Oracle® Communications Convergent Charging Controller

Messaging Manager Technical Guide

Release 15.1

April 2025

Copyright

Contents

About This Document

Scope

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

Audience

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

Prerequisites

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

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. 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.

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

Related Documents

The following documents are related to this document:

- *Advanced Control Services Technical Guide*
- *Event Detail Record Reference Guide*
- *Messaging Manager Navigator Technical Guide*
- *Messaging Manager User's Guide*
- *Session Control Agent Technical Guide*
- *Service Logic Execution Environment Technical Guide*
- *Service Management System Technical Guide*

Document Conventions

Typographical Conventions

The following terms and typographical conventions are used in the Oracle Communications Convergent Charging Controller 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 Convergent Charging Controller network or service implications of the product.

In this Chapter

This chapter contains the following topics.

What is Messaging Manager?

Introduction

Messaging Manager (MM) is a messaging system for mobile networks. It acts as a Virtual Message Point (VMP) for a variety of different messaging traffic (for example: SIP, email, and SMS). Depending upon the role that it is performing, the VMP can act as any of the following:

- Message Service Center (MSC)
- Short Message Entity (SME) that terminates and/or originates messaging traffic
- Email host.

Messaging Manager integrates advanced routing and protocol delivery options with extended service control, in order to support all forms of traditional MO SMS and MT SMS services while retaining flexible support for new types of messaging.

Processing model

Messaging Manager's architectural approach as a Virtual Message Point means that all messaging involves transactions that can combine real time charging with direct delivery to the destination. This is the "new messaging model" that is aligned with the Internet age, and replaces the previous "store-and-forward" model with higher value and lower cost infrastructure.

The VMP processes all message services in real time, but it can integrate transparently with an existing SMSC for store-and-forward processing when real time delivery is not possible. It delivers:

- High capacity messaging on low cost infrastructure
- Very flexible switching and routing serving a multiple purposes
- Proven efficiencies using real time charging and delivery
- Enhanced message services using a service creation environment (SCE)
- Performance gains over existing SMSC infrastructure
- An enhanced customer experience

Messaging Manager provides a broad range of message processing capabilities at both the network layer and at the service layer. To the network it presents standard signaling interfaces to act in the role of:

- SMS-IWMSC (SMS Inter-Working MSC)
- SMS-GMSC (SMS Gateway MSC)
- HLR (proxy and emulation services)
- Email host (with SEI)

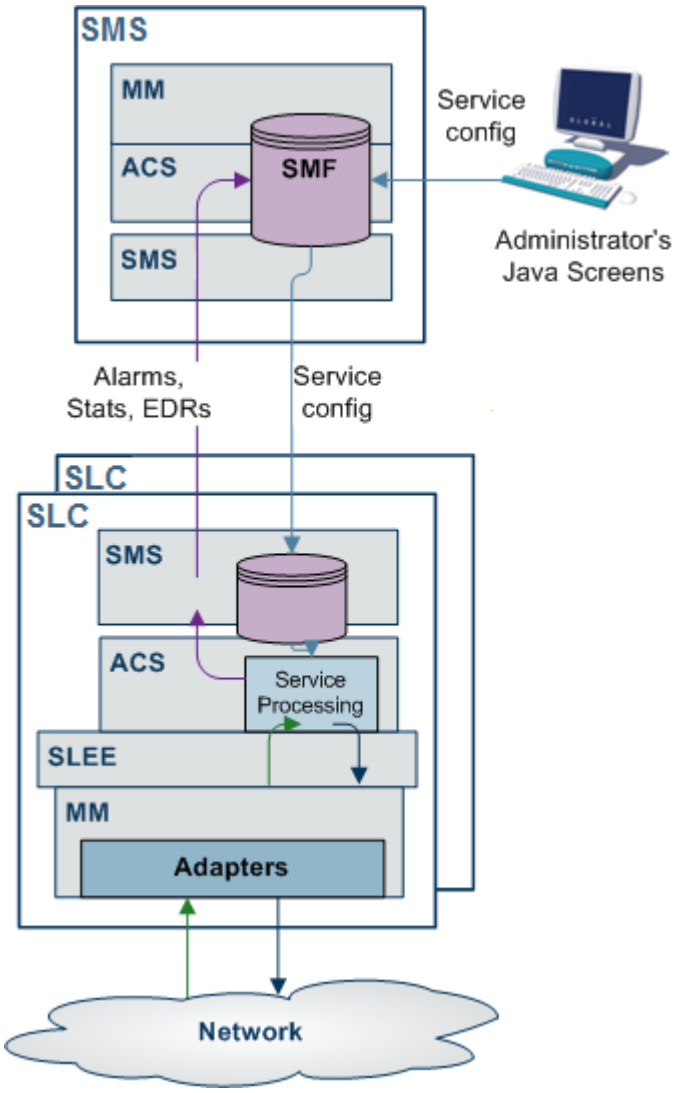
By performing multiple functions in one system, Messaging Manager simplifies the messaging infrastructure and frees up resources.

Messaging Manager components operating at the network layer can route traffic from one communication path to another and will automatically perform protocol translation based on the inbound and outbound communication paths. This can be statically configured through the management UI but can be dynamically overridden during transaction processing by the service control layers. Typically all message traffic arriving at the VMP is processed for charging (if necessary) then immediately directed to the destination. This process of delivering directly to a destination is known as First Delivery Attempt (or FDA).

When huge traffic spikes occur (such as during holiday peaks, or events such as televoting) Messaging Manager can absorb the load and groom traffic to provide smooth processing and near real time delivery.

Deployment diagram

This diagram shows the Messaging Manager deployment architecture.



Messaging Manager features

Messaging Manager provides the following features:

- FDA (First Delivery Attempt). SMS are directly delivered (through SS7) without going through an SMSC.
- Overload protection from SMS traffic peaks, for example, special events (New Year) or application peaks time (televoting). MM, enables you to offload your SMSCs and protect them from traffic peaks. This enables you to extend your capability to handle extreme traffic peaks in an efficient way.
- Value-added SMS services. These include:
  - Flash messaging
  - Auto-reply
  - Anti-spam
  - SMS copy to mobile or email
  - SMS-MT forwarding

  - Voting campaigns
- Real-time charging
- The ability to provide SPOC (Single Point of Contact) to ASPs to attract more ASPs on your networks and to differentiate your offering on value-added interactive applications

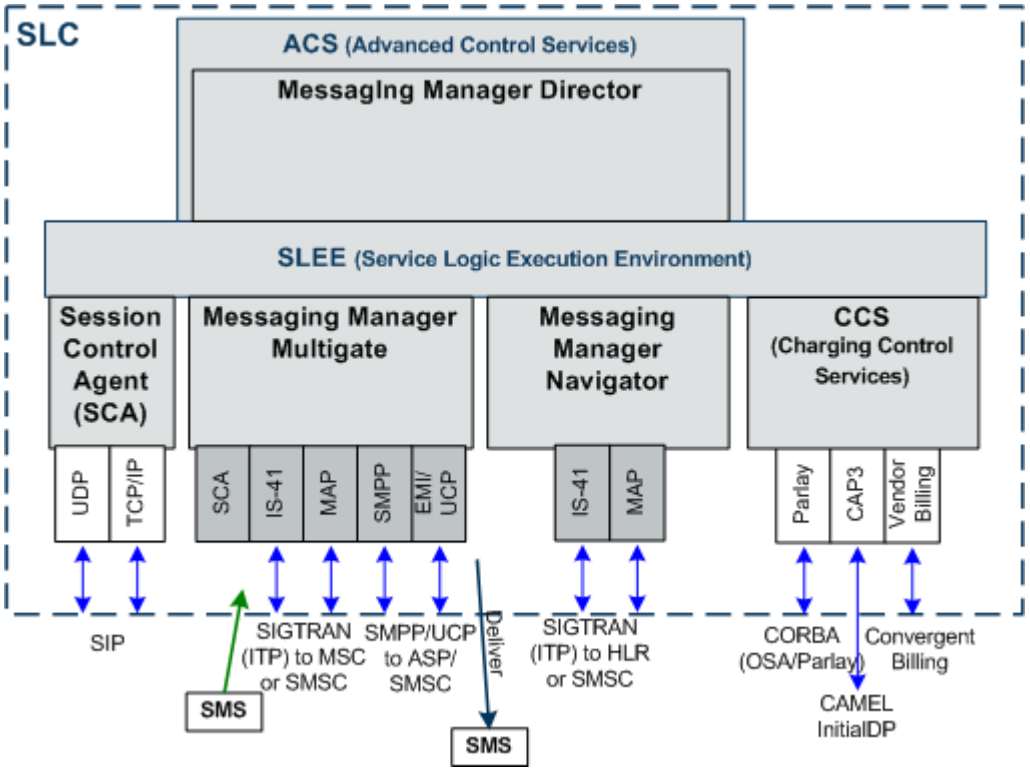
Messaging Manager components

The major Messaging Manager components are:

- **Messaging Manager Multigate:** A multi-protocol gateway and multi-function router that can receive and send short messages. Its layered architecture allows all signaling and IP protocols to connect to a common set of service logic, maintaining independence between transport protocols and the user-defined routing scheme that defines the messaging model. For a full description of this component, see *Messaging Manager Multigate* (on page 6).
- **Messaging Manager Navigator:** A Mobile Station location service that can perform and/or emulate HLR lookups by other components or network elements, caching the results to optimize network signaling and direct SMS transmission toward service logic. For a full description of this component, see *Messaging Manager Navigator Technical Guide*.
- **Messaging Manager Director:** A set of service control feature nodes that execute as a message control plan and provide enhanced logic for message delivery, routing, and charging and offers extended message attribute controls. For a full description of this component, see *Messaging Manager Director* (on page 8).
- **Messaging Manager Manager:** A central UI for management of routing schemes and message control plans that are used to configure and control all service logic components. For a full description of this component, refer to *MM User's Guide*.

Virtual Message Point components

This diagram shows the service execution components of Messaging Manager that implement the VMP, along with other platform support components.



Protocols supported

This table describes the function of each field.

Protocol	More Information
SMPP	*Configuring the SMPP Adapter* (on page 103)
EMI	*Configuring the EMI Adapter* (on page 97)
MAP	*Configuring the MAP Adapter* (on page 75)
IS-41	*Configuring the IS-41 CDMA Adapter* (on page 117) *Configuring the IS-41 TDMA Adapter* (on page 129)
SIP	*Configuring the SCA Adapter* (on page 130)

Messaging Manager Platform

Introduction

The Messaging Manager platform is required to provide basic transaction and switching functionality for the MM message service. It is a prerequisite to any additional MM components.

It provides the following functionality:

- SMS transaction management for end to end delivery
- Transaction throughput control
- Notification and status reports
- Real time convergent billing services
- Flexible multi-party billing
- Support for all standard SMS transport options
- Support for mapping between transport protocols
- Connection level load-balancing and redundancy
- Subscriber information partitioning
- Statistics generation
- Alarm generation
- CDR generation
- Provisioning and administration

Note: Collection of statistics requires the Convergent Charging Controller Service Management System (SMS) application.

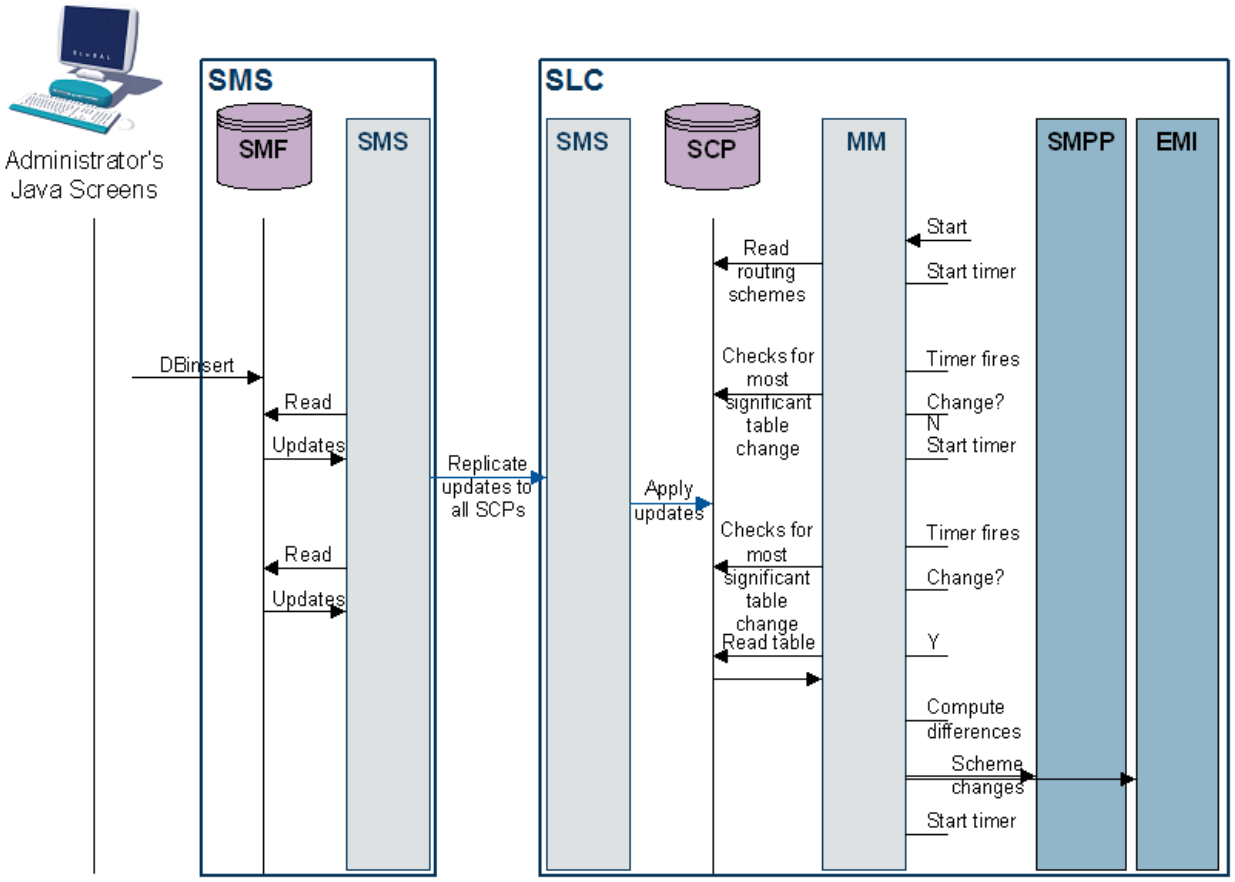
Using the Messaging Manager platform

Messaging Manager provides a platform from which multiple advanced messaging services, including but not limited to SMS-MO and SMS-MT, may be designed and implemented.

This messaging platform leverages the Convergent Charging Controller SLC Service Logic Execution Environment (SLEE). The SLEE manages each message event coming into and going from MM and provides control of these messages between the network layer and the applications. SLC is designed to maintain integrity, simplify management and ensure high performance when servicing massive messaging volumes from various underlying networks to multiple application services.

Data life cycle

The diagram below shows a time line (down the diagram) of activities that occur for the replication process.



Messaging Manager Multigate

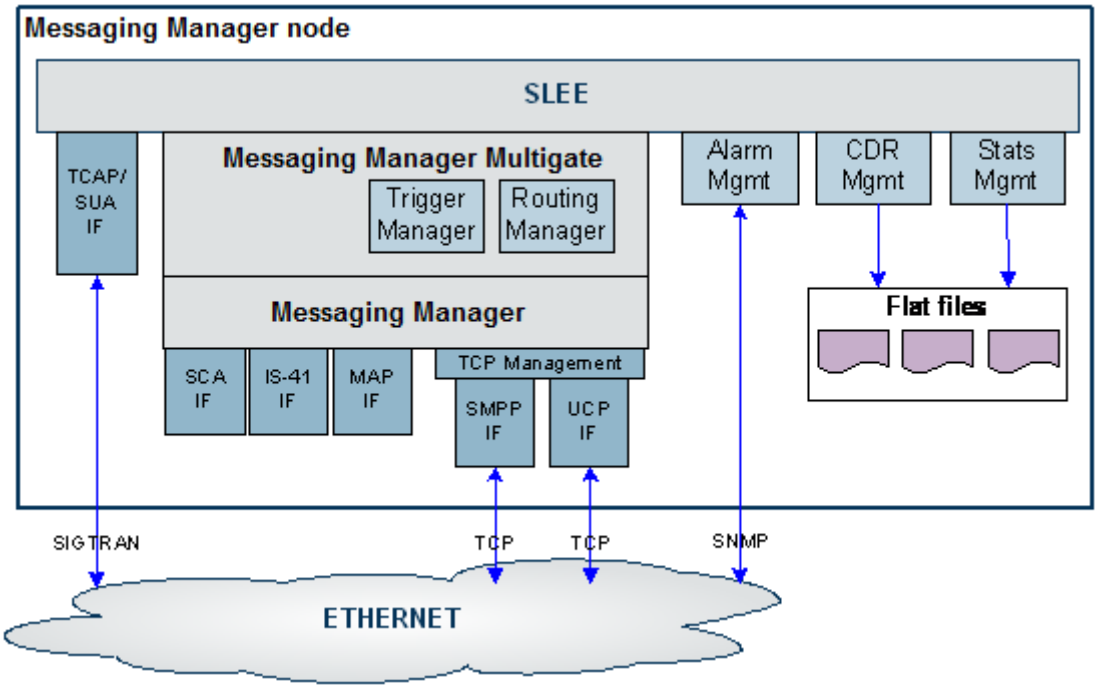
Introduction

Multigate is the core VMP component that provides multi-protocol message handling. It employs a "message model" abstraction that gives enormous power to the service designer in classifying, filtering and routing message traffic. Multigate provides the message delivery and retry logic driven by the message model and dynamic changes made by Messaging Manager Director. The following features are provided:

- Routing for all types of SMS, including protocol translation
- High speed criteria-based classification/filtering/switching
- First Delivery Attempt (FDA) to a destination handset or ASP
- Alternate delivery options for conditional and/or optimal routing
- Forwarding to a specified SMSC or ASP through a load weighted group
- Service logic triggering for charging and enhanced message services
- Service level OA&M support (statistics and alarms) and EDR management

Diagram

Here is a diagram showing the Messaging Manager Multigate module.



Using Messaging Manager Multigate

The Messaging Manager SMS Multigate module provides:

- Standard SMS-MO service support
- Standard SMS-MT service support
- SMS-MO switching from MS to ASP with First Delivery Attempt
- SMS-MO switching from MS to MS with First Delivery Attempt
- SMS-MT switching from ASP to MS with First Delivery Attempt

Routing options

Using the Messaging Manager Multigate module, MM allows individual routing based on different criteria:

- Incoming protocol, message center, originating number, or originating domain
- Destination number or destination domain

Triggering

The Messaging Manager Multigate module allows triggering of service control logic based on the following criteria:

- Incoming protocol
- Originating number or originating domain
- Destination number or destination domain

ASP connection state management

MM maintains a state for each ASP. The state defines how MM will handle the data and command flow between MM and the ASP.

Note: When a TCP connection is dropped by the ASP, the ASP can not immediately reconnect because the TCP/IP stack is in a "cleaning up" state (TIME_WAIT).

Messaging Manager Director

Introduction

Messaging Manager Director provides complete control over all aspects of the VMP services. Its advanced service control facilities enable extended and customised SMS processing, including real-time billing interaction, by supporting user defined message control plans.

Message control plans can be triggered from Messaging Manager Multigate and include service logic based many properties, such as:

- Incoming path names (that is, protocols and connections)
- Transaction types, such as Submit, Deliver, Notify or Route Info messages
- Originating and/or destination address
- Location of originating and/or destination mobile station
- Message content
- Time of day.

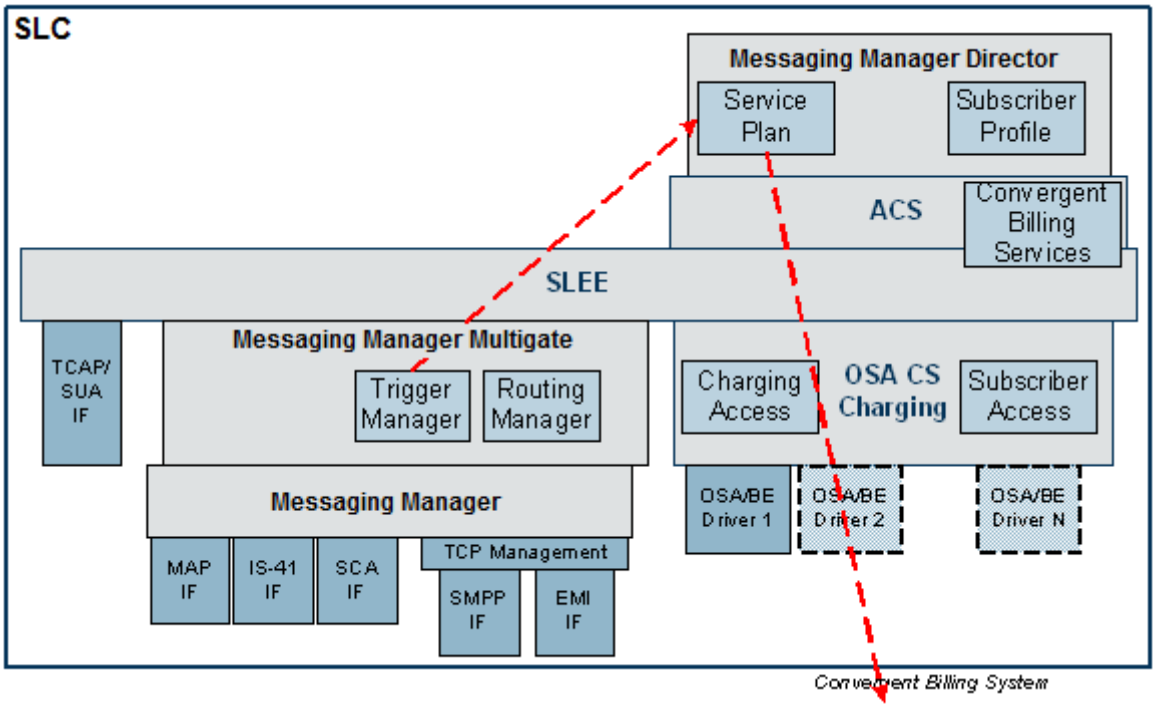
A message can be triggered from Multigate to a specific message control plan to provide extended (customer specific) service logic. For example, Messaging Manager Director may modify any routing options before signalling to Multigate to continue delivery so that Multigate routes the message according to the new options.

Messaging Manager Director can ensure that delivery proceeds only if charging is satisfied, such that delivery and charging proceeds as a single transaction.

Note: To use the features provided by the Messaging Manager Director module, the Convergent Charging Controller ACS and SMS applications must be installed.

Diagram

Here is a diagram showing the Messaging Manager Base, Messaging Manager Multigate and Messaging Manager Director modules.



Routing options

With the addition of the Messaging Manager Director module, MM allows individual routing based on the following criteria:

- Incoming protocol
- MO or MT
- Destination number
- Originating number
- Message content
- Time of day

Message Processing

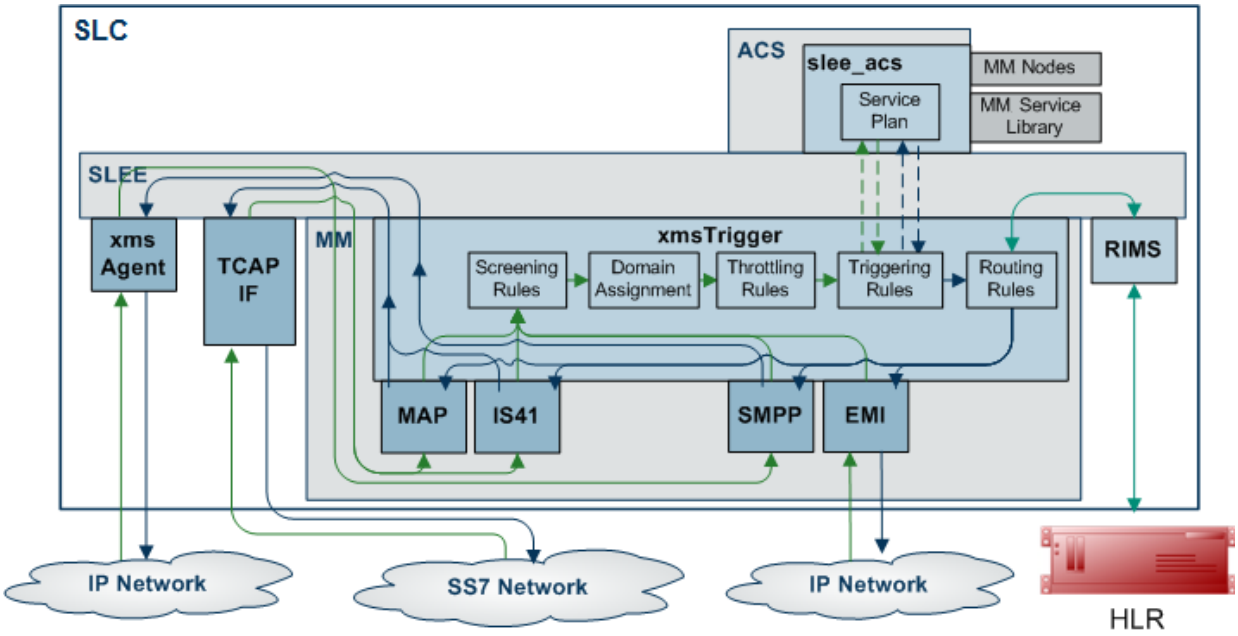
Introduction

Messaging Manager processing falls into three logical parts. Understanding the different parts is important to understand how to configure an MM service. The three parts are:

1 *Incoming classification (addressing)* (on page 10)
2 *Message processing* (on page 11)
3 *Outbound routing* (on page 11)

Message flow

This diagram shows the flow of the message through Messaging Manager:



Incoming classification (addressing)

This table describes how Messaging Manager processes inbound messages.

Stage	Description
1	Messages are received over a protocol-specific adapter. The configuration of which adapter will be used is done in **eserv.config**. For more information about **eserv.config**, see *Configuring the Required Adapters* (on page 71). For signaling protocols, the PC, SSN, GT and potentially the setting of the GPRS support parameters, are used to direct the inbound message to the correct adapter.
2	The adapter establishes the inbound connection and path for the message using the configuration in the currently deployed routing scheme. **Notes:** • The adapter matches against the connections in the paths which have been configured to be available to it. • The best match is used. • IP protocols don't have a default path. For more information about paths and connections, see Paths and Connections in *MM User's Guide*.
3	A default routing class is assigned to the message, based on its transaction type. Each transaction is classified as one of Submit, Deliver, Notify, Route Info or Command. **Exception:** If the message has a command routing class, it will be forwarded directly to the configured default path for that protocol. For more information, see Default routing in *MM User's Guide*. For more information about routing classes, see Routing Class in *MM User's Guide*.
4	Each message is assigned to a default SMSC. Operations performed by Messaging Manager will take place in a fashion consistent with the assigned SMSC name.

Stage	Description
	For more information, see SMSCs in *MM User's Guide*.
5	Screening options are applied, which potentially filter out undesired messages. For more information about screening configuration, see Screening Rules in *MM User's Guide*.
6	The originating address and destination addresses are matched against address rules to determine the originating domain name and the destination domain names. For more information about addressing rules, see Address Domains in *MM User's Guide*.

Message processing

This table describes how Messaging Manager processes messages.

Stage	Description
1	Based on the criteria assigned by the classification rules, the message is checked by congestion control. This may result in transactions being throttled. For more information about throttling, see Congestion Control in *MM User's Guide*.
2	Based on the transaction type, messages are then directed to one of four sets of trigger rules, for Submit, Deliver, Notify or Route Info transactions. This may result in triggering to ACS to run a message control plan in order to control delivery processing options. Control plans can change message parameters. Having selected a best match trigger rule it is possible to modify the transaction's routing class from its default value (assigned during incoming classification). A matching trigger rule may be one of the following: • Perform an action • Trigger to ACS to run a control plan For more information about triggering, see Triggering in *MM User's Guide*.
3	If the message was triggered to a control plan, and the control plan returned a release INAP (that is, the control plan exited after a Disconnect node, or an error exit), the ACS release cause is mapped to an action or error code. The action or error code is added to a Nack which is returned to the source of the message. For more information about action and error codes, see Messaging Manager Action and Error Codes in *MM User's Guide*.

Outbound routing

This table describes how Messaging Manager processes outbound message routing.

Stage	Description
1	Outbound routing takes place based on the routing class. When applying: • Submit routing, the key determinant of the outbound path is the message center name and the originating or domain address. • Deliver routing, the key determinant of the outbound path is the destination domain name or prefix, and/ or originating domain name or prefix. • Locate routing, the key determinant of the outbound path is the destination domain name or prefix, and/ or originating domain name or prefix. For more information about routing, see *Routing* (*32806.htm*) in *MM User's Guide*.

Stage	Description
2	One or more outbound paths may be selected by the routing rule. If there is more than one, then each is tried in turn, until "success" occurs or a permanent error is encountered.
3	The adapter for each selected path will build the appropriate PDU, based on the path protocol, and select a connection within the path for transmission.
4	If a message control plan is active, it will be notified of the outcome from outbound routing to complete any service logic, such as finalize charging, retry by switching to an alternate route.

Routing

Messaging Manager routes all messages based on one of the following:

- Routing class
- Prefix and domain

Triggering rules

Messaging Manager allows rules to be determined that will trigger a message to ACS for further processing. These ACS Triggering Rules may be set so that the message is triggered to ACS using a match based on a prefix containing one of the following:

- The originating number, originating domain
- The terminating number of the message or terminating domain

This may result in the message being triggered to ACS twice.

Terminating triggering rules

By default, Messaging Manager will perform routing of deliver requests according to the following rules.

1 If no routing prefix was added by the service logic, the originating adapter ID will be used to determine the outgoing route.

2 If a routing prefix was added by the service logic, this will be used to find the outgoing route. If both the originating and terminating service plans add prefixes, the terminating value will be used.

3 If no route is matched then the deliver will fail.

Interfaces and nodes

Routing nodes can provide connections for one of the following:

- All IP connections of a routing scheme
- Connections only for certain ASPs

This means a connection does not need to support all the capabilities of its associated routing scheme in order to be a valid connection.

Routing schemes can be configured with interface records. An interface record is a virtual IP connection that may be supplied or instantiated by real IP addresses on one or more routing nodes. Each node can assign a different IP address to an interface record.

Routing nodes are configured with a list of real 'IP addresses'. When a routing scheme is assigned to a routing node, you can map any of the routing scheme interfaces to the node's IP addresses. This defines what (if any) contribution that node makes to the scheme's routing interface requirements.

When is a Delivery Report produced?

Delivery Reports are generated and sent as a completely separate transaction (like other SMSs that Messaging Manager handles). That means existing routing and retry functionality can be used to deliver the DR.

Messaging Manager generates a delivery report (DR) in the following conditions:

- An 'early acknowledged' message is subsequently unable to be delivered. Messaging Manager can be configured to generate a delivery report regardless of whether or not the originator requested it (through the `alwaysProduceNonDeliveryReceipt` parameter).
- If Messaging Manager successfully delivers a message by FDA and the originating party requested a delivery receipt, then a delivery report is generated and sent to the originator

Delivery Report failure

Under normal (default) configuration a DR send failure is resent, however setting the parameter `singleShotDeliveryReport` to `true` will prevent a retry to be attempted.

Statistics

Any statistics produced by a Delivery Report will have INTERNAL_DR in the detail field.

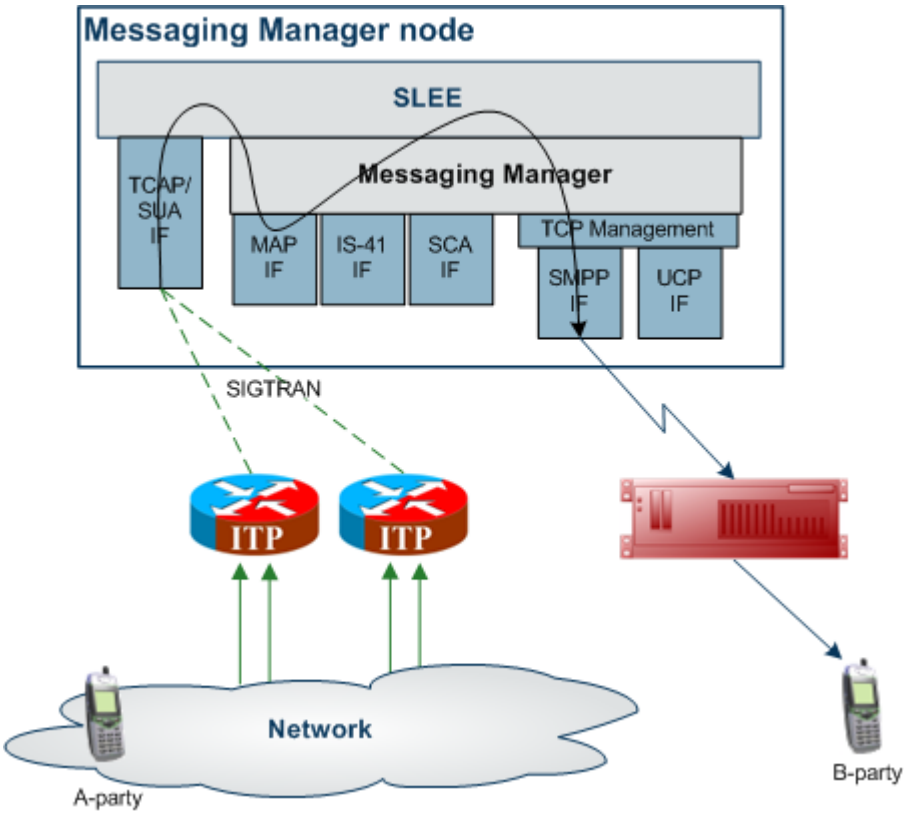
Mobile to Mobile Messaging

Description

Mobile to mobile messaging provides standard person to person short messages.

MO SMS diagram

Using only the Messaging Manager base module, MM can be configured to provide an MO SMS service. In this example we will receive mobile originating MAP messages and deliver to the SMSC over TCP/IP using the SMPP protocol. The following diagram shows the modules required.

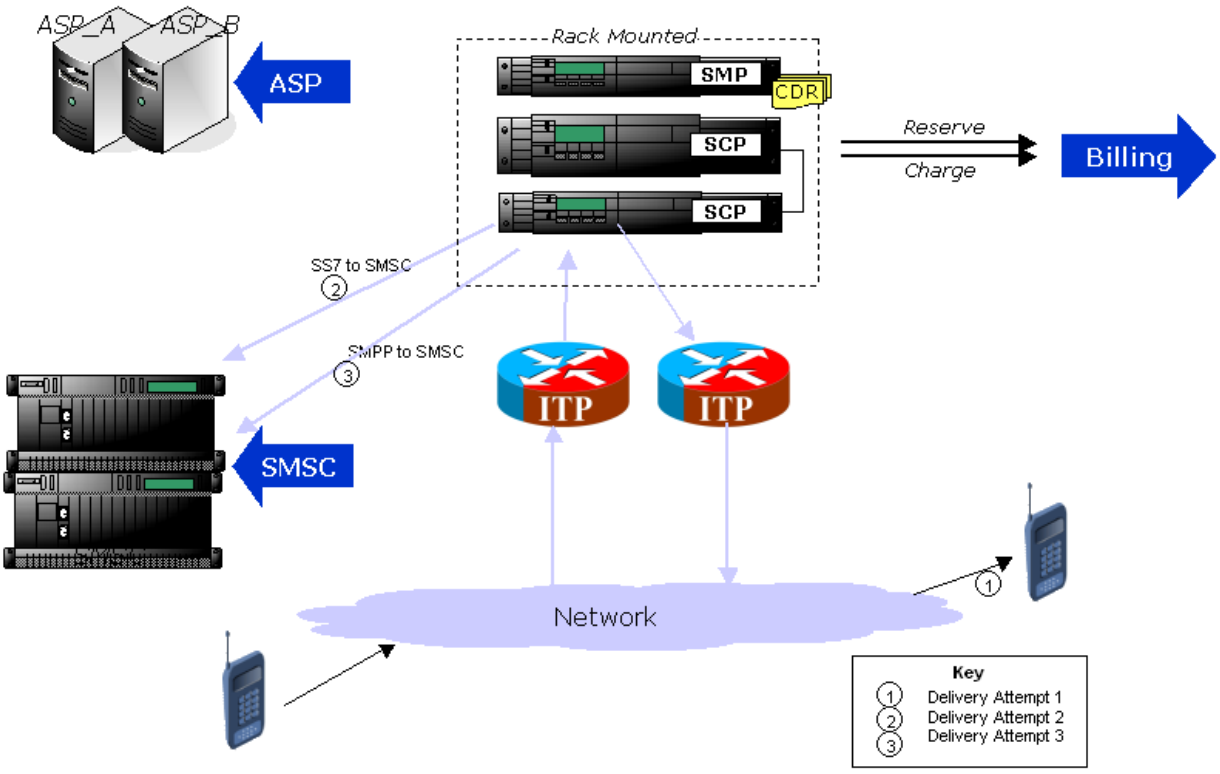


Variations

Standard person to person messaging may be configured in several different ways using MM:

- No FDA (as shown at 1)
- With FDA (as shown at 2)
- With multiple alternate routing

Multiple alternate routing allows several routes to be tried to deliver a message. Routing attempts may be similar to below:

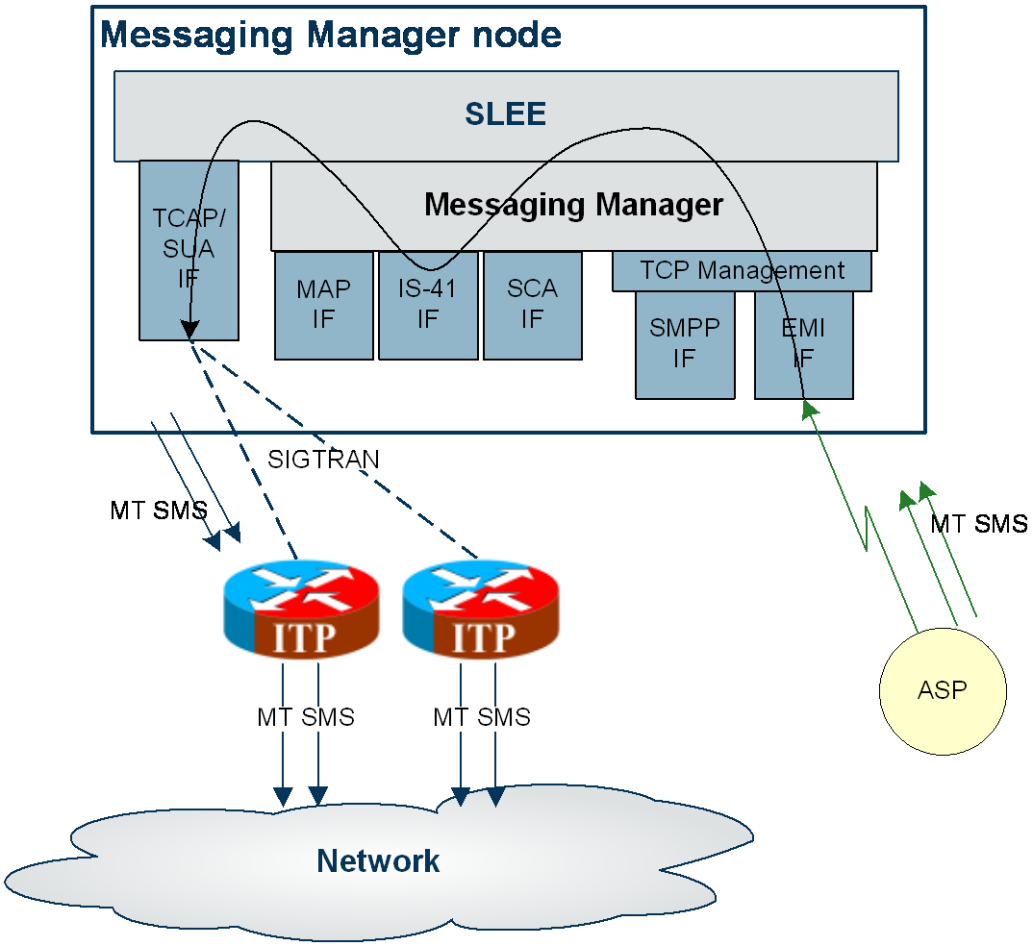


Application to Mobile Messaging

Application to Mobile diagram

Using only the Messaging Manager Base module, MM can be configured to provide an application to mobile service. In this example we will receive EMI protocol messages from ASPs and deliver them to an SMSC over SS7 using the IS-41 protocol.

The following diagram shows the modules required:



Variations

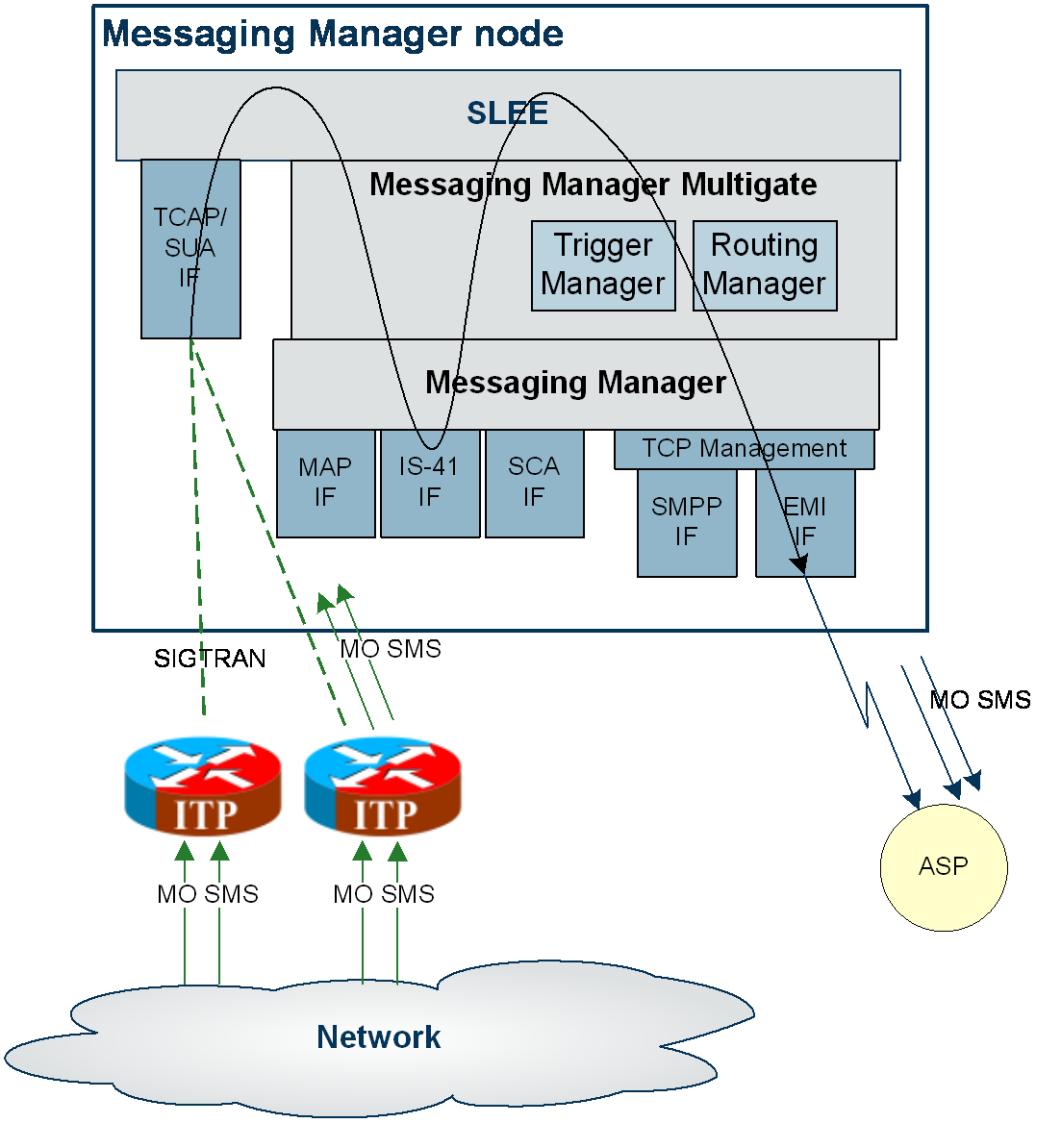
Application to Mobile messaging may be configured in several different ways using MM:

- No FDA
- With FDA only (as shown)
- With multiple alternate routing

Mobile to Application Messaging

Mobile to Application diagram

Using the Messaging Manager Multigate and the Messaging Manager Director modules, MM can be configured to provide a Mobile to Application service. In this example we will receive mobile originating IS-41 messages and deliver them to ASPs over EMI using FDA. This diagram shows the modules required.



Variations

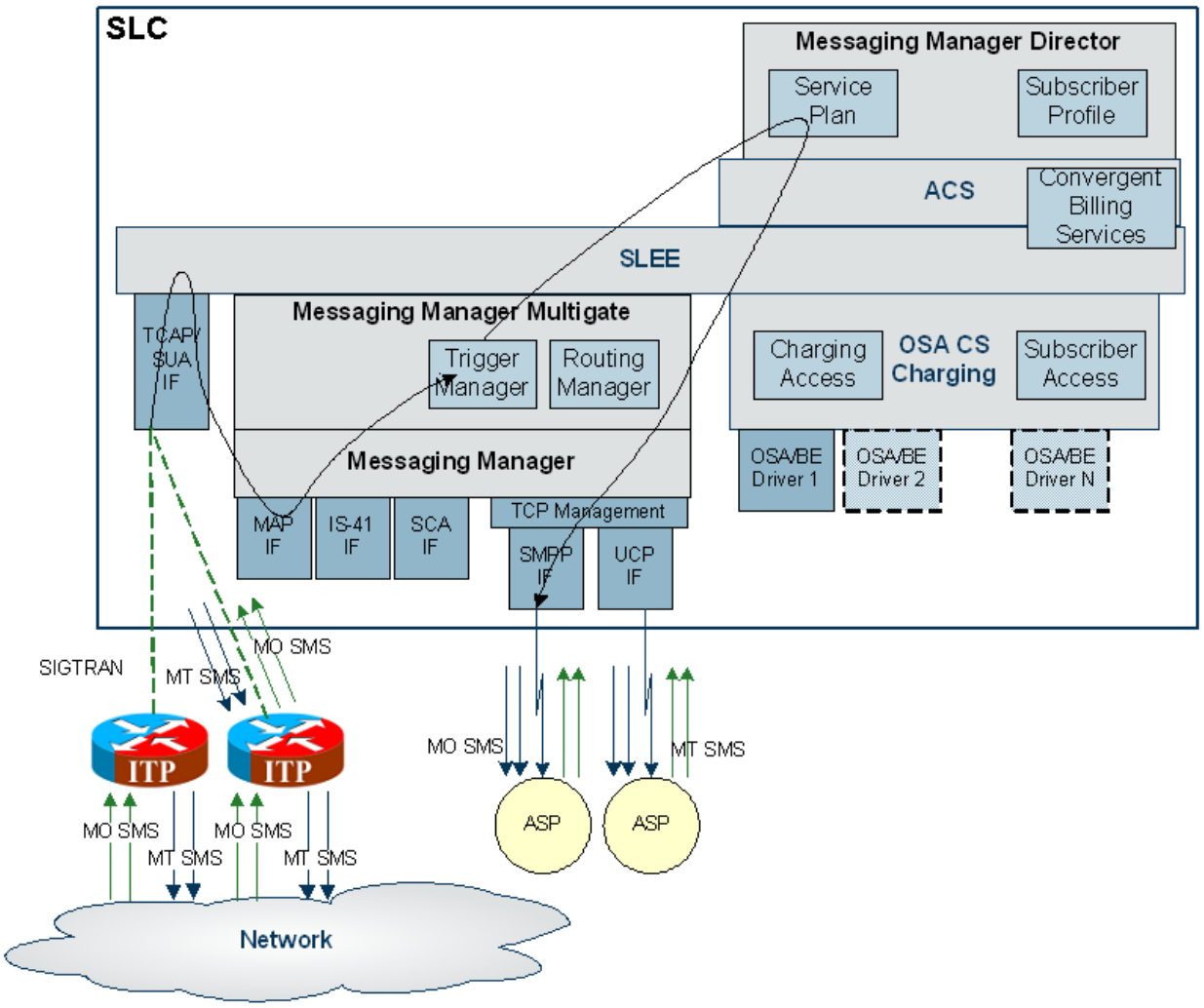
Mobile to Application messaging may be configured in several different ways using MM:

- No FDA
- With FDA only (as shown)
- With multiple alternate routing; that is, with FDA using an SMSC as an alternative should FDA fail

Mobile to Mobile triggering to ACS

Diagram

Using the Messaging Manager Multigate and the Messaging Manager Director modules, MM can be configured to provide a Mobile to Mobile service, triggering to ACS. In this example we will receive mobile originating MAP messages and deliver them to SMSCs over MAP having triggered them to ACS to offload all large messages to a separate SMSC. The following diagram shows the modules required:



Variations

Using the Messaging Manager Director module, messages may be triggered to ACS for advanced message processing in any or both of the following ways:

1. Originating number - for example, SMS MO prepaid billing
2. Terminating number - for example, ASP specific processing such as context based routing

Instant Messaging

Scenario assumptions

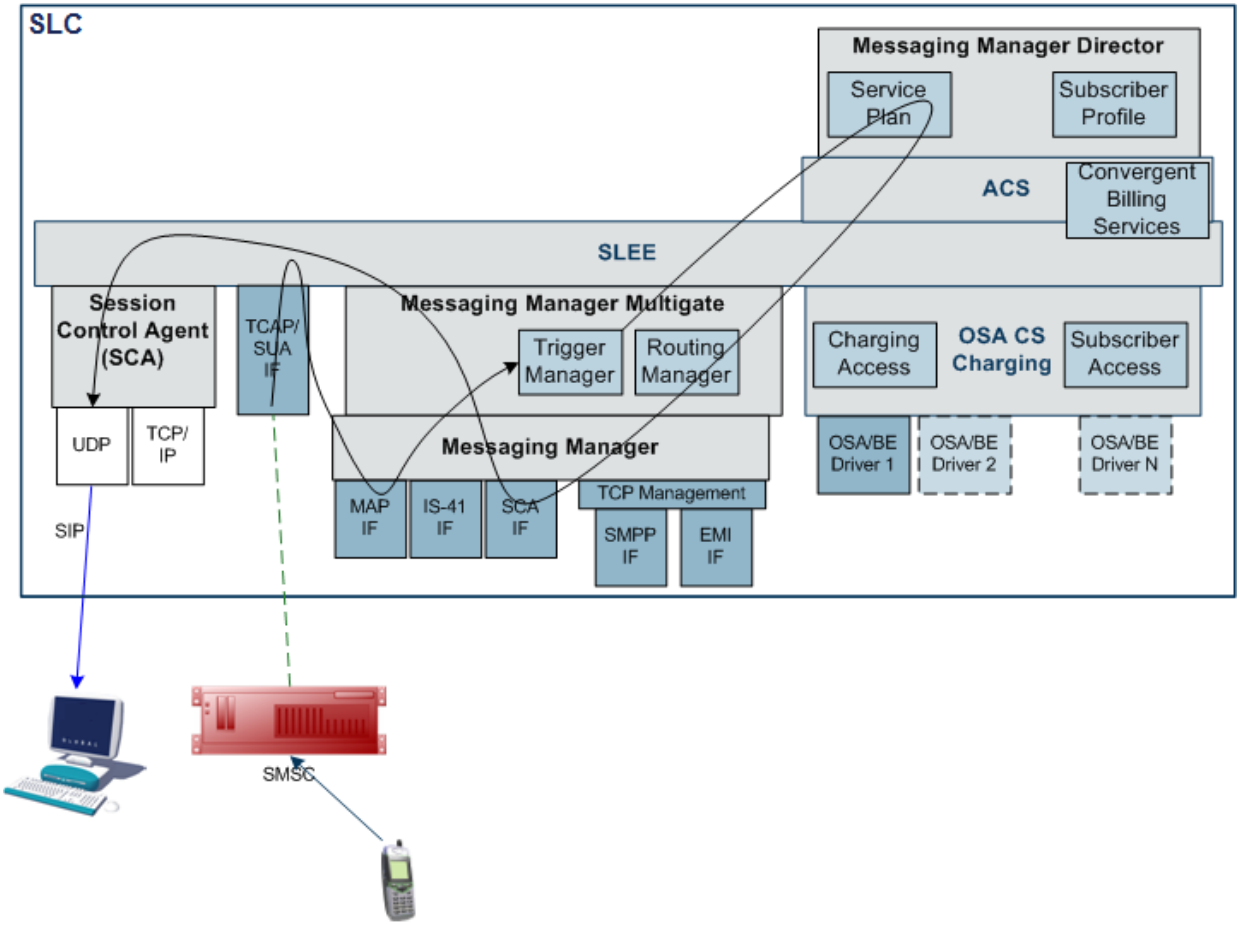
This section describes scenarios that will be supported by the SCA adapter. Three scenarios are outlined and described below. In these scenarios it is assumed that two subscribers have access to SIP instant messaging user agents and also GSM handsets for sending short messages. It is assumed that have registered their SIP URIs with a registrar and that SCA is installed on the machine. (Note that SIP routing is handled by the SCA and not covered here). In addition, it assumes there exists a special domain (Oracle.com) that can forward an instant message to a handset using SMS.

Subscriber	MSISDN	SIP
Tom	641233570	Tom@imdomain.com
Dick	641233402	Dick@imdomain.com

SMS forwarded to SIP

In this scenario, Tom sends a short message to Dick, who has enabled instant message forwarding using some mechanism not relevant to this design. This will forward a copy of the short messages to Dick's SIP user agent.

This is accomplished by executing a control plan that contains the Send Short Message Notification node (SSMN), which allows sending instant messages. The node specifies the destination address in URI format (for example, Dick@imdomain.com). The content and other information about the message is contained in a GenericSM event. This event is sent through the SLEE to the SCA Adapter. The SCA Adapter converts the GenericSM event to a SipSleeEvent and forwards this event to the SCA.



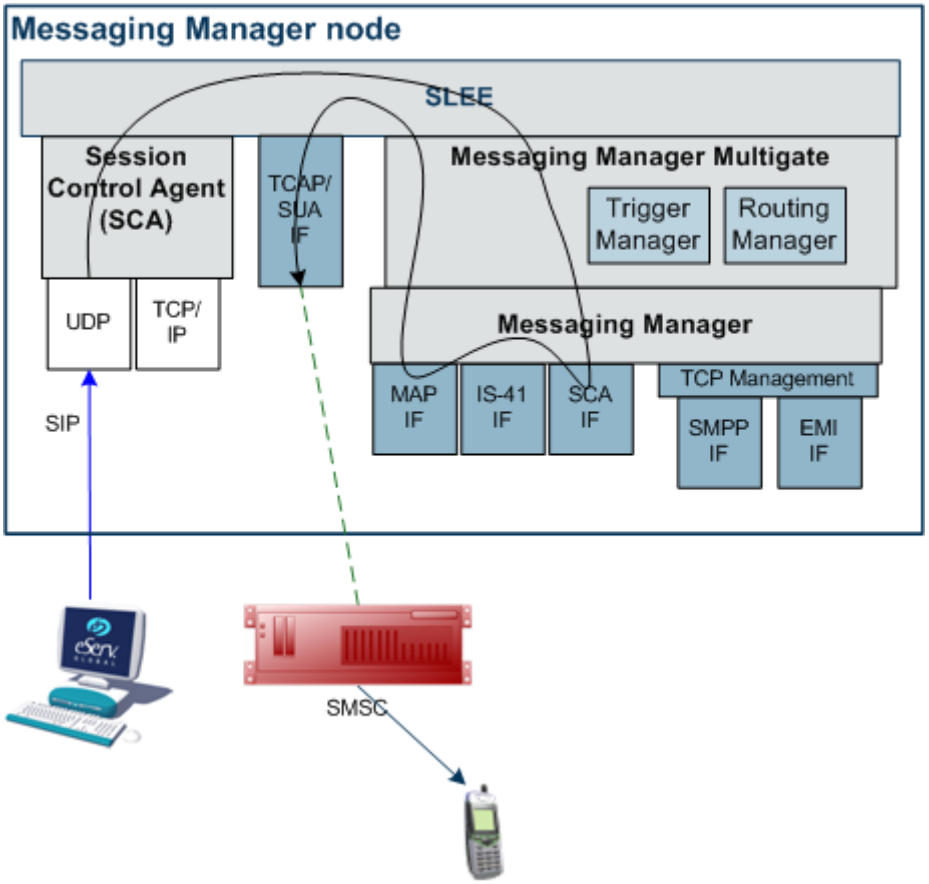
SMS to SIP

In this scenario, Tom sends a short message to a special short code (64121) that will forward the short message to Dick's instant message user agent. This scenario is similar to the scenario described above, and the path is the same as shown in that diagram, except that the destination address is specified as part of the message content (for example, "Dick@imdomain.com Watson, come here." The SSMN (or other) node extracts the destination address from the content. The short message is swallowed by MM.

SIP to SMS

In this scenario Tom sends an instant message to Dick, addressed to a handset (the E.164 telephone number - for example, 64123402). The SCA converts this message to a SipSleeEvent and passes this to the SCA Adapter. The SCA Adapter creates a GenericSM event and MM routes this to the MAP adapter. The MAP adapter sends a MAP MO-ForwardSM to the SMSC.

In this scenario, Tom uses his E.164 alias (for example, 64123570) as the originating address. This allows a reply through SMS.



Chapter 2

Configuration

Overview

Introduction

This chapter explains how to configure the Oracle Communications Convergent Charging Controller application.

In this chapter

This chapter contains the following topics.

MM and XMS naming conventions

Reading this manual and configuring Messaging Manager, you may notice that the terms MM and XMS are both used. These terms have been used interchangeably throughout the software development process as a result of software evolution. Many of the software processes and binaries use the term XMS, and in this case the documentation will continue the use of the term. The abbreviation for the Messaging Manager product is MM, and this will be used in all cases except where referring to specific software binaries and processes.

Configuration Overview

Introduction

This topic provides a high level overview of how the Messaging Manager application is configured. Configuration details for individual processes are located with the documentation for that process.

Configuration process overview

This table describes the steps involved in configuring Messaging Manager for the first time.

Stage	Description	Refer to..
1	Create the XMS section of the **eserv.config** file.	Messaging Manager *Configuration Sections in* **eserv.config** (on page 40).
2	Configure the xmsTrigger section.	*xmsTrigger* (on page 167, on page 42)
3	Configure the tracing SMSs, if required. This section is required, but may be empty.	*Tracing SMSs* (on page 58)
4	Configure the CDR collection section.	*Configuring EDR Collection* (on page 64)
5	Configure statistics collection.	*Collecting Statistics* (on page 70)
6	Set the early acknowledgement parameters. These can be done at Trigger and adapter level.	*Setting Early Acknowledgment* (on page 66).
7	Configure the persistent storage.	*Setting Pstore* (on page 68).
8	Define the screen language for the GUI interface.	*Defining the Screen Language* (on page 70).
9	Configure Messaging Manager Multigate.	*Configuring Messaging Manager Multigate* (on page 71).
10	Configure Messaging Manager Director.	*Configuring Messaging Manager Director* (on page 137).

Configuring the Environment

Configuration components

MMX is configured by the following components:

Component	Locations	Description	Further Information
eserv.config	All	**eserv.config** holds most of the configuration for MM.	**eserv.config** *Configuration* (on page 27)
acs.conf	All SLCs	Configures slee_acs, the main call processor. **acs.conf** must include the MM plug-in libraries for slee_acs, and is also used for normalization and denormalization configuration.	Configuring acs.conf for the SLC
SLEE.cfg	All SLCs	Configures which MM processes are started by the SLEE, and provides some configuration for the environment in which those processes run.	*SLEE.cfg* (on page 25)
tdp.conf	All SLCs	The tdp.conf file is the configuration file that is used by the system to define the trigger tables used to determine when to trigger a call to the SCF.	*tpd.conf* (on page 26)

Component	Locations	Description	Further Information
SMS User Interface	SMS	Provide a graphical interface for configuring many parts of SMS including: • Replication • Statistics • Alarm filtering • Reports	*SMS User's Guide*

Note: Most of these files are at least partially configured when the xmsScp package is installed.

acs.conf

The acs.conf file is the configuration for the ACS application. For more information on the acs.conf file, see *ACS Technical Guide*.

The xmsScp package will have added the following lines to the acsChassis section of the acs.conf file. No manual modification is required.

```
ServiceEntry (SMS_Submit,xmsSvcLibrary.so)
ServiceEntry (SMS_Deliver,xmsSvcLibrary.so)
ChassisPlugin libxmsChassisActions.so
MacroNodePluginFile libxmsMacroNodes.so
```

The following lines will be present if the PME service is installed.

```
ServiceEntry (CCS_SM_MO,dD,cC,dD,E,ccsSvcLibrary.so)
ServiceEntry (CCS_SM_MT,cC,dD,ccsSvcLibrary.so)
```

Note: The service names (for example: "SMS_Submit") match the services defined in the SLEE.cfg file (for details, see *SLEE.cfg* (on page 25)).

Location

The **acs.conf** file is located in **/IN/service_packages/ACS/etc/**.

acs.conf settings

Set the following in **acs.conf**:

```
UseContinueOperation 0
```

For more information about this parameter, see *ACS Technical Guide*.

eserv.config

The **eserv.config** file is a shared configuration file, from which many Convergent Charging Controller applications read their configuration. For details, see the **eserv.config** *Configuration* (on page 27) topic.

The **eserv.config** file needs to contain a *SECURE section* (on page 54) within the xmsTrigger section. This includes the list of licenses, adapters and gateways for this installation of Messaging Manager. This section is provided by Convergent Charging Controller.

SLEE.cfg

The SLEE.cfg file is the configuration file for the SLEE, see the *SLEE Technical Guide* for further details about this file.

On installation of xmsScp, some Messaging Manager-specific lines are added to the file.

Warning: Always make a backup copy of SLEE.cfg before making any modifications.

Editing the SLEE.cfg

The following lines must be present in **SLEE.cfg**:

```
SERVICEKEY=INTEGER 102 xmsAgent # xmsAgent inbound
SERVICEKEY=INTEGER 103 xmsAgent # xmsAgent outbound
INTERFACE= xmsAgent xmsAgent.sh /IN/service_packages/XMS/bin EVENT

SERVICEKEY=INTEGER 120 SMS_Submit # Added by xmsScp
SERVICEKEY=INTEGER 121 SMS_Deliver # Added by xmsScp

SERVICE=SMS_Submit 1 slee_acs SMS_Submit # Added by xmsScp
SERVICE=SMS_Deliver 1 slee_acs SMS_Deliver # Added by xmsScp
```

Note: The service names must be exactly `SMS_Submit` and `SMS_Deliver` for the Messaging Manager Director service to function correctly.

The service keys 102 and 103 are provided by a question during installation. These are the default values.

The service keys 120 and 121 must match the service keys defined in the **tdp.conf** file. For details on this see *Editing tdp.conf* (on page 26).

SLEE.cfg details for PME

The following lines are added if the PME service is installed.

```
SERVICEKEY=INTEGER 122 CCS_SM_MO # Added by xmsScp
SERVICEKEY=INTEGER 123 CCS_SM_MT # Added by xmsScp

SERVICE=CCS_SM_MO 1 slee_acs CCS_SM_MO # Added by xmsScp
SERVICE=CCS_SM_MT 1 slee_acs CCS_SM_MT # Added by xmsScp
```

tdp.conf

The tdp.conf file is the configuration file that is used by the system to define the trigger tables used to determine when to trigger a call to the SCF.

If it has been installed, the acsScp package will have created tdp.conf. If acsScp has not been installed it is necessary to take a copy of the tdp.conf.example file that is installed by Messaging Manager and rename it to tdp.conf

No manual modification of this file is required, unless SLEE service keys different to the recommended 120 and 121 are used in the SLEE.cfg (see *SLEE.cfg* (on page 25)). Always make a backup copy of tdp.conf before making any modifications. Do not modify the file in any other way except as directly instructed by an Oracle technical engineer.

For further information on the IN Call Model and tdp.conf please refer to Configuring IN Call Model Triggers.

Location

The **tdp.conf** file is located in **/IN/service_packages/XMS/etc/**.

Example tdp.conf

This is an example of a **tdp.conf** file. The file must contain the following contents:

```
# This file is supplied with the XMS product installation and
# should not be changed except as directly instructed by an
# Oracle Technical Engineer
-1 121 3 R all all
1 120 3 R all all
```

The following lines will also be present if the PME service is installed.

```
# Trigger all Submit messages to the CCS_SM_MO service
1 122 3 R all all
# Trigger all Deliver/Notify/RouteInfo messages to the CCS_SM_MT service
-1 123 3 R all all
```

Note: All lines starting with # are treated as comments.

The first line defines the terminating triggering rule. The second number on this line (121) must match the service key of the "SMS_Deliver" SLEE service.

The second line defines the originating triggering rule. The second number on this line (120) must match the service key of the "SMS_Submit" SLEE service.

eserv.config Configuration

Introduction

The **eserv.config** file is a shared configuration file, from which many Oracle Communications Convergent Charging Controller applications read their configuration. Each Convergent Charging Controller machine (SMS, SLC, and VWS) has its own version of this configuration file, containing configuration relevant to that machine. The **eserv.config** file contains different sections; each application reads the sections of the file that contains data relevant to it.

The **eserv.config** file is located in the **/IN/service_packages/** directory.

The **eserv.config** file format uses hierarchical groupings, and most applications make use of this to divide the options into logical groupings.

Configuration File Format

To organize the configuration data within the **eserv.config** file, some sections are nested within other sections. Configuration details are opened and closed using either { } or [ ].

- Groups of parameters are enclosed with curly brackets – { }
- An array of parameters is enclosed in square brackets – [ ]
- Comments are prefaced with a # at the beginning of the line

To list things within a group or an array, elements must be separated by at least one comma or at least one line break. Any of the following formats can be used, as in this example:

```
    { name="route6", id = 3, prefixes = [ "00000148", "0000473"] }
    { name="route7", id = 4, prefixes = [ "000001049" ] }
```

or

```
    { name="route6"
        id = 3
        prefixes = [
            "00000148"
            "0000473"
        ]
    }
    { name="route7"
        id = 4
        prefixes = [
            "000001049"
        ]
    }
```

or

```
    { name="route6"
```

```
        id = 3
        prefixes = [ "00000148", "0000473" ]
    }
    { name="route7", id = 4
        prefixes = [ "000001049" ]
    }
```

eserv.config Files Delivered

Most applications come with an example **eserv.config** configuration in a file called **eserv.config.example** in the root of the application directory, for example, **/IN/service_packages/eserv.config.example**.

Editing the File

Open the configuration file on your system using a standard text editor. Do not use text editors, such as Microsoft Word, that attach control characters. These can be, for example, Microsoft DOS or Windows line termination characters (for example, ^M), which are not visible to the user, at the end of each row. This causes file errors when the application tries to read the configuration file.

Always keep a backup of your file before making any changes to it. This ensures you have a working copy to which you can return.

Loading eserv.config Changes

If you change the configuration file, you must restart the appropriate parts of the service to enable the new options to take effect.

Rereading the eserv.config file

xmsTrigger re-reads the configuration from **eserv.config** on startup. It can also be told to re-read its configuration while running using the **xmsReadConfig.sh** script:

/IN/service_packages/XMS/bin/xmsRereadConfig.sh *usr*

Run the xmsRereadConfig script as the user who starts the xmsTrigger application from your console or telnet session on the SLC. In most cases the user who starts the xmsTrigger application is acs_oper.

xmsReadConfig.sh supports the following parameter.

usr

Syntax:	*usr*
Description:	The unix user which is running the instance of xmsTrigger you want to update the configuration of.
Type:	String
Optionality:	Optional (default used if not set).
Allowed:	Any valid unix userid.
Default:	root
Notes:	xmsTrigger is usually started by acs_oper, so acs_oper should usually be specified.
Example:	acs_oper

Example eserv.config

Here is the example configuration, with comments removed.

```
XMS = {

serviceLibrary = {
```

```
    validityTime = 5
    flushTime = 60
    maxAge = 3600

    DialledNumberAvailable = false

    xmsUndoNumTranslation = false
}

xmsAgent = {
    tcpWaitTimeMilliSec = 10
    rateLimitAlarmIntervalSec = 60
    additionalConnectTimeMilliSec = 0
}


xmsTrigger = {

    SECURE = {
        limits = {
            maxConcurrentTransactions = 0
            warnConcurrentTransactionsPercentage = 70
            warnConcurrentTransactionsPeriod = 5
            clearConcurrentTransactionsPeriod = 5
            maxSmsPerSecond = 200
            warnSmsPerSecond = 180
            outgoingQueueSize = 2
            outgoingQueueTimeout = 10
        }
        options = ["MCC","SAS","EDR","MCP"]
        gateways = ["MO_SMS","MT_SMS","VAS_SMS"]
        adapters = ["MAP","CDMA","TDMA","EMI","SMPP","API"]

    }

    SECUREKey = 123456

    xmsAgentInboundServiceKey = 102
    xmsAgentOutboundServiceKey = 103

    overridePluginName = true

    loadReportingPeriod = 10
    loadReportChangesOnly = false

    routingScheme = {
        loadIntervalSeconds = 600
    }

    mtTransactionLifetimeSeconds = 30

    oracleusername = ""

    oraclepassword = ""

    oracledatabase = ""

    dialledStarEncoding = 'D'
    dialledHashEncoding = 'E'

    mmxIsdn = "0640041234567"
    mmxIsdnGprs = "0640041234568"
```

```
deliveryReceiptId = "scp1:"
desegmentation = false
desegmentation_timeout = 0
desegmentation_failure_code = 1
desegmentation_failure_cause = 27

baseIDPSize = 200

allowConcatenatedFDA = true
earlyAckMC = true
earlyAckSME = true

alwaysProduceNonDeliveryReceipt = false

stopRoutingOnTransientFailure = false
stopRoutingOnPermanentFailure = false

processMsgSCI = true

successDeliveryReceiptText = "Your message to <destination> was delivered"
failureDeliveryReceiptText = "Your message to <destination> was NOT delivered"

cdr = {

    log = true
    filename = "xms_"
    tempdir = "/IN/service_packages/XMS/cdr/current/"
    maxno = 10000
    time_out = 1800
    destdir = "/IN/service_packages/XMS/cdr/closed/"
    num_files = 64000
    num_files_leaf = 64
}

tracing = {

    enabled = true
    showPrivate = true
    outputFile = "/tmp/xmsTrigger.trc"
    outputFileCycle = 512

    maxFileSizeKB = 0
    maxNumFiles = 4

    shmKey = 92403
    shmSizeKb = 64

    callsPerMinute = 2

    origAddress = [
        "0064212",
        "0064213",
        "0064214"
    ]

    destAddress = [
        "0064213",
        "0064214"
    ]

    useTONNPI = true

}
```

```
smsAuditing = {
    # Is auditing enabled? (default false)
        enabled = false

    # Output file.
    #-------------------------------------------------------#
    #                                                       #
    # IMPORTANT NOTE:                                       #
    # if the output file name is changed,                   #
    # make sure the changes are reflected in                #
    # /IN/service_packages/ACS/etc/logjob.conf              #
    # Otherwise the audit file will not be archived         #
    #                                                       #
    #-------------------------------------------------------#

       outputFile = "/IN/service_packages/XMS/tmp/xmsTrigger.aud"

    # Close and re-open the file every N calls.  Check for file size exceeded
    # at this time also.

        outputFileCycle = 32

    # should log originating address ? ( default: true)
        logOriginatingAddress = true

    # should log destination address ? ( default: true)
        logDestinationAddress = true

} #smsAuditing


statistics = {
    enable = false
}

adapters = [

    # First adapter (MAP)
    {
        lib = "xmsiMap.so"
        adapterName = "MAP1"

        pointCodes = [1001, 1002]
        SSN = 8

        earlyAckMC = true
        earlyAckSME = true

        allowConcatenatedFDA = true
        alwaysProduceNonDeliveryReceipt = false


        config = {

            disableConcatenatedSegmentPad = false

            abortMessagesWithZeroLengthTPDA = true

            allowIncoming = true
            allowOutgoing = true
            allowDirectDelivery = true
```

```
allowIncomingMap3 = true

allowRetryWhereRimsAndHLRLookupEqual = false

allowUserRequestedDeliveryReceipt = true

suppressRimsMap3imsiOAUpdate = false

lastSegmentDeliveryReceiptOnly = false

tcapInterfaceServiceKey = 22

originatingTimeout = 10

smscTimeout = 8
hlrTimeout = 3
mscTimeout = 15

rimsInterfaceName = "rimsIf"

gprsSupport = "supported"

nonGprsAdapter = "MAP2"

sgsnPrefixes = [ "000" ]

doProtocolIdMapping = false
defaultProtocolId = 0
protocolIdMap = [
    { in = 127, out = 0 }
]

fallbackAlphabet = "UCS-2"

scheduledDeliveryTime = ""

defaultMessagePriority = "Normal"

PC = 55
SSN = 8

GT = "5114406267
TT = 0

relayTranslationType = -2
relayGlobalTitleType = 2
relaySSN = -1
relayNatureOfAddress = 4

GTMap = [
    { prefix = "0010019198", value = "919827002402" }
]

#SCA = "5114406267"

natureOfAddress = 1 # international
numberPlan = 6      # E.212

defaultMapVersionSmsc = 3
defaultMapVersionMsc = 3


deliveryFailureStatusCode = 64
```

```
        throttledDeliveryFailureCause = 96 # congestion

        mscVersionCacheSize = 1000

        hybridiseMapVersions = false

        maxUnsegmentedLength = 120

        splitLongMessages = true

        alarmMask = 0

        honourReplyPath = false

        pathRetryRandomisation = 1
        pathRetrySegmentOffset = 1

        localTimeZone = "UTC"

        hlrErrorMap = [
            { error = 1, permanent = true }
            { error = 32, permanent = false }
        ]

        defaultTransientFailureErrorCode = 32
        defaultPermanentFailureErrorCode = 32

        incomingOriginatingNumberRules = [
            { fromNoa=2, prefix="04", min=4, max=32, remove=1,
            prepend="0064", resultNoa=1 }
            { fromNoa=3, prefix="4", min=4, max=32, remove=0, prepend="0064",
            resultNoa=1 }
        ]
        incomingDestinationNumberRules = [
        ]
        # outgoingOriginatingNumberRules = [ ]
        # outgoingDestinationNumberRules = [ ]

        # privateExtensions = [
        #        {extId="1,2,776,8,8,8,8,11", asn1Tags= [ 0xE0, 0x80] ,
        profileTagID=327814, name = "imei" }
        # ]

    }
} # xmsiMap.so config

# Non-GPRS MAP adapter
{
    lib = "xmsiMap.so"

    adapterName = "MAP2"

    pointCodes = [1003]

    SSN = 8

    config = {

        GT = "5114406268"
        TT = 0

        gprsSupport = "unsupported"
```

```
    }

} # xmsiMap.so config

# second adapter (EMI)
{
    lib = "mmxiEMI.so"

    SSN = 0

    adapterName = "EMI1"

    earlyAckMC = false
    earlyAckSME = true
    allowConcatenatedFDA = true

    config = {

        suppressPathInfoReport = true
        displayZeroPathReport = false
        PathReportingInterval = 60
        throttledErrorCode = 4
        transientFailureErrorCode = 4
        permanentFailureErrorCode = 3

        incomingOriginatingNumberRules = [
            { fromNoa=999, prefix="", min=1, max=32, remove=0, resultNoa=2 }
        ]
        incomingDestinationNumberRules = [
        ]
        # outgoingOriginatingNumberRules = [ ]
        # outgoingDestinationNumberRules = [ ]

        emiDefaults = {
            defaultMessagePriority = "Normal"

            pstoreNumberRules = [
                {fromNoa=999, prefix="010", min=1, max=32, remove=3,
                prepend="7" }
                { fromNoa=999, prefix="091", min=1, max=32, remove=3,
                prepend="7" }
            ]

            timestampAdvance = true
            timestampBucketSize = 5000
            timestampFlush = 2
        } # emiDefaults
    } # mmxiEMI.so config
}

# Third adapter (SMPP)
{
    lib = "mmxiSMPP.so"

    #pointCodes = [1003, 1004]

    SSN = 0

    adapterName = "SMPP1"

    earlyAckMC = false
    earlyAckSME = true
    allowConcatenatedFDA = true
```

```
config = {

    suppressPathInfoReport = true
    displayZeroPathReport = false
    PathReportingInterval = 60
    fallbackAlphabet = "UCS-2"
    TLVs = [
        {tlvID=0x0030, tlvType=0x01, profileTagID=4532781,
        direction="inbound"}
        {tlvID=0x020b, tlvType=0x02, profileTagID=4532782,
        direction="outbound"}
        {tlvID=0x1382, tlvType=0x05, profileTagID=4532783,
        direction="both"}
    ]
    smppDefaults = {
        throttledCommandStatus = 88 # ESME_RTHROTTLED (88)

        teleserviceRoutingMap = [
            { serviceType =    "", teleservice =    0,
            allowAlternateDelivery = true }
            { serviceType = "CMT", teleservice = 4098,
            allowAlternateDelivery = true }
            { serviceType = "EMS", teleservice = 4101,
            allowAlternateDelivery = true }
        ]

        dataCodingMap = [
            { data_coding = 0x08,
                alphabet = "UCS-2",
                messageClass = 1,
                messageWaitingGroup = 0,
                messageWaitingIndicator = 0,
                messageWaitingType = 0 }
            { data_coding = 0xF0,
                alphabet = "GSM8Bit",
                byteAlign = true,
                messageClass = 1,
                messageWaitingGroup = 0,
                messageWaitingIndicator = 0,
                messageWaitingType = 0 }
        ]

        includePayloadDRInfo = false]

        maxValidityPeriod = 0

        scheduledDeliveryTime = ""

        convertMessageIdToHex = false

        fixedLengthMessageId = true

    } # smppDefaults

    outgoingOriginatingNumberRules = [
        { fromNoa=2, prefix="04", min=4, max=32, remove=1,
        prepend="0064", resultNoa=1 }
        { fromNoa=3, prefix="4", min=4, max=32, remove=0, prepend="0064",
        resultNoa=1 }
    ]
    outgoingDestinationNumberRules = [
    ]
    # incomingOriginatingNumberRules = [ ]
```

```
        # incomingDestinationNumberRules = [ ]

    } # mmxiSMPP.so config
}

# IS-41 CDMA adapter
{
    lib = "xmsiIS41.so"

    adapterName = "CDMA1"
    #pointCodes = [1005, 1006]
    SSN = 18

    earlyAckMC = false
    earlyAckSME = false

    allowConcatenatedFDA = true

    config = {

        allowIncoming = true
        allowOutgoing = true

        PC = 1005
        SSN = 6
        GT = ""
        TT = 0

        originatingTimeout = 20
        smsreqTimeout = 10
        smdppTimeout = 10
        smdppTimeoutSME = 10

        fallbackAlphabet = "UCS-2"

        supportIS841 = true

        defaultMessagePriority = "Normal"

        defaultEndpointType = "SME"

        rimsInterfaceName = "rimsIf"

        pathRetryRandomisation = 1
        pathRetrySegmentOffset = 1

        alarmMask = 0

        relaySmsNotifications =false

        minHLRTransType = 3
        mdnHLRTransType = 14

        defaultTransientFailureCauseCode = 33 # destination busy
        defaultPermanentFailureCauseCode = 39 # other terminal problem

        throttledFailureCauseCode = 35 # destination resource shortage

        deliveryFailureErrorClass = 2
        deliveryFailureStatusCode = 5

        protocol = "CDMA"

        allowFDAforWEMT = true
```

```
        allowAlternateDeliveryForWEMT = true

        tcapInterfaceServiceKey = 22

        incomingOriginatingNumberRules = [
            { fromNoa=3, prefix="04", min=4, max=32, remove=1,
            prepend="0064", resultNoa=4 }
            { fromNoa=2, prefix="4", min=4, max=32, remove=0, prepend="0064",
            resultNoa=4 }
        ]
        incomingDestinationNumberRules = [
        ]
        # outgoingOriginatingNumberRules = [ ]
        # outgoingDestinationNumberRules = [ ]

    } # End IS-41 CDMA adapter config section.
}

# IS-41 TDMA adapter
{
    lib = "xmsiIS41.so"

    adapterName = "TDMA1"
    #pointCodes = [1005, 1006]
    SSN = 18

    earlyAckMC = false
    earlyAckSME = false

    allowConcatenatedFDA = true
    config = {

        allowIncoming = true
        allowOutgoing = true

        PC = 1005
        SSN = 6
        GT = ""
        TT = 0

        originatingTimeout = 20
        smsreqTimeout = 10
        smdppTimeout = 10

        defaultMessagePriority = "Normal"

        supportIS841 = true

        rimsInterfaceName = "rimsIf"

        pathRetryRandomisation = 1
        pathRetrySegmentOffset = 1

        alarmMask = 0

        defaultTransientFailureCauseCode = 8
        defaultPermanentFailureCauseCode = 7

        causeCodeMap = [
            { SMS_CauseCode = 12, failureCode = 31, permanent = false }
            { SMS_CauseCode = 9, failureCode = 16, permanent = true }
        ]
```

```
            defaultReleaseCause = 13
            defaultReleaseCausePermanent = false

            deliveryFailureErrorClass = 2
            deliveryFailureStatusCode = 5

            protocol = "TDMA"

            tcapInterfaceServiceKey = 22

            incomingOriginatingNumberRules = [
                { fromNoa=3, prefix="04", min=4, max=32, remove=1,
                prepend="0064", resultNoa=4 }
                { fromNoa=2, prefix="4", min=4, max=32, remove=0, prepend="0064",
                resultNoa=4 }
            ]
            incomingDestinationNumberRules = [
            ]
            # outgoingOriginatingNumberRules = [ ]
            # outgoingDestinationNumberRules = [ ]

        } # End IS-41 TDMA adapter config section.

    } # End IS-41 TDMA adapter definition


    # SCA (SIP) adapter
    {
        disabled = False

        interfaceName = "xmsIf"

        lib = "mmxiSCA.so"

        adapterName = "SCA1"

        config = {
            sca = {
                serviceKey = 52

                # interface = "sca"
            }

            pathReportingInterval = 60

            inboundTimeout = 5

            outboundTimeout = 5
        }
    }

    # Internal adapter (The endpoint for the SSMN macro node)
    {
        lib = "xmsiWrapper.so"
        SSN = 40
        adapterName = "Wrapper"

        config = {
            xmsTimeout = 5
            tcapTimeout = 10
            xmsWrapper = {
                interface= "xmsIf"
                pc = 55
                ssn = 7
```

```
                    type = "itu"
                    gt = ""
                }
                xmsTrigger = {
                     pc = 51
                    ssn = 3
                    type = "itu"
                    gt = ""
                }
            }
    ] # adapters
} # xmsTrigger

ChassisActions = {

    SendGenericMessageAction = {
        interfaceName = "xmsIf"
        timeoutTick = 5
        tcapOrigAddr = { PC = 0, SSN = 0, type = "itu" }
        tcapDestAddr = { PC = 0, SSN = 0, type = "itu" }
    }

    ussdChassisAction = {
        tcapInterfaceName = "hssScIf"
        timeoutTick = 5
        recordResponseTimes = false
    }
}

macroNodes = {

    SendShortMessageNode = {
        xmsiWrapperIfName = "xmsIf"

        tcapOrigAddr = { PC = 0, SSN = 0, type = "itu" }
        tcapDestAddr = { PC = 0, SSN = 0, type = "itu" }

        dateFormat = "%A %d %B %Y"
        timeFormat = "%I:%M %p"
        time24Format = "%H:%M %Z"
        callTimeFormat = "%I:%M %p"
        maximumDestinations = 1000

        numberPlan = 1

    }

    SendUSSDNotificationNode = {
        MSISDNTranslationType = 2
        destSSN = 6
        sendMsisdn = false
        sendOrigAddress = false
    }

    ADPBNode = {
        reserveOnFirstsegment = true
        bytesPerSegment = 140
        checkReservationVolumeResponse = true

} # macroNodes

xmsStore = {
    polltime = 10000
```

```
    pstore = {
        enable = true
        cache_size = 10000 # -1 mean no max size
        flush_period = 10
        over_size_max_age_seconds = 60
        max_age_seconds = 30
        max_writes_per_flush = 10
        deferred_delete = true
        userpass = "/"
        interfaceName = "xmsStoreIf"
    }
}
} # XMS
```

xmsWrapper

Syntax:	xmsWrapper = {*wrapper_parameters*}
Description:	Configuration parameters used to create dialogs to the TCAP stack, and to determine the destination address.
Note:	Remote side's configuration. Ignored in normal operation (that is, no xmsWrapperTA).

Messaging Manager Configuration Sections in eserv.config

Introduction

The XMS section of the **eserv.config** provides the configuration parameters for Messaging Manager.

It includes three sections:

- serviceLibrary - for details, see *Configuring Messaging Manager to load ACS Control Plans* (on page 151)
- xmsTrigger - for details, see *xmsTrigger* (on page 167, on page 42)
- macroNodes - for details, see *Configuring Macro Nodes* (on page 140)

High level overview

Here is a high level overview of the XMS section of the **eserv.config**.

Ensure that your **eserv.config** has at least the following sections present in it. If a section is not present, use a text editor to create the section, for completion later.

```
XMS = {

    ServiceLibrary = {
        service_library_section_parameters
    }

    xmsAgent = {
        xmsAgent_section_parameters
    }

    xmsTrigger = {

        SECURE = {
            secure_section_parameters
        }

        xmsTrigger_global_parameters
```

```
        cdr = {
            CDR_section_parameters
        }

        tracing = {
            tracing_section_parameters
        }

        adapters = [
            {first_adapter_parameters
            }

            {second_adapter_parameters
            }

            {next_adapter_parameters...
            }
        ]
    }

    macroNodes = {

        SendShortMessageNode = {
            SendShortMessageNode_parameters
        }

        SendUSSDNotificationNode = {
            SendUSSDNotificationNode_parameters
        }

        ADPBNode = {
            ADPBNode_parameters
        }
    }

    xmsStore = {
        pollTime = int
        pstore = {
            pstore_section_parameters
        }
    }
}
```

xmsAgent

xmsAgent configuration

Here is an example of the `xmsAgent` section of the **eserv.config**.

```
xmsAgent = {
    tcpWaitTimeMilliSec = 10
    rateLimitAlarmIntervalSec = 60
    additionalConnectTimeMilliSec = 0
}
```

xmsAgent parameters

`xmsAgent` accepts the following parameters.

`additionalConnectTimeMilliSec`

Syntax:	`additionalConnectTimeMilliSec = int`
Description:	The number of milliseconds to wait when connecting to a remote socket, such as an SMSC, to allow the SYN ACK to return.
Type:	Integer
Optionality:	Optional (default used if not set)
Allowed:	An integer value in the range of 0 (zero) to 100
Default:	0
Notes:	The value of `additionalConnectTimeMilliSec` should be set as low as possible to minimize the effects of any spikiness in traffic to billing engines.
Example:	`additionalConnectTimeMilliSec = 10`

`rateLimitAlarmIntervalSec`

Syntax:	`rateLimitAlarmIntervalSec = num`
Description:	The number of seconds between rate limiting alarms on a connection (s).
Type:	Integer
Optionality:	Optional (default used if not set).
Allowed:	
Default:	60
Notes:	
Example:	`rateLimitAlarmIntervalSec = 60`

`tcpWaitTimeMilliSec`

Syntax:	`tcpWaitTimeMilliSec = msec`
Description:	The number of milliseconds that xmsAgent will wait on the socket for work.
Type:	Integer
Optionality:	Optional (default used if not set).
Allowed:	
Default:	10
Notes:	
Example:	`tcpWaitTimeMilliSec = 10`

xmsTrigger

xmsTrigger configuration

Here is a high level view of the `xmsTrigger` section of the **eserv.config**.

```
xmsTrigger = {

    SECURE = {
        secure_section_parameters
    }

    SECUREKey = 123456

    xmsAgentInboundServiceKey = 102
    xmsAgentOutboundServiceKey = 103
```

```
overridePluginName = true

pollTime = 100000

loadReportingPeriod = 10

routingScheme = {
    loadIntervalSeconds = 600
}

mtTransactionLifetimeSeconds = 30

oracleusername = "scp"

oraclepassword = "scp"

oracledatabase = ""

dialledStarEncoding = 'D'
dialledHashEncoding = 'E'

mmxIsdn = "0640041234567"
mmxIsdnGprs = "0640041234568"

deliveryReceiptId = "scp1:"
desegmentation = false
desegmentation_timeout = 0
desegmentation_failure_code = 1
desegmentation_failure_cause = 27

baseIDPSize = 200

allowConcatenatedFDA = true
earlyAckMC = true
earlyAckSME = true

alwaysProduceNonDeliveryReceipt = false

limits = {
backoffPeriodMilliseconds = 1000
}


stopRoutingOnTransientFailure = false
stopRoutingOnPermanentFailure = false

processMsgSCI = true

successDeliveryReceiptText = "Your message to <destination> was delivered"
failureDeliveryReceiptText = "Your message to <destination> was NOT delivered"

cdr = {
    CDR_section_parameters
}

tracing = {
    tracing_section_parameters
}

smsAuditing = {
   smsAuditing_section_parameters
}
```

```
    pstore = {
        pstore_section_parameters
    }

    adapters = [
        {first_adapter_parameters
        }

        {second_adapter_parameter>
        }

        {next_adapter_parameters...

    }
}
```

Note: The configuration of the adapters is described in *Configuring Messaging Manager Multigate* (on page 71).

High level parameters

XMS trigger application accepts the following global parameters.

`adapters`

Description: Array of adapters and their parameters within []. See *Configuring the Required Adapters* (on page 71).

`allowConcatenatedFDA`

Syntax: `allowConcatenatedFDA = true|false`

Description: Whether or not we are allowed to perform First Delivery Attempt for concatenated messages (Messages whose segment number is >= 1).

Type: boolean

Optionality:

Allowed: true, false

Default: true

Notes: This can be overridden by each adapter. This option is consulted for each outbound route candidate because it is entirely likely one would want to enable it for IP but disable it for SS7.

Example: `allowConcatenatedFDA = true`

`alwaysProduceNonDeliveryReceipt`

Syntax: `alwaysProduceNonDeliveryReceipt = true|false`

Description: Global Un-Solicited Non-Delivery Receipt Flag.

Causes Messaging Manager to send an un-solicited non-delivery receipt in the event that:

a) Early ack was performed

b) The outgoing message could not be delivered via any route.

The default is false (no response required), however, setting the parameter to true (in the xmsTrigger section of either **eserv.config** or adapter config level) will mean a delivery receipt is sent for every delivery failure.

Type: boolean

Optionality:

Allowed: true, false

Default: false

Notes: This only applies when early ack is on.

See *Setting Early Acknowledgment* (on page 66).

Example:
```
alwaysProduceNonDeliveryReceipt = false
```

backoffPeriodMilliseconds

Syntax:
```
limits = {
   backoffPeriodMilliseconds = seconds
}
```

Description: Rejects new messages or returns errors when XMLTrigger is overloaded.

Type: Integer

Optionality: Optional (default used if not set)

Allowed: 0 to 4294967295

Default: 1000

Notes: 0 disables the feature

Example:
```
limits = {
   backoffPeriodMilliseconds = 1000
}
```

baseIDPSize

Syntax:
```
baseIDPSize = value
```

Description: The base size (in octets) of the IDP without extensions.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 200

Notes: The parameter is used to determine the size of slee events sent from XMS to ACS.

If errors similar to:

SleeException: Overfill event detected (10000)

are produced by xmsTrigger, then increasing the above paramter may resolve them.

Warning: This parameter should not be changed without the consultation of an Oracle support engineer.

Example:
```
baseIDPSize = 300
```

cdr

Description: List of CDR setting and collecting parameters within {}. See *Configuring EDR Collection* (on page 64).

Default: N/A

dialledHashEncoding

Syntax:
```
dialledHashEncoding = "encode"
```

Description: The value to use when a hash (#) is pressed.

Type: String

Optionality: Optional

Allowed:

Default: The default behaviour is to perform no special translation.

Notes: Because different protocols map asterisks and hashes to different BCD numbers (for example, MAP uses C, IS41 uses E) we cannot always rely on TC_PROTOS to get it right.

Warning Make sure these values match with the ones in acs.conf.

Example: `dialledHashEncoding = "E"`

dialledStarEncoding

Syntax: `dialledStarEncoding = "encode"`

Description: The value to use when an asterisk (*) is pressed.

Type: String

Optionality: Optional

Allowed:

Default: The default behaviour is to perform no special translation.

Notes: Because different protocols map asterisks to different BCD numbers (for example, MAP uses B, IS41 uses D) we cannot always rely on TC_PROTOS to get it right.

Warning Make sure these values match with the ones in acs.conf.

Example: `dialledStarEncoding = "D"`

deliveryReceiptId

Syntax: `deliveryReceiptId = "ID"`

Description: String prefixed to delivery receipt ID to ensure that internally generated IDs are unique.

Type:

Optionality:

Allowed: any string

Default: "" (blank)

Notes: Refer to *Delivery Receipts* (on page 180) for details.

This must be hex value only (or blank) when `convertMessageIdToHex (on page 110)` is set to true in an adapter section.

Example: `deliveryReceiptId = ""`

desegmentation

Syntax: `desegmentation = true|false`

Description: Whether or not concatenated messages (with multiple segments) are desegmented into a single message for processing (true), or processed as individual messages (false).

Type: boolean

Optionality:

Allowed: true, false

Default: false

Example: `desegmentation = false`

desegmentation_timeout

Syntax: `desegmentation_timeout= time`

Description: The timeout, for collecting the multiple segments of a concatenated message.

Type: in seconds

Optionality:

Allowed: 0 or a positive integer

Default: 0

Notes: 0 means no timeout is applied.

Example: `desegmentation= false`

desegmentation_failure_code

Syntax: `desegmentation_failure_code = n`

Description: The value of the desegmentation failure code.

Type: Integer

Optionality: Mandatory

Allowed: The result code (error type) to use for desegmentation errors:

1 = transient failure

2 = permanent failure

3 = abort

Default:

Notes: When the desegmentation process fails (for example, if an invalid segment number is received), an error will be sent for any received segments that haven't had a response yet. This does not apply if a segment fails to show up - in that case, a transient failure is returned for the other segments. The error to use is configured by the two desegmentation_failure parameters.

Example: `desegmentation_failure_code = 1`

desegmentation_failure_cause

Syntax: `desegmentation_failure_cause = n`

Description: The value of the release cause to use for desegmentation errors.

Type: Integer

Optionality: Mandatory

Allowed:

Default:

Notes:

Example: `desegmentation_failure_cause = 27`

earlyAckMC

Syntax: `earlyAckMC = true|false`

Description: Determines whether early acknowledgement is enabled for messages that are to be sent to an MC. See *Setting Early Acknowledgment* (on page 66).

Type: boolean

Optionality:

Allowed: true, false

Default: false

Notes:
Example: `earlyAckMC = false`

earlyAckSME

Syntax: `earlyAckSME = true|false`
Description: Determines whether early acknowledgement is enabled for messages that are to be sent to an SME. See *Setting Early Acknowledgment* (on page 66).
Type: boolean
Optionality:
Allowed: true, false
Default: false
Notes:
Example: `earlyAckSME = false`

failureDeliveryReceiptText

Syntax: `failureDeliveryReceiptText = "text"`
Description: The text for an unsuccessful Delivery Receipt
Type: string
Optionality:
Allowed:
Default: "Your message to <destination> was NOT delivered"
Notes: where <destination> substitutes the destination number.
Example: `failureDeliveryReceiptText = "Your message to <destination> was NOT delivered"`

internalDRTimeout

Syntax: `internalDRTimeout = seconds`
Description: The timeout in seconds for processing and sending an internally generated delivery report.
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
Default: 5
Notes:
Example: `internalDRTimeout = 10`

loadIntervalSeconds

Syntax: `loadIntervalSeconds = seconds`
Description: How often (in seconds) to check to see if we need to reload information from the database.
Type: integer
Optionality:
Allowed:
Default: 600

Notes: Reloads are always at intervals of this many seconds after midnight, for example, a value of 600 would cause reload checks at 12:00, 12:10, 12:20 and so on.

If the database has not been changed through the screens, no reload is performed.

Example: `loadIntervalSeconds = 600`

loadReportingPeriod

Syntax: `loadReportingPeriod = seconds`

Description: The maximum SMS rate (SMS per second) is reported in the syslog. The loadReportingPeriod sets the time between these reports.

Type: integer

Optionality: Optional (default used if not set)

Allowed:

Default: 1

Notes:

Example: `loadReportingPeriod = 10`

loadReportChangesOnly

Syntax: `loadReportChangesOnly = true|false`

Description: When set to true, it triggers logging at SMS rate change.

Type: boolean

Optionality: Optional (default used if not set)

Allowed: true, false

Default: false

Notes: This triggers logging when BOTH conditions are true:

[1] log period has expired

[2] last logged SMS rate has changed

Example: `loadReportChangesOnly = false`

mmxIsdn

Syntax: `mmxIsdn = "isdn"`

Description: The Messaging Manager VMSC address that will be returned in the result of a RouteInfo message.

Type: number string

Optionality:

Allowed: valid ISDN

Default:

Notes:

Example: `mmxIsdn = "0640041234567"`

mmxIsdnGprs

Syntax: `mmxIsdnGprs = "sgsn"`

Description: MMX SGSN Address that will be returned in the result of a RouteInfo message, if GPRS support was specified.

Type: number string
Optionality:
Allowed: valid ISDN
Default:
Notes:
Example: `mmxIsdnGprs = "0640041234568"`

mtTransactionLifetimeSeconds

Syntax: `mtTransactionLifetimeSeconds = seconds`
Description: How long correlation transactions between routeInfos and delivers should exist before being removed.
Type: integer
Optionality:
Allowed:
Default: 30
Notes: in seconds
Example: `mtTransactionLifetimeSeconds = 30`

oracledatabase

Syntax: `oracledatabase = "db"`
Description: The Oracle database
Type: String
Optionality: Optional
Allowed:
Default:
Notes:
Example: `oracledatabase = ""`

oraclepassword

Syntax: `oraclepassword = "pw"`
Description: The Oracle password
Type: String
Optionality: Mandatory
Allowed:
Default:
Notes:
Example: `oraclepassword = "scp"`

oracleusername

Syntax: `oracleusername = "name"`
Description: The Oracle user name
Type: String
Optionality: Mandatory
Allowed:
Default:

Notes:

Example: `oracleusername = "scp"`

overridePluginName

Syntax: `overridePluginName = true|false`

Description: Specifies whether the EDR's OAID field is set to the INTERNAL_DR name or the Wrapper Adapter name.

Type: Boolean

Optionality: Optional (default used if not set)

Allowed: true – The EDR's OAID field is set to the INTERNAL_DR name.

false – The EDR's OAID field is set to the Wrapper Adapter name.

Default: false

Notes:

Example: `overridePluginName = true`

processMsgSCI

Syntax: `processMsgSCI = true|false`

Description: Specifies whether Messaging Manager processes Send Charging Information (SCI) messages. Messaging Manager reads SCI messages to determine the data to include in return messages to the switch.

Type: Boolean

Optionality: Optional (default used if not set)

Allowed:

- true – Messaging Manager processes SCI messages.
- false – Messaging Manager ignores SCI messages.

Default: true

Notes:

Example: `processMsgSCI = true`

result

Syntax: `result = result`

Description: Protocol specific result based on the cause value.

Type:

Optionality:

Allowed:

0	Success
1	Transient Failure
2	Permanent Failure
3	Abort

Default: 1

Notes:

Example: `result = 1`

routingScheme

Description: This sub-section supplies parameters for the routing scheme.

Example:

```
routingScheme = {
  loadIntervalSeconds = 600}
```

SECUREkey

Syntax: `SECUREKey = key`

Description: Pre-defined security key.

Type: integer

Optionality:

Allowed:

Default:

Notes: Defined by Oracle

Example: `SECUREKey = 123456`

sendAcsReleaseCause

Syntax: `sendAcsReleaseCause = true|false`

Description: Used to control behaviour of release cause value sent to ACS.

Type: Boolean

Optionality: Optional

Allowed:

- true - Proper mapped value define in “Action and Error Codes” screen will be sent to ACS.
- false - Use legacy behaviour to send xms release causes (e.g. 121) to ACS.

Default: false

Notes:

Example: `sendAcsReleaseCause = true`

singleShotDeliveryReport

Syntax: `singleShotDeliveryReport = true|false`

Description: Indicates if a Delivery Report failure should be retried or not.

Type: Boolean

Optionality: Optional (default used if not set).

Allowed:

- true - use legacy behaviour for internally-generated delivery reports
- false - DR failure is retried

Default: false

Notes: Set true to use legacy behaviour for internally-generated delivery reports, that is, use a single delivery attempt through the inbound transaction. No other processing logic (for example, Trigger/Routing Rules) is applied.

Example: `singleShotDeliveryReport = true`

sleeEventSize

Syntax: `sleeEventSize = bytes`

Description: The number of bytes SLEE event should be. This sets both:

- the size of events requested by xmsTrigger, and
- the size xmsTrigger constructs SLEE events to be.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 2048

Notes: This is the default size of SLEE event to construct.

You need to set a corresponding EVENTSIZE in SLEE.cfg. For more information about SLEE.cfg, see *SLEE Technical Guide*.

Example: `sleeEventSize = 4096`

statistics

Description: List of statistics parameters within {}. See *Collecting Statistics* (on page 70).

stopRoutingOnTransientFailure

Syntax: `stopRoutingOnTransientFailure = true|false`

Description: Used to control how xmsTrigger manages transient failure.

Type: boolean

Optionality: Optional (default used if not set).

Allowed:

true	xmsTrigger will not attempt alternate routing of a message upon receipt of a transient failure.
false	

Default: false

Notes: Refer to Setting ACS Cause to Result.

Example: `stopRoutingOnTransientFailure = false`

stopRoutingOnPermanentFailure

Syntax: `stopRoutingOnPermanentFailure = true|false`

Description: Used to control how xmsTrigger manages permanent failure.

Type: Boolean

Optionality: Optional (default used if not set).

Allowed:

true	xmsTrigger will not attempt alternate routing of a message upon receipt of a permanent failure.
false	

Default: true

Notes:

Example: `stopRoutingOnPermanentFailure = false`

successDeliveryReceiptText

Syntax: `successDeliveryReceiptText = "text"`

Description: The text for the successful Delivery Receipt.

Type: string

Optionality:

Allowed:

Default: "Your message to <destination> was delivered"

Notes: where <destination> substitutes the destination number.

Example: `successDeliveryReceiptText = "Your message to <destination> was delivered"`

tracing

Description: List of SMS tracing parameters within {}. See *Tracing SMSs* (on page 58).

xmsAgentInboundServiceKey

Syntax: `xmsAgentInboundServiceKey = val`
Description: Service key for the xmsAgent, used for inbound SMPP traffic.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: The value must match the service key defined in the *SLEE.cfg* (on page 25) file.
Default: 102 (by install question)
Notes: Listen Socket = inbound.
Example: `xmsAgentInboundServiceKey = 102`

xmsAgentOutboundServiceKey

Syntax: `xmsAgentOutboundServiceKey = val`
Description: Service key for the xmsAgent, used for outbound SMPP traffic.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: The value must match the service key defined in the *SLEE.cfg* (on page 25) file.
Default: 103 (by install question)
Notes: Connect Socket = outbound
Example: `xmsAgentOutboundServiceKey = 103`

xmsStore

Description: Defines the poll interval for SLEE events and the persistent storage parameters in the Pstore sub-section. xmsStore is configured by the following syntax:

```
xmsStore = {
    pollTime = int
    pstore = {
        pstore_parameters
    }
}
```

For information, see *Configuring xmsStore* (on page 67).

SECURE section

Here is an example of the SECURE section configuration.

```
SECURE = {
    limits = {
        maxConcurrentTransactions = 0
        warnConcurrentTransactionsPercentage = 70
        warnConcurrentTransactionsPeriod = 5
        clearConcurrentTransactionsPeriod = 5
        maxSmsPerSecond = 200
        warnSmsPerSecond = 180
        outgoingQueueSize = 2
        outgoingQueueTimeout = 10
    }
    options = ["MCC","SAS","EDR","MCP"]
    gateways = ["MO_SMS","MT_SMS","VAS_SMS"]
    adapters = ["MAP","CDMA","TDMA","EMI","SMPP","API"]

}
```

adapters

Syntax: `adapters = "a1","a2" ...`

Description: Array listing the adapters supported for this installation of MM.

Type:

Optionality:

Allowed:

- MAP
- CDMA
- TDMA
- EMI
- SMPP
- API

Default: "MAP","CDMA","TDMA","EMI","SMPP","API"

Notes:

Example: `adapters = "MAP","CDMA","TDMA","EMI","SMPP","API"`

gateways

Syntax: `gateways = "g1","g2" ...`

Description: Array listing the adapters supported for this installation of MM.

Type:

Optionality:

Allowed:

- MO_SMS
- MT_SMS
- VAS_SMS

Default: "MO_SMS","MT_SMS","VAS_SMS"

Notes:

Example: `gateways = "MO_SMS","MT_SMS","VAS_SMS"`

options

Syntax: `options= "o1","o2" ...`

Description: Array listing the licence options for this installation of MM.

Type:

Optionality:

Allowed:

- MCC - Message Charging Control (billing)
- SAS - SMS Anti-Spam Screening
- EDR - Express Delivery Routing (also known as FDA)
- MCP - Message Control Pack (also known as AMC, triggering to ACS)
- API - 'Internal' protocol, that is, wrapper adapter

Default: "MCC","SAS","EDR","MCP","API"

Notes: If the line is not present, all options are enabled, but maxConcurrentTransactions is limited to 5.

Example: `adapters = "MCC","SAS","EDR","MCP","API"`

limits

The following parameters set the performance limits of the system. These are pre-defined by Oracle and should not be changed.

`clearConcurrentTransactionsPeriod`

Syntax:	`clearConcurrentTransactionsPeriod = period`
Description:	How often to report that we are below warnConcurrentTransactionsPercentage concurrent transactions.
Type:	
Optionality:	
Allowed:	
Default:	10
Notes:	In seconds
Example:	`clearConcurrentTransactionsPeriod = 10`

`maxConcurrentTransactions`

Syntax:	`maxConcurrentTransactions = period`
Description:	The maximum number of concurrent transactions that can be handled.
Type:	integer
Optionality:	
Allowed:	
Default:	5
Notes:	
Example:	`maxConcurrentTransactions = 5`

`warnConcurrentTransactionsPercentage`

Syntax:	`warnConcurrentTransactionsPercentage = pc`
Description:	Percentage of maxConcurrentTransactions at which to emit a warning.
Type:	integer
Optionality:	
Allowed:	
Default:	90
Notes:	
Example:	`warnConcurrentTransactionsPercentage = 90`

`warnConcurrentTransactionsPeriod`

Syntax:	`WARNConcurrentTransactionsPeriod = period`
Description:	How often to report we are over warnConcurrentTransactionsPercentage concurrent transactions.
Type:	
Optionality:	
Allowed:	
Default:	5
Notes:	In seconds
Example:	`WARNConcurrentTransactionsPeriod = 5`

maxSmsPerSecond

Syntax:	`ArraySize = num`
Description:	The maximum number of transactions per second we will accept before throttling.
Type:	Integer
Optionality:	Mandatory
Allowed:	
Default:	1000
Notes:	
Example:	`maxSmsPerSecond = 200`

warnSmsPerSecond

Syntax:	`warnSmsPerSecond = num`
Description:	The maximum number of transactions per second before warning that throttling is imminent.
Type:	Integer
Optionality:	Mandatory
Allowed:	
Default:	80% of maxSmsPerSecond
Notes:	
Example:	`warnSmsPerSecond = 180`

outgoingQueueSize

Syntax:	`outgoingQueueSize = num`
Description:	The size of the outgoing queue.
Type:	Integer
Optionality:	
Allowed:	
Default:	1000
Notes:	
Example:	`outgoingQueueSize = 2`

outgoingQueueTimeout

Syntax:	`outgoingQueueTimeout = num`
Description:	The timeout duration for an outgoing queue.
Type:	Integer
Optionality:	Optional
Allowed:	
Default:	
Notes:	The value is usually in seconds.
Example:	`outgoingQueueTimeout = 10`

Tracing SMSs

How does tracing work?

The Messaging Manager tracing feature allows individual messages to be identified and traced.

To activate tracing, the tracing section of the xmsTrigger configuration must be set to enabled, and the prefixes to be traced must be defined. Tracing output is written to file at a specified interval.

The tracing section of the **eserv.config** file is required, but may be empty. In this case the tracing enabled parameter will be set to false by default. For further details on how tracing works, please refer to the *Tracing* (on page 177) section of the Messaging Manager Processes chapter.

Tracing configuration

The `tracing` section of the **eserv.config** determines the SMSs that are to be traced, if any.

```
tracing = {

    enabled = true
    showPrivate = true
    outputFile = "/tmp/xmsTrigger.trc"
    outputFileCycle = 512

    maxFileSizeKB = 0
    maxNumFiles = 4

    shmKey = 92403
    shmSizeKb = 64

    callsPerMinute = 2

    origAddress = [
        "0064212",
        "0064213",
        "0064214"
    ]

    destAddress = [
        "0064213",
        "0064214"
    ]

    useTONNPI = true

}
```

Tracing parameters

Here are the tracing parameters.

`callsPerMinute`

Syntax:	`callsPerMinute = num`
Description:	The maximum number of calls per minute to trace.
Type:	Integer
Optionality:	
Allowed:	
Default:	2
Notes:	

Example: `callsPerMinute = 2`

destAddress

Syntax: `destAddress = ["add1","add2"]`

Description: Array of destination addresses for which calls will be traced.

Type:

Optionality:

Allowed:

Default: Refer to Prefix parameters.

You can specify "0" for all numbers.

Notes: Prefixes in this list (*TonNpi* format) must be enclosed in double quotes; that is,"tttnnn"

Example:
```
destAddress = [
  "0064213",
  "0064214"
]
```

enabled

Syntax: `enabled = true|false`

Description: Determines whether Messaging Manager collects tracing details of numbers whose prefixes match those specified in the origAddress and destAddress parameter lists.

Type: Boolean

Optionality:

Allowed: true, false

Default: false

Notes:

Example: `enabled = false`

maxFileSizeKB

Syntax: `maxFileSizeKB = size`

Description: The maximum file size, in KB.

Type: Integer

Optionality:

Allowed:

Default: 0 (unlimited)

Notes:

Example: `maxFileSizeKB = 0`

`maxNumFiles`

Maximum number of additional files.

Default: 4

Note: If:

- maxFileSizeKB is > 0 and
- maxNumFiles is > 0

then every outputFileCycle, a check to see if the outputFile size is > maxFileSizeKB is made.

If max file size is exceeded, the trace file is renamed to outputFile.*N* where *N* is 1 to maxNumFiles.

If all *N* files exist, then the oldest file is overwritten.

`origAddress`

Syntax: `origAddress = ["add1","add2"]`

Description: Array of originating addresses for which calls will be traced.

Type:

Optionality:

Allowed:

Default: Refer to Prefix parameters.

You can specify "0" for all numbers.

Notes: Prefixes in this list (*TonNpi* format) must be enclosed in double quotes; that is,"tttnnn"

Example:

```
origAddress = [
  "0064212",
  "0064213",
  "0064214"
]
```

`outputFile`

Syntax: `outputFile = "file"`

Description: The file that all tracing details are to be written to. The primary output file. Refer to note on "maxNumFiles".

Type:

Optionality:

Allowed:

Default: "/tmp/smsTrace.trc"

Notes: Once established, Messaging Manager cannot remove or rename the tracing output file.

Example: `outputFile = "/tmp/smsTrace.trc"`

`outputFileCycle`

Syntax: `outputFileCycle = num`

Description: Close and re-open the file every *N* calls. Checks for file size exceeded at this time also.

Type:

Optionality:

Allowed:

Default:	1024
Notes:	Tracing buffered data is flushed to the output file before closing.
Example:	`outputFileCycle = 1024`

`shmKey`

Description:	Shared memory key.
Note:	Not currently supported.

`shmSizeKb`

Description:	Size of shared memory, in KB.
Note:	Not currently supported.

`showPrivate`

Syntax:	`showPrivate = true|false`
Description:	Enables the viewing of the message payload in the tracing output.
Type:	Boolean
Optionality:	Optional
Allowed:	true — Allows viewing of payload. false — Bars viewing of payload.
Default:	false
Notes:	
Example:	`showPrivate = true`

`useTONNPI`

Syntax:	`useTONNPI = true|false`
Description:	Whether or not to include TON and NPI in the address for trace prefix matching.
Type:	Boolean
Optionality:	Optional
Allowed:	true, false
Default:	
Notes:	
Example:	`useTONNPI = true`

Auditing SMSs

How does auditing SMS text work?

The Messaging Manager auditing feature allows SMS text to be audited.

To activate auditing, the smsAuditing section of the xmsTrigger configuration must be set to enabled.

The audit line will have the following format:

```
timestamp <origination number>, <destination number>, message text
```

The smsAuditing section of the **eserv.config** file is required, but may be empty. In this case the tracing enabled parameter will be set to false by default.

If the **logFinalAuditLine** parameter is set to **true**, then audit line will have the following format:

```
timestamp <origination number>, <destination number>, message text, status, retry
count
```

Auditing Configuration

The `smsAuditing` section of the **eserv.config** determines the SMSs that are to be audited, if any.

```
smsAuditing = {
    # Is auditing enabled? (default false)
        enabled = false

    # Output file.
    #-------------------------------------------------------#
    #                                                       #
    # IMPORTANT NOTE:                                       #
    # if the output file name is changed,                   #
    # make sure the changes are reflected in                #
    # /IN/service_packages/ACS/etc/logjob.conf              #
    # Otherwise the audit file will not be archived         #
    #                                                       #
    #-------------------------------------------------------#

       outputFile = "/IN/service_packages/XMS/tmp/xmsTrigger.aud"

    # Close and re-open the file every N calls.  Check for file size exceeded
    # at this time also.

        outputFileCycle = 32

    # should log originating address ? ( default: true)
        logOriginatingAddress = true

    # should log destination address ? ( default: true)
        logDestinationAddress = true

    # should log FinalAuditLine ? ( default: false)
        logFinalAuditLine = true

} #smsAuditing
```

Auditing Parameters

Here are the auditing parameters.

`enabled`

Syntax:	`enabled = true|false`
Description:	Determines whether Messaging Manager audits SMS messages.
Type:	Boolean
Optionality:	
Allowed:	
Default:	false
Notes:	
Example:	`enabled = true`

outputFile

Syntax: `outputFile = "file"`
Description: The file that all SMS texts are to be written to.
Type:
Optionality:
Allowed:
Default: /IN/service_packages/XMS/tmp/xmsTrigger.aud
Notes:
Example: `outputFile = /IN/service_packages/XMS/tmp/xmsTrigger.aud`

outputFileCycle

Syntax: `outputFileCycle = "num"`
Description: Close and re-open the file every N calls. Checks for file size exceeded at this time.
Type: Number
Optionality:
Allowed:
Default: 32
Notes: Auditing buffered data is flushed to the output file before closing.
Example: `outputFileCycle = 32`

logOriginatingAddress

Syntax: `logOriginatingAddress = true|false`
Description: Determines whether logged sms messages should include originating address
Type: Boolean
Optionality:
Allowed:
Default: true
Notes:
Example: `logOriginatingAddress = true`

logDestinationAddress

Syntax: `logDestinationAddress = true|false`
Description: Determines whether logged sms messages should include destination address.
Type: Boolean
Optionality:
Allowed:
Default: true
Notes:
Example: `logDestinationAddress = true`

`logFinalAuditLine`

Syntax:	`logFinalAuditLine = true|false`
Description:	Determines whether to log final SMS audit line in the following format: `timestamp <origination number>, <destination number>, message text, status, retry count`
Type:	Boolean
Optionality:	
Allowed:	
Default:	false

Notes:	When this flag is set to **false**, mmx writes the SMS audit line for every path in the following format:
	`timestamp <origination number>, <destination number>, message text`
	When this flag is set to **true**, mmx writes the final SMS audit line in the following format:
	`timestamp <origination number>, <destination number>, message text, status, retry count`

Example:	`logFinalAuditLine = false`

Configuring EDR Collection

Configuring EDR collection

Messaging Manager will produce EDRs to be used in post processing as required. All EDR configuration is done in the `cdr` section of the **eserv.config** file. These EDRs will be saved to file in the location specified in the **eserv.config**.

EDRs are saved to file in tag/value pairs, separated by "|", in the following form:

```
tag1=value1|tag2=value2
```

Please note that there are some parameters that may not be changed after startup. These parameters are as follows:

- destdir
- num_files
- num_files_leaf

All other `cdr` parameters may be changed as required.

EDR configuration example

The following section sets the EDR collection and configuration for Messaging Manager. For further details on the generation and format of EDRs, please refer to the *EDRs* (on page 179) section of the Messaging Manager Processes chapter.

```
cdr = {

    log = true
    filename = "xms_"
    tempdir = "/IN/service_packages/XMS/cdr/current/"
    maxno = 10000
    time_out = 1800
    destdir = "/IN/service_packages/XMS/cdr/closed/"
    num_files = 64000
    num_files_leaf = 64
}
```

EDR parameters

Here are the EDR parameters.

`destdir`

Syntax: `destdir = "dir"`
Description: The base filestore directory for completed EDR log files.
Type: String
Optionality: Mandatory
Allowed: Valid directory path
Default: `"/IN/service_packages/XMS/cdr/closed/"`
Notes: This parameter may not be changed after installation and initial setup.
Example: `destdir = "/IN/service_packages/XMS/cdr/closed/"`

`filename`

Base filename used to create EDR log files.

Allowed: alphanumeric string
Note: The format log file name is filename + "YYYYMMDDHHMMSS.cdr"

`log`

Turns EDR logging on or off.

Default: false
Allowed: true, false

`maxno`

Maximum number of EDRs per file.

Default: 10000

`num_files`

Number of expected EDR files.

Default: 64000
Note: This parameter may not be changed after installation and initial setup.

`num_files_leaf`

Number of EDR files per leaf directory.

Default: 64
Note: This parameter may not be changed after installation and initial setup.

`tempdir`

Syntax: `tempdir = "dir"`
Description: The temporary directory for the working EDR log file.
Type: String
Optionality: Mandatory
Allowed: Valid directory path
Default: `"/IN/service_packages/XMS/cdr/current/"`

Note: A side effect of the CDR processing code of MMX is to remove all files in the tempdir when MMX starts.

Example: `tempdir = "/IN/service_packages/XMS/cdr/current/"`

`time_out`

Close EDR file after time_out (in seconds).

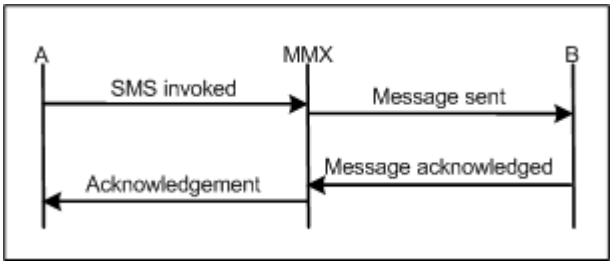
Default: 1800

Setting Early Acknowledgment

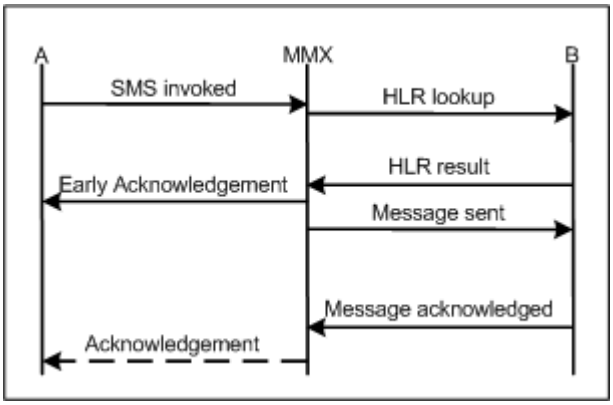
What is early ack?

There are two ways in which a delivery acknowledgement can be sent in MMX.

- MMX does not acknowledge the sending party until the message has been successfully sent. However, this cannot work in all cases, because some networks have a normal HLR lookup time that exceeds 3 seconds, and an MSC response time that exceeds 2 seconds. Forcing the sending party to wait during an FDA attempt both causes excessive use of network resources and alters the user experience.



- The alternative is to mimic the SMSC functionality and acknowledge the message the moment that the recipient appears valid. This is called an early ack. However this carries the risk that messages will be lost before they are delivered, as there is no persistence or replication.



Early ack configuration

If desired, earlyAck parameters can be defined at trigger level, and/or adapter level. The values defined at adapter level will override those defined at Trigger Level.

Note: The earlyAck parameters at adapter level are read from the adapter section of the **eserv.config** file for the outbound adapter. (Not the inbound adapter).

Messaging Manager will produce an early ack when the message is accepted by the outbound adapter for delivery, where early ack is on for one of the following:

- SME – direct delivery
- MC – delivery to a Message Center

By default, if a Nack is received later then it will not be passed back to the sender and subsequent routing will also not send an Ack.

```
XMS = {
    xmsTrigger = {
        other_Messaging Manager_configuration_options

        earlyAckSME = false
        earlyAckMC = false

        alwaysProduceNonDeliveryReceipt = false
    }
}
```

Early ack options

This feature is configurable for each adapter instance to either be:

Option	Parameter settings
Always on	`earlyAckMC = True` `earlyAckSME= True`
Apply when a “send to SME (MS or ASP)” is first encountered	`earlyAckMC = False` `earlyAckSME= True`
Always off	`earlyAckMC = False` `earlyAckSME= False`

Early ack functionality support

The early ack functionality is supported under:

- MAP 1, 2 and 3
- SMPP 3.4 and 5.0,
- EMI/UCP 4.0
- IS-41

Configuring xmsStore

About xmsStore configuration

The xmsStore configuration in the xmsTrigger section of the **eserv.config** file sets the poll interval for SLEE events, and includes configuration for persitent storage. The xmsStore configuration uses the following syntax:

```
xmsStore = {
    pollTime = int
    pstore = {
        pstore_parameters
    }
}
```

For more information about pstore parameters, see *Setting Pstore* (on page 68).

`pollTime`

Syntax:	`pollTime = time`
Description:	Cycle period to check for activity for TCP/IP sockets.
Type:	integer
Optionality:	
Allowed:	Any positive, non zero integer.
Default:	100000
Notes:	in milliseconds
Example:	`pollTime =` 100000

Setting Pstore

Pstore configuration

Persistent storage is used for long term storage of information. Information stored using pstore will be stored for as many days as required; this is set using the `pstore` configuration in **eserv.config**.

Where both early ACK and a delivery receipt have been requested for EMI messages, pstore is used to store the message id that was sent by Messaging Manager. When the message is delivered by the SMSC, it returns to message id to Messaging Manager. pstore is used to translate the SMSC message_id to the Messaging Manager generated message_id and send on in the delivery receipt.

Changing the configuration

When the configuration is changed you need to send a SIGHUP to xmsTrigger with pstore to reread the configuration file.

- `enable = false` in **eserv.config**, disables pstore, flushes cache, deletes pstore, then removes the database connection.
- `enable = true` in **eserv.config**, causes xmsTrigger to reread the pstore config, updates oraPStore with the values, and re-establishes a database connection.

Pstore configuration example

Here is an example of the `pstore` sub-section of `xmsTrigger`.

```
pstore = {
    enable = true
    cache_size = 10000 # -1 mean no max size
    flush_period = 10
    interfaceName = "xmsStoreIf"
    over_size_max_age_seconds = 60
    max_age_seconds = 30
    max_writes_per_flush = 10
    deferred_delete = true
    userpass = "/"
}
```

Pstore parameters

Here are the parameters in the `pstore` sub-section of `xmsTrigger`.

`cache_size`

Maximum number of elements in cache.

Default: 10000

Note: This value is only read on startup.

`deferred_delete`

If true, do not actually delete items from the database, just mark them as deleted.

Default: true

Allowed: true, false

`enable`

Enable or disable persistent storage.

Default: false

Allowed: true, false

`flush_period`

Time, in seconds, between flushing cache to DB.

Default: 300

Note: This value is only read on startup.

`interfaceName`

Syntax: `interfaceName = "ifname"`

Description: The name of the SLEE interface providing xmsStore services.

Type: String

Optionality: Optional (default used if not set).

Allowed:

Default: "xmsStoreIf"

Notes:

Example: `interfaceName = "xmsStoreIf"`

`max_age_seconds`

Minimum time that the item may reside in the cache (in seconds) before it can be flushed to the database.

Default: 300

`max_writes_per_flush`

Maximum number of records to write per Oracle transaction when flushing. This stops the process blocking on Oracle writes (which can cause xmsTrigger to be killed by the watchdog).

Default: -1

Note: -1 means no limit

`over_size_max_age_seconds`

Flush items older than this number of seconds when cache is overfull.

Default: 60

`userpass`

Username and password login to use to attach to the Oracle database if the host application is not already logged in.

Default:	"/"
Note:	This value is only read on startup.

Collecting Statistics

Editing the statistics configuration

When configured MM will gather extensive operational statistics. Using the Convergent Charging Controller SMS application, these statistics are stored in the database for subsequent reporting.

For a full list of the MM statistics that will be gathered please refer to the *Statistics* (on page 168) topic in the Background Processes chapter.

Statistics parameters

To collect statistics, there is a single parameter that must be set to true in the `xmsTrigger` section of the **eserv.config** file.

`enable`

Determines whether Messaging Manager collects operational statistics.

Default:	false
Allowed:	true, false

Defining the Screen Language

Introduction

The default language file sets the language that the Java administration screens start in. The user can change to another language after logging in.

The default language can be changed by the system administrator.

By default, the language is set to English. If English is your preferred language, you can skip this step and proceed to the next configuration task, Defining the Help Screen Language.

Example Screen Language

If Dutch is the language you want to set as the default, create a soft-link from the **Default.lang** file to the **Dutch.lang** file.

Chapter 3

Configuring Messaging Manager Multigate

Overview

Introduction

This chapter explains how to configure the adapters in Messaging Manager Multigate.

In this chapter

This chapter contains the following topics.

Configuring the Required Adapters

Limitation

All adapters have a section in this part of the configuration file. The array of adapters section starts as below, and is followed by a section for each available adapter.

```
#
    adapters = [
```

Note: This technical guide documents all adapters available from Convergent Charging Controller. By documenting all available adapters, Oracle in no way commits to supplying the software thus described.

Convergent Charging Controller will supply only the adapters purchased by the customer; thus configuration for each installation will differ.

Adapter overview

There are a number of interfaces that provide communication between the Messaging Manager platform and the ASPs and SMSCs. These interfaces are provided as adapter interfaces. Each of these interfaces performs several roles:

1. Receive message send requests from an ASP or MS
2. Deliver outbound short messages to an SMSC or MS, and return a delivery result to the application
3. Receive delivery notifications from an SMSC or MS, and forward to MM
4. Relay delivery notifications to an ASP or MS

MM may be configured to use as many adapters as required. Each adapter receives and transmits a different protocol.

Which adapters do I need?

This will have been determined at the time of purchasing Messaging Manager. There are several different adapters available:

- MAP - MAP adapter for GSM networks
- IS-41 CDMA - IS-41 adapter for CDMA networks
- IS-41 TDMA - IS-41 adapter for TDMA networks
- SCA - SCA adapter for support of SIP instant messaging
- SMPP - SMPP adapter for ASP/SMSC proxy connections
- EMI - UCP/EMI adapter for ASP/SMSC proxy connections

Additional adapters may be purchased at a later date as required to accommodate your changing network needs.

Number normalization

People deal with (and a database usually stores) telephone numbers in their normalized form, for example, 00441918666223. The network however gives and receives numbers in a denormalized form, that is, where the type of number (the Nature of Address) is known explicitly, for example, [International, 441918666223] for the previous example.

The number rules parameters in the adapters conform to the number normalization configuration described below.

Example:

Normalized number:	049393434	NA
De-Normalized number:	Nature of Address:	National
NA	Digits:	49393434

Possible Natures of Addresses:

This table shows the different Normalization NoA values depending on the adapter type.

NoA	MAP	EMI
Subscriber number	1	4
Unknown	2	0
National	3	2
International	4	1
Network specific	NA	3

Normalization parameters

Enter a conversion rule for each incoming NOA. This rule determines how to convert to the normalized number and a corresponding rule converts back to a denormalized number on the outgoing side.

Number normalization rules are in the following format:

```
{ fromNoa=noa, targetNoa=noa, prefix="digits"[, min=len][, max=len], remove=num,
prepend="digits", resultNoa=noa }
```

Here are the configuration parameters for number normalization.

fromNoa

Syntax: `fromNoa = noa`

Description: The original nature of address (NOA) that the number is received from. This is prior to normalization and denormalization.

Type:

Optionality:

Allowed:

Default:

Notes:

Example: `fromNoa = 306`

max

Syntax: `max = len`

Description: The maximum number length.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 999

Notes: Used in *number normalization* (on page 72) and rules.

Example: `max = 32`

min

Syntax: `min = len`

Description: The minimum number length.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 0

Notes: Used in *number normalization* (on page 72) and rules.

Example: `min = 4`

prefix

Syntax: `prefix = "digit"`

Description: This rule is applied to numbers with this prefix.

Type: String

Optionality: Optional

Allowed: One or more decimal digits

Default:

Notes: Used in *number normalization* (on page 72) and rules.

Example: `prefix = "25"`

prepend

Syntax: `prepend= "digits"`

Description: Determines the digits that are to be prepended to the number, after stripping any as specified previously.

Type: String

Optionality:

Allowed:

Default:

Notes: Used in *number normalization* (on page 72) and rules.

Example: `prepend = "1111"`

remove

Syntax: `remove = num`

Description: Determines the number of digits that are stripped from the beginning of the number.

Type: Integer

Optionality:

Allowed:

Default:

Notes: Used in *number normalization* (on page 72) and rules.

Example: `remove = 2`

resultNoa

Syntax: `resultNoa = noa`

Description: Resulting NOA after the normalization.

Type:

Optionality:

Allowed:

Default:

Notes: Used in *number normalization* (on page 72) and rules.

Example: `resultNoa = 4`

targetNoa

Syntax: `targetNoa=noa`

Description: The target NOA.

Type:

Optionality:

Allowed:

Default: 999

Notes: EMI only.

If unspecified, defaults to 999.

Used in *number normalization* (on page 72) rules.

Example: `targetNoa=999`

NOA and Normal rules

The NOA (nature of address) is a classification to determine in what realm (Local, National or International) a given phone number resides, for the purposes of routing and billing.

Details vary between different implementations of telephone systems, but the following table is representative.

Dialed Digits	NOA (aka NOC, NON)	Definition
477 9425	1 → Subscriber	Number within local telephone exchange
4 477 9425	3 → National	Number within country telephone exchange
64 4 477 9425	4 → International	Number within world telephone exchange
477 9425	2 → UNKNOWN	Numbering Scheme rule → Subscriber
0 4 477 9425	2 → UNKNOWN	Numbering Scheme rule → National
00 64 4 477 9425	2 → UNKNOWN	Numbering Scheme rule → International

In essence, the subscriber's telephone system *may* try to ascertain the nature by examining the dialed digits. If they can be understood by "built-in" mechanisms, the NOA can unambiguously be one of the values - Subscriber, National, International, or a finer classification determined by the protocol variant.

Otherwise the NOA is unknown and the dialed digits must be disambiguated by a set of (usually simple) rules specified by a Numbering Scheme.

Leading zeros are used in many countries, but the leading characters could be any arbitrary sequence that the numbering scheme could specify.

Ultimately the usage of NOA is determined by the phone network itself which may classify and possibly modify a phone number while it is being transmitted between the service logic and the switch.

Configuring the MAP Adapter

MAP adapter overview

The MAP adapter communicates between the Messaging Manager platform and:

- HLR and MSC (in FDA situations)
- SMSCs using MAP

In a system using MAP, the MAP adapter must be configured to know of all entities that may connect to it, and all entities that it may connect to.

The MAP adapter is compliant with several versions of MAP:

- Map v1
  - ETSI GTS 09.02 V3.11.0 (1995-01)
  - ETSI GTS 03.40 V3.7.0 (1995-01)
- Map v2
  - ETSI ETS 300 599 ed.3 (1996-09) - GSM 09.02
  - ETSI ETS 300 536 ed.4 (1996-10) - GSM 03.40
- Map v3
  - 3GPP/ETSI GSM TS 09.02 version 7.5.0 Release 1998
  - 3GPP/ETSI GSM TS 03.40 version 7.5.0 Release 1998

General MAP configuration

The following section of the **eserv.config** specifies the general parameters for all MAP connections.

```
adapters = [

    # First adapter (MAP)
    {
        lib = "xmsiMap.so"
        adapterName = "MAP1"

        pointCodes = [1001, 1002]
        SSN = 8

        earlyAckMC = true
        earlyAckSME = true

        allowConcatenatedFDA = true
        alwaysProduceNonDeliveryReceipt = false

        config = {
            global_parameters
        }
    }
]
```

General MAP parameters

Here are the general parameters for MAP.

`adapterName`

Syntax:	`adapterName = "name"`
Description:	Identifier for the adapter.
Allowed	Any text string, but should be meaningful, for example include the protocol used. For example: • "MAP1" for MAP • "EMI1" for EMI • "SCA1" for SCA • "SMPP1" for SMPP • "CDMA1" for IS-41 CDMA • "TDMA1" for IS-41 TDMA • "Wrapper" for Wrapper
Default:	No default.
Notes:	This name *must* also be in the configuration database before Messaging Manager will run correctly. See *MM User's Guide*.

`alwaysProduceNonDeliveryReceipt`

Local Unsolicited Non-Delivery Receipt flag.

Causes MM to send an unsolicited non-delivery receipt in the event that:

a) Early ack was performed

b) The outgoing message could not be delivered through any route.

This setting overrides the global flag.

Default:	Global *alwaysProduceNonDeliveryReceipt* (on page 44) value
Allowed:	true, false

Note: This only applies when early ack is on.
See *Setting Early Acknowledgment* (on page 66).

`config`

The parameters in this sub-section below this give the configuration for all messages for this adapter.

`lib`

Syntax: `lib = "library"`
Description: The name of the library that contains the MAP adapter being configured.
Allowed "xmsiMap.so"
Default: No default.
Example: `lib = "xmsiMap.so"`

`pointCodes`

Syntax: `pointCodes = [pc1, pc2, ...]`
Description: Destination point codes array of messages to be handled by this adapter. This parameter takes priority over SSN match.
Optionality: Optional
Allowed Defined by network administrator.
Example: `pointCodes = [1001, 1002]`

`SSN`

Syntax: `SSN = num`
Description: Destination subsystem number of messages to be handled by this adapter.
Allowed Valid subsystem number
Notes: Non-zero to handle incoming TCAP.
Example: `SSN = 18`

Global MAP configuration

The following section of the **eserv.config** specifies the global parameters for a MAP connection.

```
config = {

    disableConcatenatedSegmentPad = false

    abortMessagesWithZeroLengthTPDA = true

    allowIncoming = true
    allowOutgoing = true
    allowDirectDelivery = true

    allowIncomingMap3 = true

    allowRetryWhereRimsAndHLRLookupEqual = false

    allowUserRequestedDeliveryReceipt = true

    suppressRimsMap3imsiOAUpdate = false

    lastSegmentDeliveryReceiptOnly = false
```

```
tcapInterfaceServiceKey = 22

originatingTimeout = 10

smscTimeout = 8
hlrTimeout = 3
mscTimeout = 15

rimsInterfaceName = "rimsIf"

gprsSupport = "supported"

nonGprsAdapter = "MAP2"

sgsnPrefixes = [ "000" ]

doProtocolIdMapping = false
defaultProtocolId = 0
protocolIdMap = [
    { in = 127, out = 0 }
]

fallbackAlphabet = "UCS-2"

scheduledDeliveryTime = ""

defaultMessagePriority = "Normal"

PC = 55
SSN = 8

GT = "5114406267
TT = 0

relayTranslationType = -2
relayGlobalTitleType = 2
relaySSN = -1
relayNatureOfAddress = 4

GTMap = [
    { prefix = "0010019198", value = "919827002402" }
]

#SCA = "5114406267"

natureOfAddress = 1 # international
numberPlan = 6      # E.212

defaultMapVersionSmsc = 3
defaultMapVersionMsc = 3


deliveryFailureStatusCode = 64

throttledDeliveryFailureCause = 96 # congestion

mscVersionCacheSize = 1000

hybridiseMapVersions = false

maxUnsegmentedLength = 120

splitLongMessages = true
```

```
        alarmMask = 0

        honourReplyPath = false

        pathRetryRandomisation = 1
        pathRetrySegmentOffset = 1

        localTimeZone = "UTC"

        hlrErrorMap = [
            { error = 1, permanent = true }
            { error = 32, permanent = false }
        ]

        defaultTransientFailureErrorCode = 32
        defaultPermanentFailureErrorCode = 32

        incomingOriginatingNumberRules = [
            { fromNoa=2, prefix="04", min=4, max=32, remove=1,
            prepend="0064", resultNoa=1 }
            { fromNoa=3, prefix="4", min=4, max=32, remove=0, prepend="0064",
            resultNoa=1 }
        ]
        incomingDestinationNumberRules = [
        ]
        # outgoingOriginatingNumberRules = [ ]
        # outgoingDestinationNumberRules = [ ]

        # privateExtensions = [
        #        {extId="1,2,776,8,8,8,8,11", asn1Tags= [ 0xE0, 0x80] ,
        profileTagID=327814, name = "imei" }
        # ]

    }
```

Global MAP parameters

Here are the parameters for configuring the global MAP adapter.

`alarmMask`

Alarm masking.

Default: 0

Allowed: Sum of:

1	Mask (that is, do not generate) originating timeout alarm
2	Mask alarm for Messaging Manager-originated abort
4	Mask alarm for timeout waiting for HLR
8	Mask alarm for TCAP ABORT from HLR
16	Mask alarm for timeout waiting for MSC
32	Mask alarm for TCAP ABORT from MSC
64	Mask alarm for timeout waiting for SMSC
128	Mask alarm for TCAP ABORT from SMSC

`abortMessagesWithZeroLengthTPDA`

Syntax: `abortMessagesWithZeroLengthTPDA = true|false`

Description: Whether or not MM shall return a TCAP U-ABORT to the MSC upon receipt of a message with a zero length TP Destination-Address.

Type:	Boolean
Optionality:	Optional
Allowed:	true, false
Default:	true
Notes:	If set to false, MM sends a SM Delivery Failure with the cause set to 'InvalidSMEAddress'.
Example:	`abortMessagesWithZeroLengthTPDA = true`

`allowIncoming`

Syntax:	`allowIncoming = true|false`
Description:	Determines whether inbound messages are allowed.
Optionality:	Mandatory
Allowed	true, false
Default	false
Example:	`allowIncoming = false`

`allowOutgoing`

Syntax:	`allowOutgoing = true|false`
Description:	Determines whether initial messages may be sent out through the MAP interface.
Optionality:	Mandatory
Allowed	true, false
Default	false
Example:	`allowOutgoing = false`

`allowDirectDelivery`

Determines whether the adapter may deliver directly to MSCs.

Default:	false
Allowed:	true, false
Note:	Required parameter.

`allowIncomingMap3`

Whether to allow MAP v3 inbound processing.

Default:	true
Allowed:	true, false

`allowUserRequestedDeliveryReceipt`

Whether resending a message when a cached lookup and a HLR lookup return the same MSC/IMSI/LMSI is allowed. Useful when the MSC is not behaving (e.g. returning absent subscriber). Typically, resending a message is not needed after failing once to reach the subscriber. If false, do not allow resending a message when cached and HLR lookups return the same identifier.

Default:	false
Allowed:	true, false
Note:	Required parameter.

`allowRetryWhereRimsAndHLRLookupEqual`

Whether to allow MAP v3 inbound processing.

Default: true

Allowed: true, false

`defaultMapVersionMsc`

MAP version to use for communication with the MSC.

Default: 3

Note: If defaultMapVersionMsc is missing, it tries to find defaultMapVersion for backwards compatibility. If that is also missing uses default value.

`defaultMapVersionSmsc`

MAP version to use for communication with the SMSC.

Default: 3

Note: If defaultMapVersionSmsc is missing, it tries to find defaultMapVersion for backwards compatibility. If that is also missing, uses default value.

`defaultMessagePriority`

Syntax: `defaultMessagePriority = "str"`

Description: The priority to be applied to incoming messages that do not have a priority.

Type: String

Optionality: Optional (default used if parameter absent).

Allowed

Default "Normal"

Notes:

Example: `defaultMessagePriority = "Normal"`

`defaultPermanentFailureErrorCode`

The default MAP error code used for permanent failures when no mapping from CS1 release cause is specified.

Default: 32

Allowed: See *deliveryFailureStatusCode* (on page 82) for list of values.

`defaultProtocolId`

If a protocol ID is found that is not in the protocolIdMap, this is the result value.

Default: 0

Note: Optional parameter

`defaultTransientFailureErrorCode`

The default MAP error code used for transient failures when no mapping from CS1 release cause is specified.

Default: 32

Allowed: See *deliveryFailureStatusCode* (on page 82) for list of values.

`deliveryFailureStatusCode`

Sets the default status code to be used in a failure status report.

Default: 64

Allowed: 0 to 127. See table of values for value meaning.

Value	Description
0	Short Message received by the SME
1	Short message forwarded by the SC to the SME but the SC is unable to confirm delivery
2	Short message replaced by the SC Reserved Values
3-15	Reserved
16-31	Values specific to each SC Temporary Error, SC still trying to transfer SM
32	Congestion
33	SME busy
34	No response from SME
35	Service rejected
36	Quality of service not available
37	Error in SME
38-47	Reserved
48-63	Values specific to each SC Permanent Error, SC is not making any more transfer attempts
64	Remote procedure error
65	Incompatible destination
66	Connection rejected by SME
67	Not obtainable
68	Quality of service not available
69	No interworking available
70	SM Validity Period Expired
71	SM Deleted by originating SME
72	SM Deleted by SC Administration
73	SM does not exist (The SM may have previously existed in the SC but the SC no longer had knowledge of it or the SM may never had previously existed in the SC).
74-79	Reserved
80-95	Values specific to each SC Temporary Error
96	Congestion
97	SME busy
98	No response from SME
99	Service rejected
100	Quality of service not available

Value	Description
101	Error in SME
102-105	Reserved
106-111	Reserved
112-127	Values specific to each SC

disableConcatenatedSegmentPad

Syntax: `disableConcatenatedSegmentPad = true|false`

Description: For the MAP adapter only, controls segment padding behavior in a concatenated message.

Type: Boolean

Optionality: Optional (default used if not set).

Allowed:

- true – No non-final message segments in a concatenated message will be padded
  Note: Padding only affects messages which are UCS-2 or GSM 7-bit encoded.
- false – All non-final message segments in a concatenated message, which is UCS-2 or GSM 7-bit encoded, are padded to the maximum message segment length.

Default: false

Example: `disableConcatenatedSegmentPad = false`

doProtocolIdMapping

Syntax: `doProtocolIdMapping = true | false`

Description: Whether or not to apply mapping rules to the TP-PID field.

Optionality: Optional (default used if not set).

Default: false

Allowed: true, false

Note: If true, `defaultProtocolId` and `protocolIdMap` default values apply.

Example: `doProtocolIdMapping = true`

fallbackAlphabet

Syntax: `fallbackAlphabet = "charset"`

Description: The character set to use for outgoing messages when the protocol cannot handle the desired alphabet.

Type: String

Optionality: Optional (default used if not set)

Allowed:

- Binary
- 8859-1
- 8859-2
- 8859-3
- 8859-4
- 8859-5
- 8859-6
- 8859-7
- 8859-8
- 8859-9
- 8859-10
- 8859-11
- 8859-12
- 8859-13
- 8859-14
- 8859-15
- 8859-16
- ASCII7Bit
- 646
- UTF-8
- UCS-2
- GSM7Bit
- GSM8Bit
- GSM7BitComp
- GSMBinaryComp
- GSMUCS2Comp
- JIS
- XKJIS
- ISO-2022-JP
- PCK
- ko_KR-euc
- SMPPPictogram

Default: UCS-2

Notes: When the fallback alphabet is used, a debug entry similar to the following is logged: "unsupported alphabet: ... for map version ... fallback to: *charset*"

Example: `fallbackAlphabet = "GSM8Bit"`

gprsSupport

Syntax: `gprsSupport = "str"`

Description: Whether to ask GPRS questions of the HLR, through Messaging Manager Navigator. Also determines whether inbound messages are allocated to this adapter if separate MAP adapters have been configured for GPRS/non-GPRSmessages.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:
- "unsupported"
- "supported"
- "preferred"

Default: supported

Notes:
- When supported, or preferred, the adapter will include a GPRS supported flag in the Messaging Manager Navigator query. Messaging Manager Navigator may reply with both the VMSC and SGSN numbers. If "preferred", the MAP adapter will then use the SGSN number instead of the VMSC number. If Messaging Manager Navigator replies with only the VMSC number, this will be used instead of SGSN.
- If you are supporting both SMSCs that support GPRS and SMSCs that do not support GPRS, you need to configure two MAP adapters, setting this parameter appropriately.
- Routing to an HLR for GPRS needs the adapters associated with both the incoming and outgoing paths to be GPRS supported.

Example:
```
gprsSupport = "supported"
```

GT

Syntax: `GT = int`

Description: Originating Global Title in outgoing MAP messages.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 5114406267

Notes: Used if no match found in GTMap. Also used when MM overrides the VMSC / SGSN address in an SRI-SM response with its own address.

Example:
```
GT = "5114406267"
```

TT

Syntax: `TT = int`

Description: The translation type used for the corresponding SCCP Calling Party GT.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 0

Notes:
- This applies to both Forward-SMs and SendRoutingInfoForSM messages.
- If the `TT` value is specified in the control plan, from within RIMS IS41 Query and MAP Query nodes, it will override the default value specified in the configuration.

Example:
```
TT = 0
```

GTMap

Syntax: `GTMap = [array]`

Description: Map of incoming originating global title prefix to outgoing originating global title for MO. Also indicates the translation type for SCCP Calling Party GT.

Type: Array

Optionality: Optional (default used if not set).

Allowed:

Default: 0

Notes:

Example:

```
GTMap = [
    { prefix = "0010019198", value = "919827002402", tt = 0 }
]
```

honourReplyPath

If:

- True and the incoming MO ForwardSM has TP-Reply-Path set, allowAlternateDelivery will be set to false.
- False or the incoming MO ForwardSM has TP-Reply-Path not set, allowAlternateDelivery will be set to true

Default: false

Allowed: true, false

hybridiseMapVersions

If:

- True - MAP version 1 MO ForwardSM messages will contain a MAP phase 2 SMS-Submit.
- False they will contain a MAP phase 1 SMS-Submit.

Default: false

Allowed: true, false

lastSegmentDeliveryReceiptOnly

If true, a delivery receipt will only be sent for the last part of a concatenated message. Status report requests on the first parts of a concatenated message will be ignored.

Default: false

Allowed: true, false

localTimeZone

Time zone in which the service center (that is, the adapter) is located. The SM Service-Center-Time-Stamps and Validity-Period are set relative to this time zone.

Default: "UTC"

Allowed: Any zoneinfo database time zone is acceptable.

maxUnsegmentedLength

Length of TP-User-Data (in octets) beyond which messages will be segmented.

Default: 120

mscTimeout

Syntax: `mscTimeout = seconds`

Description: The timer for MSC delivery transactions. It is the period at which an outgoing message (that is, while Messaging Manager waits for a response from the MSC network) will be timed out. The adapter will close the connection to the network and return a "transient failure" to Messaging Manager.

Type: Integer

Optionality: Optional

Allowed:

Default: 5

Example: `mscTimeout = 5`

mscVersionCacheSize

Maximum number of entries that the cache of MSC versions can grow to.

Default: 1000

natureOfAddress

Parameter which configures the destination reference IMSI when we receive a LMSI and an IMSI from the HLR lookup.

Default: 1

numberPlan

Parameter which configures the destination reference IMSI when we receive a LMSI and an IMSI from the HLR lookup.

Default: 6

nonGprsAdapter

If separate MAP adapters have been configured for GPRS/non-GPRS messages, this setting is used in the GPRS adapter to specify the name of the corresponding MAP adapter where GPRS is not supported.

Default: ""

Allowed: String name of an adapter.

Note: Used in the adapter with the configuration of the *gprsSupport* (on page 84) parameter set to "supported", or "preferred".

originatingTimeout

The period at which an incoming MAP message (that is, while Messaging Manager/ACS process the message) will be timed out and a MAP error sent to the MSC.

Default: 10

Allowed: Seconds

pathRetryRandomisation

Syntax: `pathRetryRandomisation = seconds`

Description: The amount of randomization for the delay interval when selecting a delay for a retry.

Allowed: integer

Default: 1

Notes:

Example: `pathRetryRandomisation = 1`

pathRetrySegmentOffset

Syntax: `pathRetrySegmentOffset = seconds`

Description: How much to delay later segments of a concatenated message when a delayed retry is attempted.

Allowed: integer

Notes:

Example: `pathRetrySegmentOffset = 1`

`PC`

Originating Point Code in outgoing MAP messages.

Default: 55

Allowed: integer in range [0, 255]

Note:

`privateExtensions`

Syntax: `privateExtensions = [mapping]`

Description: List of private extension TBCD-STRINGs from the MO-ForwardSM for CPE manipulation.

Type: Array

Optionality: Optional

Allowed:

Default:

Notes:

Example:

```
privateExtensions = [
    {extId="1,2,776,8,8,8,8,11", asn1Tags= [ 0xE0, 0x80],
profileTagID=327814, name = "imei" }
]
```

For full description, including the ASN.1 notation used, see *Example - Extract IMEI into profile tag* (on page 89).

`extId`

Syntax: `extId = "value"`

Description: The ID of the private extension as a comma separated sequence of integers representing an object ID.

Type: String

Optionality: Compulsory if `privateExtensions` is present.

Allowed:

Default:

Notes: Member of the `privateExtensions` array.

Example: `extId="1,2,776,8,8,8,8,11"`

`asn1Tags`

Syntax: `asn1Tags = {tags}`

Description: A list of asn1 tag values giving the location of the TBCD-STRING within the private extension.

Type:

Optionality: Compulsory if `privateExtensions` is present.

Allowed:

Default:

Notes: Member of the `privateExtensions` array.

Example: `asn1Tags = [ 0xE0, 0x80]`

`profileTagID`

Syntax:	`profileTagID = tagid`
Description:	The mapped profile tag ID (in the Temporary Storage profile block)
Type:	Integer
Optionality:	Compulsory if `privateExtensions` is present.
Allowed:	
Default:	
Notes:	327814 is a special profile tag being the one defined in ACS for holding the IMEI. Therefore, xmsTrigger will put the value extracted from the private extension into the InitialDP.iMEI parameter as well as in a temporary storage profile tag. 327814 is the only tag value this rule applies to. Member of the `privateExtensions` array.
Example:	`profileTagID = 327814`

`name`

Syntax:	`name = "name"`
Description:	A string for debug purposes.
Type:	String
Optionality:	Compulsory if `privateExtensions` is present.
Allowed:	
Default:	
Notes:	Member of the `privateExtensions` array.
Example:	`name = "imei"`

Example - Extract IMEI into profile tag

The example configuration below is to extract the IMEI into profile tag 327814 according to the following Abstract Syntax Notation One (ASN.1) notation:

```
mo-ForwardSM OPERATION ::= {
    ARGUMENT SEQUENCE {
        sm-RP-DA CHOICE {
            imsi [0] IMPLICIT OCTET STRING ( SIZE( 3 .. 8 ) ),
            lmsi [1] IMPLICIT OCTET STRING ( SIZE( 4 ) ),
            serviceCentreAddressDA [4] IMPLICIT OCTET STRING ( SIZE( 1 .. 20 ) ),
            noSM-RP-DA [5] IMPLICIT NULL},
        sm-RP-OA CHOICE {
            msisdn [2] IMPLICIT OCTET STRING ( SIZE( 1 .. 20 ) ) ( SIZE( 1 .. 9 ) ),
            serviceCentreAddressOA [4] IMPLICIT OCTET STRING ( SIZE( 1 .. 20 ) ),
        noSM-RP-OA [5] IMPLICIT NULL},
            sm-RP-UI OCTET STRING ( SIZE( 1 .. 200 ) ),
            extensionContainer SEQUENCE {
                privateExtensionsList[0] IMPLICIT SEQUENCE (SIZE (1..10)) OF {
                    SEQUENCE{
                        extId ExtId,
                        extType MOForwardSMType OPTIONAL} OPTIONAL,
                    }
                pcs-Extensions [1] IMPLICIT SEQUENCE {
                ... } OPTIONAL,
            ... } OPTIONAL,
            }
        }
#
```

```
ExtId ::= OBJECT IDENTIFIER {0x01,0x02,0x308,0x08,0x08,0x08,0x08,0x0B}
MOForwardSMType ::= [PRIVATE 0] IMPLICIT SEQUENCE {
    imei [0] IMPLICIT OCTET STRING (SIZE(8)) OPTIONAL,
... }
```

Here is the example configuration.

```
privateExtensions = [
    {extId="1,2,776,8,8,8,8,11", asn1Tags= [ 0xE0, 0x80] , profileTagID=327814, name
    = "imei" }
]
```

protocolIdMap

List of TP-PID translations.

Default: empty

Example: Turns 7f (127, not valid in MAP 2) into 0.

```
protocolIdMap = [
   { in = 127, out = 0 }
]
```

Note: Optional parameter.

relayGlobalTitleType

Syntax: `relayGlobalTitleType = num`

Description: Specifies the global title type to use for the destination global title in relayed messages when no outbound global title is configured for the connection.

Type: Integer

Optionality: Optional (default used if not set)

Allowed:

- -1 – Uses the global title type of the incoming message
- 2 – Uses the global title type of the outgoing message

Default: 2

Notes:

Example: `relayGlobalTitleType = 2`

relayNatureOfAddress

Syntax: `relayNatureOfAddress = num`

Description: Specifies the telephone number type (nature of address) to use for the destination global title in relayed messages when no outbound global title is configured for the connection.

Type: Integer

Optionality: Optional (default used if not set)

Allowed:

- -1 – Uses the nature of address of the incoming message
- 1 – Subscriber number
- 2 – Unknown
- 3 – National
- 4 – International

Default: 4 (International)

Notes:

Example: `relayNatureOfAddress = 3`

relaySSN

Syntax: `relaySSN = num`

Description: Specifies the subsystem number to use in destination addresses in relayed messages.

Type: Integer

Optionality: Optional (default used if not set)

Allowed:
- -1 – Uses the subsystem number of the incoming message
- 0 – Does not set an subsystem number

Default: 0 (no SSN)

Notes:

Example: `relaySSN = -1`

relayTranslationType

Syntax: `relayTranslationType = num`

Description: Specifies the translation type to use for the destination global title in relayed messages when no outbound global title is configured for the connection.

Type: Integer

Optionality: Optional (default used if not set)

Allowed:
- -1 – Uses the translation type from the incoming message
- -2 – Uses the translation type configured for the outgoing message

Default: -2

Notes:

Example: `relayTranslationType = -1`

reportProtocolFailure

Syntax: `reportProtocolFailure = true|false`

Description: Destination GT is copied from the incoming message only for the MAP adapter.

Type: Boolean

Optionality: Optional (default used if not set)

Allowed:
- true – Error will be reported. This will happen in case of protocol error from RIMS.
- false – Destination GT is copied from the incoming message.

Default: false

Notes:

Example: `reportProtocolFailure = false`

rimsTimeout

Syntax: `rimsTimeout = seconds`

Description: The timer for Messaging Manager Navigator queries. It is the period at which an outgoing message (that is, while Messaging Manager waits for a response from Messaging Manager Navigator) will be timed out. The adapter will close the connection to the network and return a "transient failure" to Messaging Manager.

Type: Integer

Optionality: Optional

Allowed:

Default: 5

Example: `rimsTimeout = 5`

rimsInterfaceName

Syntax: `rimsInterfaceName = "name"`
Description: Name of the configured Messaging Manager Navigator interface.
Default: "rimsIf"
Notes: This must match the interface name in **IN/service_packages/SLEE/etc/SLEE.cfg**
Example: `rimsInterfaceName = "rimsIf"`

SCA

Originating service center address in outgoing MAP messages.

Default: none
Allowed: string
Note: Defaults to GT if not set or empty

scheduledDeliveryTime

Syntax: `scheduledDeliveryTime = "time"`
Description: The scheduled delivery time to set on outgoing submits that have previously failed at FDA.
Type: String
Optionality: Optional (default used if not set).
Allowed: Format is as per the SMPP spec.
Default: Null
Notes:
Example: `scheduledDeliveryTime = "000000001000000R"`
Configures a 10 minute delay.

sgsnPrefixes

Number rules to allow the adapter to distinguish between VMSC and SGSN for originating RIMS updates.

Default: "000"

smscTimeout

Syntax: `smscTimeout = seconds`
Description: The timer for MC deliveries. It is the period at which an outgoing message (that is, while Messaging Manager waits for a response from the SMSC network) will be timed out. The adapter will close the connection to the network and return a "transient failure" to Messaging Manager.
Type: Integer
Optionality: Optional
Allowed:
Default: 5
Example: `smscTimeout = 5`

splitLongMessages

Syntax: `splitLongMessages = true|false`

Description: If the user data is too long to fit into a single MAP SMS, this option will cause it to be split into multiple outgoing concatenated messages when set to true.

Type: Boolean

Optionality: Optional

Allowed: true, false

Default: true

Notes:

Example: `splitLongMessages = true`

SSN

Originating subsystem number in outgoing MAP messages.

Default: 8

Allowed: Valid sub system number

suppressRimsMap3imsiOAUpdate

Syntax: `suppressRimsMap3imsiOAUpdate = true|false`

Description: Determines whether to suppress imsiOA MAP version 3 update to the Rims cache on receiving a MO-ForwardSM message request.

Type: Boolean

Optionality: Optional (default used if not set)

Allowed: true, false

Default: false

Notes:

Example: `suppressRimsMap3imsiOAUpdate = true`

tcapInterfaceServiceKey

SLEE service key of the TCAP interface to use for sending outbound messages.

Default: Must match the TCAP interface's service key as specified in the **SLEE.cfg** file.

Note: Required parameter.

throttledDeliveryFailureCause

Syntax: `throttledDeliveryFailureCause = cause`

Description: MAP SM delivery failure cause to set when message is throttled.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

- SMFmemoryCapacityExceeded = 0
- SMFequipmentProtocolError = 1
- SMFequipmentNotSM_Equipped = 2
- SMFunknownServiceCentre = 3
- SMFsc_Congestion = 4
- SMFinvalidSME_Address = 5
- SMFsubscriberNotSC_Subscriber = 6

Default: 4

Notes:

Example: `throttledDeliveryFailureCause = 0`

Receiving a MAP sm-DeliveryFailure error message

When the adapter receives a MAP error message from a MSC (in direct delivery scenarios) or from a SMSC, the adapter translates the error to a Messaging Manager transient or permanent failure based on the mapping configured in the forwardSmErrorMap parameter in **eserv.config**. If the MAP error code is not specified in the forwardSmErrorMap parameter, the default is permanent failure, unless the MAP error code is 32 (sm-DeliveryFailure).

In that case the adapter uses the hard-coded mapping shown below to convert from the FailureCause parameter contained within the sm-DeliveryFailure error message to transient or permanent failure.

sm-DeliveryFailureCause	MM Result Code
0 (memory capacity exceeded)	transient failure
1 (equipment protocol error)	permanent failure
2 (equipment not SM-equipped)	permanent failure
3 (unknown service centre)	permanent failure
4 (SC congestion)	transient failure
5 (invalid SME address)	permanent failure
6 (subscriber not SC subscriber)	permanent failure

Sending a MAP sm-DeliveryFailure error message

Determining the MAP error code

When the adapter receives an Messaging Manager transient or permanent failure, the adapter translates the Messaging Manager failure message to a MAP error message. If the Messaging Manager failure message specifies the release cause (as a parameter in the Messaging Manager error message), the adapter uses the mapping to MAP error configured in the `releaseCauseMap` parameter in the **eserv.config** file. If the Messaging Manager failure message does not specify a release cause, or no mapping is configured in the `releaseCauseMap` parameter, the adapter uses the MAP error configured in the `defaultTransientFailureErrorCode` parameter or the `defaultPermanentFailureErrorCode` parameter in the **eserv.config** file.

If the MAP error code is 32

If the MAP error code determined by the above procedure is 32 (sm-DeliveryFailure) then the error message's sm-DeliveryFailureCause parameter is set to "SC congestion" for Messaging Manager transient failures, or to "subscriber not SC subscriber" for Messaging Manager permanent failures.

MM Result Code	sm-DeliveryFailureCause
transient failure	SC congestion
permanent failure	subscriber not SC subscriber

Example MAP config

In this example, SRI-SMs will be received with SSN 8 and a PC other than 1001, 1002 or 1003. This means that the message will match both MAP1 and MAP2 adapters, so one will be picked based on GPRS support. If the SRI-SM specifies that GPRS is supported, it will be passed to the MAP1 adapter, otherwise it will be passed to the MAP2 adapter.

MM will send an SRI-SM query to the HLR, and replace the address it gets in the response with its own addresses. If the HLR returns a VMSC address, MM will replace it with the GT of the MAP2 adapter (5114406268). If the HLR returns an SGSN address, MM will replace it with the GT of the MAP1 adapter (5114406267).

The calling SMSC will then send an MT-ForwardSM to one or other of these GT addresses. The network should be configured to translate this to the appropriate point code so that the MT-ForwardSM goes to the right adapter. So, for this example:

GT	PC
5114406267	1001 or 1002
5114406268	1003

Example config:

```
adapters = [

    # First adapter (MAP)
    {
        lib = "xmsiMap.so"
        adapterName = "MAP1"

        pointCodes = [1001, 1002]
        SSN = 8

        earlyAckMC = true
        earlyAckSME = true

        allowConcatenatedFDA = true
        alwaysProduceNonDeliveryReceipt = false

        config = {

            abortMessagesWithZeroLengthTPDA = true
            allowIncoming = true
            allowOutgoing = true
            allowDirectDelivery = true

            allowIncomingMap3 = true

            allowUserRequestedDeliveryReceipt = true

            lastSegmentDeliveryReceiptOnly = false

            tcapInterfaceServiceKey = 22

            originatingTimeout = 10

            smscTimeout = 8  # Timer for MC deliveries
            rimsTimeout = 3  # Timer for RIMS queries
            mscTimeout = 15  # Timer for MSC delivery transactions

            rimsInterfaceName = "rimsIf"

            gprsSupport = "supported"

            nonGprsAdapter = "MAP2"

            sgsnPrefixes = [ "000" ]

            doProtocolIdMapping = false
            defaultProtocolId = 0
```

```
protocolIdMap = [
    { in = 127, out = 0 }
]

fallbackAlphabet = "UCS-2"
scheduledDeliveryTime = ""

defaultMessagePriority = "Normal"

PC = 55
SSN = 8

GT = "5114406267"
TT = 0

GTMap = [
    { prefix = "0010019198", value = "919827002402", tt = 0 }
]

#SCA = "5114406267"

natureOfAddress = 1 # international
numberPlan = 6      # E.212

defaultMapVersionSmsc = 3
defaultMapVersionMsc = 3


defaultDeliveryFailureStatusCode = 64

throttledDeliveryFailureCause = 96 # congestion

mscVersionCacheSize = 1000

hybridiseMapVersions = false

maxUnsegmentedLength = 120

splitLongMessages = true

alarmMask = 0

honourReplyPath = false

pathRetryRandomisation = 1
pathRetrySegmentOffset = 1

localTimeZone = "UTC"

forwardSmErrorMap = [
    { error = 9, permanent = true }
    { error = 32, permanent = false }
]

defaultTransientFailureErrorCode = 32
defaultPermanentFailureErrorCode = 32

incomingOriginatingNumberRules = [
    { fromNoa=2, prefix="04", min=4, max=32, remove=1,
    prepend="0064", resultNoa=1 }
    { fromNoa=3, prefix="4", min=4, max=32, remove=0, prepend="0064",
    resultNoa=1 }
]
incomingDestinationNumberRules = [
```

```
            ]
            # outgoingOriginatingNumberRules = [ ]
            # outgoingDestinationNumberRules = [ ]

        }
    } # xmsiMap.so config

    # Non-GPRS MAP adapter
    {
        lib = "xmsiMap.so"

        adapterName = "MAP2"

        pointCodes = [1003]

        SSN = 8

        config = {

            GT = "5114406268"
            TT = 0

            gprsSupport = "unsupported"

        }

    } # xmsiMap.so config
```

Configuring the EMI Adapter

EMI adapter overview

The EMI adapter communicates between the Messaging Manager platform and any configured ASPs and SMSCs using EMI. In a system using EMI, the adapter must be configured to know of all entities that may connect to it and all entities that it may connect to.

General EMI configuration

The following section of the **eserv.config** specifies the general parameters for all EMI connections.

```
adapters =
    # second adapter (EMI)
    {
        lib = "mmxiEMI.so"

        SSN = 0

        adapterName = "EMI1"

        earlyAckMC = false
        earlyAckSME = true
        allowConcatenatedFDA = true

        config = {
            global_parameters
        }
    }
]
```

General EMI parameters

Here are the general parameters for EMI.

`adapterName`

Syntax:	`adapterName = "`*`name`*`"`
Description:	Identifier for the adapter.
Allowed	Any text string, but should be meaningful, for example include the protocol used. For example: • "MAP1" for MAP • "EMI1" for EMI • "SCA1" for SCA • "SMPP1" for SMPP • "CDMA1" for IS-41 CDMA • "TDMA1" for IS-41 TDMA • "Wrapper" for Wrapper
Default:	No default.
Notes:	This name *must* also be in the configuration database before Messaging Manager will run correctly. See *MM User's Guide*.

`config`

The parameters in this sub-section below this give the configuration for all messages for this adapter.

`lib`

Syntax:	`lib = "`*`library`*`"`
Description:	The name of the library that contains the EMI adapter being configured.
Allowed	"mmxiEMI.so"
Default:	No default.
Example:	`lib = "mmxiEMI.so"`

`SSN`

Syntax:	`SSN =` *`num`*
Description:	Destination subsystem number of messages to be handled by this adapter.
Allowed	Valid subsystem number
Notes:	Non-zero to handle incoming TCAP.
Example:	`SSN = 18`

Global EMI configuration

The following section of the **eserv.config** specifies the global parameters for all EMI connections.

```
config = {

    suppressPathInfoReport = true
    displayZeroPathReport = false
    PathReportingInterval = 60
    throttledErrorCode = 4
    transientFailureErrorCode = 4
    permanentFailureErrorCode = 3

    incomingOriginatingNumberRules = [
```

```
            { fromNoa=999, prefix="", min=1, max=32, remove=0, resultNoa=2 }
        ]
        incomingDestinationNumberRules = [
        ]
        # outgoingOriginatingNumberRules = [ ]
        # outgoingDestinationNumberRules = [ ]

        emiDefaults = {
            defaultMessagePriority = "Normal"

            pstoreNumberRules = [
                {fromNoa=999, prefix="010", min=1, max=32, remove=3,
                prepend="7" }
                { fromNoa=999, prefix="091", min=1, max=32, remove=3,
                prepend="7" }
            ]

            timestampAdvance = true
            timestampBucketSize = 5000
            timestampFlush = 2
        } # emiDefaults
    } # mmxiEMI.so config
```

Global EMI parameters

Here are the parameters for configuring the global EMI adapter.

EMI error codes

Here is a list of EMI error codes for the following:

- throttledErrorCode,
- transientFailureErrorCode
- permanentFailureErrorCode

Value	Description
1	Checksum error
2	Syntax error
3	Operation not supported by system
4	Operation not allowed
5	Call barring active
6	AdC invalid
7	Authentication failure
8	Legitimization code for all calls, failure
9	GA not valid
10	Repetition not allowed
11	Legitimization code for repetition, failure
12	Priority call not allowed
13	Legitimization code for priority call, failure
14	Urgent message not allowed
15	Legitimization code for urgent message, failure
16	Reverse charging not allowed

Value	Description
17	Legitimization code for rev. charging, failure
18	Deferred delivery not allowed
19	New AC not valid
20	New legitimization code not valid
21	Standard text not valid
22	Time period not valid
23	Message type not supported by system
24	Message too long
25	Requested standard text not valid
26	Message type not valid for the pager type
27	Message not found in SMSC
30	Subscriber hang-up
31	Fax group not supported
32	Fax message type not supported
33	Address already in list (60 series)
34	Address not in list (60 series)
35	List full, cannot add address to list (60 series)
36	RPID already in use
37	Delivery in progress
38	Message forwarded

`emiDefaults`

The `emiDefaults` sub-section contains the default EMI configurations.

`permanentFailureErrorCode`

Syntax: `permanentFailureErrorCode = code`
Description: The error code to return for permanent failures.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: Refer to *EMI error codes* (on page 99) for list of values.
Default: 3
Notes:
Example: `permanentFailureErrorCode = 3`

`suppressPathInfoReport`

Syntax: `suppressPathInfoReport = true|false`
Description: Do we want path connection reports to be suppressed?
Type: Boolean
Optionality:
Allowed: true, false
Default: true

Notes:	An ASP receives heartbeats from the SMSC when Messaging Manager is configured to operate as an SMSC, then it will respond to these heartbeats. These are logged in the xmsTrigger logfile. This can cause the logfile to fill up unnecessarily. If this parameter is set to true the heartbeat message being written to the xmsTrigger log is suppressed.
Example:	`suppressPathInfoReport = true`

`throttledErrorCode`

Error code to return to throttled messages.

Default:	4
Allowed:	Refer to *EMI error codes* (on page 99) for list of values.

`transientFailureErrorCode`

Error code to return for transient failures.

Default:	4
Allowed:	Refer to *EMI error codes* (on page 99) for list of values.

EMI Defaults configuration

The `emiDefaults` section of the **eserv.config** specifies the EMI values that will be used for connections.

```
emiDefaults = {
    defaultMessagePriority = "Normal"

    pstoreNumberRules = [
        {fromNoa=999, prefix="010", min=1, max=32, remove=3,
        prepend="7" }
        { fromNoa=999, prefix="091", min=1, max=32, remove=3,
        prepend="7" }
    ]

    timestampAdvance = true
    timestampBucketSize = 5000
    timestampFlush = 2
}
```

EMI Defaults parameters

Here are the parameters for configuring the EMI defaults in the EMI adapter.

`defaultMessagePriority`

Syntax:	`defaultMessagePriority = "`*`str`*`"`
Description:	The priority to be applied to incoming messages that do not have a priority.
Type:	String
Optionality:	Optional (default used if parameter absent).
Allowed	
Default	"Normal"
Notes:	
Example:	`defaultMessagePriority = "Normal"`

`pstoreNumberRules`

Defines the number normalization rules to be applied for SCTS mapping.

Note: Optional

See *Number Normalization* (on page 72).

Example:

```
pstoreNumberRules =  [
    # number insertion rules
    {fromNoa=999, prefix="010", min=1, max=32, remove=3, prepend="7" }
    # number lookup rules
    { fromNoa=999, prefix="091", min=1, max=32, remove=3, prepend="7" }
]
```

`timestampAdvance`

Values for configuring the EMI System Message (SM) field Emulate SMSC behavior by advancing the timestamp.

- true - emulate SMSC timestamp advance
- false - timestamp using current time

Default: true

Allowed: true, false

Note: When on (true) the IP adapter simulates the behavior of the SMSC by advancing the timestamp if another message with the same destination number is found in the message buffer.

When off (false) the IP adapter uses the current system time and the SM-field will not be unique for messages to the same destination received within the same 1 second timeframe.

`timestampBucketSize`

Sets the size of the hash map used for searching the message buffer.

Default: 5000

Note:

- The size of the message buffer is determined by the incoming message rate and the flush period - NOT by this parameter.
- This parameter can only be set at startup.

`timestampFlush`

Period between buffer cleanup.

Default: 2

Allowed: Seconds

Note: Setting this parameter to a large value will increase memory use. The total number of messages in the buffer is determined by the incoming message rate, the flush period and the number of messages held in the buffer because their timestamp was advanced.

Advancing the Timestamp

This functionality is used where EMI messages need to emulate SMSC behavior in advancing the timestamp. This is done to ensure that the SM-field is unique for messages to the same destination within the same 1 second time frame.

The functionality to advance the timestamp is implemented in the following way:

A list is kept that maps the B-number to the last timestamp sent. There is only one entry for a B-number and it contains the last timestamp sent. When the list is flushed, the current system time is compared with the last timestamp in the list. If this is older than the current system time, the entry is removed.

For speed of lookup a hash map is used that contains the B-number and a pointer to the entry in the list. The size of the map is determined (by the `timestampBucketSize` parameter. This is fixed at startup and not reread at SIGHUP.

The following parameters influence this:

- `timestampBucketSize` - For fast lookup, this value should be big. The larger the number the faster the lookup, but requires more memory. A small value gives slow lookup but requires less memory.
- `timestampFlush` - This value determines how often the linked list is flushed and therefore also the size of the list. If the list is allowed to grow huge, then the flush will take longer since the whole list is traversed. The actual number of entries in the list is determined by the incoming call rate, the value of this parameter and the number of entries whose timestamp has been advanced.

Configuring the SMPP Adapter

SMPP adapter overview

The SMPP adapter communicates between the Messaging Manager platform and any configured ASPs and SMSCs using SMPP. In a system using SMPP, the adapter must be configured to know of all entities that may connect to it and all entities that it may connect to.

General SMPP configuration

The following section of the **eserv.config** specifies the general parameters for all SMPP connections.

```
adapters = [

    # Third adapter (SMPP)
    {
        lib = "mmxiSMPP.so"

        #pointCodes = [1003, 1004]

        SSN = 0

        adapterName = "SMPP1"

        earlyAckMC = false
        earlyAckSME = true
        allowConcatenatedFDA = true

        config = {
            global_parameters
        }
    }
]
```

General SMPP parameters

Here are the general parameters for SMPP.

`adapterName`

Syntax:	`adapterName = "`*`name`*`"`
Description:	Identifier for the adapter.

Allowed	Any text string, but should be meaningful, for example include the protocol used. For example: • "MAP1" for MAP • "EMI1" for EMI • "SCA1" for SCA • "SMPP1" for SMPP • "CDMA1" for IS-41 CDMA • "TDMA1" for IS-41 TDMA • "Wrapper" for Wrapper
Default:	No default.
Notes:	This name *must* also be in the configuration database before Messaging Manager will run correctly. See *MM User's Guide*.

`config`

The parameters in this sub-section below this give the configuration for all messages for this adapter.

`lib`

Syntax:	`lib = "`*`library`*`"`
Description:	The name of the library that contains the SMPP adapter being configured.
Allowed	"mmxiSMPP.so"
Default:	No default.
Example:	`lib = "mmxiSMPP.so"`

`pointCodes`

Syntax:	`pointCodes = [`*`pc1`*`, `*`pc2`*`, ...]`
Description:	Destination point codes array of messages to be handled by this adapter. This parameter takes priority over SSN match.
Optionality:	Optional
Allowed	Defined by network administrator.
Example:	`pointCodes = [1001, 1002]`

`SSN`

Syntax:	`SSN = `*`num`*
Description:	Destination subsystem number of messages to be handled by this adapter.
Allowed	Valid subsystem number
Notes:	Non-zero to handle incoming TCAP.
Example:	`SSN = 18`

Global SMPP configuration

The following section of the **eserv.config** specifies the global parameters for all SMPP connections.

```
config = {

    suppressPathInfoReport = true
    displayZeroPathReport = false
    PathReportingInterval = 60
    fallbackAlphabet = "UCS-2"
    maxSmppShortMessageSize = 160
    smppDefaults = {
```

```
            throttledCommandStatus = 88 # ESME_RTHROTTLED (88)

            teleserviceRoutingMap = [
                { serviceType =    "", teleservice =    0,
                allowAlternateDelivery = true }
                { serviceType = "CMT", teleservice = 4098,
                allowAlternateDelivery = true }
                { serviceType = "EMS", teleservice = 4101,
                allowAlternateDelivery = true }
            ]

            dataCodingMap = [
                { data_coding = 0x08,
                    alphabet = "UCS-2",
                    messageClass = 1,
                    messageWaitingGroup = 0,
                    messageWaitingIndicator = 0,
                    messageWaitingType = 0 }
                { data_coding = 0xF0,
                    alphabet = "GSM8Bit",
                    byteAlign = true,
                    messageClass = 1,
                    messageWaitingGroup = 0,
                    messageWaitingIndicator = 0,
                    messageWaitingType = 0 }
            ]

            includePayloadDRInfo = false]

            maxValidityPeriod = 0

        } # smppDefaults

        outgoingOriginatingNumberRules = [
            { fromNoa=2, prefix="04", min=4, max=32, remove=1,
            prepend="0064", resultNoa=1 }
            { fromNoa=3, prefix="4", min=4, max=32, remove=0, prepend="0064",
            resultNoa=1 }
        ]
        outgoingDestinationNumberRules = [
        ]
        # incomingOriginatingNumberRules = [ ]
        # incomingDestinationNumberRules = [ ]

    }
```

Global SMPP parameters

Here are the parameters for configuring the global SMPP adapter.

`maxSmppShortMessageSize`

Syntax:	`maxSmppShortMessageSize = nnn` where nnn is an integer.
Description:	The maximum size of the short message. Message is sent in short message field when it is less than or equal to `maxSmppShortMessageSize` value, else it will go for tlv message payload.
Type:	Integer
Optionality:	Optional
Allowed:	

Default: 255

Notes:

Example: `maxSmppShortMessageSize = 160`

suppressPathInfoReport

Syntax: `suppressPathInfoReport = true|false`

Description: Do we want path connection reports to be suppressed?

Type: Boolean

Optionality:

Allowed: true, false

Default: true

Notes: An ASP receives heartbeats from the SMSC when Messaging Manager is configured to operate as an SMSC, then it will respond to these heartbeats. These are logged in the xmsTrigger logfile. This can cause the logfile to fill up unnecessarily.

If this parameter is set to true the heartbeat message being written to the xmsTrigger log is suppressed.

Example: `suppressPathInfoReport = true`

TLVs

Syntax: `TLVs = [tag_maps]`

Description: List of TLV tags to map to a profile buffer.

Inbound:

- The SMPP adapter checks all inbound messages for the TLVs mentioned in TLVs section, and if present, passes them to slee_acs, which then populates the configured profile buffer in the temporary profile block for each TLV that was present.

Outbound:

- The CPE can use the copy/set nodes to populate the configured profile tag IDs with data desired for the outbound TLVs and pass them to the SMPP adapter.
- The SMPP adapter uses this config to populate outbound TLVs if any of the configured fields are present.

Type: Array of TLV elements.

Optionality: Optional

Allowed:

Default:

Notes: Please ensure tlvIDs and profileTagIDs are unique within each configuration.

Example:

```
TLVs = [
    {tlvID=0x0030, tlvType=0x01, profileTagID=4532781,
    direction="inbound"}
    {tlvID=0x020b, tlvType=0x02, profileTagID=4532782,
    direction="outbound"}
    {tlvID=0x1382, tlvType=0x05, profileTagID=4532783,
    direction="both"}
 ]
```

TLVs parameters

`tlvID`

Syntax: `tlvID=<tag ID>`
Description: The TLV tag ID
Type: Integer, 2 bytes long. Can be in hexadecimal or decimal.
Optionality: Mandatory
Allowed:
Default:
Notes:
Example: `tlvID=0x0030`

`tlvType`

Syntax: `tlvType=<type>`
Description: The type of TLV value.
Type: Integer, 1 byte long, may be hexadecimal, or decimal.
Optionality: Mandatory
Allowed: Five types are supported. The mappings are:

- 0x01 -> INT8
- 0x02 -> INT16
- 0x03 -> INT32
- 0x04 -> CString
- 0x05 -> OctetString

Default:
Notes:
Example: `tlvType=0x01`

`profileTagID`

Syntax: `profileTagID=<profiletag>`
Description: The profile tag ID which the TLV maps to.
Type: Integer, 4 bytes long, can be in hexadecimal or decimal.
Optionality: Mandatory
Allowed:
Default:
Notes:
Example: `profileTagID=453281`

`direction`

Syntax: `direction="<direction>"`
Description: What type of SMPP message this TLV presents in.
Type: String
Optionality: Mandatory

Allowed: Three options are supported:
- inbound,
- outbound, or
- both.

Default:

Notes:

Example: `direction="inbound"`

SMPP Defaults configuration

The `smppDefaults` section of the **eserv.config** specifies the SMPP values that will be used for connections.

```
smppDefaults = {

    throttledCommandStatus = 88 # ESME_RTHROTTLED (88)

    teleserviceRoutingMap = [
        { serviceType =    "", teleservice =    0,
        allowAlternateDelivery = true }
        { serviceType = "CMT", teleservice = 4098,
        allowAlternateDelivery = true }
        { serviceType = "EMS", teleservice = 4101,
        allowAlternateDelivery = true }
    ]

    dataCodingMap = [
        { data_coding = 0x08,
            alphabet = "UCS-2",
            messageClass = 1,
            messageWaitingGroup = 0,
            messageWaitingIndicator = 0,
            messageWaitingType = 0 }
        { data_coding = 0xF0,
            alphabet = "GSM8Bit",
            byteAlign = true,
            messageClass = 1,
            messageWaitingGroup = 0,
            messageWaitingIndicator = 0,
            messageWaitingType = 0 }
    ]

    includePayloadDRInfo = false

    maxValidityPeriod = 0

    scheduledDeliveryTime = ""

    convertMessageIdToHex = false

    drTLVReceiptedMsgIdUsesHex = false

    fixedLengthMessageId = true

    tlvWarnSuppress = false

    sendUserMessageReferenceInConcatenatedMessages = true

}
```

SMPP Defaults parameters

Here are the parameters for configuring the SMPP defaults in the SMPP adapter.

`alphabet`

Syntax:	`alphabet = "`*`charset`*`"`
Description:	Character set of the desired alphabet.
Type:	String
Allowed:	Binary 8859-1 8859-2 8859-3 8859-4 8859-5 8859-6 8859-7 8859-8 8859-9 8859-10 8859-11 8859-12 8859-13 8859-14 8859-15 8859-16 ASCII7Bit 646 UTF-8 UCS-2 GSM7Bit GSM8Bit GSM7BitComp GSMBinaryComp GSMUCS2Comp JIS XKJIS ISO-2022-JP PCK ko_KR-euc SMPPPictogram
Example:	`alphabet = "UCS-2"`

byteAlign

Syntax: `byteAlign = true|false`

Description: Specifies whether data is byte aligned.

`byteAlign` is used in conjunction with the `alphabet` parameter when `alphabet` is set to **GSM7Bit** or **GSM8Bit**.

Type: Boolean

Optionality: Optional (default used if not set)

Allowed: true – Data is byte aligned.

false – Data is not byte aligned.

Default: When the `alphabet` parameter is set to **GSM8Bit**, the default is **true**.

When the `alphabet` parameter is to anything else, the default is **false**.

Notes:

Example:
```
alphabet = "GSM8Bit",
byteAlign = true
```

convertMessageIdToHex

Syntax: `convertMessageIdToHex = true|false`

Description: Global option to work around a potentially troublesome feature in some SMSCs, where they return hex message ids in their submit_sm responses, but decimal message ids in the payload parts of their delivery receipts.

Type: Boolean

Optionality: Optional (default used if not set).

Allowed:

true	
false	no automatic adjustment of message IDs between decimal and hex

Default: false

Notes: Without this option, the behavior interferes with the MM pstore mapping of message IDs.

If this is set to true, then the `deliveryReceiptId (on page 46)` parameter must be hex value only, not ASCII.

Example:
```
convertMessageIdToHex = true
```

dataCodingMap

Syntax: `dataCodingMap = [datacoding_parameters]`

Description: Data Coding Map allows you to modify the data coding of SMPP short messages to have a different interpretation of Message Class, Waiting Group and character set to the predefined values. If set here, it will override any associated predefined DCS value that Messaging Manager starts up with.

Optionality: Optional

Notes: For inbound messages, MM will select the data coding number (for example, 0x08) value and match the parameters for that data coding.

For outbound messages, MM will match the parameters, then uses it to calculate the data coding. If there is more than one set of parameters with all values the same, MM will select the last data coding.

Example:

```
dataCodingMap = [
    { data_coding = 0x08,
        alphabet = "UCS-2",
        messageClass = 1,
        messageWaitingGroup = 0,
        messageWaitingIndicator = 0,
        messageWaitingType = 0 }
    { data_coding = 0xF0,
        alphabet = "GSM8Bit",
        byteAlign = true,
        messageClass = 1,
        messageWaitingGroup = 0,
        messageWaitingIndicator = 0,
        messageWaitingType = 0 }
]
```

In this example, for outbound messages, if the alphabet for both data codings for 8 and F0 were the same, then MM would select F0.

data_coding

Syntax: `data_coding = coding`

Description: The MAP DCS data coding.

Allowed Hex value of number up to 255

Example: `data_coding =` 0xF0

drTLVReciptedMsgIdUsesHex

Syntax: `drTLVReceiptedMsgIdUsesHex = true|false`

Description: Determines whether to treat the value that is returned in an SMSC delivery receipt in the optional `TLV receipted_msg_ids` field as a hexadecimal value.

Type: Boolean

Optionality: Optional (default used if not set)

Allowed: true – Treat TLV `receipted_msg_ids` values as hex values

false – Treat TLV `receipted_msg_ids` values as decimal values

Default: false

Notes: You must use the `convertMessageIdToHex` (on page 110) parameter in conjunction with the `drTLVReceiptedMsgIdUsesHex` parameter.

Example: `drTLVReceiptedMsgIdUsesHex = true`

fallbackAlphabet

Syntax: `fallbackAlphabet = "charset"`

Description: The character set to use for outgoing messages when the protocol cannot handle the desired alphabet.

Type: String

Optionality: Optional (default used if not set)

Allowed:
- Binary
- 8859-1
- 8859-2
- 8859-3
- 8859-4
- 8859-5
- 8859-6
- 8859-7
- 8859-8
- 8859-9
- 8859-10
- 8859-11
- 8859-12
- 8859-13
- 8859-14
- 8859-15
- 8859-16
- ASCII7Bit
- 646
- UTF-8
- UCS-2
- GSM7Bit
- GSM8Bit
- GSM7BitComp
- GSMBinaryComp
- GSMUCS2Comp
- JIS
- XKJIS
- ISO-2022-JP
- PCK
- ko_KR-euc
- SMPPPictogram

Default: UCS-2

Notes: When the fallback alphabet is used, a debug entry similar to the following is logged: "unsupported alphabet: ... for map version ... fallback to: *charset*"

Example: `fallbackAlphabet = "GSM8Bit"`

`fixedLengthMessageId`

Syntax: `fixedLengthMessageId = true|false`

Description: An SMPP option to allow xmsTrigger to perform padding of message IDs so they are always 10 digits long.

Type: Boolean

Optionality: Optional (default used if not set).

Allowed: true, false

Default: true

Notes: When set to true, MM performs automatic padding of the hex message IDs so it is always 10 hex digits long before writing the value to pstore. This helps maintain consistency in the message ID format stored in pstore and may help with improving pstore matching for message IDs shorter than 10 digits.

This option must be used in conjunction with `convertMessageIdToHex` (on page 110).

Only message IDs shorter than 10 digits are affected. Message IDs longer than 10 digits are not padded regardless of the setting of this field.

Example: `fixedLengthMessageId = true`

includePayloadDRInfo

Syntax: `includePayloadDRInfo = true|false`

Description: If set, forces Messaging Manager to build a delivery receipt message payload containing the extra information in Appendix B of the SMPP 3.4 specification.

Whether set or not, the standard TLVs message_state and receipted_message_id will be used.

Allowed true, false

Default: false

Notes: This option only applies to Messaging Manager-generated delivery receipts. If a delivery receipt is received from an SMSC, Messaging Manager will send the SME one in the same format.

Example: `includePayloadDRInfo = false`

maxValidityPeriod

Syntax: `maxValidityPeriod = seconds`

Description: If set, and non-zero, defines the maximum possible validity period for outbound messages from the adapter.

Type: Integer

Optionality: Optional

Allowed:

Default: 0

Notes: Validity periods that would have been greater are shortened to this value, when it is set.

Example: `maxValidityPeriod = 0`

messageClass

Syntax: `messageClass = class`

Description: The MAP DCS message class.

Allowed 0 - none
1 - Display (GSM class 0 - also known as Flash)
2 - Mobile Equipment (GSM class 1)
3 - SIM (GSM class 2)
4 - External (GSM class 3)

Default: 0

Notes: Refer to GSM 03.38 version 7.2.0 for the full specification.

Example: `messageClass = 1`

`messageWaitingGroup`

Syntax:	`messageWaitingGroup = group`
Description:	The MAP DCS message waiting group.
Allowed	0 - none/not specified 1 - Discard 2 - Store
Default:	0
Notes:	Refer to GSM 03.38 version 7.2.0 for the full specification.
Example:	`messageWaitingGroup = 0`

`messageWaitingIndicator`

Syntax:	`messageWaitingIndicator = ind`
Description:	The MAP DCS message waiting indicator.
Allowed	0 - Inactive - none or not specified 1 - Active
Default:	0
Notes:	Refer to GSM 03.38 version 7.2.0 for the full specification.
Example:	`messageWaitingIndicator = 0`

`messageWaitingType`

Syntax:	`messageWaitingType = type`
Description:	The MAP DCS message waiting type.
Allowed	0 - none or not specified 1 - Voicemail 2 - Fax 3 - Email 4 - Other
Default:	0
Notes:	Refer to GSM 03.38 version 7.2.0 for the full specification.
Example:	`messageWaitingType = 0`

`sendUserMessageReferenceInConcatenatedMessages`

Syntax:	`sendUserMessageReferenceInConcatenatedMessages = true|false`
Description:	If set true (default), the optional parameter user_message_reference will be sent irrespective of the presence of the UDH concatenation properties. If set false, the optional parameter user_message_reference will not be sent if the UDH concatenation properties are populated.
Type:	Boolean
Optionality:	
Allowed:	true, false
Default:	true
Notes:	
Example:	`sendUserMessageReferenceInConcatenatedMessages = true`

SMPP command statuses

Here is a list of SMPP command statuses for use in:

- throttledCommandStatus
- acsReleaseCauseToSmppCommandStatus

Status	Description
0x0000	No error
0x0001	Message length is invalid
0x0002	Command length is invalid
0x0003	Invalid command ID
0x0004	Incorrect BIND status for given command
0x0005	ESME already in bound state
0x0006	Invalid priority flag
0x0007	Invalid registered delivery flag
0x0008	System error
0x000A	Invalid source address
0x000B	Invalid destination address
0x000C	Message ID is invalid
0x000D	Bind failed
0x000E	Invalid password
0x000F	Invalid system ID
0x0011	Cancel SM failed
0x0013	Replace SM failed
0x0014	Message queue full
0x0015	Invalid service type
0x0033	Invalid number of destinations
0x0034	Invalid distribution list name
x0040	Destination flag is invalid (submit_multi)
x0042	Submit w/replace functionality has been requested where it is either unsupported or inappropriate for the particular MC
0x0043	Invalid esm_class field data
0x0044	Cannot submit to distribution list
0x0045	submit_sm, data_sm or submit_multi failed
0x0048	Invalid source address TON
0x0049	Invalid source address NPI
0x0050	Invalid destination address TON
0x0051	Invalid destination address NPI
0x0053	Invalid system_type field
0x0054	Invalid replace_if_present flag
0x0055	Invalid number of messages
0x0058	Throttling error (ESME has exceeded allowed message limits)
0x0061	Invalid scheduled delivery time
0x0062	Invalid message validity period (Expiry time)
x0063	Predefined message ID is Invalid or specified predefined message was not found

Status	Description
x0064	ESME receiver temporary app error code
x0065	ESME receiver permanent app error code
x0066	ESME receiver reject message error code
x0067	query_sm request failed
x00C0	Error in the optional part of the PDU body
x00C1	TLV not allowed
x00C2	Invalid parameter length
x00C3	Expected TLV missing
x00C4	Invalid TLV value
x00FE	Transaction delivery failure
x00FF	Unknown error
	MPP v5.0 Only:
x0100	ESME not authorized to use specified service_type
x0101	ESME prohibited from using specified operation
x0102	Specified service_type is unavailable
x0103	Specified service_type is denied
x0104	Invalid data coding scheme
x0105	Source address subunit is invalid
x0106	Destination address subunit is invalid
x0107	Broadcast frequency interval is invalid
x0108	Broadcast alias name is invalid
x0109	Broadcast area format is invalid
x010A	Number of broadcast areas is invalid
x010B	Broadcast content type is invalid
x010C	Broadcast message class is invalid
x010D	broadcast_sm operation failed
x010E	query_broadcast_sm operation failed
x010F	cancel_broadcast_sm operation failed
x0110	Number of repeated broadcasts is invalid
x0111	Broadcast service group is invalid
x0112	Broadcast channel indicator is invalid

`scheduledDeliveryTime`

Syntax: `scheduledDeliveryTime = "time"`

Description: An outgoing submit, that has previously failed at FDA, will have a scheduled delivery time set to this.

Type: String

Optionality: Optional (default used if not set).

Allowed: Format is as per the SMPP spec. A 10 minute delay would be "000000001000000R"

Default: ""

Notes:

Example: `scheduledDeliveryTime = ""`

`teleserviceRoutingMap`

Syntax: `teleserviceRoutingMap = [`*`mappings`*`]`

Description: Array of teleservice mappings.

SMPP has an optional attribute called service_type whose value is a string like EMS or CMT. The actual strings tend to be operator-specific. We can map these to numeric IS-41 teleservices in genericSM and set the alternate delivery allowed flag for each one.

Allowed

Default: The CMT and EMS entries are defined

Notes: Any service types not appearing in this list will have alternate delivery disallowed and a teleservice of 0.

Example:
```
teleserviceRoutingMap = [
    { serviceType =    "", teleservice =    0,
    allowAlternateDelivery = true }
    { serviceType = "CMT", teleservice = 4098,
    allowAlternateDelivery = true }
    { serviceType = "EMS", teleservice = 4101,
    allowAlternateDelivery = true }
]
```

`throttledCommandStatus`

Syntax: `throttledCommandStatus =` *`status`*

Description: Command status to return to throttled messages.

Allowed Refer to *SMPP command statuses* (on page 114) for a list of values.

Default: 88

Example: `throttledCommandStatus = 88`

`tlvWarnSuppress`

Syntax: `tlvWarnSuppress =` *`true`* `|` *`false`*

Description: If true, specifies that the warning is to be suppressed when unknown tlvs are received. If the parameter is omitted, the default value (false) is used and a warning is generated for each unkonwn tlv that is received.

Type: Integer, Decimal, Array, Parameter list, String, Boolean

Optionality: Optional (default used if not set)

Allowed: true, false

Default: false

Notes:

Example: `tlvWarnSuppress = false`

Configuring the IS-41 CDMA Adapter

IS-41 CDMA adapter overview

The IS41 adapter communicates between the Messaging Manager platform and SMSCs, HLRs, and MSCs, using IS-41.

The IS41 adapter needs to be configured to know about SMSCs that it will connect to.

General IS-41 CDMA configuration

The following section of the **eserv.config** specifies the general parameters for all IS-41 CDMA connections.

```
adapters =

    # IS-41 CDMA adapter
    {
        lib = "xmsiIS41.so"

        adapterName = "CDMA1"
        #pointCodes = [1005, 1006]
        SSN = 18

        earlyAckMC = false
        earlyAckSME = false

        allowConcatenatedFDA = true

        config = {
            global_parameters
        }
    }
]
```

General IS-41 CDMA parameters

Here are the general parameters for IS-41 CDMA.

`adapterName`

Syntax:	`adapterName = "name"`
Description:	Identifier for the adapter.
Allowed	Any text string, but should be meaningful, for example include the protocol used. For example: • "MAP1" for MAP • "EMI1" for EMI • "SCA1" for SCA • "SMPP1" for SMPP • "CDMA1" for IS-41 CDMA • "TDMA1" for IS-41 TDMA • "Wrapper" for Wrapper
Default:	No default.
Notes:	This name *must* also be in the configuration database before Messaging Manager will run correctly. See *MM User's Guide*.

`lib`

Syntax:	`lib = "library"`
Description:	The name of the library that contains the CDMA (UCA-IS41) adapter being configured.
Allowed	"xmsiIS41.so"
Default:	No default.
Example:	`lib = "xmsiIS41.so"`

`pointCodes`

Syntax:	`pointCodes = [pc1, pc2, ...]`
Description:	Destination point codes array of messages to be handled by this adapter. This parameter takes priority over SSN match.
Optionality:	Optional
Allowed	Defined by network administrator.
Example:	`pointCodes = [1001, 1002]`

`SSN`

Syntax:	`SSN = num`
Description:	Destination subsystem number of messages to be handled by this adapter.
Allowed	Valid subsystem number
Notes:	Non-zero to handle incoming TCAP.
Example:	`SSN = 18`

Global IS-41 CDMA configuration

The following section of the **eserv.config** specifies the global parameters for all IS-41 CDMA connections.

```
config = {

    allowIncoming = true
    allowOutgoing = true

    PC = 1005
    SSN = 6
    GT = ""
    TT = 0

    originatingTimeout = 20
    smsreqTimeout = 10
    smdppTimeout = 10
    smdppTimeoutSME = 10

    fallbackAlphabet = "UCS-2"

    supportIS841 = true

    defaultMessagePriority = "Normal"

    defaultEndpointType = "SME"

    rimsInterfaceName = "rimsIf"

    pathRetryRandomisation = 1
    pathRetrySegmentOffset = 1

    alarmMask = 0

    relaySmsNotifications =false

    minHLRTransType = 3
    mdnHLRTransType = 14

    defaultTransientFailureCauseCode = 33 # destination busy
```

```
            defaultPermanentFailureCauseCode = 39 # other terminal problem

            throttledFailureCauseCode = 35 # destination resource shortage

            deliveryFailureErrorClass = 2
            deliveryFailureStatusCode = 5

            protocol = "CDMA"

            allowFDAforWEMT = true
            allowAlternateDeliveryForWEMT = true

            tcapInterfaceServiceKey = 22

            incomingOriginatingNumberRules = [
                { fromNoa=3, prefix="04", min=4, max=32, remove=1,
                prepend="0064", resultNoa=4 }
                { fromNoa=2, prefix="4", min=4, max=32, remove=0, prepend="0064",
                resultNoa=4 }
            ]
            incomingDestinationNumberRules = [
            ]
            # outgoingOriginatingNumberRules = [ ]
            # outgoingDestinationNumberRules = [ ]

        } # End IS-41 CDMA adapter config section.
```

Global IS-41 CDMA parameters

Here are the parameters for configuring the global IS-41 CDMA adapter.

`alarmMask`

Syntax: `alarmMask = num`

Description: Alarm masking.

Allowed Sum of:

1	Mask originating timeout alarm.
2	Mask SMSREQ timeout alarm.
4	Mask SMSREQ abort alarm.
8	Mask SMDPP timeout alarm.
16	Mask SMDPP abort alarm.

Default: 0

Example: `alarmMask = 0`

`allowAlternateDeliveryForWEMT`

Syntax: `allowAlternateDeliveryForWEMT = true|false`

Description: Allow alternate delivery if the Teleservice identifier is WEMT.

Allowed true, false

Default true

Notes: Wireless enhanced messaging teleservice (WEMT) is used to carry enhanced messaging service (EMS) over SMS.

Example: `allowAlternateDeliveryForWEMT = true`

allowFDAforWEMT

Syntax: `allowFDAforWEMT = true|false`

Description: Allow FDA if the Teleservice identifier is WEMT.

Allowed true, false

Default true

Notes: Wireless enhanced messaging teleservice (WEMT) is used to carry enhanced messaging service (EMS) over SMS.

Example: `allowFDAforWEMT = true`

allowIncoming

Syntax: `allowIncoming = true|false`

Description: Determines whether inbound messages are allowed.

Optionality: Mandatory

Allowed true, false

Default false

Example: `allowIncoming = false`

allowOutgoing

Syntax: `allowOutgoing = true|false`

Description: Determines whether initial messages may be sent out through the MAP interface.

Optionality: Mandatory

Allowed true, false

Default false

Example: `allowOutgoing = false`

defaultEndpointType

Syntax: `defaultEndpointType = "type"`

Description: The endpoint type that is assumed when the message direction cannot be determined (for example, CDMA User Ack).

Optionality: Mandatory

Allowed "SME", "MC"

Default "SME"

Example: `defaultEndpointType = "SME"`

defaultMessagePriority

Syntax: `defaultMessagePriority = "str"`

Description: The priority to be applied to incoming messages that do not have a priority.

Type: String

Optionality: Optional (default used if parameter absent).

Allowed

Default "Normal"

Notes:

Example: `defaultMessagePriority = "Normal"`

`defaultPermanentFailureCauseCode`

Syntax:	`defaultPermanentFailureCauseCode = code`
Description:	Default SMS_CauseCodes mappings for Permanent failures if no releaseCauseMap entry matches.
Optionality:	
Allowed	Refer to *SMS_CauseCodes* (on page 126) for a list of cause codes.
Default	39
Example:	`defaultPermanentFailureCauseCode = 39`

`defaultTransientFailureCauseCode`

Syntax:	`defaultTransientFailureCauseCode = code`
Description:	Default SMS_CauseCodes mappings for Transient failures if no releaseCauseMap entry matches.
Optionality:	
Allowed	Refer to *SMS_CauseCodes* (on page 126) for a list of cause codes.
Default	33
Example:	`defaultTransientFailureCauseCode = 33`

Delivery failure error class and status code

Values of the parameters:

- `deliveryFailureErrorClass`
- `deliveryFailureStatusCode`

Set the default error class and status code to be used in a failure delivery report.

SMSDelAckMsg->BearerData->MessageStatus->ErrorClass

SMSDelAckMsg->BearerData->MessageStatus->MsgStatusCode

They must form a valid pair. Valid pairs are:

Error class	Status code
0	0,1,2,3
2	4,5,31
3	4 to 11, 31

Invalid pairs will be reset to 2 and 5 respectively in the code.

`deliveryFailureErrorClass`

Syntax:	`deliveryFailureErrorClass = class`
Description:	This value will be used in the error class of delivery receipt messages which signify non-delivery of a message.
Optionality:	
Allowed	0 no error 2 temporary condition 3 permanent condition
Default	2
Example:	`deliveryFailureErrorClass = 2`

deliveryFailureStatusCode

Syntax: `deliveryFailureStatusCode = code`

Description: This value will be used in the TP-Status code of delivery receipt messages which signify non-delivery of a message.

Optionality:

Allowed

0	Message accepted
1	Message deposited to Internet
2	Message delivered
3	Message canceled
4	Network congestion
5	Network error
6	Cancel failed
7	Blocked destination
8	Text too long
9	Duplicate message
10	Invalid destination
11	Message expired
31	Unknown error

Default 5

Example: `deliveryFailureStatusCode = 5`

fallbackAlphabet

Syntax: `fallbackAlphabet = "charset"`

Description: The character set to use for outgoing messages when the protocol cannot handle the desired alphabet.

Type: String

Optionality: Optional (default used if not set)

Allowed:
- Binary
- 8859-1
- 8859-2
- 8859-3
- 8859-4
- 8859-5
- 8859-6
- 8859-7
- 8859-8
- 8859-9
- 8859-10
- 8859-11
- 8859-12
- 8859-13
- 8859-14
- 8859-15
- 8859-16
- ASCII7Bit
- 646
- UTF-8
- UCS-2
- GSM7Bit
- GSM8Bit
- GSM7BitComp
- GSMBinaryComp
- GSMUCS2Comp
- JIS
- XKJIS
- ISO-2022-JP
- PCK
- ko_KR-euc
- SMPPPictogram

Default: UCS-2

Notes: When the fallback alphabet is used, a debug entry similar to the following is logged: "unsupported alphabet: ... for map version ... fallback to: *charset*"

Example:
```
fallbackAlphabet = "GSM8Bit"
```

GT

Syntax:
```
GT = "gt"
```

Description: Originating global title in outgoing messages.

Optionality:

Allowed

Default ""

Example:
```
GT = ""
```

mdnHLRTransType

Syntax: `mdnHLRTransType = type`

Description: The global title translation type to be used for lookups to the HLR when using an MDN.

Default: 3

Notes: `mdnHLRTransType = 3`

minHLRTransType

Syntax: `minHLRTransType = type`

Description: The global title translation type to be used for lookups to the HLR when using a MIN.

Default: 3

Notes: `minHLRTransType = 3`

originatingTimeout

Syntax: `originatingTimeout = num`

Description: Configure our timeout on processing of inbound messages.

Allowed: Seconds

Notes: `originatingTimeout = 20`

pathRetryRandomisation

Syntax: `pathRetryRandomisation = seconds`

Description: The amount of randomization for the delay interval when selecting a delay for a retry.

Allowed: integer

Default: 1

Notes:

Example: `pathRetryRandomisation = 1`

pathRetrySegmentOffset

Syntax: `pathRetrySegmentOffset = seconds`

Description: How much to delay later segments of a concatenated message when a delayed retry is attempted.

Allowed: integer

Notes:

Example: `pathRetrySegmentOffset = 1`

PC

Syntax: `PC = code`

Description: Originating point code in outgoing messages.

Allowed: integer in range [0, 255]

Notes:

Example: `PC = 1005`

`protocol`

Syntax:	`protocol = "prot"`
Description:	The outbound protocol that this adapter uses.
Allowed:	"CDMA" for IS-41 CDMA "TDMA" for IS-41 TDMA.
Notes:	
Example:	`protocol = "CDMA"`

`relaySmsNotifications`

Syntax:	`relaySmsNotifications = true|false`
Description:	We can relay SMS-NOT alerts from the HLR to the SMSC, but that is usually a bad idea because the SMSC will have registered for its own alerts. It does not need ours as well.
Optionality:	Mandatory
Allowed	true, false
Default	false
Example:	`relaySmsNotifications = false`

`releaseCauseMap`

Description:	Map of releaseCauses to SMS_CauseCodes, plus default mappings for transient and permanent failures if no releaseCauseMap entry matches.
Notes:	If ACS did not release the call and the outbound adapter generated an error result code, the following releaseCodes will be used: • TransientFailure - 1001 • PermanentFailure - 1002 • Abort - 1003 Refer to *SMS_CauseCodes* (on page 126) for a list of cause codes.

`releaseCode`

Syntax:	`releaseCode = code`
Description:	INAP releaseCode identifier.
Allowed:	Refer to *SMS_CauseCodes* (on page 126) for a list of cause codes.

`rimsInterfaceName`

Syntax:	`rimsInterfaceName = "name"`
Description:	Name of the configured Messaging Manager Navigator interface.
Default:	"rimsIf"
Notes:	This must match the interface name in **IN/service_packages/SLEE/etc/SLEE.cfg**
Example:	`rimsInterfaceName = "rimsIf"`

`SMS_CauseCode`

Description:	Error code in SMDPP-Nack.
Notes:	Refer to *SMS_CauseCodes* (on page 126) for a list of cause codes.

SMS_CauseCodes

Here is a list of SMS_CauseCodes used by the following parameters:

- releaseCauseMap
- defaultTransientFailureCauseCode
- defaultPermanentFailureCauseCod
- throttledFailureCauseCode

Value	CauseCode
0	AddressVacant
1	AddressTranslationFailure
2	NetworkResourceShortage
3	NetworkFailure
4	InvalidTeleserviceID
5	OtherNetworkProblem
6-31	Values 6 through 31 reserved - treat as 5 OtherNetworkProblem
32	NoPageResponse
33	DestinationBusy
34	NoAcknowledgement
35	DestinationResourceShortage
36	SMSDeliveryPostponed
37	DestinationOutOfService
38	DestinationNoLongerAtThisAddress
39	OtherTerminalProblem
40-47	Values 40 through 47 reserved - treat as 39 OtherTerminalProblem
48-63	Values 48 through 63 reserved - treat as 36 SMSDeliveryPostponed
64	RadioInterfaceResourceShortage
65	RadioInterfaceIncompatibility
66	OtherRadioInterfaceProblem
67-95	Values 67 through 95 reserved - treat as OtherRadioInterfaceProblem
96	EncodingProblem
97	SMSOriginationDenied
98	SMSTerminationDenied
99	SupplementaryServiceNotSupported
100	SMSNotSupported
101	Value 101 reserved.
102	MissingExpectedParameter
103	MissingMandatoryParameter
104	UnrecognizedParameterValue
105	UnexpectedParameterValue
106	DataSizeError
107	OtherGeneralProblems
108-255	Values 108 through 255 reserved - treat as 107 OtherGeneralProblems.

smsreqTimeout

Syntax: `smsreqTimeout = seconds`

Description: Configure our timeout on processing of the outbound timers for the SMSREQ messages.

Allowed:

Example: `smsreqTimeout = 10`

smdppTimeout

Syntax: `smdppTimeout = seconds`

Description: Configure our timeout on processing of outbound timers for the SMDPP messages.

Allowed:

Example: `smdppTimeout = 10`

smdppTimeoutSME

Syntax: `smdppTimeoutSME = seconds`

Description: Configure our timeout on processing of outbound timers for the SMDPP messages to an SME.

Allowed:

Default: no default

Note: If it is not specified, then `smdppTimeout` will be used.

Example: `smdppTimeout = 10`

SSN

Syntax: `SSN = num`

Description: Destination subsystem number of messages to be handled by this adapter.

Allowed Valid subsystem number

Notes: Non-zero to handle incoming TCAP.

Example: `SSN = 18`

supportIS841

Syntax: `supportIS841 = true|false`

Description: IS-841 is an extension of IS-41 and allows the storing of an MSID in SMSRequestResults. The feature is needed for IMSI masking.

Allowed true, false

Default true

Note: Set this to false if your hardware does not support IS-841

Example: `supportIS841 = true`

throttledFailureCauseCode

Syntax: `throttledFailureCauseCode = code`

Description: Cause code to return when message is throttled.

Allowed Refer to *SMS_CauseCodes* (on page 126) for a list of cause codes.

Default 35

Example: `throttledFailureCauseCode = 35`

Configuring the IS-41 TDMA Adapter

IS-41 TDMA adapter overview

The IS41 adapter communicates between the Messaging Manager platform and SMSCs, HLRs, and MSCs, using IS-41.

The IS41 adapter needs to be configured to know about SMSCs that it will connect to.

General IS-41 TDMA configuration

The following section of the **eserv.config** specifies the general parameters for all IS-41 TDMA connections.

```
adapters =

    # IS-41 TDMA adapter
    {
        lib = "xmsiIS41.so"

        adapterName = "TDMA1"
        #pointCodes = [1005, 1006]
        SSN = 18

        earlyAckMC = false
        earlyAckSME = false

        allowConcatenatedFDA = true

        config = {
            global_parameters
        }
    }
]
```

General IS-41 TDMA parameters

The parameters for the IS-41 TDMA adapter are the same as those for the IS-41 CDMA adapter. See *General IS-41 CDMA parameters* (on page 118) for the description of each parameter.

Note: The `lib` parameter is the same name as for IS41 CDMA, but the `adapterName` is different.

Global IS-41 TDMA configuration

The following section of the **eserv.config** specifies the global parameters for all IS-41 TDMA connections.

```
        config = {

            allowIncoming = true
            allowOutgoing = true

            PC = 1005
            SSN = 6
            GT = ""
            TT = 0

            originatingTimeout = 20
            smsreqTimeout = 10
            smdppTimeout = 10

            defaultMessagePriority = "Normal"
```

```
        supportIS841 = true

        rimsInterfaceName = "rimsIf"

        alarmMask = 0

        releaseCauseMap = [  #<??is this still in config??>
            { releaseCode = 31, SMS_CauseCode = 12 }
            { releaseCode = 16, SMS_CauseCode = 12 }
            { releaseCode = 17, SMS_CauseCode = 9 }
        ]
        defaultTransientFailureCauseCode = 8
        defaultPermanentFailureCauseCode = 7

        causeCodeMap = [
            { SMS_CauseCode = 12, failureCode = 31, permanent = false }
            { SMS_CauseCode = 9, failureCode = 16, permanent = true }
        ]

        defaultReleaseCause = 13
        defaultReleaseCausePermanent = false

        deliveryFailureErrorClass = 2
        deliveryFailureStatusCode = 5

        protocol = "TDMA"

        tcapInterfaceServiceKey = 22

        incomingOriginatingNumberRules = [
            { fromNoa=3, prefix="04", min=4, max=32, remove=1,
            prepend="0064", resultNoa=4 }
            { fromNoa=2, prefix="4", min=4, max=32, remove=0, prepend="0064",
            resultNoa=4 }
        ]
        incomingDestinationNumberRules = [
        ]
        # outgoingOriginatingNumberRules = [ ]
        # outgoingDestinationNumberRules = [ ]

    } # End IS-41 TDMA adapter config section.
```

Global IS-41 TDMA parameters

The parameters for the IS-41 TDMA adapter are the same as those for the IS-41 CDMA adapter. See *Global IS-41 CDMA parameters* (on page 120) for the description of each parameter.

Configuring the SCA Adapter

SCA adapter overview

The SCA adapter provides support for SIP instant messaging. It interacts with the Convergent Charging Controller Session Control Agent (SCA to convert SIP instant messages (MESSAGE messages) to and from SMSs.

The SCA adapter does not provide an interface to SIP clients but instead interacts with the SCA and xmsTrigger. The SCA adapter converts (adapts) SipSleeEvents received from the SCA to GenericSM events processed by Messaging Manager. In the reverse direction, the SCA adapter converts GenericSM events to SipSleeEvents.

Note: The SCA adapter in MM interacts with the SCA). SCA must be configured in order to make the adapter work. See *SCA Technical Guide* for SCA configuration details.

General SCA configuration

The following section of the **eserv.config** specifies the general parameters for all SIP connections.

```
adapters =

    # SCA (SIP) adapter
    {
        disabled = False

        interfaceName = "xmsIf"

        lib = "mmxiSCA.so"

        adapterName = "SCA1"

        config = {
            global_parameters
        }
]
```

General SCA parameters

Here are the general parameters for SCA.

`adapterName`

Syntax:	`adapterName = "name"`
Description:	Identifier for the adapter.
Allowed	Any text string, but should be meaningful, for example include the protocol used. For example: • "MAP1" for MAP • "EMI1" for EMI • "SCA1" for SCA • "SMPP1" for SMPP • "CDMA1" for IS-41 CDMA • "TDMA1" for IS-41 TDMA • "Wrapper" for Wrapper
Default:	No default.
Notes:	This name *must* also be in the configuration database before Messaging Manager will run correctly. See *MM User's Guide*.

`disabled`

Syntax:	`disabled = true|false`
Description:	To enable or disable the adapter.
Allowed	true, false
Default:	false
Example:	`disabled = False`

`interfaceName`

Syntax:	`interfaceName = "name"`
Description:	The SLEE interface name for Messaging Manager; to send messages from SCA to Messaging Manager.
Allowed	String
Default:	"xmsIf"
Note:	This *must* match the name specified in the **SLEE.cfg** file.
Example:	`interfaceName = "xmsIf"`

`lib`

Syntax:	`lib = "library"`
Description:	The name of the library that contains the SCA adapter being configured.
Allowed	"mmxiSCA.so"
Default:	No default.
Example:	`lib = "mmxiSCA.so"`

Global SCA configuration

The following section of the **eserv.config** specifies the global parameters for all SCA connections.

```
config = {
    sca = {
        serviceKey = 52

        # interface = "sca"
    }

    pathReportingInterval = 60

    inboundTimeout = 5

    outboundTimeout = 5
}
```

Global SCA parameters

Here are the parameters for configuring the global SCA adapter.

`inboundTimeout`

Syntax:	`inboundTimeout = seconds`
Description:	The timeout, in seconds, of inbound transactions.
Type:	Integer
Optionality:	Optional
Default:	5
Example:	`inboundTimeout = 5`

`interface`

Syntax:	`interface = "name"`
Description:	The SLEE service name for SCA; to send messages from the SCA adapter to SCA.
Allowed	String

Default:	No default.
Notes:	This *must* match the entry in the **SLEE.cfg** file. Configure either this parameter or the `serviceKey` parameter, but not both.
Example:	`interface = "sca"`

`outboundTimeout`

Syntax:	`inboundTimeout = seconds`
Description:	The timeout, in seconds, of outbound transactions.
Type:	Integer
Optionality:	Optional
Default:	10
Example:	`outboundTimeout = 5`

`PathReportingInterval`

Syntax:	`PathReportingInterval = seconds`
Description:	The frequency, in seconds, of path connection reports.
Type:	Integer
Optionality:	Optional
Default:	2
Example:	`PathReportingInterval = 60`

`serviceKey`

Syntax:	`serviceKey = key`
Description:	The SLEE service key for SCA; to send messages from the SCA adapter to SCA.
Allowed	52
Default:	No default.
Notes:	This *must* match the entry in the **SLEE.cfg** file. Configure either this parameter or the `interface` parameter, but not both.
Example:	`serviceKey = 52`

Configuring the Wrapper Adapter

General Wrapper adapter configuration

The following section of the **eserv.config** specifies the general parameters for the Wrapper adapter (the endpoint for the SSMN macro node).

```
adapters =

    # Internal adapter (The endpoint for the SSMN macro node)
    {
        lib = "xmsiWrapper.so"
        SSN = 40
        adapterName = "Wrapper"

        config = {
        }
    }
```

```
}
```

General Wrapper adapter parameters

Here are the general parameters for the Wrapper adapter.

`adapterName`

Syntax:	`adapterName = "name"`
Description:	Identifier for the adapter.
Allowed	Any text string, but should be meaningful, for example include the protocol used. For example: • "MAP1" for MAP • "EMI1" for EMI • "SCA1" for SCA • "SMPP1" for SMPP • "CDMA1" for IS-41 CDMA • "TDMA1" for IS-41 TDMA • "Wrapper" for Wrapper
Default:	No default.
Notes:	This name *must* also be in the configuration database before Messaging Manager will run correctly. See *MM User's Guide*.

`lib`

Syntax:	`lib = "library"`
Description:	The name of the library that contains the Wrapper adapter being configured.
Allowed	"xmsiWrapper.so"
Default:	No default.
Example:	`lib = "xmsiWrapper.so"`

`SSN`

Syntax:	`SSN = num`
Description:	Destination subsystem number of messages to be handled by this adapter.
Allowed	Valid subsystem number
Notes:	Non-zero to handle incoming TCAP.
Example:	`SSN = 18`

Global Wrapper adapter configuration

The following section of the **eserv.config** specifies the global parameters for the Wrapper adapter.

```
config = {
    xmsTimeout = 5
    tcapTimeout = 10
    xmsWrapper = {
        interface= "xmsIf"
        pc = 55
        ssn = 7
        type = "itu"
        gt = ""
    }
    xmsTrigger = {
```

```
                pc = 51
                ssn = 3
                type = "itu"
                gt = ""
            }
        }
```

Global Wrapper adapter parameters

Here are the parameters for configuring the global Wrapper adapter.

`gt`

Syntax:	`GT = "gt"`
Description:	Global title
Note:	This is defined by the network administrator. Required only if the point code is not defined.

`pc`

Syntax:	`PC = code`
Description:	Point code
Allowed:	Integer in range [0, 255]
Example:	`PC = 55`

`ssn`

Syntax:	`SSN = num`
Description:	Subsystem number.
Allowed	Integer between 0 and 65535.
Note:	Required only if the point code has been defined.
Default:	0
Example:	`SSN = 0`

`tcapTimeout`

Syntax:	`tcapTimeout = seconds`
Description:	How long to wait for response from XMS remote wrapper.
Allowed:	
Default:	10
Note	This entry is only used in conjunction with the xmsWrapperTA SLEE interface.
Example:	`tcapTimeout = 10`

`type`

Syntax:	`type = "type"`
Description:	Protocol type.
Allowed:	"itu", "ansi"
Example:	`type = "itu"`

xmsTimeout

Syntax:	`xmsTimeout = seconds`
Description:	How long to wait for a response from XMS trigger application.
Allowed:	**Suggested value:** Look in the MAP adapter section and add up `mscTimeout` (on page 86) and `hlrTimeout` (see *Messaging Manager Navigator Technical Guide*) and add a bit of time to that (up to 5 seconds).
Default:	5
Example:	`xmsTimeout = 5`

xmsTrigger

Syntax:	`xmsTrigger = {trigger_parameters}`
Description:	Configuration parameters used to determine the source address for outgoing messages.
Note:	No dialogs needed. Address section is not used for working with SSMN node.

xmsWrapper

Syntax:	`xmsWrapper = {wrapper_parameters}`
Description:	Configuration parameters used to create dialogs to the TCAP stack, and to determine the destination address.
Note:	Remote side's configuration. Ignored in normal operation (that is, no xmsWrapperTA).

Chapter 4

Configuring Messaging Manager Director

Overview

Introduction

This chapter briefly explains how to configure the Messaging Manager Director modules. This includes details of the configuration of specific nodes that are used for the MM application.

There are many feature nodes that are supplied by the ACS application, and may be used in Messaging Manager control plans. These feature nodes are detailed in *Feature Nodes Reference Guide*; please refer to this document for instructions.

In this chapter

This chapter contains the following topics.

Configuring Chassis Actions

Introduction

The ChassisActions section of Messaging Manager contains the configuration to enable:

- The Send Short Message Notification (SSMN) node to send a GenericSM and wait for a result so that it can branch on the result type
- A USSD node to send a message

ChassisActions configuration

Here is an example `ChassisActions` configuration.

```
ChassisActions = {

    SendGenericMessageAction = {
        interfaceName = "xmsIf"
        timeoutTick = 5
        tcapOrigAddr = { PC = 0, SSN = 0, type = "itu" }
        tcapDestAddr = { PC = 0, SSN = 0, type = "itu" }
    }

    ussdChassisAction = {
        tcapInterfaceName = "hssScIf"
        timeoutTick = 5
```

```
        recordResponseTimes = false
    }
}
```

SendGenericMessageAction parameters

Here are the parameters in the `SendGenericMessageAction` sub-section of the `ChassisActions` section of the **eserv.config**.

`InterfaceName`

Syntax:	`interfaceName = "`*`name`*`"`
Description:	The SLEE interface to send GenericSMs to.
Type:	String
Optionality:	Mandatory
Default:	"xmsIf"
Example:	`interfaceName = "xmsIf"`

`tcapDestAddr`

Description:	TCAP destination address to use when ACS and MM are on different ma1chines.
Default:	Not set
Note:	Required only if MM and ACS are on separate machines. See `PC`, `SSN`, and `type` parameters.
Example:	`tcapDestAddr = [ PC = 1, SSN = 40, type = "itu" }`

`tcapOrigAddr`

Description:	TCAP originating address to use when ACS and MM are on different machines.
Default:	Not set
Note:	Required only if MM and ACS are on separate machines. See `PC`, `SSN`, and `type` parameters.

`timeoutTick`

Syntax:	`timeoutTick = `*`seconds`*
Description:	The granularity of SLEE dialog timeout checks.
Note:	This can conflict with the option of the same name in the RIMS chassis actions. In case of a conflict, the value to use is undefined.
Example:	`timeoutTick = 5`

`xmsDirectFromPrefix`

Syntax:	`xmsDirectFromPrefix = "`*`prefix`*`"`
Description:	A prefix that will be added to the source MSISDN when the **MMX Direct** transport is chosen in the notification configuration.
Type:	String
Optionality:	Optional (default used if not set).
Allowed:	
Default:	"" (no prefix is added)
Notes:	
Example:	`xmsDirectFromPrefix = "1"`

`xmsFDAFromPrefix`

Syntax:	`xmsFDAFromPrefix = "prefix"`
Description:	A prefix that will be added to the source MSISDN when the **MMX FDA** transport is chosen in the notification configuration.
Type:	String
Optionality:	Optional (default used if not set).
Allowed:	
Default:	"" (no prefix is added)
Notes:	
Example:	`xmsFDAFromPrefix = "2"`

ussdChassisAction parameters

Here are the parameters in the `ussdChassisAction` sub-section of the `ChassisActions` section of the **eserv.config**.

`recordResponseTimes`

Syntax:	`recordRespons`*`eTimes = true|false`
Description:	Whether to record response times in a temporary storage tag.
Type:	Boolean
Optionality:	Optional (default used if not set).
Allowed:	true, false
Default:	false
Notes:	If true, the response time is stored in tag 3932504, which is of type "Unsigned 32-bit Integer" and is named "USSD_NOTIF_RESPONSE_TIME".
Example:	`recordResponseTimes = false`

`tcapInterfaceName`

Syntax:	`tcapInterfaceName = "`*`name`*`"`
Description:	The SLEE interface to send the USSD message to.
Type:	String
Default:	"hssScIf"
Example:	`tcapInterfaceName = "hssScIf"`

`timeoutTick`

Syntax:	`timeoutTick =` *`seconds`*
Description:	The granularity of SLEE dialog timeout checks.
Note:	This can conflict with the option of the same name in the RIMS chassis actions. In case of a conflict, the value to use is undefined.
Example:	`timeoutTick = 5`

acsChassis configuration in acs.conf

This text shows the denormalization configuration in **acs.conf** for the Send USSD Notification node.

Note: This text only shows the configuration which is specific to the node. For more information about **acs.conf**, see *ACS Technical Guide*.

```
acsChassis
 useCustomUSSDNotificationNoARules 0|1:

sendUSSDNotification
  DenormalisationRule (00,2,0,E):
```

sendUSSDNotification

Syntax:	`sendUSSDNotification`
	` DenormalisationRule (`*`rule`*`)`
	` [...]`
	` ...:`

Description:	The denormalization rules to apply to outbound notifications from the Send USSD Notification node.
Type:	Array
Optionality:	Optional (not used if not set).
Allowed:	
Default:	None
Notes:	This parameter will only have an effect if the *useCustomUSSDNotificationNoARules* (on page 140) parameter is set to 1. The denormalization rules have the same format as the rules from the `acsChassis` section of **acs.conf**. For more information, see *ACS Technical Guide*.
Example:	`sendUSSDNotification` ` DenormalisationRule (00,2,0,E):`

useCustomUSSDNotificationNoARules

Syntax:	`useCustomUSSDNotificationNoARules 0|1`
Description:	Which rules to use when denormalizing numbers which are sent by the Send USSD Notification node.
Type:	Boolean
Optionality:	Optional (default used if not set).
Allowed:	0 Apply the standard denormalization rules. 1 Apply the denormalization rules defined in the *sendUSSDNotification* (on page 140) section.
Default:	0
Notes:	
Example:	`useCustomUSSDNotificationNoARules 1`

Configuring Macro Nodes

Introduction

Macro nodes are feature nodes that are used by ACS using the ACS Control Plan Editor, but are not part of the ACS product. They are supplied by other Convergent Charging Controller applications, but require the presence of ACS for use.

Macro nodes require some configuration to be entered into the **eserv.config** file. The following sections will detail the configuration that is necessary for the MM macro nodes.

The macro node reads the global configuration file (**eserv.config**) on initialization. Should the configuration of a macro node be changed, the configuration files must be re-read.

For more information about the macro nodes used by Messaging Manager, see *Feature Nodes Reference Guide*.

eserv.config Macro Node configuration

This is a high level view of the `macroNodes` configuration section of **eserv.config**.

```
XMS = {
    macroNodes = {
        Macro_Node_Name = {
            configuration_for_macro_node
        }
    }
}
```

macroNodes configuration

Here is an example configuration.

```
macroNodes = {

    SendShortMessageNode = {
        xmsiWrapperIfName = "xmsIf"

        tcapOrigAddr = { PC = 0, SSN = 0, type = "itu" }
        tcapDestAddr = { PC = 0, SSN = 0, type = "itu" }

        dateFormat = "%A %d %B %Y"
        timeFormat = "%I:%M %p"
        time24Format = "%H:%M %Z"
        callTimeFormat = "%I:%M %p"
        maximumDestinations = 1000

        numberPlan = 1

    }

    SendUSSDNotificationNode = {
        MSISDNTranslationType = 2
        destSSN = 6
    }

}
```

Send Short Message Node configuration

The Send Short Message Notification node is a Messaging Manager macro node that also uses the Wrapper adapter to send out the notification.

Therefore there are additional configuration tasks that must be completed for this node to function correctly:

- The node must be configured in the `macroNodes` configuration section of the **eserv.config** file.
- The Wrapper adapter must be configured in the **eserv.config** file. See *Configuring the Wrapper Adapter* (on page 133).

Send Short Message node parameter descriptions

Here are the `macroNodes` configuration parameters for the Send Short Message feature node.

callTimeFormat

Syntax: `callTimeFormat = "format"`

Description: Defines the format of the <CALL_TIME> token.

Type:

Optionality:

Allowed: See *Date and time formats* (on page 145).

Default: "%I:%M %p"

Notes:

Example: `callTimeFormat = "%I:%M %p"`

dateFormat

Syntax: `dateFormat = "format"`

Description: Defines the format of the <DATE> token.

Type:

Optionality:

Allowed: See *Date and time formats* (on page 145).

Default: “%A %d %B %Y”

Notes:

Example: `dateFormat = "%A %d %B %Y"`

maximumDestinations

Syntax: `maximumDestinations = num`

Description: Specifies the absolute limit of addresses that will get an SMS if the Profile Tag destination address type is used.

Default: 1000

Notes: The node will loop over the profile list sending notification SMSs (for an individual call) until this limit is reached. This is purely a safety limit to prevent a massive load on the system, and not intended to be the normal method of broadcasting - there is already a better way to broadcast 1 million identical SMSs.

Example: `maximumDestinations = 1000`

name

Description: Name of unit type.

Note: Unknown unit types will have a name of “units”.

numberPlan

Syntax: `numberPlan = num`

Description: The Number plan value to use on the message.

Type: Integer

Optionality: Mandatory

Allowed: These more or less follow the SMPP:

0	unknown
1	isdn
3	data
4	telex

	6	land_mobile
	8	national
	9	private
	10	ermes
	13	pc_ssn
	14	ip
	18	wap

Default: 1
Example: `numberPlan = 1`

PC

Syntax: `PC = pc`
Description: The point code.
Type: Integer
Optionality: Optional
Allowed: integer in range [0, 255]
Default:
Notes:
Example: `PC = 123`

SSN

Syntax: `SSN = num`
Description: Subsystem number
Allowed Integer in range [0, 65535]
Note: Required, if TCAP addresses are defined.
Example: `SSN = 0`

tcapDestAddr

Description: TCAP destination address to use when ACS and MM are on different ma1chines.
Default: Not set
Note: Required only if MM and ACS are on separate machines.
See `PC`, `SSN`, and `type` parameters.
Example: `tcapDestAddr = [ PC = 1, SSN = 40, type = "itu" }`

tcapOrigAddr

Description: TCAP originating address to use when ACS and MM are on different machines.
Default: Not set
Note: Required only if MM and ACS are on separate machines.
See `PC`, `SSN`, and `type` parameters.

time24Format

Syntax: `time24Format = "format"`
Description: Defines the format of the <TIME24> token.

Type:
Optionality:
Allowed: See *Date and time formats* (on page 145).
Default: "%H:%M %Z"
Notes:
Example: `time24Format = "%H:%M %Z"`

`timeFormat`

Syntax: `timeFormat = "format"`
Description: Defines the format of the <TIME> token.
Type:
Optionality:
Allowed: See *Date and time formats* (on page 145).
Default: "%I:%M %p"
Notes:
Example: `timeFormat = "%I:%M %p"`

`type`

Syntax: `type = "type"`
Description: Address type.
Allowed: "itu", "ansi"
Example: `type = "itu"`

`xmsiWrapperIfName`

Syntax: `xmsiWrapperIfName= "name"`
Description: Which SLEE interface name to send the message to.
Optionality Mandatory.
Type: String
Default: "xmsIf"
Note: This should be the interface name of the XMS remote wrapper interface.
Example: `xmsiWrapperIfName= "xmsIf"`

Notes

- While the three date and time formats, as configured above are true to their names, they need not be. Each format will be used to convert the same time_t into a string. So, the three options represent three completely interchangeable date formats.
- If ACS and Messaging Manager Director is on a different machine to Messaging Manager Multigate, the whole of the `tcapOrigAddr` and `tcapDestAddr` is required and used for routing.
- If ACS, Messaging Manager Director and Messaging Manager Multigate are installed on the same machine only the SSN from the `tcapDestAddr` is used, the `PC` and `type` must be defined, but are ignored by the system. This is because Messaging Manager uses the SSN to select the adapter to route to.

Date and time formats

The formats are all derived from the standard C function strftime, which converts a time_t into ordinary characters.

Variable	Description
%A	Replaced by national representation of the full weekday name.
%a	Replaced by national representation of the abbreviated weekday name.
%B	Replaced by national representation of the full month name.
%b	Replaced by national representation of the abbreviated month name.
%C	Replaced by (year / 100) as decimal number; single digits are preceded by a zero.
%c	Replaced by national representation of time and date.
%D	Equivalent to ``%m/%d/%y''.
%d	Replaced by the day of the month as a decimal number (01-31).
%E* %O*	POSIX locale extensions. The sequences %Ec %EC %Ex %EX %Ey %EY %Od %Oe %OH %OI %Om %OM %OS %Ou %OU %OV %Ow %OW %Oy are supposed to provide alternate representations. Additionly %OB implemented to represent alternative months names (used standalone, without day mentioned).
%e	Replaced by the day of month as a decimal number (1-31); single digits are preceded by a blank.
%F	Equivalent to ``%Y-%m-%d''.
%G	Replaced by a year as a decimal number with century. This year is the one that contains the greater part of the week (Monday as the first day of the week).
%g	Replaced by the same year as in ``%G'', but as a decimal number without century (00-99).
%H	Replaced by the hour (24-hour clock) as a decimal number (00-23).
%h	The same as %b.
%I	Replaced by the hour (12-hour clock) as a decimal number (01-12).
%j	Replaced by the day of the year as a decimal number (001-366).
%k	Replaced by the hour (24-hour clock) as a decimal number (0-23); single digits are preceded by a blank.
%l	Replaced by the hour (12-hour clock) as a decimal number (1-12); single digits are preceded by a blank.
%M	Replaced by the minute as a decimal number (00-59).
%m	Replaced by the month as a decimal number (01-12).
%n	Replaced by a newline.
%O*	The same as %E*.
%p	Replaced by national representation of either "ante meridiem" or "post meridiem", as appropriate.
%R	Equivalent to ``%H:%M''.
%r	Equivalent to ``%I:%M:%S %p''.
%S	Replaced by the second as a decimal number (00-60).
%s	Replaced by the number of seconds since the Epoch, UTC (see mktime(3)).
%T	Equivalent to ``%H:%M:%S''.

Variable	Description
%t	Replaced by a tab.
%U	Replaced by the week number of the year (Sunday as the first day of the week) as a decimal number (00-53).
%u	Replaced by the weekday (Monday as the first day of the week) as a decimal number (1-7).
%V	Replaced by the week number of the year (Monday as the first day of the week) as a decimal number (01-53). If the week containing January 1 has four or more days in the new year, then it is week 1; otherwise it is the last week of the previous year, and the next week is week 1.
%v	Equivalent to ``%e-%b-%Y''.
%W	Replaced by the week number of the year (Monday as the first day of the week) as a decimal number (00-53).
%w	Replaced by the weekday (Sunday as the first day of the week) as a decimal number (0-6).
%X	Replaced by national representation of the time.
%x	Replaced by national representation of the date.
%Y	Replaced by the year with century as a decimal number.
%y	Replaced by the year without century as a decimal number (00-99).
%Z	Replaced by the time zone name.
%z	Replaced by the time zone offset from UTC; a leading plus sign stands for east of UTC, a minus sign for west of UTC, hours and minutes follow with two digits each and no delimiter between them (common form for RFC 822 date headers).
%+	Replaced by national representation of the date and time (the format is similar to that produced by date(1)).
%%	Replaced by `%'.

Send USSD Notification Node configuration

Here are the `macroNodes` configuration parameters for the Send USSD Notification feature node.

`destSSN`

Syntax: `destSSN = int`
Description: The SSN used in the outgoing destination TCAP address.
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
Default: 6
Notes:
Example: `destSSN = 147`

`MSISDNTranslationType`

Syntax: `MSISDNTranslationType = int`
Description: The destination party translation type for USSD messages sent by the Send USSD Notification node.
Type: Integer
Optionality: Mandatory

Allowed:	Any valid translation type	For more information about valid types, see Q.713.
	0	Do not set a translation type.
Default:	0	
Notes:		
Example:	`MSISDNTranslationType = 3`	

`sendMsisdn`

Syntax:	`sendMsisdn = true|false`
Description:	To decide whether to send MSISDN in outgoing notification.
Type:	Boolean
Optionality:	Optional
Allowed:	true, false
Default:	true
Notes:	
Example:	`sendMsisdn = false`

`sendOrigAddress`

Syntax:	`sendOrigAddress = true|false`
Description:	To decide whether to send OriginationAddress in outgoing notification.
Type:	Boolean
Optionality:	Optional
Allowed:	true, false
Default:	true
Notes:	
Example:	`sendOrigAddress = false`

Note

The Send USSD Notification node uses the Send Short Message time token formatting parameters to format the time tokens used in its notifications.

Attempt Delivery Pending feature node configuration

Here are the `macroNodes` configuration parameters for the Attempt Delivery Pending feature node.

`reserveOnfirstSegment`

Syntax:	`reserveOnFirstSegment = true|false`
Description:	If set to true, send a reserve (and cancel) to the VWS for the first segment of a concatenated message.
Type:	Boolean
Optionality:	Optional (default used if not set)
Allowed:	true, false
Default:	true
Notes:	
Example:	`reserveOnFirstSegment = true`

`bytesPerSegment`

Syntax:	`bytesPerSegment = int`
Description:	Used to calculate the size of a concatenated message when billing on the last segment.
Type:	Integer
Optionality:	Optional (default used if not set)
Allowed:	
Default:	140
Notes:	
Example:	`bytesPerSegment = 140`

`checkReservationVolumeResponse`

Syntax:	`checkReservationVolumeResponse = true|false`
Description:	If set to true, check a successful reservation response for the correct numbers. If set to false, assume that the numbers are correct without checking.
Type:	Boolean
Optionality:	Optional (default used if not set)
Allowed:	true, false
Default:	true
Notes:	
Example:	`checkReservationVolumeResponse = true`

Creating Control Plans

Creating service control plans

Service control plans are created using the ACS Control Plan Editor. The MM CPE nodes are described in the XMS feature nodes chapters in *Feature Nodes Reference Guide*, however for full details see *CPE User's Guide*.

Messaging Manager allows messages to be routed to ACS for more complicated processing than Messaging Manager alone may achieve. To use this feature, all ACS service control plans to be used for processing of Messaging Manager messages must be owned by the ACS super user (that is, the ACS level 7 user), and the control plan versioning feature must be turned off. See *ACS User's Guide* and *CPE User's Guide* for details.

Configuring Messaging Manager to Send SMPP Parameters in Notifications

Setting SMPP Parameters for Notifications from a Control Plan

You can use temporary storage tags to set parameters in MAP or SMPP messages sent in notifications from a control plan. For example, to set the value of the protocol_id field in a MAP or SMPP notification message, set the MMX Protocol Identifier temporary storage tag by using a feature node such as the Set feature node or the Copy feature node in your control plan. You send the notification by placing a notifications feature node in the control plan after the Set or Copy feature node. See *Notifications Feature Nodes Table* (on page 149) for a list of notifications feature nodes that you can use.

You can also define configuration in the XMS section of the **eserv.config** file to copy temporary storage tags, byte for byte, into information elements in the user data header for MAP or SMPP messages.

Note: This feature applies to notifications sent by feature nodes that use a notifications processor, such as the XMS Send SMS Notification feature node, or the ACS Send Notification feature node. It does not apply to network originated messages modified by ACS control plans.

For more information about Convergent Charging Controller feature nodes, see *Feature Nodes Reference Guide*.

Notifications Feature Nodes Table

This table lists the feature nodes that can be used to send SMPP messages, and the palette group and Convergent Charging Controller application for each feature node. For more information about Convergent Charging Controller feature nodes, see *Feature Nodes Reference Guide*.

Note: You can use the Short Message Charging (SMCB) feature node to charge for messages you send.

Feature Node	Palette Group	Application
Send Notification	Interaction	ACS
Send Short Message	Interaction	ACS
Credit Wallet Transfer	CCS Charging	CCS
Friends And Destination Configuration	CCS Subscriber	CCS
Friends And Family Configuration	CCS Subscriber	CCS
Account Status SMS	CCS Wallet	CCS
SMS Low Balance	CCS Wallet	CCS
Call Information SMS	CCS Charging	CCS
Universal Attempt Termination with Billing	CCS Charging	CCS
Send Short Message Notification	XMS Control	Messaging Manager

Temporary Storage Tag Configuration

You configure the temporary storage tags that will be copied into the user data header in MAP or SMPP messages in the XMS ServiceConfig section of the **eserv.config** file.

```
XMS = {
...
    ServiceConfig = {
        ParameterMappings = {
            UserDataHeader = [
                {
                    profileTag=tag_code
                    informationElementIdentifier=int
                }
                {
                    profileTag=tag_code
                    informationElementIdentifier=int
                }
                ...
            ]
        }
...
    }
```

ServiceConfig Parameters

Here are the `ServiceConfig` parameters.

`profileTag`

Syntax:	`profileTag=`*`tag_code`*
Description:	The number used to identify the profile tag to copy from temporary storage to the `informationElementIdentifier` in the user data header.
Type:	Integer
Optionality:	Required
Allowed:	A valid profile tag number
Notes:	For more information about profile tags, see *ACS User's Guide*.
Example:	`profileTag=3932482`

`informationElementIdentifier`

Syntax:	`informationElementIdentifier=`*`int`*
Description:	The information element in the user data header that `profileTag` is copied to.
Type:	Integer
Optionality:	Required
Example:	`informationElementIdentifier=5`

MMX Notification Standard Application Port Example

The following example configuration maps the profile tag 3932482 to information element ID 5 in the user data header in an SMPP notification:

```
ServiceConfig = {
    ParameterMappings = {
        UserDataHeader = [

            {
                profileTag=3932482
                informationElementIdentifier=5
            }
        ]
    }
}
```

The profile tag 3932482 (MMX Notification Standard Application Port) is defined as a SHORT (two bytes), and matches the standard two byte destination port encoding used by the information element ID 5 (application port) in SMPP notifications. A Set feature node is used to set the MMX Notification Standard Application Port profile tag value to 5498, and an SMS Send SMS Notification feature node, placed after the Set feature node in the control plan, is used to send the SMPP notification. When the message is sent, the value for the application port in the user data header will be 5498.

MMX Notification Standard Application Port Example

The following example configuration maps the profile tag 29020 to information element ID 5 in the user data header in an SMPP notification:

```
ServiceConfig = {
    ParameterMappings = {
        UserDataHeader = [

            {
                profileTag=29020
                informationElementIdentifier=5
```

```
                }
            ]
        }
    }
```

In this example the custom profile tag 29020 is configured to hold an unsigned integer (four bytes), and the customer wants to set the application port in information element ID 5 to be four bytes containing 5498 in the first two bytes, and also containing 5498 in the third and fourth bytes. The customer uses a Set feature node to set profile tag 29020 to 360322426. Because 5498 is 0x157A in hexadecimal, and 360322426 is 0x157A157A, this means that the first two bytes of information element ID 5 will contain 5498, and the third and fourth bytes will also contain 5498.

Configuring Messaging Manager to load ACS Control Plans

Configuring MMX to load ACS Control Plans

Messaging Manager uses ACS control plans to perform advanced call processing. These control plans must be loaded into cache in order to be used. The following section of the configuration file sets the parameters for this.

serviceLibrary configuration

Here is example of the configuration of the `serviceLibrary` section in the **eserv.config**.

```
serviceLibrary = {

    validityTime = 5
    flushTime = 60
    maxAge = 3600

    DialledNumberAvailable = false

    xmsUndoNumTranslation = false
}
```

Note

The configuration parameters, `validityTime`, `flushTime` and `maxAge` apply only to caching of control plans which are matched by the Originator and Destination addresses.

Control plans that are matched purely by NAME of the control plan in the ACS triggering rules are actually stored in the ACS Global Cache. The validity period on that cache is controlled by the `globalProfileMaxAge` parameter in the **acs.conf** file. Please be aware that the default value for `globalProfileMaxAge` is 300 seconds, which is considerably longer than the validity period for the control plans in the Orig/Dest matching cache.

Trigger rules are defined in the GUI. Refer to *MM User's Guide*.

Where the matched ACS triggering rule does not specify a control plan name, the originating (or terminating) number is looked up in the CLI or Service Number tables within ACS to find the required control plan, and the values in the `serviceLibrary` section of the **eserv.config** are used.

serviceLibrary parameters

Here are the `serviceLibrary` parameters.

DialledNumberAvailable

Syntax:	`DialledNumberAvailable = true|false`
Description:	Retrieves the dialed number from the Original-Originating-Address
Type:	Boolean
Optionality:	Optional
Allowed:	true, false
Default:	false
Notes:	If true, retrieve the dialed number from the Original-Originating-Address. If false, do not retrieve the dialed number. Only used for IS-41.
Example:	`DialledNumberAvailable = true`

flushTime

Syntax:	`flushTime = seconds`
Description:	How often is a check made for data older than its validity time.
Type:	Positive integer
Optionality:	Optional
Allowed:	
Default:	60
Notes:	Applies to control plans matched on originator or destination addresses only.
Example:	`flushTime = 90`

maxAge

Syntax:	maxAge `= seconds`
Description:	The time after which an unused or unchanged control plan is dropped from the control plan cache.
Type:	Positive integer
Optionality:	Optional
Allowed:	
Default:	3600
Notes:	Applies to control plans matched on originator or destination addresses only.
Example:	`maxAge = 1800`

validityTime

Syntax:	`validityTime = seconds`
Description:	The maximum age of the data before it is refreshed from the database.
Type:	Positive integer
Optionality:	Optional
Allowed:	Any positive integer
Default:	5
Notes:	Applies to control plans matched on originator or destination addresses only.
Example:	`validityTime = 15`

`xmsUndoNumTranslation`

Syntax:	`xmsUndoNumTranslation = true|false`
Description:	Indicates when processing a long held reservation, whether or not the service library will undo hash and star number mapping to regain the original dialed numbers.
Type:	Boolean
Optionality:	Optional
Allowed:	true, false
Default:	false
Notes:	If true, undo hash and star number mapping to regain the original dialed numbers. If false, do not undo hash and star number mapping to regain the original dialed numbers.
Example:	`xmsUndoNumTranslation = true`

Chapter 5

Configuring Messaging Manager Services

Overview

Introduction

This chapter explains the steps required to configure several standard services.

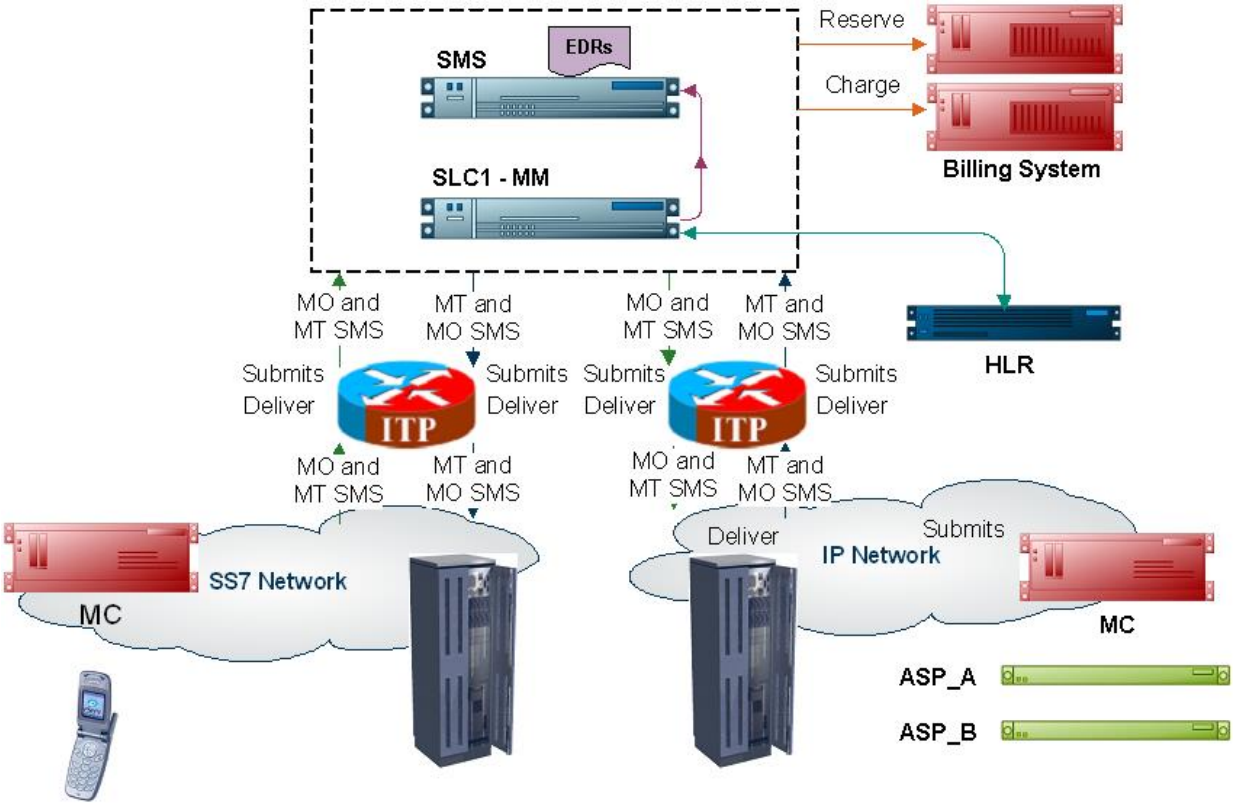
Note: The steps shown for the configuring of each service give the minimum that is required to be configured to achieve the service.

In this chapter

This chapter contains the following topics.

Example network

For the purposes of this document let us pretend that the network entities that are to be configured to use Messaging Manager are as follows:



These details will be used throughout the document as we work through examples of setting up and configuring Messaging Manager to perform standard services.

Mobile to SMSC Messaging

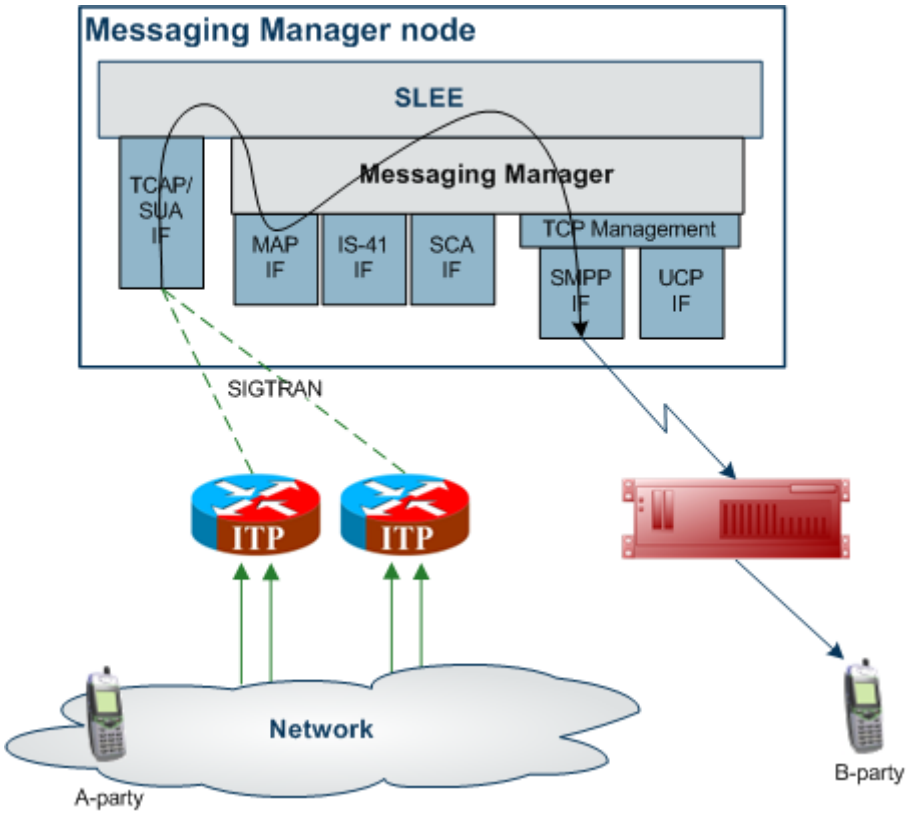
Description

The MO SMS service is a basic service that provides delivery to an SMSC of SMSs that originated from a mobile phone.

MO SMS and MT SMS are services that validate customers' requests to send and receive short messages through mobile phones. MO SMS is a service that allows customers to send short messages through a mobile phone. MT SMS is a service that allows customers to receive short messages via a mobile phone. Using Messaging Manager to provide these services ensures the collection of meaningful statistical information and customer usage records.

MO SMS diagram

Using only the Messaging Manager base module, MM can be configured to provide an MO SMS service. In this example we will receive mobile originating MAP messages and deliver to the SMSC over TCP/IP using the SMPP protocol. The following diagram shows the modules required.



Editing the eserv.config

Using the *example network* (on page 156) follow the steps below to set up MM to do basic MO SMS and MT SMS.

Step	Action
1	Take a copy of the example **eserv.config** provided at install of MM.
2	Edit the **eserv.config** file to have only two adapters, a MAP adapter and an SMPP adapter.
	Note: You can place a # at the beginning of each unwanted line.
3	Configure the MAP and SMPP adapters.
4	In the xmsTrigger area of the configuration file, configure the following features, if required: • *Collecting statistics* (on page 70) • *Generating CDRs* (on page 64)
5	Rename the **eserv.config** that is in use to a known name.
6	Rename the copy of the **eserv.config** that you have made all your changes to, to be **eserv.config**.
7	Reread the configuration file. See *Rereading the* **eserv.config** *file* (on page 28) for details.

Configuring Messaging Manager

Here is the process used to configure the service.

The following steps are carried out using the Messaging Manager screens. See *MM User's Guide*.

Stage	Description
1	Create adapters.
2	Create nodes.
3	Create schemes.
4	Create Message Center.
5	Open scheme.
6	Configure adapters.
7	Configure paths.
8	Configure connections
9	Configure domain.
10	Configure throttling.
11	Configure triggering.
12	Configure routing.

Application to Mobile Messaging

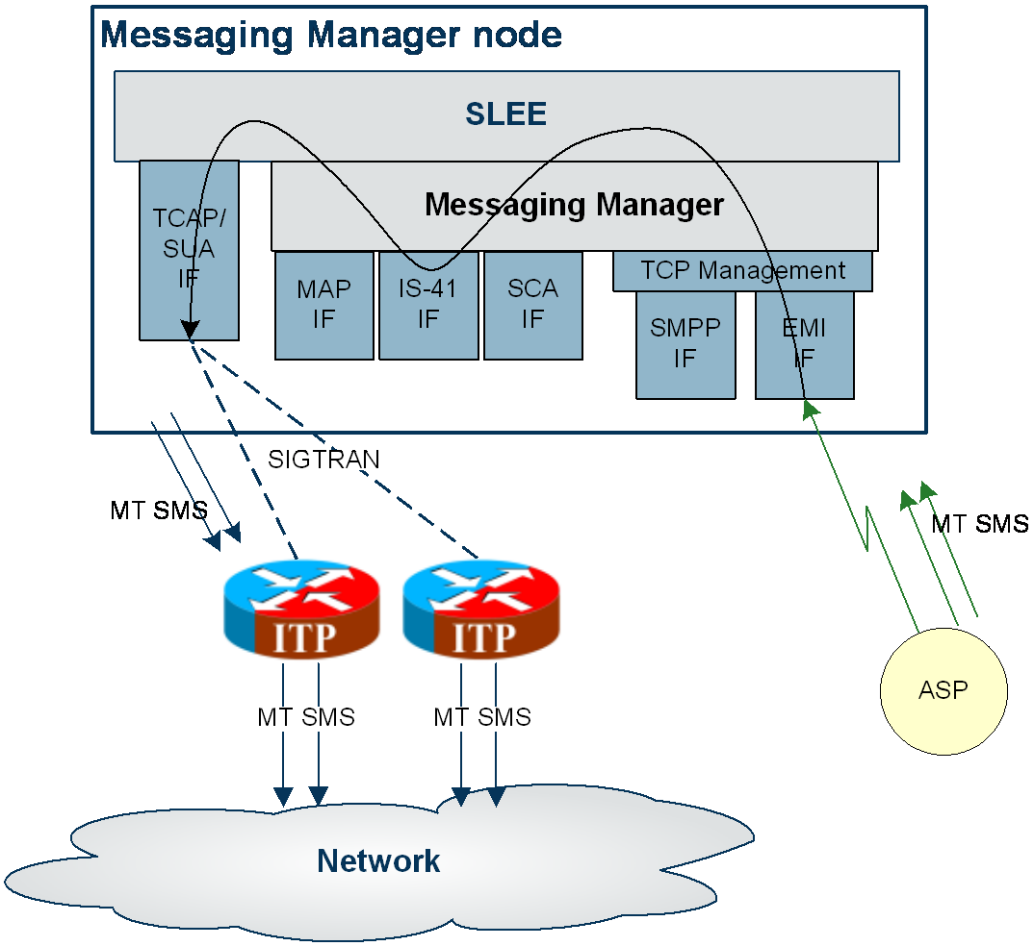
Description

Application to mobile is a simple service that provides basic delivery of SMSs to a mobile phone.

Application to Mobile diagram

Using only the Messaging Manager Base module, MM can be configured to provide an application to mobile service. In this example we will receive EMI protocol messages from ASPs and deliver them to an SMSC over SS7 using the IS-41 protocol.

The following diagram shows the modules required:



Editing the eserv.config

Using the *example network* (on page 156) follow the steps below to set up MM to do basic MT SMS.

Step	Action
1	Take a copy of the example **eserv.config** provided at install of MM.
2	Edit the **eserv.config** file to have only two adapters, an IS41 adapter and an EMI adapter.
	Note: You can place a # (comment tag) at the beginning of each unwanted line.
3	Configure the IS41 and EMI adapters.
4	In the `xmsTrigger` area of the configuration file, configure the following features, if required: • *Collecting statistics* (on page 70) • *Generating CDRs* (on page 64)
5	Rename the **eserv.config** that is in use to a known name.
6	Rename the copy of the **eserv.config** that you have made all your changes to, to be **eserv.config**.
7	Reread the configuration file. See *Rereading the* **eserv.config** *file* (on page 28) for details.

Configuring Messaging Manager

Here is the process used to configure the service.

The following steps are carried out using the Messaging Manager screens. See *MM User's Guide*.

Stage	Description
1	Create adapters.
2	Create nodes.
3	Create schemes.
4	Create Message Center.
5	Open scheme.
6	Configure adapters.
7	Configure paths.
8	Configure connections
9	Configure domain.
10	Configure throttling.
11	Configure triggering.
12	Configure routing.

Mobile to Application Messaging

Description

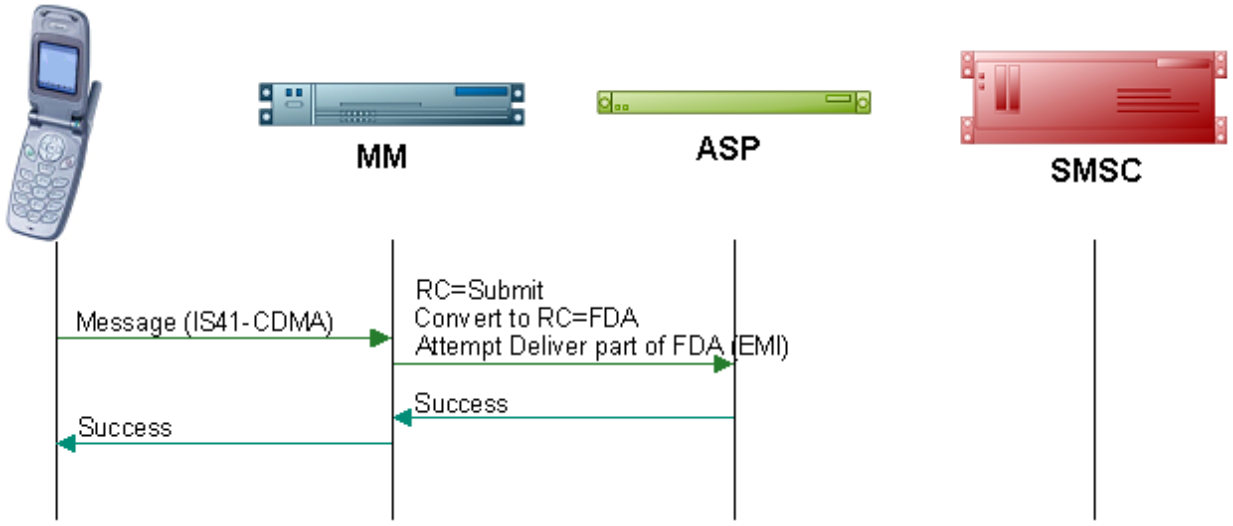
The mobile to application to service provides delivery of SMSs that originated on a mobile phone to an ASP.

Messaging Manager will attempt to deliver the SMS directly to the ASP, falling back to an SMSC if the direct delivery (FDA) fails.

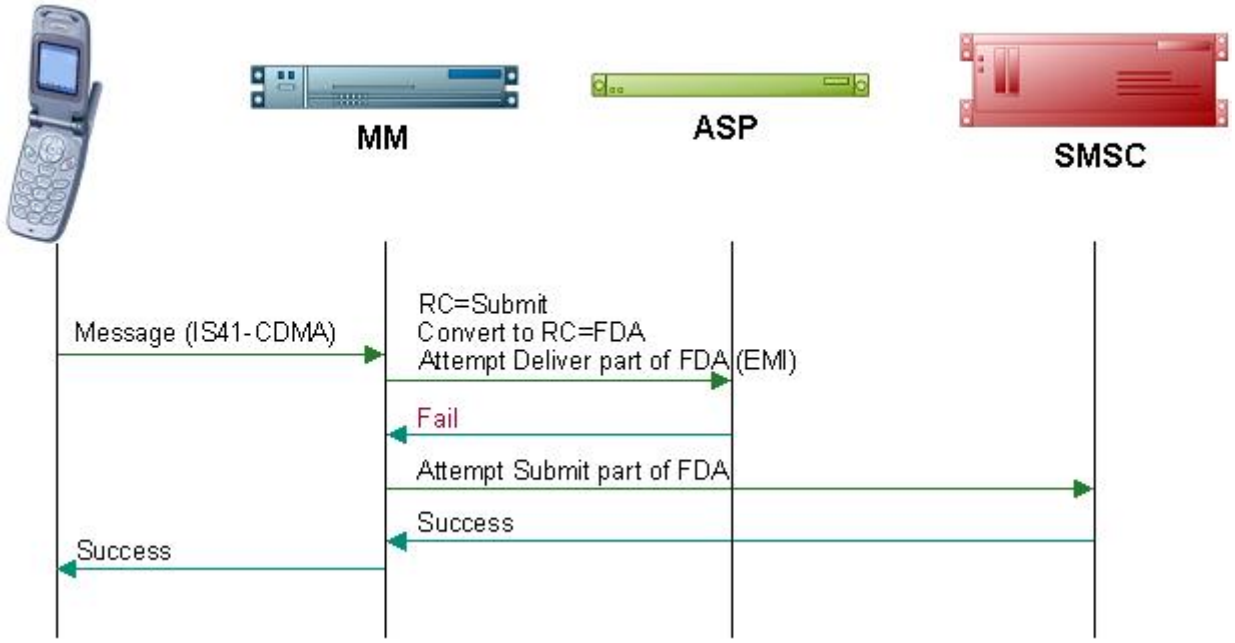
Setting the scene

This example covers the configuration required for the following flows:

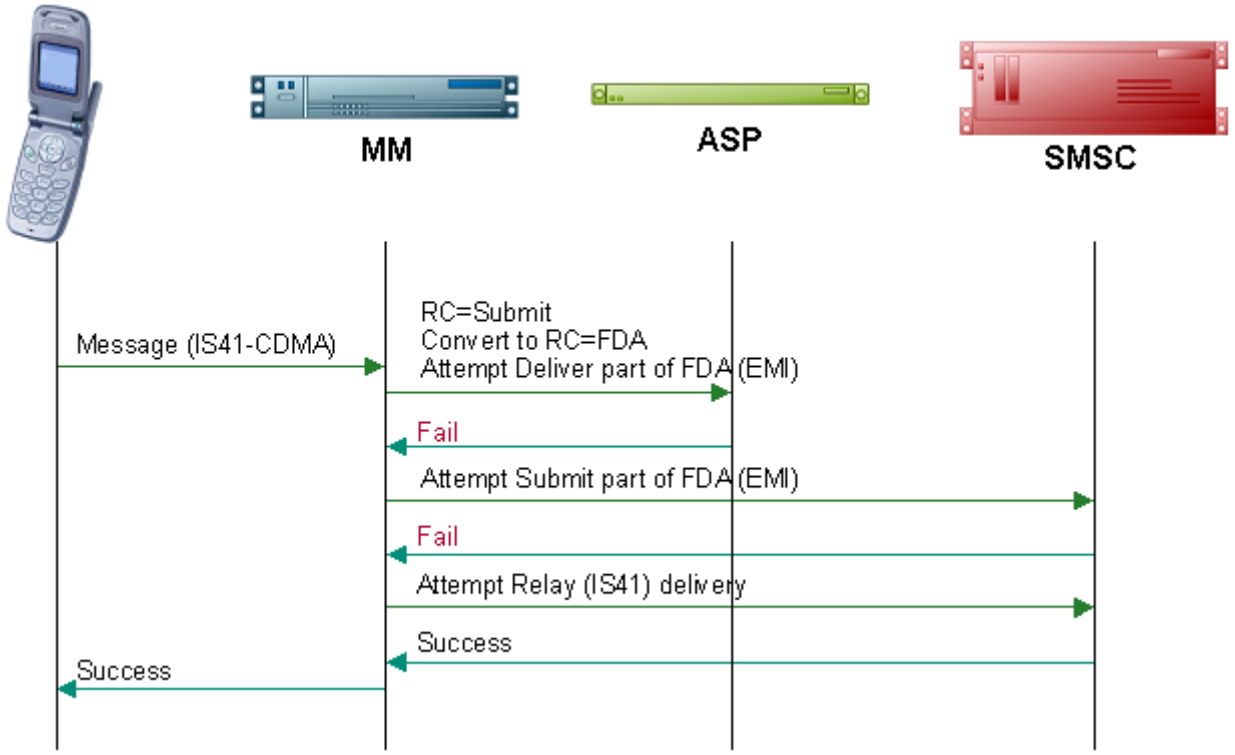
FDA deliver attempt success



FDA submit attempt success

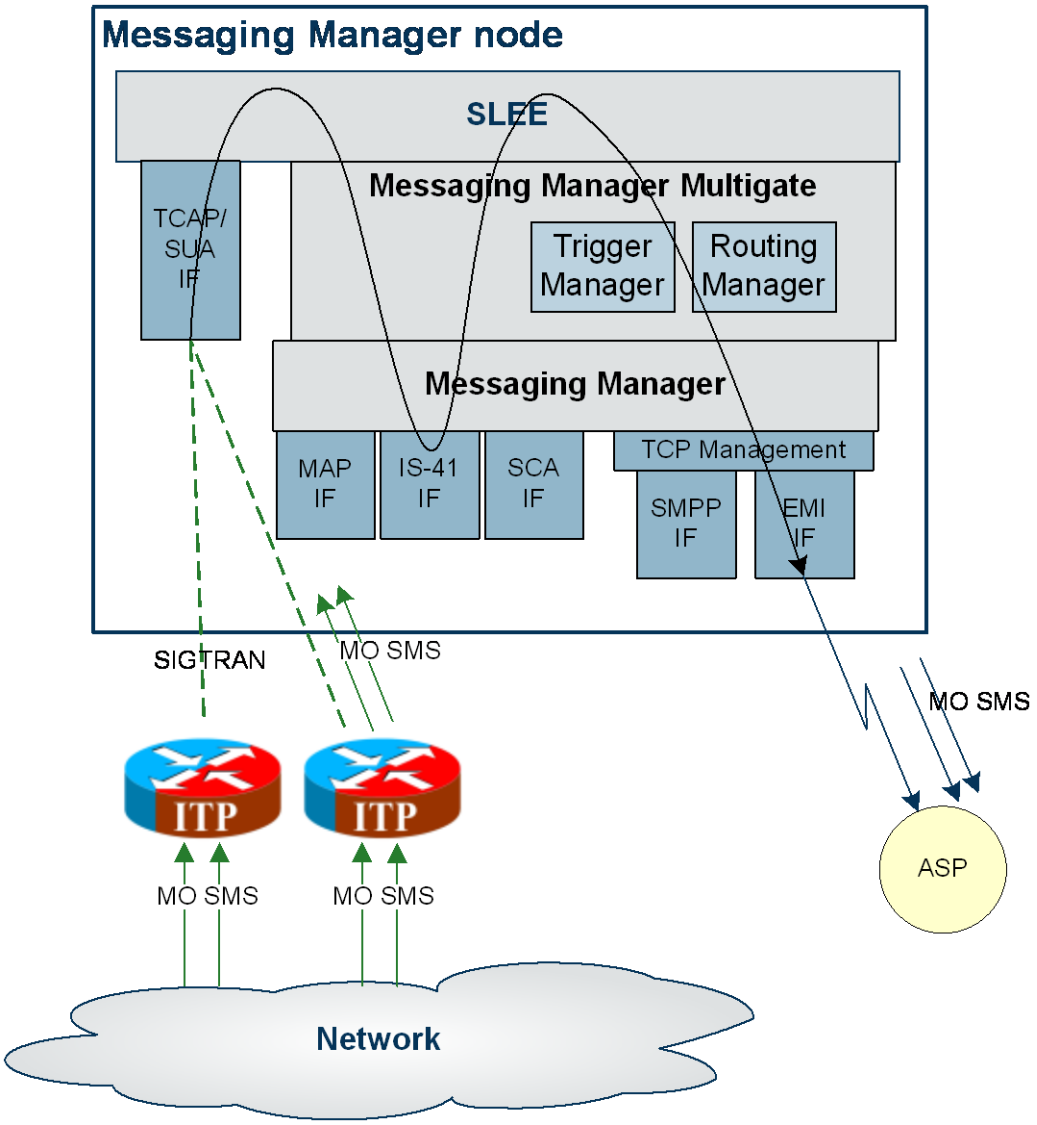


Default routing path attempt success



Mobile to Application diagram

Using the Messaging Manager Multigate and the Messaging Manager Director modules, MM can be configured to provide a Mobile to Application service. In this example we will receive mobile originating IS-41 messages and deliver them to ASPs over EMI using FDA. This diagram shows the modules required.



Editing the eserv.config

Using the *example network* (on page 156) follow the steps below to set up MM to do MO SMS to ASP with First Delivery Attempt.

Step	Action
1	Take a copy of the example **eserv.config** provided at install of MM.
2	Edit the **eserv.config** file to have only two adapters, an IS41 (CDMA) adapter and an EMI adapter.
	Note: You can place a # (comment tag) at the beginning of each unwanted line.
3	Configure the IS41 (CDMA) and EMI adapters.

Step	Action
4	In the `xmsTrigger` area of the configuration file, configure the following features, if required: • *Collecting statistics* (on page 70) • *Generating CDRs* (on page 64)
5	Rename the **eserv.config** that is in use to a known name.
6	Rename the copy of the **eserv.config** that you have made all your changes to, to be **eserv.config**.
7	Reread the configuration file. See *Rereading the* **eserv.config** *file* (on page 28) for details.

Configuring Messaging Manager

Here is the process used to configure the service.

The following steps are carried out using the Messaging Manager screens. See *MM User's Guide*.

Stage	Description
1	Create adapters.
2	Create nodes.
3	Create schemes.
4	Create Message Center.
5	Open scheme.
6	Configure adapters.
7	Configure paths.
8	Configure connections
9	Configure domain.
10	Configure throttling.
11	Configure triggering.
12	Configure routing.

Mobile to Mobile triggering to ACS

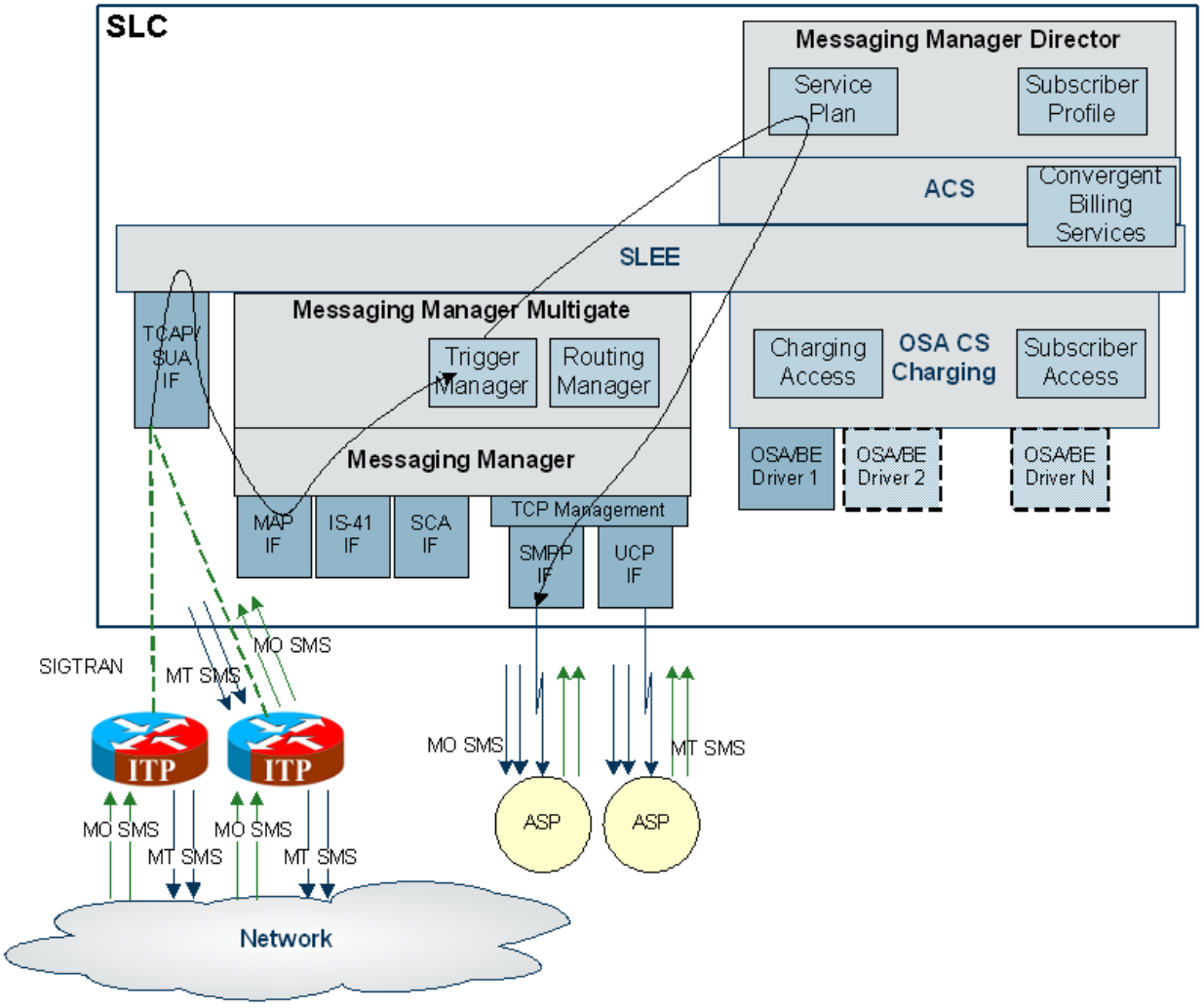
Description

The Mobile to Mobile messaging (MO SMS) service allows customers to send short messages from one mobile phone to another.

In this example messages will be triggered to an ACS control plan to route large messages to a specific SMSC.

Diagram

Using the Messaging Manager Multigate and the Messaging Manager Director modules, MM can be configured to provide a Mobile to Mobile service, triggering to ACS. In this example we will receive mobile originating MAP messages and deliver them to SMSCs over MAP having triggered them to ACS to offload all large messages to a separate SMSC. The following diagram shows the modules required:



Editing the eserv.config

Using the *example network* (on page 156) follow the steps below to set up MM to do basic MO SMS and MT SMS.

Step	Action
1	Take a copy of the example **eserv.config** provided at install of MM.
2	Edit the **eserv.config** file to have only two adapters, a MAP adapter and an SMPP adapter. **Note:** You can place a # at the beginning of each unwanted line.
3	Configure the MAP and SMPP adapters.
4	In the `xmsTrigger` area of the configuration file, configure the following features, if required: • *Collecting statistics* (on page 70)

Step	Action
	• *Generating CDRs* (on page 64)
5	Rename the **eserv.config** that is in use to a known name.
6	Rename the copy of the **eserv.config** that you have made all your changes to, to be **eserv.config**.
7	Reread the configuration file. See *Rereading the* **eserv.config** *file* (on page 28) for details.

Configuring Messaging Manager

Here is the process used to configure the service.

The following steps are carried out using the Messaging Manager screens. See *MM User's Guide*.

Stage	Description
1	Create adapters.
2	Create nodes.
3	Create schemes.
4	Create Message Center.
5	Open scheme.
6	Configure adapters.
7	Configure paths.
8	Configure connections
9	Configure domain.
10	Configure throttling.
11	Configure triggering.
12	Configure routing.

Chapter 6

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.

xmsTrigger Application

Purpose

xmsTrigger (also known as the XMS Trigger application) is a SLEE interface which is responsible for:

- Correlating and controlling all of the incoming short messages
- Converting (through adapters) and relaying messages between different SLEE interfaces
- Triggering INAP service logic on slee_acs as required during message processing

The primary tasks of xmsTrigger are:

- Receive incoming Send requests from an MM interface
- Trigger INAP to the ACS service (slee_acs)
- Possibly pass outgoing send requests to an MM interface for relay
- Receive delivery notifications from an MM interface
- Trigger INAP to the ACS service (slee_acs)
- Possibly pass delivery notifications to an MM interface for relay

Stopping and starting xmsTrigger (XMS)

xmsTrigger is a SLEE interface. Like other SLEE interfaces, it is stopped and started through the SLEE administration scripts. xmsTrigger is restarted when the SLEE is restarted.

For more information about starting and stopping the SLEE, see *SLEE Technical Guide*.

Configuration

xmsTrigger supports extensive configuration from **eserv.config**. For more information, see **eserv.config** *Configuration* (on page 27).

xmsAgent

Purpose

xmsAgent is a proxy gateway for the SMPP adapter. It will proxy traffic from a single remote ASP/MC to multiple xmsTrigger instances.

Stopping and starting xmsAgent

xmsAgent is a SLEE interface. Like other SLEE interfaces, it is stopped and started via the SLEE administration scripts. xmsAgent is restarted when the SLEE is restarted.

For more information about starting and stopping the SLEE, see *SLEE Technical Guide*.

Adapters

Adapter interface location

Adapter interfaces are supplied as libraries. These are located in **/IN/service_packages/XMS/lib**.

Adapter interfaces are libraries loaded by the xmsTrigger application and are started and accessed by this application as required.

Available adapters

There are several adapters available from Oracle, listed below.

Protocol	adapter
EMI	mmxiEMI.so
SCA	mmxiSCA.so
SMPP	mmxiSMPP.so
IS-41 (CDMA)	xmsiIS41.so
IS-41 (TDMA)	xmsiIS41.so
MAP	xmsiMap.so
Wrapper	xmsiWrapper.so

Statistics

Introduction

Messaging Manager statistics are generated by each SLC, and then transferred at periodic intervals to the Service Management System (SMS) for permanent storage and analysis. MM allows new statistics to be registered on the system without the need for manual intervention.

An existing statistics system provides functions for the collection of basic statistical events. This is provided in the Oracle SMS application. For more information about the SMS statistics subsystem, see *SMS User's Guide*.

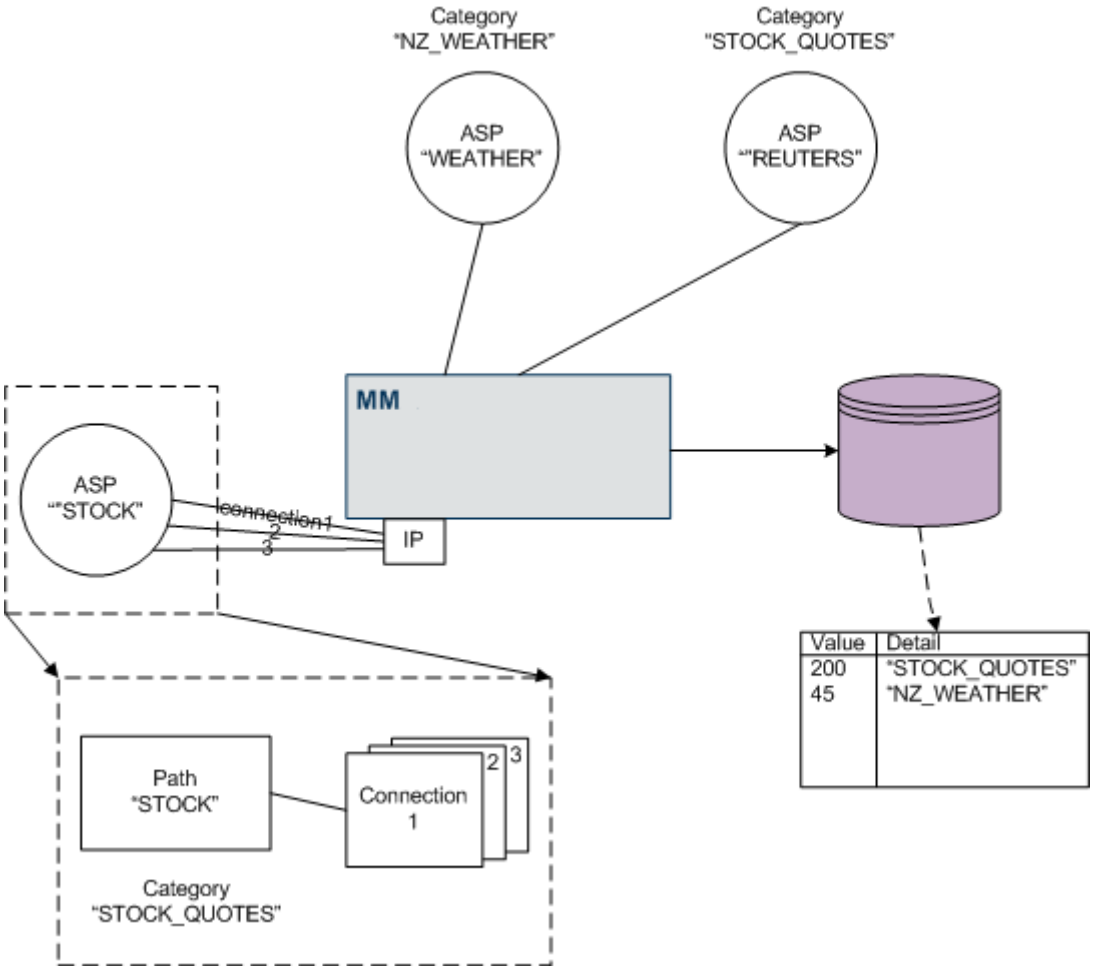
Setting up statistics collection

Paths and categories used for gathering statistics against are defined using the Messaging Manager Manager screens. Refer to *MM User's Guide*.

To turn on statistics gathering, see *Collecting Statistics* (on page 70).

Diagram

Here is an example of statistics collection. There are three ASPs. However, the Category "STOCK_QUOTES" has been applied to both ASP "STOCK" and ASP "REUTERS", so in this instance, the statistics for both ASPs will be counted and added to the database table.



Inbound message types

Every message coming into MM is one of the following transaction types. On receipt of any message, MM will increment the statistic listed below, with the inbound path's statistics category in the detail field.

Inbound Message Type	Statistic
Command	MSG_COMMAND
Submit	MSG_SUBMIT
Deliver	MSG_DELIVER
Notify	MSG_NOTIFY

Inbound Message Type	Statistic
RouteInfo	MSG_ROUTEINFO

Note: Command type messages are immediately relayed and no further processing is done. The only further statistic that can be recorded for them is to indicate a successful relay.

When the recipient acknowledges receipt of a relayed command message, MM will increment the MSG_COMMAND_SUCCESS statistic, with the inbound path's statistics category in the detail field.

For detailed information on transaction types, refer to *MM User's Guide*.

Result statistics

For all message types, except Command, a corresponding overall result statistic will also be incremented at some subsequent point in the processing.

Outcome	Result Statistic
Failed screening	FAIL_SCREENING
Throttled	FAIL_THROTTLED
Control plan accept	SUCCESS
Control plan reject / reject action	FAIL_REJECT
Successful delivery	SUCCESS
Delivery failure	FAIL_ROUTING
Inbound timeout	FAIL_TIMEOUT
Inbound abort	FAIL_ABORT

Note: The format of the statistic produced is MSG_*InboundMessageType_ResultStatistic*.

Example inbound message statistics

Here are some examples of the statistics incremented.

Example 1: When a Submit message is rejected due to any of the screening checks, MM will increment this statistic.
MSG_SUBMIT_FAIL_SCREENING

Example 2: When a Deliver message is rejected due to throttling, MM will increment this statistic.
MSG_DELIVER_FAIL_THROTTLED

Example 3: When a Deliver message is successfully delivered, MM will increment this statistic.
MSG_DELIVER_SUCCESS

Current message types

A control plan may be triggered based on the inbound message type. If direct delivery is attempted on a Submit message (that is, the action changed to “Deliver” or “FDA”), then two control plans (Submit and Deliver) may be triggered. A new Deliver message is created for the delivery attempt, so at this point the “current message” is a Deliver message and the “inbound message” is a Submit message.

This table describes the current message types.

Current Message Type	Statistic
Submit	MSG_SUBMIT
Deliver	MSG_DELIVER
Notify	MSG_NOTIFY

Control Plans result statistics

For all current message types, a result statistic will be incremented.

Result Statistic	Cause
IDP_TRIGGERED	When a message triggers a control plan.
IDP_ROUTE	When a trigger rule or control plan specifies a Route action.
IDP_RELAY	When a trigger rule or control plan specifies a Relay action.
IDP_DISCARD	In either of the following cases: • A trigger rule specifies a Discard action • A control plan sends a ReleaseCall operation with the Discard release cause, without having done a Connect or Continue
IDP_ACCEPT	In either of the following cases: • A trigger rule specifies an Accept action • A control plan sends a ReleaseCall operation with the Accept release cause, without having done a Connect or Continue
IDP_REJECT	In either of the following cases: • A trigger rule specifies a Reject action • A control plan sends a ReleaseCall operation with any release cause other than Accept or Discard, without having done a Connect or Continue

Notes:

- The format of the statistic produced is MSG_*CurrentMessageType_ResultStatistic*.
- If the message is rejected, discarded or accepted by a trigger rule or control plan, no further processing is done and the appropriate statistic is updated to indicate the final result.

Example current message type statistics

Here are some examples of the statistics incremented.

Example 1: When a Submit message triggers a control plan, MM will increment this statistic:
MSG_SUBMIT_IDP_TRIGGERED

Example 2: When a trigger rule or control plan specifies a Route action for a Deliver message, MM will increment this statistic:
MSG_DELIVER_IDP_ROUTE

Example 3: When a trigger rule or control plan specifies a Relay action for a Notify message, MM will increment this statistic:
MSG_NOTIFY_IDP_RELAY

Example 4: When a trigger rule specifies a Discard action or a control plan sends a ReleaseCall operation with a Discard release cause and without having done a Connect or Continue, for a Deliver message, MM will increment this statistic:
MSG_DELIVER_IDP_DISCARD

Example 5: When a trigger rule specifies an Accept action or a control plan sends a ReleaseCall operation with a Accept release cause and without having done a Connect or Continue, for a Submit message, MM will increment this statistic:
MSG_SUBMIT_IDP_ACCEPT

Routing class statistics

The message will then proceed to routing with a routing class:

- Deliver
- FDA
- Submit
- Relay
- Locate

These are set from the message type default, trigger rule or control plan.

For detailed information on routing classes, refer to *MM User's Guide*.

If there is no routing rule matching the specified routing class, then a virtual routing class of 'Default' will be used for all MSG_ROUTING_* statistics
This table describes the MSG_ Routing statistics.

Statistic	Description
MSG_ROUTING_*routingClass*	When a message is passed to routing.
MSG_ROUTING_*routingClass*_SUCCESS	If a message is successfully delivered and the routing class is not “FDA”.
MSG_ROUTING_FDA_SUCCESS_DELIVER	If the routing class is “FDA” and the message is successfully delivered via a Deliver routing rule.
MSG_ROUTING_FDA_SUCCESS_SUBMIT	If the routing class is “FDA” and the message is successfully delivered via a Submit routing rule.

Once a message gets to the routing stage, the overall result will be either SUCCESS, FAIL_ROUTING or TIMEOUT, with the inbound path's statistics category in the detail field. See *Result statistics* (on page 170).

Deliver

If the routing class is “Deliver” or “FDA”, MM will look for routing rules that specify a routing class of “Deliver”. Each path listed in the routing rule will be attempted in turn, until either a successful or permanent failure response is received. Statistics are recorded for each path that delivery is attempted on, with the detail field being populated by the outbound path's statistics category. The term “attempt” includes any retries that are configured for the path. The outcome of a delivery attempt is one of:

Statistic	Description
MSG_PATH_DELIVER_RETRY_SUCCESS	Delivery initially fails, but is successful on the first or subsequent retries
MSG_PATH_DELIVER_TEMP_ERR	Initial delivery and all retries return a temporary error
MSG_PATH_DELIVER_PERM_ERR	Initial delivery or one of the retries returns a permanent error (so there are no further retries)
MSG_PATH_DELIVER_TIMEOUT	Initial delivery or one of the retries times out (so there are no further retries)
MSG_PATH_DELIVER_ATTEMPT	For every delivery attempt on a path listed in a Deliver routing rule, MM will increment this statistic, with the outbound path's statistics category in the detail field

<table>
<tr><th>Statistic</th><th colspan="2">Description</th></tr>
<tr><td>MSG_PATH_DELIVER_bucket_COUNT</td><td colspan="2">For every successful delivery attempt on a path listed in a Deliver routing rule, MM will increment this statistic, where the bucket is determined by the response time from the MSC with the outbound path's statistics category in the detail field.
Where bucket can be:</td></tr>
<tr><td></td><td>Bucket</td><td>Response Time</td></tr>
<tr><td></td><td>RT1</td><td><= 1 second</td></tr>
<tr><td></td><td>RT2</td><td>> 1 second
<= 5 seconds</td></tr>
<tr><td></td><td>RT3</td><td>> 5 seconds
<= 10 seconds</td></tr>
<tr><td></td><td>RTX</td><td>> 10 seconds</td></tr>
<tr><td>MSG_PATH_DELIVER_bucket_TIME</td><td colspan="2">For every successful delivery attempt on a path listed in a Deliver routing rule, MM will add the response time to this statistic, with the outbound path's statistics category in the detail field (where bucket is determined as above).</td></tr>
</table>

Note: The number of attempts that produce a successful delivery with no retries required can be derived by subtracting the other outcomes from MSG_PATH_DELIVER_ATTEMPT.

Submit

If the routing class is “Submit” or “FDA” (where the “Deliver” leg has already failed), MM will look for routing rules that specify a routing class of “Submit”. Each path listed in the routing rule will be attempted in turn, until either a successful or permanent failure response is received.

Statistic	Description
MSG_PATH_SUBMIT_ATTEMPT	For every delivery attempt on a path listed in a Submit routing rule.
MSG_PATH_SUBMIT_TIMEOUT	For every delivery attempt that times out on a path listed in a Submit routing rule.
MSG_PATH_SUBMIT_TEMP_ERR	For every delivery attempt resulting in a transient failure, on a path listed in a Submit routing rule.
MSG_PATH_SUBMIT_PERM_ERR	For every delivery attempt resulting in a permanent failure, on a path listed in a Submit routing rule.

Note: All of these statistics will include the outbound path's statistics category in the detail field.

Locate

If the routing class is “Locate”, MM will look for routing rules that specify a routing class of “Locate”. Each path listed in the routing rule will be attempted in turn, until either a successful or permanent failure response is received.

Statistic	Description
MSG_PATH_LOCATE_ATTEMPT	For every delivery attempt on a path listed in a Locate routing rule.

Statistic	Description
MSG_PATH_LOCATE_TIMEOUT	For every delivery attempt that times out on a path listed in a Locate routing rule.
MSG_PATH_LOCATE_TEMP_ERR	For every delivery attempt resulting in a transient failure, on a path listed in a Locate routing rule.
MSG_PATH_LOCATE_PERM_ERR	For every delivery attempt resulting in a permanent failure, on a path listed in a Locate routing rule.

Note: All of these statistics will include the outbound path's statistics category in the detail field.

Other statistics

This table describes the function of each field.

Field	Description
MSG_*inboundMsgType*_FAIL_TIMEOUT	If the inbound transaction's timeout is exceeded before sending a result, MM will increment this statistic, with the inbound path's statistics category in the detail field. No other MSG_*inboundMsgType_result* statistics will be incremented in this case.
MSG_*inboundMsType*_FAIL_ABORT	If the sending entity aborts the inbound transaction, MM will increment this statistic, with the inbound path's statistics category in the detail field. No other MSG_*inboundMsgType_result* statistics will be incremented in this case.

Delivery receipt

When MM generates a delivery receipt (having delivered or failed to deliver an SMS directly), statistics will be recorded in a way that is consistent with the procedure for handling an external delivery receipt. It is intended that in future the same processing logic will be used for internally generated delivery receipts as for external ones.

Whenever it generates a delivery receipt, MM will increment the following statistics.

Statistic	Note
MSG_NOTIFY	Detail field is INTERNAL_DR path's statistics category.
MSG_ROUTING_DELIVER	
MSG_PATH_DELIVER_ATTEMPT	Detail field is Inbound path's statistics category
MSG_NOTIFY_SUCCESS	Detail field is INTERNAL_DR path's statistics category
MSG_NOTIFY_FAIL_ROUTING	NA
MSG_ROUTING_DELIVER_SUCCESS	NA
MSG_PATH_DELIVER_*bucket*_COUNT	Detail field is inbound path's statistics category. See Routing class statistics - *Deliver* (on page 172).
MSG_PATH_DELIVER_*bucket*_TIME	Detail field is inbound path's statistics category. See Routing class statistics - *Deliver* (on page 172).
MSG_PATH_DELIVER_*result*	Where *result* can be one of:

Statistic	Note
	• TIMEOUT • TEMP_ERR • PERM_ERR See routing class statistics - *Deliver* (on page 172).

Example statistics printout

Here is an example of a list of statistics produced while running MM.

```
Thu Sep  7 22:45:33 2006
Rims_Service:                          MSG_GET_ROUTING_INFO = 1484
Rims_Service:                       MSG_IS41_SMSREQ_ATTEMPT = 113
Rims_Service:                      MSG_IS41_SMSREQ_PERM_ERR = 5
Mmx_Service:                              MSG_ROUTING_FDA = 1344
Rims_Service:                  MSG_GET_ROUTING_INFO_SUCCESS = 1479
Mmx_Service:                           MSG_ROUTING_LOCATE = 74
Mmx_Service:                MSG_ROUTING_FDA_SUCCESS_SUBMIT = 21
Mmx_Service:                           MSG_ROUTING_Default = 43
Mmx_Service:                   MSG_ROUTING_Default_SUCCESS = 35
Mmx_Service:                           MSG_ROUTING_SUBMIT = 40701
Mmx_Service:                      MSG_PATH_SUBMIT_ATTEMPT = 40693
Mmx_Service:                      MSG_PATH_LOCATE_ATTEMPT = 63
Mmx_Service:                     MSG_PATH_DELIVER_TIMEOUT = 841
Mmx_Service:                    MSG_PATH_DELIVER_TEMP_ERR = 931
Mmx_Service:                    MSG_PATH_DELIVER_RT1_TIME = 103275
Mmx_Service:                   MSG_PATH_DELIVER_RT2_COUNT = 67
Mmx_Service:                     MSG_PATH_SUBMIT_PERM_ERR = 1
Mmx_Service:                    MSG_PATH_DELIVER_RT2_TIME = 192103
Mmx_Service:                   MSG_PATH_DELIVER_RT3_COUNT = 55
Mmx_Service:                   MSG_PATH_DELIVER_RTX_COUNT = 147
Mmx_Service:                                MSG_DELIVER = 81
Mmx_Service:                                 MSG_NOTIFY = 153
Mmx_Service:                      MSG_PATH_SUBMIT_TIMEOUT = 26
Acs_Service:                                  CALLS_AT = 121
Mmx_Service:                               MSG_ROUTEINFO = 63
Mmx_Service:                           MSG_SUBMIT_SUCCESS = 2637
Mmx_Service:                    MSG_SUBMIT_FAIL_THROTTLED = 10168
Mmx_Service:                       MSG_SUBMIT_FAIL_REJECT = 24
Mmx_Service:                      MSG_SUBMIT_FAIL_ROUTING = 24133
Mmx_Service:                                 MSG_SUBMIT = 52511
Mmx_Service:                        MSG_SUBMIT_FAIL_ABORT = 6798
Mmx_Service:                    MSG_PATH_DELIVER_RT1_COUNT = 256
Mmx_Service:                       MSG_SUBMIT_FAIL_TIMEOUT = 559
Mmx_Service:                        MSG_DELIVER_FAIL_ABORT = 3
Mmx_Service:                         MSG_ROUTEINFO_SUCCESS = 60
Mmx_Service:                           MSG_ROUTING_DELIVER = 279
Mmx_Service:                   MSG_ROUTEINFO_FAIL_ROUTING = 3
Mmx_Service:                      MSG_NOTIFY_FAIL_ROUTING = 3
Mmx_Service:                        MSG_NOTIFY_FAIL_ABORT = 17
Rims_Service:                    MSG_GET_ROUTING_INFO_FAIL = 5
Mmx_Service:                     MSG_PATH_DELIVER_RT3_TIME = 362381
Mmx_Service:                          MSG_SUBMIT_IDP_ROUTE = 42187
Mmx_Service:                MSG_ROUTING_FDA_SUCCESS_DELIVER = 484
Acs_Service:                             CALLS_DISCONNECTED = 479
Mmx_Service:                      MSG_DELIVER_FAIL_REJECT = 7
Mmx_Service:                        MSG_SUBMIT_IDP_ACCEPT = 343
Mmx_Service:                        MSG_SUBMIT_IDP_REJECT = 22
Mmx_Service:                       MSG_NOTIFY_FAIL_REJECT = 6
Mmx_Service:                    MSG_SUBMIT_FAIL_SCREENING = 2
Mmx_Service:                        MSG_DELIVER_IDP_ROUTE = 1419
Mmx_Service:                    MSG_NOTIFY_IDP_TRIGGERED = 7
Mmx_Service:                    MSG_DELIVER_FAIL_ROUTING = 4
Mmx_Service:                    MSG_DELIVER_IDP_TRIGGERED = 6
Mmx_Service:                     MSG_SUBMIT_IDP_TRIGGERED = 469
```

```
Mmx_Service:                          MSG_ROUTEINFO_IDP_ROUTE = 74
Mmx_Service:                               MSG_NOTIFY_SUCCESS = 31
Rims_Service:                            MSG_SET_ROUTING_INFO = 23
Mmx_Service:                       MSG_PATH_DELIVER_RTX_TIME = 3302637
Mmx_Service:                      MSG_ROUTING_LOCATE_SUCCESS = 69
Mmx_Service:                        MSG_PATH_DELIVER_ATTEMPT = 1566
Rims_Service:                         MSG_MAP_SRI_SM_ATTEMPT = 41
Mmx_Service:                             MSG_DELIVER_SUCCESS = 22
Mmx_Service:                        MSG_PATH_LOCATE_TEMP_ERR = 3
Mmx_Service:                        MSG_PATH_SUBMIT_TEMP_ERR = 25725
Mmx_Service:                            MSG_NOTIFY_IDP_ROUTE = 69
Mmx_Service:                      MSG_ROUTING_SUBMIT_SUCCESS = 1990
Mmx_Service:                     MSG_ROUTING_DELIVER_SUCCESS = 110
Mmx_Service:MSG_SUBMIT:                                EMI_ASP1 = 23
Mmx_Service:MSG_SUBMIT:                       INTERNAL_SME_Wrapper = 2
Mmx_Service:MSG_PATH_DELIVER_ATTEMPT:               MAP_SME_Conn1 = 16
Mmx_Service:MSG_SUBMIT_SUCCESS:                        EMI_ASP1 = 23
Mmx_Service:MSG_PATH_DELIVER_RT1_COUNT:             MAP_SME_Conn1 = 13
Mmx_Service:MSG_PATH_DELIVER_RT1_TIME:              MAP_SME_Conn1 = 5163
Mmx_Service:MSG_SUBMIT_SUCCESS:               INTERNAL_SME_Wrapper = 2
Mmx_Service:MSG_SUBMIT:                                SMPP_ASP1 = 44
Mmx_Service:MSG_PATH_SUBMIT_ATTEMPT:                  SMPP_SMSC1 = 42
Mmx_Service:MSG_SUBMIT_SUCCESS:                        SMPP_ASP1 = 42
Mmx_Service:MSG_SUBMIT_FAIL_REJECT:                    SMPP_ASP1 = 2
Mmx_Service:MSG_NOTIFY:                               SMPP_SMSC1 = 41
Mmx_Service:MSG_PATH_DELIVER_ATTEMPT:                  SMPP_ASP1 = 41
Mmx_Service:MSG_PATH_DELIVER_RT1_COUNT:                SMPP_ASP1 = 41
Mmx_Service:MSG_PATH_DELIVER_RT1_TIME:                 SMPP_ASP1 = 7329
Mmx_Service:MSG_NOTIFY_SUCCESS:                       SMPP_SMSC1 = 41
Mmx_Service:MSG_ROUTEINFO:                      CDMA_Foreign_SMSC = 9
Mmx_Service:MSG_PATH_LOCATE_ATTEMPT:               CDMA_SME_Conn1 = 9
Mmx_Service:MSG_PATH_LOCATE_TEMP_ERR:              CDMA_SME_Conn1 = 2
Mmx_Service:MSG_ROUTEINFO_FAIL_ROUTING:         CDMA_Foreign_SMSC = 2
Mmx_Service:MSG_ROUTEINFO_SUCCESS:              CDMA_Foreign_SMSC = 7
Mmx_Service:MSG_PATH_SUBMIT_ATTEMPT:                    EMI_SMSC1 = 25
Mmx_Service:MSG_NOTIFY:                                 EMI_SMSC1 = 23
Mmx_Service:MSG_PATH_DELIVER_ATTEMPT:                    EMI_ASP1 = 21
Mmx_Service:MSG_PATH_DELIVER_RT1_COUNT:                  EMI_ASP1 = 21
Mmx_Service:MSG_PATH_DELIVER_RT1_TIME:                   EMI_ASP1 = 3827
Mmx_Service:MSG_NOTIFY_SUCCESS:                         EMI_SMSC1 = 23
Mmx_Service:MSG_SUBMIT:                              MAP_SME_Conn1 = 11
Mmx_Service:MSG_PATH_SUBMIT_ATTEMPT:                       MAP_MC1 = 6
Mmx_Service:MSG_SUBMIT_SUCCESS:                      MAP_SME_Conn1 = 9
Mmx_Service:MSG_SUBMIT:                              CDMA_SME_Conn1 = 15
Mmx_Service:MSG_PATH_SUBMIT_ATTEMPT:                       CDMA_MC1 = 10
Mmx_Service:MSG_SUBMIT_SUCCESS:                      CDMA_SME_Conn1 = 11
Mmx_Service:MSG_PATH_SUBMIT_TIMEOUT:                       CDMA_MC1 = 3
Mmx_Service:MSG_PATH_SUBMIT_TEMP_ERR:                      CDMA_MC1 = 4
Mmx_Service:MSG_SUBMIT_FAIL_REJECT:                  CDMA_SME_Conn1 = 4
Mmx_Service:MSG_PATH_SUBMIT_TEMP_ERR:                     EMI_SMSC1 = 2
Mmx_Service:MSG_SUBMIT_FAIL_TIMEOUT:                 MAP_SME_Conn1 = 2
Mmx_Service:MSG_ROUTEINFO:                        MAP_Foreign_SMSC = 5
Mmx_Service:MSG_PATH_LOCATE_ATTEMPT:                 MAP_SME_Conn1 = 5
Mmx_Service:MSG_ROUTEINFO_SUCCESS:                MAP_Foreign_SMSC = 5
Mmx_Service:MSG_DELIVER:                          MAP_Foreign_SMSC = 5
Mmx_Service:MSG_DELIVER_SUCCESS:                  MAP_Foreign_SMSC = 5
Mmx_Service:MSG_PATH_DELIVER_ATTEMPT:               CDMA_SME_Conn1 = 10
Mmx_Service:MSG_PATH_DELIVER_RT1_COUNT:             CDMA_SME_Conn1 = 6
Mmx_Service:MSG_PATH_DELIVER_RT1_TIME:              CDMA_SME_Conn1 = 5468
Mmx_Service:MSG_PATH_DELIVER_RT2_COUNT:              MAP_SME_Conn1 = 3
Mmx_Service:MSG_PATH_DELIVER_RT2_TIME:               MAP_SME_Conn1 = 4170
Mmx_Service:MSG_DELIVER:                        CDMA_Foreign_SMSC = 4
Mmx_Service:MSG_PATH_DELIVER_TEMP_ERR:             CDMA_SME_Conn1 = 4
Mmx_Service:MSG_DELIVER_FAIL_ROUTING:           CDMA_Foreign_SMSC = 2
Mmx_Service:MSG_DELIVER_SUCCESS:                CDMA_Foreign_SMSC = 2
```

Tracing

Introduction

The Messaging Manager tracing feature, when turned on, captures all the major decision points in an MM transaction. A trace identifier is generated by the originating adapter transaction and stored in the parent context at the start of a new call. When the terminating transaction is created it fetches the stored trace identifier from the parent context.

Once tracing has been turned on for an SMS transaction, it is on for the duration of that transaction and will not be turned off.

Trace points

Here are guidelines as to the bare minimum of trace points in an Messaging Manager adapter transaction:

Input

1 Message received from network
    With which addresses?
2 Message decoding information
    Do we allow alternate delivery?
    Which protocol version is this?
    What was the message text (if showPrivate)?
3 Message passed to MM
    Result from ParentContext::handleSMSubmit?
4 Response received from MM
5 Response sent to network

Output

1 SMSubmit received from MM
    Is the delivery type SME or MC?
    Do we need to consult a third party (for example, HLR) for any reason?
    What are the addresses involved?
2 Outgoing encoding information
    Which protocol version are we using?
3 Message sent to network
4 Response received from network
5 Response sent to MM

Configuring SMSs to trace

The prefixes of SMSs that are to be traced must be configured in the tracing section of the **eserv.config** file.

Any number entered into the origAddress or destAddress lists in this section will be matched against the CLI or DN. If any match is found, then this specific Messaging Manager transaction will be traced through the system.

Setting the prefixes to be traced too broadly, (for example, to trace all prefixes) could have some impact on the system performance, depending on the free IO bandwidth and the speed of the disks where the log file is being written. We do not suggest tracing every SMS going through the system; this could impact performance.

Tracing output is cached, and the cache written to file periodically. The file is closed and reopened when a specified number of tracepoints have been traced, this flushes anything in the cache. The tracing output is appended to the file specified each time the cache is flushed, either because it is full, or the outputFileCycle period is reached. The tracing output file may become large over time, so it is recommended that it be cleaned out regularly, using the smslogcleaner.

Tracing output

Here is an example tracing output.

```
2006/09/01 04:17:01.902 ID#  1010: i.MAP: Started originating transaction for adapter MAP1
2006/09/01 04:17:01.902 ID#  1010: i.MAP: Got TCAP BEGIN - OA=00001310:9, DA=0000139:8
2006/09/01 04:17:01.904 ID#  1010: o.MAP: incomingOriginatingNumberRules unmatched:
2006/09/01 04:17:01.904 ID#  1010:        (4)'000004'
2006/09/01 04:17:01.905 ID#  1010: i.MAP: DataCodingScheme = '0'
2006/09/01 04:17:01.906 ID#  1010: i.MAP: Received MAP v2 ForwardSM
2006/09/01 04:17:01.907 ID#  1010: i.MAP: Message Type = '0'
2006/09/01 04:17:01.908 ID#  1010: i.MAP: Text = 'Test Message'
2006/09/01 04:17:01.908 ID#  1010: i.MAP: Priority = '0'
2006/09/01 04:17:01.908 ID#  1010: i.MAP: Timezone = ''
2006/09/01 04:17:01.908 ID#  1010: i.MAP: Message class = '0'
2006/09/01 04:17:01.908 ID#  1010: i.MAP: Status Report Requested = '0'
2006/09/01 04:17:01.908 ID#  1010: i.MAP: MSISDN=004001000004, SCA=0040015114406267
2006/09/01 04:17:01.908 ID#  1010: i.MAP: OA=004001000004, DA=004001000004
2006/09/01 04:17:01.908 ID#  1010: i.MAP: Alternate delivery is allowed
2006/09/01 04:17:01.909 ID#  1010: i.MAP: Sending GenericSM to MMX
2006/09/01 04:17:01.909 ID#  1010: MMX: Handling SM for adapter MAP1, Message Type Submit(0)
2006/09/01 04:17:01.909 ID#  1010: MMX: Source adapter protocol was MAP
2006/09/01 04:17:01.914 ID#  1010: MMX: Inbound path 'MAP_SME_MAP1'
2006/09/01 04:17:01.915 ID#  1010: MMX: CDR for adapter MAP1 at 20060901041701
2006/09/01 04:17:01.916 ID#  1010: MMX: Set originating domain 9242, 'NoPlan'
2006/09/01 04:17:01.917 ID#  1010: MMX: Set destination domain 9242, 'NoPlan'
2006/09/01 04:17:01.917 ID#  1010: MMX: 1 concurrent transactions, 1000 max
2006/09/01 04:17:01.918 ID#  1010: MMX: Message not throttled
2006/09/01 04:17:01.918 ID#  1010: MMX: Initialised routing class to Submit(0)
2006/09/01 04:17:01.918 ID#  1010: MMX: Looking for originating trigger rule with:
2006/09/01 04:17:01.918 ID#  1010: MMX:  - Message Type: Submit(0)
2006/09/01 04:17:01.918 ID#  1010: MMX:  - Orig. address 000004, domain 'NoPlan'
2006/09/01 04:17:01.919 ID#  1010: MMX: Submit orig. trigger rule found
2006/09/01 04:17:01.919 ID#  1010: MMX:  - Matched on domain
2006/09/01 04:17:01.919 ID#  1010: MMX:  - Default routing class Submit(0) from message type
2006/09/01 04:17:01.919 ID#  1010: MMX:  - Disabled triggering to ACS Control Plan
2006/09/01 04:17:01.919 ID#  1010: MMX:  - Action Route(2)
2006/09/01 04:17:01.926 ID#  1010: MMX: Outgoing Message Type: Submit(0)
2006/09/01 04:17:01.926 ID#  1010: MMX: Looking for destination trigger rule with:
2006/09/01 04:17:01.927 ID#  1010: MMX:  - Message Type: Submit(0)
2006/09/01 04:17:01.927 ID#  1010: MMX:  - Dest. address 000004, domain 'NoPlan'
2006/09/01 04:17:01.927 ID#  1010: MMX: No matching Submit dest. trigger rule
2006/09/01 04:17:01.927 ID#  1010: MMX:  - Disabled triggering to ACS Control Plan
2006/09/01 04:17:01.928 ID#  1010: MMX:  - Action Route(2)
2006/09/01 04:17:01.928 ID#  1010: MMX:  - Routing class Submit(0)
2006/09/01 04:17:01.934 ID#  1010: MMX: Found routing rule 101869 using:
2006/09/01 04:17:01.934 ID#  1010: MMX:  - Routing class Submit(0)
2006/09/01 04:17:01.934 ID#  1010: MMX:  - Orig. address 000004, domain 'NoPlan'
2006/09/01 04:17:01.934 ID#  1010: MMX:  - Message Centre name Default
2006/09/01 04:17:01.935 ID#  1010: MMX:  - Found path 11586, SMPP1_SMSC
2006/09/01 04:17:01.935 ID#  1010: MMX: Using path 11586, SMPP1_SMSC
2006/09/01 04:17:01.936 ID#  1010: MMX: Trying output adapter SMPP1 (SMPP-mmxiSMPP.so)
2006/09/01 04:17:01.936 ID#  1010: MMX: Message modified flag - false
2006/09/01 04:17:01.937 ID#  1010: o.SMPP: Received GenericSM from MMX
2006/09/01 04:17:01.937 ID#  1010: o.SMPP: OA=004001000004 DA=004001000004
2006/09/01 04:17:01.938 ID#  1010: o.SMPP: Text - 'Test Message'
2006/09/01 04:17:01.944 ID#  1010: o.SMPP: Got outgoing connection 'SMPP1_SMSC.S.SMPP1_SMSC'
2006/09/01 04:17:01.945 ID#  1010: o.SMPP: outgoingOriginatingNumberRules unmatched:
2006/09/01 04:17:01.945 ID#  1010:        (4)'000004'
2006/09/01 04:17:01.945 ID#  1010: o.SMPP: outgoingDestinationNumberRules unmatched:
2006/09/01 04:17:01.945 ID#  1010:        (4)'000004'
2006/09/01 04:17:01.947 ID#  1010: o.SMPP: Sending submit_sm to protocol handler
2006/09/01 04:17:01.948 ID#  1010: o.SMPP: submit_sm #3 queued to be sent out on the wire
2006/09/01 04:17:01.948 ID#  1010: MMX: Term-tran accepted GenericSM
```

```
2006/09/01 04:17:01.949 ID#  1010: i.MAP: MMX processed GenericSM successfully
2006/09/01 04:17:01.949 ID#  1010: MMX: Input adapter handled TCAP_BEGIN successfully
2006/09/01 04:17:01.951 ID#  1010: o.SMPP: Handling SMPP message:
2006/09/01 04:17:01.951 ID#  1010:   command_id =                  0x80000004 (submit_sm_res
                                    p)
2006/09/01 04:17:01.951 ID#  1010:   command_status =              0x00000000
2006/09/01 04:17:01.951 ID#  1010:   sequence_number =             0x00000003
2006/09/01 04:17:01.951 ID#  1010:   message_id =                  "no-dr"
2006/09/01 04:17:01.953 ID#  1010: o.SMPP: Result from MMX is success
2006/09/01 04:17:01.953 ID#  1010: MMX: Delivery receipt ID: SMPP1_SMSC|no-dr
2006/09/01 04:17:01.953 ID#  1010: MMX: Got result 'success' from term-trans
2006/09/01 04:17:01.958 ID#  1010: i.MAP: SMS delivery succeeded
2006/09/01 04:17:01.958 ID#  1010: i.MAP: Sending MAP v2 ForwardSM Ack.
2006/09/01 04:17:01.958 ID#  1010: MMX: Inbound adapter returned 'success'
2006/09/01 04:17:01.961 ID#  1010: o.SMPP: MMX returned success
```

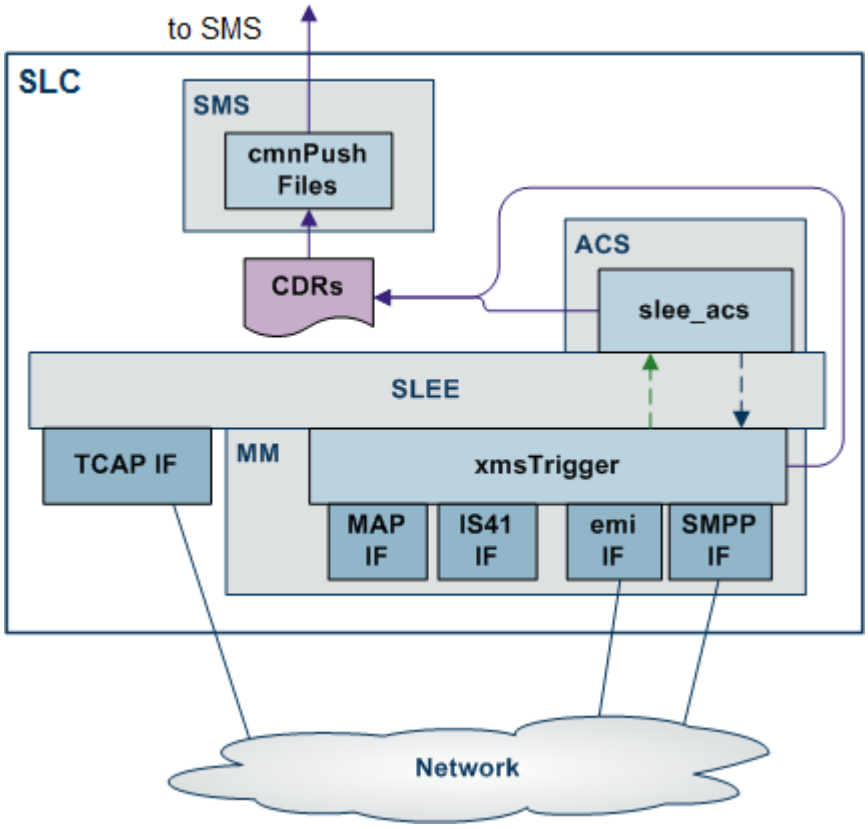
Messaging Manager EDRs

EDR collection

The xmsTrigger of Messaging Manager produces EDRs to be used in post processing as required.

Diagram

This diagram shows the components on the SLC that generate and migrate Messaging Manager EDRs.



File name and location

Messaging Manager EDRs are saved to file, in the **eserv.config**:

- With the base filename specified by the *filename* (on page 65) parameter (in the format B*ase_file_nameDate_and_time*.**cdr**)
- In the location specified by the *destdir* (on page 65) parameter (by default "`/IN/service_packages/XMS/cdr/closed/`")

List of tags

For a full list of tags and description of Messaging Manager EDRs, refer to *Event Detail Record Reference Guide*.

Delivery Receipts

Introduction

When an SMS is sent, a delivery receipt may be requested. A delivery receipt may be requested for a number of reasons:

- The sender may request a delivery receipt.
- MM may request a delivery receipt. This may be used for billing on delivery of a SMS to the handset, rather than just to the SMSC.

A delivery receipt is requested from the SMSC by setting the registered delivery flag. After delivering the SMS to the destination, the SMSC sends a delivery receipt back to the sender. This indicates if the SMS was successfully sent or not. The delivery receipt contains an ID (or delivery receipt ID) that links it to the original SMS sent.

Delivery Receipt rules

Here are the basic delivery receipt rules for all adapters.

- Delivery Receipts for FDA to an SME -
  Messaging Manager will produce delivery receipts for direct delivery where the originating party has requested a delivery receipt. This applies for successful FDAs.
  Only one attempt will be made to send these internally produced delivery receipts.
  They will be sent direct to the sender over the originating Protocol.
- Delivery Receipts Received from a Message Centre -
  Requested Delivery receipts can be sent to Messaging Manager over the IP adapters. The original transaction must have had a delivery path where either inbound or outbound processing was over the IP adapter.
- Delivery Receipts are timestamped in local time.

These delivery receipts will be subject to the normal routing rules of the relevant adapter. It is important to note that a delivery receipt can only be sent over a route that is going to an SME (FDA).

Each delivery receipt is prefixed with the value of the deliveryReceiptId parameter.

Exceptions are:

- The TDMA instance of the IS41 adapter as this will not accept delivery receipts.
- MAP v1 of the MAP protocol will not support receiving delivery receipts.

Notes:

1. **ASP -MM - SS7 SMSC** is only going to work if the HLR is configured to map the ASP's address to Messaging Manager.
2. **Handset - MM - SS7 SMSC** - Similarly, the DR is only going to go via MM (rather than straight to the handset) if the network is set up to use Messaging Manager Navigator to direct MT traffic to Messaging Manager.
3. DRs may also be produced due to the alwaysProduceNonDeliveryReceipt adapter setting.

4 Some protocols (for example, SMPP) have separate flags for requesting delivery and non-delivery receipts.

5 The Translation Type can be passed in delivery receipts sent by the MAP protocol. This requires the TT to be set in the inbound adapter's outbound TT field.

Chapter 7

Tools and Utilities

Overview

Introduction

This chapter explains the tools and utilities that are installed by the Messaging Manager application and how to configure them.

In this chapter

This chapter contains the following topics.

PME Configuration

Introduction

The **mmxPMEConfig.sh** shell script is provided to allow you to configure the per-product type profile tags that are used by MM for configuring terminating services such as PME and email/IM <-> SMS.

PME configuration script

Run the command `./mmxPMEConfig.sh` from **/IN/service_packages/XMS/bin** as acs_oper, for each of the tags listed below. This will populate the ACS global profile with the tag and value.

Syntax

The syntax is:

`./mmxPMEConfig.sh` *`tag value`*

where

- *tag* is the name of the tag
- *value* is the value for the tag

Example:

To populate the global profile tag 'MM Enhanced SMS to Email Shortcode' with the value 71, which is required for Enhanced SMS to Email processing:

```
./mmxPMEConfig.sh ENHANCED_SMS_TO_EMAIL 71
```

or

```
./mmxPMEConfig.sh ENHANCED_SMS_EMAIL 71
```

or

```
./mmxPMEConfig.sh enhanced_sms_email" 71
```

Tags

This table describes each tag.

Tag	Suggested Value
BARRING_RESPONSE|BAR_RESPONSE|bar_response"	00 (Accept), 01 (Reject), 02 (Discard)
BARRING_NOTIFY|BAR_NOTIFY|bar_notify"	True/False
WAIT_FOR_HUNTING_RESULT|WAIT_FOR_HUNTING|wait_for_hunting"	True/False
UNIQUE_AUTOREPLY_INTERVAL|AUTOREPLY_INTERVAL|autoreply_interval	**Note:** This is not required by MM 4.0 so does not have to be populated
DEFAULT_DOMAIN|domain|DOMAIN"	oracle.co.nz
TEMP_ACCESS_NUMBER_RANGE_START|TAN_START|tan_start"	7890
TEMP_ACCESS_NUMBER_RANGE_END|TAN_END|tan_end"	7899
SMS_TO_IM|SMS_IM|sms_im"	801
ENHANCED_SMS_TO_IM|ENHANCED_SMS_IM|enhanced_sms_im"	81
SMS_TO_EMAIL|SMS_EMAIL|sms_email"	701
ENHANCED_SMS_TO_EMAIL|ENHANCED_SMS_EMAIL|enhanced_sms_email"	71

Note: You can specify any of the three forms of each tag, for example, SMS_TO_IM, or SMS_IM, or sms_im"

Adding and Removing Replication Nodes

Introduction

SLCs running MM can be added to or removed from the main Messaging Manager platform configuration. This means you can take a SLC out of service for a while and then add it back in as needed.

install_routing_node.sh

You must run the following script to add a SLC to the Messaging Manager platform:

```
/IN/service_packages/XMS/bin/install_routing_node.sh
```

remove_routing_node.sh

You must run the following script to remove a SLC to the Messaging Manager platform:

```
/IN/service_packages/XMS/bin/remove_routing_node.sh
```

Chapter 8

About Installation and Removal

Overview

Introduction

This chapter provides information about the installed components for the Oracle Communications Convergent Charging Controller 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.

Pre-installation

Raw devices

When installing xmsSms on a clustered SMS, you must provide the location of the raw devices. MM can allocate tablespace storage based on raw (without a file system) partitions. This enhances the performance of MM on the SMS.

You must create the raw partitions before installing the database, using tools such as the system's format command.

The raw devices file (which you will be prompted to complete during the installation) must contain the full paths of the device files for the appropriate partitions.

The partitions must be at least as big as the required datafile sizings listed in the sizing file which will be used by the installation.

You must provide the location of the raw devices created for DEVICE_MMX_DATA1 and DEVICE_MMX_INDEX1 as the install process will use these raw devices and the location to create the data and index tablespaces used to store relevant MMX tables.

The name and location of the file is **/IN/service_packages/XMS/db/install/create/SMP/xms_devices.sql**.

Example blank file

The blank file that is provided for the person installing MM to fill out looks like this:

```
*
* $Id: xms_devices.sql,v 1.1.6.1 2006/09/05 23:35:03 radamson Exp $
* Copyright (c) 2006, Oracle
*
* This file contains the raw devices that will be
* used when creating the MMX_DATA and MMX_INDEX
* tablespaces on a cluster installation.
```

```
*
* It must be updated with the correct raw devices
* before installing the xmsSms package on a cluster.
*
*/

/* Raw device for the MMX_DATA tablespace */
define DEVICE_MMX_DATA1=

/* Raw device for the MMX_INDEX tablespace */
define DEVICE_MMX_INDEX1=
```

Example completed file

An example of a file that has been completed by the person installing MM might look something like this:

```
/*
* $Id: xms_devices.sql,v 1.1.4.1 2006/06/01 04:24:46 gthomas Exp $
* Copyright (c) 2006, Oracle
*
* This file contains the raw devices that will be
* used when creating the MMX_DATA and MMX_INDEX
* tablespaces on a cluster installation.
*
* It must be updated with the correct raw devices
* before installing the xmsSms package on a cluster.
*
*/

/* Raw device for the MMX_DATA tablespace */
define DEVICE_MMX_DATA1=/dev/md/ora0/rdsk/d92

/* Raw device for the MMX_INDEX tablespace */
define DEVICE_MMX_INDEX1=/dev/md/ora0/rdsk/d93
```

Installation and Removal Overview

Introduction

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

- Convergent Charging Controller system requirements
- Pre-installation tasks
- Installing and removing Convergent Charging Controller packages

PI packages

An installation of Messaging Manager includes the following packages, on the:

- SMS:
  - xmsSms
- SLC:
  - xmsScp

Checking the Installation

Checking the xmsSms installation

On successful installation the xmsSms package will have created the following directories:

- **/IN/service_packages/XMS**
- **/IN/service_packages/XMS/lib**
- **/IN/service_packages/XMS/tmp**
- **/IN/service_packages/XMS/db**

On successful installation the xmsSms package will have installed the following:

- **/IN/service_packages/XMS/bin/mmxPMEConfig.sh**

This shell script allows you to populate the ACS Global Profile with the tags required for PME. For more information on setting PME tags, see *PME Configuration* (on page 183).

Compiling control plans manually

If the ACS control plan compiler fails to compile control plans for any reason, you can manually compile using the Control Plan Editor. You should check the two control plans that use the Sub Control Plan node to make sure that these nodes are configured correctly. The two control plans are:

- SMS_Submit has a single instance of the Sub Control Plan that invokes SMS_to_IM_via_TAN.
- PME_Delivery has two instances of the Sub Control Plan, which invoke the IM_to_SMS and Email_to_SMS control plans. The notes beside these nodes state which sub control plan they should be attached to.

Follow these steps to compile the control plans.

Step	Action
1	Open the control plan in the Control Plan Editor.
2	Open the Sub Control Plan node.
3	Check that the right control plan is selected.
4	Save the node.
5	After this you can save and compile the main control plan.

Checking the xmsScp installation

On successful installation the xmsScp package will have installed the following binaries:

- **/IN/service_packages/XMS/bin/xmsTrigger**
- **/IN/service_packages/XMS/bin/oraPStoreCleaner**

The following configuration files will have been installed:

- **/IN/service_packages/XMS/etc/eserv.config.example**
- **/IN/service_packages/XMS/etc/tdp.conf.example**

The following shared libraries will have been installed:

- **/IN/service_packages/XMS/lib/xmsiIP.so**
- **/IN/service_packages/XMS/lib/xmsiMap.so**
- **/IN/service_packages/XMS/lib/xmsiIS41.so**
- **/IN/service_packages/XMS/lib/xmsSvcLibrary.so**
- **/IN/service_packages/XMS/lib/libxmsMacroNodes.so**

- **/IN/service_packages/XMS/lib/libnumberRules.so**
- **/IN/service_packages/XMS/lib/xmsiWrapper.so**
- **/IN/service_packages/XMS/lib/liboraPStore.so**

Post-installation Configuration

Run install_routing_node.sh

For SLCs, you must run the **install_routing_node.sh** script to include the SLC in the Messaging Manager platform configuration.

If SLC is on a local database, run the following script:

```
./install_routing_node.sh <SMS Hostname> <SMF SID> <smf user password> <SLC
Hostname> <SLC IP address> 3000 1
```

If SLC is on a remote database, run the following script:

```
./install_routing_node.sh <Remote Database Hostname> <SMF SID> <smf user password>
<SLC Hostname> <SLC IP address> 3000 1
```

Appendix A

Configuring IN Call Model Triggers

Overview

This introduces the generic configuration requirements of the Convergent Charging Controller IN Call Model.

The Convergent Charging Controller 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.

Global Parameter	Description
	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.
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.

Global Parameter	Description
	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.

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.

Global Parameter Value	Type	Description
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. • *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 Convergent Charging Controller 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)
101	n/a	ringing (Bparty)