Oracle® Communications Network Charging and Control

Universal Call Agent for ISUP Technical Guide

Release 15.2

January 2026

Copyright

Contents

About This Document

Scope

The scope of this document includes all the information required to install, configure and administer the UCAI application.

Audience

This guide was written primarily for system administrators and persons installing, configuring and administering the UCAI application. However, sections of the document may be useful to anyone requiring an introduction to the application.

Prerequisites

This manual describes system tasks that should only be carried out by suitably trained operators.

A solid understanding of Unix and a familiarity with IN concepts are an essential prerequisite for safely using the information contained in this technical guide. A good of switch configuration is required to correctly configure the UCAI. A detailed understanding of M3UA is essential for configuring the M3UA interface.

Attempting to install, remove, configure or otherwise alter the described system without the appropriate background skills, could cause damage to the system; including temporary or permanent incorrect operation, loss of service, and may render your system beyond recovery.

Although there are no prerequisites for using this guide, familiarity with the target platform would be an advantage.

Related Documents

The following documents are related to this document:

- *Service Logic Execution Environment Technical Guide*
- *Service Management System Technical Guide*
- *Service Management System User's Guide*
- *ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes*
- *ITU-T. Q.764, Signalling System No. 7 — ISDN User Part signaling procedures*

Document Conventions

Typographical Conventions

The following terms and typographical conventions are used in the Oracle Communications Network Charging and Control (NCC) documentation.

Formatting Convention	Type of Information
Special Bold	Items you must select, such as names of tabs. Names of database tables and fields.
Italics	Name of a document, chapter, topic or other publication. Emphasis within text.
Button	The name of a button to click or a key to press. **Example:** To close the window, either click **Close**, or press **Esc**.
Key+Key	Key combinations for which the user must press and hold down one key and then press another. Example: **Ctrl+P** or **Alt+F4**.
`Monospace`	Examples of code or standard output.
`Monospace Bold`	Text that you must enter.
variable	Used to indicate variables or text that should be replaced with an actual value.
menu option > menu option >	Used to indicate the cascading menu option to be selected. Example: **Operator Functions > Report Functions**
hypertext link	Used to indicate a hypertext link.

Specialized terms and acronyms are defined in the glossary at the end of this guide.

Chapter 1

System Overview

Overview

Introduction

This chapter provides a high-level overview of the application. It explains the basic functionality of the system and lists the main components.

It is not intended to advise on any specific Oracle Communications Network Charging and Control (NCC) network or service implications of the product.

In this Chapter

This chapter contains the following topics.

UCAI Introduction

Description

The Universal Call Agent for ISUP (UCAI) is a signaling element which provides Intelligent Network functionality, without having to upgrade or replace non-SS7 capable switches.

It does this by using fixed connections between incoming and outgoing calls.

UCAI controls call signaling, while internally looping the speech circuits of the call. Without the heavy load that speech traffic creates, UCAI is able to handle more calls than a service node solution.

Features

Using only the Call Set-Up Protocol ISUP, it can offer:

- Advanced number translation services such as Free phone and Number portability
- Call monitoring services such as Prepaid

Features include:

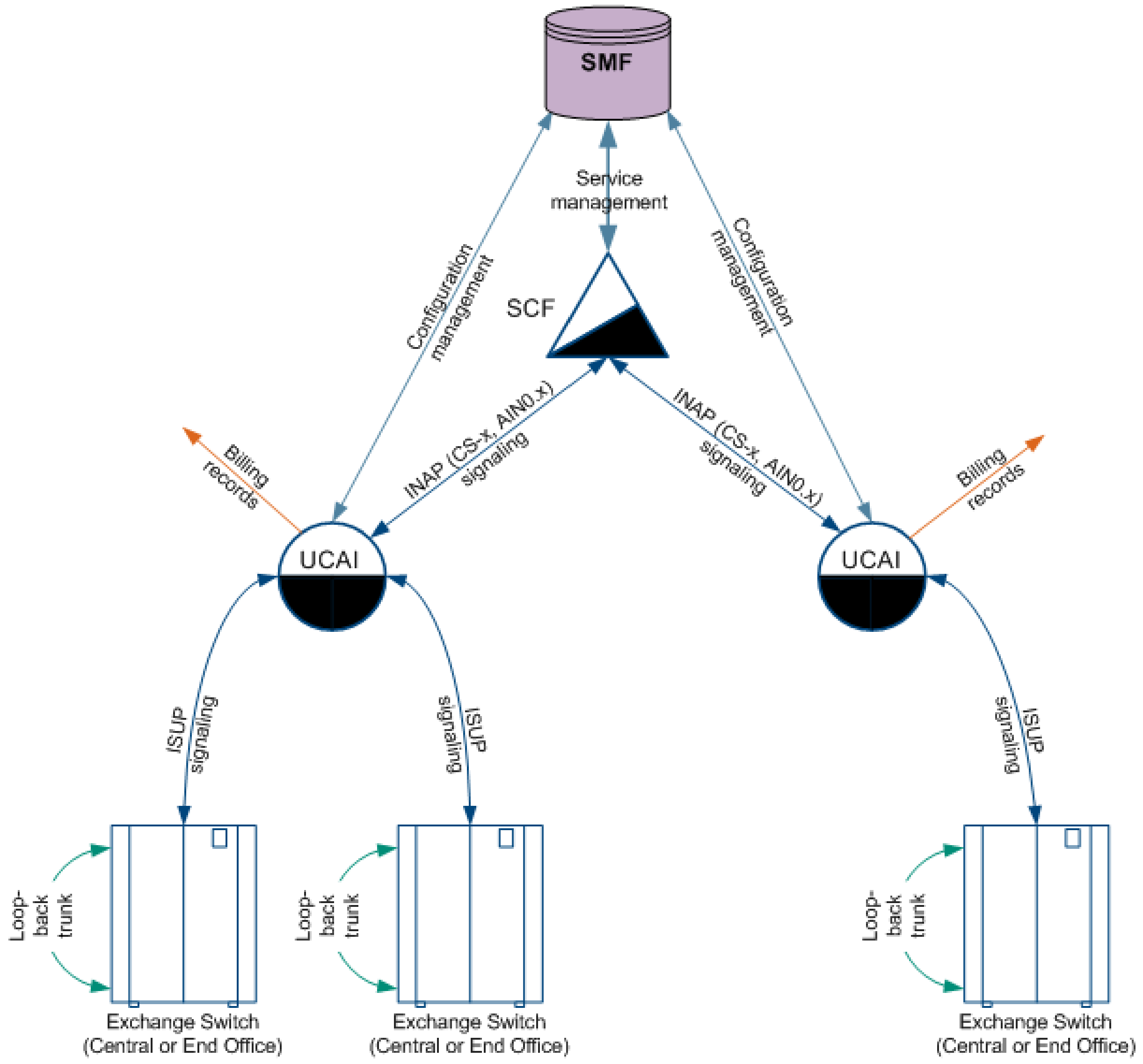
- Support for standard protocols (INAP, ISUP)
- Support for all ISUP messages, including maintenance messages (for example: blocking, unblocking, reset, and circuit group messages).
- Provides basic (CS1) IN functionality
- Leg control can be used for more advanced solutions
- Can originate both legs of a call (for example, a call can be originated from a web page)
- Each transient switch can have its own UCAI for increased network performance and resilience
- Failover and load balancing can be easily supported by configuring the network routing to send messages between mated UCAIs

- INAP between UCAI and SLC can be over TCP/IP (reducing signaling costs)
- UCAI runs on low cost general purpose hardware

Network architecture

This diagram shows how UCAI is installed within a network. There may be one or more UCAI, each connected to one or more exchanges.

The UCAI communicates with the SCF over SS7, using either ETSI/ITU-T INAP or AIN0.x. An operator may manage its configuration and alarms by SMS, or via a Q.3 agent using Q.751.



Note: Although the UCAI is shown as a separate physical entity in this diagram, it may also co-exist on the SLCs. If so, it will still communicate with the SCF over INAP (though this may not leave the SS7 stack).

Call processing

UCAI achieves SSP functionality by:

- Effectively splitting a single physical switch into 2 logical switches.

- Using the routing plan of the network, the UCAI appears as another Transient switch to the original switch for signaling routing purposes.
- Voice circuits are looped back to provide the speech path.

UCAI uses ISUP to control calls made to looped-back ISUP speech trunks.

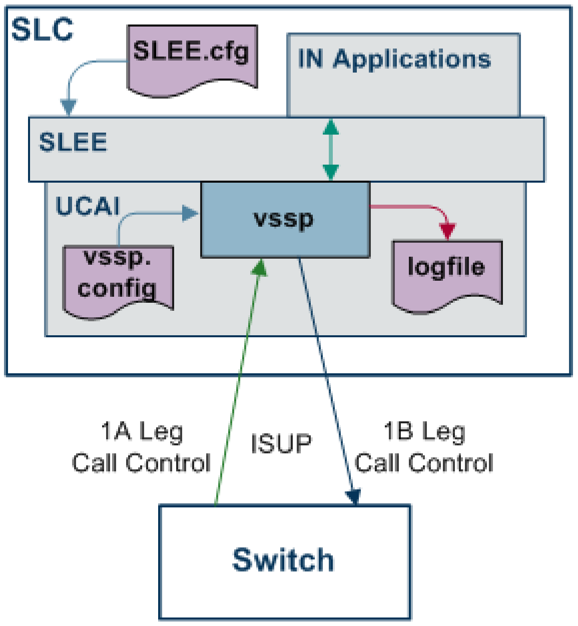
- Each speech circuit going out of the switch has a permanently mapped circuit going into the switch.
- Consequently, the UCAI can make a call on the egress circuit that is effectively a continuation of the original call (even if mapped to a destination determined by an SLC via an INAP protocol).

UCAI controls processing of the whole call, or control of an individual leg.

- It can be used to implement most IN services.
- The type of processing is specified on a per service basis.

Components

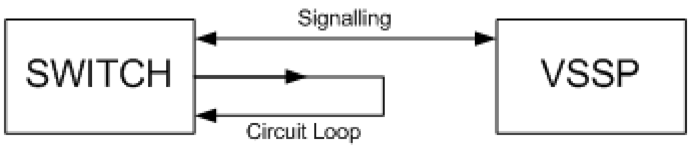
This diagram shows the main components of the UCAI hosted on a SLC.



UCAI Operation - Loop Back Mode

Basic processing

Here is an example of how UCAI works in its simplest configuration. A single incoming circuit and a single outgoing circuit.



As shown in the example, the only actual connection to UCAI is the signaling link. No speech paths are routed. All signaling, for both the incoming and outgoing circuits on the loop, are carried on the signaling link. The speech path is looped with very low visibility to the switch. This may be achieved by using a static configuration of the switch matrix, or by an external piece of hardware. In either case, the switch's call process is unaware of the loop.

Calls to UCAI must only be routed down an incoming leg. Calls routed down a UCAI incoming leg that are not intended for UCAI, are still sent to the SCF for processing. This almost always results in that call being rejected.

Calls routed to a UCAI outgoing leg are rejected.

A full understanding of this is essential to correct configuration of a UCAI into a Telco's network.

Call processing

The following steps describe what happens when a call arrives on the incoming leg to the UCAI.

Note: This process description assumes the network is configured as described above.

Stage	Description
1	The call is routed to UCAI. **Notes:** • Before this can happen, the external network must have been correctly configured to route the call into UCAI. • The last part of the host network on which the call appears is the switch which is directly connected to UCAI. This is the switch which sends the ISUP IAM message to UCAI, indicating the call is being made. • The specific circuit (that is, timeslot) which the call arrives on must be specified as "incoming". If a call arrives on a leg designated as "outgoing", it will be rejected.
2	UCAI queries an SCF to determine how the call is to be processed. This example assumes a simple number translation (for example, translating a free or premium rate phone number into its final destination).
3	UCAI sends an outgoing ISUP IAM message on the outgoing leg that corresponds to the incoming leg on which the call arrived. The IAM contains the translated number. It may also have other parameters modified by the SCF. UCAI automatically selects a circuit on to route the outgoing call on. Because it contains no switch matrix, the only circuit the call's speech can be present is the one to which the incoming leg is looped.
4	Call processing continues as it would with any other SSP. The ISUP messages are acted on by the SSP under SCF control, exactly as would happen with a conventional SSP. The SCF is unaware UCAI is different from any other sort of SSP.

Important installation planning

In order to correctly plan, install and configure a working installation you must be aware of the following issues:

- Signaling
- Loop Back
- UCAI configuration

These issues are discussed further below.

Signaling

Signaling configuration is very important to the correct functioning of UCAI. The example network configuration above only has a single signaling link. That link is the only connection between UCAI and the attached network.

Signaling for both the incoming and outgoing legs is carried on the same link. This can cause some complex issues for the configuration of the attached switch, and these must be carefully addressed.

The signaling link is set up in a completely standard manner, with point codes allocated to both ends and an agreement on the allocation of cics to specific timeslots. However, the two circuits involved are looped without the switch's knowledge. This has a major effect on the routing policy that must be configured on the attached switch.

In the example, two circuits are used. However, the switch may only make calls to the UCAI down the incoming leg. This requires the switch to be configured so routing is allocated not only on a point code basis (which permits the switch to route calls down any circuit), but also on a per circuit basis (so that calls are only routed down those circuits designated as "incoming" to the UCAI).
The details of this are highly switch-specific, but must be provided to ensure correct operation.

Configuration example

This example extends the previous example.

Assumptions:

- UCAI has point code 1
- Switch has point code 2
- Single incoming circuit is assigned cic 1
- Single outgoing circuit has cic 2
- Calls should be processed with the prefix 0800

Required Configuration:

The switch must be configured so it will route calls with the 0800 prefix to point code 1, but only on cic 1. A switch-specific example is an mml command of the form:

```
ROUTE:PREFIX=0800,DPC=1,CICS=1;
```

This command indicates calls prefixed with 0800 should be sent to point code 1, and the only available cic is 1.

If the switch does not offer this degree of routing control, there are several options discussed in *Routing Options* (on page 24).

Loop back

Ideally, the loop back is performed by the switch. However, it must be possible to do this without interfering with the switch's call processing. If this is impossible, some external form of loop back device must be employed. This may be possible with whatever device is used to extract the signaling information and pass it onto the UCAI.

UCAI configuration

UCAI must also be configured. Given the point codes and cics detailed in *Configuration* (on page 17), the configuration file need only contain:

```
connect 1/2/1 to 1/2/2.
```

Note: The full stop at the end of every configuration statement is essential.

Throttling

UCAI uses a sliding window of one second to measure load. If an initial address message (IAM) arrives less than one second after the one <CAPS limit> IAMs before it, it is rejected.

Example:

UCAI is set with a limit of 20 CAPS. This means in any single second window UCAI will allow 20 call attempts and the 21st will be rejected.

- If the IAMs arrive exactly at 1/20th of a second intervals, there will be 20 CAPS.
- If an IAM arrives even one microsecond 'early' it causes a peak of 21 CAPS and is rejected.

So, even if traffic is sent at 20 CAPS and a 20 CAPS limit is set, the occasional call will still be rejected.

Supported Messages Overview

Introduction

This table describes UCAI's main message groups:

Component	Description
Call Processing	The basic call processing package is composed of a group of simple messages to progress the call. For more information, see *Call processing messages* (on page 6) below.
Additional Call Set-up	Call setup may be made more complex by the use of extra, more complex messages to progress the call. For more information, see *Additional call setup messages* (on page 7) below.
Continuity Testing	Some networks support use of ISUP’s continuity testing procedures. UCAI is able to handle all such situations correctly. For more information, see *Continuity testing messages* (on page 8) below.
Maintenance	The UCAI supports single circuit maintenance messages, or group messages, to allow for blocking, unblocking, or resetting of circuits. For more information, see *Maintenance messages* (on page 9) below.

Call processing messages

This table lists the basic call processing messages which are supported by the UCAI.

Message Type	Description
Initial Address Message (IAM)	The Initial Address Message has five mandatory parameters and 42 optional parameters (for more information, see Table 32 of ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes). The mandatory parameters are: • nature of connection indicators • forward call indicators • calling party’s category • transmission medium requirement • called party number The UCAI handles all the mandatory and optional parameters. However, some filtering may take place, in order to ensure correct operation of the UCAI. In addition, certain pathological values for some parameters may cause rejection of the call in some cases. **Note:** If a hop counter occurs on the incoming leg and its value is zero, the call is rejected. If its value is greater than zero, the value is decremented and used on the outgoing leg. If no hop counter was provided, one is inserted with a default value.

Message Type	Description
	Use of the hop counter is very important. It avoids the possibility of a routing error in the network resulting in a 'circular call' that 'bounces' between the UCAI and its attached switches. Such an occurrence is a very dangerous network event, as it consumes very large amounts of processing power and signaling bandwidth.
Address Complete Message (ACM)	The Address Complete Message has one mandatory parameter and 19 optional parameters (for more information, see Table 21 of ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes). The mandatory parameter is backward call indicators. As with the Initial Address Message, some filtering of the message parameters may take place.
Answer (ANM)	The Answer message has no mandatory parameters and 21 optional parameters (for more information, see Table 22 of ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes). As with the Initial Address message, some filtering of the message parameters may take place.
Release (REL)	The Release message has one mandatory parameter, and up to seven optional ones (for more information, see Table 33 of ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes). The mandatory parameter is cause indicators. As with the Initial Address message, some filtering of the message parameters may take place.
Release Complete (RLC)	The Release complete message has one optional parameter (for more information, see Table 34 of ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes). The optional parameter is cause indicators. As with the Initial Address message, some filtering of the message parameters may take place.

Additional call setup messages

Call setup may be made more complex by the use of some of the more complex messages to progress the call. This table describes these message types supported by the UCAI.

Message type	Description
Subsequent Address Message (SAM)	The Subsequent Address Message has one mandatory parameter (for more information, see Table 35 of ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes). The mandatory parameter is subsequent number. The UCAI will handle use of SAM messages (also known as 'overlap' sending). However, the service itself should be made aware that this form of signalling may be in use, as it can confuse some services, if they assume all the digits are to be presented at once. This can usually be avoided by careful use of network, UCAI and service configuration, but care should be taken to avoid it. In addition, use of SAM or overlap signalling results in increased network traffic and processing overhead. It should therefore be avoided, wherever possible.

Message type	Description
Call Progress	The Call Progress has one mandatory parameter and 25 optional ones (for more information, see Table 23 of ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes). The single mandatory parameter is event information.
Connect	The Connect message has one mandatory parameter and 19 optional ones (for more information, see Table 27 of ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes). The mandatory parameter is backward call indicators. The UCAI treats the Connect message in a very similar manner to an Address Complete followed by an Answer.
Identification Request	The Identification Request message has no mandatory parameters and three optional ones (for more information, see Table 47 of ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes).
Identification Response	The Identification Response message has no mandatory parameters and seven optional ones (for more information, see Table 48 of ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes.).
Information Request	The Information Request message has one mandatory parameter and three optional ones (for more information, see Table 31 of ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes). The single mandatory parameter is information request indicators.
Information	The Information message has one mandatory parameter and six optional ones (for more information, see Table 30 of ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes). The single mandatory parameter is information indicators.
Suspend and Resume	The Suspend and Resume messages have one mandatory parameter and one optional (for more information, see Table 38 of ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes). The single mandatory parameter is suspend/resume indicators. The UCAI processes Suspend and Resume requests as required by §9.1.3 of ETR 164, Integrated Services Digital Network (ISDN); Intelligent Network (IN); Interaction between IN Application Protocol (INAP) and ISDN User Part (ISUP) version 2.

Continuity testing messages

Some networks support use of ISUP’s continuity testing procedures. UCAI handles the following cases correctly:

- Testing after an Initial Address message (this may occur after every IAM, or some form of statistical sampling may be employed).
- Testing an idle circuit.
- A call can be placed that is waiting for the results of a continuity test on a previous circuit.

Note: In theory, there should be no need for continuity checks in a modern network. The lower-level signalling should detect the failure of a speech or signalling link, and block use of those circuits. However, it can be useful for testing for correct configuration of the UCAI.

A major feature of the UCAI is speech circuits do not pass through it. Consequently, the UCAI (though able to detect failures in the signalling links) is unable to detect problems with the associated speech circuits. Instead it relies upon the attached switches to detect problems.

If the UCAI's speech circuits are mis-configured, the usual result is for the call to be made and appear to work (including billing) but no speech is heard.

Continuity testing can help guard against these problems. Periodic and/or statistical testing is recommended, unless the operator has taken great care in the configuration of the UCAI and its attached switches.

The UCAI handling of continuity testing is in accordance with §2.1.8 of ITU-T. Q.764, Signalling System No. 7 — ISDN User Part signalling procedures.

Continuity check request

The Continuity Check Request message has no parameters (for more information, see Table 39 of ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes).

Use of CCR is only valid on idle incoming circuits, and the CCR message is passed onto the corresponding outgoing circuit.

Continuity

The Continuity message has one mandatory parameter and no optional ones (for more information, see Table 28 of ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes).

The mandatory parameter is continuity indicators.

Maintenance messages

BThis table describes the single circuit and group circuit maintenance messages supported by UCAI.

Message Type	Description
Single circuit maintenance messages	Several single circuit maintenance messages are defined in ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes. Those supported by the UCAI are: • Blocking • Blocking Acknowledgment • Reset Circuit • Unblocking • Unblocking Acknowledgment. None of these have any parameters. The functionality the same as is described in ITU-T. Q.764, Signalling System No. 7 — ISDN User Part signalling procedures, but with one addition imposed by the UCAI. As the UCAI ties incoming and outgoing circuit together, any blocking or unblocking coming into the UCAI on an outgoing circuit causes a similar message to be reflected on the corresponding incoming circuit. Similarly, if any problem occurs with an outgoing leg, the UCAI blocks the corresponding incoming leg, to avoid being given calls it cannot process.
Circuit group maintenance messages	It is often more efficient to use one of the group messages defined in ITU-T. Q.763, Signalling System No. 7 — ISDN User Part formats and codes, than to use a number of the corresponding single circuit maintenance message. The UCAI currently supports the following group messages: • Circuit Group Blocking • Circuit Group Blocking Acknowledgment • Circuit Group Unblocking • Circuit Group Unblocking Acknowledgment

Message Type	Description
	• Circuit Group Reset • Circuit Group Reset Acknowledgment. The functionality is the same as is described in ITU-T. Q.764, Signalling System No. 7 — ISDN User Part signalling procedures, but with one addition imposed by the UCAI. As the UCAI ties incoming and outgoing circuit together, any blocking or unblocking coming into the UCAI on an outgoing circuit causes a similar message to be reflected on the corresponding incoming circuit. Handling of some circuit group messages can be problematical in some networks. In addition, due to the potentially arbitrary nature in which incoming and outgoing circuits can be tied together for the UCAI, a circuit group block on an outgoing leg is repeated through to the incoming side, as a series of single circuit blocks. A similar technique is used in handling circuit group resets and unblocking. Therefore, the UCAI only sends circuit group messages in response to circuit group messages received. If the network does not support circuit group messages, none will arrive at the UCAI and so the UCAI will not use them in response.

Overlap Sending

Introduction

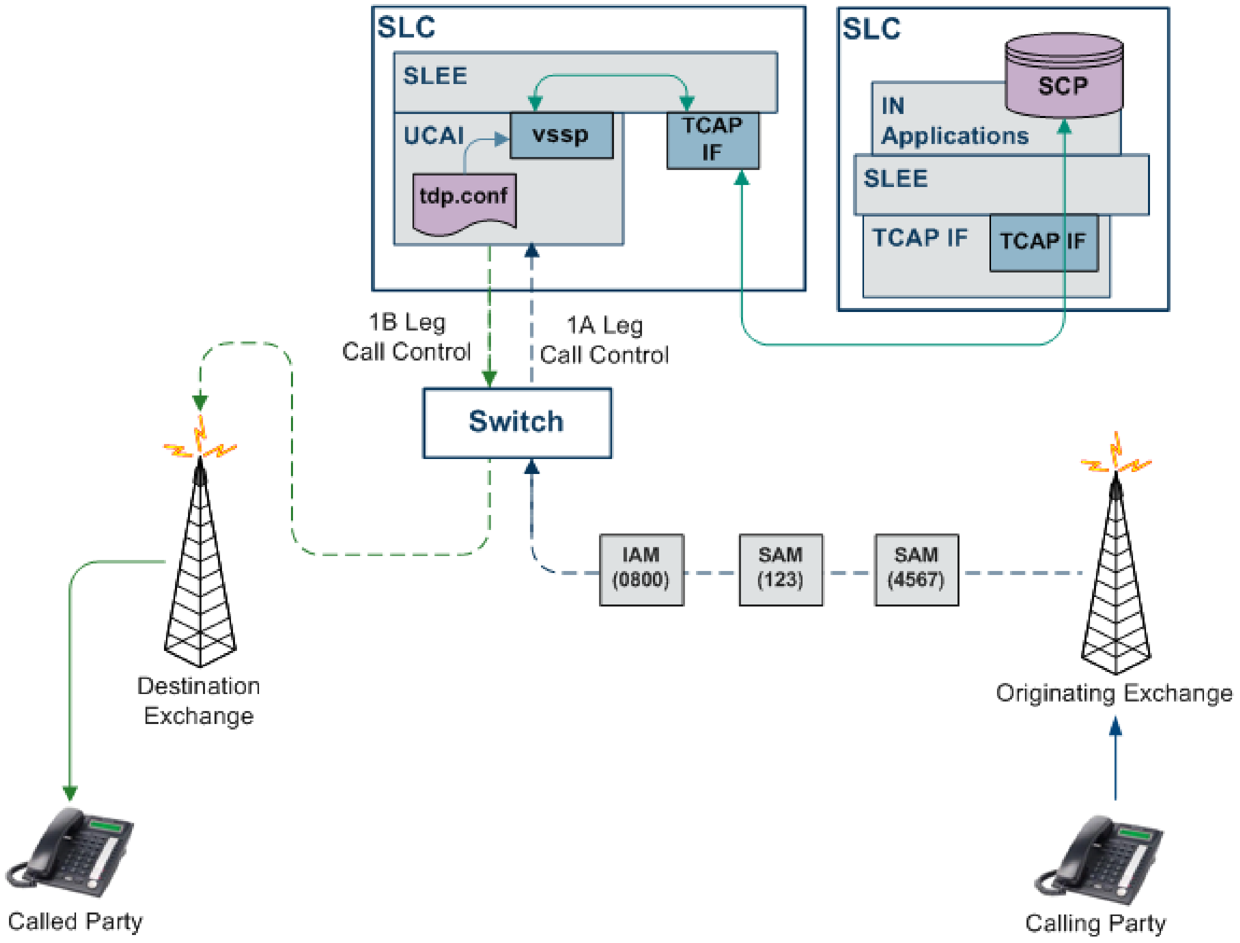
Overlap Sending or Signalling can be used in a fixed line environment where a call set up can begin even before the caller has dialed all digits of the intended destination number. This is achieved when the caller has dialed enough digits for an originating exchange using the Overlap Sending method, to determine the target exchange. The originating exchange packs the digits collected from the calling party so far, into an Initial Address Message (IAM) and sends it to the next exchange.

Subsequent digits dialed by the caller are contained within the Subsequent Address Message (SAM). Each new SAM contains additional digits dialed by the caller. UCAI invokes appropriate IN services within the call setup path based on predefined triggering rules.

Note: Overlap sending is not possible in a mobile environment since the calling party must dial all the digits of the intended destination before initiating a call setup attempt.

How Overlap Sending works

Here is an example of how UCAI supports the Overlap Sending method of setting up a call.



The originating exchange will attempt to set up a call once it has collected sufficient digits from the calling party to identify the next target exchange (tandem or ultimate destination exchange) in the call flow. It will generate an IAM (Initial Address Message) with all the digits the calling party has dialed until then, and route it along the network.

Subsequent digits dialed by the caller will be contained within the Subsequent Address Message (SAM). Depending on the caller's dialing speed, one or more SAMs may be generated. Each SAM will contain any new digits dialed since the last SAM (or the initial IAM) was sent.

IN services within the signalling path are activated based on the triggering rules used by UCAI. Currently, these triggering rules are defined in the **tdp.conf** file.

Triggering Rules

In the Overlap Sending method, the call setup begins even before the caller dials all the digits of the intended destination. However, in such a scenario, IN systems involved in the call signalling path cannot determine when to trigger the IN services. Currently, UCAI uses the triggering rules defined in the tdp.conf. This file is processed by the IN call model component of the UCAI.

UCAI triggers IN services when the incoming digits dialled so far trigger a matching rule in the tdp.conf. The decisive digits invoking the rule may be contained in:

- The IAM received from the originating exchange

- The IAM + one or more SAM messages received from the originating exchange

tdp.conf

Here is an example of the **tdp.conf** file.

```
KEEP SD
ETC RULES=6 3
3 1 3 request all 3:01473222
```

Stop digit

Typically, when the caller dials a local/national number, the originating exchange will lookup the configured numbering plan information to determine if sufficient digits have been dialed to identify the target exchange. If it recognizes the numbering pattern of the dialed digits, the originating exchange will insert a stop digit into the called number signaling sequence.

UCAI triggers the IN service when a stop digit is detected and an appropriate rule in **tdp.conf** is found. The stop digit may be contained in either of the following:

- The IAM received from the originating exchange, that is at the end of the IAM
- The IAM + one or more SAM messages received from the originating exchange, that is, at the end of the last SAM

SAM timer

Sometimes, due to a lack of numbering plans available, a local or transit exchange within a country may not correctly decipher the numbering pattern of the dialed digits (typically an international number). Consequently, the originating exchange fails to insert the stop digit. This results in the UCAI waiting an indefinite period of time for the overlap sequence to end, not knowing when to trigger the IN service. To safeguard against this situation, the UCAI is equipped with a timer, known as SAM timer. The SAM timer is used for initiating IN services when Overlap Sending is invoked for a number with an unrecognizable numbering scheme.

The UCAI waits for a fixed period of time to receive the first or next SAM in an overlap sequence before activating the IN services. When this timer has expired the UCAI assumes that the caller has completed the dialing sequence for the intended destination number and ignores any additional dialed digits. This means no new digits will be accepted from the caller/network after the timer expires.

The UCAI starts the SAM timer for a call when it receives an IAM not containing the stop digit. The SAM timer is restarted each time a new SAM is received for the call.

Note: The SAM timer will not be started if the digits contained within the IAM or the SAM are sufficient for the UCAI to trigger an IN service.

Forced trigger scenario

If the SAM timer expires and an IN Service has not been triggered, then the UCAI adds a stop digit to the dialed overlap sequence. This is known as a "forced trigger" scenario.

One of the following results may occur in a forced trigger scenario:

- A matching rule in the **tdp.conf** file is found which triggers an IN service
- A matching rule is not found and the UCAI releases the call by sending a REL message on the A-leg

IN Service Processing

After an IN service is triggered, it returns any one of the following results:

- Continue
- Connect ("cut and paste" option not set)
- Connect ("cut and paste" option set)

- ETC (treated as 'Connect' by UCAI)
- Release
- Error

After the UCAI has sent an IAM on the B-leg of the call and BEFORE an ACM message is received from the B-leg, it is possible that additional SAM messages may be received on the A-leg.

Before an IAM is sent on the B-leg

This table describes how the UCAI treats different message scenarios before an IAM is sent on the B-leg.

Scenario	Result
Additional digits are received in SAM messages while IN service is treating the request.	UCAI will store the digits.
In a forced trigger scenario, additional digits are received in SAM messages while IN service is treating the request.	UCAI will ignore the digits.
'Continue' message is received from IN service.	All digits received by UCAI (including digits received/stored when the IN service was processing the call) will be used in the calledPartyAddress field of the IAM sent on the B-leg.
'Connect' message is received from IN service with the "cut and paste” field NOT set.	The number specified in the destinationRoutingAddress field of the 'Connect' message shall be used in the calledPartyAddress field of the IAM sent on the B-leg Any additional digits received in the period when the IN service was processing the call will be ignored.
'Connect' message is received from IN service with the "cut and paste” field set.	All digits received by UCAI (including digits received/stored when the IN service was processing the call) will be used. Based on the options set in the “cut and paste” field, the required number of digits will be cut from the beginning of the calledPartyAddress received in the IAM and all the SAM messages. The cut number is then pasted to the number received in the destinationRoutingAddress. The resultant number from the cut and paste operation will be used in the calledPartyAddress field of the IAM which is sent on the B-leg. **Note:** In a forced trigger scenario any additional digits received during the IN service processing will be ignored and not be included in the resultant number when the "cut and paste" field is set.

Scenario	Result
An 'ETC' operation is received from IN service.	The number specified in that operation will be used in the calledPartyAddress field of the IAM sent on the B-leg. Any additional digits received in the period when the IN service was processing the call will NOT be used in the calledPartyAddress and will be ignored.

After an IAM is sent on the B-leg

After the UCAI has sent an IAM on the B-leg of the call and BEFORE an ACM message is received from the B-leg, it is possible that additional SAM messages may be received on the A-leg. Any SAM messages received on the A-leg of the call after the IAM has been sent on the B-leg of the call shall be transparently sent onto the B-leg of the call.

The following cases are some exceptions to this transparent sending:

- ACM is received on the B-leg.
- IN services issues a 'Connect' message with the "cut and paste" option not set.
- IN services issues an 'ETC' message.
- IN service was triggered in a forced trigger scenario.

In case 2 and 3, the UCAI will add a stop digit to the number if one is not already present. The UCAI then ignores all subsequent messages if:

- A stop digit is encountered on either the IAM or SAM message sent on the B-leg
- The IN service was triggered in a forced trigger scenario

If the UCAI has been instructed by the IN service to route the call to a totally different number/address than as intended by the caller, the SAM being relayed onto the B-leg may have an undesirable affect on the destination (or transit exchange).

For example: The IN service routes the call to an IVR number where the caller was expecting to be connected to an international party.

Transparent Passing of ISUP Messages

During call setup, the UCAI allows certain messages to be transparently passed between the A and B legs, even after the call has been established.

Here, the following ISUP messages are notable as examples:

- The call progress message (CPG) can be passed transparently between the A and B leg of a call after it has been established, that is, after the ANM message is received from B-leg.
- Facility (FAC) messages received on either A or B leg can be transparently passed onto the other leg without triggering the IN service.

Handling Called Party Suspends

In this situation, the called party suspends a call by replacing the receiver back on the hook, in order to change the physical phone that is currently connected to the call. The called party then physically attends the call from another phone on the same line. The UCAI handles this functionality by providing an appropriate safeguard that allows the B-leg to stay connected, even if the call has been suspended for a specific period of time.

On receiving a SUS (suspend) message on the B-leg, the UCAI starts a 'suspend' timer for the call. Currently, the T6 timer is implemented as the suspend timer. The UCAI will stop the suspend timer on receiving a RES (resume) message on the suspended B leg.

When the suspend timer expires, the UCAI sends a REL message on the B-leg of the call and waits for a RLC message. On receiving the RLC message, the IN call model is notified that the B-leg has been disconnected. If the IN service has armed an event for B-leg disconnect, it will be informed by the IN call model that the called party has disconnected. It is possible to configure the length of time used by the suspend timer in seconds.

Chapter 2

Configuration

Overview

Introduction

This chapter explains how to configure the Oracle Communications Network Charging and Control (NCC) application.

In this chapter

This chapter contains the following topics.

Configuration Overview

Configuration components

UCAI is configured by the following components:

Component	Locations	Description	Further Information
vssp.config	all SLC	Required to configure the UCAI process.	*UCAI Configuration File* (on page 18)
vssp.sh	all SLC	Sets UCAI process's command line startup parameters.	*Configuring the Environment* (on page 18)
m3ua.config	all SLC	Required to provide M3UA configuration. Not required if this version of UCAI does not support M3UA.	*M3UA Configuration* (on page 21)
keyfile	all SLC	Required if UCAI has a circuit limit.	*Editing the vssp.sh file* (on page 18)
tdp.conf	all SLC	Configures the IN Call Model library. Required for interaction with the SLEE.	*Configuring IN Call Model Triggers* (on page 29)
SLEE.cfg	all SLC	Sets up SLEE interfaces and applications.	*SLEE Technical Guide*

Rereading configuration

UCAI will reread its configuration when it is restarted.

Note: To restart UCAI, you must restart the SLEE. For more information about restarting the SLEE, see *SLEE Technical Guide*.

Configuring the Environment

Introduction

The UCAI environmental variables are set in the **vssp.sh** file.

The file **vssp.sh** is used to set environment variables and command line parameters before the UCAI executable is started. There are potential stack-specific environment variables as well as UCAI-generic command line options.

Editing the vssp.sh file

There are three possible command line parameters in the format:

```
vssp [-c] dir/filename [-k] dir/filename [-m] dir/filename
```

The only mandatory option is the name of the vssp configuration file. This can either be specified by the -c flag or it is the first non- flagged parameter. For more information about setting up this file, see *UCAI Configuration File* (on page 18).

In UCAI builds which require a license key file, the name of the file may be specified on the command line. This may be specified either using the -k flag or it is the second non-flagged parameter. If it is not given, UCAI will default to using the file called keyfile in the same directory as the **vssp.config** file.

In UCAI builds which use the M3UA protocol, the M3UA configuration file name can be given. This can either be set using the -m flag or it is the third non-flagged parameter. If no file name is given, UCAI will default to using a file called **m3ua.config** in the same directory as the **vssp.config** file.

Example vssp.sh file

Here is an example **vssp.sh** file.

```
LD_LIBRARY_PATH=/usr/lib/secure
export LD_LIBRARY_PATH
exec /IN/service_packages/VSSP/bin/vssp -c /IN/service_packages/VSSP/etc/vssp.config
-m /IN/service_packages/VSSP/etc/m3ua.config >>
/IN/service_packages/VSSP/tmp/logfile 2>&1
```

UCAI Configuration File

Introduction

The main UCAI configuration file is **vssp.config**. It configures the UCAI process and is located in:

/IN/service_packages/VSSP/etc

vssp.config has two sections:

1. Parameters
2. Point Code routing (Circuit Loops)

Configuring circuit loops

Circuit loops are described using the connect statement. In their simplest form connects take the form of:

```
connect circuit to circuit
```

Note: Each command in **vssp.config** must end with one of the following:

Period (.)

Semi-colon (;)

The *circuit* in the above example is composed of three parts in the format:

```
    a/b/c
```

Where:

- a = the point code of the UCAI end
- b = the point code of the remote end
- c = the circuit identification code

Example 1

A circuit loop has two circuits:

1 Incoming circuit to point code 1 from remote point code 2 on cic 1

2 Outgoing circuit from point code 1 to remote point code 2 on cic 2

This setup would be configured in **vssp.config** as:

```
connect 1/2/1 to 1/2/2.
```

Example 2

A single 2.048 Mbps ‘E1’ link has 32 timeslots (0-31), of which:

- timeslot 0 is used for link management
- timeslots 1-15 are where incoming calls have their speech arrive
- timeslot 16 for signalling
- timeslots 17-31 are used by outgoing calls

The point codes are the same as in Example 1, so vssp has point code 1 and the attached switch has point code 2.

The circuit loop section of the **vssp.config** for this scenario would appear as follows:

```
connect 1/2/1 to 1/2/17.
connect 1/2/2 to 1/2/18.
connect 1/2/3 to 1/2/19.
connect 1/2/4 to 1/2/20.
connect 1/2/5 to 1/2/21.
connect 1/2/6 to 1/2/22.
connect 1/2/7 to 1/2/23.
connect 1/2/8 to 1/2/24.
connect 1/2/9 to 1/2/25.
connect 1/2/10 to 1/2/26.
connect 1/2/11 to 1/2/27.
connect 1/2/12 to 1/2/28.
connect 1/2/13 to 1/2/29.
connect 1/2/14 to 1/2/30.
connect 1/2/15 to 1/2/31.
```

Signalling is in timeslot 16. This will need to be sent out to UCAI.

Speech sent out on timeslot 1 must return on timeslot 17, so that the called party can hear the calling party. Similarly, the speech sent out on timeslot 17 must return on timeslot 1, so the caller can hear the called party. This is repeated with timeslot 2 and 18 and so on, up to timeslot 15 and 31.

Shorthand configuration

This can also be written as:

```
connect 1/2/1 to 1/2/17 for 15.
```

The “for 15” on the end instructs the UCAI that the `connect` is to be repeated for a total of 15 times, including the one specified in the line. Each time, UCAI adds 1 to the cics for both the incoming and the outgoing circuits.

The number after the “for” indicates the total number of circuits generated and not the number of additional circuits. It includes the circuit number specified in the line it is set in.

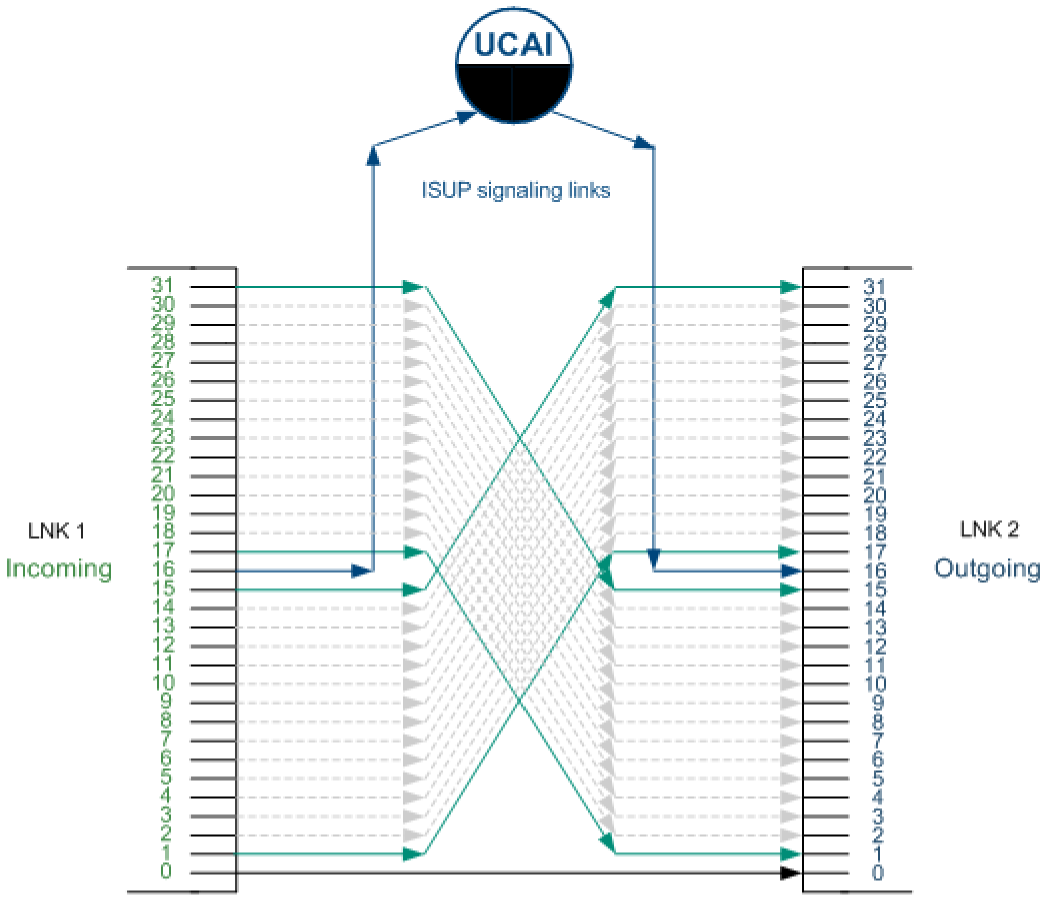
Care must be taken, as it is very easy to get the count of the circuits wrong. The final number of cics used is the first one, plus the number of repetitions minus one. For example, if you are starting at cic 1 and specify "for 2”, the final cic will be 2 and not 3.

Mistakes in this configuration will result in mismatches of conversations, for example:

- A one-way call (where one party can hear the other, but not the other way round)
- Different calls connected together
- (Most commonly) total silence at both ends

Diagram

The diagram shows the logical structure of how a loop box should be set up to handle Example 2. The speech paths are looped but the signalling is extracted and passed onto UCAI. The exact method for doing this will be very site-specific (several methods may even be employed in the same site).



Note: The connections for incoming timeslots 2 to 14 and 18 to 30 have been omitted, so the diagram is less cluttered.

Example configuration

This is an example of the **vssp.config** file (comments have been removed).

```
set option slee.
set option syslog.
set option stats.
set cdr file="cdr.log";
#set option debug.
#set option mtp debug.

set ica iam cpc 10;

connect 1/2/1 to 1/2/17 for 15;
```

Optimal configuration

Optimal configuration for the network depends on many variables, including:

- Types of service to be implemented
- Detailed estimates of the calling pattern to each service
- Trunking a call between switches
- Cost of trunk and signalling interfaces
- Network topology
- UCAI costs

It is beyond the scope of this document to detail the modeling required to produce an optimal solution.

The basic requirements for installation of the UCAI are the same regardless of the configuration chosen for a network. The requirements may be broken into the connection to the originating exchange, and the connection to the terminating exchange.

These may be subtly different from a signalling perspective, since each call arriving at the UCAI from the originating exchange may result in multiple sequential calls being made to the terminating exchange. For example, a call may be routed to an announcement prior to be routing to the eventual destination.

M3UA Configuration

Overview

This section introduces the configuration of the UCAI M3UA interface.

We will give a very simplified example of a M3UA network and its configuration to illustrate the fundamental concepts of UCAI's M3UA configuration.

In our example network we consider the last switch in a chain of SS7 switches. Everything further away from the UCAI will be pure SS7. For this example it will have a switch point code of 1.

Note: This chapter is not a substitution for a detailed knowledge of M3UA. A thorough understanding of the principles and techniques of M3UA is a vital prerequisite to successful configuration of the UCAI M3UA interface.

Switch configuration

As with standard configuration, UCAI needs loopbacks to be set up on the switch.

For this example, we will use two E1 links which are looped back to each other. (In other words, the transmit of the first link is connected to the receive of the second link and vice versa.) Signalling is not carried in the traditional timeslot 16, this timeslot is unused.

Elsewhere on the switch are signalling links that connect to the SS7-sigtran gateway (the signalling gateway (SG), commonly a Cisco ITP). It will have a point code of 2. These links have signaling on timeslot 16 but the other timeslots (1–15 and 17–31) are unused.

The switch has a link to the SG on local point code 1 and remote point code 2. It has 60 circuits. The first 30 are bi-directional, the second 30 are incoming only. These circuits are on the loopback that we built previously and not anything to do with the unused timeslots on the link to the SG. Although the linkset is between point codes 1 and 2, the point code of the remote end at the ISUP level is that of the UCAI (which we will define to be 3). This is the configuration normally used for ISUP if the traffic is being routed via some sort of SS7 router.

To summarize the configuration of the switch it has:

- Two E1 ports being used for the loopback (only carries speech, no signalling)
- A port being used signal to the SG (only carries signalling, no speech)
- Signaling is configured with a local point code of 1 and a remote point code of 2
- ISUP on the switch is configured to use the 60 speech circuits of the looped links
- ISUP DPC is set to be 3
- Point code of the UCAI and the switch is configured to route to DPC 3 via point code 2 (the SG)

SG (ITP) Configuration

The SG is assumed to be a Cisco ITP. To set it up, configure the baseline internet and the SS7 link. This provides the core SS7 and IP routing functions.

First set the port and IP address for the M3UA SCTP port, that is, the Signalling gateway Point (SGP) as follows:

Command	UCAI Mode

`cs7 m3ua 5000`
`local-ip 192.168.1.1`
	Loop Back Mode:
	The M3UA is configured on the SG.
	In this example:
	• SCTP port 5000 is used for the sigtran M3UA
	• ITP has an IP address of 192.168.1.1
	• IP address of the UCAI machine is 192.168.1.2

`cs7 m3ua 5000`
`    local-ip 192.168.1.1`
`cs7 m3ua 5001`
`    local-ip 192.168.1.2`
	Basic Inline Mode:
	In this mode, we define 2 local M3UA stacks.

VSSP configuration

The **vssp.config** for this example is:

```
connect 3/1/1 to 3/1/33 for 15;
connect 3/1/17 to 3/1/49 for 15;
```

This configuration indicates that the circuit with local (to the UCAI) point code 3, remote (to the UCAI) point code 1 and cic 1 is looped to circuit local 3, remote 1 and cic 33. This is repeated for the other 29 circuits skipping the unused timeslot 16.

When using M3UA, a second configuration file is also needed. This details the M3UA-specific configuration. For this example, it could be:

```
asp port 5000 addrs {192.168.1.2};
sgp itp1 port 5000 addrs {192.168.1.1};

gateway sg1
```

```
{
    sgp itp1 key 1234;
    pc {2, 1};
}

as pc 3 gateway {sg1};
```

This shows a local ASP on SCTP port 5000 with local address 192.168.1.2.

Note: There can only be a single port but it can be on multiple addresses. All the addresses given must belong to the local machine.

It also shows an ITP called itp1 with a port number of 5000 on the remote address of 192.168.1.1. Again, multiple addresses can be given and they must all belong to the remote SGP.

It also shows a signalling gateway called sg1. This contains the single SGP itp1 and identifies the routing key on the peer SGP. It also shows a list of point codes. The first is the point code of the SG itself (in this case 2) and any following point codes are those which can be reached via the SG. For this example, we specify that point code 1 may be reached via this SG.

It also shows our local AS. For this example, it has a point code of 3 and is peered with SG sg1.

M3UA configuration file syntax

Lexical considerations

The M3UA configuration file supports two sorts of comments:

- comment to end of line is indicated by a double forward slash (// with no space in between them).

  Example:
  ```
  Comment to end of line // This is the M3UA configuration file for the test
  network.
  ```
- comments that run over several lines are introduced by a /* and ended with a */.

  Example:
  ```
  /*
  * Now we set up the various SGPs in our network. There are three of
  * them.
  *
  * sgp1 --- located in The Fort this deals with its local area.
  * sgp2 --- located in The Hoist this is a hot standby for sgp1.
  * sgp3 --- located in Heaven this handles all international traffic.
  */
  ```

Whitespace

Whitespace is only used to separate other lexical items. Otherwise it is ignored.

Example: The following methods of defining an asp are identical.

```
asp port 5000 addrs {192.168.1.2};

asp
    port 5000
    addrs { 192.168.1.2 }
;
```

Example M3UA configuration file

This text shows an example M3UA configuration:

```
#start file
asp port 6969 addrs {192.168.26.117};

sgp itp1 port 2905 addrs {192.168.26.215};

gateway sg1
{
    sgp itp1 key 6969;
    pc {999, 4805, 4807};
}

as pc 4701 gateway {sg1};
#end file
```

Using more whitespace, this file could also be displayed as follows:

```
#start file
asp
    port 6969
    addrs {192.168.26.117}
;

sgp itp1
    port 2905
    addrs {192.168.26.215}
;

gateway sg1
{
    sgp itp1 key 6969;
    pc {999, 4805, 4807};
}

as
    pc 4701
    gateway {sg1}
;
#end file
```

Routing Options

Introduction

It is possible for attached switches to be configured so calls are only presented to the UCAI on circuits designated as “incoming” to the UCAI. There may be switches that do not allow routing on a per cic basis, and this section deals with several possible workarounds for this problem.

These options are necessarily, highly switch-specific, but one or more of them should be effective.

Per link fix

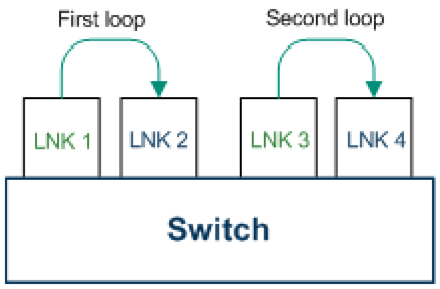
Some switches do not perform their routing based on point code. Instead they route based on sending calls down a specific link or linkset. In this case, the fix is fairly simple.

Routing should be configured so an entire link or linkset is designated as either all “incoming” to the UCAI or “outgoing” from the UCAI. Routing is then configured so calls are only routed down the links or link sets designated as “incoming” to the UCAI.

Example: A hypothetical mml command might be of the form:

```
ROUTE:PREFIX=0800,LINKS=LNK1,LNK3;
```

where the links LNK1 and LNK3 are designated as “incoming” to the UCAI and the unreferenced links (LNK2 and LNK4) are for the “outgoing” from the UCAI circuits. This would lead us to a loop-back structure looking something like this diagram.



Quite how the signalling would work in this situation is somewhat unclear. However, it is possible to do it with just the one signalling link, provided care is taken to assign the cics correctly.

One possible method would be to have LNK1 using cics in the range 1 to 31 (based on the timeslots) and then to have the rest shifted up by 32, so LNK2 would be 33 to 63, LNK3 65 to 95 and LNK4 97 to 127.

This would lead to a vssp.config file which looks like:

```
connect 1/2/1 to 1/2/33 for 31 except 15.
connect 1/1/65 to 1/2/97 for 31.
```

This assumes:

- The UCAI has point code 1 and the attached switch point code 2
- timeslot 16 of LNK1 carries the signaling.

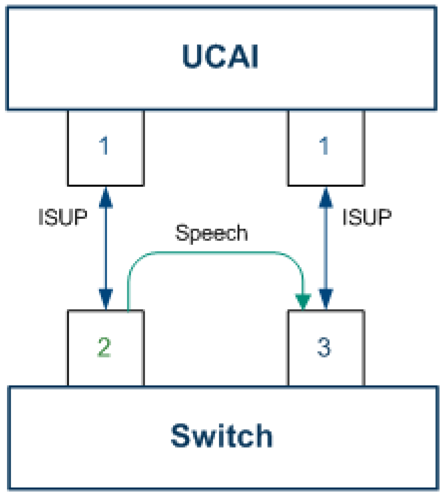
Multiple point code fix

Another possible option to work around routing problems in an attached switch, is to use multiple point codes on the attached switch. This means a switch may be able to support multiple links between itself and the UCAI.

These links may have different sets of point codes at the switch end. In this case, calls can be routed out from the switch to the UCAI from one point code, and be sent back into the switch from the UCAI on another point code.

Example:

If the UCAI has point code 1 and the switch has point codes 2 and 3, our network diagram would look something like this diagram.



Note: The major difference between this and the 'per link' fix is there are now two signalling links involved. The signalling for the traffic from the switch to the UCAI uses one link, while the signalling for the traffic coming from the UCAI to the switch uses the other.

Routing should be configured so this is the case. All traffic to the UCAI should be routed out of the switch's point code 2 link, to the UCAI's point code 1. No traffic should be routed out of the switch's point code 3.

A hypothetical mml command might be of the form:

```
ROUTE:PREFIX=0800,OPC=2,DPC=1;
```

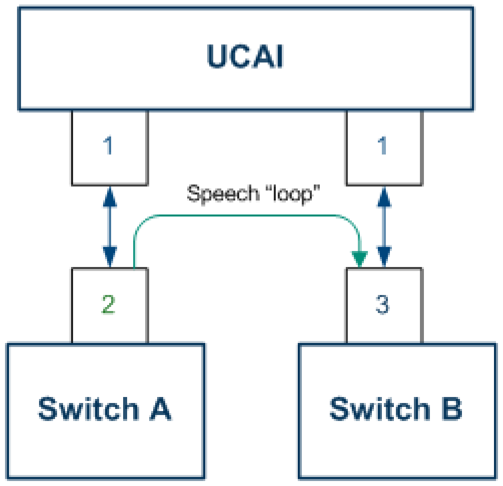
Both signalling links must also be configured so they can send signalling information. This is usually a lower-level operation than setting up routing information. The configuration for UCAI only requires connecting the inputs to the outputs:

```
connect 1/2/0 to 1/3/0 for 32 except 0 and 16.
```

This assumes timeslot 16 carries the signalling on both the links.

Two switch fix

This method solves the problem of dealing with routing issues, by placing the UCAI between two switches (called 'A' and 'B' in this example).



In this case, UCAI is simply modifying the signalling on an otherwise ordinary link between two switches. The timeslots associated with link management and speech (usually, all except 16) are left intact. UCAI takes the signalling out to be processed.

To implement this method, configure all the nodes to have their signalling point codes allocated according to the signalling requirements.

In the example:

- switch ‘A’ talks with its point code of 2 to a remote point code of 1
- switch ‘B’ talks with its point code of 3 to a remote point code of 1, and
- UCAI talks with its point code of 1 to both 2 and 3.

Switch ‘B’ is configured to route no calls to point code 1 and switch ‘A’ is configured to route calls to UCAI to point code 1.

This configuration can be performed on either a point code or a link basis (as illustrated in the previous examples). There is no requirement for excluding any timeslots from use (apart from the management and signaling ones, usually 0 and 16).

Configure UCAI with just 30 incoming and outgoing circuits, as follows:

```
 connect 1/2/0 to 1/3/0 for 32 except 0 and 16.
```

Configuring UCAI

Case studies

The following two case studies illustrate the configuration modes of the UCAI.

Loop back UCAI

A GSM mobile operator has invested in a network that does not support CS-1 IN features. To allow it to provide a prepaid function, the operator deploys a service node solution that requires all prepaid calls to be trunked to the service node platform for service logic and then back to the eventual destination.

This solution has obvious cost and scalability issues, but allows entry to the emerging prepaid mobile market. The operator is caught out by the unforeseen rapid growth of prepaid mobiles combined with the drastic reduction in charges due to strong competition. He now finds that the solution is prohibitively expensive and the limit of the solution’s scalability is reached.

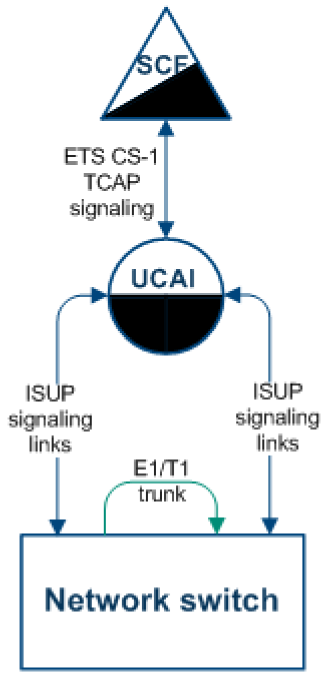
Two UCAIs are installed (co-resident on hardware with prepaid service logic SLC), connected to the operator’s SS7 network through STPs. Loop-back trunks are deployed on each of the operator’s mobile service switching centers (MSC), and signalling for these trunks is routed to UCAIs. The GSM HLR (Home Location Register) is configured with a service flag, to route calls from prepaid mobiles to UCAI trunks, allowing service logic to be invoked.

For mobile to mobile calls, the eventual destination of the call is not known, so routing between switches may be counter-productive, since the destination switch may need to route the call back or on to another switch. A possibility exists that, for calls to fixed phones or other mobile operators’ phones, the call could be triggered at the point of interconnect between the operators using an in-line UCAI. This would reduce the number of loop-backs required. However, the complexities of configuring switch routes that are based on calling party (being prepaid) and called party (whether fixed or other operator mobile), are considered too complex and costly in terms of man-power to be worth the saving in loop-back trunks.

Loop back UCAI example

This example shows a loop-back UCAI configuration. This configuration is optimal for triggering services:

- That often result in the call being terminated on the same switch from which it originated
- When the eventual destination for the call is unpredictable



In line UCAI

An application service provider (ASP), offering advanced services to other operators via their own switch, wishes to offer services beyond the capabilities of the AIN build of software currently available on the switch. The vast majority of the ASP's traffic originates from other networks, and traverses their switch to offer the service and route the call.

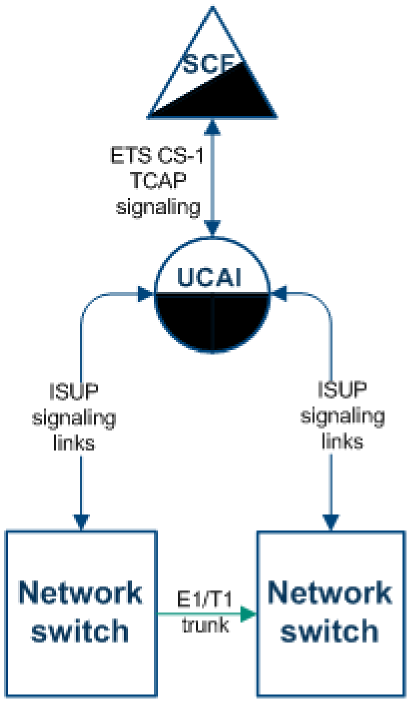
In this case, in-line UCAIs are very helpful, as most of the traffic is via points of interconnection. A UCAI placed on the signalling line of the other operator's switch and the ASP's network can control all incoming calls, avoiding loop-backs on the ASP switch entirely. Some loop-backs may be required for traffic originating on the ASP's network.

In line UCAI example

This example shows an in line or inter-switch UCAI configuration. This type of configuration is optimal for calls for which the vast majority of calls requiring UCAI processing originate from, or are eventually routed to, a network address that is not on the originating network switch, but which can be efficiently reached via the destination network switch.

Example: This diagram shows a network where:

- the originating switch is a local exchange (end office)
- the destination switch is a trunk level or international switch (central office), and
- the service to be triggered via the UCAI is a national or international freephone service.



Configuring IN Call Model Triggers

Overview

This introduces the generic configuration requirements of the NCC IN Call Model.

The NCC IN Call Model is not a separate product, rather it is a set of libraries that is bound into a final usable interface (such as the UCA-ISUP).

Environment variables

This table describes the UNIX shell environment variables to be configured.

Environment Variable Name	Description	Example Value
TDP_DEFINITIONS	Defines the full path name of the Trigger Detection Point definition file.	**/IN/service_packages/SLEE/etc/tdp.conf**

Trigger detection point (TDP) definition file

The **tdp.conf** file has two sections:

1. A number of configuration parameters.
2. The trigger tables used to determine when to trigger a call to the SCF.

Example: This text shows an example **tdp.conf** file:

```
# A comment
KEEP SD
ETC RULES=6 3
3 1 3 request all 123 6
```

```
 4 2 4 notify all 222 keep
 3 1 3 request 2:122 3:222 5 keep
```

Note: All lines starting with # are treated as comments. If no TDP definition file is defined, a default action is taken where:

- ALL calls are triggered to the SCF with a service key of 1 (one) and a trigger point of 3 (analyzedInformation.)
- None of the global configuration parameters are considered set.

Global configuration parameters

The following configuration parameters may be set once on individual lines in the TDP definition file.

Global Parameter	Description
KEEP SD	If defined ALL all stop digits (defined by the BCD digit 'F') on the end of called party numbers are kept in the called party number. By default the stop digit is stripped from ALL triggered numbers.
CAMEL	This parameter is intended for CAMEL testing purposes only and should not be defined under normal usage. If defined, the called party number is also copied into the intialDP's calledPartyBCDNumber CAMEL parameter. The NOA of the called party number becomes the BCD number type.
ADDITIONALNUMS	If defined, the IN Call Model will request all additional numbers available from the underlying protocol and insert them into the InitialDP message sent to the SLC. All these additional numbers are placed into a G8 extension in the InitialDP except any additional calling party number that is placed in the additionalCallingPartyNumber field.
ETC RULES= c or ETC RULES= c s	If defined then additional EstablishTemporaryConnection (ETC) rules are used. If the integer c is defined, the correlationID in all ETC messages from the SCF are appended on to the end of the assistingSSPIPRoutingAddress that is used, the digits are padded to a width of c digits. If s is also defined, then the scfID of the ETC is also appended on afterwards in the same way. **For example:** With "ETC RULES=6 4" and an ETC message with: assistingSSPIPRoutingAddress =1111, correlationID =55, scfID =0x42 Then the actual assistingSSPIPRoutingAddress used will be "11110000550042".
USER LIB = *library*	If defined the call model will use the user written shared object *library* specified by the full pathname library when dealing with ApplyCharging operations.
AC=*a,b,c....*	Sets the TCAP application context used by the call model to the comma separated list of OIDs supplied.
ORIG_PC= pc	If defined, all InitialDPs will be sent with an SCCP calling party (origination) address that includes a Point Code defined by the integer pc. If not defined, and ORIG_SSN and ORIG_GT are not defined, all InitialDPs will be sent without an SCCP calling party address. **Note:** This value may be defined in hex using a prefix of 0x.

Global Parameter	**Description**
ORIG_SSN= ssn	If defined, all initialDPs will be sent with an SCCP calling party (origination) address that includes a subsystem number defined by the integer ssn. If not defined, and ORIG_PC and ORIG_GT are not defined, all InitialDPs will be sent without an SCCP calling party address.

ORIG_GT=1, n, addr or
ORIG_GT=2, t, addr or
ORIG_GT=3, t, p, addr or
ORIG_GT=4, t, p, n, addr
	If defined, all initialDPs will be sent with an SCCP calling party (origination) address that includes a Global Title defined by the integers n, t, p and the number string addr.
	The initial value (1 to 4) identifies the Global Title type:
	• n is the NOA
	• t is the translation type
	• p is the numbering plan
	• addr is the address digits (0 to 9, A to F)
	
	If not defined, and ORIG_PC and ORIG_SSN are not defined, all InitialDPs will be sent without an SCCP calling party address.

DEST_PC= pc	If defined, all initialDPs will be sent with an SCCP called party (destination) address that includes a Point Code defined by the integer pc. **Note:** This value may be defined in hex using a prefix of 0x.
DEST_SSN= ssn	If defined, all initialDPs will be sent with an SCCP called party (destination) address that includes a subsystem number defined by the integer ssn.

DEST_GT=1, n, addr or
DEST_GT=2, t, addr or
DEST_GT=3, t, p, addr or
DEST_GT=4, t, p, n, addr
	If defined all initialDPs will be sent with an SCCP called party (destination) address that includes a Global Title defined by the integers n, t, p and the number string addr.
	The initial value (1 to 4) identifies the Global Title type:
	• n is the NOA
	• t is the translation type
	• p is the numbering plan
	• addr is the address digits (0 to 9, A to F)

ACH WARN PERIOD=*period*	Sets the default ApplyCharging warning to occur *period* seconds before the end of the call.
ACH RESOURCE=*ad*	Sets the default ApplyCharging warning announcement/tone to use the resource identified by the address digits *ad*. **Note:** This is only applicable if the underlying controlled call supports the ability to play announcements or tones.
ACH ANNOUNCE=*messageId*	Causes the default ApplyCharging warning to use the announcement with message identifier *messageId*. **Note:** This is only applicable if the underlying controlled call supports the ability to play announcements or tones.
ACS TONE=*id,dur*	Causes the default ApplyCharging warning to use tone with identifier *id* for a duration of *dur* seconds. **Note:** This is only applicable if the underlying controlled call supports the ability to play announcements or tones.

Note 1

If none of these entries is defined all InitialDPs will be sent without an SCCP calling party address.

Trigger detection point definitions

After any global parameters have been set, the configuration file may take one or more trigger detection point (TDP) definitions.

Each line defines a single trigger; its trigger parameter values that get sent and the conditions under which it gets sent.

Each line takes the following form:

```
tdp svcKey eventType msgType cgPn cdPn [wild] [keep]
```

The table below defines the meanings and forms of these parameters.

Global Parameter Value	Type	Description
tdp	integer	This integer value defines the point that the TDP is triggered at. Together with `cgPn`, *cdPn* and *wild* it defines the condition that the trigger will fire on. See the TDP event type table for a list of valid values and meanings.
svcKey	integer	This parameter defines the serviceKey value that will be inserted into the initialDP message when this trigger fires.
eventType	integer	This parameter defines the eventTypeBCSM value that will be inserted into the InitialDP message when this trigger fires. See the TDP event type table for a list of valid values and meanings. Generally this will be the same value as *tdp*.
msgType	request or notify	This parameter defines whether the TDP is sent as a TDP-R (request) or TDP-N(notify). Generally request is used here.
cgPn	*num* or *nat*:*num* or all	This parameter defines the calling party numbers that will trigger the TDP. Together with *tdp*, *cdPn* and *wild* it defines the condition that the trigger will fire on. • *num* defines the prefix of the calling party digits, numbers must begin with these digits for the trigger to fire. • *nat* is optional and defines additionally a nature of address (NOA) of the calling party that must match for the trigger to fire. If not provided a nature of 2 (unknown) is assumed. If all is defined then ALL calling party numbers will match.
cdPn	*num* or *nat*:*num* or all	This parameter defines the called party numbers that will trigger the TDP. Together with *tdp*, *cgPn* and *wild* it defines the condition that the trigger will fire on. • *num* defines the prefix of the called party digits, numbers must begin with these digits for the trigger to fire.

Global Parameter Value	Type	Description
		• *nat* is optional and defines additionally a nature of address (NOA) of the called party that must match for the trigger to fire. If not provided a nature of 2 (unknown) is assumed. If all is defined then ALL called party numbers will match.
wild	integer	This optional parameter defines the number of digits that must be present in the called party numbers before the TDP will trigger. Together with *tdp*, *cgPn* and *cdPn* it defines the condition that the trigger will fire on. If set the trigger will not fire until the called party number has this number of digits. **Note:** The *wild* parameter can be set to a special value of "stop". If it is set to this value, then the trigger will only fire when a stop digit is received.
keep	-	If this optional flag is defined then all numbers triggered by this TDP will keep their stop digits (if they have one).

TDP event type values

The following table defines the list of TDPs as defined by the CS-1 standard. It also defines the point at which the trigger will be instantiated by the NCC IN Call Model.

TDP	CS-1 Trigger Name	Call Model TDP Creation Point
1	origAttemptAuthorized	digitsReceived
2	collectedInfo	digitsReceived
3	analyzedInformation	digitsReceived
4	routeSelectFailure	released (cause != 16, 17, 18, 19, 21 or 31)
5	oCalledPartyBusy	released (Aparty, cause==17)
6	oNoAnswer	released (Aparty, cause==18, 19 or 21)
7	oAnswer	answered(Aparty)
8	oMidCall	not supported
9	oDisconnect	released (Aparty, cause==16 or 31)
10	oAbandon	released (Aparty, cause==16 or 31)
12	termAttemptAuthorized	digitsReceived
13	tCalledPartyBusy	released (Bparty, cause==17)
14	tNoAnswer	released (Bparty, cause==18, 19 or 21)
15	tAnswer	answered(Bparty)
16	tMidCall	not supported
17	tDisconnect	released (Bparty, cause==16 or 31)
18	tAbandon	released (Bparty, cause==16 or 31)
100	n/a	ringing (Aparty)

TDP	CS-1 Trigger Name	Call Model TDP Creation Point
101	n/a	ringing (Bparty)

Chapter 3

Background Processes

Overview

Introduction

This chapter explains the processes that are started automatically by Service Logic Execution Environment (SLEE).

Note: This chapter also includes some plug-ins to background processes which do not run independently.

In this chapter

This chapter contains the following topics.

vssp

Purpose

vssp is the binary that runs the UCAI application.

No other binaries are installed with UCAI.

Startup

This task is started by the SLEE, by the following lines in **SLEE.cfg**:

```
INTERFACE=VSSP vssp.sh /IN/service_packages/VSSP/bin EVENT
```

Notes:

The actual startup script name may vary.

If you configured UCAI to run without the SLEE, you can also start UCAI from the command line.

Configuration

vssp accepts the following parameters from **vssp.config**.

```
unset option slee.
set option syslog.
set option stats.
set cdr file="filename";
set option debug.
set option mtp debug.
set option stats.
set option overload backoff period to 1 seconds;
set startup time seconds.
set network network_identifier.

filter iam drop [{number,} number and] number.
```

```
connect circuit to circuit [for number_of_circuits].
```

M3UA configuration

vssp supports these parameters from **m3ua.config**.

```
asp
    port port
    addrs {ip_address}
;

sgp itp1
    port port
    addrs {ip_address}
;

gateway sg1
{
    sgp itp1 key int;
    pc {int, [int, ...]};
}

as
    pc int
    gateway {sg1}
;
```

Command line parameters

vssp supports the following command line parameters:

Note: These parameters are usually specified in the startup script, **vssp.sh**.

```
vssp -c configuration_file [-k keyfile] [-m m3ua_config_file]
```

where:

Parameter	Description
-c *configuration_file*	The path and name of the vssp configuration file.
-k *keyfile*	The path and name of the keyfile file, which sets the maximum number of circuits UCAI can use. The default is a file named **keyfile** in the same directory as the vssp configuration file.
-m *m3ua_config_file*	The path and name of the M3UA configuration file. The default is a file named **m3ua.config** in the same directory as the vssp configuration file.

Standard options

The following parameters are available for all types of UCAI installation.

`cdr file`

Syntax:	`set cdr file="`*`filename`*`";`
Description:	When set, defines the directory and filename to which vssp logs CDRs.
Type:	String
Optionality:	Optional (default used if not set)
Allowed:	
Default:	Not set

Notes:	The equals sign (=) is optional.

Examples:	`set cdr file="cdr.log";`
	`set cdr file "cdr.log";`

`connect`

Defines the circuit loops for vssp. For more information about configuring circuit loops, see *Configuring circuit loops* (on page 18).

Default:	none
Allowed:	`connect` *`circuit`* `to` *`circuit`*`[ for` *`number of_circuits`*`][ except` *`cic`*`[ and` *`cic`*`]].`
Note:	The full stop or semi-colon on the end is essential if you are not specifying a number of circuits.

`debug`

Turns vssp debug output on or off. If enabled, an operator can debug SSP functionality, as well as lower-level ISUP functionality.

The SSP functionality details how calls are being progressed and the ISUP debugging prints out all the ISUP messages sent and received.

Default:	off
Format:	`set option debug`
Notes:	This output is sent out on the debugging output of the VSSP (stdout).

`filter`

Drops specified parameters within an IAM. Drops the parameters with the name value of any of the given *number*s on an outgoing IAM message.

Default:	empty set
Format:	`filter iam drop [{`*number*`,}` *number* `and]` *number* **Where:** • Square brackets enclose optional entries • Curly brackets enclose zero or more entries
Note:	Mandatory parameters cannot be dropped.

`limit`

Sets a CAPS limit on vssp. The first number is the CAPS limit and the second is the release cause to be used if the CAPS limit is exceeded.

Default:	Cause defaults to 42.
Format:	`set option limit to` *`number`* `caps cause [`*`cause_number`*`]`
Notes:	The `to` can be replaced with an equals sign. *cause_number* is optional.

`mtp debug`

Turns mtp traffic output from vssp on or off. If on, a hexadecimal representation of all the ISUP messages is written to the vssp's debugging output (stdout).

Default:	off
Format:	`set option mtp debug`

`network`

Sets the network identifier in the MTP3 header to be the specified value.

Default:	2
Note:	This only applies to the M3UA and Data Kinetics stacks. 2 is national network.

`set ica iam cpc`

Sets the calling party category on the legs of ICA calls. The CPC can be set on just the A-leg or on both legs.

Default:	none
Format:	`set ica iam cpc` *`A-leg_number`* `[`*`B-leg_number`*`]` Where: • *number* is the override value for the calling party category.
Notes:	By default, the CPC set for the A-leg number will be used for the B-leg. There is an NCC extension to INAP that allows the CPC to be set on the A-leg of an ICA-created call. Service logic may override a set B-leg value.

`overload backoff period`

Syntax:	`set option overload backoff period =` *`int`* `seconds;`
Description:	Sets the time period in seconds during which vssp rejects new requests to create a SLEE dialog and returns an error.
Default:	1
Allowed values:	0 to 4294967295
Notes:	• vssp marks itself as overloaded and rejects new initial address messages (IAMs) for the configured backoff period, when it fails to create a SLEE dialog. • 0 means no overload limit.
Example:	`set option overload backoff period = 10 seconds;`

`slee`

Enable or disable the interface between vssp and the SLEE. If the SLEE interface is disabled, vssp acts simply as a call router, in effect providing a null service.

Default:	on
Format:	`unset option slee`
Note:	This option should only be used for debugging.

`startup time`

Seconds vssp takes to reset all its circuits at start up.

Default:	15
Allowed:	number
Format:	`set startup time =` *`seconds`*
Note:	If there are a large number of circuits, the default may need to be increased to avoid temporary congestion of the network. The equals sign (=) is optional.

`stats`

If set, vssp will log a basic calculation of the number of calls being processed by vssp per second.

Default:	off
Format:	set option stats

`syslog`

Send error messages to the syslog as well as UNIX stderr.

Any errors will now be logged to syslog. The syslog identifier is set to VSSP, the facility is set to LOG_USER and errors are logged at level LOG_ERR.

Default:	off
Format:	`set option syslog`

M3UA parameters

The M3UA configuration uses a cascading definition structure. The first parameter (file) is defined by four other parameters. Those four parameters are in turn defined by other parameters set later in the file.

For more information about how these parameters work together to configure vssp, see *M3UA Configuration* (on page 21).

vssp supports these parameters from the **m3ua.config** file.

Note: These parameters are only available if you have installed the M3UA version of UCAI.

`as`

Application Server definition. Assigns ASPs to ASs.

Default:

Syntax: `as pc` *number* `gateway {` *gateway_list*`};`

Where:

- *number* is the Point Code number of the AS
- *gateway_list* is the list of SGs the AS is peered with.

Note:

`asp`

Defines the local ASP.

Default:

Syntax: `asp port` *port* `addrs {`*ip*`};`

Where:

- *port* is the port the AS process is running on
- *ip* is the IP address the AS process is running on

`gateway`

Gateway definitions.

Default:

Syntax:

```
gateway identifier
{
     sgp itp1 key key
     pc {pcs};
}
```

Where:

- *identifier* is a unique identifier
- *key* is a keyed_sgsp parameter
- *pcs* is a list of comma separated point codes

`sgp`

Signalling Gateway Process definitions.

Default:

Syntax: `sgp itp1 port` *port* `addrs {`*ip*`};`

Where:

- *port* is the port the SG process is running on
- *ip* is the IP address the SG process is running on

Example configuration

This is an example of the **vssp.config** file (comments have been removed).

```
set option slee.
set option syslog.
set option stats.
set cdr file="cdr.log";
#set option debug.
#set option mtp debug.

set ica iam cpc 10;

connect 1/2/1 to 1/2/17 for 15;
```

Example M3UA configuration file

This text shows an example M3UA configuration:

```
#start file
asp port 6969 addrs {192.168.26.117};

sgp itp1 port 2905 addrs {192.168.26.215};

gateway sg1
{
    sgp itp1 key 6969;
    pc {999, 4805, 4807};
}

as pc 4701 gateway {sg1};
#end file
```

Using more whitespace, this file could also be displayed as follows:

```
#start file
asp
    port 6969
    addrs {192.168.26.117}
;

sgp itp1
    port 2905
    addrs {192.168.26.215}
;
```

```
gateway sg1
{
    sgp itp1 key 6969;
    pc {999, 4805, 4807};
}

as
    pc 4701
    gateway {sg1}
;
#end file
```

Output

vssp writes error messages to the system messages file, and also writes additional output to the following default:

/IN/service_packages/VSSP/output/outputFile.log

Note: May vary as per configuration in the startup script.

Chapter 4

Troubleshooting

Overview

Introduction

This chapter explains the important processes on each of the server components in NCC, and describes a number of example troubleshooting methods that can help aid the troubleshooting process before you raise a support ticket.

In this chapter

This chapter contains the following topics.

Common Troubleshooting Procedures

Introduction

Refer to *System Administrator's Guide* for troubleshooting procedures common to all NCC components.

Turning on debug

The vssp process supports debug. To turn on debug for vssp, set the debug option in the **vssp.config** file and reload the configuration.

For information about turning on debug, see *debug* (on page 37).

Important: Turning on debug increases the load on the vssp process. It should only be used on a production platform when absolutely necessary.

Turning on M3UA vssp debug

The vssp process supports debug for its M3UA. To turn on M3UA debug for vssp, set the mtp debug option in the **vssp.config** file and reload the configuration.

For information about turning on debug, see *mtp debug* (on page 37).

Important: Turning on debug increases the load on the vssp process. It should only be used on a production platform when absolutely necessary.

Checking current processes

You can check which processes are running using the standard UNIX command: ps. To find processes being run by Oracle software, you can grep for the string 'oper', which will display all processes being run by the application operator accounts (for example, acs_oper, ccs_oper and smf_oper).

Note: Some processes which are required for proper functioning may be run by other users, including root or the user which runs the webserver.

Example command: `ps -ef | grep oper`

For more information about the ps command, see the system documentation for the ps command.

You can also check how much of the processor a process is using by running the standard UNIX tool: top. If you have some baseline measurements, you will be able to compare it with the current load.

Example command: `top`

Tip: Some processes should only have one instance. If there are two or more instances, this may indicate a problem. For example, there will usually only be one timerIF running on each SLC.

For more information about which processes should be running on each node, check the Process List for each node in *Installation Guide*.

Checking installed packages

To check the details of an installed package, use the `pkginfo` command.

Example command: `pkginfo -l smsSms`

Example output: This is an example of the output of the example command above.

```
   PKGINST:  smsSms
      NAME:  Oracle smsSms
  CATEGORY:  application
      ARCH:  sun4u
   VERSION:  3.1.0
    VENDOR:  Oracle
    PSTAMP:  smsNode20041020104925
  INSTDATE:  Oct 20 2004 13:15
     EMAIL:  support@oracle.com
    STATUS:  completely installed
     FILES:      348 installed pathnames
                  39 directories
                  89 executables
              152448 blocks used (approx)
```

For more information about the `pkginfo` utility, see the system documentation.

Checking network connectivity

Network connectivity will affect any process which requires communication between two different network addresses.

Network connectivity should support ssh sessions between the two machines experiencing the problem.

If you can open an ssh session between the two machines, check the following before contacting Level 1 support with details:

- If the address of either of the machines specified in the Node Management screens is a hostname, check that the hostnames used in the ssh sessions are the hostnames specified in the Node Management screen.

If you cannot ssh, check the following before contacting Level 1 support with details:

- Check that the hostname is resolving correctly in the DNS.
- Check that the physical network connection is working correctly.
- Check that the inetd and sshd are running.
- Check that sshd is listening on the expected port.
- Check that the smf_oper and acs_oper accounts are not locked, and that the username and password combinations being used are correct.

Checking configuration files

One of the significant areas where faults can occur and be remedied is in the configuration of processes. Configuration files can be edited by any standard text editor. A backup of the existing configuration file should always be taken before editing a configuration file.

For more information about the configuration files used in this application, see *Configuration User's Guide*.

For more information about the configuration file for a specific program or tool, see the section named after the binary in question.

Possible Problems

Introduction

This topic lists common problems and actions you can take to investigate or solve them. This list enables you to check for alarms based on the overall behavior you are experiencing.

Circuit mismatches

If any of the following are true, it may indicate that the circuit loop configuration is mismatched:

1 One party can hear the other, but not the other way round.
2 Different calls connected together.
3 Total silence at both ends.

Check the **vssp.config** file for possible mismatches. For more information about configuring circuit loops, see *Configuring circuit loops* (on page 18).

Chapter 5

About Installation and Removal

Overview

Introduction

This chapter provides information about the installed components for the Oracle Communications Network Charging and Control (NCC) application described in this guide. It also lists the files installed by the application that you can check for, to ensure that the application installed successfully.

In this Chapter

This chapter contains the following topics.

UCAI Requirements

ISUP Signaling Link Set Dimensioning

Originating Exchange Signaling

The signaling capacity requirements for the SS7 links to the originating network switch are dependent on the following parameters:

- BHCA for service
- Required normal network link load
- Average number of octets send and received per call

The bit rate capacity can be estimated as:

r=(*x*.*b*)/450

Where:

- *x* = the BCHA number
- *b* = number of octets per call (sent or received)

Example:

Assuming 100 octets per call (both sent and received), a service with 300000 BHCA would require 66.7 Kbits per second.

If 64Kbit s^{-1} links are used, and a link load of 60% is required, a link would support 38.4 Kbit s^{-1}.

Therefore, 2 links would be required to support the load. For redundancy, an additional link is normally installed unless the link load figure takes account of this.

ISUP Signalling Link Set Dimensioning

Terminating Exchange Signaling

Since the terminating trunk signaling may need to establish and clear more than one call per call to the service, the terminating exchange signaling requirements are a multiple of the originating signaling requirements.

The average number of calls that must be established as a result of each incoming call determines the required multiple.

Example: If each service invocation requires a call to be established to an SRF and then routed to final destination, a multiplication factor of 2 would be applied to determine the terminating exchange signaling requirements.

Trunk dimensioning

The number of trunks that must be configured is totally dependent on the number of simultaneous calls UCAI must handle.

This is dependent on traffic forecasts and quality of service (QoS) data. If a service is expected to have x busy hour call attempts (BHCA), with each call lasting an average of y seconds, then the number of simultaneous calls C is:

c=(x.y)/3600

The number of VSSP trunks is the number of simultaneous calls divided by the number of speech channels in a trunk (E1=30, T1=24).

When QoS is taken into consideration, the number of trunks will be increased, to allow for the non-uniform profile of real service usage.

Example: A service requires 10000 BHCA and has an average call hold time of 2 minutes. The number of simultaneous calls is therefore 3334, which is 14 T1 trunks. To achieve the desired QoS requires a 30% over provisioning of capacity, which results in 19 T1 trunks.

Installation and Removal Overview

Introduction

For information about the following requirements and tasks, see *Installation Guide*:

- NCC system requirements
- Pre-installation tasks
- Installing and removing NCC packages

UCA-ISUP package

An installation of UCA-ISUP includes the following package, on the SLC:

- VSSP

Add vssp entry to SLEE.cfg

Configure UCAI to use the SLEE by adding an entry for the vssp process in **SLEE.cfg**.

Unless the setup of this installation of UCAI is unusual, the line should be:

```
INTERFACE=VSSP vssp.sh /IN/service_packages/VSSP/bin EVENT
```

For more information about adding entries to **SLEE.cfg**, see *SLEE Technical Guide*.

Checking the Installation

Introduction

Refer to these checklists to ensure that UCAI has installed correctly.

Checklist

Follow the steps in this checklist to ensure UCAI has been installed on a SLC machine correctly.

Step	Action
1	Log in to SLC machine as root.
2	Check the following directory structure exists with subdirectories: **/IN/service_packages/VSSP**
3	Check the directory contains subdirectories and that all are owned by: acs_oper user (group oracle)
4	Check the entries of following file: **/IN/service_packages/SLEE/etc/SLEE.cfg** SLEE startup entries for VSSP on a SLC: `INTERFACE=VSSP vssp.sh /IN/service_packages/VSSP/bin EVENT`
5	Check that the processes listed in the process lists are running on the relevant machine. For a list of the processes which should be running, see *Process list* (on page 49).

Process list

If the application is running correctly, the following processes should be running on each SLC:

- Started during SLEE startup:
  - vssp