NewNet Mobile Messaging
LGP R02.08.04

Operator Manual

Release 17.4 Revision A

February 2019

Table of Contents

List of Figures

Chapter 1

Introduction

Topics:

1.1 About this Document

This document discusses the operation and administration of the NewNet Mobile Messaging Log Processor (LGP).

The Log Processor is a product from the NewNet Mobile Messaging product family of SS7 message routing and network querying products.

Because the available functions are licensed and depend on the specific NewNet Mobile Messaging implementation, not all functions and/or applications contained in this document may be relevant or applicable to the system you will be working with. Actual screen presentation may differ from the screens presented in this document due to software changes or browser configurations.

1.2 Scope

This document discusses the functionality of the NewNet Mobile Messaging LGP component.

1.3 Intended Audience

This document is meant for all of you interested in how the Log Processor can best be used, but mainly for:

- **Implementation Engineers** who are responsible for the pre-installation, on-site installation, and configuration of NewNet Mobile Messaging in the end-user environment;
- **Maintenance and Support Engineers** who are responsible for maintaining the total system environment of which Log Processor is a part;
- **Network Operators** who are in charge of the daily operation of the NewNet Mobile Messaging systems and infrastructure.

1.4 Documentation Conventions

Typeface or Symbol	Meaning	Example
Bold	Refers to part of a graphical user interface.	Click **Cancel**.
`Courier`	Refers to a directory name, file name, command, or output.	The `billing` directory contains...
`<pointed brackets>`	Serves as a placeholder for text that the user will replace, as appropriate in context.	The file is called `MGRdata.xml.<ip>.gz`, where `<ip>` is the server's IP address.
`[square brackets]`	Indicates an optional command.	`[--validateonly]`

Typeface or Symbol	Meaning	Example
Note:	Indicates information alongside normal text, requiring extra attention.	**Note:** Ensure that the configuration...
\ (Unix)	Denotes line continuation; the character should be ignored as the user types the example, and ENTER should only be pressed after the last line.	`% grep searchkey \` `data/*.dat`

1.5 Locate Product Documentation on the Customer Support Site

Access to NewNet's Customer Support site is restricted to current NewNet customers only. This section describes how to log into the NewNet Customer Support site and locate a document. Viewing the document requires Adobe Acrobat Reader, which can be downloaded at www.adobe.com.

1. Log into the NewNet Customer Support site.

   Note: If you have not registered for this new site, click the **Register Here** link. Have your customer number available. The response time for registration requests is 24 to 48 hours.

2. Click the **Product Support** tab.
3. Use the Search field to locate a document by its part number, release number, document name, or document type. The Search field accepts both full and partial entries.
4. Click a subject folder to browse through a list of related files.
5. To download a file to your location, right-click the file name and select **Save Target As**.

Chapter 2

System Overview

Topics:

2.1 Introduction

The Log Processor (LGP) is the log storage component. The LGP:

- Collects traffic record files from Router/Firewall (RTR/FWL) nodes and loads them into a local database
- Transfers traffic record files from RTR and FWL nodes
- Parses and loads individual records into an SQL database
- Implements a programmatic interface for running queries against the loaded records
- Satisfies performance, availability, and manageability requirements

The Manager (MGR) GUI with Log Viewer (LGV) can be used to:

- Examine traffic logs for network troubleshooting
- Trace messages associated with customer complaints
- Control SMS fraud attempts

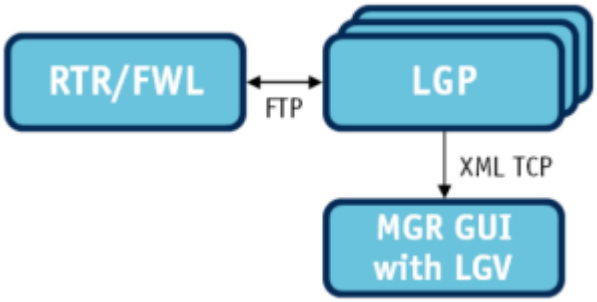

Figure 1: LGP overview

2.2 Scalable Architecture

An LGP configuration can contain one or more independent LGP nodes, providing a larger total storage and processing capacity. The LGP nodes in a multi-node configuration communicate with one another to coordinate the data collection and loading process.

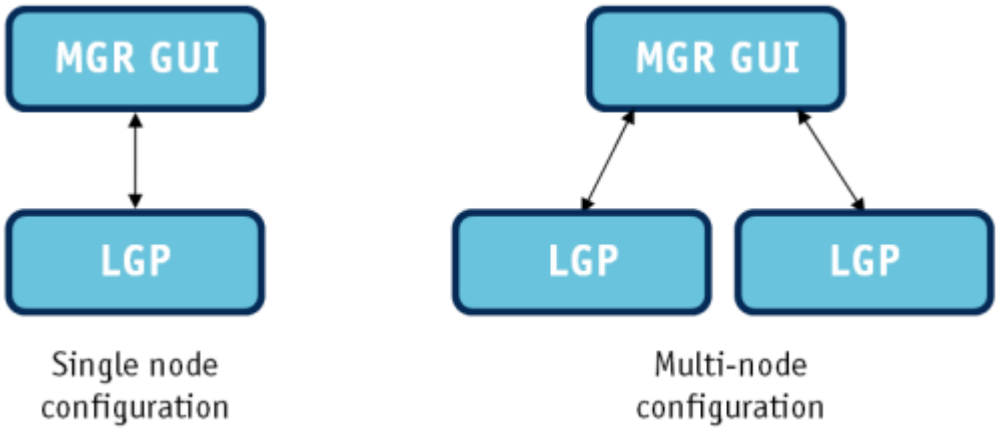

Figure 2: Node configurations

One dynamically selected master LGP node divides the data to be loaded among the active LGP nodes. The LGP nodes then retrieve and store their allocated traffic records from RTR and FWL nodes. The information is distributed at traffic log-file level to each LGP node's local SQL storage in a shared-nothing architecture (each node is independent and self-sufficient) with no replication.

To minimize the disk space requirement, information integrity relies on:

- Regular backups
- Keeping traffic data that has not yet been backed up on the original RTR/FWL systems. The loaded information is stored on the nodes in a local SQL database, with each decoded record representing a row in an SQL table.

The LGP uses a MySQL relational database server. Each LGP node has an XML interface so that clients can perform predefined queries on the local database. In a multi-node LGP deployment, the MGR client queries each node in the LGP system and merges the results.

2.3 Log Processor Logic

2.3.1 Modules

Each LGP node is composed of the following modules, running as separate UNIX processes and performing specialised tasks.

2.3.1.1 Controller

The *controller* is the main module on the LGP node and is responsible for system start-up, configuration, monitoring, and coordination with other LGP nodes.

2.3.1.2 Master

The elected *master* LGP node runs the master module at polling time to distribute the data to be loaded among the active LGP nodes. The master starts one process for each poll.

This module connects to each configured RTR/FWL node, gathers information about available files, and then allocates the files to all LGP nodes.

2.3.1.3 Collector

The *collector* module connects to the hosts indicated by the controller configuration and fetches the new files as allocated by the master.

2.3.1.4 Loader

The *loader* module reads local log files, decodes the individual records to plain format, and inserts them into the SQL database.

2.3.1.5 Query

The *query* module handles an XML GUI client query connection.

2.3.2 Data Storage

Traffic records are stored in a MySQL database. The loader module loads traffic records into the database, while the query module queries them.

2.3.3 Software Processes

The software running on the LGP host server consists of several executables:

- One instance of the LGP process containing the LGP kernel
- One instance of the LGP process running the loader functions
- One instance of the LGP process for each active query
- One instance of the master process, running the master part of the collection algorithm, executed on one LGP node in the system at poll time
- One instance of the collector process, executed on each LGP node at poll time
- One instance of the watchdog process, monitoring the LGP process and restarting and configuring it if required

The LGP configuration is maintained in the configuration file and the LGP MIB, which can be addressed using SNMP.

2.3.4 System Software Components

The LGV assumes that at least one LGP and all LGP tools are installed. For more information, refer to the Full Element Installation Manual.

2.4 Multi-Instance Support

Multi instance feature allows multiple NMM users (up to 10, including the existing `'textpass'` user) be created on the same node, each of whom will be able to run one instance of LGP on a logging element server.

Note: A separate LICENSE is required for each NMM user.

Chapter 3

Logging

Topics:

3.1 Introduction

The LGP supports two types of logging:

- Transactional logging, also called message logging, in which log records are created post-transaction
- Event logging, in which log records are created instantaneously, as events occur (limited to short messages and unexpected TCAP messages)

Note: Event logging is only supported for the RTR, HUB, and PBC.

3.2 Message Logging

Transaction logging, also called message logging, is the RTR's mechanism for logging inbound SMSC traffic. Log records are created post-transaction. These records provide information about each short message's origins, message fields, and how it was routed.

To enable message logging, create message logging profiles in the MGR (**Logging ➤ Messages ➤ Profile**) and optionally assign as defaults for categories such as MO messages, MT violations, etc. (**Logging ➤ Messages ➤ Properties**). Then, when you create a routing or counting rule, you can select whether the messages that match the rule are logged according to the applicable default profile, according to a specific profile, or not at all. You can create up to 100 message log profiles.

Important: The logging of MT status reports must be configured in the MOR rule, the MTOR rule and the default MO log profile `logPropDefaultProfileForMo` to enable MT status report logging. The logging of AT notifications must be configured in the AOR rule, the ATOR rule and the default AO log profile `logPropDefaultProfileForAo` to enable AT notification logging.

Note: While logging the addresses, the values of the addresses captured will be same as the addresses present in the messages received and transmitted by the RTR. The RTR can change the value of the TON/NPI based on the following logic:

1. In case of ASN1 extended format Ton/NPI value of the address is retained.
2. In case of ASN1 extended with Country and Network
   a. TON/NPI of the Short number is logged as 6/0.
   b. If NPI of the msisdn is set as ISDN Telephony then the TON/NPI is set as 1/1.
   c. If NPI of the msisdn is not ISDN Telephony then the original TON/NPI is retained.

3.2.1 Logging Fatal Protocol Violations

Messages may contain protocol violations that will cause the RTR to reject them before the rule engine evaluates them. These messages are not subject to routing or counting rules, and are therefore logged according to separate profiles.

A violation occurs when one or more of the following conditions apply to an inbound operation:

- An invalid SMSC address at MAP layer
- An invalid recipient address
- An invalid originator address

- An invalid IMSI
- An invalid MSC address
- A forged MSC address
- A forged IMSI
- A forged LMSI
- An unknown SMSC address at SCCP layer
- An unknown SMSC address at MAP layer
- Conflicting SMSC addresses
- A spoofed SMSC address at SCCP layer
- A spoofed SMSC address at MAP layer
- An unsolicited MtForwardSm operation (that is, one not preceded by a SendRoutingInfo operation)
- A spoofed originator address

3.2.2 Logging CDMA- and TDMA-Specific Fields

The RTR supports logging of CDMA- and TDMA-specific SMPP message fields.

The CDMA-specific SMPP message fields are:

- `privacy_indicator`
- `source_subaddress`
- `dest_subaddress`
- `user_response_code`
- `language_indicator`
- `number_of_messages`
- `callback_num`
- `display_time`
- `ms_validity`
- `alert_on_message_delivery`
- `its_reply_type`
- `its_session_info`

The TDMA-specific SMPP message fields are:

- `callback_num_pres_ind`
- `callback_num_atag`
- `sms_signal`
- `alert_on_message_delivery`

3.2.3 Creating Message Logging Profiles

To create a logging profile:

1. In the left navigation bar, select **Logging ➤ Messages ➤ Profile**.
   The Logging Profiles tab appears.
2. Click **Add New**.
   A new Logging Profiles tab appears.
3. Enter a unique name for the profile in the **Name** box (maximum 31 characters).

4. Optionally enter a description of the profile in the **Description** box.
5. Select the message type(s) that the profile applies to:
6. In the **Processing Directory** box, enter the location in which to store the log files while they are being created (defaults to `/var/TextPass/log/processing`).

   Note: In a multi-instance setup, the processing directory will be shared by all the RTR instances running on a node.

7. In the **Finished Directory** box, enter the location in which to store the log files after they are created (defaults to `/var/TextPass/log/available`).

   Note: In a multi-instance setup, the finished directory will be shared by all the RTR instances running on a node.

8. In the **Copy 1 of Finished Directory** through **Copy 9 of Finished Directory** boxes, enter directories in which to created hard links to the files in the finished directory (by default, there are no copies of the finished directory).

   Note: The finished directory and all finished directory copies must be on the same disk partition as the processing directory.

9. In the **Filename Template** box, enter the template to use to name the log files (defaults to `log_%h_%U_%Y%m%d_%H%M%S_%3.dat`).

   Important: If multi-instances of RTR are running on the same node, then it is important to include `%U` escape sequence, which will be translated to `UID` (operating system user identifier). This ensures that multiple instances of RTR do not try to create files with identical names.

10. In the **Max. File Size** box, enter the maximum size of a log file in bytes (defaults to 1048576 bytes, which is 1 MB). The range is 1024 byes (1 KB) to 1073741824 bytes (1 GB).
11. In the **Max. File Duration** box, enter the maximum duration of a log file in seconds (defaults to 3600 seconds, which is 1 hour). The range is 1 second to 2,678,400 seconds (1 month).
12. In the **Max. File Records** box, enter the maximum number of records to allow in a log file (defaults to 10000 records). The range is 1 record to 10,000,000 records.
13. Select the file format from the **File Format** list:

    - ASN.1 Extended
    - ANS.1 Extended with country and network info (default)

14. In the **Starting Sequence Number** box, enter the number with which to start the log file numbering sequence (defaults to 0).
15. From the **Suspect/Trusted Messages** list, select the type of messages to log:

    - Only suspect
    - Only trusted
    - Both suspect and trusted (default)

    Suspect/Trusted Messages logging pertains to whether the RTR/FWL considers the message source to be suspect or trusted. Inbound MO messages may come from a suspected or trusted MSC, while inbound SendRoutingInfoForSm operations and MT messages may come from a suspected or trusted SMSC. Refer to the Firewall Guide for more information about suspect and trusted qualifications.

16. From the **Failed/Succeeded Messages** list, select the type of messages to log:

- Only failed
- Only succeeded
- Both failed and succeeded (default)

Failed/Succeeded Messages logging pertains to whether a message was successfully delivered from the Mobile Messaging system to the destination. For outbound MT messages, the destination is an MSC (MS). For outbound AT messages, the destination is an application.

17. From the **Accepted/Rejected Messages** list, select the type of messages to log:

- Only accepted
- Only rejected
- Both accepted and rejected (default)

Accepted/Rejected Messages logging pertains to whether the Mobile Messaging system accepts a message or not. This applies to inbound MO and inbound AO messages only.

18. From the **Legitimate/Violated Messages** list, select the type of messages to log:

- Only legitimate
- Only violated
- Both legitimate and violated (default)

Legitimate/Violated Messages logging pertains to whether a message passes or fails a spoof check. Legitimate messages are messages that passed the spoof check. Violated messages are messages that failed the spoof check.

19. From the **Expired/Deleted Messages** list, select the type of messages to log:

- Do not log (default)
- Only expired
- Only deleted
- Both expired and deleted
- Only replaced
- Both expired and replaced
- Both deleted and replaced
- Expired, deleted and replaced

Expired/Deleted Messages logging pertains to the AMS delivery result. Expired messages reached their validity period expiration or their maximum number of delivery attempts. Deleted messages were manually deleted from the AMS by a user or by an application. Replaced messages were replaced in the AMS.

20. From the **Status Report Messages** list, select the type of messages to log:

- Do not log (default)
- Status reports

21. From the **Copied/Forwarded Messages** list, select the type of messages to log:

- Do not log (default)
- Only copied
- Only forwarded
- Both copied and forwarded

22. From the **Transparent User Data Level** list, select the level of user data logging:

    - Use Global Setting
    - Always
    - Protocol Violation
    - Never
    - Encrypt Always

    Note: If the value of the logging profile parameter is `Use Global Setting`, the RTR will use the semi-static parameter `logtransparentuserdatalevel`.

23. Click **Save**.

    The MGR creates the logging profile and closes the tab.

24. Activate the profile.

3.2.4 Configuring Message Logging Properties

The message logging properties are used to set a log profile as the default log profile for certain messages or violation types.

Note: Changing the default profile will change the logging for all rules that are configured with this default profile.

Prerequisites:

- Logging profile

To configure logging properties:

1. In the left navigation bar, select **Logging ➤ Messages ➤ Properties**.

   The Logging Properties tab appears.

2. Select a logging profile for mobile-originating or IMS-originating traffic from the **MO Messages** list.
3. Select a logging profile for mobile-terminating traffic from the **MT Messages** list.
4. Select a logging profile for application-originating traffic from the **AO Messages** list.
5. Select a logging profile for application-terminating traffic from the **AT Messages** list.
6. Select a logging profile to use in case of a fatal protocol violation in a mobile-originating message from the **MO Violations** list.
7. Select a logging profile to use in case of a fatal protocol violation in a mobile-terminating message from the **MT Violations** list.
8. Select a logging profile to be used for mobile-originating SMS commands from the **MO Commands** list.
9. Select a logging profile to be used for application-originating SMS commands from the **AO Commands** list.
10. Select a logging profile to be used for mobile number portability (MNP) violations of originator(in case of MO message) from the **MNP Violations** list.
11. Click **Save**.

    The MGR saves the logging properties and closes the tab.

3.2.5 Configuring Message Log Filters

Prerequisites:

- A Log Processor (LGP) must be installed and configured.

To configure a message log filter:

1. In the left navigation bar, select **Logging ➤ Messages ➤ Filters**.

   The Filters tab appears.

2. Click **Add New**.
   A new Filters tab appears.
3. Enter a unique name for the filter in the **Name** box (maximum 31 characters).
4. Optionally enter a description of the filter in the **Description** box.
5. From the list on the left, select matching:

   - Match all elements
   - Match any element

6. Click **Add Element**.
   The list of available elements appears.
7. Add an element to the filter by doing one of the following:

   - Clicking an element (highlighted in gray)
   - Clicking
     
     to expand a list of elements, then clicking an element

8. Select a condition for the element from the **Condition** list:

   - Equals
   - Does not equal
   - Contains
   - Does not contain
   - Is not set
   - Starts with
   - Ends with
   - Match Flexibly

   Note:

   1. The "Match flexibly" condition is supported only for the user data fields ("*messagefields_userData*", "*smsSubmit_smsUserData*", "*smsDeliver_smsUserData*" and "*userData_normalizedText*"). It is meant for efficient searching of the user data text when the LGP 'flexible text search' functionality is licensed and also enabled through the configuration. Refer to *Flexible User Data Text Search* for more details.

   2. The field "*userData_normalizedText*" supports only the "Match flexibly" option and none of the other matching conditions listed above.

9. Enter a value in the **Value** box.
10. Optionally add more elements to the filter and set their conditions and values.

    Note: To remove an element from the filter, click
    
    next to it.

11. Click **Save**.
    The MGR creates the message log filter and closes the tab.

3.2.6 Using Message Log Search

Prerequisites:

- A Log Processor (LGP) must be installed and configured
- Message log filter
- Message log column

To search the information in the logs:

1. In the left navigation bar, select **Logging ➤ Messages ➤ Search**.
   The Search Messages tab appears.
2. Click
   
   next to the **Start Date** box and select the date and time at which to start the log search.
3. Click
   
   next to the **End Date** box and select the date and time at which to end the log search.
4. In the **Max Results** box, specify a maximum number of log records to return.
5. Select a log filter from the **Filter** list.

   Note: A warning message will be shown in the result page if the filter contains the 'userData' field when the encryption license is enabled. This is because a filter with 'userData' field would normally return empty results with encrypted data.

6. Select a view column from the **View Columns** list. The option **Default** can be selected to get a result with the default columns.

   Note: There are 5 fixed columns (**ID**, **Date/Time**, **Hostname**, **originator Number**, **recipient Number**). The dynamic columns will be added after the fixed columns, according to the selected View Column. Although the same information is mentioned correctly in MGR operator manual in section - 12.2.5 (Using Message Log Search) at point no.- 6, which can also be referred for necessary correction in LGP OM.

   Note: If user data is included in View Columns and the user data is encrypted while **View Encrypted Data** is false for the user, the user data will be empty in the results.

7. Select the checkbox next to **Execute Query in Background** to perform the search in background.

   The dynamic columns in the output columns will be available with the background query.

Note: The background query result-set will be available on **Logging ➤ Background Query** link. Refer to section *Background Query* for more details on background query.

8. Click **Search**.

   The MGR searches for records based on the starting time, ending time, and filter, and returns any log records that match (up to the maximum number that you specify).

   Note: The maximum size of a request is 20kB. If the size is bigger than 20kB, an error will be returned.

   Note: If a decryption error occurred (e.g. reading the key file) while decrypting the UserData, the UserData field will be used to show the error.

9. To export the search results as a comma-separated value (CSV) file that can be opened in programs such as Microsoft Excel and OpenOffice.org Calc, click **Export**.

   Note: Message Log Search displays all the matching logs on GUI irrespective of whether the 'File Format' is selected as 'ASN.1 Extended' or 'ASN.1 Extended with country and network info' during the configuration of Log Profile on MGR. For Message Log Search using '`inboundMessageType`' as the Filter Element, Filter Value can only be selected as

   '`trustedMtFwdSmWithCountryAndNetworkInfo`'.

   For example, add a new filter with unique name (say: messagetype) on Filter Element '`inboundMessageType`' with Condition 'equals' and Value '`trustedMtFwdSmWithCountryAndNetworkInfo`'.

   Log search result for this filter will fetch the records for both "`trustedMtFwdSm`" and "`trustedMtFwdSmWithCountryAndNetworkInfo`".

3.3 Event Logging

Event log records are created instantaneously, when an event occurs. Each event record is defined in ASN.1 form, and expresses what event has occurred to which object (such as a short message) in what moment. Multiple events for the same object can be correlated and sorted by timestamp, producing the event trail for that object. You can use the Event Logs Search within the Customer Care Interface (CCI) to view the event trail of a short message.

To enable event logging, create an event logging profile in the MGR (**Logging ➤ Events ➤ Profile**) and assign it to a property (**Logging ➤ Events ➤ Properties**). The supported objects are short messages and unexpected TCAP messages. You can create up to 100 event log profiles.

3.3.1 Event Trail Events

The following events can appear in an event trail:

Category	Event	Description
GSM	gsmOrigInRequest	Incoming MO message
	gsmOrigInResponse	Response to an incoming MO message
	gsmOrigOutRequest	Outgoing MO message

Category	Event	Description
	gsmOrigOutResponse	Response to an outgoing MO message
	gsmOrigRejection	Incoming MO message rejected
	gsmTermInRequest	Incoming MT message
	gsmTermInResponse	Response to an incoming MT message
	gsmTermOutRequest	Outgoing MT message
	gsmTermOutResponse	Response to an outgoing MT message
	gsmTermRejection	Incoming MT message rejected
	gsmRoutingInfoQuery	HLR query
Application	appOrigInRequest	Incoming AO message
	appOrigInResponse	Response to an incoming AO message
	appOrigOutRequest	Outgoing AO message
	appOrigOutResponse	Response to an outgoing AO message
	appOrigRejection	Incoming AO message rejected
	appTermInRequest	Incoming AT message
	appTermInResponse	Response to an incoming AT message
	appTermOutRequest	Outgoing AT message
	appTermOutResponse	Response to an outgoing AT message
	appTermRejection	Incoming AT message rejected
AMS	amsStoreRequest	AMS store request
	amsStoreResponse	AMS store response
	amsDeliveryAttempt	AMS message delivery attempt
	amsDeliveryRejection	AMS message delivery attempt rejected
	amsTermination	AMS message termination
RTR	routingDecision	(Rule-based) RTR routing decision
	externalCondition	The result of an external condition evaluation
	notification	A delivery notification or status report has been created for the SM
Personalized Services	smCopied	For this message, a copy was created
	copyCreated	This message is a copy; it was created by copying another message
	copyRejection	Delivery of copy was rejected internally and dropped

Category	Event	Description
	forwardAttempt	Forwarding was attempted for this message
	forwardRejection	A forwarding attempt for this message was rejected internally
	smAutoReplied	An auto reply message has been issued for this short message. This event shows the ID of the ARP message.
	autoReplyCreated	An ARP message has been created. This event shows the ID of original SM, the name of the requesting service and the ARP message text.
	signatureInserted	A signature has been inserted to an (inbound) SM. This event shows the name of the requesting service.
PBC	prepaidCharging	Prepaid charging parameters

3.3.2 Logging of Unexpected TCAP Messages

In case of fraud attempts between operators, unexpected TCAP end messages or TCAP continue messages with a ReturnResult will be received by an operator. These unexpected messages are identified as messages with no existing dialog and can be logged as stipulated in section 1.1.3 of AA.50 and section 3.1.3 of IR.71, industry-standard documents for SMS fraud detection and prevention.

Only the unexpected TCAP messages will be logged when they arrive at the RTR.

Note: Late responses on outgoing TCAP dialogs (i.e. responses after a timeout) will result in logging these responses as unexpected TCAP messages.

The following TCAP information will be logged:

- MTP3 originating point code
- SCCP calling party address (including country and network when provisioned)
- SCCP called party address (including country and network when provisioned)
- TCAP message type (end or continue)
- TCAP originating transaction ID (in case of TCAP continue)
- TCAP destination transaction ID
- Application context name

Logging of unexpected TCAP messages only logs messages that are received directly on the RTR/FWL. Unexpected TCAP messages going directly to other SMS components (for example, the SMSC in FWL-only deployments) are not logged. Those messages should be logged by the SMSC, as the RTR/FWL does not see these messages. If these other SMS components do not have unexpected TCAP logging capabilities, all MT traffic should be directed through the RTR/FWL, so unexpected TCAP message logging of the RTR/FWL can be used for these SMS components too.

3.3.3 Creating Event Logging Profiles

To create a logging profile:

1. In the left navigation bar, select **Logging ➤ Events ➤ Profile**.

   The Event Logging Profiles tab appears.

2. Click **Add New**.

   A new Event Logging Profiles tab appears.

3. Enter a unique name for the profile in the **Name** box (maximum 31 characters).
4. Optionally enter a description of the profile in the **Description** box.
5. In the **Processing Directory** box, enter the location in which to store the log files while they are being created (defaults to `/var/TextPass/log/processing`).

   Note: In a multi-instance setup, the processing directory will be shared by all the NMM component instances running on a node.

6. In the **Finished Directory** box, enter the location in which to store the log files after they are created (defaults to `/var/TextPass/log/available`).

   Note: In a multi-instance setup, the finished directory will be shared by all the NMM component instances running on a node.

7. In the **Filename Template** box, enter the template to use to name the log files (defaults to `%N_event_%U_%h_%Y%m%d_%H%M%S_%3.dat`).

   Important: As multiple components on the same system may share an event logging profile configuration, it is important to include the `%N` escape sequence, which will be translated to the component name. This ensures that multiple components will not try to create files with identical names.

   Important: If multi-instances of NMM components are running on the same node, then it is important to include `%U` escape sequence, which will be translated to `UID` (operating system user identifier). This ensures that multiple components will not try to create files with identical names.

8. In the **Max. File Size** box, enter the maximum size of a log file in bytes (defaults to 1,048,576 bytes, which is 1 MB). The range is 1024 byes (1 KB) to 1,073,741,824 bytes (1 GB).
9. In the **Max. File Duration** box, enter the maximum duration of a log file in seconds (defaults to 3600 seconds, which is 1 hour). The range is 1 second to 2,678,400 seconds (1 month).
10. In the **Max. File Records** box, enter the maximum number of records to allow in a log file (defaults to 10,000 records). The range is 1 record to 10,000,000 records.
11. In the **Starting Sequence Number** box, enter the number with which to start the log file numbering sequence (defaults to 0).
12. Click **Save**.

    The MGR creates the logging profile and closes the tab.

13. Activate the profile.

3.3.3.1 Configuring Event Logging Copies

For each event logging profile, you can create a list of directories in which the LGP will save copies of the files in the profile's **Finished Directory**. You can use this functionality to ensure that an automatic backup of completed files exists. The list for each event logging profile can contain up to 10 directories.

Note: The **Finished Directory** and all directories containing copies must be on the same disk partition as the **Processing Directory**.

Prerequisites:

- Event logging profile

To add a directory to an event logging copies list:

1. In the left navigation bar, select **Logging ➤ Events ➤ Profile**.
   The Event Logging Profiles tab appears.
2. Click the name of an event logging profile.
3. In the Event Logging Copies section, click **Add New**. A new Event Logging Copies List tab appears.
4. In the **Copy Directory** box, enter a directory in which the LGP should create hard links to the files in the **Finished Directory** (defaults to `/var/TextPass/backup`).
   You can only add a directory to the list if the event logging profile is deactivated.
5. From the **Event Log Profile** list, select the profile to use for this copy location (defaults to the profile that you clicked earlier).
6. Click **Save**.
   The MGR closes the tab and adds the directory to the Event Logging Copies list.
7. On the event logging profile tab, click **Add New** to add another directory to the list, or click **Save** to save the profile.

3.3.4 Configuring Event Logging Properties

The event logging properties are used to set a specific log profile for certain events.

Prerequisites:

- Event logging profile

To configure event logging properties:

1. In the left navigation bar, select **Logging ➤ Events ➤ Properties**.
   The Event Logging Properties tab appears.
2. Select an event logging profile for:

Option	Description
Short Message Events	Events related to the processing of Short Messages, that are used for the Event Logs Search within the Customer Care Interface (CCI) component.
Rogue TCAP Events	Events that are used for unexpected TCAP messages (that is, TCAP messages for which no TCAP dialog exists). All unexpected TCAP End and TCAP Continue messages with a `ReturnResult` message will be logged to this profile when they arrive at the RTR.

3. Click **Save**.
   The MGR saves the event logging properties and closes the tab.

3.3.5 Configuring Event Log Filters

Event log filtering is available for unexpected TCAP events, but it is not available for the short message events.

Prerequisites:

- A Log Processor (LGP) must be installed and configured.

To configure an event log filter:

1. In the left navigation bar, select **Logging ➤ Events ➤ Filters**.
   The Filters tab appears.
2. Click **Add New**.
   A new Filters tab appears.
3. Enter a unique name for the filter in the **Name** box (maximum 31 characters).
4. Optionally enter a description of the filter in the **Description** box.
5. From the list on the left, select matching:
   - Match all elements
   - Match any element
6. Click **Add Element**.
   The list of available elements appears.
7. Add an element to the filter by doing one of the following:
   - Clicking an element (highlighted in gray)
   - Clicking
     
     to expand a list of elements, then clicking an element
8. Select a condition for the element from the **Condition** list:
   - Equals
   - Does not equal
   - Contains
   - Does not contain
   - Is not set
   - Starts with
   - Ends with
9. Enter a value in the **Value** box.
10. Optionally add more elements to the filter and set their conditions and values.

    Note: To remove an element from the filter, click
    
    next to it.
11. Click **Save**.
    The MGR creates the log filter and closes the tab.

3.3.6 Using Event Log Search

Prerequisites:

- A Log Processor (LGP) must be installed and configured
- Event log filter.
- Message view column

To search the information in the logs:

1. In the left navigation bar, select **Logging ➤ Events ➤ Search**.
   The Search Events tab appears.
2. Click
   
   next to the **Start Date** box and select the date and time at which to start the log search.
3. Click
   
   next to the **End Date** box and select the date and time at which to end the log search.
4. In the **Max Results** box, specify a maximum number of log records to return.
5. Select a log filter from the **Filter** list.
6. Select the checkbox next to **Execute Query in Background** to perform the search in background.

   Note: The background query result-set will be available on **Logging ➤ Background Query** link. Refer to section *Background Query* for more details on background query.
7. Click **Search**.
   The MGR searches for records based on the starting time, ending time, and filter, and returns any log records that match (up to the maximum number that you specify).

   Note: The maximum size of a request is 20kB. If the size is bigger than 20kB, an error will be returned.
8. To export the search results as a comma-separated value (CSV) file that can be opened in programs such as Microsoft Excel and OpenOffice.org Calc, click **Export**.

3.4 Configurable User Data Logging

The level of user data logging can be configured to meet local privacy regulations. The text of messages, for which user data should not be logged, will be masked by a string of Xs.

To allow flexibility and privacy protection, the maximum logging level is determined and locked by the license key. Therefore, the log level that is configured in the semi-static configuration file and per logging profile can never exceed the maximum log level that is defined in the license.

The following table specifies information for the semi-static variable `logtransparentuserdatalevel` and per logging profile parameter **Transparent User Data Level**.

logtransparentuserdatalevel	This governs the system-wide setting for the transparent User data level. If the variable is less than the License key, the RTR gives an error.
Transparent User Data Level	This parameter can be configured per logging profile. If this parameter is configured with a value less than the license key, the RTR gives an error.

The following table specifies the possible configuration values for the semi-static variable `logtransparentuserdatalevel` and the logging profile parameter **Transparent User Data Level** for different License key values:

License Key values	**Allowed logtransparentuserdatalevel values**	**Allowed Transparent User Data Level values**
always	always, protocolViolationsOnly, never	always, protocolViolationsOnly, never, Use Global Setting
protocol violations only	protocolViolationsOnly, never	protocolViolationsOnly, never, Use Global Setting
never	never	never, Use Global Setting

The following tables describe different possible configuration for the semi-static parameter `logtransparentuserdatalevel` and per logging parameter **Transparent User Data Level**. Based on the license key, the possible configuration values for semi-static & per logging profile parameter are:

License Key value is "always"

logtransparentuserdatalevel	**Transparent User Data Level**	**User-data Logged**
always	always	Yes
always	protocolViolationsOnly	Protocol-Violations
always	never	No
protocol violations only	always	Protocol-Violations
protocol violations only	protocolViolationsOnly	Protocol-Violations
protocol violations only	never	No
never	always	No
never	protocolViolationsOnly	No
never	never	No
always	encrypt-always	Yes (encrypt Data)
encrypt-always	always	Yes (encrypt Data)

License Key value is "protocolViolationsOnly"

logtransparentuserdatalevel	Transparent User Data Level	User-data Logged
never	never	No
never	protocolViolationsOnly	No
protocol violations only	never	No
protocol violations only	protocolViolationsOnly	Protocol-Violations
never	always	Not possible
always	always	Not possible
always	protocolViolationsOnly	Not possible
always	never	Not possible
protocolViolationsOnly	always	Not possible
any value	encrypt-always	Not possible
encrypt-always	any value	Not possible

License Key value is "never"

logtransparentuserdatalevel	Transparent User Data Level	User-data Logged
never	never	No
protocol violations only	protocolViolationsOnly	Not possible
never	protocolViolationsOnly	Not possible
protocol violations only	never	Not possible
never	always	Not possible
always	always	Not possible
always	protocolViolationsOnly	Not possible
always	never	Not possible
protocol violations only	always	Not possible
any value	encrypt-always	Not possible
encrypt-always	any value	Not possible

3.5 Log File Creation

Log files are always generated in the processing directory. When the file has been generated, it is moved to the finished directory. Files in the finished directory can be further processed by the operator.

The log record is written to the log file that is currently open. This log file will be closed and physically written to disk based when any of the following conditions become true:

- Maximum file duration time limit reached. If the time since the creation of the log file has exceeded the value defined in the maximum file duration as configured in the log profile, the log file is closed and written to disk, and a new log file is created.
- Maximum file size limit reached. If the log file size reaches the maximum file size as configured in the log profile, the log file is closed and written to disk, and a new log file is created.
- Maximum file records limit reached. If the number of log records reaches the maximum log records as configured in the log profile the log file is closed and written to disk, and a new log file is created.

3.6 Background Query

When you execute Message or Event search in background then MGR sends the search request to LGP and waits for response in background. During the background search you can perform other tasks on MGR GUI. Background query is not impacted by logout. You can logout of MGR and later re-login to see background query result . You can track the status of background queries using **Logging ➤ Background Query** link. When the query is completed, the status will be displayed as `completed`. You can click on the completed background query result row to view the result-set or download as CSV. When the background query completes, an indication is displayed on top-right corner of MGR GUI. Indication icon is displayed when you login or refresh MGR page or open a new tab after completion of Background Query.

Prerequisites to perform Background Query:

- A Log Processor (LGP) must be installed and configured
- Message or Event log filter

To search the information in the logs:

1. In the left navigation bar, select **Logging ➤ Messages ➤ Search** for message search or select **Logging ➤ Events ➤ Search** for event search.

   The Search Messages tab appears.
2. Click
   
   next to the Start Date box and select the date and time at which to start the log search.
3. Click
   
   next to the End Date box and select the date and time at which to end the log search.
4. In the Max Results box, specify a maximum number of log records to return.
5. Select a log filter from the Filter list.
6. Select the checkbox next to **Execute Query in Background** to perform the search in background.
7. Select a view column from the **View Columns** list. The option **Default** can be selected to get a result with the default columns.

   Note: If the user changes the selected **View Columns** while a search is in-progress, the result for background query will be as per the altered view set.
8. Click Search.
9. An alert message will appear to indicate that the query will be executed in background. Select "OK" to start executing background query.

The MGR searches for records based on the starting time, ending time, and filter, and returns any log records that match (up to the maximum number that you specify).

Note: The maximum size of a request is 20kB. If the size is bigger than 20kB, an error will be returned.

10. You can track the status of background queries (In-progress or completed) using **Logging ➤ Background Query** link.
11. Once background query is completed, there would be indication icon displayed on MGR GUI at right top with Green color. Indication icon is displayed when user logged-in, refresh page or open a new tab after completion of background query.
12. Once query status is changed to `Completed`, you can click on background query row to view the result-set.
13. To export the search results as a comma-separated value (CSV) file that can be opened in programs such as Microsoft Excel and OpenOffice.org Calc, click Export.

    Background Query is helpful in performing long searches, searching on LGP Archive records which may take a very long duration.

Note: There are some restrictions on Background Query

1. You are not allowed to delete in-progress query.
2. Only one background query can be in-progress state in a domain.

3.6.1 Background Query Configuration

For Background Query following parameters should be provisioned in `/usr/TextPass/etc/mgr.cnf`:

1. `background_query_dir`: This parameter specifies path where MGR will store Background Query search results. You must ensure that sufficient space is available on disk at this path otherwise Background Query cannot be performed. You can change the path of this directory when MGR is in stopped state. Please ensure that old results are copied to new path otherwise you will not be able to view older results on MGR GUI.

   Note: You must ensure that the owner of `background_query_dir` folder is `textpass` and `textpass` user is able to create files in `background_query_dir` folder.

2. `minimum_size_required`: Minimum free disk space required to perform Background Query is set in 'minimum_size_required'. Default value is '1024 MB'. If the free space available in background_query_dir is less than minimum_size_required', then background query will not be performed.
3. `max_background_query_records`: Maximum records that can be retrieved by a Background Query. The default value is `65535`.

   Note: Due to browser and cache memory limitation, it is strongly recommended not to increase the value of this parameter more than `65535`. If there can be more than `65535` records, the background query can be split in multiple queries using shorter search ranges.

   Sample configuration

```
<logviewer maximum_records="1000" background_query_dir="/var/TextPass/MGR/lgp"
minimum_size_required="1024" max_background_query_records="65535"/>
```

Note: KeepAlive Timer settings must be configured on MGR to detect background query failure due to unexpected issues like remote server restart or network cable failure. Please refer to the section Configure KeepAlive Timer of the Full Element Installation Manual.

Chapter 4

Log Processing

Topics:

4.1 Coordination Among Nodes

The LGP system is scalable. Therefore, adding additional nodes increases the system's total capacity and throughput. Fault-tolerance requirements also mean that if a node becomes unavailable, the other nodes handle its load until it is repaired.

However, data is not replicated among nodes, as system availability does not require all data to be accessible in the event of a node failure. Therefore, the nodes must be able to distribute data among themselves without duplicating information.

This functionality is accomplished by running a distributed load-sharing algorithm that uses a multicast UDP protocol to:

- Select a master node
- Coordinate polling of RTR/FWL nodes

4.2 Allocation Among Nodes

Log file distribution among LGP is implemented in a master-slave topology. One LGP node in the LAN is recognized as the master node; it runs the allocation procedure as the master process at configured poll times.

Allocation is accomplished by remotely (through FTP or SFTP) renaming log files on the Router nodes to subdirectories named after the LGP nodes in the system (e.g. `/var/TextPass/log/available/lgp1/<name>.dat`).

Note: Unsecured protocols such as FTP and telnet may introduce security risk to your network. The customers are at their own risks if the unsecured protocols are enabled and used on their systems.

The allocation time is encoded as a UNIX timestamp followed by a comma, prefixed to the original file name. This timestamp is used to reallocate files that have not been collected by the node to which they were allocated, in the event that the node fails (`/var/TextPass/log/available/lgp1/<timestamp>,<node>,<name>.dat`).

The allocation process attempts to allocate data evenly over all LGP nodes in the system, after file loading. A LGP node can have a configured static weight (`tpconfig` attribute `lgpstaticweight`) that multiplies the volume of data the node receives from the allocation (for example, if one LGP node has more disk space).

4.3 Collection

All LGP nodes use FTP or SFTP to fetch RTR log files to a local directory for loading.

Note: Unsecured protocols such as FTP and telnet may introduce security risk to your network. The customers are at their own risks if the unsecured protocols are enabled and used on their systems.

Each node runs the collector module after:

- The master finishes, or

- A configurable time elapses from the last configured poll time

Each LGP node collects files from its own subdirectory on the Router server, which has the same name as the LGP node.

The collection process is:

1. Files are remotely renamed with a `TRANS` extension and the transfer start timestamp (to prevent the master from reallocating a potentially old file).

   For example: `1237302903,rtr1,rtr_traff_log.dat.TRANS`

   Note: Files with a TRANS extension and a timestamp older than a configurable time are considered failed during transfer. Their TRANS extension is removed, and they are collected again.

2. Files are transferred to the `TRANSFER` subdirectory of the local load directory (to prevent the loader module from loading a partial file).
3. During transfer:
   a) The timestamp prefix is removed, and
   b) The file name is prefixed with the source RTR node name
4. When the transfer finishes:
   a) The local file copy is moved to the local load directory
   b) The files are kept on the RTR node with a DONE extension (if the transfer was successful). The DONE files are deleted after a configurable period (`lgpdoneclean`).

   Note: Files with a DONE extension are not reallocated by the master, regardless of their age, to prevent loading the same file on two different nodes.

5. When the files are loaded locally and the local database is backed up, the DONE files are removed from the RTR after a configurable time period (`lgpdoneclean`)

   This procedure ensures that, if a LGP node fails, the files can still be loaded manually.

4.3.1 Compression

To lower bandwidth usage, compression can be enabled on the transfer of the data files from the RTR to the LGP when using SFTP (not possible for FTP). Compression can be enabled by setting the `lgpcompresssftpdata` attribute.

Note: This functionality lowers bandwidth usage of the transfer, but will put extra load on the CPU of the RTR and the LGP.

4.4 Loading

After the RTR log files are transferred to a local directory on the LGP node, determined by the `lgplocallogdir` parameter, the loader module picks up each file in alphabetical order, decodes each file, and loads the contents of each file into the local database.

Each database record corresponds to one row in the records file.

The loader module obtains the source RTR node name from the filename prefix that the collector module added. If the loader finds a file with the same name and source node already present in the

database, the loader considers the file a duplicate and moves it to the DUPLICATE subdirectory of the load directory.

If errors are encountered during the load, all of the file's records are rolled back from the database and the file is moved to the ERROR subdirectory.

After a successful load, the loader commits the file to the database and removes it from the local load directory.

4.5 Querying

The LGV interface can query the data in a local LGP's database using an XML request/response over a TCP socket connection.

The reporting period, record types, and field match values can be specified in the XML request structure. The resulting matching records are returned encoded in an XML response structure.

This interface is intended to be used only by the MGR.

4.6 Recovery

If a LGP node fails and all disk-stored data is lost (in the database or in the local load directory), the data can be recovered if the database was backed up.

Note: This procedure only applies to storage failures in which all data has been lost. Normally mirrored disks are used, so one of them can fail without causing data loss.

To recover:

1. Resolve the issue that caused the failure or install a new LGP node.
   If you select the latter option, duplicate the original configuration. **Do not start the LGP node.**
2. Restore the last database backup.
3. Manually transfer (by FTP) the TRANS and DONE files in the log files subdirectory named after the LGP node from all Router nodes to the local load directory.
   For example, if the LGP node is named `lgp1`, manually transfer all TRANS and DONE files in the `lgp1` directory to the local load directory.

   Note: Unsecured protocols such as FTP and telnet may introduce security risk to your network. The customers are at their own risks if the unsecured protocols are enabled and used on their systems.
4. Rename all files with a new name:
   a) Remove the timestamp prefix
   b) Add the originating RTR node name as a prefix
   c) Remove the TRANS or DONE extension
      For example, change `10000011,rtr_traff_log.dat.DONE` to `rtr1,rtr_traff_log.dat`.
5. Start the LGP node.

Chapter 5

Configuration

Topics:

5.1 Introduction

The semi-static configuration consists of two files:

- Host-specific configuration file: Contains parameters for a specific LGP and is located at `/usr/TextPass/etc/<hostname>_config.txt`, where `<hostname>` is the host name of the LGP
- Common configuration file: Contains parameters that are common to all LGPs and is located at `/usr/TextPass/etc/common_config.txt`

Configuration parameters can be placed in either file. In case of a conflict in the settings of a parameter, the host-specific configuration file always takes precedence over the common configuration file.

5.2 Configuration File Structure

This diagram depicts the entities in the LGP semi-static configuration file.

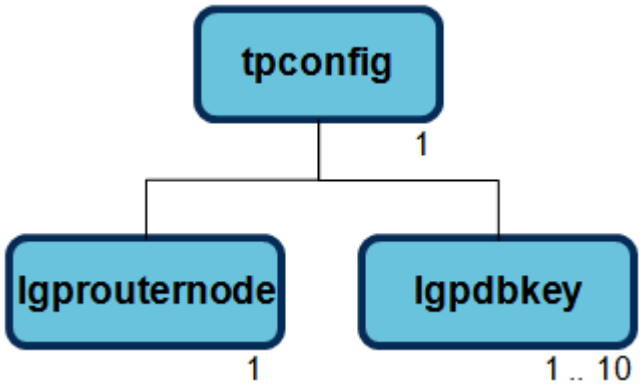

Figure 3: Configuration file structure

5.3 Semi-Static Configuration

This section describes the semi-static configuration file entities and attributes.

5.3.1 tpconfig Entity

This section describes the `tpconfig` attributes.

5.3.1.1 ipaddress

Mandatory/Optional

Optional

Location

Host-specific configuration file

Description

IP address of the server.

5.3.1.2 runtextlgpprocess

Mandatory/Optional

Mandatory (for running the LGP)

Location

Host-specific configuration file

Description

Specifies whether the LGP process should be started. Should be "true" for running the LPG.

Valid Values

- true
- false

Default

false

5.3.1.3 runtextpassprocess

Mandatory/Optional

Mandatory (for running the RTR)

Location

Host-specific configuration file

Description

Specifies if the RTR process should be started. Should be "true" for running the RTR.

Valid Values

- true
- false

5.3.1.4 runtpfclientprocess

Mandatory/Optional

Optional

Location

Host-specific configuration file

Description

Specifies whether the `tp_fclient` file transfer client should be started. Should be "true" if the common semi-static configuration file is needed.

Valid Values

- true
- false

5.3.1.5 lgpalertdelay

Mandatory/Optional

Optional

Location

Common configuration file

Description

Number of minutes to wait before retransmitting a loader trap.

Default

15 minutes

5.3.1.6 lgpaliveinterval

Mandatory/Optional

Optional

Location

Common configuration file

Description

Period (in seconds) with which to update the LGP node's statistics in the TNL heartbeat user data for node coordination. It is recommended to set this parameter to the same value as the TNL heartbeat interval.

Default

15 seconds

5.3.1.7 lgpasndisablefields

Mandatory/Optional

Optional

Location

Common configuration file

Description

List of ASN records to disable, as a comma-delimited list of `structure.[substructure.]*field`. The names come from the ASN log record specification. Ensure that you do not disable fields that might be necessary for lookups.

Default

"" (all fields are loaded)

5.3.1.8 lgpcompresssftpdata

Mandatory/Optional

Optional

Location

Common configuration file

Description

Boolean to specify if compression on the transfer of the data files when using SFTP is enabled (true) or disabled (false). This variable has no effect when using FTP.

Default

false

5.3.1.9 lgpdbclean

Mandatory/Optional

Optional

Location

Common configuration file

Description

Database clean schedule, as a comma-delimited list of `hh24mi` times. For example, 1 A.M. and 1 P.M. is 0100,1300.

Default

"" (no cleaning)

5.3.1.10 lgpdbcleanolderthan

Mandatory/Optional

Optional

Location

Common configuration file

Description

Maximum file age (in hours) in the database. Files that were loading into the database before this time are deleted at the next database clean.

Default

240 hours

5.3.1.11 lgpdbdatabase

Mandatory/Optional

Optional

Location

Common or host-specific configuration file

Description

Name of the database in the database server to use. Must not be in use by another LGP node.

Default

lgp

Note: If multiple instances of LGP are running on same server, then `lgpdbdatabase` must be specified in host-specific configuration file and two NMM users must not use the same `lgpdbdatabase`. Hence, in a multi-instance setup, the user must specify this parameter in its host specific file and ensure that non-`textpass` user does not use `'lgp'` database.

5.3.1.12 lgpdbpassword

Mandatory/Optional

Mandatory

Location

Common or host-specific configuration file

Description

Password for the database user, a scrambled octet string, obtained from password xor reverse (username). Change the password from the default, for security reasons. The password should be changed on the MySQL database for this user as well.

All Mobile Messaging components should use unique database user names and passwords, for example for the LGP:

- User: LGPuser
- Password: LGPuser123

Default

LGPuser123

Note: If multiple instances of LGP are running on same server, then `lgpdbpassword` must be specified in the host-specific configuration file of each user.

5.3.1.13 lgpdbport

Mandatory/Optional

Optional

Location

Common configuration file

Description

Port to use for connecting to the database server (0 indicates "use system default").

Default

0

5.3.1.14 lgpdbuser

Mandatory/Optional

Mandatory

Location

Common or host-specific configuration file

Description

User name to use for the database server.

All Mobile Messaging components should use unique database user names and passwords, for example for the LGP:

- User: LGPuser
- Password: LGPuser123

Default

LGPuser

Note: If multiple instances of LGP are running on the same server then `lgpdbuser` must be specified in host-specific configuration file and two NMM users must not use the same `lgpdbuser`. Hence, in a multi-instance setup, the user must specify this parameter in its host specific configuration file.

5.3.1.15 lgpdoneclean

Mandatory/Optional

Optional

Location

Common configuration file

Description

Maximum age (in hours) of files on the SFTP server which were successfully transferred to the LGP server. A file from the SFTP server will be removed if the maximum age has passed.

Default

0 (files are deleted immediately after transfer)

5.3.1.16 lgpfiletransferprotocol

Mandatory/Optional

Optional

Location

Common configuration file

Description

Specifies the file transfer protocol used by the LGP to retrieve the log records from the RTRs.

Note: Unsecured protocols such as FTP and telnet may introduce security risk to your network. The customers are at their own risks if the unsecured protocols are enabled and used on their systems.

Default

sftp

5.3.1.17 lgpfulltextsearch

Mandatory/Optional

Optional

Location

Common configuration file/Host-configuration file

Description

Indicates whether the flexible user data text search functionality is supported or not. The functionality is supported only when this parameter is set to “true” and the “Full Text Search” license option is enabled on the LGP.

Note that this parameter can be set only when the LGP is inactive.

Default

false

5.3.1.18 lgploadedrestartwaitinterval

Mandatory/Optional

Optional

Location

Common configuration file

Description

Number of seconds to wait before restarting the loader process, if it exited with a fatal error.

Default

60 seconds

5.3.1.19 lgploaderbatchsize

Mandatory/Optional

Optional

Location

Common configuration file

Description

The batch size in number of records for the batch files to be inserted into the database.

Default

10000

5.3.1.20 lgploadertempdir

Mandatory/Optional

Optional

Location

Common configuration file

Description

Local directory to be used for creating temporary files used for batch loading. This directory is best placed on a tmpfs or ramfs file system. Placing this directory on the disk has no advantage. The data is not integral for the data safety.

Default

/tmp

5.3.1.21 lgploadertimeoutinterval

Mandatory/Optional

Optional

Location

Common configuration file

Description

Maximum time (in seconds) between consecutive loader statistics. If the loader does not provide feedback to the controller after this amount of time, it is considered dead and is restarted.

Default

120 seconds

5.3.1.22 lgplocallogdir

Mandatory/Optional

Optional

Location

Common or host-specific configuration file

Description

Path to the local directory used for temporarily storing log files until they are loaded. Can be given as an absolute path (recommended) or as a relative path from the LGP binary. `lgpquerysqllogspath` and `lgplocallogdir` should not use the same directory.

If the parameter is not specified, the default directory used is /var/TextPass/LGP. When the parameter is specified, it must specify an existing directory, in which the textpass user has permission to manage files.

Default

`/var/TextPass/LGP`

Note: If multiple instances of LGP are running on the same node then `lgplocallogdir` must be specified in host-specific configuration file and two NMM users must not use the same `lgplocallogdir`. The default directory used is `/var/<<user name>>/LGP` for all users except `textpass`. Hence, in a multi-instance setup, the user must specify this parameter in its host specific configuration file and make sure that the directory has permissions to be managed by the user.

5.3.1.23 lgpmaxcollectorrun

Mandatory/Optional

Optional

Location

Common configuration file

Description

Maximum run time (in seconds) for the master process. If the process takes longer, it is assumed that the process failed and it is killed.

Default

20 minutes

5.3.1.24 lgpmaxloadtime

Mandatory/Optional

Optional

Location

Common configuration file

Description

Maximum time (in minutes) for the LGP local file queue to be loaded. If, at the current loading speed, it is estimated that it will be surpassed, a trap is triggered. Set to a value that will allow operator intervention before the system is irrecoverably out-of-sync.

Default

60 minutes

5.3.1.25 lgpmaxmastercollectorwait

Mandatory/Optional

Optional

Location

Common configuration file

Description

Maximum interval (in seconds) for which the collector module waits for the master process, running on the master node, to complete before starting.

Default

5 minutes

5.3.1.26 lgpmaxmasterrun

Mandatory/Optional

Optional

Location

Common configuration file

Description

Maximum time (in seconds) for the master process. If the process takes longer, it is assumed that the process failed and it is killed.

Default

20 minutes

5.3.1.27 lgpmaxnumberof

Mandatory/Optional

Optional

Location

Common configuration file

Description

Number of queries that can be run in parallel.

Default

10

5.3.1.28 lgpnodename

Mandatory/Optional

Mandatory

Location

Host-specific configuration file

Description

Name of the LGP node, used during collection. Each node in an LGP system (in a LAN) must have a unique name.

Default

lgp

Note: If multiple instances of LGP are running on the same node then ensure that each LGP is assigned a unique `lgpnodename`.

5.3.1.29 lgppollperiod

Mandatory/Optional

Optional

Location

Common configuration file

Description

Polling period (in minutes). Files are retrieved from RTR nodes at this frequency, between lgpPollStart and lgpPollStop. Applies to all RTR nodes. Must be the same on all LGP nodes in a system.

Default

15 minutes

5.3.1.30 lgppollstart

Mandatory/Optional

Optional

Location

Common configuration file

Description

Start time of the polling scheme, as a UNIX time stamp. When set to 0, polling will start immediately. Otherwise, the start time must be set to an absolute value that indicates the moment when polling will start. Applies to all RTR nodes. Must be the same on all LGP nodes in a system.

Default

0

5.3.1.31 lgppollstop

Mandatory/Optional

Optional

Location

Common configuration file

Description

End time of the polling scheme, as a UNIX time stamp. When set to -1, polling will never stop. Otherwise, the stop time must be set to an absolute value that indicates the moment when polling will start. Applies to all RTR nodes. Must be the same on all LGP nodes in a system.

Default

0

5.3.1.32 lgpquerybacklog

Mandatory/Optional

Optional

Location

Common configuration file

Description

Size of the query server socket backlog (number of entries in the new connections list).

Default

10

5.3.1.33 lgpquerygeneratesqllogs

Mandatory/Optional

Optional

Location

Common configuration file

Description

SQL query tracing flag. If enabled, all SQL queries are written in a separate file.

Default

0

5.3.1.34 lgpquerymaxwait

Mandatory/Optional

Optional

Location

Common configuration file

Description

Maximum run time for a query (in seconds). When this time expires, the query is terminated and an error is returned to the client.

Default

20 seconds

5.3.1.35 lgpquerysqllogspath

Mandatory/Optional

Optional

Location

Common configuration file

Description

Complete path and file name for the SQL query trace file. `lgpquerysqllogspath` and `lgplocallogdir` should not use the same directory.

Default

sql.log

5.3.1.36 lgprecycleinterval

Mandatory/Optional

Optional

Location

Common configuration file

Description

Time (in hours) after which files that were allocated on the SFTP server to an LGP node but were not picked up are redistributed. Should be set to the medium defect repair time, in case of an LGP node failure.

Default

5 days

5.3.1.37 lgpstartupwait

Mandatory/Optional

Optional

Location

Common configuration file

Description

Interval (in seconds) for which a newly started node stays in the start-up state and does not assume the master role.

Default

2 minutes

5.3.1.38 lgpstaticweight

Mandatory/Optional

Optional

Location

Common configuration file

Description

Static weight for the node, used to multiply the data size allocated to the current node. For example, if the static weight is 2 and all other nodes have a static weight of 1, this node will receive twice as many bytes to load. Normally, this parameter should be set to 1 for all identical nodes in an LGP system.

Default

1

5.3.1.39 lgptransclean

Mandatory/Optional

Optional

Location

Common configuration file

Description

Maximum age for files in transfer (in hours). If a file is in the transfer state (has the TRANS extension) after this much time, it is collected again.

Default

1 hour

5.3.1.40 snmppropalarmownipaddress

Mandatory/Optional

Optional

Location

Common/Host configuration file

Description

The IPv4 address of the TextPass node. If set, this address will be used as source address for sending SNMP traps. This parameter is used to populate Trap Agent address for IPv4/IPv6 address. If this parameter is not set, then the IPv4 address of first network interface is used to populate Trap Agent Address in SNMP Traps.

Valid Value
IPv4 address

Default

Empty string

5.3.1.41 snmppropalarmownipv6address

Mandatory/Optional

Optional

Location

Common/Host configuration file

Description

The IPv6 address or hostname of the TextPass node. If set, this address will be used as source address for sending SNMP traps.

Valid Values

IPv6 address or hostname(maximum length can be of 255 characters)

Default

Empty string

5.3.1.42 snmpproplistenaddresstype

Mandatory/Optional

Optional

Location

Common/Host configuration file

Description

This parameter indicates whether SNMP Listener type is IPv4 only or Dual-stack.

Valid Values

ipv4 or dual

Default

ipv4

5.3.2 lgprouternode Entity

This section describes the `lgprouternode` attributes.

5.3.2.1 name

Mandatory/Optional

Mandatory

Location

Common configuration file

Description

Host address or name of the RTR's SFTP server.

Valid Values

1 - 64 character string

5.3.2.2 dir

Mandatory/Optional

Mandatory

Location

Common configuration file

Description

Remote log directory path on the RTR. Cannot contain a colon (:).

Valid Values

1 - 64 character string

5.3.2.3 user

Mandatory/Optional

Mandatory

Location

Common configuration file

Description

SFTP user name on the RTR.

Valid Values

1 - 64 character string

5.3.2.4 password

Mandatory/Optional

Optional

Location

Common configuration file

Description

Remote log directory path on the RTR. A scrambled octet string, obtained from password xor reverse(routerNodeUser). Unscrambled password cannot contain a colon (:).

Valid Values

0 - 64 character string

5.3.2.5 pattern

Mandatory/Optional

Optional

Location

Common configuration file

Description

Log file pattern, used to select the files to be loaded from the remote log directory. Perl-style regular expression; must match only log files and must not contain the string start and end `^ $` (these are added by the master and the collector nodes). The recommended form is to match the file extension (`.*\.dat`). Cannot contain a colon (`:`).

Valid Values

1 - 64 character string

5.3.3 lgprouternode Entity

This section describes the `lgprouternode` attributes.

5.3.3.1 dbkey

Mandatory/Optional

Mandatory

Location

Common configuration file

Description

A comma separated list of database column names. This comma separated list will be used to create a database key. The key will use the same order as is provisioned. Column names will be validated. No more than 16 columns will be allowed in single key.

Valid Values

0 - 1054 character string

5.3.4 lgpprop Entity

This section describes the **lgpprop** attributes. Note that the following attributes can be set only when the LGP is inactive.

5.3.4.1 normalisationmap

Mandatory/Optional

Optional

Location

Common configuration file/Host-configuration file

Description

A newline ('\n' or
) separated list of character-sets that can be specified in plain text format or in HTML format.

The list will be used as a tokenization map while pre-processing the user data text in the log record fields "smsSubmit_smsUserData", "smsDeliver_smsUserData" and "messageFields_userData", as well as during the pre-processing of any flexible text search query string. Note that although the name of this configuration parameter is "normalisationmap", it is actually not used for the normalization of any text. Rather it is used to control the tokenization step, the output of which is then further normalized by collating consecutive identical tokens into a single token.

As part of the pre-processing, original text characters belonging to each character-set contained in the list will be tokenized using the corresponding first character of the set.

When this entity is not configured, the default normalization map will be used.

Example

If the configured list is:

“aA\nbB\n3eE”

Then “A” and “a” will become “a”,

“B” and “b” will become “b”,

“3”, “e” and “E” will become “3”.

Valid Values

0 - 1000 character string

5.3.4.2 wordboundaries

Mandatory/Optional

Optional

Location

Common configuration file/Host-configuration file

Description

A list of characters that can be specified in plain text format or in HTML format.

The characters contained in the list will be used as word boundaries while pre-processing the user data text in the log record fields “smsSubmit_smsUserData”, “smsDeliver_smsUserData” and “messageFields_userData”, as well as during the pre-processing of any flexible text search query string. Note that the control characters in ASCII and extended ASCII sets are implicitly considered as word boundaries and hence do not need to be configured.

As part of the pre-processing, if the characters listed in this entity are encountered then they will be replaced by “white-space” character.

When this entity is not configured, the default set of word boundaries will be used.

Example

If the configured list is:

“!@#”

Then the characters “!”, “@”, “#” will be replaced by white-space.

Default

0 - 1000 character string

5.3.4.3 stopwords

Mandatory/Optional

Optional

Location

Common configuration file/Host-configuration file

Description

A comma (',') separated list of words that can be specified in plain text format or in HTML format.

The words contained in the list will be used as stop words while pre-processing the user data text in the log record fields "smsSubmit_smsUserData", "smsDeliver_smsUserData" and "messageFields_userData", as well as during the pre-processing of any flexible text search query string.

As part of the pre-processing, if the words listed in this entity are encountered then they will be removed.

When this entity is not configured, the default set of stop words will be used.

Example

If the configured list is:

"about,an,are"

Then the words "about", "an", "are" will be removed.

Valid Values

0 - 10000 character string

5.4 Dynamic Configuration

This section describes dynamic configuration of LGP supported on MGR GUI.

Parameter	Usage
Convert To Archive Storage After	Files that were loaded into the DB more than this time ago shall be archived at the next DB archive schedule. Default value is '-1' which indicate that no archiving will be performed. The value should be greater than or equal to 24 hours to perform the archiving. The maximum allowed value is 26304 hours (1096 days).
Database Archive Schedule	DB archive schedule, written as a comma separated list of hh24mm times. For example 1 AM and 1 PM is 0100,1300. Default value is 0135. Maximum 24 durations can be configured as DB archive schedule. DB archive schedule must be set to off-peak hour period. Archival/Un-archival procedure can take time and transaction/event files are not loaded during archival/un-archival. Configuring DB archive schedule parameter to peak-hours will impact the performance of the LGP loader and LGP query module.

Parameter	Usage
Background Query Max Run Time	Maximum run time for a background query, in seconds. When the user executes a background query from the MGR GUI, then LGP uses "Background Query Max Run Time" instead of `lgpquerymaxwait` parameter. **Note:** For this parameter is only applicable for Background queries. For normal queries `lgpmaxquerywait` parameter is applicable.

Refer to the MGR Operator Manual for more information about the dynamic configuration and background query.

5.5 Customizing Database Keys

You can configure the LGP database keys using an SNMP table called `lgpDbKeyTable`, which is configured using the semi-static configuration attribute `lgpdbkey`. This functionality allows you to customize your LGP database keys in accordance with your actual query usage, to make your most commonly executed queries more efficient.

Note: Changing the default `lgpDbKeyTable` entries will affect loading and querying speed.

The table can contain a maximum of 10 entries. Each entry contains a list of comma-separated column names. The LGP uses each table entry to construct a database key.

`lgpDbKeyTable` entries must be set when the LGP is inactive. They will only affect tables that are created after the entries are set. The LGP verifies that the column names are valid; if any column name is invalid, the LGP will return an SNMP error and write an error to syslog, indicating which column name is invalid.

Default Database Keys

`lgpDbKeyTable` contains the following default entries:

Entry	Column Names
1	`rec_time, inboundMessageType`
2	`mapMsisdn_gsmAddress_number, rec_time`
3	`smsDeliver_smsScTimestamp`
4	`correlatedSriSm_sccpCgPa_sccpAddress_globalTitle_number, rec_time`
5	`sccpCgPa_sccpAddress_globalTitle_number, rec_time`
6	`sccpCdPa_sccpAddress_globalTitle_number, rec_time`
7	`rejectInfo_rejectCause`

The LGP uses these entries until SNMP set is used to change or add an entry. Then, the LGP uses the customized entries.

Available Database Keys

To view the columns that are available for use as database keys:

1. Log in to the LGP server as user root.
2. Log in to MySQL as user root:
   ```
   # mysql --login-path=mysql_root
   ```
3. Execute:
   ```
   # mysql> show tables from <lgp_db>;
   ```
   Where `<lgp_db>` is the name of the LGP database. The default database name is `lgp`.
4. Select the newest table that begins with `lgp_inb_msg_` and ends with a date in format YYYYMMDD. Then, execute:
   ```
   mysql> select column_name from information_schema.columns where \
   table_name='<lgp_inb_msg_YYYYMMDD>' and table_schema='<lgp_db>';
   ```
   Where `<lgp_inb_msg_YYYYMMDD>` is the table that you selected.
5. From this list, select the columns that you want to use as database keys.

Chapter 6

System Management

Topics:

6.1 Introduction

This chapter describes the command-line tools available to assist in determining the status of the system or locating the cause of the problem, if any.

6.2 Stopping the System

The LGP is designed to run unattended and almost maintenance-free for normal operation. If the LGP process needs to be stopped, execute the following command at the command prompt of the LGP node:

```
tp_stop --tp_lgp
```

This command will gracefully shut down the LGP and stop the LGP process and the watchdog process.

6.3 Starting the System

To start the LGP process, execute the following command at the command prompt of the LGP node:

```
tp_start --tp_lgp
```

This command will start the LGP and the watchdog process.

During initialisation, the LGP process uses the `tp_config` tool to load the configuration.

When the LGP node restarts after an unplanned outage (such as a power failure), the LGP process is restarted automatically and resumes service.

6.4 Log Processor Statistics

On each LGP server, you can check the following read-only SNMP properties for information about the server's operation and performance.

Property	Description
`lgpActiveFlag`	Indicates if the LGP node is active and participating in polling, loading, and querying.
`lgpIsMasterFlag`	Indicates if the LGP node is the master node in the LGP system.
`lgpLastCollection`	UNIX timestamp of the last successful collection (FTP poll).
`lgpQueueFiles`	Number of files in the local log directory waiting to be loaded on the LGP node.
`lgpQueueSize`	Size of the files (in KB) in the local log directory waiting to be loaded on the LGP node.

Property	Description
`lgpLoadRecs`	Average loading speed in records per second, over the last five minutes.
`lgpLoadBytes`	Average loading speed in bytes per second, over the last five minutes.
`lgpDbFiles`	Number of files loaded in the LGP database.
`lgpDbSize`	Size of the files (in KB) loaded in the LGP database.

6.5 Watchdog Process

The watchdog process and the Mobile Messaging component process communicate via Unix signals.

The watchdog process expects contact from the Mobile Messaging component process every second. If the component process does not contact the watchdog for six seconds, the watchdog stops and restarts the component process.

If a signal is missed, the watchdog writes the following message in the syslog:

```
Missing health signal, missed <number of signals> signals, allowed <max number of
missed signals>
```

If the watchdog stops the component process, it writes the following messages:

```
Missed <number of signals> health signals: trying to cleanly abort process <process
 ID>
Application killed (<process ID>), waiting <number of seconds> seconds before
restarting
```

When the watchdog attempts to restart the component process, it writes the following message:

```
Application restarted, number of unsuccessfully restarts <number of restarts>,
application was running for <number of seconds> seconds
```

If the component process dies or is stopped by the watchdog three times within 30 minutes, the watchdog stops attempting to restart the process and writes the following message:

```
Application terminated, too many restarts within predefined interval
```

To monitor the `syslog`, execute:

```
# tail -f /var/log/messages
```

6.6 System Verification

6.6.1 Basic System Verification

For basic verification of the LGP status, execute the following command at the command prompt:

```
tp_status
```

The `tp_status` command provides the operational state and uptime of all configured devices and components.

6.6.2 Advanced System Verification

For more detailed information about the LGP system, use the `tp_walk` command-line tool (as user textpass) to retrieve information about specific LGP counters, as follows:

```
$ tp_walk OID
```

Where `OID` is an object identifier uniquely identifying an SNMP attribute group that refers to the respective Log Processor verification counters (see *Log Processor Statistics*)

For more information about `tp_walk`, refer to the Tools Operator Manual.

Additional LGP troubleshooting methods are investigation of core files and analysis of XML configuration data files.

6.7 Command-line Tools for Troubleshooting

The following command-line tools are available for troubleshooting purposes:

Tool	Description	More Information
`tp_status`	Provides the operational state and uptime of NewNet Mobile Messaging components.	Refer Tools Operator Manual
`tp_system`	Allows you to: • View software and hardware information • Activate licenses • Boot the system • Enable and disable subscriptions to the trap service	Refer Tools Operator Manual
`tp_walk`	Provides the real-time value of any SNMP attribute.	Refer Tools Operator Manual
`tp_walkall`	Provides the real-time value of all SNMP attributes.	Refer Tools Operator Manual
`tp_lgp_archive`	Allows exporting the LGP database based on specified date range and desired storage engine.	Refer section *tp_lgp_archive*.
`tp_lgp_query`	Allows to check the LGP transaction/event table engine type.	Refer section *tp_lgp_query*.
`tp_lgp_archive_online`	Allows you to archive or unarchive the LGP tables based on the specified date range.	Refer section *tp_lgp_archive_online*.

6.7.1 tp_lgp_archive

The `tp_lgp_archive` tool allows you to export the LGP database based on specified date range and desired storage engine.

6.7.1.1 Synopsis

```
tp_lgp_archive --start_time=timestamp --end_time=timestamp --engine=string
[--output_file=string] [--dbuser=string] [--dbpwd=string] [--host=hostname|IP]
[--db=string][--port=number]
```

6.7.1.2 Options

Tool	Description
--start_time	Start time of the table records. Records will be fetched from start time to end time. Format: `YYYYMMDD/20121009`. This is a mandatory parameter.
--end_time	End of the table records. Records will be fetched from start time to end time. Format: `YYYYMMDD/20121109`. This is a mandatory parameter.
--engine	MySQL table engine in which you want to export the data (ARCHIVE or InnoDB). This is a mandatory parameter.
--output_file	`lgp_archive_script` will create a single output file. The name is taken from the optional `output_file`. The default name is `/tmp/mysql_archive_data.sql` To guarantee uniqueness, a date string is appended.
--dbuser	Username to connect to the LGP database.
--dbpwd	Password to connect to the LGP databases.
--host	Hostname of LGP Server.
--db	Name of LGP database.
--port	MySQL Server listen port
--help	Displays help

6.7.1.3 Example

The following example command:

```
tp_lgp_archive  --start_time=20131201 --end_time=20131202 --engine=ARCHIVE
--output_file=/tmp/lgp_export
```

Creates the file `/tmp/lgp_export.sql`.

6.7.2 tp_lgp_query

The `tp_lgp_query` tool allows to query LGP database to check the engine type of LGP transaction/event tables and write output in XML format.

This helps to fetch following information:

1. List tables along with their storage engine
2. Get exact day from where InnoDB and Archive table exists.

Two types of query is supported in tp_lgp_query tool:

1. `archiveTableRecord`: It returns the following result in XML format:
   - InnoDB first available table (Day)
   - InnoDB last available table (Day)
   - Archive first available table (Day)
   - Archive last available table (Day)
2. `archiveFullTableRecord`: Returns list of all tables along-with their engine type (InnoDB or Archive) in xml format.

6.7.2.1 Synopsis

```
/opt/TextPass/LGP/Rxx.yy.zz.ww/bin/tp_lgp_query --query=string --lgp_ip=string
--lgp_port=integer --output_file=string
```

where `Rxx.yy.zz.ww` is the LGP release installed on the system.

6.7.2.2 Options

Tool	Description
--query	Query type. Supported values: `archiveTableRecord` and `archiveFullTableRecord`.
--lgp_ip	LGP Server IP-Address.
--lgp_port	LGP Query Listen port. If not specified then default port (33003) will be used.
--output_file	Output file absolute path.

6.7.2.3 Example

The following example command:

```
/opt/TextPass/LGP/Rxx.yy.zz.ww/bin/tp_lgp_query --query=archiveTableRecord
--lgp_ip=127.0.0.1 --lgp_port=30003 --output_file=/tmp/lgp_query.xml
```

Creates the file `/tmp/lgp_query.sql`.

6.7.3 tp_lgp_archive_online

The tp_lgp_archive_online tool allows you to archive or unarchive the LGP tables based on specified date range.

Note: Before using this tool, make sure that the archiving has been disabled from MGR/config files.

Key features for this tool are discussed underneath:

- Can be executed from cronjob.
- In multi-instance setup, separate instance of tool must be executed for each LGP instance.
- Does not impact normal LGP working. Hence loading can happen in parallel.
- Should be executed in low traffic period. In case of high traffic period, LGP loading and tool execution in parallel can lead to high disk I/O utilization.
- Reads database keys from config files. If there is a change in database keys, the change is reflected only on the tables processed from the next execution.
- Exits if gets any error in table processing.

6.7.3.1 Synopsis

```
./tp_lgp_archive_online  --start_date=timestamp --end_date=timestamp [--archive]
[--innodb]
[--maximum_days] [--nocheck] [--dbuser=string] [--dbpwd=string] [--host=hostname|IP]
 [--db=string][--port=number]
```

6.7.3.2 Options

Tool	Description
-- start_date	This option indicates start date of the table records. Records are fetched from start date to end date. This is a mandatory parameter. Format: YYYYMMDD/20121009 .
--end_date	This option indicates end of the table records. Records are fetched from start date to end date. This is a mandatory parameter. Format: YYYYMMDD/20121109.
--archive	This option indicates that the specified MySQL tables are to be archived. This cannot be used with -innodb option.
--innodb	This option indicates that the specified MySQL tables are to be unarchived. This cannot be used with -archive option.
-- maximum _days	This option indicates number of days for which tables will be processed in one execution cycle. **Note:** Default value is 0. For all the values less than or equal to 0. It will process all the tables in specified date range.

Tool	Description
--nocheck	This option indicates that no validation is required on the input provided by user. It is mandatory to use this option while executing the tool in cronjob.
--dbuser	This option indicates username to connect to the LGP database.
--dbpwd	This option indicates password to connect to the LGP database.
--host	This option indicates hostname of LGP Server.
--db	This option indicates name of LGP database.
--port	This option indicates MySQL Server listen port.
--help	This option displays help.

6.7.3.3 Example

1) The following command archives all the LGP tables between start date and end date.

```
tp_lgp_archive_online --start_date=20150427 --end_date=20150429 --archive
```

OUTPUT:

```
[textpass@spf53 ~]$ tp_lgp_archive_online --start_date=20150427 --end_date=20150429
 --archive
(PERL) Utility for converting LGP tables into Archive or InnoDB engine format.
Please confirm the database settings:
LGP Database Username:         LGPuser
LGP Database Password:         LGPuser123
LGP Database Name:             lgp
LGP Database portnumber:       3306
LGP Start Time:                20150427
LGP End Time:                  20150429
MySQL Engine:                  ARCHIVE

Is this information correct? [Y/n] >> y
Processing lgp_inb_msg_20150427
Archiving of lgp_inb_msg_20150427 Successful!
Processing lgp_inb_msg_20150428
Archiving of lgp_inb_msg_20150428 Successful!
Processing lgp_inb_msg_20150429
Archiving of lgp_inb_msg_20150429 Successful!
```

2) The following command unarchives all the LGP tables between start date and end date.

```
tp_lgp_archive_online --start_date=20150427 --end_date=20150429 --innodb
```

OUTPUT:

```
[textpass@spf53 ~]$ tp_lgp_archive_online --start_date=20150427 --end_date=20150429
 --innodb
(PERL) Utility for converting LGP tables into Archive or InnoDB engine format.
Please confirm the database settings:
LGP Database Username:         LGPuser
LGP Database Password:         LGPuser123
LGP Database Name:             lgp
LGP Database portnumber:       3306
LGP Start Time:                20150427
LGP End Time:                  20150429
MySQL Engine:                  InnoDB
Is this information correct? [Y/n] >> y
```

```
Processing lgp_inb_msg_20150429
 unarchiving of lgp_inb_msg_20150429 successful!
Processing lgp_inb_msg_20150428
 unarchiving of lgp_inb_msg_20150428 successful!
Processing lgp_inb_msg_20150427
 unarchiving of lgp_inb_msg_20150427 successful!
```

3) The following command unarchives all the LGP tables of next two days (i.e. --maximum_days=2) from the start date.

OUTPUT:

```
[textpass@krakatau ~]$ tp_lgp_archive_online --start_date=20120419 --end_date=20160422
 --innodb
--maximum_days=2 --dbuser=LGPuser -dbpwd=LGPuser123 --host=localhost --db=lgp
--port=3306
(PERL) Utility for converting LGP tables into Archive or InnoDB engine format.
Please confirm the database settings:
LGP Database Username:         LGPuser
LGP Database Password:         LGPuser123
LGP Database Name:             lgp
LGP Database portnumber:       3306
LGP Start Time:                20120419
LGP End Time:                  20160422
MySQL Engine:                  InnoDB
Is this information correct? [Y/n] >> y
Processing lgp_evt_sm_endpoints_20150420 table
 unarchiving of lgp_evt_sm_endpoints_20150420 successful!
Processing lgp_evt_sm_finalstatus_20150420 table
 unarchiving of lgp_evt_sm_finalstatus_20150420 successful!
Processing lgp_evt_sm_endpoints_20141106 table
 unarchiving of lgp_evt_sm_endpoints_20141106 successful!
Processing lgp_evt_sm_finalstatus_20141106 table
 unarchiving of lgp_evt_sm_finalstatus_20141106 successful!
Processing lgp_evt_sm_gsm_orig_in_request_20141106 table
 unarchiving of lgp_evt_sm_gsm_orig_in_request_20141106 successful!
Processing lgp_evt_sm_gsm_orig_in_response_20141106 table
 unarchiving of lgp_evt_sm_gsm_orig_in_response_20141106 successful!
```

4) The following command archives all the LGP tables of 27th April 2015(i.e. only single day LGP tables) where 'start_date' is equals to 'end_date'.

```
tp_lgp_archive_online --start_date=20150427 --end_date=20150427 --archive
```

OUTPUT:

```
[textpass@spf53 ~]$ tp_lgp_archive_online --start_date=20150427 --end_date=20150427
 --archive
(PERL) Utility for converting LGP tables into Archive/InnoDb engine format.
First please confirm the database settings:
LGP Database Username:         LGPuser
LGP Database Password:         LGPuser123
LGP Database Name:             lgp
LGP Database portnumber:       3306
LGP Start Time:                20150427
LGP End Time:                  20150427
MySQL Engine:                  ARCHIVE
Is this information correct? [Y/n] >> y
Processing lgp_inb_msg_20150427
Archiving of lgp_inb_msg_20150427 Successful!
```

5) Using tp_lgp_archive_online tool in cronjob:

The following command executes this script everyday at 00:05 Hrs . This archives all the tables of next two days (i.e. -maximum_days=2) from the start date.

For example:

Edit the crontab with following command:

```
#crontab -e
```

Add the below line in crontab:

```
5 * * * * su - textpass -c "tp_lgp_archive_online --start_date=20150427
--end_date=20150429
--archive -maximum_days=2 --nocheck" >/dev/null 2>&1
```

6) Using tp_lgp_archive_online tool in cronjob:

For multi-instance configuration, you need to run this script for every user.

The following command executes this script for tpuser01 everyday at 00:05 Hrs. This archives all the tables of next two days (i.e. -maximum_days=2) from the start date.

For example:

Edit the crontab with following command:

```
#crontab -e
```

Add the below line in crontab:

```
5 * * * * su - tpuser01 -c "tp_lgp_archive_online --start_date=20150427
--end_date=20150429
--archive -maximum_days=2 --dbuser=LGPuser1 -dbpwd=LGPuser1234 --host=localhost
--db=lgp_1
--port=3306 --nocheck” >/dev/null 2>&1
```

7) While doing data unarchiving using this tool, if LGP contains archived tables beyond the end date specified, the tool exits with below error:

Table(s) beyond to end date in lgp database are still in Archive Engine.

The following command unarchives all the LGP tables of 27th April 2015:

```
tp_lgp_archive_online --start_date=20150427 --end_date=20150427 --innodb
OUTPUT:
[textpass@spf53 ~]$ tp_lgp_archive_online --start_date=20150427 --end_date=20150427
 --innodb

(PERL) Utility for converting LGP tables into Archive or InnoDB engine format.

Please confirm the database settings:
                LGP Database Username:         LGPuser
                LGP Database Password:         LGPuser123
                LGP Database Name:             lgp
                LGP Database portnumber:       3306
                LGP Start Time:                20150427
                LGP End Time:                  20150427
                MySQL Engine:                  InnoDB


 Is this information correct? [Y/n] >> y
Table(s) beyond to end date in lgp database are still in Archive Engine.
```

8) While doing data archiving using this tool, if LGP contains unarchived tables prior to the start date specified, the tool exits with below error:

Table(s) prior to start date in lgp database are still in InnoDB Engine.

The following command archives all the LGP tables of 27th April 2015:

```
tp_lgp_archive_online --start_date=20150427 --end_date=20150427 --archive
OUTPUT:
[textpass@spf53 ~]$ tp_lgp_archive_online --start_date=20150427 --end_date=20150427
 --archive

(PERL) Utility for converting LGP tables into Archive or InnoDB engine format.

Please confirm the database settings:
                LGP Database Username:         LGPuser
                LGP Database Password:         LGPuser123
                LGP Database Name:             lgp
                LGP Database portnumber:       3306
                LGP Start Time:                20150427
                LGP End Time:                  20150427
                MySQL Engine:                  ARCHIVE


 Is this information correct? [Y/n] >> y
Table(s) prior to start date in lgp database are still in InnoDB Engine.
```

6.8 Commands for Troubleshooting

The following operating system tools are available for troubleshooting:

Tool	Description
`gcore`	Generates a core file
`hostid`	Provides the host ID of the system (required for a license)
`ifconfig`	Provides an overview of the IP configuration
`netstat`	Provides an overview of IP network statistics
`ping`	An IP diagnostic command
`top`	Shows statistics about active processes
`ps`	Provides information about processes
`sar`	Provides performance data
`tcpdump`	Trace network traffic on TCP/IP level

Refer to the Red Hat Enterprise Linux documentation at *http://www.redhat.com/docs/*.

6.9 Log Record Archive & Purging

LGP supports following two mechanisms to manage old records:

- Archiving: Archiving allows user to Archive LGP Tables older than configured number of hours.
- Cleanup: Cleanup allows user to Clean LGP Tables older than configured number of hours.

The above two options can be used to increase retention time of log records and purge old records.

Parameter	Configuration	Usage
Convert To Archive Storage After	Dynamic configuration, can be configured in MGR GUI (**Logging ➤ Properties**).	Files that were loaded into the DB more than this time ago shall be archived at the next DB archive schedule. Default value is `-1` which indicate that no archiving will be performed. The value should be greater than or equal to 24 hours to perform the archiving. The maximum allowed value is 26304 hours (1096 days). The `Convert To Archive Storage After` parameter should be configured with a value lower than the configured value of `lgpdbcleanolderthan`; otherwise no archiving would be performed, instead old records would be directly deleted.
lgpdbcleanolderthan	Semi-static configuration.	Maximum table age (in hours) in the database. Files loaded before `lgpdbcleanolderthan` hours are deleted at the next database clean.

LGP converts the log record tables older than the configured value of "`Convert To Archive Storage After`" into MySQL Archive storage engine.

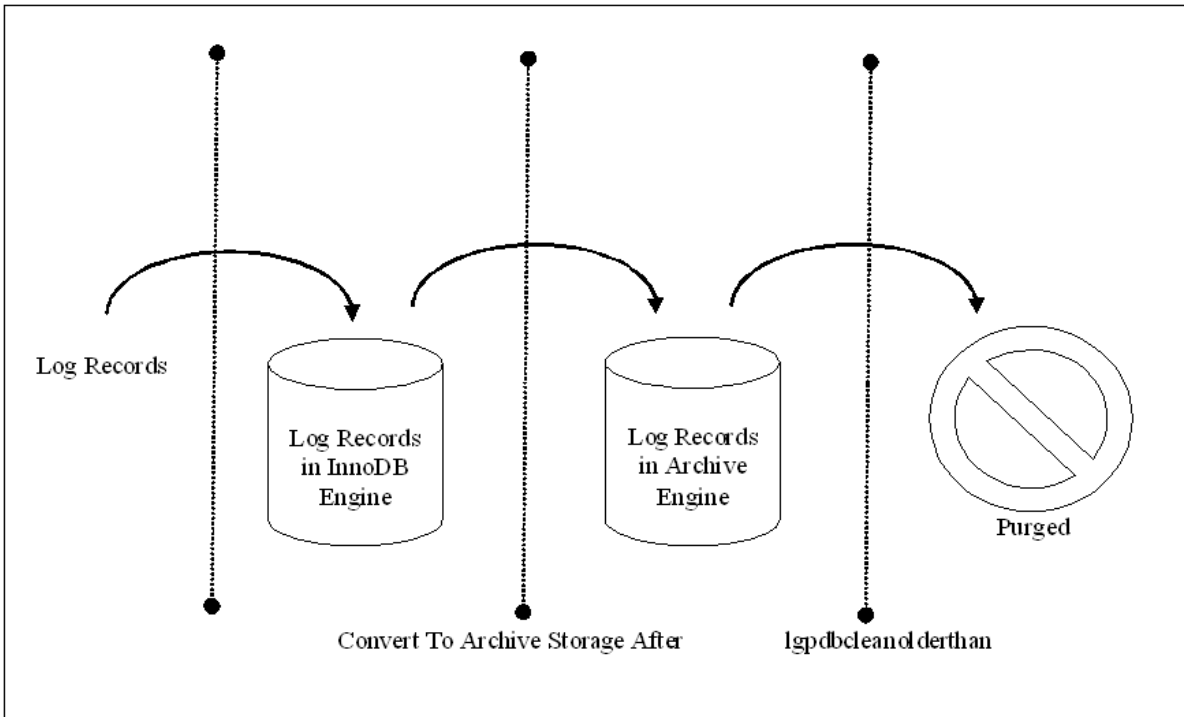

Figure 4: Log Record Archive & Purging

6.9.1 Disk Space Usage in Archive Storage Engine

MySQL Archive storage engine stores the data in ZLIB format which consumes very less disk space as compared to default InnoDB engine. Archiving helps in retaining the log records for a longer duration.

6.9.2 Query Log Records Stored In Archive Storage Engine

The log record query on data stored in Archive storage engine takes significantly longer time than logs records stored in InnoDB storage engine.

Background Query can be used to perform long running queries on Archive tables. Refer to the MGR Operator Manual for more information about the "Background Query".

6.9.3 Conversion from Innodb to Archive

Convert To Archive Storage After and **Database Archive Schedule** can be configured on MGR GUI to schedule conversion of tables to Archive format.

Note: Conversion from Innodb to Archive takes significant time and also LGP Loader and Query module are impacted by it, so archival must be scheduled at off-peak time.

6.9.4 Conversion from Archive to InnoDB

To convert the tables back to InnoDB storage engine, increase the value of **Convert To Archive Storage After** on MGR GUI to schedule conversion of tables to InnoDB format. If you want to disable the archiving feature then set the value of **Convert To Archive Storage After** to `-1`. LGP will convert all archive tables into InnoDB on next **Database Archive Schedule** configured time.

Note:

1. Conversion from Archive to InnoDB takes significant time and also LGP Loader and Query module are impacted by it, so un-archival must be scheduled at off-peak time
2. Conversion from Archive to InnoDB require significant disk space in database partition as InnoDB tables takes much more space as compared to Archive tables. You must ensure that disk partition have enough space to convert all the tables into InnoDB engine otherwise LGP will raise `archiveFailed` trap.

6.9.5 Purging of Log Records

LGP purges the log record (drop tables) older than the configured value of `lgpdbcleanolderthan` to clean-up the disk space.

6.10 Flexible User Data Text Search

LGP supports efficient and enhanced searching on the user data text fields by leveraging the MySQL Full Text Search (FTS) capability for the InnoDB engine. This flexible search functionality needs to be configured and triggered from the LGV, and it is allowed only if the **Enable Full Text Search** license

option is enabled and the semi-static configuration parameter lgpfulltextsearch is set to “true”. Refer to the MGR Operator Manual for more details on configuring the flexible text search.

In order to support flexible user data search, LGP first carries out pre-processing of the user data contained in the fields

“smsSubmit_smsUserData”, “smsDeliver_smsUserData” and “messageFields_userData”.

The pre-processing involves tokenization and normalization of the user data text as per the following main steps:

1. *Replace word boundaries with the white-space character*
2. *Remove the stop words*
3. *Tokenize each character according to the normalization map*

After the pre-processing is completed, the tokenized and normalized version of the user data text is stored in the LGP database under the attribute name: “userData_normalizedText”. This attribute is added as a new column in the ‘lgp_inb_msg’ table schema.

If the flexible text search functionality is enabled and an appropriate query is triggered from the LGV, then the corresponding query text is also pre-processed by LGP in exactly the same manner as was earlier done for the user data text, and the resulting tokenized and normalized query text string is actually used for searching the database.

Note:

1. For the flexible text search feature, conversion of the original text characters from UTF-8 and UCS2 to Unicode, as well as reverse conversion of normalized text characters from Unicode to UTF-8, is supported.
2. For 8-bit encoded binary data, instead of pre-processing and normalizing the user data it is converted to hexadecimal format and copied unchanged to the “userData_normalizedText” column.
3. LGP Query for "match_flexibly" condition(in LGV) is not supported for gsm 8-bit encoded binary data which is stored in LGP database table in hex form.
4. The flexible user data text search functionality will not work for the messages with encrypted user data.

6.10.1 Replace word boundaries with the white-space character

The word boundary can be any 8-bit/16bit UTF-8 character that is defined as word boundary by the configuration. The following characters are considered word boundaries in most languages and by default; the LGP will consider these as word boundary:

- Start of text (SOT) and End of text (EOT);
- White space;
- Carriage return (CR) and Line Feed (LF);
- New line character: New Line (NEL);
- Control characters of ASCII and extended ASCII set (0x00,0x1f and 0x7f,0x9f];
- Other not-printable ASCII characters;
- Punctuation used in most Latin languages as word breaks: ~`!@#$%^&*()+-_=[]{}\|:";'<>?,./

It is possible to define a customized word boundary set using the semi-static parameter “wordboundaries”. When not provisioned, the default word boundary set will be used.

If the configuration is incorrect, a syslog message "Error while converting the word boundaries from UTF-8 to Unicode" will appear.

Refer *Sample Common Configuration File* for an example of default word boundaries configuration.

To view the current word boundaries on a LGP system, execute:

```
tp_walk --tp_lgp lgpPropertiesWordBoundaries
```

Note:

1. By default A_B is considered two words A and B. This is controlled by the word boundary set. If the "_" is not in the customized set, A_B is considered one word.

2. If the user does not configure the word-boundaries in the common configuration file or the host-configuration file then the default word boundaries set in the LGP will be used.

6.10.2 Remove the stop words

The stop words can be any 8-bit/16bit UTF-8 character that is defined as stop words by the configuration.

Any word that is defined in the stop words list is removed from the white-space delimited user data obtained as a result of the previous step.

The default set of stop words defined are:

```
"a,about,an,are,as,at,be,by,com,de,en,for,from,how,i,in,is,
it,la,of,on,or,that,the,this,to,was,what,when,where,who,will,
with,und,the,www"
```

To view the current stop words on a LGP system, execute:

```
tp_walk --tp_lgp lgpPropertiesStopWords
```

It is possible to define a customized stop words set using the semi-static parameter “stopwords”. When not provisioned, the default stop words set will be used.

If the configuration is incorrect, a syslog message "Error while converting the stop words from UTF-8 to Unicode" will appear.

Refer to the LGP Flexible Text Search User Manual for more details.

Refer to *Sample Common Configuration File* for an example of default stop words configuration.

6.10.3 Tokenize each character according to the normalization map

After the steps defined in sections 6.11.1 and 6.11.2 have been performed, tokenization of the characters contained in the user data is carried out.

During tokenization, similar characters are mapped to the same token. The characters that are mapped to respective token can be set using the semi-static configuration attribute “normalisationmap”.

Note: Tokenization automatically removes any character that is not included in the configured tokenization map.

6.10.3.1 Format of the tokenization map

The format of the “normalisationmap” attribute is a list of character groups separated by the new line character (
).

All characters of the first character group are mapped to the token which is the 1st character of that group, all characters of the second character group are mapped to the 1st character of that group, and so on. All characters that are not specified in the map are dropped during tokenization.

Specify international characters using the decimal HTML encoding of the Unicode character. Refer to:

http://www.utf8-chartable.de/unicode-utf8-table.pl?unicodeinhtml=dec for character codes

For example, assume that:

- Characters a, A, and a-umlaut (ä) should be mapped to token a
- Characters b, B, and 6 should be mapped to token b
- Character C should be mapped to token C

The character code for a-umlaut is 228. Therefore, the configuration file should contain:

normalisationmap="aAä
bB6
C"

Refer to the LGP Flexible Text Search User Manual for more details.

Note: Changes to the normalisationmap parameter will overwrite the LGP's default mapping. If you want to append to the default tokenization map, ensure that you preserve it when you modify the normalisationmap attribute.

If the configuration is incorrect, a syslog message "Error while converting the normalization map from UTF-8 to Unicode" will appear.

To view the current tokenisation mapping on a LGP system, execute:

```
tp_walk --tp_lgp lgpPropertiesNormalise
```

Refer to *Sample Configuration File* for an example of default normalization map configuration.

6.10.3.2 Tokenization and Normalization examples

These examples of tokenisation are based on the default tokenisation map:

1. The string "many dollars" is tokenised into:

```
m4ny d0114r5
```

2. The string "Ellen" is tokenised into:

```
3113n
```

After tokenization, LGP applies normalization, which collapses multiple identical tokens into a single token.

For example:

1. "**m4ny d0114r5**" will become "**m4ny d014r5**"

2. "**3113n**" will become "**313n**"

Chapter 7

Trap Service

Topics:

7.1 Introduction

Up to eight SNMP managers can subscribe to the LGP's trap service. Whenever a trap condition occurs, the LGP will send an SNMP trap to any SNMP management station that is subscribed to the trap service.

To subscribe an SNMP manager to the trap service, add an entry to the Alarm Station Table that contains the IP address (IPv4 or IPv6) or Hostname) of the SNMP manager and a UDP port number to which SNMP traps should be sent. The Alarm Station Table is also SNMP manageable; refer to the `TEXTPASS-GEN` MIB for more information about this table.

The LGP always originates SNMP traps from UDP port 11173 and terminates them in the UDP ports that are specified in the Alarm Station Table. The community string that the LGP specifies in SNMP traps is always equal to `public`.

The LGP uses an SMNP trap daemon to log generated SNMP traps locally in `/var/log/messages`.The daemon uses UDP port 11173.

Note:

1. If '`snmpPropAlarmOwnIpv6Address`' parameter in semi-static configuration file is set, then specified address will be used as source address for sending SNMP traps to SNMP Manager with address of type IPv6.
2. If '`snmpPropAlarmOwnIpAddress`' parameter in semi-static configuration file is set, then specified address will be used as source address for sending SNMP traps to SNMP Manager with address of type IPv4.

7.2 SNMP Manager

For configuration and monitoring purposes, an SMNP Manager or Management Station issues SNMPv1 requests to the LGP. SNMP Manager should send such requests to UDP port 11961 of the LGP. The LGP does not enforce an SNMP Manager to originate requests from any specific UDP port (any UDP port can be used for this purpose).

The device requires an SNMP Manager to use a community string equal to:

- `public` for read operations
- `private` for set operations

The LGP silently discards any request that does not satisfy these requirements.

Note: If '`snmpPropListenAddressType`' parameter in semi-static configuration file is set to 'dual', then LGP will accept requests on both IPv4 and IPv6 .

7.3 LGP Traps

The following traps are related to the LGP. They are defined in the `TEXTPASS-LGP` MIB.

Trap	Severity	Description
`controllerFailed`	Error	The LGP controller module encountered a fatal error condition.
`loaderFailed`	Error	The DB Loader component encountered a fatal error condition.
`queuesLowLoad`	Warning	The load time (estimated by dividing the size of the files on disk waiting to be loaded by the average loading speed in bytes per second for the last five minutes) is greater than the maximum configured acceptable load delay.
`masterFailed`	Error	The poll master module encountered a fatal error condition.
`collectorFailed`	Error	The poll collector module encountered a fatal error condition.
`queryFailed`	Error	The database query module encountered a fatal error condition.
`archiveFailed`	Critical	The archiving or un-archiving module encountered a Fatal error condition.

7.4 License Traps

Refer to *License Warnings*.

7.5 File Transfer Traps

The following traps are related to file transfer.

Trap	Severity	Description
`collectorError`	Error	The poll collector module encountered a non-fatal error condition (for example, the connection to one of the RTR nodes failed).
`masterError`	Error	The poll master module encountered a non-fatal error condition (for example, the connection to one of the RTR nodes failed).

7.6 Trap Configuration

The `lgpalertdelay` attribute in the XML-based configuration file defines the number of minutes to wait before retransmitting a loader trap. See *Configuration* for more information.

7.7 Trap Filtering

To prevent flooding of trap receivers, Mobile Messaging SNMP traps can be filtered per trap receiver, through a combination of blacklisting and white-listing. The blacklist and white list are configured in two tables, associated with the Alarm Stations Table:

- The white list table contains a list of all traps that should be sent toward an alarm station (wildcards are allowed);
- The blacklist table contains a list of traps that should be blocked for a particular alarm station (wildcards are allowed).

Note: The white list is applied before the blacklist. An empty white list is identical to a white list of a single wildcard ("*"). An empty blacklist does not block any trap.

7.8 SNMP Trap Reference

Refer to the NewNet Mobile Messaging SNMP Trap Reference Guide for a complete overview of the Mobile Messaging SNMP traps. The SNMP Trap Reference Guide is available on the Mobile Messaging documentation CD.

Chapter 8

Security

Topics:

8.1 Introduction

This chapter provides an overview of security aspects of the LGP.

8.2 Controlling System Access

Command line access to Mobile Messaging systems is controlled using the available operating system security mechanisms. Please refer to the operating system documentation for more information.

8.3 Access to SMS User Data

The ability to view decoded SMS user data (ASCII format) is subject to the RTR's privacy license setting. To allow for maximum flexibility and privacy protection, the maximum logging level is determined (and locked) by the license key.

Chapter 9

Software License

Topics:

9.1 Introduction

Some Mobile Messaging software components are licensed features; therefore, the appropriate software licenses must be purchased before the corresponding functionality can be used.

This chapter discusses commercially licensed features, how to check your licenses, and how to install new Mobile Messaging licenses.

9.2 Licensed Items

The following LGP licensed items are available:

Licensed Item	Possible Values	M/O
LGP product enabled	Enabled/Disabled	M/O
Short message event logging	Enabled/Disabled	M/O
Enable Full Text Search	Enabled/Disabled	M/O

9.2.1 Multi-Instance License

Multiple instances feature allows you to run multiple LGPs (up to 10 instances) on the same node. Multi-instance license should be enabled for NMM user to run one additional instance of LGP. To run one additional instance of LGP from newly created NMM user (using script `tp_manage_user`), instance license for newly create user id (operating system user identifier) should be enabled in license file.

9.3 License Behaviour

9.3.1 Functional License

All licensed items are subject to a specification in the LGP license key.

If a licensed item is not specified in the license, the functionality is not available on the LGP or in the MGR.

9.3.2 Capacity License

The LGP depends on the capacity license of the RTR/FWL and does not have a capacity license itself.

9.4 Checking Your License

To view the current license values, execute the following command on the command line:

```
$ tp_system [system]
```

Where `[system]` is a resolvable host name or the IP address of an LGP system (if omitted, `localhost` is used).

The following example shows a sample output of the `tp_system` command. The license key and license information are provided.

Example:

```
$ tp_system --tp_lgp

Identification:
    TextPass/LGP R02.08.04.00
    Linux 3.10.0-862.14.4.el7.x86_64  #1 SMP Fri Sep 21 09:07:21 UTC 2018
    Linux build

Uptime:
    0 days 10h:37m:50s

License key:
    XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX


License information:
    License version 15
    License number TST-93813
    Hardware ID 1823e946
    Ext Serial No SGH820TXXN
    Instance Serial No 200
    License exceeds in 2113 hours
    License exceeds at Sun Mar 10 00:00:00 2019
    License issue number 14
    Hide User Data in Events disabled
    VmWare Support disabled
    Core Count 0
    Encrypt User Data disabled
    Short Message Event Logging enabled
    Advanced Reporting enabled
    Enable Full Text Search enabled

Alarm stations:
    127.0.0.1:11173
```

If multi-instance license is enabled for user id `200` (`'textpass'`) in the license file, then `tp_system` will display the instance user id as shown below:

```
        Instance Serial No 200
```

To enable the full-text feature the license parameter “Enable Full Text Search” should be enabled:

```
        Enable Full Text Search enabled
```

9.5 License Warnings

The following SNMP traps are available to warn when the license limits are exceeded:

- `licenseWillExpire`—The temporary license will expire in the given number of hours
- `licenseExpired`—The license validity period has expired

The license expiration traps are generated at these intervals:

- `licenseWillExpire`—Before it expires, 14 days in advance
- `licenseWillExpire`—7 days in advance
- `licenseWillExpire`—3 days in advance
- `licenseWillExpire`—1 day in advance
- `licenseExpired`—At the moment it expires, and every hour until fixed

Note: To adapt the license in a timely manner and avoid expiry, traps should be properly handled by the Network Management System.

Appendix A

Sample Log Records

Topics:

A.1 Trusted MoFwdSm with Country and Network Information

The following sample log file contains one `trustedMoFwdSmWithCountryAndNetworkInfo` record.

```
File: log_testermachine_31_20130508_110042_447.dat, record: 1

          7f 64 81 e1  80 13 32 30  31 33 30 35  30 38 31 31  30 31 34 30
          2d 30 34 30  30 81 01 02  a3 0e 80 01  00 81 09 52  6f 75 74 65
          54 6f 4d 53  a8 0a 80 04  43 e1 10 08  81 02 64 65  a9 08 80 02
          42 08 81 02  64 65 ac 0d  80 07 91 94  71 02 94 09  41 81 02 64
          65 ad 0d 80  07 91 94 88  68 66 35 93  81 02 64 65  ae 0e 80 08
          62 02 92 64  16 65 59 f1  81 02 64 65  af 39 80 02  00 00 81 02
          00 ab a2 0d  80 07 91 94  09 51 44 66  40 81 02 64  65 83 01 00
          84 01 00 85  13 32 30 31  33 30 35 30  39 31 31 30  31 34 30 2d
          30 34 30 30  87 05 48 65  6c 6c 6f 93  02 10 e1 b5  34 80 01 00
          a2 23 80 01  00 a1 0e 80  08 62 02 02  69 68 33 78  f9 81 02 64
          65 a3 0e 80  08 91 94 27  99 99 99 19  11 81 02 64  65 83 01 00
          a5 03 80 01  00 8a 02 04  d2

7f 64 81 e1      trustedMoFwdSmWithCountryAndNetworkInfo = {
80 13 32 30 31 33 30 35 30 38 31 31 30 31 34 30 2d 30 34 30 30
                   timestamp = '20130508110140-0400'
81 01 02           routingAction = routeToMs(2)
a3 0e              responseInfo = {
80 01 00             submissionResult = success(0)
81 09 52 6f 75 74 65 54 6f 4d 53
                     moRoutingRule = 'RouteToMS'
                   }
a8 0a              sccpCgPa = {
80 04 43 e1 10 08    sccpAddress = {
                       reservedForNationalUse = 0
                       routingIndicator = ssn
                       globalTitleIndicator = 0
                       subsystemNumberIndicator = present
                       signallingPointCodeIndicator = present
                       signallingPointCode = 4321
                       subsystemNumber = 8
                     }
81 02 64 65          country = 'de'
                   }
a9 08              sccpCdPa = {
80 02 42 08          sccpAddress = {
                       reservedForNationalUse = 0
                       routingIndicator = ssn
                       globalTitleIndicator = 0
                       subsystemNumberIndicator = present
                       signallingPointCodeIndicator = absent
                       subsystemNumber = 8
                     }
81 02 64 65          country = 'de'
                   }
ac 0d              mapSmsc = {
80 07 91 94 71 02 94 09 41
                     gsmAddress = 'international/telephone 491720499014'
81 02 64 65          country = 'de'
                   }
ad 0d              mapMsisdn = {
80 07 91 94 88 68 66 35 93
                     gsmAddress = 'international/telephone 498886665339'
81 02 64 65          country = 'de'
                   }
ae 0e              mapImsi = {
```

```
80 08 62 02 92 64 16 65 59 f1
                      imsi = 262-02-9466156951
81 02 64 65           country = 'de'
                    }
af 39               smsSubmit = {
80 02 00 00           smsServices =  {
                        replyPath(0) = false(0)
                        userDataHeaderIndication(1) = false(0)
                        statusReportRequest(2) = false(0)
                        rejectDuplicates(5) = false(0)
                      }
81 02 00 ab           smsMessageReference = 171
a2 0d                 smsRecipient = {
80 07 91 94 09 51 44 66 40
                        gsmAddress = 'international/telephone 499015446604'
81 02 64 65             country = 'de'
                      }
83 01 00              smsProtocolId = 0
84 01 00              smsDataCodingScheme = 0
85 13 32 30 31 33 30 35 30 39 31 31 30 31 34 30 2d 30 34 30 30
                      smsValidityPeriod = '20130509110140-0400'
87 05 48 65 6c 6c 6f
                      smsUserData = Hello
                    }
93 02 10 e1         originatingPointCode = 4321
b5 34               outboundMt = {
80 01 00              sriSmRoutingAction = pass(0)
a2 23                 sriSmResponseInfo = {
80 01 00                queryResult = success(0)
a1 0e                   mapImsi = {
80 08 62 02 02 69 68 33 78 f9
                          imsi = 262-02-0968633879
81 02 64 65               country = 'de'
                        }
a3 0e                   mapMsc = {
80 08 91 94 27 99 99 99 19 11
                          gsmAddress = 'international/telephone 49729999999111'
81 02 64 65               country = 'de'
                        }
                      }
83 01 00              mtFwdSmToMscRoutingAction = pass(0)
a5 03                 mtFwdSmToMscResponseInfo = {
80 01 00                deliveryResult = success(0)
                      }
8a 02 04 d2           destinationPointCode = 1234
                    }
                  }
```

A.2 Suspect MoFwdSm with Country and Network Information

The following sample log file contains one `suspectMoFwdSmWithCountryAndNetworkInfo` record.

```
File: /log1.asd.mbalance.com_20070411_131145_003.dat, record: 1

          7f 65 82 01  58 80 13 32  30 30 37 30  34 31 31 31  33 31
31 35
          30 2b 30 32  30 30 81 01  05 a2 08 80  01 11 81 03  61 62
63 84
          05 00 00 00  00 00 a8 1a  80 0b 12 95  00 11 04 13  56 05
00 00
          00 81 02 6e  6c 82 07 74  65 6c 66 6f  72 74 a9 0d  80 0b
```

```
12 08
          00 11 04 94  56 15 32 54  06 ac 12 80  07 91 13 56  93 99
99 f9
          81 02 6e 6c  82 03 6b 70  6e ad 0d 80  07 91 13 56  14 32
54 f6
          81 02 6e 6c  ae 15 80 06  02 14 12 32  54 f6 81 02  6e 6c
82 07
          74 65 6c 66  6f 72 74 b0  7f 80 02 00  00 81 01 00  82 01
00 83
          01 00 84 01  00 a5 0d 80  07 91 23 56  16 32 54 f6  81 02
62 65
          86 5e 61 61  61 61 61 61  61 61 61 61  61 61 61 61  61 61
61 61
          61 61 61 61  61 61 61 61  61 61 61 61  61 61 61 61  61 61
61 61
          61 61 61 61  61 61 61 61  61 61 61 61  61 61 61 61  61 61
61 61
          61 61 61 61  61 61 61 61  61 61 61 61  61 61 61 61  61 61
61 61
          61 61 61 61  61 61 61 61  61 61 61 61  61 61 61 61  be 47
a0 15
          80 06 02 14  12 32 54 f6  81 02 6e 6c  82 07 74 65  6c 66
6f 72
          74 a1 16 80  07 91 13 56  95 99 99 f9  81 02 6e 6c  82 07
74 65
          6c 66 6f 72  74 a2 16 80  07 91 13 56  05 00 00 f0  81 02
6e 6c
          82 07 74 65  6c 66 6f 72  74

7f 65 82 01 58   suspectMoFwdSmWithCountryAndNetworkInfo = {
80 13 32 30 30 37 30 34 31 31 31 33 31 31 35 30 2b 30 32 30 30
                   timestamp = '20070411131150+0200'
81 01 05           routingAction = discardWithAck(5)
a2 08              rejectInfo = {
80 01 11             rejectCause = matchingMoRoutingRule(17)
81 03 61 62 63       moRoutingRule = 'abc'
                   }
84 05 00 00 00 00 00
                   ignoredRejectCauses =  {
                     spoofedOriginatorAddress(19) = false(0)
                   }
a8 1a              sccpCgPa = {
80 0b 12 95 00 11 04 13 56 05 00 00 00
                     sccpAddress = {
                       reservedForNationalUse = 0
                       routingIndicator = gt
                       globalTitleIndicator = 4
                       subsystemNumberIndicator = present
                       signallingPointCodeIndicator = absent
                       subsystemNumber = 149
                       translationType = 0
                       numberingPlan = isdnTelephony
                       encodingScheme = bcdOddNumberOfDigits
                       natureOfAddressIndicator = international
                       addressInformation = 31655000000
                     }
81 02 6e 6c          country = 'nl'
82 07 74 65 6c 66 6f 72 74
                     network = 'telfort'
                   }
a9 0d              sccpCdPa = {
80 0b 12 08 00 11 04 94 56 15 32 54 06
                     sccpAddress = {
                       reservedForNationalUse = 0
                       routingIndicator = gt
```

```
                       globalTitleIndicator = 4
                       subsystemNumberIndicator = present
                       signallingPointCodeIndicator = absent
                       subsystemNumber = 8
                       translationType = 0
                       numberingPlan = isdnTelephony
                       encodingScheme = bcdOddNumberOfDigits
                       natureOfAddressIndicator = international
                       addressInformation = 49655123456
                     }
                   }
ac 12              mapSmsc = {
80 07 91 13 56 93 99 99 f9
                     gsmAddress = 'international/telephone
31653999999'
81 02 6e 6c          country = 'nl'
82 03 6b 70 6e       network = 'kpn'
                   }
ad 0d              mapMsisdn = {
80 07 91 13 56 14 32 54 f6
                     gsmAddress = 'international/telephone
31654123456'
81 02 6e 6c          country = 'nl'
                   }
ae 15              mapImsi = {
80 06 02 14 12 32 54 f6
                     imsi = 204-12-123456
81 02 6e 6c          country = 'nl'
82 07 74 65 6c 66 6f 72 74
                     network = 'telfort'
                   }
b0 7f              smsCommand = {
80 02 00 00          smsServices =  {
                       userDataHeaderIndication(1) = false(0)
                       statusReportRequest(2) = false(0)
                     }
81 01 00             smsMessageReference = 0
82 01 00             smsProtocolId = 0
83 01 00             smsCommandType = 0
84 01 00             smsMessageNumber = 0
a5 0d                smsRecipient = {
80 07 91 23 56 16 32 54 f6
                       gsmAddress = 'international/telephone
32656123456'
81 02 62 65            country = 'be'
                     }
86 5e 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
61 61 61
61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
61 61 61
61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
61 61 61
61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61
                     smsCommandData =
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaa
                   }
be 47              infoFromHlr = {
a0 15                mapImsi = {
80 06 02 14 12 32 54 f6
                       imsi = 204-12-123456
81 02 6e 6c            country = 'nl'
82 07 74 65 6c 66 6f 72 74
                       network = 'telfort'
                     }
```

```
a1 16                 mapMsc = {
80 07 91 13 56 95 99 99 f9
                        gsmAddress = 'international/telephone
31655999999'
81 02 6e 6c             country = 'nl'
82 07 74 65 6c 66 6f 72 74
                        network = 'telfort'
                      }
a2 16                 mapSgsn = {
80 07 91 13 56 05 00 00 f0
                        gsmAddress = 'international/telephone
31655000000'
81 02 6e 6c             country = 'nl'
82 07 74 65 6c 66 6f 72 74
                        network = 'telfort'
                      }
                    }
                  }
```

A.3 Trusted SriSm with Country and Network Information

The following sample log file contains one `trustedSriSmWithCountryAndNetworkInfo` record.

```
File: logging_Srism_Mtor_djerba_200_20150415_113612_110.dat, record: 1

          7f 66 81 9a  80 13 32 30  31 35 30 34  31 35 31 31  33 36 35 31
          2d 30 34 30  30 81 01 00  a3 3c 80 01  00 a1 0e 80  08 62 02 72
          02 00 14 32  f0 81 02 64  65 a3 0d 80  07 91 94 71  02 00 35 31
          81 02 64 65  86 0e 4d 54  4f 52 5f 72  75 6c 65 5f  6e 61 6d 65
          87 08 73 72  69 73 6d 72  65 71 a8 11  80 0b 12 08  00 12 04 94
          71 02 94 09  00 81 02 64  65 a9 11 80  0b 12 06 00  12 04 94 71
          22 22 12 11  81 02 64 65  ac 0d 80 07  91 94 71 02  94 09 00 81
          02 64 65 ad  0d 80 07 91  94 71 22 22  12 11 81 02  64 65

7f 66 81 9a      trustedSriSmWithCountryAndNetworkInfo = {
80 13 32 30 31 35 30 34 31 35 31 31 33 36 35 31 2d 30 34 30 30
                   timestamp = '20150415113651-0400'
81 01 00           routingAction = pass(0)
a3 3c              responseInfo = {
80 01 00             queryResult = success(0)
a1 0e                mapImsi = {
80 08 62 02 72 02 00 14 32 f0
                       imsi = 262-02-7200041230
81 02 64 65            country = 'de'
                     }
a3 0d                mapMsc = {
80 07 91 94 71 02 00 35 31
                       gsmAddress = 'international/telephone 491720005313'
81 02 64 65            country = 'de'
                     }
86 0e 4d 54 4f 52 5f 72 75 6c 65 5f 6e 61 6d 65
                     mtRoutingRule = 'MTOR_rule_name'
87 08 73 72 69 73 6d 72 65 71
                     sriqRoutingRule = 'srismreq'
                   }
a8 11              sccpCgPa = {
80 0b 12 08 00 12 04 94 71 02 94 09 00
                     sccpAddress = {
                       reservedForNationalUse = 0
                       routingIndicator = gt
                       globalTitleIndicator = 4
```

```
                      subsystemNumberIndicator = present
                      signallingPointCodeIndicator = absent
                      subsystemNumber = 8
                      translationType = 0
                      numberingPlan = isdnTelephony
                      encodingScheme = bcdEvenNumberOfDigits
                      natureOfAddressIndicator = international
                      addressInformation = 491720499000
                    }
81 02 64 65           country = 'de'
                  }
a9 11               sccpCdPa = {
80 0b 12 06 00 12 04 94 71 22 22 12 11
                    sccpAddress = {
                      reservedForNationalUse = 0
                      routingIndicator = gt
                      globalTitleIndicator = 4
                      subsystemNumberIndicator = present
                      signallingPointCodeIndicator = absent
                      subsystemNumber = 6
                      translationType = 0
                      numberingPlan = isdnTelephony
                      encodingScheme = bcdEvenNumberOfDigits
                      natureOfAddressIndicator = international
                      addressInformation = 491722222111
                    }
81 02 64 65           country = 'de'
                  }
ac 0d               mapSmsc = {
80 07 91 94 71 02 94 09 00
                    gsmAddress = 'international/telephone 491720499000'
81 02 64 65           country = 'de'
                  }
ad 0d               mapMsisdn = {
80 07 91 94 71 22 22 12 11
                    gsmAddress = 'international/telephone 491722222111'
81 02 64 65           country = 'de'
                  }
                }
```

A.4 Suspect SriSm with Country and Network Information

The following sample log file contains one `suspectSriSmWithCountryAndNetworkInfo` record.

```
File: log_islay-vm1_990_20150409_113046_031.dat, record: 1

          7f 67 81 c6  80 13 32 30  31 35 30 34  30 39 31 31  33 30 34 36
          2b 30 32 30  30 81 01 00  a3 27 80 01  00 a5 13 80  08 02 04 08
          00 00 00 55  f8 81 02 6e  6c 82 03 6b  70 6e 86 05  6d 74 72 2d
          31 87 06 73  72 69 71 2d  31 84 05 00  00 00 00 00  a8 16 80 0b
          12 08 00 11  04 13 56 13  13 13 05 81  02 6e 6c 82  03 6b 70 6e
          a9 16 80 0b  12 06 00 11  04 13 56 03  10 00 01 81  02 6e 6c 82
          03 6b 70 6e  ac 12 80 07  91 13 56 13  13 13 f5 81  02 6e 6c 82
          03 6b 70 6e  ad 12 80 07  91 13 56 03  10 00 f1 81  02 6e 6c 82
          03 6b 70 6e  bf 22 23 30  21 80 06 6d  74 6f 78 2d  31 81 08 65
          63 2d 61 70  70 2d 31 82  04 0a 00 00  23 83 01 01  88 01 00 89
          01 01

7f 67 81 c6      suspectSriSmWithCountryAndNetworkInfo = {
80 13 32 30 31 35 30 34 30 39 31 31 33 30 34 36 2b 30 32 30 30
                   timestamp = '20150409113046+0200'
```

```
81 01 00           routingAction = pass(0)
a3 27              responseInfo = {
80 01 00             queryResult = success(0)
a5 13                scrambledImsi = {
80 08 02 04 08 00 00 00 55 f8
                       imsi = 204-08-0000000558
81 02 6e 6c            country = 'nl'
82 03 6b 70 6e         network = 'kpn'
                     }
86 05 6d 74 72 2d 31
                     mtRoutingRule = 'mtr-1'
87 06 73 72 69 71 2d 31
                     sriqRoutingRule = 'sriq-1'
                   }
84 05 00 00 00 00 00
                   ignoredRejectCauses =  {
                     unknownSccpSmscAddress(9) = false(0)
                     unknownMapSmscAddress(10) = false(0)
                     conflictingSmscAddresses(11) = false(0)
                     spoofingSccpSmscAddress(12) = false(0)
                     spoofingMapSmscAddress(13) = false(0)
                   }
a8 16              sccpCgPa = {
80 0b 12 08 00 11 04 13 56 13 13 13 05
                     sccpAddress = {
                       reservedForNationalUse = 0
                       routingIndicator = gt
                       globalTitleIndicator = 4
                       subsystemNumberIndicator = present
                       signallingPointCodeIndicator = absent
                       subsystemNumber = 8
                       translationType = 0
                       numberingPlan = isdnTelephony
                       encodingScheme = bcdOddNumberOfDigits
                       natureOfAddressIndicator = international
                       addressInformation = 31653131315
                     }
81 02 6e 6c          country = 'nl'
82 03 6b 70 6e       network = 'kpn'
                   }
a9 16              sccpCdPa = {
80 0b 12 06 00 11 04 13 56 03 10 00 01
                     sccpAddress = {
                       reservedForNationalUse = 0
                       routingIndicator = gt
                       globalTitleIndicator = 4
                       subsystemNumberIndicator = present
                       signallingPointCodeIndicator = absent
                       subsystemNumber = 6
                       translationType = 0
                       numberingPlan = isdnTelephony
                       encodingScheme = bcdOddNumberOfDigits
                       natureOfAddressIndicator = international
                       addressInformation = 31653001001
                     }
81 02 6e 6c          country = 'nl'
82 03 6b 70 6e       network = 'kpn'
                   }
ac 12              mapSmsc = {
80 07 91 13 56 13 13 13 f5
                     gsmAddress = 'international/telephone 31653131315'
81 02 6e 6c          country = 'nl'
82 03 6b 70 6e       network = 'kpn'
                   }
ad 12              mapMsisdn = {
```

```
80 07 91 13 56 03 10 00 f1
                     gsmAddress = 'international/telephone 31653001001'
81 02 6e 6c          country = 'nl'
82 03 6b 70 6e       network = 'kpn'
                   }
bf 22 23           ecResponseData = {
30 21                ecResponse = {
80 06 6d 74 6f 78 2d 31
                       extConditionRule = 'mtox-1'
81 08 65 63 2d 61 70 70 2d 31
                       applicationName = 'ec-app-1'
82 04 0a 00 00 23      clientIpAddress = 10.0.0.35
83 01 01               evaluationResult = true(1)
88 01 00               ldapError = 0
89 01 01               recepientDomain = 1
                     }
                   }
                 }
```

A.5 Trusted MtFwdSm with Country and Network Information

The following sample log file contains one `trustedMtFwdSmWithCountryAndNetworkInfo` record.

```
File: logging_Srism_Mtor_djerba_200_20150415_113612_110.dat, record: 1

          7f 68 82 01  00 80 13 32  30 31 35 30  34 31 35 31  31 33 36 35
          31 2d 30 34  30 30 81 01  00 a3 19 80  01 00 81 0e  4d 54 4f 52
          5f 72 75 6c  65 5f 6e 61  6d 65 82 04  4d 54 49 52  a8 11 80 0b
          12 08 00 12  04 94 71 02  94 09 00 81  02 64 65 a9  11 80 0b 12
          08 00 12 04  94 59 23 56  44 87 81 02  64 65 ac 0d  80 07 91 94
          71 02 94 09  00 81 02 64  65 ad 0e 80  08 62 02 72  02 00 14 48
          f2 81 02 64  65 af 34 80  02 00 00 a1  0d 80 07 91  94 71 42 25
          22 23 81 02  64 65 82 01  00 83 01 00  84 13 32 30  31 35 30 34
          31 35 31 32  33 35 31 33  2b 30 31 30  30 86 04 53  50 41 4d be
          50 a0 11 80  0b 12 08 00  12 04 94 71  02 94 09 00  81 02 64 65
          a1 0d 80 07  91 94 71 02  94 09 00 81  02 64 65 a2  0d 80 07 91
          94 71 22 22  12 11 81 02  64 65 a3 0e  80 08 62 02  72 02 00 14
          32 f0 81 02  64 65 a5 0d  80 07 91 94  71 02 00 35  31 81 02 64
          65

7f 68 82 01 00   trustedMtFwdSmWithCountryAndNetworkInfo = {
80 13 32 30 31 35 30 34 31 35 31 31 33 36 35 31 2d 30 34 30 30
                   timestamp = '20150415113651-0400'
81 01 00           routingAction = pass(0)
a3 19              responseInfo = {
80 01 00             deliveryResult = success(0)
81 0e 4d 54 4f 52 5f 72 75 6c 65 5f 6e 61 6d 65
                     mtRoutingRule = 'MTOR_rule_name'
82 04 4d 54 49 52    mtiRoutingRule = 'MTIR'
                   }
a8 11              sccpCgPa = {
80 0b 12 08 00 12 04 94 71 02 94 09 00
                     sccpAddress = {
                       reservedForNationalUse = 0
                       routingIndicator = gt
                       globalTitleIndicator = 4
                       subsystemNumberIndicator = present
                       signallingPointCodeIndicator = absent
                       subsystemNumber = 8
                       translationType = 0
                       numberingPlan = isdnTelephony
```

```
                        encodingScheme = bcdEvenNumberOfDigits
                        natureOfAddressIndicator = international
                        addressInformation = 491720499000
                      }
81 02 64 65           country = 'de'
                    }
a9 11               sccpCdPa = {
80 0b 12 08 00 12 04 94 59 23 56 44 87
                      sccpAddress = {
                        reservedForNationalUse = 0
                        routingIndicator = gt
                        globalTitleIndicator = 4
                        subsystemNumberIndicator = present
                        signallingPointCodeIndicator = absent
                        subsystemNumber = 8
                        translationType = 0
                        numberingPlan = isdnTelephony
                        encodingScheme = bcdEvenNumberOfDigits
                        natureOfAddressIndicator = international
                        addressInformation = 499532654478
                      }
81 02 64 65           country = 'de'
                    }
ac 0d               mapSmsc = {
80 07 91 94 71 02 94 09 00
                      gsmAddress = 'international/telephone 491720499000'
81 02 64 65           country = 'de'
                    }
ad 0e               mapImsi = {
80 08 62 02 72 02 00 14 48 f2
                      imsi = 262-02-7200041842
81 02 64 65           country = 'de'
                    }
af 34               smsDeliver = {
80 02 00 00           smsServices =  {
                        replyPath(0) = false(0)
                        userDataHeaderIndication(1) = false(0)
                        statusReportIndication(2) = false(0)
                        moreMessagesToSend(5) = false(0)
                      }
a1 0d                 smsOoriginator = {
80 07 91 94 71 42 25 22 23
                        gsmAddress = 'international/telephone 491724522232'
81 02 64 65             country = 'de'
                      }
82 01 00              smsProtocolId = 0
83 01 00              smsDataCodingScheme = 0
84 13 32 30 31 35 30 34 31 35 31 32 33 35 31 33 2b 30 31 30 30
                      smsScTimestamp = '20150415123513+0100'
86 04 53 50 41 4d     smsUserData = SPAM
                    }
be 50               correlatedSriSm = {
a0 11                 sccpCgPa = {
80 0b 12 08 00 12 04 94 71 02 94 09 00
                        sccpAddress = {
                          reservedForNationalUse = 0
                          routingIndicator = gt
                          globalTitleIndicator = 4
                          subsystemNumberIndicator = present
                          signallingPointCodeIndicator = absent
                          subsystemNumber = 8
                          translationType = 0
                          numberingPlan = isdnTelephony
                          encodingScheme = bcdEvenNumberOfDigits
                          natureOfAddressIndicator = international
```

```
                          addressInformation = 491720499000
                        }
81 02 64 65             country = 'de'
                      }
a1 0d                 mapSmsc = {
80 07 91 94 71 02 94 09 00
                        gsmAddress = 'international/telephone 491720499000'
81 02 64 65             country = 'de'
                      }
a2 0d                 mapMsisdn = {
80 07 91 94 71 22 22 12 11
                        gsmAddress = 'international/telephone 491722222111'
81 02 64 65             country = 'de'
                      }
a3 0e                 mapImsi = {
80 08 62 02 72 02 00 14 32 f0
                        imsi = 262-02-7200041230
81 02 64 65             country = 'de'
                      }
a5 0d                 mapMsc = {
80 07 91 94 71 02 00 35 31
                        gsmAddress = 'international/telephone 491720005313'
81 02 64 65             country = 'de'
                      }
                    }
                  }
```

A.6 Suspect MtFwdSm with Country and Network Information

The following sample log file contains one `suspectMtFwdSmWithCountryAndNetworkInfo` record.

```
File: log_islay-vm1_990_20150409_113046_000.dat, record: 1

          7f 69 82 01  4a 80 13 32  30 31 35 30  34 30 39 31  31 33 30 34
          37 2b 30 32  30 30 81 01  00 a3 12 80  01 0a 81 05  6d 74 72 2d
          31 82 06 6d  74 69 72 2d  31 84 05 00  00 00 00 00  a8 16 80 0b
          12 08 00 11  04 13 56 13  13 13 05 81  02 6e 6c 82  03 6b 70 6e
          a9 16 80 0b  12 08 00 11  04 13 56 03  00 01 01 81  02 6e 6c 82
          03 6b 70 6e  ac 12 80 07  91 13 56 13  13 13 f5 81  02 6e 6c 82
          03 6b 70 6e  ad 13 80 08  02 04 08 00  00 30 94 f4  81 02 6e 6c
          82 03 6b 70  6e af 41 80  02 00 00 a1  12 80 07 91  13 56 03 10
          32 f4 81 02  6e 6c 82 03  6b 70 6e 82  01 00 83 01  00 84 13 32
          30 30 34 31  32 33 31 32  33 35 39 34  38 2d 30 34  31 35 86 0c
          54 65 73 74  20 4d 65 73  73 61 67 65  be 40 a0 16  80 0b 12 08
          00 11 04 13  56 13 13 13  05 81 02 6e  6c 82 03 6b  70 6e a1 12
          80 07 91 13  56 13 13 13  f5 81 02 6e  6c 82 03 6b  70 6e a2 12
          80 07 91 13  56 03 10 00  f6 81 02 6e  6c 82 03 6b  70 6e bf 3d
          36 86 01 00  a8 0a 80 01  0a 81 05 6d  74 72 2d 31  bf 3c 0a 80
          04 00 0a 65  01 81 02 7d  06 9f 3d 10  31 39 32 2e  31 36 38 2e
          31 2e 31 3a  39 30 30 31  9f 3e 04 33  67 67 70

7f 69 82 01 4a   suspectMtFwdSmWithCountryAndNetworkInfo = {
80 13 32 30 31 35 30 34 30 39 31 31 33 30 34 37 2b 30 32 30 30
                    timestamp = '20150409113047+0200'
81 01 00            routingAction = pass(0)
a3 12               responseInfo = {
80 01 0a              deliveryResult = systemFailureError(10)
81 05 6d 74 72 2d 31
                      mtRoutingRule = 'mtr-1'
82 06 6d 74 69 72 2d 31
                      mtiRoutingRule = 'mtir-1'
```

```
                    }
84 05 00 00 00 00 00
                    ignoredRejectCauses =  {
                      unknownSccpSmscAddress(9) = false(0)
                      unknownMapSmscAddress(10) = false(0)
                      conflictingSmscAddresses(11) = false(0)
                      spoofingSccpSmscAddress(12) = false(0)
                      spoofingMapSmscAddress(13) = false(0)
                    }
a8 16               sccpCgPa = {
80 0b 12 08 00 11 04 13 56 13 13 13 05
                      sccpAddress = {
                        reservedForNationalUse = 0
                        routingIndicator = gt
                        globalTitleIndicator = 4
                        subsystemNumberIndicator = present
                        signallingPointCodeIndicator = absent
                        subsystemNumber = 8
                        translationType = 0
                        numberingPlan = isdnTelephony
                        encodingScheme = bcdOddNumberOfDigits
                        natureOfAddressIndicator = international
                        addressInformation = 31653131315
                      }
81 02 6e 6c           country = 'nl'
82 03 6b 70 6e        network = 'kpn'
                    }
a9 16               sccpCdPa = {
80 0b 12 08 00 11 04 13 56 03 00 01 01
                      sccpAddress = {
                        reservedForNationalUse = 0
                        routingIndicator = gt
                        globalTitleIndicator = 4
                        subsystemNumberIndicator = present
                        signallingPointCodeIndicator = absent
                        subsystemNumber = 8
                        translationType = 0
                        numberingPlan = isdnTelephony
                        encodingScheme = bcdOddNumberOfDigits
                        natureOfAddressIndicator = international
                        addressInformation = 31653000101
                      }
81 02 6e 6c           country = 'nl'
82 03 6b 70 6e        network = 'kpn'
                    }
ac 12               mapSmsc = {
80 07 91 13 56 13 13 13 f5
                      gsmAddress = 'international/telephone 31653131315'
81 02 6e 6c           country = 'nl'
82 03 6b 70 6e        network = 'kpn'
                    }
ad 13               mapImsi = {
80 08 02 04 08 00 00 30 94 f4
                      imsi = 204-08-0000003494
81 02 6e 6c           country = 'nl'
82 03 6b 70 6e        network = 'kpn'
                    }
af 41               smsDeliver = {
80 02 00 00           smsServices =  {
                        replyPath(0) = false(0)
                        userDataHeaderIndication(1) = false(0)
                        statusReportIndication(2) = false(0)
                        moreMessagesToSend(5) = false(0)
                      }
a1 12                 smsOriginator = {
```

```
80 07 91 13 56 03 10 32 f4
                        gsmAddress = 'international/telephone 31653001234'
81 02 6e 6c             country = 'nl'
82 03 6b 70 6e          network = 'kpn'
                      }
82 01 00              smsProtocolId = 0
83 01 00              smsDataCodingScheme = 0
84 13 32 30 30 34 31 32 33 31 32 33 35 39 34 38 2d 30 34 31 35
                      smsScTimestamp = '20041231235948-0415'
86 0c 54 65 73 74 20 4d 65 73 73 61 67 65
                      smsUserData = Test Message
                    }
be 40               correlatedSriSm = {
a0 16                 sccpCgPa = {
80 0b 12 08 00 11 04 13 56 13 13 13 05
                        sccpAddress = {
                          reservedForNationalUse = 0
                          routingIndicator = gt
                          globalTitleIndicator = 4
                          subsystemNumberIndicator = present
                          signallingPointCodeIndicator = absent
                          subsystemNumber = 8
                          translationType = 0
                          numberingPlan = isdnTelephony
                          encodingScheme = bcdOddNumberOfDigits
                          natureOfAddressIndicator = international
                          addressInformation = 31653131315
                        }
81 02 6e 6c             country = 'nl'
82 03 6b 70 6e          network = 'kpn'
                      }
a1 12                 mapSmsc = {
80 07 91 13 56 13 13 13 f5
                        gsmAddress = 'international/telephone 31653131315'
81 02 6e 6c             country = 'nl'
82 03 6b 70 6e          network = 'kpn'
                      }
a2 12                 mapMsisdn = {
80 07 91 13 56 03 10 00 f6
                        gsmAddress = 'international/telephone 31653001006'
81 02 6e 6c             country = 'nl'
82 03 6b 70 6e          network = 'kpn'
                      }
                    }
bf 3d 36            outboundSip = {
86 01 00              mtFwdSmToImsRoutingAction = pass(0)
a8 0a                 mtFwdSmToImsResponseInfo = {
80 01 0a                deliveryResult = systemFailureError(10)
81 05 6d 74 72 2d 31    mtRoutingRule = 'mtr-1'
                      }
bf 3c 0a              iiw = {
80 04 00 0a 65 01       hostId = 681217
81 02 7d 06             portNumber = 32006
                      }
9f 3d 10 31 39 32 2e 31 36 38 2e 31 2e 31 3a 39 30 30 31
                      sipServerAddress = '192.168.1.1:9001'
9f 3e 04 33 67 67 70
                      sipClientName = '3ggp'
                    }
                  }
```

A.7 Received SubmitSm with Country and Network Information

The following sample log file contains one `receivedSubmitSmWithCountryAndNetworkInfo` record.

```
File: /log1.asd.mbalance.com_20070411_131258_016.dat, record: 1

      7f 74 82 05  22 80 13 32  30 30 37 30  34 31 31 31  33 31 33 30
      33 2b 30 32  30 30 81 01  05 a2 08 80  01 14 81 03  61 62 63 91
      04 61 70 70  31 92 03 e0  01 00 b3 82  04 f1 a2 0d  80 07 91 23
      56 16 32 54  f6 81 02 62  65 a4 12 80  07 91 13 56  03 00 00 f0
      81 02 6e 6c  82 03 6b 70  6e 85 02 00  f0 86 01 05  87 02 00 20
      8a 81 a0 61  62 63 64 65  66 67 68 69  6a 61 62 63  64 65 66 67
      68 69 6a 61  62 63 64 65  66 67 68 69  6a 61 62 63  64 65 66 67
      68 69 6a 61  62 63 64 65  66 67 68 69  6a 61 62 63  64 65 66 67
      68 69 6a 61  62 63 64 65  66 67 68 69  6a 61 62 63  64 65 66 67
      68 69 6a 61  62 63 64 65  66 67 68 69  6a 61 62 63  64 65 66 67
      68 69 6a 61  62 63 64 65  66 67 68 69  6a 61 62 63  64 65 66 67
      68 69 6a 61  62 63 64 65  66 67 68 69  6a 61 62 63  64 65 66 67
      68 69 6a 61  62 63 64 65  66 67 68 69  6a 61 62 63  64 65 66 67
      68 69 6a 8c  01 01 8d 01  09 8e 01 01  92 13 32 30  30 37 30 34
      31 32 31 33  31 31 31 31  2b 30 32 30  30 93 01 00  94 82 03 f6
      61 62 63 64  65 66 67 68  69 6a 6b 6c  6d 6e 6f 70  71 72 73 74
      75 76 77 78  79 7a 61 62  63 64 65 66  67 68 69 6a  6b 6c 6d 6e
      6f 70 71 72  73 74 75 76  77 78 79 7a  61 62 63 64  65 66 67 68
      69 6a 6b 6c  6d 6e 6f 70  71 72 73 74  75 76 77 78  79 7a 61 62
      63 64 65 66  67 68 69 6a  6b 6c 6d 6e  6f 70 71 72  73 74 75 76
      77 78 79 7a  61 62 63 64  65 66 67 68  69 6a 6b 6c  6d 6e 6f 70
      71 72 73 74  75 76 77 78  79 7a 61 62  63 64 65 66  67 68 69 6a
      6b 6c 6d 6e  6f 70 71 72  73 74 75 76  77 78 79 7a  61 62 63 64
      65 66 67 68  69 6a 6b 6c  6d 6e 6f 70  71 72 73 74  75 76 77 78
      79 7a 61 62  63 64 65 66  67 68 69 6a  6b 6c 6d 6e  6f 70 71 72
      73 74 75 76  77 78 79 7a  61 62 63 64  65 66 67 68  69 6a 6b 6c
      6d 6e 6f 70  71 72 73 74  75 76 77 78  79 7a 61 62  63 64 65 66
      67 68 69 6a  6b 6c 6d 6e  6f 70 71 72  73 74 75 76  77 78 79 7a
      61 62 63 64  65 66 67 68  69 6a 6b 6c  6d 6e 6f 70  71 72 73 74
      75 76 77 78  79 7a 61 62  63 64 65 66  67 68 69 6a  6b 6c 6d 6e
      6f 70 71 72  73 74 75 76  77 78 79 7a  61 62 63 64  65 66 67 68
      69 6a 6b 6c  6d 6e 6f 70  71 72 73 74  75 76 77 78  79 7a 61 62
      63 64 65 66  67 68 69 6a  6b 6c 6d 6e  6f 70 71 72  73 74 75 76
      77 78 79 7a  61 62 63 64  65 66 67 68  69 6a 6b 6c  6d 6e 6f 70
      71 72 73 74  75 76 77 78  79 7a 61 62  63 64 65 66  67 68 69 6a
      6b 6c 6d 6e  6f 70 71 72  73 74 75 76  77 78 79 7a  61 62 63 64
      65 66 67 68  69 6a 6b 6c  6d 6e 6f 70  71 72 73 74  75 76 77 78
      79 7a 61 62  63 64 65 66  67 68 69 6a  6b 6c 6d 6e  6f 70 71 72
      73 74 75 76  77 78 79 7a  61 62 63 64  65 66 67 68  69 6a 6b 6c
      6d 6e 6f 70  71 72 73 74  75 76 77 78  79 7a 61 62  63 64 65 66
      67 68 69 6a  6b 6c 6d 6e  6f 70 71 72  73 74 75 76  77 78 79 7a
      61 62 63 64  65 66 67 68  69 6a 6b 6c  6d 6e 6f 70  71 72 73 74
      75 76 77 78  79 7a 61 62  63 64 65 66  67 68 69 6a  6b 6c 6d 6e
      6f 70 71 72  73 74 75 76  77 78 79 7a  61 62 63 64  65 66 67 68
      69 6a 6b 6c  6d 6e 6f 70  71 72 73 74  75 76 77 78  79 7a 61 62
      63 64 65 66  67 68 69 6a  6b 6c 6d 6e  6f 70 71 72  73 74 75 76
      77 78 79 7a  61 62 63 64  65 66 67 68  69 6a 6b 6c  6d 6e 6f 70
      71 72 73 74  75 76 77 78  79 7a 61 62  63 64 65 66  67 68 69 6a
      6b 6c 6d 6e  6f 70 71 72  73 74 75 76  77 78 79 7a  61 62 63 64
      65 66 67 68  69 6a 6b 6c  6d 6e 6f 70  71 72 73 74  75 76 77 78
      79 7a 61 62  63 64 65 66  67 68 69 6a  6b 6c 6d 6e  6f 70 71 72
      73 74 75 76  77 78 79 7a  61 62 63 64  65 66 67 68  69 6a 6b 6c
      6d 6e 6f 70  71 72 73 74  75 76 77 78  79 7a 61 62  63 64 65 66
      67 68 69 6a  6b 6c 6d 6e  6f 70 71 72  73 74 75 76  77 78 79 7a
```

```
      61 62 63 64  65 66 67 68  69 6a 6b 6c  6d 6e 6f 70  71 72 73 74
      75 76 77 78  79 7a 61 62  63 64 65 66  67 68 69 6a  6b 6c 6d 6e
      6f 70 71 72  73 74 75 76  77 78 79 7a  61 62 63 64  65 66 67 68
      69 6a 6b 6c  6d 6e 6f 70  71 72 73 74  75 76 77 78  79 7a 61 62
      63 64 65 66  67 68 69 6a  6b 6c 6d 6e  6f 70 71 72  73 74 75 76
      77 78 79 7a  61 62 63 64  65 66 67 68  69 6a 6b 6c  6d 6e 6f 70
      71 72 73 74  75 76 77 78  79 7a 61 62  63 64 65 66  67 68 69 6a
      6b 6c 6d 6e  6f 70 71 72  73 74 75 76  77 78 79 7a  61 62 63 64
      65 66 67 68  69 6a 6b 6c  6d 6e 6f 70  71 72 73 74  75 76 77 78
      79 7a 61 62  63 64 65 66  67 68 69 6a  6b 6c 6d 6e  6f 70 71 72
      73 74 75 76  77 78 79 7a  61 62 63 64  65 66 67 68  69 6a 6b 6c
      6d 6e 6f 70  71 72 73 74  75 76 77 78  79 7a 9f 22  02 23 29

7f 74 82 05 22   receivedSubmitSmWithCountryAndNetworkInfo = {
80 13 32 30 30 37 30 34 31 31 31 33 31 33 30 33 2b 30 32 30 30
                    timestamp = '20070411131303+0200'
81 01 05            routingAction = discardWithAck(5)
a2 08               rejectInfo = {
80 01 14              rejectCause = matchingAoRoutingRule(20)
81 03 61 62 63        aoRoutingRule = 'abc'
                    }
91 04 61 70 70 31
                    applicationName = 'app1'
92 03 e0 01 00      applicationShortNumber = 'abbreviated/unknown 1000'
b3 82 04 f1         messageFields = {
a2 0d                 originatorAddress = {
80 07 91 23 56 16 32 54 f6
                        gsmAddress = 'international/telephone 32656123456'
81 02 62 65             country = 'be'
                      }
a4 12                 recipientAddress = {
80 07 91 13 56 03 00 00 f0
                        gsmAddress = 'international/telephone 31653000000'
81 02 6e 6c             country = 'nl'
82 03 6b 70 6e          network = 'kpn'
                      }
85 02 00 f0           dataCodingScheme = 240
86 01 05              protocolIdentifier = 5
87 02 00 20           notificationType =  {
                        deliveryNotification(0) = false(0)
                        nonDeliveryNotification(1) = false(0)
                        bufferedNotification(2) = false(0)
                      }
8a 81 a0 61 62 63 64 65 66 67 68 69 6a 61 62 63 64 65 66 67 68 69 6a
61 62 63
64 65 66 67 68 69 6a 61 62 63 64 65 66 67 68 69 6a 61 62 63 64 65 66
67 68 69
6a 61 62 63 64 65 66 67 68 69 6a 61 62 63 64 65 66 67 68 69 6a 61 62
63 64 65
66 67 68 69 6a 61 62 63 64 65 66 67 68 69 6a 61 62 63 64 65 66 67 68
69 6a 61
62 63 64 65 66 67 68 69 6a 61 62 63 64 65 66 67 68 69 6a 61 62 63 64
65 66 67
68 69 6a 61 62 63 64 65 66 67 68 69 6a 61 62 63 64 65 66 67 68 69 6a
61 62 63
64 65 66 67 68 69 6a
                      userData =
abcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghi
jabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefghijabcdefgh
ijabcdefghijabcdefghij
8c 01 01              moreMessagesToSend = true(1)
8d 01 09              priority = 9
8e 01 01              replyPathIndicator = true(1)
92 13 32 30 30 37 30 34 31 32 31 33 31 31 31 31 2b 30 32 30 30
                      validityPeriod = '20070412131111+0200'
```

```
93 01 00             singleShotIndicator = false(0)
94 82 03 f6 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
```

```
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73
74 75 76
77 78 79 7a          billingIdentifier =
'abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnop
qrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefg
hijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwx
yzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmno
pqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdef
ghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvw
xyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmn
opqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcde
fghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuv
wxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklm
nopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcd
efghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstu
vwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijkl
mnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabc
defghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz'
9f 22 02 23 29       portNumber = 9001
                   }
                 }
```

A.8 Received DeliverSm with Country and Network Information

The following sample log file contains one `receivedDeliverSmWithCountryAndNetworkInfo` record.

```
File: /log1.asd.mbalance.com_20070411_131410_064.dat, record: 1

          7f 75 82 01  52 80 13 32  30 30 37 30  34 31 31 31  33 31
34 31
          32 2b 30 32  30 30 81 01  03 a2 08 80  01 16 81 03  61 62
63 91
          04 61 70 70  31 92 03 e0  01 00 b3 82  01 21 a2 12  80 07
91 13
          56 03 00 00  f0 81 02 6e  6c 82 03 6b  70 6e a4 16  80 07
91 13
          56 05 00 00  f0 81 02 6e  6c 82 07 74  65 6c 66 6f  72 74
85 02
          00 f0 86 01  05 8a 81 a0  00 01 02 03  04 05 06 07  08 09
0a 0b
          0c 0d 0e 0f  10 11 12 13  14 15 16 17  18 19 1a 1b  1c 1d
1e 1f
          20 21 22 23  24 25 26 27  28 29 2a 2b  2c 2d 2e 2f  30 31
32 33
          34 35 36 37  38 39 3a 3b  3c 3d 3e 3f  40 41 42 43  44 45
46 47
          48 49 4a 4b  4c 4d 4e 4f  50 51 52 53  54 55 56 57  58 59
5a 5b
```

```
          5c 5d 5e 5f  60 61 62 63  64 65 66 67  68 69 6a 6b  6c 6d
6e 6f
          70 71 72 73  74 75 76 77  78 79 7a 7b  7c 7d 7e 7f  80 81
82 83
          84 85 86 87  88 89 8a 8b  8c 8d 8e 8f  90 91 92 93  94 95
96 97
          98 99 9a 9b  9c 9d 9e 9f  8c 01 01 8e  01 00 95 13  32 30
30 37
          30 34 31 30  31 33 31 31  31 31 2b 30  32 30 30 bb  11 80
06 02
          04 18 32 54  f6 81 02 6e  6c 82 03 6b  70 6e bc 16  80 07
91 13
          56 95 99 99  f9 81 02 6e  6c 82 07 74  65 6c 66 6f  72 74
9f 22
          02 23 29

7f 75 82 01 52   receivedDeliverSmWithCountryAndNetworkInfo = {
80 13 32 30 30 37 30 34 31 31 31 33 31 34 31 32 2b 30 32 30 30
                    timestamp = '20070411131412+0200'
81 01 03            routingAction = blockWithAck(3)
a2 08               rejectInfo = {
80 01 16              rejectCause = matchingAtRoutingRule(22)
81 03 61 62 63        atRoutingRule = 'abc'
                    }
91 04 61 70 70 31
                    applicationName = 'app1'
92 03 e0 01 00      applicationShortNumber = 'abbreviated/unknown
1000'
b3 82 01 21         messageFields = {
a2 12                 originatorAddress = {
80 07 91 13 56 03 00 00 f0
                        gsmAddress = 'international/telephone
31653000000'
81 02 6e 6c             country = 'nl'
82 03 6b 70 6e          network = 'kpn'
                      }
a4 16                 recipientAddress = {
80 07 91 13 56 05 00 00 f0
                        gsmAddress = 'international/telephone
31655000000'
81 02 6e 6c             country = 'nl'
82 07 74 65 6c 66 6f 72 74
                        network = 'telfort'
                      }
85 02 00 f0           dataCodingScheme = 240
86 01 05              protocolIdentifier = 5
8a 81 a0 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13
14 15 16
17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d
2e 2f 30
31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45 46 47
48 49 4a
4b 4c 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f 60 61
62 63 64
65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 78 79 7a 7b
7c 7d 7e
7f 80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f 90 91 92 93 94 95
96 97 98
99 9a 9b 9c 9d 9e 9f
                    userData = ..............................
!"#$%&'()*+,-
./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr
stuvwxyz{|}~.€ ‚ƒ„…†‡^‰Š Œ Ž  ‘’“”•-—˜™š œ žŸ
8c 01 01            moreMessagesToSend = true(1)
8e 01 00            replyPathIndicator = false(0)
```

```
95 13 32 30 30 37 30 34 31 30 31 33 31 31 31 31 2b 30 32 30 30
                    serviceCentreTimestamp = '20070410131111+0200'
bb 11               originatedImsi = {
80 06 02 04 18 32 54 f6
                      imsi = 204-08-123456
81 02 6e 6c           country = 'nl'
82 03 6b 70 6e        network = 'kpn'
                    }
bc 16               originatedMscAddress = {
80 07 91 13 56 95 99 99 f9
                      gsmAddress = 'international/telephone
31655999999'
81 02 6e 6c           country = 'nl'
82 07 74 65 6c 66 6f 72 74
                      network = 'telfort'
                    }
9f 22 02 23 29      portNumber = 9001
                  }
                }
```

A.9 Received Notification with Country and Network Information

The following sample log file contains one `receivedNotificationWithCountryAndNetworkInfo` record.

```
File: /log1.asd.mbalance.com_20070411_131437_082.dat, record: 1

          7f 76 81 80  80 13 32 30  30 37 30 34  31 31 31 33  31 34
33 38
          2b 30 32 30  30 81 01 03  a2 08 80 01  16 81 03 61  62 63
91 04
          61 70 70 31  92 03 e0 01  00 b3 51 a2  05 80 03 e0  02 00
a4 0d
          80 07 91 13  26 07 39 30  f8 81 02 6e  6c 8c 01 00  95 13
32 30
          30 37 30 34  31 30 31 33  31 31 31 31  2b 30 32 30  30 96
01 03
          97 04 67 11  11 21 98 13  32 30 30 37  30 34 31 31  31 32
31 31
          31 31 2b 30  32 30 30 9f  22 02 23 29

7f 76 81 80      receivedNotificationWithCountryAndNetworkInfo = {
80 13 32 30 30 37 30 34 31 31 31 33 31 34 33 38 2b 30 32 30 30
                    timestamp = '20070411131438+0200'
81 01 03            routingAction = blockWithAck(3)
a2 08               rejectInfo = {
80 01 16              rejectCause = matchingAtRoutingRule(22)
81 03 61 62 63        atRoutingRule = 'abc'
                    }
91 04 61 70 70 31
                    applicationName = 'app1'
92 03 e0 01 00      applicationShortNumber = 'abbreviated/unknown 1000'
b3 51               messageFields = {
a2 05                 originatorAddress = {
80 03 e0 02 00          gsmAddress = 'abbreviated/unknown 2000'
                      }
a4 0d                 recipientAddress = {
80 07 91 13 26 07 39 30 f8
                        gsmAddress = 'international/telephone 31627093038'
81 02 6e 6c             country = 'nl'
                      }
```

```
8c 01 00              moreMessagesToSend = false(0)
95 13 32 30 30 37 30 34 31 30 31 33 31 31 31 31 2b 30 32 30 30
                      serviceCentreTimestamp = '20070410131111+0200'
96 01 03              deliveryStatus = deliveryFailed(3)
97 04 67 11 11 21     errorCode = 1729171745
98 13 32 30 30 37 30 34 31 31 31 32 31 31 31 31 2b 30 32 30 30
                      deliveryTimestamp = '20070411121111+0200'
9f 22 02 23 29        portNumber = 9001
                    }
                  }
```

A.10 AMS Delivery Attempt for MoAt with Country and Network Information

The following sample log file contains one `amsDeliveryAttemptForMoAtWithCountryAndNetworkInfo` record.

```
File: /log1.asd.mbalance.com_20070411_131842_000.dat, record: 1

          7f 81 06 81  ab 80 13 32  30 30 37 30  34 31 31 31  33 31
38 34
          32 2b 30 32  30 30 b3 58  a2 09 80 07  91 13 06 88  88 88
f8 85
          02 00 f0 86  01 05 8a 0a  61 62 63 64  65 66 67 68  69 6a
8b 05
          00 03 02 ff  de 8e 01 00  bb 0a 80 08  02 14 12 32  54 76
98 f0
          bd 09 80 07  91 13 06 00  00 00 f0 9f  1f 0b 66 77  64 66
65 72
          72 65 72 65  74 9f 21 03  00 ff ff 9f  28 02 00 80  b7 18
80 01
          00 a2 08 80  01 00 82 03  61 62 63 83  04 61 70 70  31 84
03 e0
          01 00 99 01  00 9a 01 03  9b 13 32 30  30 37 30 34  31 31
31 32
          31 38 34 32  2b 30 32 30  30 9c 05 31  32 33 34 35
7f 81 06 81 ab   amsDeliveryAttemptForMoAtWithCountryAndNetworkInfo =
 {
80 13 32 30 30 37 30 34 31 31 31 33 31 38 34 32 2b 30 32 30 30
                    timestamp = '20070411131842+0200'
b3 58               messageFields = {
a2 09                 originatorAddress = {
80 07 91 13 06 88 88 88 f8
                        gsmAddress = 'international/telephone 31608888888'
                      }
85 02 00 f0           dataCodingScheme = 240
86 01 05              protocolIdentifier = 5
8a 0a 61 62 63 64 65 66 67 68 69 6a
                      userData = abcdefghij
8b 05 00 03 02 ff de
                      userDataHeader = 000302ffdeh
8e 01 00              replyPathIndicator = false(0)
bb 0a                 originatedImsi = {
80 08 02 14 12 32 54 76 98 f0
                        imsi = 204-12-1234567890
                      }
bd 09                 serviceCentreAddress = {
80 07 91 13 06 00 00 00 f0
                        gsmAddress = 'international/telephone 31600000000'
                      }
```

```
9f 1f 0b 66 77 64 66 65 72 72 65 72 65 74
                     alphanumericRecipient = 'fwdferreret'
9f 21 03 00 ff ff    originatingPointCode = 65535
9f 28 02 00 80       gsmMessageReference = 128
                   }
b7 18              outboundAt = {
80 01 00             routingAction = pass(0)
a2 08                responseInfo = {
80 01 00               deliveryResult = successful(0)
82 03 61 62 63         atRoutingRule = 'abc'
                     }
83 04 61 70 70 31    applicationName = 'app1'
84 03 e0 01 00       applicationShortNumber = 'abbreviated/unknown 1000'
                   }
99 01 00           moreMessagesToSend = false(0)
9a 01 03           numberOfPreviousAttempts = 3
9b 13 32 30 30 37 30 34 31 31 31 32 31 38 34 32 2b 30 32 30 30
                    serviceCentreTimestamp = '20070411121842+0200'
9c 05 31 32 33 34 35
                    messageIdentifier = 49-842216501
                 }
```

Appendix B

Log Record ASN.1 Data Types

Topics:

B.1 Log Record ASN.1 Data Types

The following sample intercept file contains a submit and delivery intercept record for single SIPO-MT flow:

```
-- ------------------------------------------------------------------------
--
--   (c) Copyright 2004-2015 NewNet
--
--   This software is proprietary to and embodies the confidential technology
--   of NewNet. Possession, use, duplication or dissemination of the
--   software and media is authorized only pursuant to a valid written license
--   from NewNet.
--
-- ------------------------------------------------------------------------
-- $Id: logging.asn1,v 1.79 2015/07/09 11:29:13 opaliwal Exp $
-- ------------------------------------------------------------------------

LOGGING DEFINITIONS IMPLICIT TAGS ::=
BEGIN

--- Following construct defines all possible log records.

InboundMessage ::=  --snacc isPdu:"TRUE"
CHOICE {
    -- Following are log records that will be generated when the log format is
    -- equal to 'asn1Extended'

    trustedMoFwdSm                                  [APPLICATION 0] IMPLICIT
TrustedMoFwdSm,
    suspectMoFwdSm                                  [APPLICATION 1] IMPLICIT
SuspectMoFwdSm,
    trustedSriSm                                    [APPLICATION 2] IMPLICIT
TrustedSriSm,
    suspectSriSm                                    [APPLICATION 3] IMPLICIT
SuspectSriSm,
    trustedMtFwdSm                                  [APPLICATION 4] IMPLICIT
TrustedMtFwdSm,
    suspectMtFwdSm                                  [APPLICATION 5] IMPLICIT
SuspectMtFwdSm,

    -- event                                          [APPLICATION 30] IMPLICIT
 Event,

    -- Following are log records that will be generated when the log format is
    -- equal to 'asn1ExtendedWithCountryAndNetworkInfo'

    trustedMoFwdSmWithCountryAndNetworkInfo         [APPLICATION 100] IMPLICIT
MoFwdSmWithCountryAndNetworkInfo,
    suspectMoFwdSmWithCountryAndNetworkInfo         [APPLICATION 101] IMPLICIT
MoFwdSmWithCountryAndNetworkInfo,

    trustedSriSmWithCountryAndNetworkInfo           [APPLICATION 102] IMPLICIT
TrustedSriSmWithCountryAndNetworkInfo,
    suspectSriSmWithCountryAndNetworkInfo           [APPLICATION 103] IMPLICIT
SuspectSriSmWithCountryAndNetworkInfo,
    trustedMtFwdSmWithCountryAndNetworkInfo         [APPLICATION 104] IMPLICIT
TrustedMtFwdSmWithCountryAndNetworkInfo,
    suspectMtFwdSmWithCountryAndNetworkInfo         [APPLICATION 105] IMPLICIT
SuspectMtFwdSmWithCountryAndNetworkInfo,
   -- trustedCdmaMoFwdSmWithCountryAndNetworkInfo      [APPLICATION 106] IMPLICIT
```

```
 TrustedCdmaMoFwdSmWithCountryAndNetworkInfo,
    receivedSubmitSmWithCountryAndNetworkInfo          [APPLICATION 116] IMPLICIT
ReceivedSubmitSmWithCountryAndNetworkInfo,
    receivedDeliverSmWithCountryAndNetworkInfo         [APPLICATION 117] IMPLICIT
ReceivedDeliverSmWithCountryAndNetworkInfo,
    receivedNotificationWithCountryAndNetworkInfo      [APPLICATION 118] IMPLICIT
ReceivedNotificationWithCountryAndNetworkInfo,

    extCondMessageWithCountryAndNetworkInfo            [APPLICATION 131] IMPLICIT
EventWithCountryAndNetworkInfo,
    amsDeliveryAttemptForAoMtWithCountryAndNetworkInfo [APPLICATION 132] IMPLICIT
EventWithCountryAndNetworkInfo,
    amsDeliveryAttemptForMoMtWithCountryAndNetworkInfo [APPLICATION 133] IMPLICIT
EventWithCountryAndNetworkInfo,
    amsDeliveryAttemptForMoAtWithCountryAndNetworkInfo [APPLICATION 134] IMPLICIT
EventWithCountryAndNetworkInfo,
    amsDeliveryAttemptForAoAtWithCountryAndNetworkInfo [APPLICATION 135] IMPLICIT
EventWithCountryAndNetworkInfo,
   -- Note: The AoAt log record is also used for logging AT-Store-AT messages coming
 out of the AMS.

    amsTerminateWithCountryAndNetworkInfo              [APPLICATION 136] IMPLICIT
EventWithCountryAndNetworkInfo,
    notifEventWithCountryAndNetworkInfo                [APPLICATION 137] IMPLICIT
EventWithCountryAndNetworkInfo,
    copyForwardEventWithCountryAndNetworkInfo          [APPLICATION 138] IMPLICIT
EventWithCountryAndNetworkInfo,
    msCommandWithCountryAndNetworkInfo                 [APPLICATION 139] IMPLICIT
CommandWithCountryAndNetworkInfo,

    -- Note: Registration log record is also used for De-Registration of UE
    imsRegistrationWithCountryAndNetworkInfo           [APPLICATION 140] IMPLICIT
RegistrationWithCountryAndNetworkInfo
}




-- Following record is generated for an inbound MO-ForwardSM operation
-- that is considered to be trusted.

TrustedMoFwdSm ::= SEQUENCE {
    timestamp                  [0] IMPLICIT GeneralizedTime,
    routingAction              [1] IMPLICIT MoFwdSmRoutingAction,
    -- rejectInfo and responseInfo are mutually exclusive.
    -- rejectInfo is included when the action is 'discardWithAck',
    -- 'discardWithNak', or 'discardWithNoResponse'.
    -- responseInfo is included otherwise.
    rejectInfo                 [2] IMPLICIT SEQUENCE {
        rejectCause                [0] IMPLICIT RejectCause,
        moRoutingRule              [1] IMPLICIT NameString OPTIONAL,
        moExtConditionRule         [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    responseInfo               [3] IMPLICIT SEQUENCE {
        submissionResult             [0] IMPLICIT MoFwdSmSubmissionResult,
        moRoutingRule              [1] IMPLICIT NameString
    } OPTIONAL,
    -- sccpCgPaOfFirstSegment and sccpCdPaOfFirstSegment are only included when
    -- the MO-ForwardSM has been received in a segmented TCAP dialogue.
    sccpCgPaOfFirstSegment     [6] IMPLICIT SccpAddress OPTIONAL,
    sccpCdPaOfFirstSegment     [7] IMPLICIT SccpAddress OPTIONAL,
    sccpCgPa                   [8] IMPLICIT SccpAddress,
```

```
    sccpCdPa                    [9] IMPLICIT SccpAddress,
    mapSmsc                    [12] IMPLICIT GsmAddress OPTIONAL,
    mapMsisdn                  [13] IMPLICIT GsmAddress OPTIONAL,
    mapImsi                    [14] IMPLICIT Imsi OPTIONAL,
    -- smsSubmit and smsCommand are mutually exclusive.
    smsSubmit                  [15] IMPLICIT SEQUENCE {
        smsServices                 [0] IMPLICIT SmsSubmitServices,
        smsMessageReference         [1] IMPLICIT MessageReference,
        smsRecipient                [2] IMPLICIT GsmAddress,
        smsProtocolId               [3] IMPLICIT ProtocolId,
        smsDataCodingScheme         [4] IMPLICIT DataCodingScheme,
        smsValidityPeriod           [5] IMPLICIT GeneralizedTime OPTIONAL,
        smsUserDataHeader           [6] IMPLICIT UserDataHeader OPTIONAL,
        smsUserData                 [7] IMPLICIT UserData
    } OPTIONAL,
    smsCommand                 [16] IMPLICIT SEQUENCE {
        smsServices                 [0] IMPLICIT SmsCommandServices,
        smsMessageReference         [1] IMPLICIT MessageReference,
        smsProtocolId               [2] IMPLICIT ProtocolId,
        smsCommandType              [3] IMPLICIT CommandType,
        smsMessageNumber            [4] IMPLICIT MessageNumber,
        smsRecipient                [5] IMPLICIT GsmAddress,
        smsCommandData              [6] IMPLICIT CommandData
    } OPTIONAL,
   originatingPointCode         [19] IMPLICIT PointCode OPTIONAL
}

-- Following record is generated for an inbound MO-ForwardSM operation
-- that is considered to be suspect.

SuspectMoFwdSm ::= SEQUENCE {
    timestamp                   [0] IMPLICIT GeneralizedTime,
    routingAction               [1] IMPLICIT MoFwdSmRoutingAction,
    -- rejectInfo and responseInfo are mutually exclusive.
    -- rejectInfo is included when the action is 'discardWithAck',
    -- 'discardWithNak', or 'discardWithNoResponse'.
    -- responseInfo is included otherwise.
    rejectInfo                  [2] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        moRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        moExtConditionRule          [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    responseInfo                [3] IMPLICIT SEQUENCE {
        submissionResult            [0] IMPLICIT MoFwdSmSubmissionResult,
        moRoutingRule               [1] IMPLICIT NameString
    } OPTIONAL,
    ignoredRejectCauses         [4] IMPLICIT IgnoredRejectCauses,
    -- sccpCgPaOfFirstSegment and sccpCdPaOfFirstSegment are only included when
    -- the MO-ForwardSM has been received in a segmented TCAP dialogue.
    sccpCgPaOfFirstSegment      [6] IMPLICIT SccpAddress OPTIONAL,
    sccpCdPaOfFirstSegment      [7] IMPLICIT SccpAddress OPTIONAL,
    sccpCgPa                    [8] IMPLICIT SccpAddress,
    sccpCdPa                    [9] IMPLICIT SccpAddress,
    mapSmsc                    [12] IMPLICIT GsmAddress OPTIONAL,
    mapMsisdn                  [13] IMPLICIT GsmAddress OPTIONAL,
    mapImsi                    [14] IMPLICIT Imsi OPTIONAL,
    -- smsSubmit and smsCommand are mutually exclusive.
    smsSubmit                  [15] IMPLICIT SEQUENCE {
        smsServices                 [0] IMPLICIT SmsSubmitServices,
        smsMessageReference         [1] IMPLICIT MessageReference,
        smsRecipient                [2] IMPLICIT GsmAddress,
        smsProtocolId               [3] IMPLICIT ProtocolId,
        smsDataCodingScheme         [4] IMPLICIT DataCodingScheme,
        smsValidityPeriod           [5] IMPLICIT GeneralizedTime OPTIONAL,
        smsUserDataHeader           [6] IMPLICIT UserDataHeader OPTIONAL,
```

```
        smsUserData                 [7] IMPLICIT UserData
    } OPTIONAL,
    smsCommand                  [16] IMPLICIT SEQUENCE {
        smsServices                 [0] IMPLICIT SmsCommandServices,
        smsMessageReference         [1] IMPLICIT MessageReference,
        smsProtocolId               [2] IMPLICIT ProtocolId,
        smsCommandType              [3] IMPLICIT CommandType,
        smsMessageNumber            [4] IMPLICIT MessageNumber,
        smsRecipient                [5] IMPLICIT GsmAddress,
        smsCommandData              [6] IMPLICIT CommandData
    } OPTIONAL,
   originatingPointCode         [19] IMPLICIT PointCode OPTIONAL,
    -- infoFromHlr comprises information pertaining to the originator of the
    -- MO-ForwardSM
    infoFromHlr                 [30] IMPLICIT SEQUENCE {
        mapImsi                     [0] IMPLICIT Imsi OPTIONAL,
        mapMsc                      [1] IMPLICIT GsmAddress OPTIONAL,
        mapSgsn                     [2] IMPLICIT GsmAddress OPTIONAL
    } OPTIONAL
}

-- Following record is generated for an inbound SendRoutingInfoForSM operation
-- that is considered to be trusted.

TrustedSriSm ::= SEQUENCE {
    timestamp                   [0] IMPLICIT GeneralizedTime,
    routingAction               [1] IMPLICIT SriSmRoutingAction,
    -- rejectInfo and responseInfo are mutually exclusive.
    -- rejectInfo is included when the action is 'blockWithTemporaryError',
    -- 'blockWithPermanentError', 'blockWithNoResponse', or 'blockWithAck'.
    -- responseInfo is included otherwise.
    rejectInfo                  [2] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        mtRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        mtExtConditionRule          [2] IMPLICIT NameString OPTIONAL,
        sriqRoutingRule             [3] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    responseInfo                [3] IMPLICIT SEQUENCE {
        queryResult                 [0] SriSmQueryResult,
        mapImsi                     [1] Imsi OPTIONAL,
        mapLmsi                     [2] Lmsi OPTIONAL,
        mapMsc                      [3] GsmAddress OPTIONAL,
        mapSgsn                     [4] GsmAddress OPTIONAL,
        mtRoutingRule               [6] IMPLICIT NameString OPTIONAL,
        sriqRoutingRule             [7] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    sccpCgPa                    [8] SccpAddress,
    sccpCdPa                    [9] SccpAddress,
    mapSmsc                     [12] GsmAddress,
    mapMsisdn                   [13] GsmAddress
}

-- Following record is generated for an inbound SendRoutingInfoForSM operation
-- that is considered to be suspect.

SuspectSriSm ::= SEQUENCE {
    timestamp                   [0] IMPLICIT GeneralizedTime,
    routingAction               [1] IMPLICIT SriSmRoutingAction,
    -- rejectInfo and responseInfo are mutually exclusive.
    -- rejectInfo is included when the action is 'discardWithAck',
    -- 'discardWithNak', or 'discardWithNoResponse'.
    -- responseInfo is included otherwise.
    rejectInfo                  [2] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        mtRoutingRule               [1] IMPLICIT NameString OPTIONAL,
```

```
        mtExtConditionRule          [2] IMPLICIT NameString OPTIONAL,
        sriqRoutingRule             [3] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    responseInfo                [3] IMPLICIT SEQUENCE {
        queryResult                 [0] IMPLICIT SriSmQueryResult,
        mapImsi                     [1] IMPLICIT Imsi OPTIONAL,
        mapLmsi                     [2] IMPLICIT Lmsi OPTIONAL,
        mapMsc                      [3] IMPLICIT GsmAddress OPTIONAL,
        mapSgsn                     [4] IMPLICIT GsmAddress OPTIONAL,
        scrambledImsi               [5] IMPLICIT Imsi OPTIONAL,
        mtRoutingRule               [6] IMPLICIT NameString OPTIONAL,
        sriqRoutingRule             [7] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    ignoredRejectCauses         [4] IMPLICIT IgnoredRejectCauses,
    sccpCgPa                    [8] IMPLICIT SccpAddress,
    sccpCdPa                    [9] IMPLICIT SccpAddress,
    mapSmsc                     [12] IMPLICIT GsmAddress,
    mapMsisdn                   [13] IMPLICIT GsmAddress
}

-- Following record is generated for an inbound MT-ForwardSM operation
-- that is considered to be trusted.

TrustedMtFwdSm ::= SEQUENCE {
    timestamp                   [0] IMPLICIT GeneralizedTime,
    routingAction               [1] IMPLICIT MtFwdSmRoutingAction,
    -- rejectInfo and responseInfo are mutually exclusive.
    -- rejectInfo is included when the action is 'discardWithAck',
    -- 'discardWithNak', or 'discardWithNoResponse'.
    -- responseInfo is included otherwise.
    rejectInfo                  [2] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        mtRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        mtExtConditionRule          [2] IMPLICIT NameString OPTIONAL,
        mtiRoutingRule              [3] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    responseInfo                [3] IMPLICIT SEQUENCE {
        deliveryResult              [0] IMPLICIT MtFwdSmDeliveryResult,
        mtRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        mtiRoutingRule              [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    -- sccpCgPaOfFirstSegment and sccpCdPaOfFirstSegment are only included when
    -- the MT-ForwardSM has been received in a segmented TCAP dialogue.
    sccpCgPaOfFirstSegment      [6] IMPLICIT SccpAddress OPTIONAL,
    sccpCdPaOfFirstSegment      [7] IMPLICIT SccpAddress OPTIONAL,
    sccpCgPa                    [8] IMPLICIT SccpAddress,
    sccpCdPa                    [9] IMPLICIT SccpAddress,
    mapSmsc                     [12] IMPLICIT GsmAddress OPTIONAL,
    mapImsi                     [13] IMPLICIT Imsi OPTIONAL,
    mapLmsi                     [14] IMPLICIT Lmsi OPTIONAL,
    smsDeliver                  [15] IMPLICIT SEQUENCE {
        smsServices                 [0] IMPLICIT SmsDeliverServices,
        smsOriginator               [1] IMPLICIT GsmAddress,
        smsProtocolId               [2] IMPLICIT ProtocolId,
        smsDataCodingScheme         [3] IMPLICIT DataCodingScheme,
        smsScTimestamp              [4] IMPLICIT GeneralizedTime,
        smsUserDataHeader           [5] IMPLICIT UserDataHeader OPTIONAL,
        smsUserData                 [6] IMPLICIT UserData
    } OPTIONAL,
    statusReport                [16] IMPLICIT SEQUENCE {
        smsServices                 [0] IMPLICIT StatusReportServices,
        smsMessageReference         [1] IMPLICIT MessageReference,
        smsRecipient                [2] IMPLICIT GsmAddress,
        smsScTimestamp              [3] IMPLICIT GeneralizedTime,
        smsDischargeTime            [4] IMPLICIT GeneralizedTime,
```

```
        smsStatus                   [5] IMPLICIT Status
    } OPTIONAL,
    correlatedSriSm             [30] IMPLICIT SEQUENCE {
        sccpCgPa                    [0] IMPLICIT SccpAddress OPTIONAL,
        mapSmsc                     [1] IMPLICIT GsmAddress OPTIONAL,
        mapMsisdn                   [2] IMPLICIT GsmAddress,
        mapImsi                     [3] IMPLICIT Imsi,
        mapLmsi                     [4] IMPLICIT Lmsi OPTIONAL,
        mapMsc                      [5] IMPLICIT GsmAddress OPTIONAL,
        mapSgsn                     [6] IMPLICIT GsmAddress OPTIONAL
    } OPTIONAL,
    outboundMt                  [21] IMPLICIT OutboundMt OPTIONAL
}

-- Following record is generated for an inbound MT-ForwardSM operation
-- that is considered to be suspect.

SuspectMtFwdSm ::= SEQUENCE {
    timestamp                   [0] IMPLICIT GeneralizedTime,
    routingAction               [1] IMPLICIT MtFwdSmRoutingAction,
    -- rejectInfo and responseInfo are mutually exclusive.
    -- rejectInfo is included when the action is 'discardWithAck',
    -- 'discardWithNak', or 'discardWithNoResponse'.
    -- responseInfo is included otherwise.
    rejectInfo                  [2] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        mtRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        mtExtConditionRule          [2] IMPLICIT NameString OPTIONAL,
        mtiRoutingRule              [3] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    responseInfo                [3] IMPLICIT SEQUENCE {
        deliveryResult              [0] IMPLICIT MtFwdSmDeliveryResult,
        mtRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        mtiRoutingRule              [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    ignoredRejectCauses         [4] IMPLICIT IgnoredRejectCauses,
    -- sccpCgPaOfFirstSegment and sccpCdPaOfFirstSegment are only included when
    -- the MT-ForwardSM has been received in a segmented TCAP dialogue.
    sccpCgPaOfFirstSegment      [6] IMPLICIT SccpAddress OPTIONAL,
    sccpCdPaOfFirstSegment      [7] IMPLICIT SccpAddress OPTIONAL,
    sccpCgPa                    [8] IMPLICIT SccpAddress,
    sccpCdPa                    [9] IMPLICIT SccpAddress,
    mapSmsc                     [12] IMPLICIT GsmAddress OPTIONAL,
    mapImsi                     [13] IMPLICIT Imsi OPTIONAL,
    -- smsDeliver and statusReport are mutually exclusive.
    smsDeliver                  [15] IMPLICIT SEQUENCE {
        smsServices                 [0] IMPLICIT SmsDeliverServices,
        smsOriginator               [1] IMPLICIT GsmAddress,
        smsProtocolId               [2] IMPLICIT ProtocolId,
        smsDataCodingScheme         [3] IMPLICIT DataCodingScheme,
        smsScTimestamp              [4] IMPLICIT GeneralizedTime,
        smsUserDataHeader           [5] IMPLICIT UserDataHeader OPTIONAL,
        smsUserData                 [6] IMPLICIT UserData
    } OPTIONAL,
    statusReport                [16] IMPLICIT SEQUENCE {
        smsServices                 [0] IMPLICIT StatusReportServices,
        smsMessageReference         [1] IMPLICIT MessageReference,
        smsRecipient                [2] IMPLICIT GsmAddress,
        smsScTimestamp              [3] IMPLICIT GeneralizedTime,
        smsDischargeTime            [4] IMPLICIT GeneralizedTime,
        smsStatus                   [5] IMPLICIT Status
    } OPTIONAL,
    correlatedSriSm             [30] IMPLICIT SEQUENCE {
        sccpCgPa                    [0] IMPLICIT SccpAddress OPTIONAL,
        mapSmsc                     [1] IMPLICIT GsmAddress OPTIONAL,
```

```
        mapMsisdn                   [2] IMPLICIT GsmAddress,
        mapImsi                     [3] IMPLICIT Imsi,
        mapLmsi                     [4] IMPLICIT Lmsi OPTIONAL,
        mapMsc                      [5] IMPLICIT GsmAddress OPTIONAL,
        mapSgsn                     [6] IMPLICIT GsmAddress OPTIONAL
    } OPTIONAL
}

HssQuery ::= SEQUENCE {
    timestamp                   [0] IMPLICIT GeneralizedTime OPTIONAL,
    routingAction               [1] IMPLICIT SriSmRoutingAction OPTIONAL,
    -- rejectInfo and responseInfo are mutually exclusive.
    -- rejectInfo is included when the action is 'blockWithTemporaryError',
    -- 'blockWithPermanentError', 'blockWithNoResponse', or 'blockWithAck'.
    -- responseInfo is included otherwise.
    rejectInfo                  [2] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        mtRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        mtExtConditionRule          [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    -- userDataAnswer
    shPullResponseInfo          [3] IMPLICIT SEQUENCE {
        queryResult                 [0] IMPLICIT SriSmQueryResult,
        resultCode                  [1] IMPLICIT DiameterShResultCode OPTIONAL,
        experimentalResult          [2] IMPLICIT DiameterShResultCode OPTIONAL,
        imsUserState                [3] IMPLICIT ImsUserState OPTIONAL,
        scscfName                   [4] IMPLICIT SipUrl OPTIONAL,
        hssRule                     [5] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    -- imsPublicUserIdentity, imsPublicServiceIdentity Url, now MSISDN
    userIdentity                [13] IMPLICIT GsmAddressInfo OPTIONAL,
    iiw                         [60] IMPLICIT IiwId OPTIONAL
}

-- Following record is generated for an inbound MO-ForwardSM operation
-- that is considered to be trusted, suspect, or a status report.

MoFwdSmWithCountryAndNetworkInfo ::= SEQUENCE {
    timestamp                   [0] IMPLICIT GeneralizedTime,
    routingAction               [1] IMPLICIT MoFwdSmRoutingAction,
    -- rejectInfo and responseInfo are mutually exclusive.
    -- rejectInfo is included when the action is 'discardWithAck',
    -- 'discardWithNak', or 'discardWithNoResponse'.
    -- responseInfo is included otherwise.
    rejectInfo                  [2] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        moRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        moExtConditionRule          [2] IMPLICIT NameString OPTIONAL,
        mnpViolation                [3] IMPLICIT MnpViolation OPTIONAL
    } OPTIONAL,
    responseInfo                [3] IMPLICIT SEQUENCE {
        submissionResult              [0] IMPLICIT MoFwdSmSubmissionResult,
        moRoutingRule               [1] IMPLICIT NameString
    } OPTIONAL,
    ignoredRejectCauses         [4] IMPLICIT IgnoredRejectCauses OPTIONAL,
    -- sccpCgPaOfFirstSegment and sccpCdPaOfFirstSegment are only included when
    -- the MO-ForwardSM has been received in a segmented TCAP dialogue.
    sccpCgPaOfFirstSegment      [6] IMPLICIT SccpAddressInfo OPTIONAL,
    sccpCdPaOfFirstSegment      [7] IMPLICIT SccpAddressInfo OPTIONAL,
    sccpCgPa                    [8] IMPLICIT SccpAddressInfo,
    sccpCdPa                    [9] IMPLICIT SccpAddressInfo,
    mapSmsc                     [12] IMPLICIT GsmAddressInfo OPTIONAL,
    mapMsisdn                   [13] IMPLICIT GsmAddressInfo OPTIONAL,
    mapImsi                     [14] IMPLICIT ImsiInfo OPTIONAL,
    -- smsSubmit and smsCommand are mutually exclusive.
```

```
    smsSubmit                   [15] IMPLICIT SEQUENCE {
        smsServices                  [0] IMPLICIT SmsSubmitServices,
        smsMessageReference          [1] IMPLICIT MessageReference,
        smsRecipient                 [2] IMPLICIT GsmAddressInfo,
        smsProtocolId                [3] IMPLICIT ProtocolId,
        smsDataCodingScheme          [4] IMPLICIT DataCodingScheme,
        smsValidityPeriod            [5] IMPLICIT GeneralizedTime OPTIONAL,
        smsUserDataHeader            [6] IMPLICIT UserDataHeader OPTIONAL,
        smsUserData                  [7] IMPLICIT UserData
    } OPTIONAL,
    smsCommand                  [16] IMPLICIT SEQUENCE {
        smsServices                  [0] IMPLICIT SmsCommandServices,
        smsMessageReference          [1] IMPLICIT MessageReference,
        smsProtocolId                [2] IMPLICIT ProtocolId,
        smsCommandType               [3] IMPLICIT CommandType,
        smsMessageNumber             [4] IMPLICIT MessageNumber,
        smsRecipient                 [5] IMPLICIT GsmAddressInfo,
        smsCommandData               [6] IMPLICIT CommandData
    } OPTIONAL,
    originatingPointCode        [19] IMPLICIT PointCode OPTIONAL,
    outboundMo                  [20] IMPLICIT OutboundMo OPTIONAL,
    outboundMt                  [21] IMPLICIT OutboundMt OPTIONAL,
    outboundAo                  [22] IMPLICIT OutboundAo OPTIONAL,
    outboundAt                  [23] IMPLICIT OutboundAt OPTIONAL,
    storage                     [24] IMPLICIT StorageInfo OPTIONAL,
    -- infoFromHlr comprises information pertaining to the originator of the
    -- MO-ForwardSM
    infoFromHlr                 [30] IMPLICIT SEQUENCE {
        mapImsi                      [0] IMPLICIT ImsiInfo,
        mapMsc                       [1] IMPLICIT GsmAddressInfo OPTIONAL,
        mapSgsn                      [2] IMPLICIT GsmAddressInfo OPTIONAL
    } OPTIONAL,
    recipientRoutingNumber      [33] IMPLICIT RoutingNumber OPTIONAL,
    ecResponseData              [34] IMPLICIT SEQUENCE OF EcResponseData OPTIONAL,
    ssiInfo                     [50] IMPLICIT SsiServiceInfo OPTIONAL,
    --2184
    hssQuery                    [60] IMPLICIT HssQuery OPTIONAL,
    outboundSip                 [61] IMPLICIT OutboundSip OPTIONAL,
    originalIiw                 [62] IMPLICIT IiwId OPTIONAL
}


-- Following record is generated for an inbound SendRoutingInfoForSM operation
-- that is considered to be trusted.

TrustedSriSmWithCountryAndNetworkInfo ::= SEQUENCE {
    timestamp                   [0] GeneralizedTime,
    routingAction               [1] SriSmRoutingAction,
    -- rejectInfo and responseInfo are mutually exclusive.
    -- rejectInfo is included when the action is 'discardWithAck',
    -- 'discardWithNak', or 'discardWithNoResponse'.
    -- responseInfo is included otherwise.
    rejectInfo                  [2] IMPLICIT SEQUENCE {
        rejectCause                  [0] IMPLICIT RejectCause,
        mtRoutingRule                [1] NameString OPTIONAL,
        mtExtConditionRule           [2] NameString OPTIONAL,
        sriqRoutingRule              [3] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    responseInfo                [3] IMPLICIT SEQUENCE {
        queryResult                  [0] SriSmQueryResult,
        mapImsi                      [1] ImsiInfo OPTIONAL,
        mapLmsi                      [2] Lmsi OPTIONAL,
        mapMsc                       [3] GsmAddressInfo OPTIONAL,
        mapSgsn                      [4] GsmAddressInfo OPTIONAL,
        mtRoutingRule                [6] IMPLICIT NameString OPTIONAL,
```

```
        sriqRoutingRule             [7] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    sccpCgPa                    [8] SccpAddressInfo,
    sccpCdPa                    [9] SccpAddressInfo,
    mapSmsc                     [12] GsmAddressInfo,
    mapMsisdn                   [13] GsmAddressInfo,
    ecResponseData              [34] IMPLICIT SEQUENCE OF EcResponseData OPTIONAL,
    mtRoutingRuleSkipped        [49] IMPLICIT NameString OPTIONAL,
    ssiInfo                     [50] IMPLICIT SsiServiceInfo OPTIONAL,
    --2184
    hssQuery                    [60] IMPLICIT HssQuery OPTIONAL
}

-- Following record is generated for an inbound SendRoutingInfoForSM operation
-- that is considered to be suspect.

SuspectSriSmWithCountryAndNetworkInfo ::= SEQUENCE {
    timestamp                   [0] IMPLICIT GeneralizedTime,
    routingAction               [1] IMPLICIT SriSmRoutingAction,
    -- rejectInfo and responseInfo are mutually exclusive.
    -- rejectInfo is included when the action is 'discardWithAck',
    -- 'discardWithNak', or 'discardWithNoResponse'.
    -- responseInfo is included otherwise.
    rejectInfo                  [2] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        mtRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        mtExtConditionRule          [2] IMPLICIT NameString OPTIONAL,
        sriqRoutingRule             [3] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    responseInfo                [3] IMPLICIT SEQUENCE {
        queryResult                 [0] IMPLICIT SriSmQueryResult,
        mapImsi                     [1] IMPLICIT ImsiInfo OPTIONAL,
        mapLmsi                     [2] IMPLICIT Lmsi OPTIONAL,
        mapMsc                      [3] IMPLICIT GsmAddressInfo OPTIONAL,
        mapSgsn                     [4] IMPLICIT GsmAddressInfo OPTIONAL,
        scrambledImsi               [5] IMPLICIT ImsiInfo OPTIONAL,
        mtRoutingRule               [6] IMPLICIT NameString OPTIONAL,
        sriqRoutingRule             [7] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    ignoredRejectCauses         [4] IMPLICIT IgnoredRejectCauses,
    sccpCgPa                    [8] IMPLICIT SccpAddressInfo,
    sccpCdPa                    [9] IMPLICIT SccpAddressInfo,
    mapSmsc                     [12] IMPLICIT GsmAddressInfo,
    mapMsisdn                   [13] IMPLICIT GsmAddressInfo,
    ecResponseData              [34] IMPLICIT SEQUENCE OF EcResponseData OPTIONAL,
    mtRoutingRuleSkipped        [49] IMPLICIT NameString OPTIONAL,
    ssiInfo                     [50] IMPLICIT SsiServiceInfo OPTIONAL,
    --2184
    hssQuery                    [60] IMPLICIT HssQuery OPTIONAL
}

-- Following record is generated for an inbound MT-ForwardSM operation
-- that is considered to be trusted.

TrustedMtFwdSmWithCountryAndNetworkInfo ::= SEQUENCE {
    timestamp                   [0] IMPLICIT GeneralizedTime,
    routingAction               [1] IMPLICIT MtFwdSmRoutingAction,
    -- rejectInfo and responseInfo are mutually exclusive.
    -- rejectInfo is included when the action is 'discardWithAck',
    -- 'discardWithNak', or 'discardWithNoResponse'.
    -- responseInfo is included otherwise.
    rejectInfo                  [2] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        mtRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        mtExtConditionRule          [2] IMPLICIT NameString OPTIONAL,
```

```
        mtiRoutingRule              [3] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    responseInfo                [3] IMPLICIT SEQUENCE {
        deliveryResult              [0] IMPLICIT MtFwdSmDeliveryResult,
        mtRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        mtiRoutingRule              [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    -- sccpCgPaOfFirstSegment and sccpCdPaOfFirstSegment are only included when
    -- the MT-ForwardSM has been received in a segmented TCAP dialogue.
    sccpCgPaOfFirstSegment      [6] IMPLICIT SccpAddressInfo OPTIONAL,
    sccpCdPaOfFirstSegment      [7] IMPLICIT SccpAddressInfo OPTIONAL,
    sccpCgPa                    [8] IMPLICIT SccpAddressInfo,
    sccpCdPa                    [9] IMPLICIT SccpAddressInfo,
    mapSmsc                     [12] IMPLICIT GsmAddressInfo OPTIONAL,
    mapImsi                     [13] IMPLICIT ImsiInfo OPTIONAL,
    mapLmsi                     [14] IMPLICIT Lmsi OPTIONAL,
    smsDeliver                  [15] IMPLICIT SEQUENCE {
        smsServices                 [0] IMPLICIT SmsDeliverServices,
        smsOriginator               [1] IMPLICIT GsmAddressInfo,
        smsProtocolId               [2] IMPLICIT ProtocolId,
        smsDataCodingScheme         [3] IMPLICIT DataCodingScheme,
        smsScTimestamp              [4] IMPLICIT GeneralizedTime,
        smsUserDataHeader           [5] IMPLICIT UserDataHeader OPTIONAL,
        smsUserData                 [6] IMPLICIT UserData
    } OPTIONAL,
    statusReport                [16] IMPLICIT SEQUENCE {
        smsServices                 [0] IMPLICIT StatusReportServices,
        smsMessageReference         [1] IMPLICIT MessageReference,
        smsRecipient                [2] IMPLICIT GsmAddressInfo,
        smsScTimestamp              [3] IMPLICIT GeneralizedTime,
        smsDischargeTime            [4] IMPLICIT GeneralizedTime,
        smsStatus                   [5] IMPLICIT Status
    } OPTIONAL,
    correlatedSriSm             [30] IMPLICIT SEQUENCE {
        sccpCgPa                    [0] IMPLICIT SccpAddressInfo OPTIONAL,
        mapSmsc                     [1] IMPLICIT GsmAddressInfo OPTIONAL,
        mapMsisdn                   [2] IMPLICIT GsmAddressInfo,
        mapImsi                     [3] IMPLICIT ImsiInfo,
        mapLmsi                     [4] IMPLICIT Lmsi OPTIONAL,
        mapMsc                      [5] IMPLICIT GsmAddressInfo OPTIONAL,
        mapSgsn                     [6] IMPLICIT GsmAddressInfo OPTIONAL
    }  OPTIONAL,
    --BG19606-19607
    outboundMt                  [21] IMPLICIT OutboundMt OPTIONAL,
    ecResponseData              [34] IMPLICIT SEQUENCE OF EcResponseData OPTIONAL,
    mtRoutingRuleSkipped        [49] IMPLICIT NameString OPTIONAL,
    ssiInfo                     [50] IMPLICIT SsiServiceInfo OPTIONAL,
    --2184
    hssQuery                    [60] IMPLICIT HssQuery OPTIONAL,
    outboundSip                 [61] IMPLICIT OutboundSip OPTIONAL
}

-- Following record is generated for an inbound MT-ForwardSM operation
-- that is considered to be suspect.

SuspectMtFwdSmWithCountryAndNetworkInfo ::= SEQUENCE {
    timestamp                   [0] IMPLICIT GeneralizedTime,
    routingAction               [1] IMPLICIT MtFwdSmRoutingAction,
    -- rejectInfo and responseInfo are mutually exclusive.
    -- rejectInfo is included when the action is 'discardWithAck',
    -- 'discardWithNak', or 'discardWithNoResponse'.
    -- responseInfo is included otherwise.
    rejectInfo                  [2] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        mtRoutingRule               [1] IMPLICIT NameString OPTIONAL,
```

```
        mtExtConditionRule          [2] IMPLICIT NameString OPTIONAL,
        mtiRoutingRule              [3] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    responseInfo                [3] IMPLICIT SEQUENCE {
        deliveryResult              [0] IMPLICIT MtFwdSmDeliveryResult,
        mtRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        mtiRoutingRule              [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    ignoredRejectCauses         [4] IMPLICIT IgnoredRejectCauses,
    -- sccpCgPaOfFirstSegment and sccpCdPaOfFirstSegment are only included when
    -- the MT-ForwardSM has been received in a segmented TCAP dialogue.
    sccpCgPaOfFirstSegment      [6] IMPLICIT SccpAddressInfo OPTIONAL,
    sccpCdPaOfFirstSegment      [7] IMPLICIT SccpAddressInfo OPTIONAL,
    sccpCgPa                    [8] IMPLICIT SccpAddressInfo,
    sccpCdPa                    [9] IMPLICIT SccpAddressInfo,
    mapSmsc                     [12] IMPLICIT GsmAddressInfo OPTIONAL,
    mapImsi                     [13] IMPLICIT ImsiInfo OPTIONAL,
    -- smsDeliver and statusReport are mutually exclusive.
    smsDeliver                  [15] IMPLICIT SEQUENCE {
        smsServices                 [0] IMPLICIT SmsDeliverServices,
        smsOriginator               [1] IMPLICIT GsmAddressInfo,
        smsProtocolId               [2] IMPLICIT ProtocolId,
        smsDataCodingScheme         [3] IMPLICIT DataCodingScheme,
        smsScTimestamp              [4] IMPLICIT GeneralizedTime,
        smsUserDataHeader           [5] IMPLICIT UserDataHeader OPTIONAL,
        smsUserData                 [6] IMPLICIT UserData
    } OPTIONAL,
    statusReport                [16] IMPLICIT SEQUENCE {
        smsServices                 [0] IMPLICIT StatusReportServices,
        smsMessageReference         [1] IMPLICIT MessageReference,
        smsRecipient                [2] IMPLICIT GsmAddressInfo,
        smsScTimestamp              [3] IMPLICIT GeneralizedTime,
        smsDischargeTime            [4] IMPLICIT GeneralizedTime,
        smsStatus                   [5] IMPLICIT Status
    } OPTIONAL,
    correlatedSriSm             [30] IMPLICIT SEQUENCE {
        sccpCgPa                    [0] IMPLICIT SccpAddressInfo OPTIONAL,
        mapSmsc                     [1] IMPLICIT GsmAddressInfo OPTIONAL,
        mapMsisdn                   [2] IMPLICIT GsmAddressInfo,
        mapImsi                     [3] IMPLICIT ImsiInfo,
        mapLmsi                     [4] IMPLICIT Lmsi OPTIONAL,
        mapMsc                      [5] IMPLICIT GsmAddressInfo OPTIONAL,
        mapSgsn                     [6] IMPLICIT GsmAddressInfo OPTIONAL
    }  OPTIONAL,
    ecResponseData              [34] IMPLICIT SEQUENCE OF EcResponseData OPTIONAL,
    mtRoutingRuleSkipped        [49] IMPLICIT NameString OPTIONAL,
    ssiInfo                     [50] IMPLICIT SsiServiceInfo OPTIONAL,
    --2184
    hssQuery                    [60] IMPLICIT HssQuery OPTIONAL,
    outboundSip                 [61] IMPLICIT OutboundSip OPTIONAL
}

-- Following record is generated when an AO/SM has been received from an
-- application.

ReceivedSubmitSmWithCountryAndNetworkInfo ::= SEQUENCE {
    timestamp                   [0] IMPLICIT GeneralizedTime,
    routingAction               [1] IMPLICIT AoRoutingAction,
    -- rejectInfo and responseInfo are mutually exclusive.
    -- rejectInfo is included when the action is 'discardWithAck' or
    -- 'discardWithNak'.
    -- responseInfo is included otherwise.
    rejectInfo                  [2] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        aoRoutingRule               [1] IMPLICIT NameString OPTIONAL,
```

```
        aoExtConditionRule          [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    responseInfo                [3] IMPLICIT SEQUENCE {
        submissionResult            [0] IMPLICIT AoSubmissionResult,
        routingErrorCode            [1] IMPLICIT ErrorCode OPTIONAL,
        aoRoutingRule               [2] IMPLICIT NameString,
        serviceCentreTimestamp      [3] IMPLICIT GeneralizedTime OPTIONAL
    } OPTIONAL,
    applicationName             [17] IMPLICIT NameString,
    applicationShortNumber      [18] IMPLICIT GsmAddress,
    messageFields               [19] IMPLICIT SEQUENCE {
        -- originatorAddress and alphanumericOriginator are mutually
        -- exclusive
        -- recipientAddress and alphanumericRecipient are mutually
        -- exclusive
        originatorAddress           [2] GsmAddressInfo OPTIONAL,
        recipientAddress            [4] GsmAddressInfo OPTIONAL,
        dataCodingScheme            [5] DataCodingScheme OPTIONAL,
        protocolIdentifier          [6] ProtocolId OPTIONAL,
        notificationType            [7] NotificationType,
        userData                    [10] UserData,
        userDataHeader              [11] UserDataHeader OPTIONAL,
        moreMessagesToSend          [12] BOOLEAN OPTIONAL,
        priority                    [13] Priority,
        replyPathIndicator          [14] BOOLEAN OPTIONAL,
        deferredDeliveryTime        [17] GeneralizedTime OPTIONAL,
        validityPeriod              [18] GeneralizedTime OPTIONAL,
        singleShotIndicator         [19] BOOLEAN,
        billingIdentifier           [20] BillingId OPTIONAL,
        tariffClass                 [25] TariffClass OPTIONAL,
        serviceDescription          [26] ServiceDescription OPTIONAL,
        alphanumericOriginator      [30] AlphanumericAddress OPTIONAL,
        alphanumericRecipient       [31] AlphanumericAddress OPTIONAL,
        portNumber                  [34] PortNumber OPTIONAL,
        sourcePort                  [46] PortNumber OPTIONAL,
        destinationPort             [47] PortNumber OPTIONAL,
        endToEndAckRequest          [50] EndToEndAckRequest OPTIONAL,
        endToEndMessageType         [52] EndToEndMessageType OPTIONAL,
        messageReference            [53] MessageReference OPTIONAL,
        privacy                     [54] Privacy OPTIONAL,
        numberOfMessages            [55] NumberOfMessages OPTIONAL,
        language                    [56] Language OPTIONAL,
        payloadType                 [63] PayloadType OPTIONAL,
        sourceSubAddress            [67] SubAddress OPTIONAL,
        destSubAddress              [68] SubAddress OPTIONAL,
        userResponseCode            [69] UserResponseCode OPTIONAL,
        displayTime                 [70] DisplayTime OPTIONAL,
        callbackNumbers             [71] SEQUENCE OF CallbackNumber OPTIONAL,
        msValidityIndicator         [72] MsValidityIndicator OPTIONAL,
        msValidityPeriod            [73] MsValidityPeriod OPTIONAL,
        alertOnMessageDelivery      [74] AlertOnMessageDelivery OPTIONAL,
        smsSignal                   [75] SmsSignal OPTIONAL,
        sourceBearerType            [76] BearerType OPTIONAL,
        destBearerType              [77] BearerType OPTIONAL,
        smDefaultMsgId              [78] SmDefaultMessageId OPTIONAL,
        sourceNetworkType           [79] NetworkType OPTIONAL,
        destNetworkType             [80] NetworkType OPTIONAL
    },
    outboundMt                  [21] IMPLICIT OutboundMt OPTIONAL,
    outboundAo                  [22] IMPLICIT OutboundAo OPTIONAL,
    outboundAt                  [23] IMPLICIT OutboundAt OPTIONAL,
    storage                     [24] IMPLICIT StorageInfo OPTIONAL,
    recipientRoutingNumber      [33] IMPLICIT RoutingNumber OPTIONAL,
    ecResponseData              [34] IMPLICIT SEQUENCE OF EcResponseData OPTIONAL,
    originalMessageFields       [46] OriginalMessageFields OPTIONAL,
```

```
    ssiInfo                     [50] IMPLICIT SsiServiceInfo OPTIONAL,
    --2184
    hssQuery                    [60] IMPLICIT HssQuery OPTIONAL,
    outboundSip                 [61] IMPLICIT OutboundSip OPTIONAL
}

-- Following record is generated when an AT/SM has been received from an SMSC.

ReceivedDeliverSmWithCountryAndNetworkInfo ::= SEQUENCE {
    timestamp                   [0] IMPLICIT GeneralizedTime,
    routingAction               [1] IMPLICIT AtiRoutingAction,
    -- rejectInfo and responseInfo are mutually exclusive.
    rejectInfo                  [2] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        atRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        atExtConditionRule          [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    responseInfo                [3] IMPLICIT SEQUENCE {
        deliveryResult              [0] IMPLICIT AtDeliveryResult,
        routingErrorCode            [1] IMPLICIT ErrorCode OPTIONAL,
        atRoutingRule               [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    applicationName             [17] IMPLICIT NameString,
    applicationShortNumber      [18] IMPLICIT GsmAddress,
    messageFields               [19] IMPLICIT SEQUENCE {
        -- recipientAddress and alphanumericRecipient are mutually
        -- exclusive
        originatorAddress           [2] GsmAddressInfo,
        recipientAddress            [4] GsmAddressInfo OPTIONAL,
        dataCodingScheme            [5] DataCodingScheme,
        protocolIdentifier          [6] ProtocolId,
        userData                    [10] UserData,
        userDataHeader              [11] UserDataHeader OPTIONAL,
        moreMessagesToSend          [12] BOOLEAN,
        priority                    [13] Priority OPTIONAL,
        replyPathIndicator          [14] BOOLEAN,
        serviceCentreTimestamp      [21] GeneralizedTime OPTIONAL,
        originatedImsi              [27] ImsiInfo OPTIONAL,
        originatedMscAddress        [28] GsmAddressInfo OPTIONAL,
        alphanumericOriginator      [30] AlphanumericAddress OPTIONAL,
        alphanumericRecipient       [31] AlphanumericAddress OPTIONAL,
        portNumber                  [34] PortNumber OPTIONAL,
        sourcePort                  [46] PortNumber OPTIONAL,
        destinationPort             [47] PortNumber OPTIONAL,
        endToEndAckRequest          [50] EndToEndAckRequest OPTIONAL,
        endToEndMessageType         [52] EndToEndMessageType OPTIONAL,
        messageReference            [53] MessageReference OPTIONAL,
        privacy                     [54] Privacy OPTIONAL,
        numberOfMessages            [55] NumberOfMessages OPTIONAL,
        language                    [56] Language OPTIONAL,
        payloadType                 [63] PayloadType OPTIONAL,
        sourceSubAddress            [67] SubAddress OPTIONAL,
        destSubAddress              [68] SubAddress OPTIONAL,
        userResponseCode            [69] UserResponseCode OPTIONAL,
        displayTime                 [70] DisplayTime OPTIONAL,
        callbackNumbers             [71] SEQUENCE OF CallbackNumber OPTIONAL,
        msValidityIndicator         [72] MsValidityIndicator OPTIONAL,
        msValidityPeriod            [73] MsValidityPeriod OPTIONAL,
        alertOnMessageDelivery      [74] AlertOnMessageDelivery OPTIONAL,
        smsSignal                   [75] SmsSignal OPTIONAL,
        sourceBearerType            [76] BearerType OPTIONAL,
        destBearerType              [77] BearerType OPTIONAL,
        smDefaultMsgId              [78] SmDefaultMessageId OPTIONAL,
        sourceNetworkType           [79] NetworkType OPTIONAL,
        destNetworkType             [80] NetworkType OPTIONAL
```

```
    },
    outboundAo                  [22] IMPLICIT OutboundAo OPTIONAL,
    outboundAt                  [23] IMPLICIT OutboundAt OPTIONAL,
    storage                     [24] IMPLICIT StorageInfo OPTIONAL,
    ecResponseData              [34] IMPLICIT SEQUENCE OF EcResponseData OPTIONAL,
    -- insideRejectInfo and insideResponseInfo are mutually exclusive.
    originalMessageFields       [46] OriginalMessageFields OPTIONAL,
    insideRejectInfo            [47] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        atiRoutingRule              [1] IMPLICIT NameString OPTIONAL,
        atiExtConditionRule         [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    insideResponseInfo          [48] IMPLICIT SEQUENCE {
        deliveryResult              [0] IMPLICIT AtDeliveryResult,
        routingErrorCode            [1] IMPLICIT ErrorCode OPTIONAL,
        atiRoutingRule              [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    ssiInfo                     [50] IMPLICIT SsiServiceInfo OPTIONAL
}

EventWithCountryAndNetworkInfo ::= SEQUENCE {
    timestamp                   [0] IMPLICIT GeneralizedTime,
    applicationName             [17] IMPLICIT NameString OPTIONAL,
    applicationShortNumber      [18] IMPLICIT GsmAddress OPTIONAL,
   messageFields               [19] IMPLICIT MessageFields,
    outboundMt                  [21] IMPLICIT OutboundMt OPTIONAL,
    outboundAt                  [23] IMPLICIT OutboundAt OPTIONAL,
    moreMessagesToSend          [25] IMPLICIT BOOLEAN OPTIONAL,
    numberOfPreviousAttempts    [26] IMPLICIT INTEGER OPTIONAL,
    serviceCentreTimestamp      [27] IMPLICIT GeneralizedTime,
    messageIdentifier           [28] IMPLICIT AmsMessageId OPTIONAL,
    isNotificationMessage       [36] IMPLICIT BOOLEAN OPTIONAL,
    unConditionalForward        [37] IMPLICIT BOOLEAN OPTIONAL,
-- The follow field is the internal event for diagnostics:
    event                       [38] IMPLICIT Event OPTIONAL,
    ssiInfo                     [50] IMPLICIT SsiServiceInfo OPTIONAL,
    --2184
    hssQuery                    [60] IMPLICIT HssQuery OPTIONAL,
    outboundSip                 [61] IMPLICIT OutboundSip OPTIONAL,
    atNotificationInfo          [62] IMPLICIT SEQUENCE {
       deliveryResult              [0] IMPLICIT AtDeliveryResult,
       routingErrorCode            [1] IMPLICIT ErrorCode OPTIONAL
    } OPTIONAL,
    mtStatusReportInfo          [63] IMPLICIT SEQUENCE {
       deliveryResult              [0] IMPLICIT MtFwdSmDeliveryResult
    } OPTIONAL,
   destEmailAddr             [65] IMPLICIT VisibleString (SIZE (1..2600)) OPTIONAL
}

CommandWithCountryAndNetworkInfo ::= SEQUENCE {
    timestamp                   [0] IMPLICIT GeneralizedTime OPTIONAL,
    applicationName             [17] IMPLICIT NameString OPTIONAL,
    applicationShortNumber      [18] IMPLICIT GsmAddress OPTIONAL,
    messageFields               [19] IMPLICIT MessageFields OPTIONAL,
    moreMessagesToSend          [25] IMPLICIT BOOLEAN OPTIONAL,
    serviceCentreTimestamp      [27] IMPLICIT GeneralizedTime OPTIONAL,
    messageIdentifier           [28] IMPLICIT AmsMessageId OPTIONAL,
-- The follow field is the internal event for diagnostics:
    event                       [38] IMPLICIT Event OPTIONAL,
    msgCommand                  [39] IMPLICIT MsgCommand OPTIONAL,
    smResult                    [40] IMPLICIT ModifyResult OPTIONAL,
    cmdOriginatorAddress        [41] IMPLICIT GsmAddressInfo OPTIONAL,
    cmdAlphanumericOriginator   [42] IMPLICIT AlphanumericAddress OPTIONAL,
    cmdRecipientAddress         [43] IMPLICIT GsmAddressInfo OPTIONAL,
    foundCount                  [44] IMPLICIT INTEGER OPTIONAL,
```

```
    cancelMode                  [45] IMPLICIT CancelMode OPTIONAL
}

RegistrationWithCountryAndNetworkInfo ::= SEQUENCE {
    timestamp                   [0] IMPLICIT GeneralizedTime,
    originalIiw                 [62] IMPLICIT IiwId OPTIONAL,
    registrationInfo            [63] IMPLICIT SEQUENCE {
        registrationType           [0] RegistrationType,
        expireValue                [1] ExpireValue,
        recipientAddress           [4] GsmAddressInfo
    },
    atmRequestResult            [64] IMPLICIT AtmRequestResult
}

MessageFields ::= SEQUENCE {
    -- originatorAddress and alphanumericOriginator are mutually exclusive
    -- recipientAddress and alphanumericRecipient are mutually exclusive
    protocol                    [0] Protocol OPTIONAL,
    originatorAddress           [2] GsmAddressInfo OPTIONAL,
    recipientAddress            [4] GsmAddressInfo OPTIONAL,
    dataCodingScheme            [5] DataCodingScheme OPTIONAL,
    protocolIdentifier          [6] ProtocolId OPTIONAL,
    notificationType            [7] NotificationType OPTIONAL,
    userData                    [10] UserData OPTIONAL,
    userDataHeader              [11] UserDataHeader OPTIONAL,
-- For reasons of backward compatibility this field is present in the parent
    moreMessagesToSend          [12] BOOLEAN OPTIONAL,
    priority                    [13] Priority OPTIONAL,
    replyPathIndicator          [14] BOOLEAN OPTIONAL,
    deferredDeliveryTime        [17] GeneralizedTime OPTIONAL,
    validityPeriod              [18] GeneralizedTime OPTIONAL,
    singleShotIndicator         [19] BOOLEAN OPTIONAL,
    billingIdentifier           [20] BillingId OPTIONAL,
    deliveryStatus              [22] DeliveryStatus OPTIONAL,
    errorCode                   [23] ErrorCode OPTIONAL,
    tariffClass                 [25] TariffClass OPTIONAL,
    serviceDescription          [26] ServiceDescription OPTIONAL,
    originatedImsi              [27] ImsiInfo OPTIONAL,
    originatedMscAddress        [28] GsmAddressInfo OPTIONAL,
    serviceCentreAddress        [29] GsmAddressInfo OPTIONAL,
    alphanumericOriginator      [30] AlphanumericAddress OPTIONAL,
    alphanumericRecipient       [31] AlphanumericAddress OPTIONAL,
    gsmStatusReportType         [32] GsmStatusReportType OPTIONAL,
    originatingPointCode        [33] PointCode OPTIONAL,
    portNumber                  [34] PortNumber OPTIONAL,
    gsmMessageReference         [40] MessageReference OPTIONAL,

    sourcePort                  [46] PortNumber OPTIONAL,
    destinationPort             [47] PortNumber OPTIONAL,
    endToEndAckRequest          [50] EndToEndAckRequest OPTIONAL,
    endToEndMessageType         [52] EndToEndMessageType OPTIONAL,
    messageReference            [53] MessageReference OPTIONAL,
    privacy                     [54] Privacy OPTIONAL,
    numberOfMessages            [55] NumberOfMessages OPTIONAL,
    language                    [56] Language OPTIONAL,
    payloadType                 [63] PayloadType OPTIONAL,
    sourceSubAddress            [67] SubAddress OPTIONAL,
    destSubAddress              [68] SubAddress OPTIONAL,
    userResponseCode            [69] UserResponseCode OPTIONAL,
    displayTime                 [70] DisplayTime OPTIONAL,
    callbackNumbers             [71] SEQUENCE OF CallbackNumber OPTIONAL,
    msValidityIndicator         [72] MsValidityIndicator OPTIONAL,
    msValidityPeriod            [73] MsValidityPeriod OPTIONAL,
    alertOnMessageDelivery      [74] AlertOnMessageDelivery OPTIONAL,
    smsSignal                   [75] SmsSignal OPTIONAL,
```

```
    sourceBearerType            [76] BearerType OPTIONAL,
    destBearerType              [77] BearerType OPTIONAL,
    smDefaultMsgId              [78] SmDefaultMessageId OPTIONAL,
    sourceNetworkType           [79] NetworkType OPTIONAL,
    destNetworkType             [80] NetworkType OPTIONAL,
    xsMessageType               [82] XsMessageTypeInfo OPTIONAL
}

OriginalMessageFields ::= SEQUENCE {
-- Fields here should exactly match those in MessageFields by their tag values
    originatorAddress           [2] GsmAddressInfo OPTIONAL,
    recipientAddress            [4] GsmAddressInfo OPTIONAL
}

-- Following record is generated when an notification for an AO/SM has been
-- received from an SMSC.

ReceivedNotificationWithCountryAndNetworkInfo ::= SEQUENCE {
    timestamp                   [0] IMPLICIT GeneralizedTime,
    routingAction               [1] IMPLICIT AtiRoutingAction,
    -- rejectInfo and responseInfo are mutually exclusive.
    rejectInfo                  [2] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        atRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        atExtConditionRule          [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    responseInfo                [3] IMPLICIT SEQUENCE {
        deliveryResult              [0] IMPLICIT AtDeliveryResult,
        routingErrorCode            [1] IMPLICIT ErrorCode OPTIONAL,
        atRoutingRule               [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    applicationName             [17] IMPLICIT NameString,
    applicationShortNumber      [18] IMPLICIT GsmAddress,
    messageFields               [19] IMPLICIT SEQUENCE {
        -- recipientAddress and alphanumericRecipient are mutually
        -- exclusive
        originatorAddress           [2] GsmAddressInfo,
        recipientAddress            [4] GsmAddressInfo OPTIONAL,
        moreMessagesToSend          [12] BOOLEAN,
        priority                    [13] Priority OPTIONAL,
        serviceCentreTimestamp      [21] GeneralizedTime OPTIONAL,
        deliveryStatus              [22] DeliveryStatus,
        errorCode                   [23] ErrorCode OPTIONAL,
        deliveryTimestamp           [24] GeneralizedTime OPTIONAL,
        alphanumericOriginator      [30] AlphanumericAddress OPTIONAL,
        alphanumericRecipient       [31] AlphanumericAddress OPTIONAL,
        portNumber                  [34] PortNumber OPTIONAL,
        sourcePort                  [46] PortNumber OPTIONAL,
        destinationPort             [47] PortNumber OPTIONAL,
        endToEndAckRequest          [50] EndToEndAckRequest OPTIONAL,
        endToEndMessageType         [52] EndToEndMessageType OPTIONAL,
        messageReference            [53] MessageReference OPTIONAL,
        privacy                     [54] Privacy OPTIONAL,
        numberOfMessages            [55] NumberOfMessages OPTIONAL,
        language                    [56] Language OPTIONAL,
        payloadType                 [63] PayloadType OPTIONAL,
        sourceSubAddress            [67] SubAddress OPTIONAL,
        destSubAddress              [68] SubAddress OPTIONAL,
        userResponseCode            [69] UserResponseCode OPTIONAL,
        displayTime                 [70] DisplayTime OPTIONAL,
        callbackNumbers             [71] SEQUENCE OF CallbackNumber OPTIONAL,
        msValidityIndicator         [72] MsValidityIndicator OPTIONAL,
        msValidityPeriod            [73] MsValidityPeriod OPTIONAL,
        alertOnMessageDelivery      [74] AlertOnMessageDelivery OPTIONAL,
        smsSignal                   [75] SmsSignal OPTIONAL,
```

```
        sourceBearerType            [76] BearerType OPTIONAL,
        destBearerType              [77] BearerType OPTIONAL,
        smDefaultMsgId              [78] SmDefaultMessageId OPTIONAL,
        sourceNetworkType           [79] NetworkType OPTIONAL,
        destNetworkType             [80] NetworkType OPTIONAL
    },
    outboundAt                  [23] IMPLICIT OutboundAt OPTIONAL,
    storage                     [24] IMPLICIT StorageInfo OPTIONAL,
    ecResponseData              [34] IMPLICIT SEQUENCE OF EcResponseData OPTIONAL,
    -- insideRejectInfo and insideResponseInfo are mutually exclusive.
    originalMessageFields       [46] OriginalMessageFields OPTIONAL,
    insideRejectInfo            [47] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        atiRoutingRule              [1] IMPLICIT NameString OPTIONAL,
        atiExtConditionRule         [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    insideResponseInfo          [48] IMPLICIT SEQUENCE {
        deliveryResult              [0] IMPLICIT AtDeliveryResult,
        routingErrorCode            [1] IMPLICIT ErrorCode OPTIONAL,
        atiRoutingRule              [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    ssiInfo                     [50] IMPLICIT SsiServiceInfo OPTIONAL
}


-- Following sub-record is included when an inbound operation results in an
-- outbound MT/SM.

OutboundMt ::= SEQUENCE {
    sriSmRoutingAction          [0] SriSmRoutingAction,
    sriSmRejectInfo             [1] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        mtRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        mtExtConditionRule          [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    sriSmResponseInfo           [2] IMPLICIT SEQUENCE {
        queryResult                 [0] SriSmQueryResult,
        mapImsi                     [1] ImsiInfo OPTIONAL,
        mapLmsi                     [2] Lmsi OPTIONAL,
        mapMsc                      [3] GsmAddressInfo OPTIONAL,
        mapSgsn                     [4] GsmAddressInfo OPTIONAL,
        mtRoutingRule               [5] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    mtFwdSmToMscRoutingAction   [3] MtFwdSmRoutingAction OPTIONAL,
    mtFwdSmToMscRejectInfo      [4] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        mtRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        mtExtConditionRule          [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    mtFwdSmToMscResponseInfo    [5] IMPLICIT SEQUENCE {
       deliveryResult               [0] IMPLICIT MtFwdSmDeliveryResult OPTIONAL,

       mtRoutingRule                [1] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    mtFwdSmToSgsnRoutingAction   [6] MtFwdSmRoutingAction OPTIONAL,
    mtFwdSmToSgsnRejectInfo      [7] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        mtRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        mtExtConditionRule          [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    mtFwdSmToSgsnResponseInfo    [8] IMPLICIT SEQUENCE {
       deliveryResult               [0] IMPLICIT MtFwdSmDeliveryResult OPTIONAL,

       mtRoutingRule                [1] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
```

```
    ecResponseData               [9] IMPLICIT SEQUENCE OF EcResponseData OPTIONAL,
    destinationPointCode         [10] IMPLICIT  PointCode OPTIONAL,
    isMultiPartMtMessage         [11] IMPLICIT BOOLEAN OPTIONAL
}

-- 2184
-- Following sub-record is included when an inbound operation results in an
-- outbound SIP.

OutboundSip ::= SEQUENCE {
    mtFwdSmToImsRoutingAction   [6] MtFwdSmRoutingAction OPTIONAL,
    mtFwdSmToImsRejectInfo      [7] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        mtRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        mtExtConditionRule          [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    mtFwdSmToImsResponseInfo    [8] IMPLICIT SEQUENCE {
        deliveryResult              [0] IMPLICIT MtFwdSmDeliveryResult OPTIONAL,

        mtRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        sipErrorCode                [2] IMPLICIT SipErrorCode OPTIONAL,
        rpErrorCause                [3] IMPLICIT RpErrorCause OPTIONAL,
        tpFailureCause              [4] IMPLICIT TpFailureCause OPTIONAL
    } OPTIONAL,
    ecResponseData               [9] IMPLICIT SEQUENCE OF EcResponseData OPTIONAL,
    iiw                         [60] IMPLICIT IiwId OPTIONAL,
    sipServerAddress            [61] IMPLICIT SipUrl OPTIONAL,
    sipClientName               [62] IMPLICIT NameString OPTIONAL,
    sipEndPointName             [63] IMPLICIT NameString OPTIONAL,
    sipEndPointTransport        [64] IMPLICIT SipTransport OPTIONAL
}

-- Following sub-record is included when an inbound operation results in an
-- outbound MO/SM.

OutboundMo ::= SEQUENCE {
    submissionResult                        [0] IMPLICIT MoFwdSmSubmissionResult,
    smscName                                [1] IMPLICIT NameString,
    rejectCause                             [2] IMPLICIT MoRejectCause OPTIONAL
}

-- Following sub-record is included when an inbound operation results in an
-- outbound AO/SM.
OutboundAo ::= SEQUENCE {
    submissionResult             [0] IMPLICIT AoSubmissionResult,
    applicationName              [1] IMPLICIT NameString,
    applicationShortNumber       [2] IMPLICIT GsmAddress,
    serviceCentreName            [3] IMPLICIT NameString OPTIONAL,
    scNodeName                   [4] IMPLICIT NameString OPTIONAL,
    scTerminationPointName       [5] IMPLICIT NameString OPTIONAL,
    routingErrorCode             [6] IMPLICIT ErrorCode OPTIONAL,
    serviceCentreTimestamp       [7] IMPLICIT GeneralizedTime OPTIONAL,
    smppMessageId                [8] IMPLICIT SmppMessageId OPTIONAL
}

-- Following sub-record is included when an inbound operation results in an
-- outbound AT/SM.

OutboundAt ::= SEQUENCE {
    routingAction               [0] AtRoutingAction,
    rejectInfo                  [1] IMPLICIT SEQUENCE {
        rejectCause                 [0] IMPLICIT RejectCause,
        atRoutingRule               [1] IMPLICIT NameString OPTIONAL,
        atExtConditionRule          [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
```

```
    responseInfo                [2] IMPLICIT SEQUENCE {
        deliveryResult              [0] IMPLICIT AtDeliveryResult,
        routingErrorCode            [1] IMPLICIT ErrorCode OPTIONAL,
        atRoutingRule               [2] IMPLICIT NameString OPTIONAL
    } OPTIONAL,
    applicationName             [3] IMPLICIT NameString,
    applicationShortNumber      [4] IMPLICIT GsmAddress,
    ecResponseData              [5] IMPLICIT SEQUENCE OF EcResponseData OPTIONAL
}

-- Following sub-record is included when an inbound operation results in a
-- storage of the SM.

StorageInfo ::= SEQUENCE {
    storageResult               [0] IMPLICIT StorageResult,
    routingErrorCode            [1] IMPLICIT ErrorCode OPTIONAL,
    -- applicationName and applicationShortNumber are included when the
    -- concerning SM is to be terminated in an application.
    applicationName             [2] IMPLICIT NameString OPTIONAL,
    applicationShortNumber      [3] IMPLICIT GsmAddress OPTIONAL,
    queue                       [4] IMPLICIT QueueIndex
}

-- Following is the construct for a GSM address with country/network
-- information.

GsmAddressInfo ::= SEQUENCE {
    gsmAddress  [0] IMPLICIT GsmAddress,
    -- country/network is only included when the address can be associated with
    -- one of the provisioned countries/networks.
    country      [1] IMPLICIT Country OPTIONAL,
    network      [2] IMPLICIT Network OPTIONAL
}

-- Following is the construct for an SCCP address with country/network
-- information.

SccpAddressInfo ::= SEQUENCE {
    sccpAddress  [0] IMPLICIT SccpAddress,
    -- country/network is only included when the address can be associated with
    -- one of the provisioned countries/networks.
    country      [1] IMPLICIT Country OPTIONAL,
    network      [2] IMPLICIT Network OPTIONAL
}

-- Following is the construct for an IMSI with country/network
-- information.

ImsiInfo ::= SEQUENCE {
    imsi         [0] IMPLICIT Imsi,
    -- country/network is only included when the address can be associated with
    -- one of the provisioned countries/networks.
    country      [1] IMPLICIT Country OPTIONAL,
    network      [2] IMPLICIT Network OPTIONAL
}

-- The following sub-record is included if SSI is enabled
SsiServiceInfo ::= SEQUENCE {
    originatorServices       [0] IMPLICIT SsiServiceString OPTIONAL,
    recipientServices        [1] IMPLICIT SsiServiceString OPTIONAL
}

-- ID of AMS Queue
QueueIndex ::= INTEGER (1..1000)
```

```
-- SMPP message ID
SmppMessageId ::= OCTET STRING -- -- (SIZE (1..64))

-- String used for names of applications, networks, rules, etc.
NameString ::= OCTET STRING (SIZE (1..31))

-- SCCP address.
-- For ITU-T, refer to Q.713 section 3.4 for encoding.
-- For Japanese SS7, refer to JT Q713
-- For ANSI, refer to ANSI T1.112-1996
SccpAddress ::= OCTET STRING (SIZE (0..255))

-- GSM address. Refer to GSM 03.40 version 5.8.1 release 1996 section 9.1.5
-- for encoding.
GsmAddress ::= OCTET STRING (SIZE (0..20))

-- IMSI. Refer to GSM 09.02 for encoding.
Imsi ::= OCTET STRING (SIZE (1..8))

-- LMSI. Refer to GSM 09.02 for encoding.
Lmsi ::= OCTET STRING (SIZE (4))

-- Data Coding Scheme. Refer to GSM 03.38 for encoding.
DataCodingScheme ::= INTEGER (0..255)

-- Protocol ID. Refer to GSM 03.40 version 5.8.1 release 1996 section 9.2.3.9
-- for encoding.
ProtocolId ::= INTEGER (0..255)

-- Message Reference. Refer to GSM 03.40 version 5.8.1 release 1996 section
-- 9.2.3.6 for encoding.
MessageReference ::= INTEGER (0..65535)

-- Message Number. Refer to GSM 03.40 version 5.8.1 release 1996 section
-- 9.2.3.18 for encoding.
MessageNumber ::= INTEGER (0..255)

-- Command Type. Refer to GSM 03.40 version 5.8.1 release 1996 section
-- 9.2.3.19 for encoding.
CommandType ::= INTEGER (0..255)

-- User Data Header. Refer to GSM 03.40 version 5.8.1 release 1996 section
-- 9.2.3.24 for encoding. Note that UDHL is NOT included.
UserDataHeader ::= OCTET STRING (SIZE (0..255))

-- User Data. Refer to GSM 03.40 version 5.8.1 release 1996 section
-- 9.2.3.24 for encoding. Note that 7-bit characters are unpacked to octets.
UserData ::= OCTET STRING (SIZE (0..255))

-- 7-bit characters are unpacked to octets.
UserDataUtf8 ::= OCTET STRING (SIZE (0..255))

-- Status. Refer to GSM 03.40 version 5.8.1 release 1996 section
-- 9.2.3.15 for encoding.
Status ::= INTEGER (0..255)

-- Command Data. Refer to GSM 03.40 version 5.8.1 release 1996 section
-- 9.2.3.21 for encoding.
CommandData ::= OCTET STRING (SIZE (0..255))

-- Country code according to ISO 3166 (e.g. nl for Netherlands)
Country ::= OCTET STRING (SIZE (2..2))

-- Name of mobile network as assigned during configuration of TextPass
Network ::= OCTET STRING (SIZE (1..31))
```

```
-- Priority.
Priority ::= INTEGER (0..255)

-- CIMD Tariff Class
TariffClass ::= INTEGER (0..99)

-- CIMD Service Description.
ServiceDescription ::= INTEGER (0..99)

-- Alphanumeric address
-- When an alphanumeric address contains @ as last character then RTR will replace
 @ with escape(0x1B) and @ character in log file
-- abcdefg@ will be represented as 61 f1 98 5c 36 9f 37 00 ( last character @ is
replaced by escape and @ )
-- abc@ will be represented as 61 f1 78 03 00 ( last character @ is replaced by
escape and @ )
-- abcdefg will be represented as 61 f1 98 5c 36 9f 01  ( no change in this address
 as it does not contain @ in last )
-- abc@defg will be represented as 61 f1 18 40 2e 9b cf ( no change in this address
 as it does not contain @ in last )
AlphanumericAddress ::= VisibleString (SIZE (1..20))

-- Port Number on HUB.
PortNumber ::= INTEGER (0..65535)

-- Point code (for ITU-T, range is 0-16383, for ANSI, range is 0-16777215, for
Japanese SS7, range is 0-65535)
PointCode ::= INTEGER (0..16777215)

-- MXP Error Code
ErrorCode ::= ENUMERATED
{
    errNoError(0),
    errMtTimeout(1),
    errMtAbsentSubscriber(2),
    errMtSystemFailure(3),
    errMtDataMissing(4),
    errMtUnexpectedDataValue(5),
    errMtFaciltyNotSupported(6),
    errMtUnidentifiedSubscriber(7),
    errMtIllegalSubscriber(8),
    errMtIllegalEquipment(9),
    errMtSubscriberBusyForMtSm(10),
    errMtInvalidSmeAddress(11),
    errMtEquipmentProtocolError(12),
    errMtEquipmentNotSmEquipped(13),
    errMtMemoryCapacityExceeded(14),
    errMtOtherMapError(15),
    errMtTcapAborted(16),
    errMtSccpAborted(17),
    errMtNoPagingError(18),
    errMtImsiDetachError(19),
    errMtRoamingRestrictions(20),
    errMtShortMsgType0NotSupportedError(21),
    errMtCanNotReplaceShortMsgError(22),
    errMtUnspecifiedProtocolIdError(23),
    errMtMsgClassNotSupportedError(24),
    errMtUnspecifiedDataCodingSchemeError(25),
    errMtTpduNotSupported(26),
    errMtSimStorageFullError(27),
    errMtNoSmStorageCapabilityInSimError(28),
    errMtErrorInMs(29),
    errMtSimApplToolKitbusyError(30),
    errMtSimDataDownloadError(31),
```

```
    errMtApplSpecificError(32),
    errMtEquipUnspecifiedErrorCause(33),
    errMtUeDeregistered(34),
    errMtNoResponseViaIpsmGw(35),
    errMtBlockedByMtRule(65535),
    errSrismTimeout(16777217),
    errSrismSystemFailure(16777218),
    errSrismDataMissing(16777219),
    errSrismUnexpectedDataValue(16777220),
    errSrismFacilityNotSupported(16777221),
    errSrismUnknownSubscriber(16777222),
    errSrismAbsentSubscriber(16777223),
    errSrismCallBarred(16777224),
    errSrismTeleserviceNotProvisioned(16777225),
    errSrismOtherMapError(16777226),
    errSrismTcapAborted(16777227),
    errSrismSccpAborted(16777228),
    errSrismMsDeregisteredError(16777229),
    errSrismMsPurgedError(16777230),

    errAtSystemError(33554433),
    errAtShuttingDown(33554434),
    errAtMxpFailure(33554435),
    errAtMxpTimeout(33554436),
    errAtTxFailure(33554437),
    errAtTemporaryError(33554438),
    errAtPermanentDestError(33554439),
    errAtPermanentMsgError(33554440),
    errAtAppNotAvailable(33554441),
    errAtSourceNotAvailable(33554442),
    errAtBlockedByThroughputControl(33554443),
    errAtLoginInvalid(33554444),
    errAtLoginNotAllowed(33554445),
    errAtLoginTooManySessions(33554446),
    errAtLoginNoSmscConnection(33554447),
    errAtAlreadyLoggedIn(33554448),
    errAtOperationNotAllowed(33554449),
    errAtOperationNotSupported(33554450),
    errAtResponseTimeout(33554451),
    errAtInvalidSyntax(33554452),
    errAtInvalidChecksum(33554453),
    errAtBlockedByAtRule(33554454),
    errAtRecipientError(33554455),
    errAtAppNotExisting(33554456),
    errAtInvalidMsgType(33554457),
    errAtInvalidMsg(33554458),
    errAtInvalidTimePeriod(33554459),
    errAtInvalidAddress(33554460),
    errAtMsgNotFound(33554461),
    errAtDeliveryInProgress(33554462),
    errAtTransparent(33554463),
    errAtLoginNoResources(33554464),
    errAtTxWindowFull(33554465),
    errAtMaxMessagesBuffered(33554466),
    errAtInvalidBindStatus(33554467),
    errAtNoRoutingRule(33554468),
    errAtConversionFailed(33554469),

    errAmsDeviceNotActive(50331649),
    errAmsDbError(50331650),
    errAmsStoreFull(50331651),
    errAmsQueueFull(50331652),
    errAmsRecipBufferFull(50331653),
    errAmsInvalidQueue(50331654),
    errAmsInvalidMessage(50331655),
```

```
    errAmsInvalidValidityTime(50331656),
    errAmsInvalidDeferredDelivery(50331657),
    errAmsStorageRateExceeded(50331658),
    errAmsDeliveryAttemptsExceeded(50331659),
    errRtrBlockedByThroughputControl(67108864),
    errRtrStorageFailure(67108865),
    errRtrTimeout(67108866),
    errRtrBlockedByRule(67108867),
    errRtrNoRuleMatching(67108868),
    errRtrLicenseExceeded(67108869),
    errRtrOriginatorInBlackList(67108870),
    errRtrOriginatorNotInWhiteList(67108871),
    errRtrMessageTooLong(67108872),
    errMoTimeout(83886083),
    errMoDropped(83886084),
    errMoDiscarded(83886085),
    errMoTprRejected(83886086),
    errMoSystemFailure(83886089),
    errMoDataMissing(83886090),
    errMoUnexpectedDataValue(83886091),
    errMoFaciltyNotSupported(83886092),
    errMoUnknownServiceCentre(83886093),
    errMoServiceCentreCongestion(83886094),
    errMoInvalidSmeAddress(83886095),
    errMoSubscriberNotScSubscriber(83886096),
    errMoOtherMapError(83886097),
    errMoTcapAborted(83886098),
    errMoSccpAborted(83886099),
    errHssOtherError(100663297),
    errCapScfUnavailable(117440512),
    errCapUnassignedNumber(117440513),
    errCapUnidentfiedSubscriber(117440514),
    errCapCongestion(117440515),
    errCapFacilityNotSupported(117440516),
    errCapSmTransferRejected(117440517)
}


-- Billing ID as passed with UCP or SMPP AO/SM.
BillingId ::= OCTET STRING -- -- (SIZE (1..1024))

-- Some bits of an SMS-SUBMIT PDU. Refer to GSM 03.40 version 5.8.1 release 1996
-- section 9.2.3 for details regarding the bits.
SmsSubmitServices ::= BIT STRING {
    rejectDuplicates(5),
    statusReportRequest(2),
    userDataHeaderIndication(1),
    replyPath(0)
}

-- Some bits of an SMS-COMMAND PDU. Refer to GSM 03.40 version 5.8.1 release 1996
-- section 9.2.3 for details regarding the bits.
SmsCommandServices ::= BIT STRING {
    statusReportRequest(2),
    userDataHeaderIndication(1)
}

-- Some bits of an SMS-DELIVER PDU. Refer to GSM 03.40 version 5.8.1 release 1996
-- section 9.2.3 for details regarding the bits. Note that GSM 03.40 uses inversed
-- logic for more-messages-to-send (i.e. 0=MMS, 1=no MMS). The bit in the log uses
-- postive logic ((i.e. 0=no MMS, 1=MMS).
SmsDeliverServices ::= BIT STRING {
    moreMessagesToSend(5),
    statusReportIndication(2),
    userDataHeaderIndication(1),
```

```
    replyPath(0)
}

-- Some bits of an SMS-STATUS-REPORT PDU. Refer to GSM 03.40 version 5.8.1 release
 1996
-- section 9.2.3 for details regarding the bits.  Note that GSM 03.40 uses inversed
-- logic for more-messages-to-send (i.e. 0=MMS, 1=no MMS). The bit in the log uses
-- postive logic ((i.e. 0=no MMS, 1=MMS).
StatusReportServices ::= BIT STRING {
    moreMessagesToSend(5),
    statusReportQualifier(2)
}

-- Notification types.
NotificationType ::= BIT STRING {
    deliveryNotification (0),
    nonDeliveryNotification (1),
    bufferedNotification (2)
} (SIZE (8))

IgnoredRejectCauses ::= BIT STRING {
    unknownSccpSmscAddress(9),
    unknownMapSmscAddress(10),
    conflictingSmscAddress(11),
    spoofingSccpSmscAddress(12),
    spoofingMapSmscAddress(13),
    spoofedOriginatorAddress(19)
} (SIZE (32))

-- Result of an AO/SM
AoSubmissionResult ::= ENUMERATED {
    successful(0),
    sourceError(1),
    routingError(2),
    destinationTempError(3),
    destinationPermError(4),
    messagePermError(5)
}

-- Result of an AT/SM
AtDeliveryResult ::= ENUMERATED {
    successful(0),
    sourceError(1),
    routingError(2),
    destinationTempError(3),
    destinationPermError(4),
    messagePermError(5)
}

-- Result of storing an SM.
StorageResult ::= ENUMERATED {
    successful (0),
    destinationTempError(3),
    destinationPermError(4),
    messagePermError(5),
    queueFull(6),
    databaseError(7),
    invalidQueue(8),
    invalidStoreMessage(9),
    deviceNotOperating(10),
    amsStorageFull(11),
    recipientQueueFull(12),
    invalidValidity(13),
    invalidDeferred(14),
    storageRateExceeded(15)
```

```
}

-- Result of an MO-ForwardSM operation
MoFwdSmSubmissionResult ::= ENUMERATED {
    success(0),
    timeout(1),
    dropped(2),
    discarded(3),
    rejectedByTextPass(4),
    rejectedByApplication(5),
    unknown(6),
    systemFailureError(10),
    dataMissingError(11),
    unexpectedDataValueError(12),
    facilityNotSupportedError(13),
    invalidSmeAddressError(81),
    unknownServiceCentreError(85),
    scCongestionError (86),
    subscriberNotScSubscriberError (87),
    otherErrors(99),
    sccpAborted(100),
    tcapAborted(127)
}

-- Result of an SendRoutingInfoForSM operation
SriSmQueryResult ::= ENUMERATED {
    success(0),
    timeout(1),
    systemFailureError(10),
    dataMissingError(11),
    unexpectedDataValueError(12),
    facilityNotSupportedError(13),
    unknownSubscriberError(14),
    absentSubscriberError(15),
    callBarredError(16),
    teleServiceNotProvisionedError(17),
    otherErrors(99),
    sccpAborted(100),
    msDeregisteredError(108),
    msPurgedError(109),
    fallbackToMapVersion1Requested(125),
    fallbackToMapVersion2Requested(126),
    tcapAborted(127)
}

-- Result of an MT-ForwardSM operation
MtFwdSmDeliveryResult ::= ENUMERATED {
    success(0),
    timeout(1),
    systemFailureError(10),
    dataMissingError(11),
    unexpextedDataValueError(12),
    facilityNotSupportError(13),
    absentSubscriberError(15),
    unidentifiedSubscriberError(18),
    illegalSubscriberError(19),
    illegalEquipmentError(20),
    subscriberBusyForMtSmsError(21),
    invalidSmeAddressError(81),
    equipmentProtocolError(82),
    equipmentNotSmEquippedError(83),
    memoryCapacityExceededError(84),
    otherErrors(99),
    sccpAborted(100),
    noPagingError(105),
```

```
    imsiDetachedError(106),
    roamingRestrictionsError(107),
    shortMsgType0NotSupportedError(110),
    canNotReplaceShortMsgError(111),
    unspecifiedProtocolIdError(112),
    msgClassNotSupportedError(113),
    unspecifiedDataCodingSchemeError(114),
    tpduNotSupported(115),
    simStorageFullError(116),
    noSmStorageCapabilityInSimError(117),
    errorInMs(118),
    simApplToolKitbusyError(119),
    simDataDownloadError(120),
    applSpecificError(121),
    equipUnspecifiedErrorCause(122),
    ueDeregistered(123),
    noResponseViaIpsmGw(124),
    fallbackToMapVersion1Requested(125),
    fallbackToMapVersion2Requested(126),
    tcapAborted(127)

}

-- Result of a delivery attempt as reported in a notification on an AO/SM.
DeliveryStatus ::= ENUMERATED
{
    noStatusAvailable (0),
    inProgress (1),
    validityPeriodExpired (2),
    deliveryFailed (3),
    deliverySuccessful (4),
    noResponse (5),
    lastNoResponse (6),
    cancelled (7),
    deleted (8),
    deletedByCancel (9),
    scheduled (10),
    accepted (11),
    rejected (12),
    skipped (13),
    replaced (14)
}

-- Routing action for an AO/SM
AoRoutingAction ::= ENUMERATED {
    routeToSmsc(0),
    routeToMs(2),
    routeToMsFallbackToSmsc(3),
    discardWithNak(4),
    discardWithAck(5),
    storeForDeliveryToMs(6),
    routeToMsFallbackToStorage(7),
    storeForDeliveryToApplication(8),
    routeToApplicationFallbackToStorage(9),
    storeForForwardingAsAo(10),
    routeToSmscFallbackToStorage(11)
}

-- Routing action for an outgoing AT/SM or notification on an AO/SM
AtRoutingAction ::= ENUMERATED {
    pass(0),
    blockWithTemporaryError(1),
    blockWithPermanentError(2),
    blockWithAck(3)
}
```

```
-- Routing action for an incoming AT/SM or notification on an AO/SM
AtiRoutingAction ::= ENUMERATED {
    -- need to keep supporting the old ATO actions for backwards compatibility
    pass(0),
    blockWithTemporaryError(1),
    blockWithPermanentError(2),
    blockWithAck(3),
    -- the ATI routing actions, starting at 10.
    discardWithAck(10),
    discardWithTempError(11),
    discardWithPermMessageError(12),
    discardWithPermRecipientError(13),
    routeToApplication(14),
    routeToApplicationFallbackToStorage(15),
    storeForDeliveryToApplication(16),
    routeToSmscAsAo(17),
    routeToSmscAsAoFallbackToStorage(18),
    storeForForwardingToSmscAsAo(19)
}

-- Routing action for an MO-ForwardSM operation
MoFwdSmRoutingAction ::= ENUMERATED {
    routeToSmsc(0),
    routeToApplication(1),
    routeToMs(2),
    routeToMsFallbackToSmsc(3),
    discardWithNak(4),
    discardWithAck(5),
    routeToMsFallbackToApplication(6),
    discardWithNoResponse(7),
    storeForDeliveryToMs(8),
    routeToMsFallbackToStorage(9),
    storeForDeliveryToApplication(10),
    routeToApplicationFallbackToStorage(11),
    routeToSmscAsAo(12),
    routeToMsFallbackToSmscAsAo(13),
    routeToAoSmscGroup(14),
    routeToMsFallbackToAoSmscGroup(15)
}

-- Routing action for an SendRoutingInfoForSM operation
SriSmRoutingAction ::= ENUMERATED {
    pass(0),
    blockWithTemporaryError(1),
    blockWithPermanentError(2),
    discardSilently(3),
    pretendSuccess(4),
    release(5)
}

-- Routing action for an MT-ForwardSM operation
MtFwdSmRoutingAction ::= ENUMERATED {
    pass(0),
    blockWithTemporaryError(1),
    blockWithPermanentError(2),
    discardSilently(3),
    pretendSuccess(4),
    release(5)
}

AmsMessageId ::= OCTET STRING -- -- (SIZE (5))
    -- Order should be the same as AMS_message_id_t
    -- First byte is the AMS identifier
    -- last 4 bytes is the message identifier
```

```
GsmStatusReportType ::= ENUMERATED
{
    phase1 (1),
    phase2 (2)
}

-- twisted nibble encoded, first byte contains number of valid bytes.
-- in case of an odd number of digits, a 0-nibble is used as filler.
RoutingNumber ::= OCTET STRING (SIZE (2..5))

-- Human-readable CSV containing service names
SsiServiceString ::= OCTET STRING (SIZE(0..4096))

EndToEndAckRequest ::= BIT STRING -- -- {
-- --     readAck (0),
-- --     userAck (1)
-- -- } (SIZE (8))

EndToEndMessageType ::= ENUMERATED
{
    normalMessage (0),
    userAck (1),
    readAck (2),
    deliveryReceipt (3),
    intermediateNotification (4)
}

Privacy ::= ENUMERATED
{
    notRestricted (0),
    restricted (1),
    confidential (2),
    secret (3)
}

NumberOfMessages ::= INTEGER -- -- (0..65535)
-- number of messages in a mailbox

Language ::= ENUMERATED
{
    unspecified (0),
    english (1),
    french (2),
    spanish (3),
    german (4),
    portuguese (5),
    cantonese (6),
    mandarin (7),
    kangul (8),
    bahasa (9),
    hindi (10),
    urdu (11),
    tagalog (12),
    youroba(13),
    swahili (14),
    gaelic (15),
    hebrew (16),
    nihongo (17),
    russian (18),
    arabic (19),
    dutch(20),
    italian (21),
    polish (22),
    vietnamese (23),
```

```
    greek (24),
    yiddish (25),
    thai (26),
    laotian (27),
    persian (28),
    frenchCreole (29),
    armenian (30),
    navaho (31),
    hungarian (32),
    monKhmer (33),
    gujarathi (34),
    ukrania (35),
    czech (36),
    pennsylvaniaDutch (37),
    miao (38),
    norwegian (39),
    slovak (40),
    swedish (41),
    serbian (42),
    kru (43),
    rumanian (44),
    lithuanian (45),
    finnish (46),
    punjabi (47),
    formosan (48),
    croatian (49),
    bosnian (50),
    turkish (51),
    llocano (52),
    bengali (53),
    danish (54),
    flemish (55),
    syrian (56),
    tamil (57),
    samoan (58),
    malayalam (59),
    cajun (60),
    amharic (61)
}

PayloadType ::= ENUMERATED
{
    wdp(0),
    wcmp(1)
}

SubAddress ::= OCTET STRING --  --(SIZE(2..23))
-- binary data, encoded according to ANSI-41.

UserResponseCode ::= INTEGER (0..255)

DisplayTime ::= ENUMERATED
{
    temporary(0),
    default(1),
    invoke(2)
}

DigitMode ::= ENUMERATED
{
    tbcd(0),
    ascii(1)
}

CallbackNum ::= SEQUENCE
```

```
{
    digitMode       [0] DigitMode,
    address         [1] GsmAddress
}

Presentation ::= OCTET STRING -- -- (SIZE(1))
-- binary data, encoded according to CMT-136

CallbackNumAtag ::= OCTET STRING -- -- (SIZE(0..65))
-- binary data, encoded according to CMT-136

CallbackNumber ::= SEQUENCE
{
    number          [0] CallbackNum,
    presentation    [1] Presentation OPTIONAL,
    alphaTag        [2] CallbackNumAtag OPTIONAL
}

MsValidityIndicator ::= ENUMERATED
{
    indefinitely(0),
    powerDown(1),
    regAreaChanges(2),
    displayOnly(3),
    relative(4)
}

TimeUnit ::= ENUMERATED
{
    second(0),
    minute(1),
    hour(2),
    day(3),
    week(4),
    month(5),
    year(6)
}

MsValidityPeriod ::= SEQUENCE
{
    unit        [0] TimeUnit,
    multiplier  [1] INTEGER
}

AlertOnMessageDelivery ::= ENUMERATED
{
    default(0),
    lowPriority(1),
    mediumPriority(2),
    highPriority(3)
}

SmsSignal ::= OCTET STRING --  -- (SIZE(2))
-- binary data, encoded according to CMT-136

BearerType ::= ENUMERATED
{
    unknown(0),
    sms(1),
    csd(2),
    packetData(3),
    ussd(4),
    cdpd(5),
    dataTAC(6),
    flexReflex(7),
```

```
    cellBroadcast(8)
}

SmDefaultMessageId ::= INTEGER (0..255)

NetworkType ::= ENUMERATED
{
    unknown(0),
    gsm(1),
    tdma(2),
    cdma(3),
    pdc(4),
    phs(5),
    iden(6),
    amps(7),
    pagingNetw(8)
}

XsMessageTypeInfo ::= ENUMERATED
{
    forwardedMessage (1),
    copiedMessage (2),
    forwardedToEmail(3),
    copiedToEmail(4)
}

Protocol ::= ENUMERATED
{
    ucp (0),
    cimd (1),
    smpp (2)
}

-- Internal event for diagnostics:
Event ::= INTEGER

MnpViolation ::= ENUMERATED
{
    mnpViolationNone(0),
    mnpViolationForeignCountry(1),
    mnpViolationNumberOfForeignNetwork(2),
    mnpViolationMscOfForeignNetwork(3),
    mnpViolationInvalidImsi(4),
    mnpViolationForeignImsi(5),
    mnpViolationCallBarred(6),
    mnpViolationTsvcNotProv(7),
    mnpViolationImsiNeeded(8)
}

MsgCommand ::= ENUMERATED
{
    commandInvalid(1),
    commandMqcaModify(2),
    commandMqcaQuery(3),
    commandMqcaCancel(4),
    commandMqcaAlert(5)
}

ModifyResult ::= ENUMERATED
{
    successful(0),
    messageNotFound(1),
    deliveryInProgress(2),
    notAllowed(3),
    noAmsAvailable(4),
```

```
    databaseError(5),
    invalidMessage(6)
}

CancelMode ::= ENUMERATED
{
    allByDestination (0),
    singleById       (2)
}

EcResponseData ::= SEQUENCE
{
    extConditionRule [0] IMPLICIT NameString,
    applicationName  [1] IMPLICIT NameString,
    clientIpAddress  [2] IMPLICIT INTEGER,
    evaluationResult [3] IMPLICIT BOOLEAN,
    -- all fields below should be optional.
    attributesSet    [4] IMPLICIT EcAttributeMask OPTIONAL,
    attributesReset  [5] IMPLICIT EcAttributeMask OPTIONAL,
    diameterStatus   [6] IMPLICIT DiameterStatus OPTIONAL,
    textInEvaluationResponse [7] IMPLICIT TextInEvaluationResponse OPTIONAL,
    -- 2184
    ldapError        [8] IMPLICIT INTEGER OPTIONAL,
    recepientDomain  [9] IMPLICIT RecepientDomain OPTIONAL
}

RecepientDomain ::= ENUMERATED
{
    ss7Domain (0),
    imsDomain (1),
    ss7ThenImsDomain (2),
    imsThenSs7Domain (3)
}

TextInEvaluationResponse ::= OCTET STRING (SIZE(0..255))
-- any text that the EC application needs to add to log, encoded in UTF-8.

RejectCause ::= ENUMERATED {
    invalidSccpSmscAddress(0),
    invalidMapSmscAddress(1),
    invalidRecipientAddress(2),
    invalidOriginatorAddress(3),
    invalidImsi(4),
    invalidMscAddress(5),
    forgingMscAddress(6),
    forgingImsi(7),
    forgingLmsi(8),
    unknownSccpSmscAddress(9),
    unknownMapSmscAddress(10),
    conflictingSmscAddresses(11),
    spoofingSccpSmscAddress(12),
    spoofingMapSmscAddress(13),
    unsolicitedMtForwardSm(14),
    matchingMtRoutingRule(15),
    matchingMtExternalConditionRule(16),
    matchingMoRoutingRule(17),
    matchingMoExternalConditionRule(18),
    spoofedOriginatorAddress(19),
    matchingAoRoutingRule(20),
    matchingAoExternalConditionRule(21),
    matchingAtRoutingRule(22),
    matchingAtExternalConditionRule(23),
--    matchingCdmaMoRoutingRule(24),
--    matchingCdmaMoExternalConditionRule(25),
    mnpViolation(26),
```

```
    matchingAtiRoutingRule(27),
    matchingAtiExternalConditionRule(28),
    noSpaceForSegmentingHeader(29),
    earlyRecipientSriSmFailure(30),
    originatorListViolation(31),
    messageTooLong(32),
    originatorThroughputViolation(33),
    matchingSriqRoutingRule(34),
    unidentfiedSubscriber(35),
    absentSubscriber(36),
    facilityNotSupported(37),
    deliveryFailure(38),
    systemFailure(39),
    subscriberBusyForMtSms(40),
    illegalSubscriber(41),
    illegalEquipment(42),
    matchingMtiRoutingRule(43),
    unknownSubscriber(44)
}

EcAttributeMask ::= BIT STRING (SIZE(32))
-- string of 32 boolean values, 1 means applicable, 0 means do nothing

DiameterStatus ::= INTEGER (-9..6000)
-- result code of the Diameter answer, as received by the PBC

MoRejectCause ::= ENUMERATED {
    increaseInLengthMakesTransparentRoutingImpossible(0)
}

--2184

IiwId ::= SEQUENCE
{
    hostId                      [0] INTEGER,
    portNumber                  [1] PortNumber OPTIONAL
}

SipErrorCode ::= INTEGER
RpErrorCause ::= INTEGER
TpFailureCause ::= INTEGER
DiameterShResultCode ::= INTEGER

ImsUserState ::= ENUMERATED
{
    registered(0),
    notRegistered(1),
    registeredUnregServices(2),
    authenticationPending(3)
}

SipUrl ::= VisibleString (SIZE (1..128))
TelUrl ::= VisibleString (SIZE (1..128))

-- ExpireValue. Refer to TS 29.228 Appendixes B2.2 and B2.3
-- And also refer RFC 3261 Section 20.19
ExpireValue ::=  OCTET STRING (SIZE (4))

RegistrationType ::= ENUMERATED
{
    explicitDeRegistration (0),
    failureDeRegistration   (1),
    registration (2)
}
```

```
AtmRequestResult ::= ENUMERATED
{
    success(0),
    timeout(1),
    dataMissingError(11),
    unexpectedDataValueError(12),
    unknownSubscriberError(14),
    callBarredError(16),
    teleserviceNotProvisionedError(17),
    atmNotAllowedError(41),
    bearerServiceNotProvisionedError(42),
    illegalSSOperationError(43),
    ssSubscriptionViolationError(44),
    ssErrorStatus(45),
    ssIncompatibilityError(46),
    informationNotAvailableError(47),
    otherErrors(99),
    tcapAborted(127)
}

SipTransport ::= ENUMERATED
{
    tcp (0),
    udp (1),
    sctp(2)
}

END
```

Note: The `timestamp` field of the log represents the time when the log is generated for incoming rules and outgoing rules. The log is always created when the processing of the rules (either on the incoming leg, or on the outgoing leg) are finished. For instance, if other components (database server, intermediate storage) are involved in the message processing, the additional time difference will be introduced between the time when the message arrives in the system, and the timestamp of the log. Also any network delay will increase the difference. Therefore, the timestamp of the log record is likely different from the arrival time of the message.

Note: The `ecResponseData` field directly inside the `suspectMtFwdSmWithCountryAndNetworkInfo` and `trustedMtFwdSmWithCountryAndNetworkInfo` records the MTOX rule information.

Appendix C

Event ASN.1 Data Types

Topics:

C.1 Event ASN.1 Data Types

This appendix provides the event ASN.1 data types (as defined in `event.asn1`) used to generate the RTR log records.

```
-- ------------------------------------------------------------------------
--
--   (c) Copyright 2004-2015 NewNet
--
--   This software is proprietary to and embodies the confidential technology
--   of NewNet. Possession, use, duplication or dissemination of the
--   software and media is authorized only pursuant to a valid written license
--   from NewNet.
--
-- ------------------------------------------------------------------------
-- $Id: event.asn1,v 1.51 2015/02/03 06:01:57 a3kuma10 Exp $
-- ------------------------------------------------------------------------

LOGGING DEFINITIONS IMPLICIT TAGS ::=
BEGIN

Event ::= [APPLICATION 30] SEQUENCE {
    timeStamp                  [0] GeneralizedTime,                  --(Timestamp
 for the event)
    objectId                    [1] ObjectIdentifier,
    objectType                  [2] Object
}


Object ::= CHOICE {
    rogueTcap                  [1] SpoofAttempt,                    --(Rogue TCAP
 message that cannot be matched to an existing dialog)
    shortMessage                [2] ShortMessage,
    routingInfoQuery            [3] RoutingInfoQuery

    -- add new object types here.
}

--------------------------
-- Short Message Events --
--------------------------

ShortMessage ::= CHOICE {
    -- GSM / SS7 side events
    gsmOrigInRequest              [0] EvtSmGsmOrigInRequest,
    gsmOrigInResponse             [1] EvtSmGsmOrigInResponse,
    gsmOrigOutRequest             [2] EvtSmGsmOrigOutRequest,
    gsmOrigOutResponse            [3] EvtSmGsmOrigOutResponse,
    gsmOrigRejection              [4] EvtSmGsmOrigRejection,
    gsmTermInRequest              [5] EvtSmGsmTermInRequest,
    gsmTermInResponse             [6] EvtSmGsmTermInResponse,
    gsmTermOutRequest             [7] EvtSmGsmTermOutRequest,
    gsmTermOutResponse            [8] EvtSmGsmTermOutResponse,
    gsmTermRejection              [9] EvtSmGsmTermRejection,
    gsmRoutingInfoQuery          [10] EvtSmGsmRoutingInfoQuery,

    -- application side events
    appOrigInRequest             [20] EvtSmAppOrigInRequest,
    appOrigInResponse            [21] EvtSmAppOrigInResponse,
    appOrigOutRequest            [22] EvtSmAppOrigOutRequest,
    appOrigOutResponse           [23] EvtSmAppOrigOutResponse,
```

```
    appOrigRejection              [24] EvtSmAppOrigRejection,
    appTermInRequest              [25] EvtSmAppTermInRequest,
    appTermInResponse             [26] EvtSmAppTermInResponse,
    appTermOutRequest             [27] EvtSmAppTermOutRequest,
    appTermOutResponse            [28] EvtSmAppTermOutResponse,
    appTermRejection              [29] EvtSmAppTermRejection,

    -- IMS side events
    imsOrigInRequest              [40] EvtSmImsOrigInRequest,
    imsOrigInResponse             [41] EvtSmImsOrigInResponse,
    imsOrigRejection              [44] EvtSmImsOrigRejection,
    imsTermOutRequest             [45] EvtSmImsTermOutRequest,
    imsTermOutResponse            [46] EvtSmImsTermOutResponse,

    -- AMS events
    amsStoreRequest               [60] EvtSmAmsStoreRequest,
    amsStoreResponse              [61] EvtSmAmsStoreResponse,
    amsDeliveryAttempt            [62] EvtSmAmsDeliveryAttempt,
    amsTermination                [63] EvtSmAmsTermination,
    amsDeliveryRejection          [64] EvtSmAmsDeliveryRejection,

    -- central processing events
    routingDecision               [70] EvtSmRoutingDecision,
    externalCondition             [71] EvtSmExternalCondition,
    notification                  [72] EvtSmNotification,
    smCopied                      [73] EvtSmCopied,
    copyCreated                   [74] EvtSmCopyCreated,
    copyRejection                 [75] EvtSmCopyRejection,
    forwardAttempt                [76] EvtSmForwardAttempt,
    forwardRejection              [77] EvtSmForwardRejection,
    prepaidCharging               [78] EvtSmPrepaidCharging,
    smAutoReplied                 [79] EvtSmAutoReplied,
    autoReplyCreated              [80] EvtSmAutoReplyCreated,
    signatureInserted             [81] EvtSmSignatureInserted,

    -- auxiliary events
    endpoints                     [90] EvtSmEndpoints,
    finalStatus                   [91] EvtSmFinalStatus
}

-------------------------------------
-- ShortMessage events (EvtSmXyz) --
-------------------------------------

-- Incoming GSM MO message
EvtSmGsmOrigInRequest ::= SEQUENCE
{
    originator                  [0] GsmAddress,         -- MAP sm-RP-OA
    recipient                   [1] GsmAddress,         -- GSM 03.40 TP-DA
    smsc                        [2] GsmAddress,         -- MAP sm-RP-DA
    originatingAddress          [3] SccpMscSgsnAddress, -- SCCP CgPA
    message                     [4] SmMessage,
    pid                         [5] ProtocolId,
    originatorImsi              [6] ImsiInfo OPTIONAL,
    statusReportRequest         [7] GsmStatusReportPhase OPTIONAL,
    validityExpirationTime      [8] UnixTimestamp OPTIONAL,
    deferredDeliveryTime        [9] UnixTimestamp OPTIONAL,
    originatingPointCode       [10] PointCode OPTIONAL
}

-- Response to incoming GSM MO message
EvtSmGsmOrigInResponse ::= GsmResponse

-- Outgoing GSM MO message
EvtSmGsmOrigOutRequest ::= SEQUENCE
```

```
{
    smscAddress                 [0] SccpAddressInfo,  -- SCCP CdPA
    smscName                    [1] NameString OPTIONAL
}

-- Response to outgoing GSM MO message
EvtSmGsmOrigOutResponse ::= GsmResponse

-- Reject causes for incoming GSM MO message
EvtSmGsmOrigRejection ::= CHOICE
{
    spoofingDetected            [0] EmptySequence,
    droppedDueToDecimation      [1] EmptySequence,
    fallbackUnavailable         [2] EmptySequence,
    storageUnavailable          [3] EmptySequence,
    invalidMoTag                [4] EmptySequence,
    invalidOriginatorAddress    [5] EmptySequence,
    invalidMscAddress           [6] EmptySequence,
    invalidMapSmscAddress       [7] EmptySequence,
    invalidRecipientAddress     [8] EmptySequence,
    throughputLimit             [9] ThroughputLimit,
    noCamelChargingServer      [10] EmptySequence,
    noRoutingPathAvailable     [11] EmptySequence,
    mnpViolation               [12] EmptySequence,
    segmentationImpossible     [13] EmptySequence,
    callerAborted              [14] EmptySequence

    -- more reject causes other than "routing decision"...
}

--     rejectedByApp               [2] RejectedByApp,
--     rejectedByRouter            [3] EmptySequence,

-- Incoming GSM MT Request
EvtSmGsmTermInRequest ::= SEQUENCE
{
    originator                  [0] GsmAddress,          -- GSM 03.40 TP-OA
    recipient                  [1] GsmAddress OPTIONAL, -- inherited from previous
 SRI-SM
    smsc                        [2] GsmAddress,          -- MAP sm-RP-OA
    terminatingAddress          [3] SccpMscSgsnAddress,  -- SCCP CdPA
    message                     [4] SmMessage,
    recipientImsi               [5] ImsiInfo,
    smscTimestamp               [6] UnixTimestamp,
    pid                         [7] ProtocolId
}

-- Response to incoming GSM MT Request
EvtSmGsmTermInResponse ::= GsmResponse

-- Outgoing GSM MT Request
EvtSmGsmTermOutRequest ::= SEQUENCE
{
    terminatingAddress          [0] SccpMscSgsnAddress, -- SCCP CdPA
    recipientImsi               [1] ImsiInfo,
    destinationPointCode        [2] PointCode OPTIONAL
}

-- Response to outgoing GSM MT Request
EvtSmGsmTermOutResponse ::= GsmResponse

-- Reject causes for incoming GSM MT message
EvtSmGsmTermRejection ::= CHOICE
{
    unsolicitedMessage          [1] EmptySequence,
```

```
                                                    -- 2 == deprecated disapproved,
                                                  -- see externalCondition instead.

    invalidSccpMscAddress       [3] EmptySequence,
    invalidMapSmscAddress       [4] EmptySequence,
    invalidRecipientAddress     [5] EmptySequence,
    invalidOriginatorAddress    [6] EmptySequence,
    invalidImsi                 [7] EmptySequence,
    invalidMscAddress           [8] EmptySequence,
    forgingMscAddress           [9] EmptySequence,
    forgingImsi                [10] EmptySequence,
    forgingLmsi                [11] EmptySequence,
    unknownSccpSmscAddress     [12] EmptySequence,
    unknownMapSmscAddress      [13] EmptySequence,
    conflictingSmscAddress     [14] EmptySequence,
    spoofingSccpSmscsAddress   [15] EmptySequence,
    spoofingMapSmscAddress     [16] EmptySequence

    -- more reject causes other than "routing decision"...
}

-- GSM HLR Query & Result
EvtSmGsmRoutingInfoQuery ::= SEQUENCE
{
    msisdn                      [0] GsmAddress, -- MAP msisdn
    result                      [1] GsmRoutingInfoQueryResult
}


-- Incoming AO Request
EvtSmAppOrigInRequest ::= SEQUENCE
{
    originator                  [0] GsmAddress,
    recipient                   [1] GsmAddress,
    application                 [2] EsmeApplication OPTIONAL,
    message                     [3] SmMessage,
    notificationRequested       [4] AppNotificationRequest OPTIONAL,
    validityExpirationTime      [5] UnixTimestamp OPTIONAL,
    deferredDeliveryTime        [6] UnixTimestamp OPTIONAL
}

-- Response to incoming AO Request
EvtSmAppOrigInResponse ::= GenericResponse

-- Outgoing AO Request
EvtSmAppOrigOutRequest ::= SEQUENCE
{
    smsc                        [0] SmAppServiceCentre,
    application                 [1] EsmeApplication OPTIONAL
}

-- Response to outgoing AO Request
EvtSmAppOrigOutResponse ::= GenericResponse

-- Reject causes for incoming AO message
EvtSmAppOrigRejection ::= CHOICE
{
    -- RTR range
    throughputLimit             [0] ThroughputLimit,
    noCamelChargingServer       [1] EmptySequence,
    messageTooLong              [2] EmptySequence,
    segmentationImpossible      [3] EmptySequence,
    originatorListViolation     [4] EmptySequence,
```

```
    -- HUB range
    sendMxpReqFailed          [10] EmptySequence,
    timeoutMxpReq             [11] EmptySequence,
    appNotAuthorised          [12] EmptySequence,
    appNotAllowed             [13] EmptySequence,
    systemError               [14] EmptySequence

}

-- Incoming AT Request
EvtSmAppTermInRequest ::= SEQUENCE
{
    originator                 [0] GsmAddress,
    recipient                  [1] GsmAddress,
    serviceCentre              [2] SmAppServiceCentre,
    application                [3] EsmeApplication,
    message                    [4] SmMessage
}

-- Response to incoming AT Request
EvtSmAppTermInResponse ::= GenericResponse

-- Outgoing AT Request
EvtSmAppTermOutRequest ::= SEQUENCE
{
    application                [0] EsmeApplication
}

-- Response to outgoing AT Request
EvtSmAppTermOutResponse ::= GenericResponse

-- Reject causes for incoming AT message
EvtSmAppTermRejection ::= CHOICE
{
-- RTR range
    throughputLimit            [0] ThroughputLimit,

-- HUB range
    sendMxpReqFailed          [10] EmptySequence,
    timeoutMxpReq             [11] EmptySequence,
    smscNotAllowed            [12] EmptySequence,
    systemError               [13] EmptySequence
}

-- Incoming IMS MO Request
EvtSmImsOrigInRequest ::= SEQUENCE
{
    originator                 [0] GsmAddress, -- built from URI (note) --
    recipient                  [1] GsmAddress, -- GSM 03.40 TP-DA --
    fromUri                    [2] URI,
    toUri                      [3] URI,
    message                    [4] SmMessage
}

-- (note): this number is taken from the tel: URI in the P-Asserted-Identity, --
-- and is encoded as if it were a MAP AddressString with TON = internat. (1)  --
-- and NPI = ISDN E164 (1). --

-- Response to incoming IMS MO Request
EvtSmImsOrigInResponse ::= CHOICE
{
    success                    [0] EmptySequence,
    failure                    [1] EvtSmImsOrigInFailure
-- note: a timeout on the RTR side will be encoded as an rp error TemporaryFailure
 --
```

```
}

EvtSmImsOrigInFailure ::= CHOICE
{
    sipError                    [0] INTEGER, -- see IANA assignments for SIP --
    rpCause                     [1] INTEGER  -- see Short Message RP 24.011 --
}

-- Reject Causes for incoming IMS MO message
EvtSmImsOrigRejection ::= CHOICE
{
    noRouterAvailable      [0] EmptySequence,
    mtResponseNotExpected  [1] EmptySequence,
    invalidOriginator      [2] EmptySequence,
    invalidRecipient       [3] EmptySequence,
    badSipRequest          [4] EmptySequence,
    noRoutingPathAvailable [5] EmptySequence  -- generated by RTR.
}

-- Outgoing IMS MT Request
EvtSmImsTermOutRequest ::= SEQUENCE
{
    fromUri                     [0] URI,
    toUri                       [1] URI
}

-- Response to outgoing IMS MT Request
EvtSmImsTermOutResponse ::= CHOICE
{
    success                     [0] EmptySequence,
    failure                     [1] EvtSmImsTermOutFailure,
    timeout                     [2] EmptySequence
}

EvtSmImsTermOutFailure ::= CHOICE
{
    sipError                    [0] INTEGER, -- see IANA assignments for SIP --
    rpCause                     [1] INTEGER  -- see Short Message RP 24.011 --
}

-- Attempt to store an SM in a AMS
EvtSmAmsStoreRequest ::= SEQUENCE
{
    id                          [0] INTEGER,         -- AMS - ID
    queue                       [1] INTEGER,
    bufferedSrRequested         [2] BOOLEAN,
    storeAfterDeliveryAttempt   [3] BOOLEAN
}

-- Response to store attempt
EvtSmAmsStoreResponse ::= CHOICE
{
    success                     [0] AmsStoreParameters,
    failure                     [1] AmsStoreFailure,
    timeout                     [2] EmptySequence
}

-- Attempt of the AMS to deliver a SM (possibly as AO)
EvtSmAmsDeliveryAttempt ::= SEQUENCE
{
    moreMessagesToSend          [0] BOOLEAN
}

-- Indication of AMS that SM terminated inside AMS
EvtSmAmsTermination ::= SEQUENCE
```

```
{
    reason                      [0] AmsTerminationReason
}

-- RTR-internal rejection of a AMS delivery attempt
EvtSmAmsDeliveryRejection ::= CHOICE
{
    throughputLimit             [0] ThroughputLimit,
    destinationUnavailable      [1] EmptySequence
}

-- (Rule-based) routing decision for an SM
EvtSmRoutingDecision ::= SEQUENCE
{
    action                      [0] RoutingAction,
    rule                      [1] RoutingRule OPTIONAL -- not present upon "default
 routing"
}

-- The result of an external condition evaluation.
EvtSmExternalCondition ::= SEQUENCE
{
    application                 [0] EcApplication,
    result                      [1] BOOLEAN,
    rule                        [2] RoutingRule OPTIONAL,
    signatureRequested          [3] BOOLEAN OPTIONAL
}

-- A notification or status report has been created for the SM at hand.
EvtSmNotification ::= EmptySequence

-- A copy has been created of the SM at hand
EvtSmCopied ::= SEQUENCE
{
    childId                     [0] ObjectIdentifier,
    destination                 [1] GeneralizedAddress
}

EvtSmCopyCreated ::= SEQUENCE
{
    parentId                    [0] ObjectIdentifier,
    serviceName                 [1] NameString
}

EvtSmCopyRejection ::= CHOICE
{
    nonHplmnRecipient           [0] EmptySequence,
    loopPrevented               [1] EmptySequence,
    recipientValidationFailed   [2] EmptySequence,
    storageUnavailable          [3] EmptySequence,
    emailCopyFailed             [4] EmptySequence
}

EvtSmForwardAttempt ::= SEQUENCE
{
    destination                 [0] GeneralizedAddress
}

EvtSmForwardRejection ::= CHOICE
{
    loopPrevention              [0] EmptySequence,
    simDataDownload             [1] EmptySequence,
    nonHplmnBNumber             [2] EmptySequence,
    nonHplmnCNumber             [3] EmptySequence,
    unsupportedCNumber          [4] EmptySequence,
```

```
    emailForwardingFailed       [5] EmptySequence
}

-- Prepaid charging parameters
EvtSmPrepaidCharging ::= SEQUENCE
{
    result                      [0] PrepaidChargingResult,
    chargedParty                [1] GeneralizedAddress,
    protocol                    [2] ChargingProtocol
}

-- An Auto-Reply message has been issued for this SM.
EvtSmAutoReplied ::= SEQUENCE
{
   childId                  [0] ObjectIdentifier  -- ID of the Auto-Reply message.
}

-- An Auto-Reply message has been created.
EvtSmAutoReplyCreated ::= SEQUENCE
{
    parentId                    [0] ObjectIdentifier, -- ID of original SM.
    serviceName                [1] NameString,       -- name of requesting service.

    message                     [2] SmMessage
}

-- A signature has been inserted to a (inbound) SM.
EvtSmSignatureInserted ::= SEQUENCE
{
    serviceName                 [0] NameString
}

-- Auxiliary event for the CCI to search for messages.
EvtSmEndpoints ::= SEQUENCE
{
    originator          [0] GeneralizedAddress,
    recipient          [1] GeneralizedAddress OPTIONAL -- may be missing for MT-MT
 routing
}

EvtSmFinalStatus ::= SEQUENCE
{
    finalStatusCode             [0] FinalStatusCode
}

--------------------------------------------------
-- Shared ShortMessage - specific types (SmXyz) --
--------------------------------------------------

SmMessage ::= SEQUENCE
{
    data                        [0] SmData OPTIONAL,  -- user data without header
    concatenationInfo           [1] ConcatInfo OPTIONAL,
    userDataHeaderInfo          [2] UserDataHeaderInfo OPTIONAL
}

SmData ::= SEQUENCE
{
    length                      [0] INTEGER,
   content                   [1] SmContent OPTIONAL -- presence depends on license
}

SmContent ::= CHOICE
{
    text                        [0] SmText,           -- if readable text.
```

```
    data                        [1] OCTET STRING      -- if binary data.
}

SmText ::= VisibleString (SIZE (0..1024))             -- UTF-8 encoded.

SmGsmMapError ::= ENUMERATED
{
    other(0),
    systemFailure(1),
    dataMissing(2),
    unexpectedDataValue(3),
    facilityNotSupported(4),
    unknownSubscriber(5),
    absentSubscriber(6),
    callBarred(7),
    teleServiceNotProvisioned(8),
    unidentifiedSubscriber(9),
    illegalSubscriber(10),
    illegalEquipment(11),
    subscriberBusyForMtSms(12),
    invalidSmeAddress(13),
    equipmentProtocol(14),
    equipmentNotSmEquipment(15),
    memoryCapacityExceeded(16),
    unknownServiceCentre(17),
    scCongestion(18),
    subscriberNotScSubscriber(19),
    msgWaitList(20),
    noPagingError(21),                    -- BG2318: Separate Absent Subscriber
errors/ Add new MAP errors
    imsiDetachedError(22),
    roamingRestrictionsError(23),
    msDeregisteredError(24),
    msPurgedError(25),
    shortMsgType0NotSupportedError(26),
    canNotReplaceShortMsgError(27),
    unspecifiedProtocolIdError(28),
    msgClassNotSupportedError(29),
    unspecifiedDataCodingSchemeError(30),
    tpduNotSupported(31),
    simStorageFullError(32),
    noSmStorageCapabilityInSimError(33),
    errorInMs(34),
    simApplToolKitbusyError(35),
    simDataDownloadError(36),
    applSpecificError(37),
    equipUnspecifiedErrorCause(38),
    ueDeregistered(39),
    noResponseViaIpsmGw(40)
}

SmAppServiceCentre ::= SEQUENCE
{
    smscName                    [0] NameString,
    nodeName                    [1] NameString,
    terminationPointName        [2] NameString
}

--------------------------------
-- Routing Info Query Events  --
--------------------------------

RoutingInfoQuery ::= CHOICE
{
    -- GSM / SS7 side events (HLR)
```

```
    gsmInRequest                [0] EvtRiqGsmInRequest,
    gsmInResponse               [1] EvtRiqGsmInResponse,
    gsmOutRequest               [2] EvtRiqGsmOutRequest,
    gsmOutResponse              [3] EvtRiqGsmOutResponse,

    -- IMS side (HSS)

    imsOutRequest               [4] EvtHssUserDataRequest, -- UDR --
    imsOutResponse              [5] EvtHssUserDataAnswer  -- UDA --
}


-------------------------------
-- HSS Query Events          --
-------------------------------

EvtHssUserDataRequest ::= SEQUENCE
{
    publicId                    [0] URI,
    imsUserState               [1] BOOLEAN, -- whether query involves imsUserState
 --
    scscfName                  [2] BOOLEAN  -- whether query involves scscfName --
}

EvtHssUserDataAnswer ::= CHOICE
{
    success                     [0] HssUserDataQuerySuccess,
    failure                     [1] HssUserDataQueryFailure,
    timeout                     [2] NULL
}

HssUserDataQuerySuccess ::= SEQUENCE
{
    -- maybe it will be useful to log the publicId again here as [0]? --
    imsUserState                [1] ImsUserState OPTIONAL,
    scscfName                   [2] URI OPTIONAL
}

HssUserDataQueryFailure ::= EmptySequence -- to be understood and defined --

ImsUserState ::= ENUMERATED
{
    notRegistered(0),
    registered(1),
    registeredUnregServices(2),
    authenticationPending(3)

}

-----------------------------------------
-- Routing Info Query events (EvtRiqXyz)  --
-----------------------------------------

-- Incoming GSM query (SRI-SM)
EvtRiqGsmInRequest ::= SEQUENCE
{
    msisdn                      [0] GsmAddress
}

-- Response to incoming GSM query
EvtRiqGsmInResponse ::= GsmRoutingInfoQueryResult

-- Outgoing GSM query (SRI-SM)
EvtRiqGsmOutRequest ::= EmptySequence
```

```
-- Response to outgoing GSM query
EvtRiqGsmOutResponse ::= GsmRoutingInfoQueryResult

-----------------------------------------
-- (Shared) GSM - specific types (GsmXyz) --
-----------------------------------------

GsmRoutingInfoQueryResult ::= CHOICE
{
    success                     [0] GsmRoutingInfoQuerySuccess,
    failure                     [1] GsmFailure,
    timeout                     [2] EmptySequence
}

GsmRoutingInfoQuerySuccess ::= SEQUENCE
{
    msc                         [0] GsmAddress OPTIONAL,
    sgsn                        [1] GsmAddress OPTIONAL,
    imsi                        [2] ImsiInfo
}

GsmResponse ::= CHOICE
{
    success                     [0] EmptySequence,
    failure                     [1] GsmFailure,
    timeout                     [2] EmptySequence
}

GsmFailure ::= SEQUENCE
{
    reason                      [0] GsmFailureReason
}

GsmFailureReason ::= CHOICE
{
    sccpAborted                 [0] NULL,
    tcapAborted                 [1] NULL,
    mapError                    [2] SmGsmMapError
}

-- this is a MAP AddressString (29.002) OR --
-- a BCD Called Party Address (24.008), from RP (24.011), OR --
-- a TP-Address (23.040) --
--
-- The more common TypeOfNumber values and Numbering Plan values   --
-- map correctly between these standards. Some other values don't. --

GsmAddress ::= SEQUENCE {
    ton                         [0] TypeOfNumber,
    npi                         [1] NumberingPlan,
    address                     [2] Address,
    country                     [3] Country OPTIONAL,
    network                     [4] Network OPTIONAL
}

-----------------------------------------
-- (Shared) IMS - specific types (ImsXyz) --
-----------------------------------------

URI ::= UTF8String

-----------------------------------------
-- (Shared) AMS - specific types (AmsXyz) --
-----------------------------------------
```

```
AmsStoreParameters ::= SEQUENCE
{
    messageIdentifier           [0] AmsMessageId,
    amsNodeAddress              [1] AmsNodeAddress,
    validityExpirationTime      [2] UnixTimestamp OPTIONAL,
    serviceCentreTimestamp      [3] UnixTimestamp OPTIONAL
}

AmsStoreFailure ::= CHOICE
{
    temporaryError                  [0] MxipErrorCode,
    permanentErrorOnThisMessage     [1] MxipErrorCode,
    permanentErrorOnThisRecipient   [2] MxipErrorCode
}

AmsTerminationReason ::= ENUMERATED
{
    expired(0),
    deleted(1),
    replaced(2)
}

AmsNodeAddress ::= CHOICE
{
    ipv4                        [0] Ipv4Address
}

-----------------------------------------
-- Shared SCCP - specific types (SccpXyz) --
-----------------------------------------

SccpMscSgsnAddress ::= CHOICE
{
    msc                         [0] SccpAddressInfo,
    sgsn                        [1] SccpAddressInfo
}

--------------------------
-- miscellaneous  types --
--------------------------

GenericResponse ::= CHOICE
{
    success                     [0] EmptySequence,
    failure                     [1] EmptySequence,
    timeout                     [2] EmptySequence
}

NameString ::= OCTET STRING (SIZE (1..31))

UnixTimestamp ::= INTEGER(0..1000000000000)            -- A large value forces the
 type of long long

RoutingRule ::= SEQUENCE
{
    name                        [0] NameString,
    type                        [1] RuleType
}

RuleType ::= ENUMERATED
{
    mo(0),
    mt(1),
    ao(2),
    ato(3),
```

```
    ati(4),
    unsupported(5),
    mti(6),
    igm(7)
}

ThroughputLimit ::= SEQUENCE
{
    cause                       [0] ThroughputLimitCause
}

ThroughputLimitCause ::= ENUMERATED
{
    license(0),
    rule(1),
    application(2),
    applicationGroup(3),
    serviceClass(4),
    serviceCenter(5)
}

MxipErrorCode ::= INTEGER

GeneralizedAddress ::= SEQUENCE {
    address                     [0] Address,
    country                     [1] Country OPTIONAL,
    network                     [2] Network OPTIONAL
}

Address ::= VisibleString (SIZE (0..38))             -- can be numeric or
alphanumeric (UTF-8)

EsmeApplication ::= SEQUENCE {
    shortNumber                 [0] ShortNumber,
    name                        [1] NameString
}

ShortNumber ::= VisibleString (SIZE (1..38))

ConcatInfo ::= SEQUENCE {
    totalSeg                    [0] INTEGER(0..255),
    currentSeg                  [1] INTEGER(0..255),
    ref                         [2] INTEGER(0..65535)
}

-- comma-separated UTF-8 string of user data header information
-- element names. Upon overflow, the last character is '%'.
UserDataHeaderInfo ::= VisibleString (SIZE (0..64))

GsmStatusReportPhase ::= INTEGER(1..2)

AppNotificationRequest ::= BIT STRING {
    delivery(0),
    nonDelivery(1),
    buffered(2)
}

ObjectIdentifier ::= OCTET STRING (SIZE (1..32))

AmsMessageId ::= OCTET STRING(SIZE(5))

RoutingAction ::= ENUMERATED {
    blockPermanent(0),
    blockTemporary(1),
    blockWithAck(2),
```

```
    blockWithNoResponse(3),
    discardWithAck(4),
    discardWithNak(5),
    discardWithNoResponse(6),
    discardWithPermError(7),
    discardWithPermRecipientError(8),
    discardWithTempError(9),
    pass(10),
    release(11),
    storeForApp(12),
    storeForForwardingAsAo(13),
    storeForMobileStation(14),
    toApp(15),
    toAppFallbackStore(16),
    toMobileStation(17),
    toMobileStationFallbackApp(18),
    toMobileStationFallbackSmsc(19),
    toMobileStationFallbackSmscAsAo(20),
    toMobileStationFallbackStore(21),
    toSmsc(22),
    toSmscAsAo(23),
    toSmscFallbackStore(24),
    toSmscAsAoFallbackStore(25)
}

ImsiInfo ::= SEQUENCE {
    imsi                        [0] Imsi,
    country                     [1] Country OPTIONAL,
    network                     [2] Network OPTIONAL
}

Imsi        ::= OCTET STRING (SIZE (1..20))

SpoofAttempt ::= CHOICE {
    anchor                      [0] SpoofAttemptAnchorParams
}

TcapHeader ::= SEQUENCE {
    tcapMessageType             [1] TcapTag,                            --(Type of
 Message)
   tcapOrigTransId            [2] TcapTransId OPTIONAL,             --(Originator
 Transaction ID)
   tcapDestTransId            [3] TcapTransId,                      --(Destination
 Transaction ID)
    protoVersionTag             [4] ProtocolVersion OPTIONAL,           --(Dialog
version number)
    dialogTag                   [5] TcapTag OPTIONAL,                   --(Type of
 Dialog)
   appContext                [6] ApplicationContext OPTIONAL        --(Application
 Context Name in text)
}

SpoofAttemptAnchorParams ::= SEQUENCE {
    mtp                         [0] MtpHead,
    sccp                        [1] SccpHead,
    tcap                        [2] TcapHeader
}

MtpHead ::= SEQUENCE {
    mtp3OrigPointCode           [0] PointCode                           --(MTP3
originating point code)
}

PointCode ::= INTEGER (0..16777215)
```

```
SccpHead ::= SEQUENCE {
    cgPa                        [0] SccpAddressInfo,                    --(SCCP
calling party address)
    cdPa                        [1] SccpAddressInfo                     --(SCCP
called party address)
}

TcapTag ::= ENUMERATED { unknown(0), dialoguerequest(96), dialogueresponse(97),
begin(98), end(100), continue(101), abort(103)}

TcapTransId ::= OCTET STRING (SIZE (0..4))

ApplicationContext ::= VisibleString (SIZE (0..1024))

SccpAddressDigits ::= VisibleString (SIZE (0..20))

GlobalTitle ::= SEQUENCE {
    np                          [0] NumberingPlan OPTIONAL,
    number                      [1] SccpAddressDigits OPTIONAL,
    nai                         [2] NatureOfAddress OPTIONAL
}

SccpAddress ::= SEQUENCE {
    pointCode                   [0] PointCode OPTIONAL,
    subSystemNumber             [1] SubSystemNumber OPTIONAL,
    globalTitle                 [2] GlobalTitle
}

SccpAddressInfo ::= SEQUENCE {
    sccpAddress                 [0] SccpAddress,
    country                     [1] Country OPTIONAL,
    network                     [2] Network OPTIONAL
}

-- Numbering Plan and Type of Number (Nature of Address) are in a format --
-- suitable for the GsmAddress type. See GsmAddress. --
-- Note that the missing values in this enumeration correspond to --
-- conflicting values between the address formats, or spare/reserved. --

NumberingPlan ::= ENUMERATED {
    unknown(0),
    isdnTelephony(1), -- E.164 numbering plan --

    data(3), -- X.121 --
    telex(4), -- F.69 --

    national(8),
    private(9)
}

TypeOfNumber ::= ENUMERATED {
    unknown(0),
    international(1),
    national(2),
    networkSpecific(3),
    -- 4 is subscriber nr OR dedicated access, short code --
    alphaNumeric(5) -- (note) --
}

-- (note): alphaNumeric is 23.040 only, but is important --
-- enough to earn its enum value. It is 'reserved' in --
-- both 29.002 and 24.008. --

ProtocolVersion ::= INTEGER (0..255)
```

```
SubSystemNumber ::= INTEGER (0..255)

-- This is NatureOfAddress as used in SCCP Global Titles. --
-- Beware, it is NOT the 29.002 AddressString Nature of Address! --

NatureOfAddress ::= ENUMERATED
{
    unknown(0),
    subscriberNumber(1),
    reservedForNationalUse(2),
    nationalSignificantNumber(3),
    internationalNumber(4)
}

Country ::= VisibleString (SIZE (2..2))

Network ::= VisibleString (SIZE (1..31))

ProtocolId ::= INTEGER (0..255)

EmptySequence ::= SEQUENCE { }

Ipv4Address ::= VisibleString -- -- (SIZE (1..16))

EcApplication ::= SEQUENCE
{
    name                        [0] NameString
}

FinalStatusCode ::= ENUMERATED
{
    delivered(0),
    forwarded(1),
    rejected(2),        -- (nak)
    deleted(3),         -- accepted but not delivered/forwarded
    dropped(4)          -- no response
}

PrepaidChargingResult ::= ENUMERATED
{
    success(0),
    subscriberUnknown(1),
    notEnoughCredit(2),
    chargingServerUnavailable(3),
    chargingServerNotResponding(4),
    otherError(5)
}

ChargingProtocol ::= CHOICE
{
    camel       [0] EmptySequence,
    diameter    [1] EmptySequence,
    smppPlus    [2] EmptySequence
}

END
```

Appendix D

Log Record Reject Causes

Topics:

D.1 Log Record Reject Causes

The log records contain reject causes that indicate the type of error that occurred. This table describes the reject causes.

Reject Cause	Explanation
invalidSccpSmscAddress (0)	• The SCCP address of the SMSC contained an invalid digit (not 0-9) or was longer than 15 digits. • A message from a trusted SMSC was send via the suspect path. • An MT or SRI-SM message from a suspect SMSC was not in international format. • An MT or SRI-SM message from a suspect SMSC was not routed on global title (GT).
invalidMapSmscAddress (1)	The service center address in the MAP layer contained an invalid digit (not 0-9) or was longer than 20 digits.
invalidRecipientAddress (2)	The recipient address contained an invalid digit (not 0-9) or was longer than 20 digits.
invalidOriginatorAddress (3)	The originator address contained an invalid digit (not 0-9) or was longer than 20 digits.
invalidImsi (4)	The IMSI was too long or contained an invalid digit (not 0-9).
invalidMscAddress (5)	• The MSC or SGSN address was longer than 15 digits or contained an invalid digit (not 0-9). • The MSC address could not be found (in the case of a suspect MtForwardSm). • The SGSN address could not be found (in the case of a suspect MtForwardSm).
forgingMscAddress (6)	The SMSC was addressing a suspect MtForwardSm to the virtual point code of the FWL (this should not occur, as the SRI-SM should direct the message to the GT of the FWL).
forgingImsi (7)	A suspect MtForwardSm was received with an IMSI that was not given out before or was past the timeout value.
forgingLmsi (8)	A suspect MtForwardSm was received with a LMSI that was not given out before or was past the timeout value.
unknownSccpSmscAddress (9)	A suspect MtForwardSm was received with a SCCP address that did not match any network number ranges or network prefixes, or did not match any configured country.
unknownMapSmscAddress (10)	A suspect MtForwardSm was received with a MAP layer service center address that did not match any network number ranges or network prefixes, or did not match any configured country.

Reject Cause	Explanation
conflictingSmscAddresses (11)	A suspect MtForwardSm was received in which the MAP service center address did not correspond to the SCCP calling party (CGPA) address.
spoofingSccpSmscAddress (12)	A spoofed suspect MtForwardSm was detected (spoofing was done on the SCCP address).
spoofingMapSmscAddress (13)	A spoofed suspect MtForwardSm was detected (spoofing was done on the MAP service center address).
unsolicitedMtForwardSm (14)	A suspect MtForwardSm was received without a corresponding SRI-SM. The IMSI is from the operator itself, but is outside the range allocated for IMSIs by the FWL.
matchingMtRoutingRule (15)	An MtForwardSm was rejected because of an MTOR rule.
matchingMtExternalConditionRule (16)	An MtForwardSm was rejected because of an MTOX rule.
matchingMoRoutingRule (17)	An MoForwardSm was rejected because of an MOR rule.
matchingMoExternalConditionRule (18)	An MoForwardSm was rejected because of an MOX rule.
spoofedOriginatorAddress (19)	A spoofed MoForwardSm was detected (spoofing was done on the originator address).
matchingAoRoutingRule (20)	An AO message was rejected because of an AOR rule. If the log record does not indicate a rule name, this means that the message was rejected because it did not match any AOR rule.
matchingAoExternalConditionRule (21)	An AO message was rejected because of an AOX rule.
matchingAtRoutingRule (22)	An AT message was rejected because of an ATOR rule.
matchingAtExternalConditionRule (23)	An AT message was rejected because of an ATOX rule.
mnpViolation (26)	An MO message was rejected because of an MNP violation.
matchingAtiRoutingRule (27)	An AT message was rejected because of an ATIR rule.
matchingAtiExternalConditionRule (28)	An AT message was rejected because of an ATIX rule.
noSpaceForSegmentingHeader (29)	According to specifications , a message was rejected because there was no space for a segmenting header. If an AO message, submitted via SMPP, contains `sar` fields and the "total messages" value is greater than 1, then the message is a segment of a sequence of concatenated messages. To forward such an AO message through an MT path, requires the sar fields be converted to a user data header. This requires the RTR to prefix six or seven bytes to each segment. There may exist scenarios where such prefixing will result in an individual segment greater than

Reject Cause	Explanation
	140 bytes. In this case the message is undeliverable by the MT path and the following line will be logged to syslog: `Application <xxx> of ID <xx> sent unexpected data` Applications are required to explicitly divide long messages into segments of 133 bytes or less, allowing just enough space for the user data header.
earlyRecipientSriSmFailure(30)	A message was rejected because Early recipient SendRoutingInfoForSm failed with a Permanent Error.
originatorListViolation(31)	A message was rejected because the originator of the message AO/SM fails to comply with the configured originator list. If the originator white list is configured, the originator should be part of this list. If the originator black list is configured, the originator should NOT be part of this list.
messageTooLong(32)	The message was rejected because the payload of the message was long and it was configured to reject the message if the payload of the AO/SM is long for a single MT/SM.
originatorThroughputViolation(33)	A message was rejected because throughput of the originator was exceeding against any one of the following controlling instance • the originating application • the originating application group • the service class
matchingSriqRoutingRule(34)	A SendRoutingInfoForSm was rejected because of an SRIQ rule.
unidentfiedSubscriber(35)	A message was rejected because the subscriber is not registered in the PLMN (i.e. mobile subscriber is no longer being served by the MSC or SGSN address that was returned in the MAP-SRI response).
absentSubscriber(36)	A message was rejected because the subscriber was not available. The reason of the subscriber unavailable can be IMSI was detached, no paging error or roaming restricted etc.
facilityNotSupported(37)	A message was rejected because the terminating network has no SMS support in the network.
deliveryFailure(38)	A message was rejected because a) the destination MS had no memory capacity available to store the message b) protocol error occurred on the destination MS equipment c) the destination MS equipment was not equipped to handle the message.

Reject Cause	Explanation
systemFailure(39)	A message was rejected because of a problem in another entity. The type of entity or network resource may be indicated by a network resource parameter.
subscriberBusyForMtSms(40)	A message was rejected because congestion was encountered at the visited MSC.
illegalSubscriber(41)	A message was rejected because the destination MS failed authentication.
illegalEquipment(42)	A message was rejected because the IMEI of the destination MS was blacklisted in the EIR.
matchingMtiRoutingRule(43)	A MT message was rejected because of a MTIR rule.
unknownSubscriber(44)	A message was rejected because there is no directory number for the mobile subscriber (GSM 09.02).

D.2 Log Record Ignored Reject Causes

A reject cause is something that may happen during the processing of a message which may or may not cause the processed message to be rejected.

If that *reject cause* does not actually lead to the rejection of the message, but is ignored (due to the configuration), it is called an *ignored reject cause*. You may want to ignore a reject cause because you want to be graceful and keep processing the message anyway.

The corresponding configuration parameters are:

- `firewallmtactionforunknownsccpaddress`
- `firewallmtactionforunknownmapaddress`
- `firewallmtactionforconflictingaddress`
- `firewallmtactionforsccpsmscaddressspoofing`
- `firewallmtactionformapsmscaddressspoofing`

The reject cause is *ignored* whenever the parameter is set to `"pass"` rather than any of the block-actions, but the ignored reject cause will be flagged `(1)` in the `IgnoredRejectCauses` bitmask in the log record. If the reject cause has not occurred, the flag is not raised and the bit is not set (0).

The log records contain the following ignored reject causes:

- `unknownSccpSmscAddress (9)`
- `unknownMapSmscAddress (10)`
- `conflictingSmscAddress (11)`
- `spoofingSccpSmscAddress (12)`
- `spoofingMapSmscAddress (13)`
- `spoofedOriginatorAddress (19)`

Appendix E

Point Code Fields in Log Records

Topics:

E.1 Originating Point Code

`originatingPointCode` field is present in following transaction log records:

- `TrustedMoFwdSm`
- `SuspectMoFwdSm`
- `TrustedMoFwdSmWithCountryAndNetworkInfo`
- `SuspectMoFwdSmWithCountryAndNetworkInfo`

`originatingPointCode` field is present in following event log record:

- `gsmOrigInRequest`

Originating Point Code field includes the SCCP Point Code value of the inbound MO message, if the CgPA contains a PC; this is true irrespective of whether the RI is set to PC/SSN or GT. Otherwise, if the CgPA does not contain a PC then Originating Point Code field includes the MTP3 (or M3UA) OPC of the inbound MO message, if the RI is set to PC/SSN; however, if the RI is set to GT and CgPA does not contain a PC, then Originating Point Code field is set to NULL.

E.2 Destination Point Code

`destinationPointCode` field is present in following transaction log sub-records:

- `outboundMt`

`destinationPointCode` field is present in following event log record:

- `gsmTermOutRequest`

Destination Point Code field includes the SCCP Point Code value of the outbound MT message, if the CdPA contains a PC; this is true irrespective of whether the RI is set to PC/SSN or GT. Otherwise, if the CdPA does not contain a PC then Destination Point Code field includes the MTP3 (or M3UA) DPC of the outbound MT message, if the RI is set to PC/SSN; however, if the RI is set to GT and CdPA does not contain a PC, then Destination Point Code field is set to NULL.

Appendix F

Sample Configuration File

Topics:

F.1 Sample Common Configuration File

```
<tpconfig

    lgpdbdatabase="lgp"
    lgpdbuser="LGPuser"
    lgpdbpassword="LGPuser123"

    lgpnodename="lgp1"
    lgpstaticweight="1"
    lgploadertimeoutinterval="120"
    lgpaliveinterval="15"
    lgppollstart="0"
    lgppollstop="0"
    lgppollperiod="15"
    lgplocallogdir="/var/TextPass/LGP/"
    lgpdbport="0"
    lgpdbcleanolderthan="240"
    lgpdbclean="0200"
    lgpmaxmastercollectorwait="300"
    lgpstartupwait="120"
    lgprecycleinterval="120"
    lgpmaxloadtime="60"
    lgpmaxmasterrun="1800"
    lgptransclean="1"
    lgpdoneclean="0"
    lgpmaxcollectorrun="1800"
    lgpasndisablefields="trustedSriSm"
    lgploadedrestartwaitinterval="60"
    lgpalertdelay="15"
    lgpmaxnumberof="10"
    lgpquerymaxwait="20"
    lgpquerybacklog="10"
    lgpquerygeneratesqllogs="false"
    lgpquerysqllogspath="/var/TextPass/LGP/sql/sql.log"
    lgpfiletransferprotocol="sftp"
    lgpcompresssftpdata="false"


  >

    <!-- Add for each RTR from which log records to be retrieved:  -->
    <lgprouternode
        name="rtr1"
        dir="/var/TextPass/log/available"
        user="textpass"
        password="TextPass"
        pattern=".*\.dat"
    />

    <lgpdbkey
        dbkey="rec_time, inboundMessageType"
    />
    <lgpdbkey
        dbkey="mapMsisdn_gsmAddress_number, rec_time"
    />
    <lgpdbkey
        dbkey="smsDeliver_smsScTimestamp"
    />
    <lgpdbkey
        dbkey="correlatedSriSm_sccpCgPa_sccpAddress_globalTitle_number, rec_time"
```

```
    />
    <lgpdbkey
        dbkey="sccpCgPa_sccpAddress_globalTitle_number, rec_time"
    />
    <lgpdbkey
        dbkey="sccpCdPa_sccpAddress_globalTitle_number, rec_time"
    />
    <lgpdbkey
        dbkey="rejectInfo_rejectCause"
    />
    <lgpprop
        normalisationmap="0oO&#246;&#214;&#10;1iIlL!\/&#10;2zZ&#10;3eE&
        #10;4aA&#228;&#196;&#10;5sS$&#223;&#10;6&#10;7tT&#10;8bB&#10;9gG&#10;
        cC&#10;dD&#10;fF&#10;hH&#10;jJ&#10;kK&#10;mM&#10;nN&#10;pP&#10;qQ&#10;
        rR&#10;uU&#252;&#220;&#10;vV&#10;wW&#10;xX&#10;yY"
       stopwords="a,about,an,are,as,at,be,by,com,de,en,for,from,how,i,in,is,it,la,

        of,on,or,that,the,this,to,was,what,when,where,who,will,with,und,the,www"
        wordboundaries="~`!@#$%^&amp;*()+-_=[]{}\\|:&quot;;'&lt;&gt;?,./"
    />

</tpconfig>
```

F.2 Sample Host-Specific Configuration File

```
<tpconfig
    ipaddress="127.0.0.1"
    runtextpassprocess="false"
    runtpfclientprocess="true"
    runtextlgpprocess="true"
    lgpfulltextsearch=”true”
    lgpnodename="lgp1"
    >


    <fxferfile
       localpath="/usr/TextPass/etc/common_config.txt"
       serverpath="/usr/TextPass/etc/common_config.txt"
       validate="/usr/TextPass/bin/tp_config --validatecommonconfig SERVERFILE"
    />
    <fxferfile
        localpath="/usr/TextPass/etc/MGRdata.xml.gz"
        serverpath="/usr/TextPass/etc/MGRdata.xml.127.0.0.1.gz"
    />

    <trapreceiver ipaddress="127.0.0.1" udpport="11173"/>

</tpconfig>
```

Appendix G

References

Topics:

G.1 References

1. 3GPP TS 29.002 version 3.20 Release 1999; Digital cellular telecommunications systems (Phase2+); Mobile Application Part (MAP) specification
2. 3GPP TS 23.066 version 4.0.0 Release 4; Digital cellular telecommunications systems (Phase2+) (GSM); Universal Mobile Telecommunications System (UMTS); Support of Mobile Number Portability (MNP); Technical Realization; Stage 2
3. The Simple Network Management Protocol IETF - RFC 1157
4. NewNet Mobile Messaging MGR Operator Manual
5. NewNet Mobile Messaging RTR Operator Manual

Glossary

#

3GPP	3rd Generation Partnership Project

A

AMS	Active Message Store Provides store-and-forward functionality for SMS messages.
AO	Application Originated Short message traffic that is originated by an application.
AOR	Application-Originated Routing Routing rule that operates on application-originated (AO) messages. Address of Record
AOX	Application-Originated eXternal condition External condition rule that operates on application-originated (AO) messages.
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
AT	Application Terminated Short message traffic that terminates at an application.

A

ATIR	Incoming application-terminated routing Routing rule that operates on incoming application-terminated (AT) messages.
ATIX	Incoming application-terminated eXternal condition External condition rule that operates on incoming application-originated (AO) messages.
ATOR	Outgoing application-terminated routing Routing rule that operates on outgoing application-terminated (AT) messages.
ATOX	Outgoing application-terminated eXternal condition External condition rule that operates on outgoing application-originated (AO) messages.

C

CCI	Customer Care Interface A Web-based interface that allows customer care agents to assist SMS subscribers.
CDMA	Code Division Multiple Access A channel access method used by radio communication technologies. CDMA employs spread-spectrum technology and a special coding scheme (where each transmitter is assigned a code) to allow multiple users to be multiplexed over the same physical channel. CDMA, the most common cellular wireless technology deployed in North

C

America, is being replaced by GSM. See also GSM.

CPU — Central Processing Unit

CSV — Comma-separated values

The comma-separated value file format is a delimited data format that has fields separated by the comma character and records separated by newlines (a newline is a special character or sequence of characters signifying the end of a line of text).

D

DB — Database

Daughter Board

Documentation Bulletin

F

FTP — File Transfer Protocol

A client-server protocol that allows a user on one computer to transfer files to and from another computer over a TCP/IP network.

FWL — Firewall

Helps protect subscribers from receiving unwanted messages and provides statistical information and message details about inbound suspect messages.

G

GSM — Global System for Mobile Communications

GT — Global Title Routing Indicator

G

GUI	Graphical User Interface The term given to that set of items and facilities which provide the user with a graphic means for manipulating screen data rather than being limited to character based commands.

I

IETF	Internet Engineering Task Force
IMSI	International Mobile Subscriber Identity
IP	Internet Protocol IP specifies the format of packets, also called datagrams, and the addressing scheme. The network layer for the TCP/IP protocol suite widely used on Ethernet networks, defined in STD 5, RFC 791. IP is a connectionless, best-effort packet switching protocol. It provides packet routing, fragmentation and re-assembly through the data link layer.

L

LAN	Local Area Network A private data network in which serial transmission is used for direct data communication among data stations located in the same proximate location. LAN uses coax cable, twisted pair, or multimode fiber. See also STP LAN.
LGP	Log Processor Collects and processes data for the Log Viewer to display.

L

LGV	Log Viewer Logs information about NewNet Mobile Messaging operations and displays it in the Manager.

M

MAP	Mobile Application Part
MGR	A Web-based interface for managing NewNet Mobile Messaging components. Prior to Suite 6, the Configuration Manager (CM) provided this functionality.
MIB	Management Information Database
MNP	Mobile Number Portability
MO	Mobile Originated Refers to a connection established by a mobile communication subscriber. Everything initiated by the mobile station is known as mobile originated.
MOR	Mobile-Originated Routing Routing rule that operates on mobile-originated (MO) messages.
MOX	Mobile-Originated eXternal condition External condition rule that operates on mobile-originated (MO) messages.
MSC	Mobile Switching Center

M

MT	Mobile Terminated All transmissions that reach the mobile station and are accepted by it, such as calls or short messages.
MTOR	Outgoing mobile-terminated routing Routing rule that operates on outgoing mobile-terminated (MT) messages.
MTOX	Outgoing mobile-terminated external condition External condition (EC) rule that operates on outgoing mobile-terminated (MT) messages.

P

PBC	Prepaid Billing Controller Performs prepaid charging using the Diameter, CAMEL, or SMPP+ interface.
ping	A network tool used to determine if a target host can be reached across an IP network. Ping estimates the round-trip time and packet loss (if any) rate between hosts.

R

RFC	Request for Comment RFCs are standards-track documents, which are official specifications of the Internet protocol suite defined by the Internet Engineering Task Force (IETF) and its steering group the IESG.
RTR	Router Routes all types of SMS traffic.

S

SCCP	Signaling Connection Control Part
SFTP	SSH File Transfer Protocol (sometimes also called Secure File Transfer Protocol) A client-server protocol that allows a user on one computer to transfer files to and from another computer over a TCP/IP network over any reliable data stream. It is typically used over typically used with version two of the SSH protocol.
SGSN	Serving GPRS Support Node
SMPP	Short Message Peer-to-Peer Protocol An open, industry standard protocol that provides a flexible data communications interface for transfer of short message data.
SMS	Short Message Service
SMSC	Short Message Service Center
SNMP	Simple Network Management Protocol. An industry-wide standard protocol used for network management. The SNMP agent maintains data variables that represent aspects of the network. These variables are called managed objects and are stored in a management information base (MIB). The SNMP protocol arranges managed objects into groups.
SS7	Signaling System #7

T

TCAP	Transaction Capabilities Application Part
TCP	Transfer Control Protocol
TDMA	Time Division Multiple Access A time division multiplex approach which assigns a fixed number of slots per round. The slots can reflect the requirements of the individual stations. If these requirements are known, TDMA can support high efficiency.
TNL	NewNet Mobile Messaging Network Layer NewNet proprietary interface over which Mobile Messaging components communicate.
Tools	A collection of command-line tools for managing and troubleshooting NewNet Mobile Messaging components.
trap	A mechanism used in the context of SNMP (Simple Network Management Protocol) for one-way event notification.
TS	Test Strategy Traffic Server Technical Specification Teleservices

U

UDP	User Datagram Protocol

U

UMTS	Universal Mobile Telecommunications System The standard for 3G used by GSM service providers. UMTS includes voice and audio services, for fast data, graphic and text transmissions, along with transmission of moving images and video.

X

XML	eXtensible Markup Language A version of the Standard Generalized Markup Language (SGML) that allows Web developers to create customized tags for additional functionality.