8.1 2001/12/11 13:14:59 brian
Branched for new development.

Revision 0.7.6.1 2001/12/11 11:06:00 brian
Corrected PS output.

Revision 0.7 2001/12/11 03:22:53 brian
Added doc files.

CHAPTER 1

Introduction

1.1. Overview

Here is a reference[M2PA03].

This is a footnote[1].

This is an endnote[1]. This is another endnote[2].

1.2. Document Organization

1.3. Terminology

Signalling Gateway Process (SGP)
: A signalling gateway process is a process instance acting as a transport endpoint and provides the functions of a signalling gateway.

Application Server Process (ASP)
: An application server process is a process instance acting as a transport endpoint that obtains services from a remote process of a signalling gateway and provides those services to a local instance of an application server.

Endnotes for Chapter 1

[1] This is an endnote.

[2] This is an endnote.

[1] This is a footnote.

CHAPTER 2

Data Model

2.1. Overview

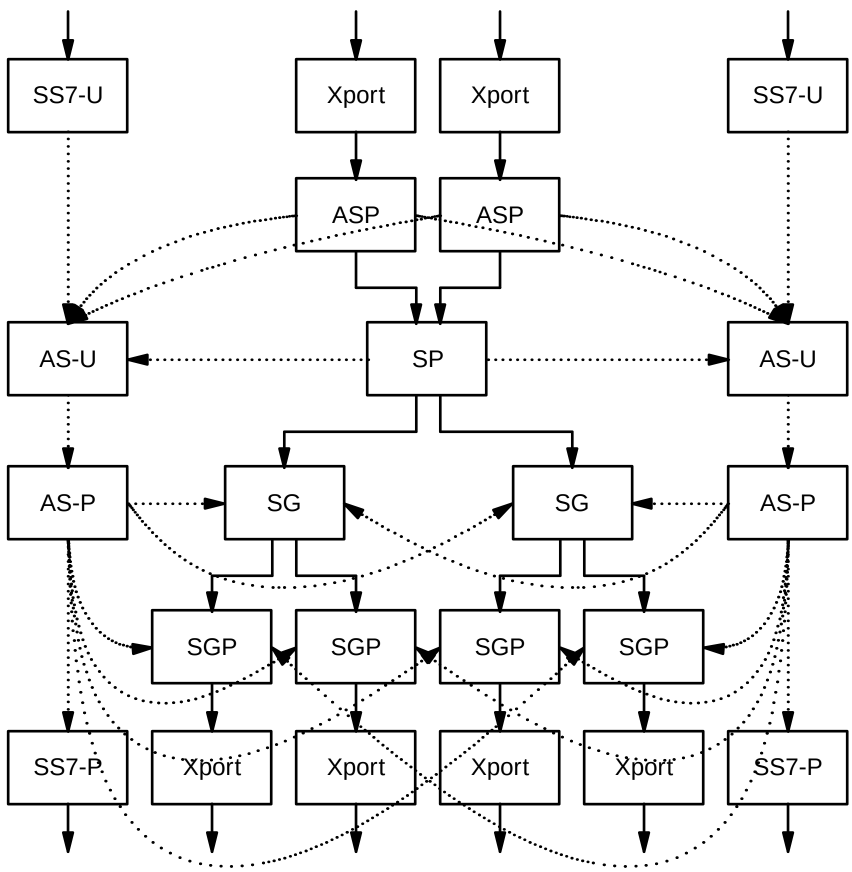

Figure 2-1. Overview

Figure 2-1 illustrates the general data model for SIGTRAN protocols. The main characteristics of the model are as follows:

SS7-U Each SS7 User is an opened STREAM. SS7 Users are direct users of an Application Server (AS). Although an Application Server is not necessarily associated with a local SS7 User, each local SS7 User is associated with one and only one Application Server (AS).

SS7-P Each SS7 Provider is a linked STREAM. SS7 Providers are direct providers of services to an Application Server (AS). Although an Application Server is not necessarily associated with a local SS7 Provider, each local SS7 Provider is associated with one and only one Application Server.

XPort Each transport endpoint is a linked STREAM and is connected to either an ASP or an SGP. On the local end of the STREAM is a transport data structure which contains information about the linked stream.

SGP Each transport endpoint connected to an SGP is represented by an SGP data structure. This data structure acts as a proxy for the connected SGP.

ASP Each transport endpoint connected to an ASP is represented by an ASP data structure. This data structure acts as a proxy for the connected ASP.

SP Each local Signalling Process (SP) has an SP data structure. This permits a single host to represent multiple ASP or SGP. Each local Signalling Process (SP) is either an ASP or an SGP or both (an SPP), but there is only one SP data structure per Signalling Process.

SG A group of transport endpoints connected to SGP that make up the same Signalling Gateway (SG) is represented by an SG structure.

AS An application server (AS) is an instance of a route (link) key and an AS data structure is associated with each route (link) key. Application Servers (AS) can provide service locally to an SS7 User (SS7-U) or provide service remotely to a number of ASP; or obtain service locally from an SS7 Provider (SS7-P) or obtain service remotely from a number of SGP.

2.1.1. XPort to ASP/SGP Relationship

There is a direct relationship between transport endpoint structures and ASP or SGP structures.

A transport endpoint can be associated either with ASP structures or with SGP structures (but not both).[1] This is because the process at the remote end of the transport connection is either an ASP or an SGP but not both. A transport endpoint can be associated with more than one ASP or SGP structure, but, only one ASP or SGP structure can be associated with a transport endpoint for a given protocol level. So, for example, a given transport endpoint might be associated with 5 ASPs, one for M2UA, one for M3UA, one for ISUA, one for SUA and one for TUA.

There is a one-to-many relationship between a transport endpoint and ASP (SGP).[2]

2.1.2. SG to SGP Relationship

An SG data structure can be associated with more than one SGP data structure; however, the associated SG data structures must be for the same protocol level as the SG with which they are associated. An SGP can only be associated with one and only one SG data structure.

The SG represents the remote SG instance which is providing the SGPs represented by the associated SGP data structures.

There is a one-to-many relationship between SG and SGP.

2.1.3. SP to ASP Relationship

An SP data structure can be associated with more than one ASP data structure; however, the associated ASP data structure must be for the same protocol level as the SP with which they are associated. An ASP can only be associated with one and only one SP data structure.

The SP represents a local SGP instance which is serving the ASPs represented by the associated ASP data structures.

There is a one-to-many relationship between SP and ASP.

2.1.4. SP to SG Relationship

The SP represents a local ASP instance which is obtaining service from the SGs represented by the associated SG data structures. An SP can be associated with more than one SG, and an SG can be associated with more than one SP. The relationship contains the ASP Identifier that is used by the SP when communicating with the associated SG.

There is a many-to-many relationship between SP and SG.

---

[1] There is also the concept of an SPP which acts as both ASP and SGP or IPSP in the SIGTRAN model, but this has not yet been implemented.

[2] Current implementation provides only for a one-to-one relationship. This is not a serious limitation, because multiplexing multiple UAs over a single SCTP association using Payload Protocol Identifier is not common.

2.1.5. AS Relationships

2.1.5.1. AS to ASP Relationship

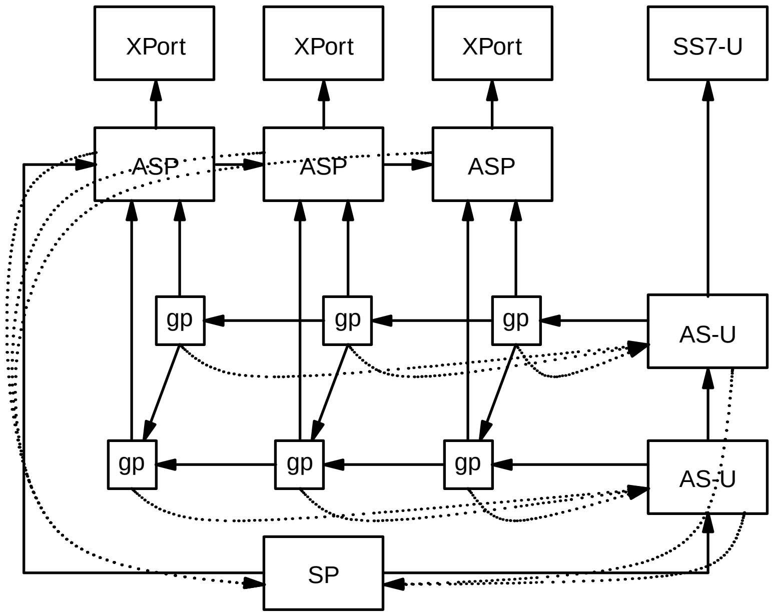

Figure 2-2. AS to ASP Relationship

Figure 2-2 illustrates the AS to ASP relationship.

An AS structure is related to an ASP structure in that the AS structure represents the AS at the remote ASP (or local SS7 User) that is being served by the local SGP (SP).

The AS structure contains the state of the AS at the local SGP (down, pending, inactive, activating, deactivating, active).

The GP (graph) structures between the AS and ASP structures contain the state of the AS for the given remote ASP (down, inactive, activating, deactivating, active).

The ASP structure contains the state of the remote ASP at the local SGP (down, coming up, going down, up).

There is a many-to-many relationship between AS and ASP.

2.1.5.2. AS to SGP Relationship

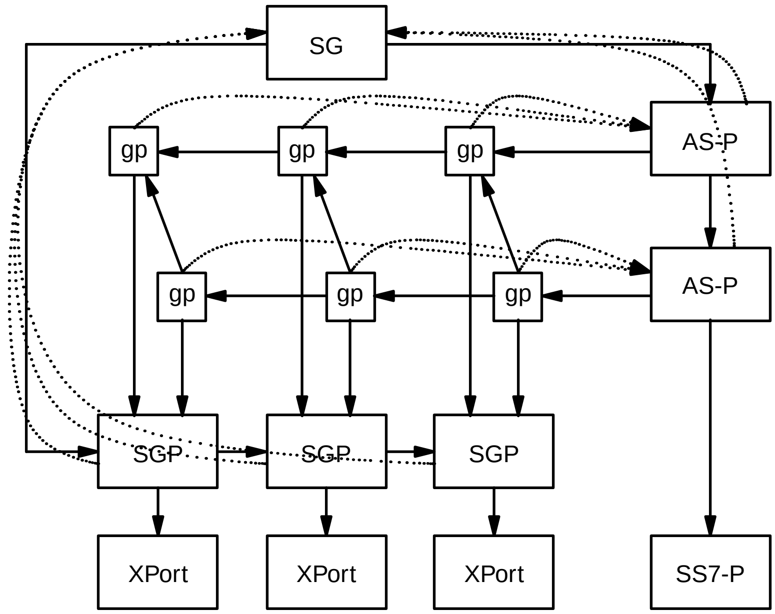

Figure 2-3. AS to SGP Relationship

Figure 2-3 illustrates the AS to SGP relationship.

An AS structure is related to an SGP structure in that the AS structure represents the AS at the local ASP (SP) that is being served by the SGP (or local SS7 Provider).

The AS structure contains the state of the AS at the local ASP (down, inactive, activating, deactivating, active).

The GP (graph)s structures between the AS and SGP structures contain the state of the AS for the given remote SGP (down, inactive, activating, deactivating, active).

The SGP structure contains the state of the remote SGP at the local ASP (down, coming up, going down, up).

There is a many-to-many relationship between AS and SGP.

2.1.5.3. AS to SP Relationship

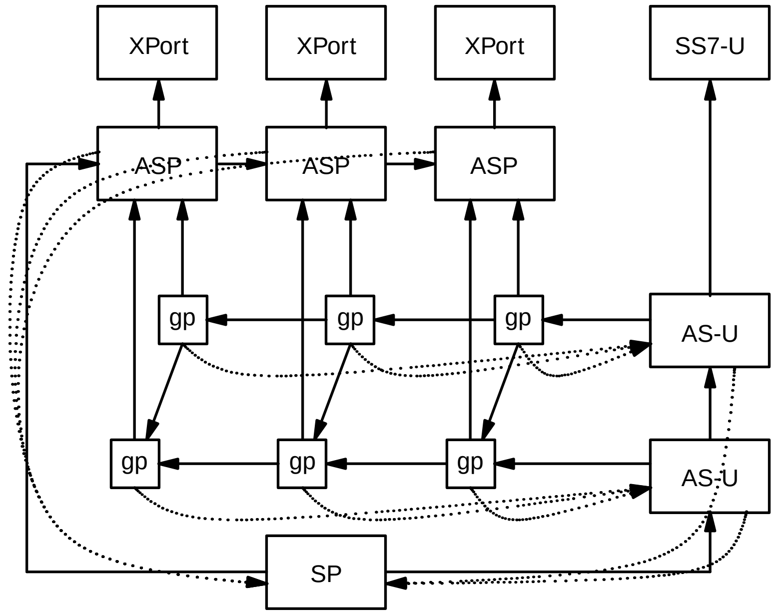

Figure 2-4. AS to SP Relationship

Figure 2-4 illustrates the AS to SP relationship.

An AS structure is related to an SP structure in that the AS structure represents the AS that is being served by the local SGP (SP). **FIXME: more description here....**

An AS can be associated with one and only one SP; however, and SP can be associated with more than one AS. It can be viewed that an SP has a list of all the AS with which it is associated.

There is a one-to-many relationship between SP and AS.

2.1.5.4. AS to SG Relationship

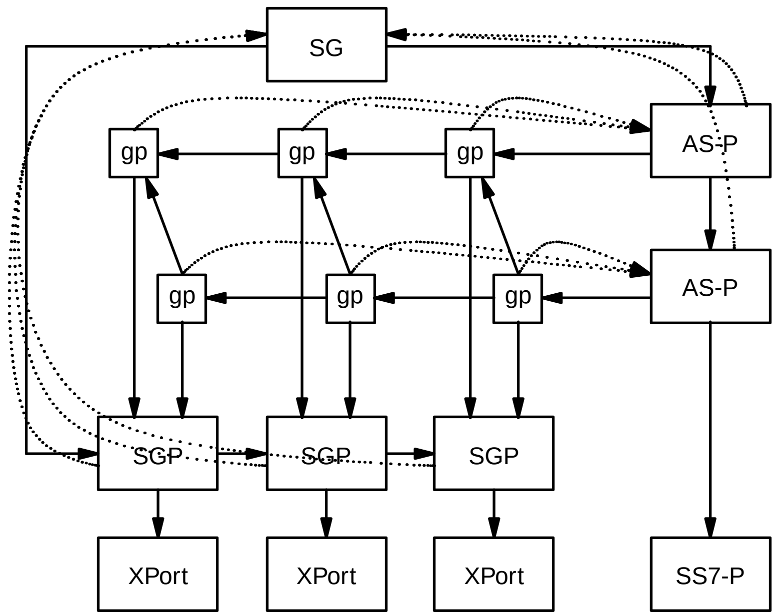

Figure 2-5. AS to SG Relationship

Figure 2-5 illustrates the AS to SG relationship.

An AS structure is related to an SG structure in that the AS structure represents the AS at the local ASP (SP) that is being served by the SG. It contains the Routing Context (RC) or Interface Identifier (IID) that is used by the AS for the associated SG. Where the AS is served by multiple SGs, the relationship also contains the routing tables that describe the accessibility of SS7 destinations via the associated SG.

There is a many-to-many relationship between SG and AS.

CHAPTER 3

Data Structure Life-Cycle

3.1. Creation

This section discusses the creation of UA data structures.

3.1.1. Boot-Time Configuration

3.1.2. Layer Management Configuration

3.1.2.1. Static Configuration

Static configuration is performed by Layer Management in advance of any run-time event.

The Layer Manger has the following commands at its disposal.

3.1.2.1.1. GET/ADD/CHA/DEL an SP

An SP data structure is examined, added, changed or deleted using the UA_IOCGCONFIG, UA_IOCSCONFIG, UA_IOCTCONFIG and UA_IOCCCONFIG controls using the object type UA_OBJ_TYPE_SP.

The argument to the control is a SP configuration structure as follows:

```
typedef struct ua_config {
        ulong type;     /* object type */
        ulong id;       /* object id */
        ulong cmd;      /* object command */
        /* followed by object-specific config structure */
} ua_config_t;

typedef struct ua_conf_sp {
        struct spid;    /* paired signalling process identifier */
        ulong cost;     /* cost */
        ulong tmode;    /* traffic mode */
} ua_conf_sp_t;
```

The configuration structure contains the following fields:

type The object type. For SP objects, the type is always UA_OBJ_TYPE_SP.

id The identifier for the object. The identifier is returned from the UA_ADD command and is provided to the UA_CHA and UA_DEL commands. The caller may select a unique identifier of its own when performing the UA_ADD command. Otherwise, the caller sets this value to zero(0) and the driver will assign a unique identifier and return it in the return argument from the UA_ADD command.

cmd The command can be one of the following:

- UA_ADD This command requests the UA driver to create the specified object and return any driver-selected identifiers.
- UA_CHA This command requests the UA driver to change the cost or traffic handling mode associated with the specified SP structure.
- UA_DEL This command request the UA driver to delete the specified object.

spid This is the identifier of the UA_OBJ_TYPE_SG object with which this SP is to be associated. This identifier can be coded zero to indicate that no existing SG objects are associated with this SP object.

cost The cost of the SP to the associated SGs. Where AS-Ps have multiple AS-Us to which they provide service, this cost value helps the driver to determine which AS-U will be used to reach the SS7 User. This is the default cost that will be assigned to an AS-U when it is dynamically created.

tmode When routing messages from the AS-U to remote ASPs, the traffic handling mode of the AS-U is used to determine which of a number of active ASPs will be selected to receive the traffic. This is the default traffic handling mode that will be assigned to an AS-U when it is dynamically created.

SP data structures do not have to be associated with any other data structure at creation. When an SP data structures are created management blocked for enabling and activation. This means that no AS-U or ASP later associated with the SP will have an enabling or activation procedure initiated on them until the SP is unblocked.

Deleting an SP data structure without force (UA_IOCSCONFIG) requires that all associated data structures have been previously deleted. Deleting an SP data structure with force (UA_IOCCCONFIG) will delete all other data structures associated with the SP data structure being deleted.

3.1.2.1.2. GET/ADD/CHA/DEL an SG

An SG data structure is examined, added, changed or deleted using the UA_IOCGCONFIG, UA_IOCSCONFIG, UA_IOCTCONFIG and UA_IOCCCONFIG controls using the object type UA_OBJ_TYPE_SG.

The argument to the control is a SP configuration structure as follows:

```
typedef struct ua_config {
        ulong type;     /* object type */
        ulong id;       /* object id */
        ulong cmd;      /* object command */
        /* followed by object-specific config structure */
} ua_config_t;

typedef struct ua_conf_sp {
        struct spid;    /* paired signalling process identifier */
        ulong cost;     /* cost */
        ulong tmode;    /* traffic mode */
} ua_conf_sp_t;
```

SG data structures must be associated with an existing SP data structure at creation time. SG data structures are created management blocked for enabling and activation. This means that no AS-P or SGP later associated with the SG will have an enabling or activation procedure initiated on them until the SG is unblocked.

Deleting an SG data structure without force (UA_IOCSCONFIG) requires that all associated data structures (other than the SP data structure) have been previously deleted. Deleting an SG data structure with force (UA_IOCCCONFIG) will delete all other data structures associated with the SG data structure (except the SP data structure).

3.1.2.1.3. GET/ADD/CHA/DEL an ASP

An ASP data structure is examined, added, changed or deleted using the UA_IOCGCONFIG, UA_IOCSCONFIG, UA_IOCTCONFIG and UA_IOCCCONFIG controls using the object type UA_OBJ_TYPE_ASP.

The argument to the control is a SPP configuration structure as follows:

```
typedef struct ua_config {
        ulong type;     /* object type */
        ulong id;       /* object id */
        ulong cmd;      /* object command */
        /* followed by object-specific config structure */
} ua_config_t;

typedef struct ua_conf_spp {
        struct spid;    /* signalling process identifier */
        ulong aspid;    /* ASP Id */
        ulong cost;     /* cost */
} ua_conf_spp_t;
```

ASP data structures must be associated with an existing SP data structure at creation time. When an ASN data structure is created it is created in the ASP_DOWN state and is management blocked for enabling. This means that no automatic procedure will be initiated on the ASP to move it from the ASP_DOWN state until it is unblocked. Also, any receive ASPUP messages will be refused due to management blocking.

Deleting an ASP data structure without force (UA_IOCSCONFIG) requires that all associated data structure (other than the SP data structure) have been previously deleted. Deleting an ASP data structure with force (UA_IOCCCONFIG) will delete all other data structures associated with the ASP data structure (except the SP data structure).

3.1.2.1.4. GET/ADD/CHA/DEL an SGP

An SGP data structure is examined, added, changed or deleted using the UA_IOCGCONFIG, UA_IOCSCONFIG, UA_IOCTCONFIG and UA_IOCCCONFIG controls using the object type UA_OBJ_TYPE_SGP.

The argument to the control is a SPP configuration structure as follows:

```
typedef struct ua_config {
        ulong type;     /* object type */
        ulong id;       /* object id */
        ulong cmd;      /* object command */
        /* followed by object-specific config structure */
} ua_config_t;

typedef struct ua_conf_spp {
        struct spid;    /* signalling process identifier */
        ulong aspid;    /* ASP Id */
        ulong cost;     /* cost */
} ua_conf_spp_t;
```

SGP data structures must be associated with an existing SG data structure at creation time. When an ASN or SGP data structure is created it is created in the ASP_DOWN state and is management blocked for enabling. This means that no automatic procedure will be initiated on the SGP to move it from the ASP_DOWN state until it is unblocked.

Deleting an SGP data structure without force (UA_IOCSCONFIG) requires that all associated data structure (other than the SG data structure) have been previously deleted. Deleting an SGP data structure with force (UA_IOCCCONFIG) will delete all other data structures associated with the SGP data structure (except the SG data structure).

3.1.2.1.5. GET/ADD/CHA/DEL an AS

An AS data structure is examined, added, changed or deleted using the UA_IOCGCONFIG, UA_IOCSCONFIG, UA_IOCTCONFIG and UA_IOCCCONFIG controls using the object type UA_OBJ_TYPE_AS_U or UA_OBJ_TYPE_AS_P.

The argument to the control is a AS configuration structure as follows:

```
typedef struct ua_config {
        ulong type;     /* object type */
        ulong id;       /* object id */
        ulong cmd;      /* object command */
        /* followed by object-specific config structure */
} ua_config_t;

typedef struct ua_conf_as {
        ulong spid;                             /* signalling process identifier */
        ulong iid;                              /* interface id */
        struct sl_addr add;                     /* signalling link address */
} ua_conf_as_t;
```

AS-U data structures must be associated with an existing SP data structure at creation time. AS-P data structures must be associated with an existing SG data structure at creation time. When an AS data structure is created it is created in the AS_DOWN state and is management blocked for activation. This means that no automatic procedure will be initiated on the AS-U or AS-P to move it from the AS_DOWN state until it is unblocked.

3.1.2.2. Dynamic Configuration

Dynamic configuration is performed in response to UA driver run-time events.

3.1.2.2.1. SS7 User Streams

When an SS7 User stream is opened, an SS7-U data structure is allocated for the stream. This data structure contains information about the stream including the device number.

After being opened, and whenever the stream is in an unbound (unattached) state, an SS7-U structure is not associated with an AS-U.

When an SS7 User stream is bound or attached, an AS data structure might be associate with the SS7 User stream. The list of existing AS data structures is searched for a match on the route (link) key based on the information from the bind (attach). Only the AS-U data structures associated with the Signalling Process (SP) indicated in the protocol address of the bind (attach) request are searched.

If an AS-U data structure exists whose Routing Key (Link Key) information matches the requested protocol address, the AS data structure will be associated with the SS7-U structure. However, if the AS data structure was already associated with an SS7 User, then the bind (attach) request fails with an error indicating that the protocol address is already bound (attached).

If no AS data structure exists for the SP data structure indicated by the protocol address in the bind (attach) request, and the SP does not support dynamic registration, the request will fail with an error indicating that there is no such protocol address.

If no AS data structure exists for the SP data structure indicated by the protocol address in the bind (attach) request, and the SP supports dynamic registration, a registration indication will be given to Layer Management identifying the SP for the request and the protocol address used in the request and a response timer is started. Layer Management is expected to issue a registration response that contains the SP data structure identifier and either indicates failure, or the identifier of the AS-U data structure which has been added to satisfy the request. If the response timer expires before an response is forthcoming from Layer Management, the bind (attach) request will be refused.

When an SS7 User connectionless stream is bound, or an SS7 User connection-oriented stream is connected, or an SS7 User link is enabled, the UA driver attempts to activate the AS-U. This results in the activation of the AS-P and the subsequent activation with local SS7-P or remote subtending SGP in each SG.

Should all of the active or inactive SS7 Providers and SGP supporting an SS7 User stream fail, an M_HANGUP will be delivered to the application on the stream. The application must close and reopen the stream or otherwise reset the stream to make it available again for use.

3.1.2.2.2. SS7 Provider Streams

SS7 Provider streams are I_LINK or I_PLINK'ed under the UA multiplexing driver in the newly opened state.

- For M2UA, the SS7 provider stream is an SL (Signalling Link) stream. newly opened state.
- For M3UA, the SS7 provider stream is an MTP (Message Transfer Part) stream, either connectionless or connection oriented.
- For SUA, the SS7 provider stream is an SCCP (Signalling Connection Control Part) stream.
- For TUA, the SS7 provider stream is a TR/TC (Transaction Handing or Component Handling) stream.
- For ISUA, the SS7 provider stream is an ISUP (ISDN Uer Part) CC (Call Control) stream.

After an SS7 Provider stream is linked, it needs to be associated with an AS-P data structure by Layer Management. Layer Management must create an AS data structure (if one does not yet exist) using the UA_ADD command to the UA_IOCSCONFIG control for object type UA_OBJ_TYPE_AS_P, and associate it with the linked SS7 Provider stream using the UA_ADD command to the UA_IOCSCONFIG control for object type UA_OBJ_TYPE_SS7_P.

Whenever an SS7 Provider stream is associated with an AS-P,[1] the SS7 Provider is considered to be ASP_UP for the AS-P. This moves the AS-P that has no associated SGP and is otherwise in the AS_DOWN state to the AS_INACTIVE state.

Whenever an SS7 Provider stream is disassociated with an AS-P,[2] the SS7 Provider is considered to be ASP_DOWN for the AS-P. This moves the AS-P that has no associated SGP to the AS_DOWN state.

Whenever an AS-U is activated and the corresponding AS-P needs to be activated, the SS7 Provider stream will be bound, connected, attached or enabled.

If an SS7 provider stream should receive an M_ERROR or M_HANGUP from below indicating that a fatal error has occurred on the stream, the SS7-P will be moved to the ASP_DOWN state for the AS-P and management will be informed. It is the

[1] An SS7 Provider stream is associated with an AS-P by Layer Management issuing the UA_IOCSCONFIG or UA_IOCCCONFIG control with the UA_ADD command on a UA_OBJ_TYPE_SS7_P object type.

[2] An SS7 Provider stream is disassociated with an AS-P by Layer Management issuing the UA_IOCSCONFIG or UA_IOCCCONFIG control with the UA_DEL command on the UA_OBJ_TYPE_SS7_P object type.

responsibility of management to then unlink, close, reopen and relink, or otherwise reset the stream.

3.1.2.2.3. Transport Streams

Transport streams are I_LINK or I_PLINK'ed under the UA multiplexing driver in the connected state.

After a transport stream is linked, it needs to be associated with an ASP or SGP structure by Layer Management. Layer management must create an ASP or SGP structure (if one does not yet exist) using the UA_ADD command to the UA_IOCSCONFIG control for object type UA_OBJ_TYPE_ASP or UA_OBJ_TYPE_SGP, and associate it with the linked Transport stream using the UA_ADD command to the UA_IOCSCONFIG control for object type UA_OBJ_TYPE_XP_SCTP, UA_OBJ_TYPE_XP_TCP or UA_OBJ_TYPE_XP_SSCOP.

3.1.2.2.4. SGP Transport Streams

When an SGP Transport stream is linked, it needs to be associated with an SGP data structure by Layer Management. Layer Management must create an SGP data structure (if one does not yet exist) and associate it with the linked Transport stream. The Layer Manager must also associate the SGP data structure with an SG data structure.

After the transport stream is associated with an SGP and SG data structure, Layer Management needs to initiate initiate the ASP Up procedure. LM initiates the ASP Up procedure by issuing the UA_IOCCMGMT control using the UA_ASP_UP command and the UA_OBJ_TYPE_SS7_P object type and identifier.

After successful completion of the ASP Up procedures, the UA driver will check whether there are any AS configured for the SGP and SG in question that are ACTIVE or PENDING. If there are active or pending AS, the UA driver will attempt to activate each such AS, where appropriate according to the traffic mode of the AS, for the newly available SGP.

Should the SGP transport stream receive an M_ERROR or M_HANGUP, the SGP failure will be reported to management. It is the responsibility of management to unlink the SGP transport stream, close, reopen and relink the stream.

Should the SGP transport stream abort (receive a T_DISCON_IND) then the SGP failure will be reported to management. In this case, it is the responsibility of management to unlink, reconnect, and relink the stream. This may also be accomplished using the UA_IOCCPASS controls to pass IOCTL controls to the linked stream. The ASP Up procedure will also have to be reinvoked on the stream.

Should the SGP transport stream receive an unsolicited ASP Down Ack indicating that the SGP has gone down, the driver will make one attempt to invoke the ASP Up procedure, and if the attempt fails, will inform management that the ASP is in the ASP Down state due to SGP failure.

3.1.2.2.5. ASP Transport Streams

When an ASP Transport stream is linked, it needs to be associated with an ASP data structure by Layer Management. Layer Management must create an ASP data structure (if one does not yet exist) and associate it with the linked Transport stream. The layer manager may alternately associate the ASP data structure with an SP data structure.

After the transport stream is associated with an ASP or SP data structure, the UA driver will permit the ASP Up procedures to occur. Until this point, an ASP Up received on the transport stream will be queued.

In the case where an ASP transport stream is associated with an ASP data structure, the ASP Up procedure need not supply an ASP Id parameter, unless the ASP data structure does not contain an ASP Id.

In the case where an ASP transport stream is associated with an SP data structure, the ASP Up procedure must contain an ASP Id parameter. The ASP Id parameter received in the ASPUP request message will be used to:

(i) Identify an existing ASP data structure with the corresponding ASP Id.

(ii) Create a new ASP data structure with the corresponding ASP Id. The new ASP data structure is created, the ASP data structure will support the full set of AS provided by the SP.

After the transport stream has successfully completed the ASP Up procedures, the receipt of a REG REQ message will cause the UA driver to attempt to register the given route (link) key. Registration of the route (link) key will be performed by scanning the list of AS-U associated with the SP receiving the REG REQ. If an AS-U exists for the routing (link) key, the Interface Identifier or Routing Context associated with the existing AS-U will be returned in the REG RESP message.

If an AS-U does not exist for the routing (link) key and the SP supports dynamic registration, an AS-U structure will be created. In addition, AS-P structures for the SG providing service to the SP will also be created. This could trigger additional REG REQ messages to the SGP supporting the newly formed AS-P. Only once one of the AS-P have been successfully registered will the REG RESP message be sent to the registering ASP.

AS-U created in this fashion will be marked for dynamic registration so that when the last ASP de-registers for the AS, the AS-U and associated AS-P can be deregistered and deleted.

As a result of a successful REG REQ, the ASP data structure will be associated with an AS data structure representing the registered route (link) key.

The addition of or configuration of a new AS data structure may trigger the UA to generate further REG REQ messages to SGP (maybe peers) that are supporting the AS. The UA driver may wait for the responses from these subsequent REG REQ messages before replying to the initial REG REQ message.

After the transport stream is successfully configured or registered, the receipt of an ASPAC REQ message will cause the UA driver to attempt to activate the AS. If the AS is pending already active for a different ASP, the activation immediately succeeds. If the AS is inactive, the necessary steps to activate the AS are taken by the driver. This could include:

(i) Binding, connecting, attaching and enabling SS7 provider streams to provide service for the AS.

(ii) Activating the AS to remote SGP.

Only once the AS has been successfully activated will the ASPAC Ack be returned to the requesting remote ASP. Otherwise, the ASPIA Ack or MGMT ERR is returned instead. Management is informed of failures to activate remote ASPs for a given AS, so that management may take corrective action if the failure is due to misconfiguration of the system.

Should the ASP transport stream receive an M_ERROR or M_HANGUP, the SGP failure will be reported to management. It is the responsibility of management to unlink the ASP transport stream, close, reopened and relink the stream.

Should the ASP transport stream abort (receive a T_DISCON_IND) then the ASP failure will be reported to other ASPs using the AS with a MGMT NTFY message, and management will be informed. In this case, it is the responsibility of management to unlink the stream and await a new connection indication from the ASP.

Should the transport stream receive an ASPDN Request indicating that the ASP wishes to go down, the driver will information management and place the ASP in the ASP_DOWN state for all associated AS. The driver will await a new ASP Up Request to reestablish the ASP. No failure notification is given to ASP peers.

3.2. Destruction

This section discusses the destruction of UA data structures.

3.3. ASP Up and ASP Down Procedures

Enabling and disabling of ASPs can be performed automatically in some circumstances, or under the control of Layer Management. ASPs may also be management blocked from being enabled or disabled automatically.

3.3.1. Management Enabling and Disabling

Layer management can enable or disable a local or remote ASP at any time using the UA_ASPUP or UA_ASPDN command to the UA_IOCCMGMT control using one of the following object types and identifiers:

UA_OBJ_TYPE_SS7_U
: Only the UA_ASPDN command is valid for this object type. This will result in the sending of M_HANGUP to the specified local SS7-U stream.

UA_OBJ_TYPE_ASP
: Only the UA_ASPDN command is valid for this object type. This will result in the sending of an unsolicited ASPDN Ack to the remote ASP.

UA_OBJ_TYPE_SP
: The UA_ASPDN command will result in sending an unsolicited ASPDN Ack to all remote ASPs and a M_HANGUP to all local SS7-Us. Subtending local SS7-Ps will be deactivated, and remote SGPs will be sent the ASPDN Request to initiate the ASP Down procedures with the remote SGP.

  The UA_ASPUP command will result in subtending remote SGPs not in the ASP_UP or ASP_WACK_ASPUP state being sent the ASPUP Request to enable the local ASP.

UA_OBJ_TYPE_SG
: The UA_ASPDN command will invoke the ASP Down procedure for all SGP subtending the specified SG. This may also result in taking down the local ASP and sending unsolicited ASPDN Ack to remote ASPs should the SG be the only SG servicing the remote ASPs.

  The UA_ASPUP command will invoke the ASP Up procedure for all SGP subtending the specified SG.

UA_OBJ_TYPE_SGP
: The UA_ASPDN command will invoke the ASP Down procedure for the specified remote SGP. This may also result in taking down the local ASP and sending unsolicited ASPDN Ack or M_HANGUP to remote ASPs or local SS7-Us should the SGP be the only means of servicing the local ASP.

The UA_ASPUP command will result in the remote SGP not in the ASP_UP or ASP_WACK_ASPUP state being sent the ASPUP Request to initiate the ASP Up procedures with the remote SGP.

3.3.1.1. Automatic Blocking of Enabling

In some circumstances the driver must be automatically blocked from attempts to re-enable or re-disable an ASP. This occurs when the first automatic attempt to successfully complete the ASP Up procedure fails. The driver will inform management of the failure to enable the ASP and will block the ASP from further attempts at enabling.

3.3.2. Automatic Enabling and Disabling

3.3.2.1. Management Blocking of Enabling

Management can block a local ASP from attempts by the driver to automatically enable or re-enable the local ASP to all remote SG. This is performed using the UA_IOCCMGMT control with the UA_ASPUP_BLOCKING command and the UA_OBJ_TYPE_SP object type and identifier.

Management can also block a local ASP from attempts by the driver to automatically enable or re-enable the local ASP to a specific remote SG or remote SGP. This is performed using the UA_IOCMGMT control with the UA_ASPUP_BLOCKING command and the UA_OBJ_TYPE_SG or UA_OBJ_TYPE_SGP object type and identifier.

Management may also block a remote ASP from enabling or re-enabling the remote ASP to the local SGP. This is performed using the UA_IOCCMGMT control with the UA_ASPUP_BLOCKING command and the UA_OBJ_TYPE_ASP object type and identifier.

3.4. ASP Registration and Deregistration Procedures

Registration and deregistration can be performed automatically in some circumstances, or under the control of Layer Management.

3.4.1. Management Activation an Deactivation

3.4.1.1. Automatic Blocking of Activation

3.4.2. Automatic Activation and Deactivation

3.4.2.1. Management Blocking of Activation

Management can block a local AS-U from attempts by the driver to automatically activated or re-activate the local AS to all local SS7-P or remote SG. This is performed using the UA_IOCCMGMT control with the UA_ASPAC_BLOCKING command and the UA_OBJ_TYPE_AS_U object type and identifier. This also serves to block the AS-U from attempts by remote ASP to activate for the AS-U to the local SGP.

Management can block a local ASP from attempts by the driver to automatically activate or re-activate the local AS-U to a specific local SS7-P or remote SG or SGP. This is performed using the UA_IOCCMGNT control with the UA_ASPAC_BLOCKING command and the UA_OBJ_TYPE_SS7_P, UA_OBJ_TYPE_SG or UA_OBJ_TYPE_SGP object type and identifier. This also serves to block the AS-U from attempts by remote ASP to active fro the AS-U to the local SGP, when a resource necessary to activate is blocked.

Blocking has no effect on the current state of the object. For example, to prepare to take an AS-U out of service would require performing the UA_ASPAC_BLOCKING management control on the AS-U and then performing a UA_ASP_INACTIVE management control to deactivate the AS-U.

3.4.3. Management Registration and Deregistration

3.4.4. Automatic Registration and Deregistration

3.5. ASP Active and ASP Inactive Procedures

CHAPTER 4

Data

This chapter describes the individual data structures that make up the UA driver configuration database.

4.1. SP Data Structure

The SP data structure is as follows:

```
typedef struct sp {
        HEAD_DECLARATION (sp);                  /* head declaration */
        ulong cost;                             /* cost */
        ulong tmode;                            /* traffic mode */
        SLIST_COUNT (np);                       /* NP (SP graph) */
        SLIST_HEAD (as);                        /* AS list */
        SLIST_COUNT (spp);                      /* ASP list */
        m2ua_timers_sp_t timers;                /* timers */
        m2ua_opt_conf_sp_t config;              /* configuration */
} sp_t;
```

4.1.1. Per-AS Information

4.1.2. Per-SG Information

4.1.3. Per-XP-ASP Information

4.2. SG Data Structure

The SG data structure is shown below

```
typedef struct sp {
        HEAD_DECLARATION (sp);                  /* head declaration */
        ulong cost;                             /* cost */
        ulong tmode;                            /* traffic mode */
        SLIST_COUNT (np);                       /* NP (SP graph) */
        SLIST_HEAD (as);                        /* AS list */
        SLIST_COUNT (spp);                      /* ASP list */
        m2ua_timers_sp_t timers;                /* timers */
        m2ua_opt_conf_sp_t config;              /* configuration */
} sp_t;
```

The fields of this data structure are as follows:

next, prev
Linkage into master list of SP/SG data structures.

refcnt Structure reference count.

lock Structure lock.

id Identifier which uniquely identifies the SG data structure within the UA multiplexing driver. This identifier is used by Layer Management to identify the SG.

state Run-time state of the SG data structure. The SG data structure can be in one of the following states:

ASP_DOWN — All of the SGP in the SG are in the down state to the current SP.

ASP_WACK_ASPUP	All of the SGP in the SG are in the down state to the current SP, but the SP is activating for one or more of the SGP in the SG.
ASP_WACK_ASPDN	All of the SGP in the SG are in the down state, are coming up or going down, but the SP is going down for at least one SGP.
ASP_UP	All of the SGP in the SG are in the down state, are coming up or going down, or are in the up state, but at least one SGP in the SG is in the up state.

flags Run-time flags associated with the SG data structure.

type The type of the SG data structure is always UA_OBJ_TYPE_SG.

cost The cost of this SG amoung other SGs supporting the SP. SG data structures of the same lesser numerical cost will be used before SGs of higher numerical cost.

tmode The default traffic mode of the SG. This is the traffic handling mode for distributing traffic over the SGP that make up the SG. The traffic handling mode for the SG may be one of the following:

UA_TMODE_OVERRIDE	In the over-ride traffic mode, traffic will only be sent to one SGP within the SG at a time.
UA_TMODE_LOADSHARE	In the load-share traffic mode, traffic will be distributed over the available (active) SGP according to a load sharing scheme. Typically, SLS (Signalling Link Selection) value is used for load sharing.
UA_TMODE_BROADCAST	In the broadcast traffic mode, traffic will be sent to each of the available (active) SGP within the SG.

This is only the default traffic mode that will be inherited when new AS-P are created associated with this SG. The traffic mode of the AS-P determines how traffic associated with the AS will be distributed over SGP within the SG.

aspid The default ASP Identifier of the SP within this SG. This is the ASP Identifier for identifying the associated ASP (SP) to the SG. This is only the default ASP Identifier that will be inherited when new SGP are created and associated with this SG. The ASP Identifier in the SGP structure is the precise identifier that will be provided to the SGP in the ASPUP Request.

np.list, *np.numb*, *np.u.c.down*, *np.u.c.wack_aspup*, *np.u.c.wack_aspdn*, *np.u.c.up*
This is a graph (headed by *np.list* and containing *np.numb* nodes) that contains the list of SP data structures associated with this SG data structure. The counts (*np.u.c.down np.u.c.wack_aspup*, *np.u.c.wack_aspdn* and *np.u.c.up*) indicate the number of elements in the graph (SP structures) that are in the respective states: ASP_DOWN, ASP_WACK_ASPUP, ASP_WACK_ASPDN and ASP_UP.

as.list, as.numb
This is a list of the AS-P structures and the number of AS-P structures associated with this SG.

spp.list, spp.numb, spp.u.c.down, spp.u.c.wack_aspup, spp.u.c.wack_aspdn, spp.u.c.up
This is a list of the SGP structures and the number of SGP structures associated with this SG and the number of SGP in each state for the SG.

timers Run-timer timers associated with the SG.

config Configuration options for the SG.

4.2.1. SG Data Structure Relationships

The relationship between the SG data structure and other UA data structures is illustrated in *Figure 4-1*.

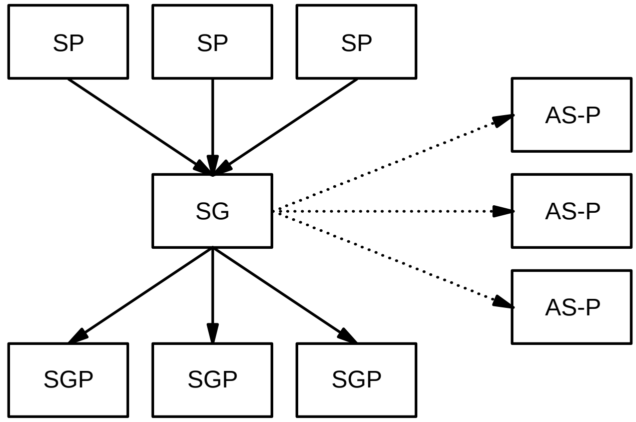

Figure 4-1. SG Data Structure Relationships

4.2.1.1. Per-AS Information

4.2.1.2. Per-SP Information

4.2.1.3. Per-XP Information

4.3. XP Data Structure

The XP data structure is as follows:

```
typedef struct xp {
} xp_t;
```

4.3.1. XP-ASP Data Structure

4.3.1.1. Per-AS Information

4.3.1.2. Per-SP Information

4.3.2. XP-SGP Data Structure

4.3.2.1. Per-AS Information

4.3.2.2. Per-SG Information

4.4. DP Data Structure

The DP data structure is as follows:

```
typedef struct xp {
} xp_t;
```

4.4.1. Per-AS Information

4.5. LP Data Structure

The LP data structure is as follows:

```
typedef struct lp {
} lp_t;
```

4.5.1. LP-ASP Data Structure

4.5.1.1. Per-XP-ASP Information

4.5.2. LP-SGP Data Structure

4.5.2.1. Per-XP-SGP Information

4.5.3. LP-SS7 Data Structure

4.5.3.1. Per-AS Information

4.6. AS Data Structure

The AS data structure is as follows:

```
typedef struct as {
        HEAD_DECLARATION (as);                  /* head delcaration */
        uint32_t rc;                            /* Routing Context */
        struct mtp_addr add;                    /* MTP address */
        SLIST_HEAD (sl);                        /* SL for this AS */
        SLIST_COUNT (ap);                       /* AP (AS graph) list */
        SLIST_LINKAGE (sp, as, sp);             /* SP linkage */
        SLIST_COUNT (gp);                       /* GP (SPP graph) list */
        m2ua_timers_as_t timers;                /* timers */
        m2ua_opt_conf_as_t config;              /* configuration */
} as_t;
```

The fields of this data structure are as follows:

next, prev
: Linkage into master list of AS data structures.

refcnt
: Structure reference count.

lock
: Structure lock.

id
: Identifier which uniquely identifies the AS data structure within the UA multiplexing driver. This identifier is used by Layer Management to identify the AS.

state
: Run-time state of the AS data structure. The AS data structure can be in one of the following states.

AS_DOWN
: All of the ASPs or SGPs associated with the AS are in the ASP_DOWN state.

AS_INACTIVE
: At least one of the ASPs or SGPs associated with the AS is in the ASP_UP or ASP_WACK_ASPDN state.

AS_WACK_ASPAC
: At least one of the ASPs or SGPs associated with the AS is in the ASP_UP or ASP_WACK_ASPDN state. At least one of the ASPs or SGPs is activating for the AS. None of the ASPs or SGPs is in the AS_ACTIVE or AS_WACK_ASPIA states.

AS_WACK_ASPIA
: At least one of the ASPs or SGPs associated with the AS is in the ASP_UP or ASP_WACK_ASPDN state. At least one of the ASPs or SGPs is deactivating for the AS. None of the ASPs or SGPs is in the AS_ACTIVE state.

AS_ACTIVE
: At least one of the ASPs or SGPs associated with the AS is in the ASP_UP or ASP_WACK_ASPDN state. At least one of the ASPs or SGPs is in the AS_ACTIVE state.

flags
: Run-time flags associated with the AS. The following flags are defined for the AS:

AS_PENDING
: When set, indicates that the AS is inactive (in the AS_DOWN or AS_INACTIVE or AS_WACK_ASPAC state), but the associated providers (local SS7-P or remote SGP) have not yet been deactivateed pending the arrival of a new activation request from the

local SS7-U or remote ASPs.

AS_ACTIVE
: When set, this flag indicates that the AS is in the AS_WACK_ASPIA or AS_ACTIVE states; i.e, that the AS is actively processing traffic.

  When clear, this flag indicates that the AS is in the AS_DOWN, AS_INACTIVE or AS_WACK_ASPAC states. This flag will be clear when the AS_PENDING flag is set.

type The type of the AS data structure is one of the following:

UA_OBJ_TYPE_AS_U
: The object is an AS-U data structure.

UA_OBJ_TYPE_AS_P
: The object is an AS-P data structure.

iid or rc
: The Interface Identifier or Routing Context for the AS.

add The address (Routing Key or Link Key) associated with the AS.

ss7.list, ss7.numb
: The list and number of corresponding SS7 Users or Providers associated with the AS-U or AS-P.

ap.list, ap.numb, ap.u.c.down, ap.u.c.inactive, ap.u.c.wack_aspac, ap.u.c.wack_aspia, ap.u.c.active
: The list and number of AP (graph) nodes and the counts of the node states for the various states. The graph contains the relationship between the AS and the AS of the other type. That is, the relationship between AS-U and AS-P. The node also contains the state of the AS-U to the AS-P.

sp.sp, sp.next, sp.prev
: The linkage of the AS-U to the SP or the AS-P to the SG.

gp.list, gp.numb, gp.u.c.down, gp.u.c.inactive, gp.u.c.wack_aspac, gp.u.c.wack_aspia, gp.u.c.active
: The list and number of GP (graph) nodes and the counts of the node states for the various states. The graph contains the relationship between the AS-U and ASPs using the AS, or AS-P and SGPs providing the AS, including the state of the ASP (or SGP) for the AS.

timers Run-time timers for the AS.

config Optional configuration for the AS.

4.6.1. Per-AS Information

Per-AS information is contained in the AP graph node. The AP graph node structure is as follows:

```
typedef struct ap {
        ulong state;                        /* state of this AS-U for this AS-P */
        SLIST_LINKAGE (as, ap, u);          /* AS-U linkage */
        SLIST_LINKAGE (as, ap, p);          /* AS-P linkage */
} ap_t;
```

This data structure contains the following fields:

state

u.as, u.next, u.prev

p.as, p.next, p.prev

4.6.2. Per-ASP/SGP Information

Per-ASP/SGP information is contained in the GP graph node. The GP graph node structure is as follows:

```
typedef struct gp {
        ulong state;                        /* AS state for this SPP */
        uint32_t iid;                       /* AS iid for this SPP */
        SLIST_LINKAGE (as, gp, as);         /* AS linkage */
        SLIST_LINKAGE (spp, gp, spp);       /* SPP linkage */
} gp_t;
```

This data structure contains the following fields:

state Run-time state of the ASP or SGP for the AS.

iid or rc
The Interface Identifier or Routing context used by the ASP or SGP to refer to the AS.

as.as, as.next, as.prev
the continuing list of AS that are served by or for the same SPP expressed as graph nodes.

spp.spp, spp.next, spp.prev
The continuing list of ASPs or SGPs that belong to the same AS expressed as graph nodes.

CHAPTER 5

Examples

5.1. ASP Examples

5.1.1. Simple ASP Example

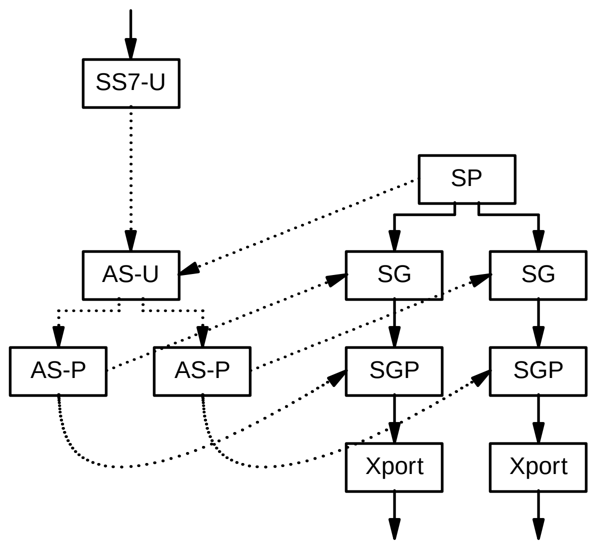

Figure 5-1. Simple ASP Example

Figure 5-1 illustrates a simple ASP example. In the example, a local SS7 User is obtaining services from two Signalling Gateways.

When the SS7 User STREAM is opened, and bound, a registration request is issued locally and either responded to by the Layer Manager or sent to the SGP associated with the configured SG and an AS Structure is created.

Per-AS structures for the SP, SG, and SGP data structures are created. The UA driver attempts to move the AS to the registered state for all configured SG and to the active state for all configured SGP. To complete this, it performs whatever registrations and activations are necessary on behalf of the AS.

When active, data traffic from the SS7 User is distributed over the available SG and SGP according to traffic mode and load sharing algorithms.

Data traffic received at the SGP XPort has the SGP selected on the basis of protocol payload identifier. The SGP then looks up the AS structure based on the RC/IID in the data message. The AS then passes the received data directly to the SS7 User.

5.2. SG Examples

5.2.1. Simple SGP Example

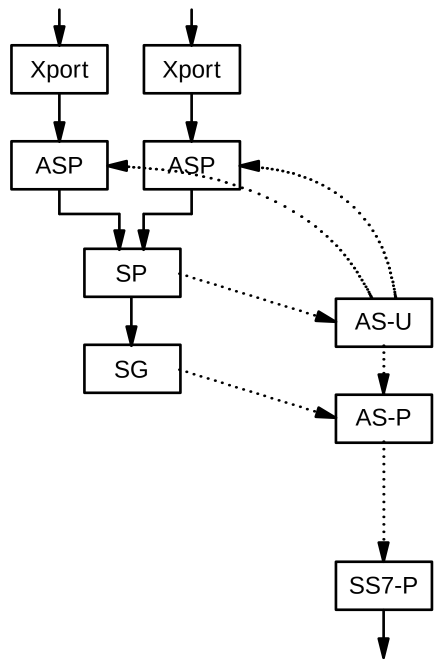

Figure 5-2. Simple SGP Example

Figure 5-2 illustrates a simple SGP example. In the example, a local SS7 Provider is providing services to two remote ASPs.

When an ASP goes up and registers for the Application Server (AS), the UA driver uses an existing static configuration or requests the Layer Manager to configure the Application Server for use. Either a copy of the per-AS structures for ASP and SP already exist, or these copies are created when the Layer manager configures the AS for service. The UA driver marks the Application Server as registered.

When an ASP goes active for the Application Server (AS), the UA driver activates the SS7 Provider (if it is not already active) and moves the AS to the active state.

5.2.2. Multiple peer SGP Example

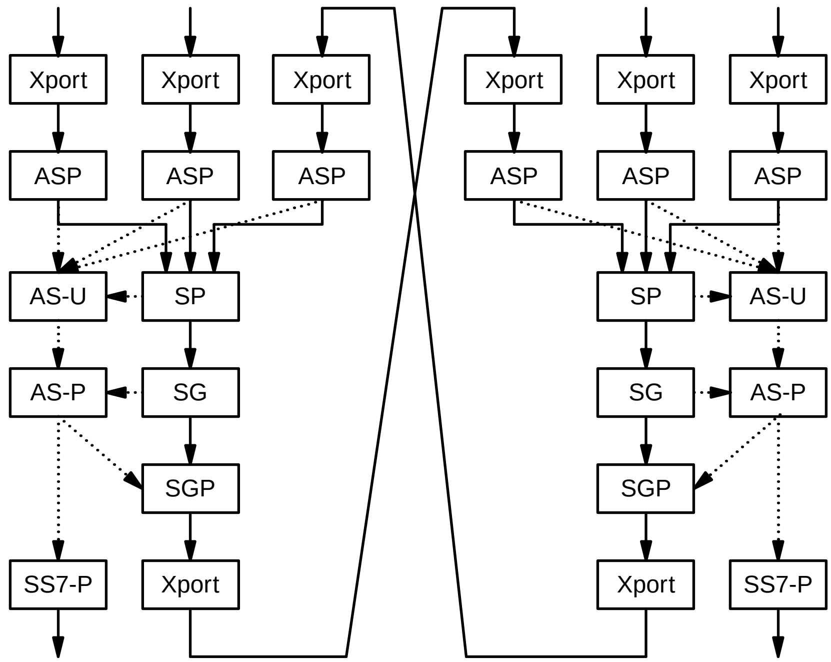

Figure 5-3. Multiple peer SGP Example

5.3. Combined Examples

5.3.1. Simple Combined Example

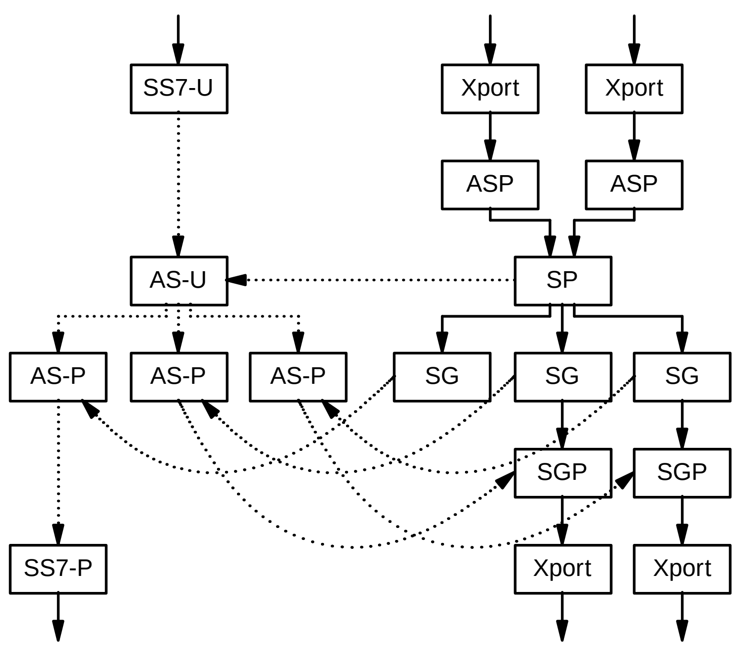

Figure 5-4. Simple Combined Example

Figure 5-4 illustrates a simple Combined example.

5.3.2. Simple Pass-Through Example

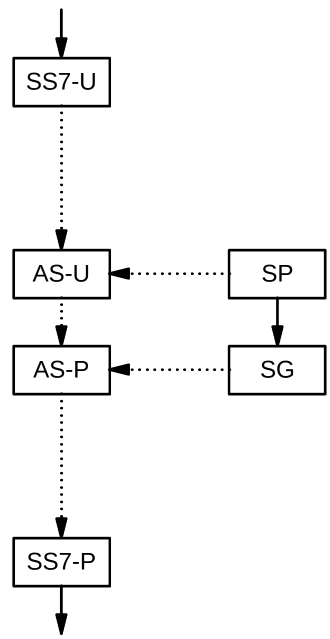

Figure 5-5. Simple Pass-Through Example

Figure 5-5 illustrates a simple combined pass-through example. In this example the SP is acting as both ASP and SG. This is the OpenSS7 stack stand-alone mode where no peer redundant processors are defined.

Data and control primitives from the SS7 User (SS7-U) are channelled through the associated Application Server (AS) directly to the SS7 Provider (SS7-P). Data and control primitives from the SS7 Provider are passed directly to the SS7 User through that Application Server (AS) in the same fashion.

5.3.3. Pass-Through with Redundant Processor

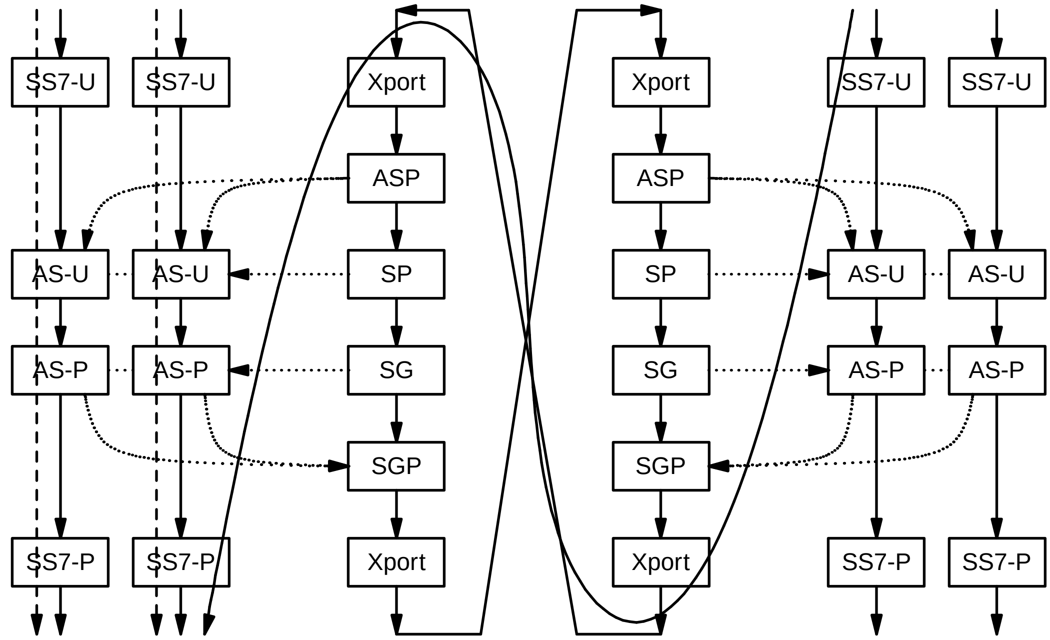

Figure 5-6. Pass-Through with Redundant Processor

Figure 5-6 illustrates a pass-through with redundant processor example. In this example, the SG is acting as both ASP and SGP. This is the OpenSS7 stack redundant configuration where a peer redundant processor is defined.

Data and control primitives are normally passed directly between the SS7 User and the local SS7 Provider on each processor.

5.3.4. Shared Redundant Processor

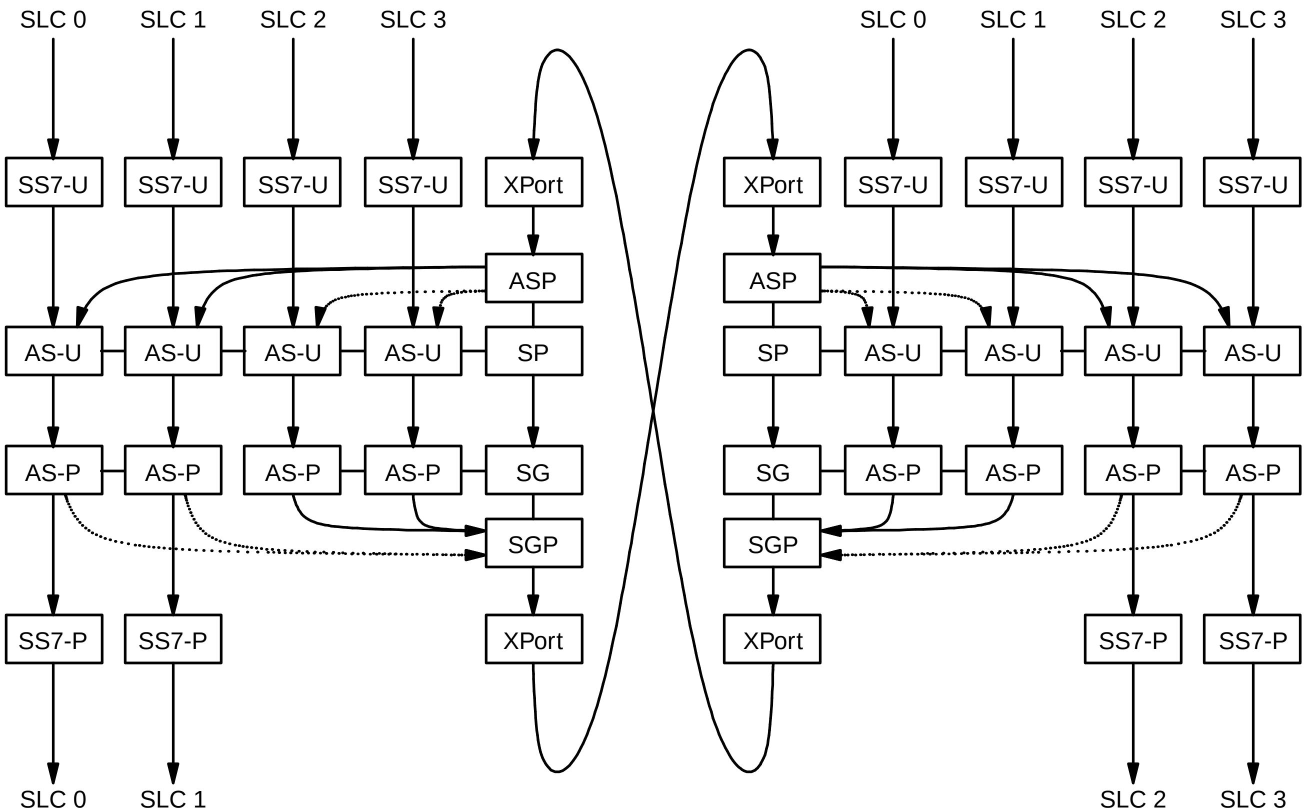

Figure 5-7. Shared Redundant Processor

Figure 5-7 illustrates a shared redundant processor example. In this example, each processor is acting as both ASP and SGP. Processor #1 passes through SLC 0 and SLC 1. Processor #2 passes through SLC 2 and SLC 3. However, Processor #1 also provides remote access to SLC 0 and SLC 1 to Processor #2, and Processor #2 also provides remote access to SLC 2 and SLC 3 to Processor #1. This means that all four signalling links (SLC 0 through SLC 3) are available to both processors.

The following points are notable:

(1) Only two transport connections are required regardless of the number of signalling links supported.

(2) Should the local ULP process fail, its local SS7-U streams will be closed, however, the other processor will begin handling traffic for local SS7-P streams and an outage will not occur. This makes it possible to upgrade, for example, MTP Level 3 processes on the one processor without disrupting service.

APPENDIX A

An Appendix

A.1. Test Header

Figure A-1. Overview

References

M2PA03.
T. George, R. Dantu, M. Kalla, H. J. Schwarzbauer, G. Sidebottom, K. Morneault, “SS7 MTP2-User Peer-to-Peer Adaptation Layer,” <draft-ietf-sigtran-m2pa-03.txt>, Internet Engineering Task Force - Signalling Transport Working Group (July 20, 2001). Work In Progress.

List of Illustrations

Table of Contents