Oracle® Communications Convergent Charging Controller

Testing Utilities User's Guide

Release 12.0.5
F53851-01
April 2022

ORACLE

Oracle Communications Convergent Charging Controller Testing Utilities User's Guide, Release 12.0.5

F53851-01

Contents

A About the SS7 Protocol Suite

B Supported Protocol Fields for mipt

Preface

This guide describes how to use the Oracle Communications Convergent Charging Controller testing utilities.

Audience

This document is intended for Convergent Charging Controller network operators, system administrators, and system integrators who do functional testing of applications, load testing, and external interface testing.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.

Convention	Meaning
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

About the Convergent Charging Controller Testing Utilities

This chapter provides an overview of the Oracle Communications Convergent Charging Controller testing utilities.

Overview of Convergent Charging Controller Testing Utilities

Convergent Charging Controller communicates internally using a language known as G8-Intelligent Network Application Protocol (G8-INAP), which is a subset of Capability Set 1 (CS1) INAP but also includes bits of CS2 INAP and CAMEL Application Part (CAP). Using this common language allows Convergent Charging Controller components to perform functions without having to translate the low-level languages used by the telephony network.

Convergent Charging Controller interfaces communicate with the physical network in whichever protocol the network demands. The interfaces translate the messages from the physical network into the G8-INAP messages. The passing of messages between Convergent Charging Controller and the interfaces takes place in the Convergent Charging Controller Service Logic Execution Environment (SLEE), where many interfaces can communicate concurrently. Because they are not tied to low-level network languages, Convergent Charging Controller components can be portable and plug into any network as long as an effective interface exists.

The Convergent Charging Controller testing utilities include:

- The Service Logic Program Instance Tester (**slpit**) utility

  The **slpit** utility allows you to test call processing by Convergent Charging Controller applications without the need for a physical telephony network.

- The Messaging over Internet Protocol Tester (**mipt**) utility

  The **mipt** utility allows you to test the sending and receiving of messages over internet-based protocols.

- The short message service center (**smsc**) test tool

  The **smsc** test tool emulates various parts of the short message service (SMS) environment to enable testing of SMS messaging.

The distinction between the **smsc** test tool and the two utilities is that you initiate **smsc** through configuration parameters as part of Convergent Charging Controller startup, whereas you run the two utilities independently from the command line.

Overview of the slpit Utility

The **slpit** utility is a tool that you can use to do functional testing of Convergent Charging Controller applications, high load testing, and external interface testing without concern for the protocol of a given network. From the perspective of the test application, the **slpit** utility emulates a real interface that converts the network messages to and from G8-INAP. It communicates with the application by way of the SLEE, just like a regular interface.

> **Note:**
>
> In this context, *application* or *Convergent Charging Controller application* refers to the SLEE process to which the **slpit** utility is communicating. Usually, this process is either **slee_acs**, which is the main ACS process, or **xmsTrigger**, which is the main XMS process. But it can also be **m3ualf**, which is also a SLEE process. The **m3ualf** process is further described in this section.

The **slpit** utility has the following characteristics:

- It allows you to effectively test Intelligent Network (IN) applications without requiring a physical telephony network or a low-level network-specific test tool.
- You can use it to do functional testing of Convergent Charging Controller applications without concern for a particular network protocol. As long as the application provides the correct functionality in G8-INAP, it will perform the same way on a particular network with the appropriate interface.
- It acts as a normal TCAP interface to trigger IN platform service logic, emulating a service switching point (SSP) and specialized resource function (SRF) interactions.

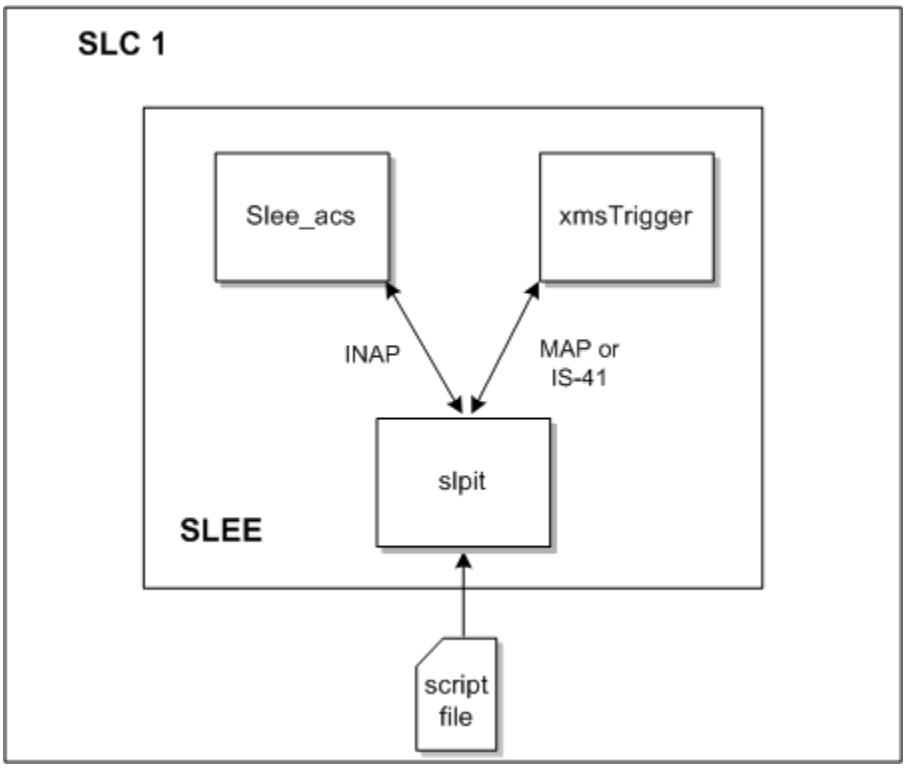
  The **slpit** utility supports the following IN protocols: CAP, MAP, SCCP, CAP3 GPRS and IS-41.
- It uses a script file in which you define the INAP operations that are sent and received for one or more types of calls. A single instance of **slpit** can run many call instances and many calls can be in-progress at once. The script initiates a call and you can specify different distributions and throughput rates. Multiple protocols are supported.
- You can use it to do moderate load testing and external interface testing, in addition to using it for functional testing of applications.
- You can run it in the same SLEE as the application being tested or in a separate SLEE using appropriate TCAP interfaces.

On a production Convergent Charging Controller system, the **slee_acs** process and the **xmsTrigger** process communicate with a process called **m3ualf**, using a TCAP-like protocol. The **m3ualf** process is also a SLEE process. The **m3ualf** process turns the TCAP-like events into messages that are sent over the IP network in a protocol stack that consists of one of the following:

- MAP over TCAP over SCCP over M3UA over SCTP over IP
- CAP over TCAP over SCCP over M3UA over SCTP over IP

Figure 1-1 shows **slpit** running in the same SLEE as the application being tested.

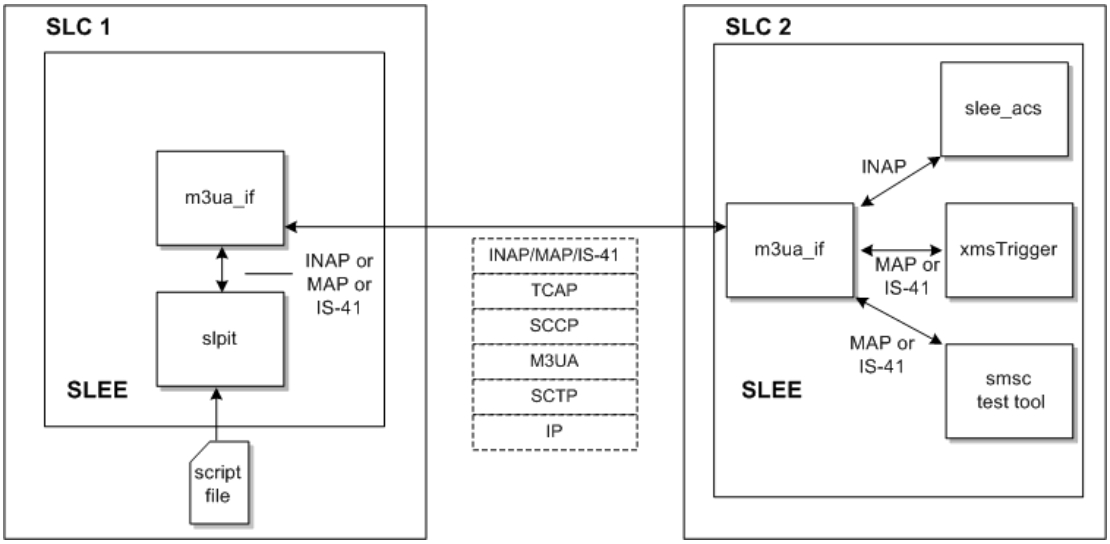
Figure 1-1 slpit Testing Application in the Same SLEE



You can also use **slpit** to generate INAP, MAP, CAP3 GPRS or IS-41 over a specific set of protocols if you run it on a different machine than the one where the application is being tested.

Figure 1-2 shows **slpit** running in a SLEE on a separate machine from the one where the application is being tested.

Figure 1-2 slpit Testing Application in a Separate SLEE



See "Testing Calls and Messages Using the slpit Utility " for more information.

Overview of the mipt Utility

The **mipt** utility is a test tool that allows you to send and receive messages. Depending on the protocol, the **mipt** utility can act as:

- An application service provider (ASP) or a Short Message Service Center (SMSC) when the protocol is one of the following:
  - Short Message Peer to Peer (SMPP)
  - External Machine Interface (EMI)
- An ASP when the protocol is Media Transfer Protocol (MTP) level 3 User Adaptation layer (M3UA)
- A Diameter agent or a Diameter server for the Diameter protocol

You can run multiple instances of **mipt**, acting as ASPs or SMSCs, that communicate with each other on the same machine.

You can use the following protocols with **mipt**:

- Remote Authentication Dial-In User Service (RADIUS)

  RADIUS is a network protocol that is the predecessor to the Diameter protocol and, like Diameter, it is used for authentication, authorization, and accounting.
- Diameter

  Diameter is an authentication, authorization, and accounting protocol. You can also use the Diameter protocol for policy control and resource control.
- EMI

  EMI connects mobile telephones to SMSCs.
- M3UA

  The M3UA protocol enables the SS7 protocol User Part SCCP, as well as others, to run over the internet protocol instead of telephony equipment. It is generally transmitted by using the services of Stream Control Transmission Protocol (SCTP).
- SMPP

  SMPP transfers short message data between message centers and routing and messaging facilities. It is commonly used to transfer short messages and it allows service providers to submit messages in bulk.
- SUA

  The SUA protocol facilitates the transfer of SCCP user messages, such as TCAP, between the signalling gateway and the ASP.

See "Supported Protocol Fields for mipt" for a list of supported fields for each of the protocols that you can use.

To test messages using these protocols, you create a text file, called the *script file*, that contains the operations or messages that you want to test. The **mipt** utility accepts the script file as input and processes the operations that you have defined.

See "Testing IP Interactions with the mipt Utility " for more information.

Overview of the smsc Test Tool

The **smsc** test tool is a multipurpose test tool that runs as a Service Logic Execution Environment (SLEE) interface.

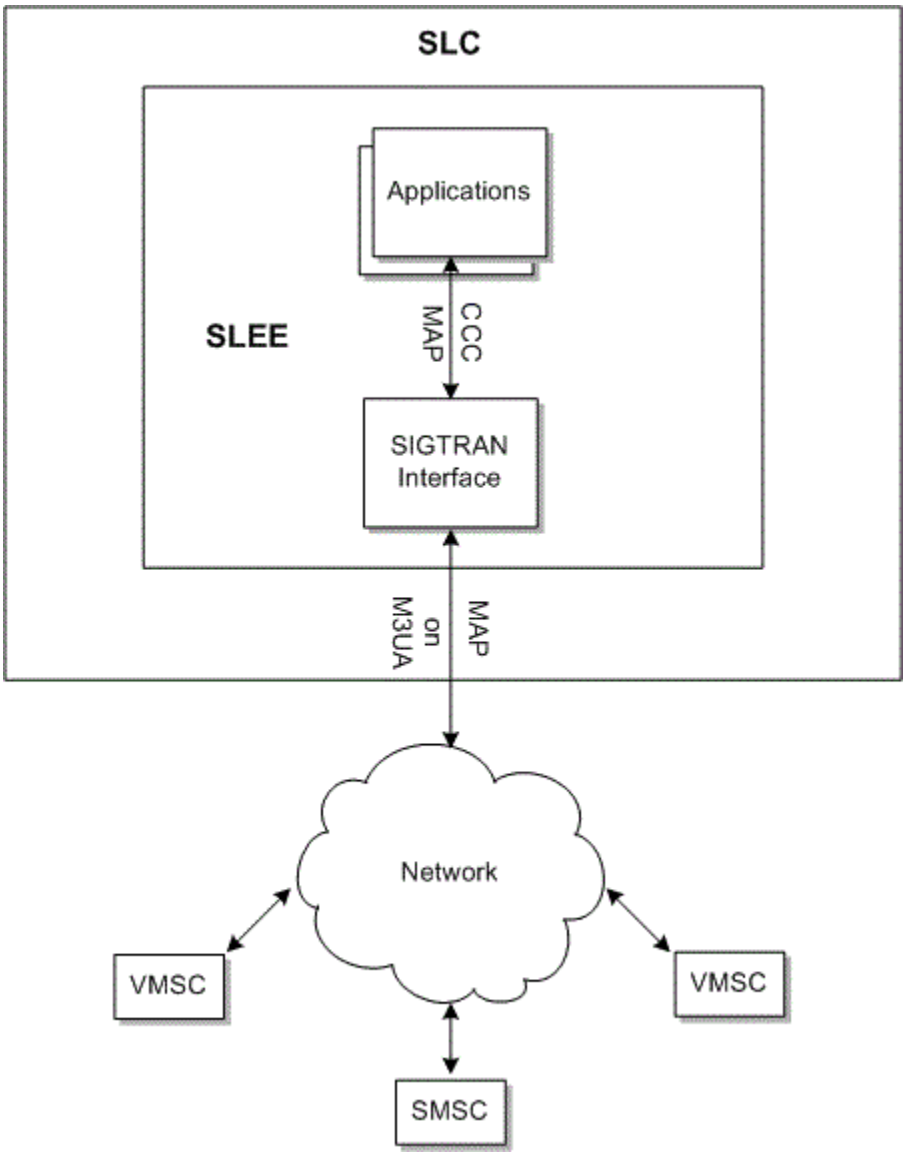
The SMSC test tool emulates the following parts of a short message service (SMS) messaging environment:

- Visitor Mobile Switching Center (VMSC)

  A mobile switching center (MSC) is a telephone exchange in a GSM mobile network. The VMSC is the MSC that the destination phone is attached to, which could be distant from its home network, and, hence, is a visitor.
- Short Message Service Center (SMSC)
- Home Location Register (HLR)

Figure 1-3 illustrates how Convergent Charging Controller applications connect to SMSC, HLR, or VMSC in a production environment.

Figure 1-3 Convergent Charging Controller Connecting to SMSC, HLR, or VMSC in a Production Environment



The SMSC attaches to the SLEE as a Transaction Capabilities Application Part (TCAP) interface. The simulated SMSC handles both Mobile Application Part (MAP) and IS-41 (also known as ANSI-41) incoming short message requests. The **smsc** test tool can simulate an SMSC, a Home Location Register (HLR), and a messaging service center (MSC) at MAP levels 1 through 3. It can also perform one CAP 3 GPRS operation, ActivitytestGPRS.

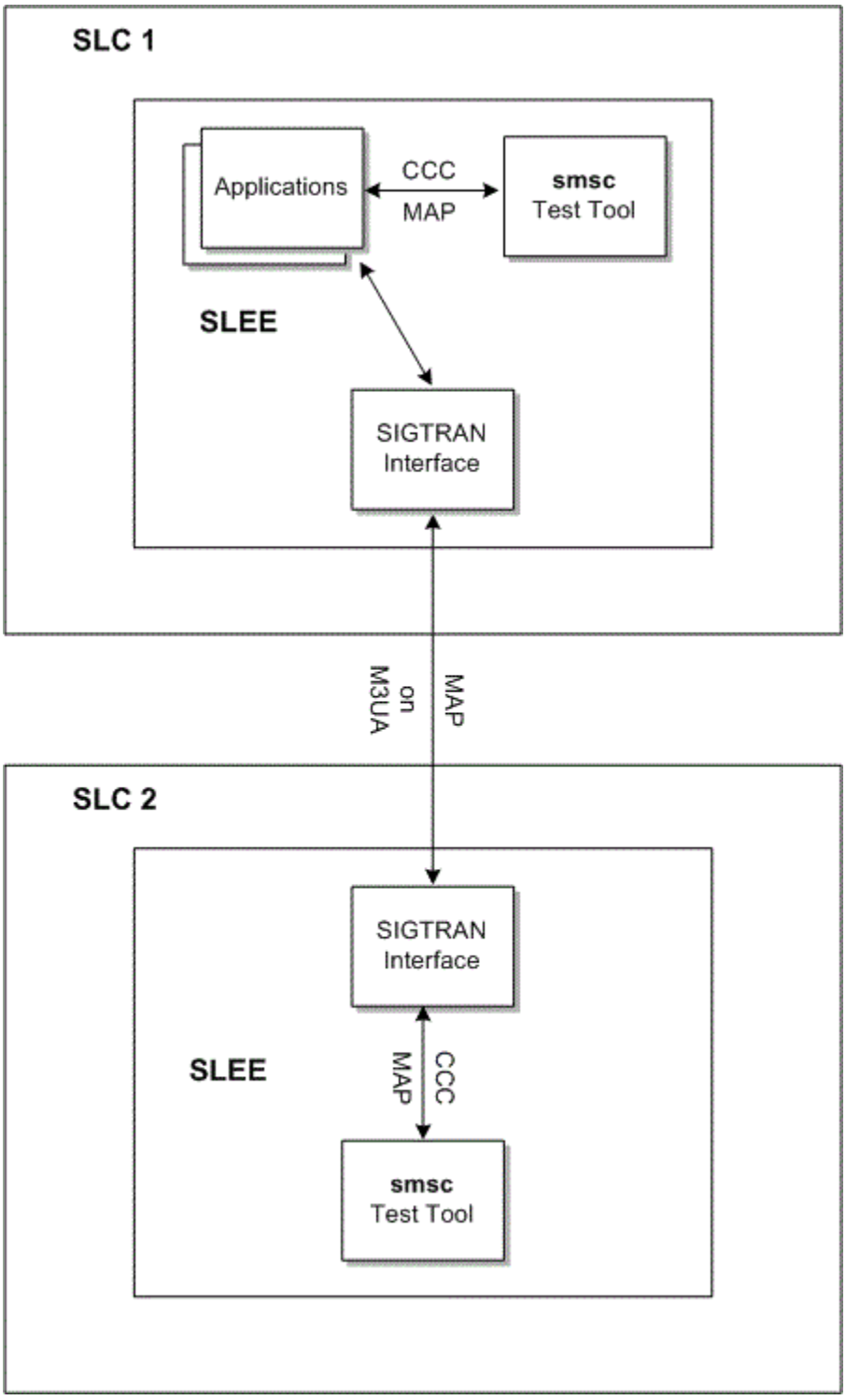
The **smsc** test tool can handle the following operations for MAP and IS-41:

- MAP
  - FORWARD-SHORT_MESSAGE
  - MT-FORWARD-SM
  - SEND-ROUTING-INFO-FOR-SM (HLR operation)

  - SEND-ROUTING-INFORMATION (HLR operation)
  - PROCESS-UNSTRUCTURED-SS-REQUEST
  - UNSTRUCTURED-SS-NOTIFY
- IS-41
  - SmsDeliveryPointToPoint (SMDPP)
  - SMSRequest (HLR operation)
  - SMSNotification

Figure 1-4 illustrates how the **smsc** test tool replaces SMSC, HLR, or VMSC in a testing environment. The diagram illustrates the **smsc** test tool running on the same SLC (SLC1) that you are testing on, and running on a remote SLC (SLC2). The latter case is necessary to test the interaction between the application and the SIGTRAN interface.

Figure 1-4 The smsc Test Tool Environment



See "Testing Messaging with the SMSC Test Tool " for more information.

2

Testing Calls and Messages Using the slpit Utility

This chapter describes how to use the Oracle Communications Convergent Charging Controller **slpit** utility.

About the slpit Utility

The **slpit** utility sends and receives Intelligent Network Application Part (INAP) operations and acts as an interface to the Transaction Capabilities Application Part (TCAP) protocol.

About the SS7 Protocol Suite gives a brief overview of the Signalling System 7 (SS7) protocol suite, of which INAP and TCAP are a part.

The **slpit** utility processes operations from an input text file rather than a real network. The input text file is called the *script file* and it is a file that you create. In the script file, you add commands and send and receive operations that define the call sequences that you want to test. See "Creating the slpit Script File" for more information.

The utility parses the responses from the test application and compares them to the responses that the script file expects.

There are two ways to run the **slpit** utility. In the first way, it replaces the **m3ualf** process so that **slpit** communicates with **slee_acs** and **xmsTrigger** but does not send anything over the IP network. This allows you to test the higher layers of a protocol but does not include any processing that would normally happen inside **m3ualf**.

The second way to run the **slpit** utility allows you to test certain functions that happen in the **m3ualf** process. To run the utility this way, you must configure two machines. For example, if you configure machines SLC1 and SLC2, you configure SLC1 exactly like a production SLC, with **slee_acs** and **xmsTrigger** talking to **m3ualf**. You configure SLC2 with only **m3ualf** and the **slpit** utility on it. See "Running slpit in a Separate SLEE" for information about running the **slpit** utility in this configuration.

See "About the Convergent Charging Controller Testing Utilities " for an overview of the **slpit** utility.

Running the slpit Utility

The **slpit** utility is located in the following directory:

```
/IN/service_packages/TEST_TOOLS/bin
```

The basic command for running the **slpit** utility specifies a service and the name of the script input file. Additional command line options allow you to request validation of the script file, define a global variable, specify the debug output, specify output options, and perform various other actions.

Command Syntax

To run the **slpit** utility, use one of the following commands:

```
slpit -k [<option>...] [<script>]
slpit -V
slpit -h
```

The only command-line option that is typically required to run **slpit** is -k, which allows you to specify a service key other than the arbitrary default of 101. The following example shows the simplest command to run **slpit** with a script file:

```
slpit -k 1 <script file>
```

You can alternatively provide the script on the command line rather than in a file by using the following syntax where <script> is your script code:

```
slpit -k 1 < <script>
```

When you are trying to correct syntax errors in a script, the **-c** option is useful because it causes **slpit** to exit immediately after parsing, without running any calls:

```
slpit -c <script file>
```

Command-Line Options

The **slpit** utility takes the following command-line options:

-A
Constructs ANSI SCCP addresses rather than ITU addresses, which is the default.

-a
Act as an application instead of the default interface. For more information, see "Running slpit in a Separate SLEE".

-c
Validate the script file and exit.

-C <CSV_file>
Writes the following values to the specified comma-separated values (CSV) file every ten seconds: the time, calls per second (CPS), and outstanding call count.

-D <name>=<value>
Predefines a global variable with the specified name and value, where <name> is the name of the variable and <value> can be an integer or a double quoted string. Defining a global variable could be useful for making a change to the script easier in the future. For example, you could define the destination phone number and then refer to it in the call definition using the variable. Then, in the future, when you want to define a new call with a different number, you would only need to change the number in one place.

-d <level>
Sets the level of debug output. Valid levels are 0 to 5, with 0 indicating no output and 5 indicating the maximum output.

-g
Makes the utility more tolerant of errors, causing it to continue, if possible, rather than abort.

-h
Prints version and build information, like the **-V** option, plus a summary of the usage information.

-i <interval>
Report call summary information at the interval specified, which is a number of seconds.

-I <name>
Adds the value of <name> as a suffix to the interface name. This option is required if you run more than one instance of **slpit** simultaneously in the same SLEE.

-k <key>
Initiates calls with the specified service key value rather than 101, which is the default. The service key values are defined in the **/IN/service_packages/SLEE/etc/SLEE.cfg** file.

-M <interval>
Used with the **-m** option to write average timing information per primitive to the CSV file at the interval specified by <interval>. If the specified interval is **0**, the average timing information will be written when the script completes. This parameter works only if the **slpit** script expects a response because the average durations cannot be calculated otherwise.

-m <directory>
Enables logging of timestamps per TCAP primitive for messages sent and received. The utility writes the information by call type to a comma-separated values (CSV) file in the specified directory.

-N
Instructs **slpit** not to add itself as an interface to the SLEE as the SLEE has added it already.

-n
Makes **slpit** ignore any received TCAP_CANCEL messages.

-O <flags>
Enables the specified output flags. See "Output Options" for more information.

-o <level>
Sets the level of normal output. Valid levels are 0 to 5, with 0 indicating no output and 5 indicating the maximum level. The default is 3, which produces a reasonable amount of output that is not excessive.

-p <protocol>
Sets the preferred TCAP protocol to use when there is a conflict between the INAP/CAP and MAP tag values, as there is some overlap. Valid values are `map`, `is41`, and `inap`. The default is `inap`.

-R
Recreates the main dialog, if it no longer exists, using the last received originating reference as a correlation ID. This is required for the CAP3 GPRS message sequence. This option does not work if the **-a** option is also submitted.

-T
Enables the use of the SLEE Timer interface for delays between sending new requests, or between responses to inbound requests. Without this option, delays are handled by polling. Use of this option is not recommended. The Timer interface is not ideal for this purpose.

-v
Enables verbose output, setting the output level to the maximum. This is equivalent to setting the **-o** option to 5.

-vv or -v -v
Sets the debug output level to the maximum.

-V
Prints version and build information and then exits.

Output Options

In addition to the overall output level that is controlled by the **-o** option, you can enable the following more specific output features with the **-O** option. Most of these options are enabled automatically at various numeric output levels.

calldefntrace
Displays a brief summary of the call definition at each step in the call execution, including an indication of the current step. Automatically enabled at overall output level 4.

callrate
Displays the average call rate achieved before the **slpit** utility terminates. It calculates the call rate by dividing the total run time by the number of calls started.

callsummary
Prints a table summarizing the number of calls run, the number of successful calls, and the number of partially and totally failed calls. Call types that have at least one aborted call are marked with four asterisks (****); call types with failed calls are marked with a single asterisk (*).

fullcallsummary
Prints a more detailed call summary table than the one produced by the **callsummary** option. The information is the same as produced by the **callsummary** option, but the format of the table produced by **callsummary** is more concise.

triptiming
Records and displays round-trip message times for each call.

sleecheck
Checks for changes in the count of free resource objects in the SLEE at the end of the run. If there are any changes, displays a table of the free counts. The resources can be calls, dialogs, events, and application instances. The rows for resource types that showed a positive delta are marked with a single asterisk (*); those with a negative delta, which indicates a potential memory leak, are marked with four asterisks (****). A positive delta in free resource counts indicates that running one or more calls caused resources to be freed. This is not uncommon with Advanced Control Services (ACS), which is prompted to free a SLEE management event when the first call event arrives.

parsedebug
Enables the extremely verbose debug output for the GNU Bison parser. This option is not automatically enabled at any output level because it is useful primarily when debugging the parser.

Running slpit in a Separate SLEE

When running the **slpit** utility in a separate SLEE, you must specify the **-a** and **-k** command-line options. The **-k** option must specify the SLEE service key that is assigned to the **m3ualf** process in the **/IN/service_packages/SLEE/etc/SLEE.cfg** file on the machine where the **m3ualf** process is running.

Exit Codes

Table 2-1 describes the exit codes that the **slpit** utility uses to indicate whether the run was successful. The **slpit** utility writes the exit code to **stdout** (standard output), which you can redirect to a file if you wish.

```
slpit -k [<option>...] [<script>] > <outfile>
```

Table 2-1 slpit Exit Codes

Exit Code	Description
0	Execution completed successfully
1	General or usage error, which usually indicates that the command-line options were not valid.
2	Script parsing error. Either **slpit** could not read the script file or it encountered a syntax error in the script. The utility displays diagnostics on **stderr** (standard error output).
4	The initial connection to the SLEE failed, most likely because no SLEE is running.
5	A SLEE entity that **slpit** required could not be contacted. This can occur when **slpit** is directed to use the Timer interface for running timers but the interface could not be found.
6	Call creation failed. From most likely to least likely, the possible reasons include: the service key for the call being created is not configured; a resource for a SLEE dialog or for call instances has been exhausted; the SLEE for the service configured on the service key never started or has been stopped.
10	At least one call instance failed. There were no errors that prevented **slpit** from running to completion, but at least one call instance ended in the FAILED state.
11	At least one call instance aborted. There were no errors that prevented **slpit** from running to completion, but at least one call instance ended in the ABORTED state.

Managing Script File Processing

The script file is an input file in which you define the call instances that you want the **slpit** utility to process. Call instances are defined with a set of commands and INAP operations

that you specify in sets of `send` and `receive` messages. The following example shows the beginning of a call instance definition:

```
define call assisting_ip_pa {
    SERVICE_NUMBER ?= "555801"

    send {
      initialdp
        calledpartynumber SERVICE_NUMBER
        callingpartynumber "40002000"
        callingpartyscategory 10
        locationnumber "40002000"
        eventtypebcsm analyzedinformation
    }

    receive {
      establishtemporaryconnection
        address "1234"
    }
```

You start call processing by including a `startcall` command; for example:

```
startcall assisting_ip_pa using once
```

The **slpit** utility can reference the call types that you define in the script file only after the script file has been parsed. Starting a call creates a call distribution but the distribution does not start creating new call instances until script processing completes.

When you run the **slpit** utility, it processes all distributions and calls in the script file before stopping.

In general, each call that **slpit** executes produces one call instance and one or two dialogs in the SLEE. The first dialog is called the *main* dialog. The second dialog, which will exist only for parts of some calls, is referred to as the *assisting* dialog.

The first message sent for a call must be an `InitialDP` or an appropriate `TC_BEGIN` message. Alternatively, the first action in a call can be a receive message with a `cs1InitiateCallAttempt` or a `cap4InitiateCallAttempt` operation.

> **Note:**
>
> Sending an `AssistRequestInstructions` message creates a second dialog on the same call instance to simulate the dialog between the intelligent peripheral and the service control point (SCP).
>
> When **slpit** runs in the same SLEE as the application, the correlation ID for each dialog is not required to match. Normally, the TCAP interface would resolve the correlation ID to create the second dialog on the correct call instance.

> **Note:**
>
> Receiving a `DisconnectForwardConnection` operation is not a special case. The shutdown of the assisting dialog must be explicitly stated.

The **slpit** utility does little validation to ensure valid call flow. It primarily ensures that the dialog is handled correctly. For example, you do not get a warning if you forget to send an `ApplyChargingReport` message at the end of a monitored call, but you do get a warning if you do not explicitly terminate a dialog.

Using Distributions

The **slpit** utility creates a distribution with a list of call types and other parameters that control the launching of calls, which is known as the call rate, and the terminating condition of the distribution, which is generally the number of calls launched. A distribution processes the call types in a round-robin fashion until the completion condition is met. The **slpit** utility allows you to create the following types of distributions:

- A uniform distribution has an interval and a total call count. The interval specifies the number of seconds that are to elapse before launching each call until the total number of calls is reached. The practical minimum interval is greater than a microsecond but less than a millisecond. The following `startcall` line, for example, would run the call every 0.5 seconds for a total of 10 times:

  ```
  startcall using uniform 0.5 10
  ```

- A Poisson distribution has a lambda value and a total call count. The lambda value represents the average interval between calls rather than the exact interval.
- The `once` distribution launches one of each specified call type immediately.

  A `once` distribution will run through the contents of the given call type once and report a result of `SUCCESS`, `FAILED` or `ABORT`.

The type of distribution is determined by the type of testing that you are performing. You specify the distribution type in the script file using the `startcall` command, for example:

```
startcall <id> using <distribution>
```

So for a script in which you wanted to run only one call that was started with `define call 982 {`, you would have to start the call with a line like the following:

```
startcall 982 using once
```

Using TCAP Primitives

All TCAP messages are primitives although some primitives are not messages. That is, some primitives are transferred only inside the local machine. A TCAP primitive contains zero or more TCAP components and can be one of the types described in Table 2-2.

Table 2-2 Types of TCAP Primitives

Primitive Type	Description
Unidirectional	A single standalone instruction. It is both the start and end of a dialog.

Table 2-2 (Cont.) Types of TCAP Primitives

Primitive Type	Description
Begin	Begins a dialog with other primitives coming after it.
Continue	A subsequent primitive sent on an existing dialog with other primitives coming after it.
End	The last primitive, which closes its dialog.
Abort	Closes the dialog, possibly due to an error.
Cancel	Closes the dialog when the invoke timer expires without receiving a response. This is an example of a primitive that is not a message.

Receiving Expected Operations

Each received message corresponds to a single TCAP primitive and can contain one or more INAP operations.

The received message must contain the expected INAP operations in the order specified in the `receive` message section of the call definition. See "Call Definition Commands and Messages" for more information about defining calls.

> **Note:**
>
> Operations can come in one primitive but also can sometimes come in separate primitives, depending on the application or the service.

If the received operation types do not match the expected operation types, the **slpit** utility aborts the call. If the parameters received for each operation do not match the expected parameters, **slpit** reports the result of the call as `FAILED` but continues to process the remaining operations in the call definition. See "About Expressions and Comparators" for more information about parameter values.

Receive operations can time out, which prevents **slpit** from waiting for call completion when the call has been lost. The global default for timeout is 15 seconds.

When a timeout occurs, you can execute a sequence of messages to finish the call. The default action is to abort the call, which closes any open dialogs. Aborting the call is not likely to be the most desirable behavior, however, because it does not cause a TCAP ABORT message to be sent to the application when it is running in the same SLEE. Therefore, if you expect a timeout, you should override the default with a more appropriate action.

Managing Dialogs

The send message includes options that allow you to specify a particular dialog on which to send and also to end a SLEE dialog. You can also use the abort primitive to abort a dialog and use other messages to send an error to ACS.

Specifying a Particular Dialog

To send operations specifically on either the `main` or `assisting` dialog, specify the dialog in the `send` primitive. For example, the following `send` primitive sends the operations on the `assisting` dialog:

```
send assisting {
    <operations>
}
```

To send on the main dialog, specify `main` instead of `assisting`. If you do not specify a dialog, the `main` dialog is assumed.

Ending a SLEE Dialog

The easiest way to end a SLEE dialog is to include the **end** option in the last send message in the dialog, as shown in the following example:

```
send end {
    ...
}
```

When a call completes, whether it is successful or aborted, the **slpit** utility automatically closes any open dialogs. If the **slpit** utility runs in the same SLEE as the application, the application receives only an indication that the dialog is closed and might not handle it in the same way that it does the shutdown of a real TCAP dialog. Therefore, if a call ends with a status of Failed or Okay, and it has dialogs open, the **slpit** utility displays a warning message. If a call was aborted, you can assume it might have an open dialog.

The **slpit** utility terminates a dialog when the application sends or receives a terminating event. When the **slpit** utility ends a dialog, it writes a line of output that indicates the number of messages that are still in the queue. Usually, you can ignore these messages because only internal SLEE messages will be left.

Aborting a SLEE Dialog

You can also explicitly abort a dialog by using the `abort` message. Specify the **open** option to abort any open dialogs.

```
abort  [main | assisting | open]
```

Completing Calls

Each call instance finishes in one of the following states:

- Aborted

  Execution of the call was interrupted because something was sufficiently wrong that the call could not or should not continue. For example, an attempt was made to send an event when a dialog was no longer available, or a run-time error occurred.

- Failed

  The call was not completely successful but the errors were not sufficient to interrupt the call. The most likely cause is a discrepancy between the received and expected

parameters for an operation. The call is failed but allowed to continue because the difference might not be significant.

- Okay

  The call completed without errors.

A call can finish for the following reasons:

- The call execution reaches the end of the call definition; the final call state will be either Failed, or Okay depending on whether there were errors in the run.
- A `finish call` command is executed: The call run is immediately finished either with its current state or the override state that is specified in the `finish call` command.
- The **slpit** utility encounters a serious error: The utility aborts the call immediately.

Cancelling slpit

You can run multiple calls with **slpit**, either by specifically defining each call in the script input or by using the uniform or Poisson distribution models.

By default, the **slpit** utility does not stop generating calls if any call aborts or fails. You can change this behavior by using the `cancel after` command. You can place this command anywhere outside a call definition in a **slpit** script. The command has the following forms, each of which is self explanatory:

```
cancel after none
cancel after abort
cancel after failure
cancel after abort or failure
```

See "The Call Sequence" for more information about these commands.

You can also allow the **slpit** utility to continue after an abort or failure until a specified limit is reached.

You can use the following form of the cancel command to cancel a run after a specified number of failures or aborts occurs.

```
cancel after <number> [abort|failure]
```

If the number of specified aborts or failures occurs for the call, this command causes the **slpit** utility to stop call processing and exit. The program accepts either `abort` or `aborts`. It also accepts either `failure` or `failures`.

The following command specifies a time limit, in seconds, on the number of failed or aborted calls that the **slpit** utility can receive before it cancels call processing and exits:

```
cancel after <number> [aborts|failures]in <number> seconds
```

The program accepts either `second` or `seconds`.

Sending an Error

You can send an error to ACS or the application that you are testing by using either the `tcapReject` message or the `error` message. See "tcapreject" and "error" for more information.

Creating the slpit Script File

The **slpit** utility processes a script file that consists of a few commands and a set of INAP send and receive operations, which define the progress of one or more call instances.

Syntax

The following syntax conventions are used to describe the commands and operations that appear in the script file.

[ ]
Square brackets indicate that the enclosed items are optional. For example, the `correlationalid` parameter in the following operation is optional.

```
establishtemporaryconnection
  address <digits>
  [correlationid <digits>]
```

|
A pipe separates one or more choices. For example, in the following `finish call` operation, you can optionally specify a final state of `aborted`, `failed`, or `okay`.

```
finish call [aborted|failed|okay]
```

...
An ellipsis indicates that an item can be repeated one or more times. In the following example, `part` must occur at least once but the ellipsis indicates that it can be repeated one or more times.

```
[variableparts <part> [<part>...]]
```

<>
Angle brackets indicate a placeholder that you replace with a specific value. The placeholder typically specifies the value's atomic token or basic data type such as `<integer>`, `<string>`, or `<bcd>`.

The **slpit** utility supports the following three styles of comments, which can appear anywhere in the script file:

- //

  Two forward slashes indicate C++ style comments that can extend to the end of a line. The following line illustrates a full line comment:

  ```
  //This is a full line comment        calledpartynumber "049393520" // This is an
  in-line comment
  ```

- #

  A pound character indicates shell-style comments that can extend to the end of a line.

- /*... */

  Text enclosed by asterisks and then forward slashes indicate C-style comments that can extend multiple lines between the beginning and ending delimiters.

Commands

You can include the following commands in the script file in addition to the messages and operations that define a call:

include <file>
Includes the named file in the **slpit** script, enabling you to include a call sequence that is defined in a separate file. The value of `<file>` includes the directory path to the file's location.

define call <ID> { <call sequence> }
Defines a set of call sequence messages and operations. The `<ID>` is an identifier you assign to the call and use to reference the call in other commands. See "The Call Sequence" for more information about `<call sequence>`.

startcall <ID> [<ID>...] using <distribution> [<seed>] maxconcurrent <limit>] \ [after <wait_seconds>]
Defines the call types, the number of calls, and the call rate at which to start generating calls. You can start multiple call types and call rates by including multiple `startcall` commands in the script file.
For all distribution types, you can specify a random seed, which is a number that will be used to initialize the call rate. If not specified, the current clock time is used.
For all distribution types, you can also specify a maximum concurrent number of calls to hold open. This overrides any calls-per-second (CPS) rate and causes a lower CPS rate to be used. This is useful for specifying the maximum load that can be supported for the test system.
For all distribution types, you can specify that the block of calls are to be run after a wait time of a specified number of seconds. This is useful for specifying a stepped call rate, in which you define one `startcall` for each step, with each one timed to begin after the preceding one has finished.
After the keyword `using`, you can use the following forms of the command:

```
uniform <delay> <count>
once
poisson <delay> <count>
poisson <delay> <ramp> <count>
```

The `<delay>`, `<count>`, and `<ramp>` values must be defined as a number with a decimal point.
The `<delay>` parameter is the average interval between calls. You can alternatively express this value as calls per second and you can do so by using the `cps` keyword. For example `10.0 cps` is equivalent to a `<delay>` value of `0.1`.
The first form of the `poisson` command generates calls at random with the average interval between calls specified by the `<delay>` parameter.
The second form of the `poisson` command ramps up from zero calls per second to 1/ `<delay>` calls per second, taking about `<ramp>` seconds to reach the maximum call rate. It then flattens off at that rate.

Call Definition Commands and Messages

The basic format of a call definition looks like this:

```
define call <ID> {
    <call sequence>
}
```

Each call type is identified by an ID that can be either a number or a name that starts with a letter and contains only letters, digits, and underscores.

The `<call sequence>` consists of a set of call definition messages that describe the progress of a call.

The Call Sequence

The call sequence consists of one or more of the following call definition messages:

```
send [end] [assisting|main] { <message details> ... }
receive [assisting|main] { <reponse details> ... }
allow receive abort assisting
[send] abort [assisting|main|open]
wait <delay>
<ID> = <expression>
<ID> ?= <expression>
waitforcalls <delay> seconds|calls
finish call [aborted|failed|okay]
default timeout none
default timeout <expression> [ { <new call sequence> } ]
close [assisting|main|open]
cancel after [none|abort|fail] [or [abort|fail] ]
```

send

The **slpit** `send` message sends an event containing one or more operations, as determined by the message details, which you can modify through the use of various flags.

```
send [end] [assisting|main] { <message details> ... }
```

The `end` flag causes the messages to be sent as the final event on the dialog. You can use the `assisting` or `main` flag to override the dialog on which the message is sent. See "Managing Script File Processing" for more information.

receive

The **slpit** utility expects to receive an event containing one or more operations as determined by the response details in a `receive` message, including CS1InitiateAttempt and CAP4InitiateCallAttempt operations. You can use either the `assisting` or the `main` flag to override the dialog on which the message is expected to arrive.

```
receive [assisting|main] { <reponse details> ... }
```

See "Managing Script File Processing" for more information.

allow receive abort

An `allow receive abort` message indicates that the **slpit** utility should expect an abort to arrive from TCAP on the specified dialog at some time in the future. This is different from `receive` in that the **slpit** utility does not stop and wait for the abort, but continues processing.

```
allow receive abort assisting
```

abort

An `abort` message causes the **slpit** utility to send a TCAP abort on the specified dialog or dialogs (the default is the `main` dialog). Specifying open dialogs causes the **slpit** utility to abort any dialogs still open for the call.

```
[send] abort [assisting|main|open]
```

wait

The `wait` message causes the **slpit** utility to pause its processing of the call for a specified delay or until it is interrupted by a received event. You can specify the delay as an integer value representing microseconds or as a floating point value representing seconds. In other words, if the value contains a decimal point, the unit value is seconds. If it does not contain a decimal point, the unit value is microseconds. The following example illustrates the format of the message:

```
wait <delay>
```

<ID> = <expression>

The **slpit** utility uses the `<ID> = <expression>` definition to evaluate an expression and assign its value to a named variable. You can always assign an expression to ID using `<ID> = <expression>` but `<ID> ?= <expression>` only assigns an expression to `ID` if `ID` has not been already defined in the script.

```
<ID> = <expression>
<ID> ?= <expression>
```

waitforcalls

The `waitforcalls`message causes the **slpit** utility to pause its processing for a specified number of seconds or until the specified number of new calls started by TCAP_BEGIN requests have finished. You specify the number of seconds to wait or the number of calls to process. The following example illustrates the format of the message:

```
wait <delay> seconds|calls
```

finish call

The `finish call` message finishes the call. You can specify a final state of `aborted`,`failed`, `or okay` to override the established state. For example, `finish call okay` causes a failed call to be recorded as successful.

```
finish call [aborted|failed|okay]
```

default timeout

The `default timeout` message specifies the default timeout that the **slpit** utility uses when waiting for a message. If you specify the `{<new call sequence>}` section, **slpit** will run the new call sequence when the timeout occurs rather than the lines that follow in the main call definition. If you specify `none`, it turns off the timeout altogether.

```
default timeout <expression> [ {<new call sequence>} ]
default timeout none
```

close

The `close` message closes the SLEE dialog by way of a `DIALOG CLOSED` event on the given dialog. If you use the `open` option, the **slpit** utility closes all open dialogs.

```
close [assisting|main|open]
```

cancel

The `cancel after` message forces the **slpit** utility to exit any call immediately when the call fails or is aborted. This feature is most useful when running multiple calls in one **slpit** run, as when using the uniform and Poisson call distribution models.

This message has the following four formats:

- `cancel after none`

  This format prevents the **slpit** utility from exiting the run on the abort or failure of the call.

- `cancel after abort`

  This format causes the **slpit** utility to stop processing or generating calls and exit if the call aborts. You can substitute the word `aborts` for `abort`.

- `cancel after fail`

  This format causes the **slpit** utility to stop call processing and exit if the call fails. You can substitute the words `failure`, `failures`, and `fails` for `fail`.

- `cancel after abort or fail`

  This format causes the **slpit** utility to stop processing or generating calls and exit if the call aborts or fails. You can substitute words as described in the other formats of the `cancel` message.

You can use multiple `cancel` messages like this in the same call definition to handle calls that might not fail before a certain command, but could fail after another command.

You can also define a global cancellation strategy outside of a call definition. See "Cancelling slpit" for more information.

About Expressions and Comparators

An expression generates a value that you can use, for example, as the parameter value for a `send` operation. The simplest form of an expression is a constant value. For example, "5551234" appearing in a **slpit** script is usually an expression that generates a digit string. More complex expressions are supported:

- Expressions that use the value of a variable by name.
- Expressions that use limited integer arithmetic: subtraction, addition, and multiplication. Integer arithmetic expressions may also contain parentheses for grouping.

For example:

```
callConnectedElapsedTime(talktime - 20) * 10
```

You can also specify ranges of numbers as an expression, including the Nature of Address of the generated numbers, which defaults to 3, if not specified. The syntax looks like this:

```
CLI = RANGE [ (<integer: NoA>) ] "<start of range>" "<end of range>" SEQUENTIAL|RANDOM
[DISTINCT|MULTIPLE]
```

For example,

```
CLI = RANGE(4) "49900010001" "49900020001" SEQUENTIAL
```

or

```
CLI = RANGE "49900010001" "49900020001" RANDOM
```

> **Note:**
>
> Use the `DISTINCT` or `MULTIPLE` option with `RANGE` for `initialdp callingpartynumber` messages only.
>
> - `MULTIPLE` means a calling party can have multiple calls in progress.
> - `DISTINCT` means a calling party can have only one call in progress at a time.
>
> See "initialdp" for more information.

You can obtain values from a text file, such as one for vouchers. For example, the following expression will take a line from **vouchers.txt** and use that value wherever VOUCHERNUM is used:

```
VOUCHERNUM=FROM_FILE "vouchers.txt"
```

If you want to randomly use rows from **vouchers.txt**, you need to randomize the file before you pass it to the **slpit** utility. Not having enough rows in your file to match the number of calls causes the **slpit** utility to produce an error and stop after the numbers run out.

You can obtain INAP numbers from a text file. For example, the APARTY expression will take a line from the **APARTY_INAP.txt** file and use the value wherever APARTY is used:

```
APARTY = FROM_INAP_FILE "APARTY_INAP.txt"
```

If sufficient rows are not available in your file to match the number of calls, the **split** utility produces an error and stops once the numbers have run out. For example:

```
(5) 111121 screening 1 presind 2 numberplan 3 innorni 0
```

section of the **slpit**:

```
define call  deciseconds_camel {
APARTY = FROM_INAP_FILE "APARTY_INAP.txt"
BPARTY = FROM_INAP_FILE "BPARTY_INAP.txt"
send {
       initialdp
       appContext          "0,4,0,0,1,0,50,1"
       calledpartynumber     BPARTY
       callingpartynumber    APARTY
       locationNumber    "111144"
}
}
```

A comparator is a pattern for checking received values such as the parameters in received operations. There are three comparators:

- `any`
- `[=] <expression>`
- `<comparator> -> <ID>`

The `any` comparator matches any value.

The simplest comparator is an expression that tests for equality. You can optionally precede the expression with = to make the equality test explicit. Because the simplest expression is a constant value, comparators usually test for equality with a simple constant value. It might also be useful to compare to the value of a variable.

The last comparator generates a match or a mismatch based on the result of the comparator, which can be any other comparator. It stores the value being checked in the variable named by `<ID>`. This allows you to store a received parameter value for later use.

In the syntax description, `<integer comparator>` indicates that you can include any comparator at that point, but the comparison should be for an integer, so the expression or expressions underlying the comparison should generate integers. The same thing applies for other comparator types like `<number comparator>`.

```
<number>:
[(<noa>)] <digits>
[screening <integer>]
[presind <integer>]
[numberplan <integer>]
[innorni <integer>]
```

For outgoing numbers, the following default values are substituted for any field not specified:

```
noa=3 screening=0 presind=1 numberplan=1 innorni=0
```

For incoming numbers, any value is allowed for fields that have not been specified except `<digits>`.

Call Messages

Call messages are divided into `send` message operations and `receive` message operations.

This chapter does not explain the semantics of INAP, MAP, or CAP operations, except where the mapping from the parameters in the script to those in the actual operations is not obvious. Please refer to the relevant standards documentation for the descriptions and procedures for particular operations. See the following standards documents for more information:

- *Intelligent Network (IN); Intelligent Network Capability Set 1 (CS1); Core Intelligent Network Application Protocol (INAP); Part 1: Protocol specification*. European Telecommunication Standard, ETS 300-374-1, September 1994.
- *3rd Generation Partnership Project; Technical Specification Group Core Network; Customized Applications for Mobile network Enhanced Logic (CAMEL) Phase 4; CAMEL Application Part (CAP) specification (Release 5)*. 3GPP 29.978 5.4.0 (2003-06).
- *Digital cellular telecommunications system (Phase 2+); Mobile Application Part (MAP) specification (GSM 09.02 version 7.5.0 Release 1998)*. ETSI TS 100 974 V7.5.0 (2000-07).

In the syntax descriptions in this section, `<integer expression>` indicates that an expression should appear at that point and the expression should produce an integer. Likewise for the syntax `<number expression>`. The syntax `<integer comparator>` indicates that you can

include any comparator at that point, but the comparison should be for an integer, so the expression or expressions underlying the comparison should generate integers. Likewise for the syntax `<number comparator>`. See "About Expressions and Comparators" for more information.

Send Message Operations

You can use the following operations in the `send` message portion of a call definition.

activityTestResult

This operation is available for use with CAMEL Phase 1. It is the returned result for activityTest. This operation has no parameters.

```
activityTestResult
```

alertServiceCentre [<parameters>]

You can use this operation for MAP handling. It sends alerts between MSC and HLR and it has the following parameters in any order:

Table 2-3 alertServiceCenter Parameters

Parameter	Value
msisdn	<number expression>
serviceCentreAddress	<number expression>

alertServiceCenterWithoutResult [<parameters>]

You can use this operation for MAP handling. It sends alerts between MSC and HLR and it has the following parameter in any order:

Table 2-4 alertServiceCenterWithoutResult Parameters

Parameter	Value
msisdn	<number expression>
serviceCentreAddress	<number expression>

anyTimeInterrogation [<parameters>]

This operation queries for information between GSM SCF and HLR. It has the following parameters:

Table 2-5 anyTimeInterrogation Parameters

Parameter	Value	Min.	Max
requestedInfo	(locationInformation | subscriberState)	NA	NA
imsi	<bcd>	NA	NA
msisdn	<bcd>	NA	NA
qmScf	<bcd>	NA	NA

Table 2-5 (Cont.) anyTimeInterrogation Parameters

Parameter	Value	Min.	Max
sccp_orig_pc	<integer>	0	65535
sccp_orig_ssn	<integer>	0	255
sccp_orig_tt	<integer>	0	255
sccp_orig_np	<integer>	NA	NA
sccp_orig_noa	<integer>	0	127
sccp_orig_rti	<integer>	0	1
sccp_orig_digits	<digits>	0	15
sccp_dest_pc	<integer>	0	65535
sccp_dest_ssn	<integer>	0	255
sccp_dest_tt	<integer>	0	255
sccp_dest_np	<integer>	0	15
sccp_dest_noa	<integer>	0	127
sccp_dest_rti	<integer>	0	1
sccp_dest_digits	<digits>	NA	NA

Table 2-6 lists the valid combinations of the fields that make up a global title:

Table 2-6 Fields that make up a global title

Global Title Type	Fields
1	noa, digits
2	tt, digits
3	tt, np, digits
4	tt, np, noa, digits

For global title types 3 and 4, the encoding is always binary coded decimal (BCD) that is 1 when there is an odd number of digits and 2 when there is an even number of digits.

applychargingreport

This operation provides feedback from the service switching function (SSF) to the service control function (SCF). It has the following format:

```
applychargingreport
    thresholdtime <integer>
    endofcallindicator <integer>
    [freecallindicator <integer>]
```

The following format is available for use with INAP CAMEL extensions:

```
applychargingreport
    receivingSide <number>
    (timeNoTariffSwitch <number> |
     timeSinceTariffSwitch <number> |
     timeSinceTariffSwitch <number> tariffSwitchInterval <number> )
```

```
    [ callActive <number> ]
    [ callReleaseAtTcpExpiry <number> ]
```

applychargingreportGprs

This operation provides a report from the GPRS SCF to the GSM SSF. It has the following format:

```
applyChargingReportGprs
    ( gprsvolumeifnotariffswitch <integer> |
      gprsvolumesincelasttariffswitch <integer> [ gprstariffswitchinterval
<integer> ] |
      gprstimeifnotariffswitch <integer> |
      gprstimesincelasttariffswitch <integer> [ gprstariffswitchinterval
<integer> ] )
    <qos-list>
    gprsActive <integer>
    [ gprsPdPid <integer> ]
    [ <ChargingRollover> ]
```

The `<qos-list>` section is one or more of the following in any order:

```
gprsrequestedqos <gprs-info>
gprsnegotiatedqos <gprs-info>
gprssubscribedqos <gprs-info>
```

The `<gprs-info>` data is the same as defined in `intialDpGprs`. See "InitialDpGprs" for more information.

The `<ChargingRollover>` section is optional, consisting of either:

```
<TransferredVolumeRollOver> | <ElapsedTimeRollOver
```

The `<TransferredVolumeRollOver>` parameter consists of a choice of:

```
    gprsrovolumeifnotariffswitch <integer> |
    <ro-VolumeIfTariffSwitch>
```

Where `<ro-VolumeIfTariffSwitch>` consists of a sequence of:

```
gprsrovolumesincelasttariffswitch <integer> (optional)
gprsrovolumetariffswitchinterval <integer> (optional)
```

The `<ElapsedTimeRollOver>` consists of a choice of:

```
    gprsrotimeifnotariffswitch <integer> |
    <ro-TimeIfTariffSwitch>
```

Where `<ro-TimeIfTariffSwitch>` consists of a sequence of:

```
    gprsrotimesincelasttariffswitch <integer> (optional)
    gprsrotimetariffswitchinterval <integer> (optional)
```

applyChargingReportAckGprs

This operation has no parameters.

```
applyChargingReportAckGprs
```

assistrequestinstructions

This operation is used by the SSF to report a specific charging event to the SCF in response to the `ApplyCharging` operation. It has the following format:

```
assistrequestinstructions
[correlationid <digits>]
```

The message generated by this operation causes the event that contains it to be sent automatically as the first event on a new assisting dialog.

Although you can include the `correlationid` parameter in the script, it is ignored and overwritten with the value from the most recently received `EstablishTemporaryConnectioin` operation.

callinformationreport

This operation sends specific call information to the SCF as requested by a previous `callinformationrequest` operation. This operation has the following format:

```
callinformationreport [<parameters>]
```

A `callinformationreport` operation should have one or more of the following parameters, appearing in any order, matching the information requested in the relevant `callinformationrequest` operation:

```
callattemptelapsedtime <integer expression>
callstoptime <digits>
callconnectedelapsedtime <integer expression>
calledaddress <number expression>
releasecause <cause expression>
```

The `callattemptelapsedtime` parameter is measured in seconds while the `callconnectedelapsedtime` parameter is measured in deciseconds. The `callstoptime` parameter is a string in the format: YYMMDDHHMMSS.

cap4InitiateCallAttemptResult

This operation sends a response to a `cap4InitiateCallAtempt` request and has the following format:

```
cap4InitiateCallAttemptResult [<parameters>]
```

A `cap4IniateCallAttemptResult` operation can have one or more of the following parameters, appearing in any order:

```
offeredCamel4Functionalities <integer expression>
supportedCamelPhases <integer expression>
releaseCallArgExtensionAllowed
```

The `offeredCamel4Functionalities` and `supportedCamelPhases` parameters are 16-bit string values.

collecteduserinformation

This operation has the following format:

```
collecteduserinformation
digits <digits>
```

This is not a distinct operation. It represents the result form of the INAP operation, `promptAndCollectUserOperation`. The `digits` parameter corresponds to the `digitsResponse` tag in the result.

entityReleasedGprs

Use this operation when the GPRS session is detached or a PDP context is disconnected and the related event is not equipped for reporting. This operation has the following format:

```
entityReleasedGprs
gprsReleaseCause <integer>
[ gprsPdPid <integer> ]
```

entityReleasedAckGprs

This operation has no parameters. It is the returned result for `entityReleasedGprs`.

error

An `error` operation has the following format:

```
error <name> [ invokeId <invoke-ID> ]
```

An `error` operation generates a U-ERROR component in the outgoing message. The `name` parameter determines the error code used. The following values are valid:

```
cancelled
cancelfailed
etcfailed
improper callerresponse
missingcustomerrecord
missingparameter
parameteroutofrange
requestedinfoerror
systemfailure
taskrefused
unavailableresource
unexpectedcomponentsequence
unexpecteddatavalue
unexpectedparameter
unknownlegid
```

Some errors would typically have additional error codes, but the **slpit** utility supports only the ones listed here.

The `<invoke-ID>` value is from the last received INVOKE component, unless you specifically define it with the `invokeId` parameter.

eventreportbcsm

This operation notifies the SCF of a call-related event that was requested by the SCF in a previous `RequtestReportBCSMEvent` operation. Examples of call-related events are busy and no answer. This operation has the following format:

```
eventreportbcsm [<event>...]
```

The `event` parameter has the following format:

```
eventtypebcsm <type>
[misccallinfo <misccallinfo> | monitormode <mode>]
[legid <legid> | ( <integer> )]
[eventspecificinfo <info>]
```

The <mode> parameter has one of the following values:

```
interrupted
notifyAndContinue
transparent
```

The <legid> parameter has one of the following values:

```
[sendingsideid] <legtype>
[receivingsideid] <legtype>
```

The <legtype> parameter has one of the following values:

```
ltleg1
ltleg2
```

Event-specific information includes the following:

```
busycause <cause>
releasecause <cause>
failurecause <cause>
calledpartynumber <number>
```

eventReportGprs

This operation notifies the GSM SCF of a GPRS session or PDP context related events:

```
eventReportGprs
gprsEventType <number>
[ gprsPdPid <integer> ]
```

eventReportAckGprs

This operation has no parameters.

eventReportSms

This operation notifies the GSM service control function (gsmSCF) of a previously requested short message related event. This message has no parameters.

informServiceCentre

This operation is required for SMS gateway procedures between MSC and HLR. This message has the following format:

```
informServiceCentre [<parameters>]
```

The parameters consist of the following values:

```
storedMSISDN <number expression>
```

initialdp

This operation is used after a trigger detection point (TDP) to issue a request for service. This message has the following format:

```
initialdp [<parameters>]
```

An `initialdp` message can have any of the following parameters, specified in any order:

```
calledpartynumber <number expression>
originalcalledpartynumber <number expression>
callingpartynumber <number expression>
redirectingpartynumber <number expression>
locationnumber <number expression>
additionalcallingpartynumber <number expression>
callingpartyscategory <categoryvalue or number>
callingpartyspin <digits>
origredirreason <integer> redirindicator <integer>
eventtypebcsm <type>
appcontext <string>
extension <integer> <type> <integer> <digits> <integer>
extension <integer> <type> <integer> <digits>
idp_sccp_orig_pc <integer> // 0 - 65535
idp_sccp_orig_ssn <integer> // 0 - 255
idp_sccp_orig_tt <integer> // 0 - 255
idp_sccp_orig_np <integer> // 0 - 15
idp_sccp_orig_noa <integer> // 0 - 127
idp_sccp_orig_rti <integer> // 0 or 1
idp_sccp_orig_digits <digits>
idp_sccp_dest_pc <integer> // 0 - 65535
idp_sccp_dest_ssn <integer> // 0 - 255
idp_sccp_dest_tt <integer> // 0 - 255
idp_sccp_dest_np <integer> // 0 - 15
idp_sccp_dest_noa <integer> // 0 - 127
idp_sccp_dest_rti <integer> // 0 or 1
idp_sccp_dest_digits <digits>
```

Note:

Use the `DISTINCT` or `MULTIPLE` option with `RANGE` for `initialdp callingpartynumber` messages only. See "initialdp".

The following parameter is available for UCP handling:

```
AspID <string>
```

The following parameters are available for use with INAP CAMEL extensions:

```
iMSI <bcd>
countryCode <digits> networkCode <digits> locationAreaCode <integer> [ cellID
<integer> ]
[ bearerCapCodingStandard <number> bearerCapITC <number> [ bearerCapTransferMode
<number> bearerCapITR <number> bearerCapUIProtol <number> ]]
hlCharacteristicsId <number>
calledPartyBCDNumber [ ( <integer> ) ] "number" [ numberPlan <integer> ]
vlrNumber [ ( <integer> ) ] "number" [ numberPlan <integer> ]
ageoflocationinfo <digits>
subscriberstate <digits>
locationnumberlocationinfo [ ( <integer> ) ] "number" [ numberPlan <integer> ]
extBearerService <hex digits>
extTeleService <hex digits>
callreference <string>
[ callForwardingSSPending ]
iMEI <bcd>
```

```
[ Eutrancgimcc <digits> Eutrancgiimnc <digits> Eutrancgi <integer> ]
[Taidmcc <digits> Taidmnc <digits> Taid <integer> ]
```

The `countryCode` and `networkCode` values can be only three digits long.

The `Eutrancgimcc`, `Eutrancgimnc`, `Taidmcc`, and `Taidmnc` values can be only three digits long.

Bearer capability fields are optional and are divided in two stages as shown above. If the second stage is not present, the following default values are assigned: `bearerCapTransferMode` = `BC_TM_CIRCUIT` (0x0), `bearerCapITR` = `BC_ITR_64_KBIT_S` (0x10) and `bearerCapUIProtol` = `BC_UIL1_NOT_PRESENT` (0xff)

The following values are available for Bearer Capability fields:

Table 2-7 bearerCapcodingStandard Bearer Capability Values

Constants	Value
BC_CS_ITU_T	0x00
BC_CS_ISO_IEC	0x01
BC_CS_NATIONAL	0x02
BC_CS_NETWORK	0x03

Table 2-8 bearerCapITC Bearer Capability Values

Constants	Value
BC_ITC_SPEECH	0x00
BC_ITC_UDI	0x08
BC_ITC_RDI	0x09
BC_ITC_3_1_KHZ_AUDIO	0x10
BC_ITC_UDI_TA	0x11
BC_ITC_7_KHZ_AUDIO	0x11
BC_ITC_VIDEO	0x18

Table 2-9 bearerCapTransferMode Bearer Capability Values

Constants	Value
BC_TM_CIRCUIT	0x0
BC_TM_PACKET	0x2

Table 2-10 bearerCapITR Bearer Capability Values

Constants	Value
BC_UIL1_ITU_V110_I460_X30	0x01
BC_UIL1_G711_U_LAW	0x02
BC_UIL1_G711_A_LAW	0x03
BC_UIL1_G721_32_KBIT_S	0x04
BC_UIL1_H221_H242	0x05
BC_UIL1_H223_H245	0x06

Table 2-10 (Cont.) bearerCapITR Bearer Capability Values

Constants	Value
BC_UIL1_NON_ITU_SRA	0x07
BC_UIL1_ITU_V120	0x08
BC_UIL1_X31_HDLC	0x09
BC_UIL1_NOT_PRESENT	0xff

Table 2-11 bearerCapUIProtol Bearer Capability Values

Constants	Value
BC_UIL1_ITU_V110_I460_X30	0x01
BC_UIL1_G711_U_LAW	0x02
BC_UIL1_G711_A_LAW	0x03
BC_UIL1_G721_32_KBIT_S	0x04
BC_UIL1_H221_H242	0x05
BC_UIL1_H223_H245	0x06
BC_UIL1_NON_ITU_SRA	0x07
BC_UIL1_ITU_V120	0x08
BC_UIL1_X31_HDLC	0x09
BC_UIL1_NOT_PRESENT	0xff

InitialDpGprs

When a trigger is detected at a detection point in the general GPRS state machines, this operation requests instructions from the GSM SCF. This message has the following format:

```
initialDpGprs
gprsEventType <integer>
gprsMsisdn <number expression>
gprsImsi <number expression>
gprsOriginatingReferenceNumber <number expression>
[ gprsEndUserAddress <PdpTypeOrganisation> <PdpTypeNumber> [ <address byte> ] ]
[ gprsrequestedqos <qos-info> ]
[ gprssubscribedqos <qos-info> ]
[ gprsnegotiatedqos <qos-info> ]
[ gprsaccesspointname <string> ]
[ gprschargingid <integer> ]
[ gprslocationinformation
gprsmobilecountrycode <bcd>
gprsmobilenetworkcode <bcd>
gprsmobilelocationareacOde <bcd>
gprscellidentity <integer>
gprspdpinitiationtype <integer> ]
[ gprsggsnaddress <integer> [ <integer> ] ]
[ sgsnNumber <number expression> ]
```

The `<qos-info>` variable can have one of the following parameter values, all of which are integers:

Table 2-12 InitialDpGprs

Parameter	Min	Max
gprsqosprioritylevel	0	255
gprsqosdeloferrsdu	0	7
gprsqosdelorder	0	3
gprsqostrafficclass	0	7
gprsqosmaxsdusize	0	255
gprsqosmaxbrforuplink	0	255
gprsqosmaxbrfordownlink	0	255
gprsqossduerrratio	0	15
gprsqosresidualber	0	15
gprsqostrafhlingpri	0	3
gprsqostransferdelay	0	63
gprsqosguabrforuplink	0	255
gprsqosguabrfordownlink	0	255

InitialDpSms

After it detects a TDP-R, the SMS SSF uses this operation to request instructions from the GSM SCF to complete the short-message submission to the SMSC or the short message delivery to the served subscriber. This message has the following format:

```
initialDpSms
[<parameters>]
```

An `initialDpSms` message can have the following parameters in any order:

```
callingPartyNumber <number expression>
destinationSubscriberNumber <number expression>
idp_sccp_orig_pc <integer> // 0 - 65535
idp_sccp_orig_ssn <integer> // 0 - 255
idp_sccp_orig_tt <integer> // 0 - 255
idp_sccp_orig_np <integer> // 0 - 15
idp_sccp_orig_noa <integer> // 0 - 127
idp_sccp_orig_rti <integer> // 0 or 1
idp_sccp_orig_digits <digits>
idp_sccp_dest_pc <integer> // 0 - 65535
idp_sccp_dest_ssn <integer> // 0 - 255
idp_sccp_dest_tt <integer> // 0 - 255
idp_sccp_dest_np <integer> // 0 - 15
idp_sccp_dest_noa <integer> // 0 - 127
idp_sccp_dest_rti <integer> // 0 or 1
idp_sccp_dest_digits <digits>
vlrNumber [ ( <integer> ) ] "number" [ numberPlan <integer> ]
countryCode <digits> networkCode <digits> locationAreaCode <integer> [ cellID
<integer> ]
mscAddr <digits>
smscAddr <digits>
```

mergeCallSegmentResult

This message has the following format with no parameters:

```
mergeCallSegmentResult
```

moForwardSm

This service forwards mobile-originated short messages between the serving mobile switching center (MSC) or the SGSN and the SMS internetworking MSC. This message has the following format:

```
moForwardSm [<parameters>]
```

For MAP version 3, this is a mobile-originated Forward-SM message, which is distinct from the `mtForwardSm` message. For MAP versions 1 and 2, this operation is a Forward-SM and it can originate or terminate from a mobile device, depending on the type of PDU in the SM-RP-UI.

A Forward-SM message has the following parameters in any order:

```
MapVersion <integer>

SegmentedBegin
SegmentedBody

imsiOA <number expression>  //optional, only valid for MAP version 3
privateExtension <comma separated object ID string> <integer ASN.1 tag> <hex
value> // optional, only valid for MAP version 3
countryCode <digits> networkCode <digits> locationAreaCode <integer> cellID
<integer>  // optional, valid for MAP versions 2 and 3
```

The `countryCode`, `networkCode`, `locationAreaCode`, and `cellID` parameters are used to construct the global cell ID. The `countryCode` and `networkCode` values can be only three digits long.

The `SM_RP_DA` field can be service center (MO) on an IMSI (MT):

```
imsi <number expression> (optional for MAP v2/v3 segmented body)
lmsi <number expression> (optional)
ServiceCentreAddressDA <number expression>
```

In a MAP version 2 or 3 segmented MT message, the `imsi` parameter is omitted in segments after the initial segment. If it is omitted, the MT message is encoded with the `noSM_RP_DA` parameter set.

The SM_RP_OA field can be an MSISDN (MO) or a service center (MT):

```
msisdn <number expression>
ServiceCentreAddressOA <number expression> (optional for MAP v2/v3 segmented
body)
```

In a MAP version 2 or 3 segmented MT message, the originating service center is omitted in segments subsequent to the initial segment. If it is omitted, the MT message is encoded with the `noSM_RP_OA` parameter set.

Depending on the MAP version, the SM_RP_UI field can contain one of the following PDUs:

Table 2-13 PDUs

PDU	MAP Version	Comment
SMS-SUBMIT	1, 2, and 3	NA
SMS-DELIVER	1 and 2	In MT-ForwardSM for version 3
SMS-STATUS-REPORT	2	In MT-ForwardSM for version 3)

The type of PDU is determined by the message type indicator, TP-MTI:

```
TP_MTI <number>
```

For an SMS-SUBMIT PDU, with TP-MTI=1, the following parameters are available:

```
TP_VPF <number>
replyPath
requestStatusReport
TP_MR <number>
TP_DA [ToN] <string> | TP_DA <number expression>
TP_DCS <number>
TP_VP { <1 or 7 octets (numbers)> }
userDataHeader { <number> <number> ... }
userDataText <string>
```

> **Note:**
>
> For TP_DA, alphabetic characters (non-telephony digits) are allowed only if ToN = 5 (alphanumeric).

For an SMS-DELIVER (TP-MTI=0), the following parameters are available:

```
moreMessages <0-1>
replyPath
TP_OA [ToN] <string>
TP_DCS <number>
userDataHeader { <number> <number> ... }
userDataText <string>
```

> **Note:**
>
> For TP_OA, alphabetic characters (non-telephony digits) are allowed only if ToN=5 (alphanumeric).

For an SMS-STATUS-REPORT (TP-MTI=2), the following parameters are available:

```
moreMessages <0-1>
TP_MR <number>
TP_RA [ToN] <string>
TP_DCS <number>
userDataHeader { <number> <number> ... }
userDataText <string>
```

> **Note:**
>
> For TP_RA, alphabetic characters (non-telephony digits) are allowed only if ToN=5 (alphanumeric).

Table 2-14 Min and Max for Parameters

Parameter	Value	Min	Max
sccp_orig_pc	<integer>	0	65535
sccp_orig_ssn	<integer>	0	255
sccp_orig_tt	<integer>	0	255
sccp_orig_np	<integer>	0	15
sccp_orig_noa	<integer>	0	127
sccp_orig_rti	<integer>	0	1
sccp_orig_digits	<digits>	0	15
sccp_dest_pc	<integer>	0	65535
sccp_dest_ssn	<integer>	0	255
sccp_dest_tt	<integer>	0	255
sccp_dest_np	<integer>	0	15
sccp_dest_noa	<integer>	0	127
sccp_dest_rti	<integer>	0	1
sccp_dest_digits	<digits>	0	15

There must be exactly one each of `imsi`, `lmsi`, `ServiceCentreAddressDA`, and `noSM_RP_DA`. There must be exactly one of `msisdn`, `ServiceCentreAddressOA`, `noSM_RP_OA`, and `imsiOA`. You can use the `imsiOA` parameter only for MAP3.

If the `SegmentedBegin` parameter is present, the only other parameters allowed are `MapVersion` and `IMSI`. The result is that a TCAP_BEGIN message is sent with the appropriate application context but with no component (the User Information part of the TCAP_BEGIN message contains a MAP-OPEN with an optional `IMSI` in it.) If the `IMSI` parameter is present in the `SegmentedBegin`, the RP-DA in the ForwardSM should be a `LMSI`, but this is not enforced by the **slpit** utility.

If the `SegmentedBody` parameter is present, a normal `moForwardSM` operation is sent (in a `Continue`) but with no application context. You must always pair `SegmentedBegin` and `SegmentedBody` operations with appropriate MAP versions and a `receive{}`message between them.

The following segmentation scenarios are valid.

- `IMSI` and `LMSI` parameters in segmented message:

Table 2-15 IMSI and LMSI Parameters Segmentation Scenario

Primitive	MAP-OPEN	ForwardSM RP-DA	Segment
TCAP_BEGIN	imsi	N/A	Begin
TCAP_CONTINUE	N/A	lmsi	Body #1
TCAP_CONTINUE	N/A	noSM-RP-DA	Body #2

- `IMSI` only in a segmented message:

Table 2-16 IMSI-only Parameters Segmentation Scenario

Primitive	MAP-OPEN	ForwardSM RP-DA	Segment
TCAP_BEGIN	empty	N/A	Begin
TCAP_CONTINUE	N/A	lmsi	Body #1
TCAP_CONTINUE	N/A	noSM-RP-DA	Body #2

- `IMSI` and `LMSI` in a non-segmented message:

Table 2-17 IMSI and LMSI Non-Segmentation Scenario

Primitive	MAP-OPEN	ForwardSM RP-DA	Segment
TCAP_BEGIN	imsi	imsi	N/A

- `IMSI` only in a non-segmented message:

Table 2-18 IMSI-only Non-Segmentation Scenario

Primitive	MAP-OPEN	ForwardSM RP-DA	Segment
TCAP_BEGIN	empty	imsi	N/A

You may specify either the `userDataheader` or the `userDatatext` or both. You must specify the header byte by byte, and in decimal or hex (with 0x as a prefix) – for example, `userDataHeader {0x17 0x34}`. The header is automatically prefixed with a one-byte length field.

The `userDatatext` parameter will be added to the packet after the `userDataheader` parameter in either GSM 7-bit (default) or Unicode UCS2/UTF16 (big endian, meaning the most significant bytes in multi-byte data types are stored first) or binary, depending on the value of the data coding scheme TP_DCS.

The **slpit** utility does not support compressed user data.

You can specify TP_VF (validity period format) and TP_VP (validity period). See GSM 03.40 v7.5.0 sections 9.2.3.3 and 9.2.3.12 for encoding details. For example:

- tp_vf 0

  VPF of 0, or not specified, means no validity period format.

- tp_vf 1

  Enhanced format (new to MAP version 3) tp_vp {0x42 0x80 0x00 0x00 0x00 0x00 0x00}. Relative; 128 (0x80) seconds and single-shot=true.

- tp_vf 2

  Relative format tp_vf { 128 }, where 128 is decimal and means 645 minutes.

- tp_vpf 3

  Absolute format tp_vp { 0x40 0x50 0x32 0x61 0x10 0x20 0x00}. 2004-05-23 16:01:02 GMT.

mtForwardSM

This operation is a MAP version 3 mobile-terminated `Forward-SM` and is available for use with MAP. It forwards mobile terminated short messages between the gateway mobile switching center (MSC) and the servicing MSC or the SGSN.

```
mtForwardSm [<parameters>]
```

An `mtForwardSm` operation should have the following parameters, appearing in any order:

```
MapVersion <integer>  // only 3 is valid
SegmentedBegin
SegmentedBody
```

The `SM_RP_DA` field must be an IMSI number for `MT-ForwardSM` if the message is not getting segmented. If the message is segmented, the `SM_RP_DA` can be a LMSI, in which case the `segmentedBegin` should contain the IMSI:

```
imsi <number expression>   // optional in the SegmentedBegin
lmsi <number expression>   // optional in the SegmentedBody
```

In a segmented `MT-ForwardSM` operation, the IMSI is omitted in segments following the initial segment. If it is omitted, the operation is encoded with the `noSM_RP_DA` parameter set.

The SM_RP_OA field must be a service center for MT-ForwardSM:

```
ServiceCentreAddressOA <number expression> (optional)
```

In a segmented `MT-ForwardSM` operation, the originating service center is omitted in segments following the initial segment. If it is omitted, the operation is encoded with the `noSM_RP_OA` parameter set.

The SM_RP_UI field must contain an `SMS-DELIVER` or an `SMS-STATUS-REPORT` for `MT-ForwardSM`. For more information, see "moForwardSm".

```
sccp_orig_pc <integer> // 0 - 65535
sccp_orig_ssn <integer> // 0 - 255
sccp_orig_tt <integer> // 0 - 255
sccp_orig_np <integer> // 0 - 15
sccp_orig_noa <integer> // 0 - 127
sccp_orig_rti <integer> // 0 or 1
sccp_orig_digits <digits>
sccp_dest_pc <integer> // 0 - 65535
sccp_dest_ssn <integer> // 0 - 255
sccp_dest_tt <integer> // 0 - 255
sccp_dest_np <integer> // 0 - 15
sccp_dest_noa <integer> // 0 - 127
sccp_dest_rti <integer> // 0 or 1
sccp_dest_digits <digits>
```

If `SegmentedBegin` is present, the only other parameters allowed are `MapVersion` and IMSI. This results in a `TCAP_BEGIN` being sent with the appropriate application context but no component (the User Information part of the `TC_BEGIN` contains a `MAP-OPEN` with an optional IMSI in it). If the IMSI is present in the `SegmentedBegin`, the `RP-DA` in the `ForwardSM` should be a LMSI, but this is not enforced by the **slpit** utility.

If `SegmentedBody` is present, a normal `moForwardSM` is sent in a `CONTINUE` but with no application context. You are responsible for always pairing `SegmentedBegin` and `SegmentedBody` operations, with matching `MapVersions` and a `receive{}` message between them.

See the segmentation scenarios in the "moForwardSm" section for more information.

You may specify either the `userData` header or the text or both. You must specify the header byte by byte in decimal or hex (with a 0x prefix) – for example, `userData { 0x02 0x17 0x34}`. The first number in the user data header should be the length, in bytes, of the remainder of the user data header. In this case, `0x02` indicates that there are two more bytes to follow in the header.

Any user data text will be copied after the user data header, with bit padding inserted to align to a septet boundary.

readyForSM

This operation is available for MAP handling. It is used between the message switching center (MSC) and the VLR and between the VLR and the HLR. If a subscriber has available memory, the MSC initiates this service and the VLR indicates this condition to the HLR. If a subscriber, whose message waiting flag is active in the VLR, has radio contact in the MSC, the VLR initiates this service.Likewise, if a subscriber has available memory, the SGSN initiates this service to indicate this to the HLR. Also, if a subscriber whose message waiting flag is active in the SGSN has radio contact in the GPRS, the SGSN initiates this service.

```
readyForSM [<parameters>]
```

A `readyForSM` operation should have the following parameters, appearing in any order.

```
imsi <bcd>
alertReason <alertReason> | <number>
```

The `<alertReason>` value can be one of the following:

```
ms_Present
memoryAvailable
```

reportSMDeliveryStatus

This operation is available for MAP handling. It is used by the message switching center (MSC) to set the message waiting data into the HLR or to inform the SLR of a successful short message (SM) transfer after polling.

```
reportSMDeliveryStatus [<parameters>]
```

A `reportSMDeliveryStatus` operation should have the following parameters, appearing in any order.

```
msisdn <number expression>
serviceCentreAddress <number expression>
smDeliveryOutcome <smDeliveryOutcome> | <number>
```

The <smDeliveryOutcome> value can be one of the following:

```
memoryCapacityExceeded
absentSubscriber
successfulTransfer
```

sendRoutingInfoForSm

This operation is for MAP handling and has the following format.

```
sendRoutingInfoForSm [<parameters>]
```

A sendRoutingInfoForSm operation should have the following parameters, appearing in any order.

```
MapVersion <number>
msisdn <number expression>
AttemptDelivery <0-1>
ServiceCentreAddress <number expression>
(optional parameters)
MessageTypeIndicator <number>
OriginatingSmeAddr <number expression>
GprsSupport <0-1>
sccp_orig_pc <integer> // 0 - 65535
sccp_orig_ssn <integer> // 0 - 255
sccp_orig_tt <integer> // 0 - 255
sccp_orig_np <integer> // 0 - 15
sccp_orig_noa <integer> // 0 - 127
sccp_orig_rti <integer> // 0 or 1
sccp_orig_digits <digits>
sccp_dest_pc <integer> // 0 - 65535
sccp_dest_ssn <integer> // 0 - 255
sccp_dest_tt <integer> // 0 - 255
sccp_dest_np <integer> // 0 - 15
sccp_dest_noa <integer> // 0 - 127
sccp_dest_rti <integer> // 0 or 1
sccp_dest_digits <digits>
```

sendRoutingInformation

This operation is available for MAP handling and has the following format:

```
sendRoutingInformation [<parameters>]
```

A sendRoutingInformation operation should have the following parameters, appearing in any order:

```
interrogationType <0-1>
gmscAddress <number expression>
msisdn <number expression>
(optional parameters)
sccp_orig_pc <integer> // 0 - 65535
sccp_orig_ssn <integer> // 0 - 255
sccp_orig_tt <integer> // 0 - 255
sccp_orig_np <integer> // 0 - 15
sccp_orig_noa <integer> // 0 - 127
sccp_orig_rti <integer> // 0 or 1
sccp_orig_digits <digits>
sccp_dest_pc <integer> // 0 - 65535
sccp_dest_ssn <integer> // 0 - 255
sccp_dest_tt <integer> // 0 - 255
sccp_dest_np <integer> // 0 - 15
```

```
sccp_dest_noa <integer> // 0 - 127
sccp_dest_rti <integer> // 0 or 1
sccp_dest_digits <digits>
```

An `interrogationType` value of 0 indicates a basic call; a value of 1 indicates a forwarding call.

smsNotification

This operation is available for IS-41 support. Your application can expect to receive one of these from an MSC when a SME comes back online after sleeping through a direct delivery attempt.

```
smsNotification [<parameters>]
```

An `smsNotification` operation can have the following parameters.

```
smsnot_MIN <bcd-string> // 10 digit number
smsnot_ESN <integer> <integer> // 0..2^8, 0..2^24
```

smsNotificationResult

The response to an `smsNotification` operation does not contain any parameters.

```
smsNotificationResult
```

locationRequest

This operation is available for IS-41 support and has the following format:

```
locationRequest [<parameters>]
```

A `locationRequest` operation has the following parameters:

```
locreq_BID <integer> <integer> <integer> <integer> // 0..2^16, 0..2^8, 0..2^24, 0..2^8
locreq_DIGITS <integer> <integer> <bcd-string>
locreq_MSCID <integer> <integer> // 0..2^16, 0..2^8
locreq_SYSTEMMYTYPECODE <integer>
sccp_orig_pc <integer> // 0 - 65535
sccp_orig_ssn <integer> // 0 - 255
sccp_orig_tt <integer> // 0 - 255
sccp_orig_np <integer> // 0 - 15
sccp_orig_noa <integer> // 0 - 127
sccp_orig_rti <integer> // 0 or 1
sccp_orig_digits <digits>
sccp_dest_pc <integer> // 0 - 65535
sccp_dest_ssn <integer> // 0 - 255
sccp_dest_tt <integer> // 0 - 255
sccp_dest_np <integer> // 0 - 15
sccp_dest_noa <integer> // 0 - 127
sccp_dest_rti <integer> // 0 or 1
sccp_dest_digits <digits>
```

The `locreq_BID` (Billing ID) parameters are: `Market ID`, `Switch Number`, `ID Number`, and `Segment Counter`. The `locreq_DIGITS` parameters are: `Type of Digits`, `Nature of Number`, and `BCD Digits`. The `locreq_MSCID` parameters are: `Market ID`, and `Switch Number`.

The `locreq_SYSTEMMYTYPECODE` parameter sets `VENDOR_IDENTIFIER_`, which is the only content of `locreq_SYSTEMMYTYPECODE`.

You can use the following values for `VENDOR_IDENTIFIER`:

```
VENDOR_IDENTIFIER_NotUsed = 0,
VENDOR_IDENTIFIER_EDS = 1,
VENDOR_IDENTIFIER_Astronet = 2,
VENDOR_IDENTIFIER_LucentTechnologies = 3,
VENDOR_IDENTIFIER_Ericsson = 4,
VENDOR_IDENTIFIER_GTE = 5,
VENDOR_IDENTIFIER_Motorola = 6,
VENDOR_IDENTIFIER_NEC = 7,
VENDOR_IDENTIFIER_NORTEL = 8,
VENDOR_IDENTIFIER_NovAtel = 9,
VENDOR_IDENTIFIER_Plexsys = 10,
VENDOR_IDENTIFIER_DigitalEquipmentCorp = 11,
VENDOR_IDENTIFIER_INET = 12,
VENDOR_IDENTIFIER_Bellcore = 13,
VENDOR_IDENTIFIER_AlcatelSEL = 14,
VENDOR_IDENTIFIER_Tandem = 15,
VENDOR_IDENTIFIER_QUALCOMM = 16,
VENDOR_IDENTIFIER_Aldiscon = 17,
VENDOR_IDENTIFIER_Celcore = 18,
VENDOR_IDENTIFIER_TELOS = 19,
VENDOR_IDENTIFIER_Stanilite = 20,
VENDOR_IDENTIFIER_CoralSystems = 21,
VENDOR_IDENTIFIER_SynacomTechnology = 22,
VENDOR_IDENTIFIER_DSC = 23,
VENDOR_IDENTIFIER_MCI = 24,
VENDOR_IDENTIFIER_NewNet = 25,
VENDOR_IDENTIFIER_SemaGroupTelecoms = 26,
VENDOR_IDENTIFIER_LGInformationAndCommunications = 27,
VENDOR_IDENTIFIER_CBIS = 28,
VENDOR_IDENTIFIER_Siemens = 29
```

locationRequestResult

This is the response to a `locationRequest` operation.

```
locationRequestResult [<parameters>]
```

A `locationRequestResult` operation has the following parameters:

```
smsreq_ESN <integer> <integer> // 0..2^8, 0..2^24
smsreq_MIN <integer> // must be 10 digits
smsreq_MSCID <integer> <integer> // 0..2^16, 0..2^8
```

The `MSCID`, `ESN` and `MIN` parameters are mandatory in the response. However, if the `ESN` is unknown, it is set to "0 0"; if the `MIN` is unknown it is set to "0000000000".

smsRequest

This operation is available for IS-41 support. Your applications should never receive one of these. It is included only to assist in testing scenarios in which the service control point, acting as an SMSC, receives an unexpected operation.

```
smsRequest [<parameters>]
```

A `smsRequest` operation can have the following parameters.

```
smsreq_MIN <bcd-string> // 10 digit number
smsreq_IMSI <number expression>
smsreq_MDN <integer> <integer> <bcd-string>
smsreq_ESN <integer> <integer> // 0..2^8, 0..2^24
smsreq_notificationIndicator <integer> // 0..255
```

```
smsreq_teleserviceIdentifier <integer> // 0..65535
sccp_orig_pc <integer> // 0 - 65535
sccp_orig_ssn <integer> // 0 - 255
sccp_orig_tt <integer> // 0 - 255
sccp_orig_np <integer> // 0 - 15
sccp_orig_noa <integer> // 0 - 127
sccp_orig_rti <integer> // 0 or 1
sccp_orig_digits <digits>
sccp_dest_pc <integer> // 0 - 65535
sccp_dest_ssn <integer> // 0 - 255
sccp_dest_tt <integer> // 0 - 255
sccp_dest_np <integer> // 0 - 15
sccp_dest_noa <integer> // 0 - 127
sccp_dest_rti <integer> // 0 or 1
sccp_dest_digits <digits>
```

The `smsreq_MDN` parameters are: `Type of Digits`, `Nature of Number`, and `BCD Digits`.

smsRequestResult

This is the response to an `smsRequest` operation.

```
smsRequestResult [<parameters>]
```

An `smsRequest` operation can be an `Ack` or a `Nack`. `Acks` contain an address, and can also return `ESN` data. `Nacks` may specify a value for the `accessDeniedReason` parameter.

```
smsreq_ESN <integer> <integer> // 0..2^8, 0..2^24
smsreq_address <integer> <integer> <bcd-string> // NoN, NPI, digits
smsreq_accessDeniedReason <integer> // 0..255
```

smsDeliveryPointToPoint

This operation is available for IS-41 support. You can use this to deliver a short message over IS-41.

```
smsDeliveryPointToPoint [<parameters>]
```

The **slpit** utility supports three text-based teleservices, CDMA 4098, CDMA 4101, and TDMA 32513 and two use cases for the text message. The first is human-readable text with an optional header. The text is encoded as 7-bit ASCII for CDMA or IRA for TDMA. (The actual encoding step is independent of the `smdpp_userDataEncoding` parameter.) You may not specify both text and header for a TDMA message.

The CDMA text plus header has the following format:

tag length encoding num_fields header padding-1 text padding-2

The text and header fields can have the following values:

Table 2-19 smsDeliveryPointToPoint values

Field	Value
tag	0x01
length	The number of octets after this one.
encoding	The first five bits of the `smdpp_userDataEncoding` value. Note that everything after this is shifted 3 bits to the left.

Table 2-19 (Cont.) smsDeliveryPointToPoint values

Field	Value
num_fields	The number of characters (7 or 8-bit) after this octet.
header	Zero or more octets of GSM user-data-header. This is taken directly from the `smdpp_userDataHeader` value.
padding-1	Padding required to make the header end on a septet boundary. This is only done if the encoding is 2 (7-bit ASCII, default) or 3 (IA5).
text	Encoded message text. This will always be 7-bit ASCII.
padding-2	Padding required to make this whole block end on an octet boundary.

The TDMA text has the following format:

length type padding text

The fields can have the following values:

Table 2-20 TDMA values

Field	Value
length	The number of octets following this one.
type	The first 5 bits of `smdpp_userDataEncoding`.
padding	3 bits of padding, so that text starts on an octet boundary.
text	Text as something resembling IRA, with each character 7 bits wide but stored in an octet with the high bit off.

With the second use case for text messaging, you cannot use the human-readable text parameter (`smdpp_userDataText`); you must put the raw bytes of the message into the header (`smdpp_userDataHeader`). The header and text are packed into the message for the different use cases as follows:

The CDMA header only has the following format:

tag length encoding num_fields data padding

These fields have the following values:

Table 2-21 CDMA header values

Field	Value
tag	NA
length	NA
encoding	NA
num_fields	NA
data	The octets specified in `smdpp_userDataHeader` shifted 3 bits to the left.
padding	Empty bits required to bring the block to an octet boundary.

The TDMA header has the following format:

length type padding data

These fields have the following values:

Table 2-22 TDMA header values

Field	Value
length	NA
type	NA
padding	NA
data	The octets specified in the smdpp_userDataHeader

Note that the translation of text from the human-readable input form to ASCII or IRA is not perfect. When in doubt, try using the header to set the raw data.

A `smsDeliveryPointToPoint` may have the following parameters.

Table 2-23 smsDeliveryPointToPoint Parameters

Parameter	Type	Value
smdpp_teleservice	<integer>	4098, 4100, 32513
smdpp_MIN	<bcd-string>	10 digit
smdpp_ESN	<integer>< integer>	$0..2^8$, $0..2^{24}$
smdpp_origAddr	<integer>< integer><bcd-string>	NoN, NPI, digits
smdpp_origOrigAddr	<integer>< integer><bcd-string>	NoN, NPI, digits
smdpp_destAddr	<integer>< integer><bcd-string>	NoN, NPI, digits
smdpp_origDestAddr	<integer>< integer><bcd-string>	NoN, NPI, digits
smdpp_messageCount	<integer>	$0..2^8$
smdpp_notInd	<integer>	$0..2^8$
smdpp_chargeInd	<integer>	$0..2^8$
smdpp_userDataEncoding	<number>	NA
smdpp_userDataHeader	{<number>...}	NA
smdpp_userDataText	<string>	NA
CDMA	{ <parameters> }	NA
TDMA	{ <parameters> }	NA

The `smdpp_teleservice` parameter is mandatory, and must be set according to IS-41-D before the TDMA or CDMA sections can be used. For more information about these parameters, please consult TIA/EIA-41-D-1997 (IS-41), 3GPP2 C.S0015-A (CDMA) and TIA/EIA-136-710-C (TDMA).

Each teleservice may place a particular restriction on the data specified. These restrictions aren't generally enforced by the **slpit** utility, because you might want to send bad data. The following are common restrictions:

- CDMA 4098

No userdata header present.

- CDMA 4100

  Encoding type is 0.

- TDMA 32513

  No userdata header present.

The following lists show the common encoding values:

- CDMA

Table 2-24 CDMA values

Value	Name	Width
0	octet-unspecified	8
1	Extended protocol message	NA
2	7-bit ASCII (default)	7
3	IA5	7
4	UNICODE	16
5	Shift JIS	8 / 16
6	Korean	8 / 16
7	Latin/Hebrew	8
8	Latin	8

- TDMA

Table 2-25 TDMA values

Value	Name	Width
1	IRA	7
2	User specific	8
3	Latin	8
5	Latin/Hebrew	8

If you are using the CDMA teleservices, you can specify the following parameters in the CDMA subsection:

Table 2-26 CDMA parameters

Parameter	Type	Value
smdpp_messageId	<integer><integer><boolean>	0..2^4, 0..2^{16}, true/false
smdpp_validityPeriod	<integer>	See 4.5.6.1 of 3GPP2 CS 15-A
smdpp_validityPeriod	<string>	YYMMDDhhmmss
smdpp_deferredDeliveryTime	<integer>	See 4.5.6.1 of 3GPP2 CS 15-A
smdpp_deferredDeliveryTime	<string>	YYMMDDhhmmss

Table 2-26 (Cont.) CDMA parameters

Parameter	Type	Value
smdpp_priorityInd	<integer>	0..3
smdpp_privacyInd	<integer>	0..3
smdpp_languageInd	<integer>	0..255
smdpp_alertOnDelivery	<integer>	0..3
smdpp_DAKRequested	<boolean>	true/false
smdpp_MAKRequested	<boolean>	true/false
smdpp_RAKRequested	<boolean>	true/false

If you are using the TDMA teleservice, you can specify the following parameters in the TDMA subsection:

Table 2-27 TDMA parameters

Parameter	Type	Value
smdpp_messageTypeInd	<integer>	$0..2^3$
smdpp_messageRef	<integer>	$0..2^{13}$
smdpp_privacyInd	<integer>	$0..2^3$
smdpp_urgencyInd	<integer>	$0..2^2$
smdpp_DAKRequested	<boolean>	true/false
smdpp_MAKRequested	<boolean>	true/false
smdpp_messageUpdating	<boolean>	true/false
smdpp_vp_absolute	<integer>	0..1
smdpp_vp_relativeTimerValue	<integer>	0..255
smdpp_vp_absoluteSeconds	<integer>	$0..2^{32}$
smdpp_vp_absoluteTZOffsetDirection	<integer>	0..1
smdpp_vp_absoluteTZOffsetMinutes	<integer>	0..720
smdpp_vp_absoluteTZOffsetDSI	<integer	0..1

smsDeliveryPointToPointResult

This is the response to an `smsDeliveryPointToPoint` operation.

```
smsDeliveryPointToPointResult [<parameters>]
```

An `smsDeliveryPointToPoint` operation may be an `Ack` or a `Nack`. `Nacks` contain an `SMS_CauseCode` parameter, specified as follows:

```
smdpp_causeCode <integer> // 0..2^8
```

slpitLegResult

This operation is available for INAP level 2 (CS-2) handling.

```
slpitLegResult
```

This operation has no parameters.

specializedresourcereport

This operation has no parameters.

```
specializedresourcereport
```

tcapreject

```
tcapreject
    problemtype <type>
    generalproblem <problem>
```

A `tcapreject` operation will have the **slpit** utility send a `TCAP_REJECT` primitive on the main dialog. The problem type and ID are taken from the parameters. The mandatory parameter is `problemtype` which must be an integer from the following list:

Table 2-28 Integer List

Integer	Value
-1	none
0	general
1	invoke
2	return_result
3	return_error

The `generalproblem` parameter is also an integer, from 0 to 255. The reject source is set to `TCAP_REJECT_LOCAL`.

unstructuredSS

This operation is available for MAP handling. It sends a `MAP2_ProcessUnstructuredSSRequest`. The only language available is the default GSM alphabet because that is the only language that `TC_PROTOS` currently supports.

```
unstructuredSS [<parameters>]
```

An `unstructuredSS` operation can have the following parameters, appearing in any order:

```
msisdn <number expression>
msisdnReference <number expression>
originatingReference <number expression>
destinationReference <number expression>
ussd <string>
iMEI <bcd>
countryCode <digits>
networkCode <digits>
locationAreaCode <integer>
cellID <integer>
sccp_orig_pc <integer> // 0 - 65535
sccp_orig_ssn <integer> // 0 - 255
sccp_orig_tt <integer> // 0 - 255
sccp_orig_np <integer> // 0 - 15
sccp_orig_noa <integer> // 0 - 127
```

```
sccp_orig_rti <integer> // 0 or 1
sccp_orig_digits <digits>
sccp_dest_pc <integer> // 0 - 65535
sccp_dest_ssn <integer> // 0 - 255
sccp_dest_tt <integer> // 0 - 255
sccp_dest_np <integer> // 0 - 15
sccp_dest_noa <integer> // 0 - 127
sccp_dest_rti <integer> // 0 or 1
sccp_dest_digits <digits>
```

The `countryCode` and `networkCode` can be only three digits long.

A `MapOpen` is inserted into the TCAP primitive's `UserInformation` area. The `msidnReference` is used to populate the `msisdnReference` in the `MapOpen`. The `destinationReference` populates the `destinationReference` in the `MapOpen`. The `originatingReference` is used to populate `originationReference` in the `MapOpen` The `msisdn` is used to populate the `msisdn` parameter in the `UnstructuredSSRequest`.

[empty]

The **slpit** utility can also send empty TCAP primitives. You can accomplish this by leaving the body of the `send` message blank as shown in the following example:

```
send
{
}
```

Receive Message Operations

The following operations are available in the `receive` message portion of a call definition.

abort

The format of this operation is:

```
abort
```

Use this operation to receive aborts that you expect. The Calls Aborted count is not updated, but the Calls Succeeded count is updated. When you use this operation, the script expects an abort so the test is successful in that it received one and continues. This is especially useful when running multiple calls because a standard abort would cause **slpit** to stop processing.

activityTest

This operation is available for use with CAMEL Phase 1. It is used to check continued existence of a relationship between SCF and SSF. This operation has no parameters.

```
activityTest
```

anyTimeInterrogation

This operation is available for handling MAP. It is used for time information enquiries between GSM SCF and HLR. It has the following format and parameters:

```
anyTimeInterrogation
    [ locationInformation
    [ age <integer comparator> ]
    [ geographical <number comparator> ]
```

```
    [ vrl <number comparitor> ]
    [ location <number comparator> ]
    [ cell <number comparator> ]
]
```

You can define the parameters to the `locationInformation` part of this operation in any order.

applycharging

This operation has the following format:

```
applycharging
    thresholdtime <integer comparator>
    [warningtime <integer comparator>]
```

The following format is available for handling CAMEL:

```
applyCharging
    maxDuration <integer>
    [ release <integer> tone <integer> ]
    [ tariff <integer> ]
```

applyChargingGprs

This operation is available for GPRS handling and it takes one of the following two forms:

```
applyChargingGprs
    gprsTransferredVolume <integer>
    [ gprstariffswitchinterval <integer> ]
    [ gprsPdPid <integer> ]

applyChargingGprs
    gprsElapsedTime <integer>
    [ gprstariffswitchinterval <integer> ]
    [ gprsPdPid <integer> ]
```

callinformationrequest

This operation requests the SSF to record information about a call and use the `CallInformationReport` operation to report it to the SCF. This operation has the following format:

```
callinformationrequest <requested fields>
```

A `callinformationrequest` operation must have at least one of the following labels requesting particular information. They may appear in any order:

```
callattemptelapsedtime
callstoptime
callconnectedelapsedtime
calledaddress
releasecause
```

These are effectively flags and do not have any associated values.

cap4InitiateCallAttempt

This operation requests the SSF to initiate a CAP4 call attempt and returns the result in a `cap4InitiateCallAttemptResult` message. It has the following format:

```
cap4InitiateCallAttempt [<parameters>]
```

A `cap4InitiateCallAttempt` operation may have any of the following parameters, in any order:

```
callingpartnumber <number comparator>
callreference <number comparator>
callsegmentid <integer>
destroutingaddr <number comparator>
gsmscf <number comparator>
legid <integer>
suppresstcsi
```

collectinformation

This operation requests the SSF to perform the call processing actions that collect destination information from a calling party. This operation has the following format, with no parameters.

```
collectinformation
```

connect

This operation requests the SSF to route a call to its destination. It has the following format:

```
connect [<parameters>]
```

The `connect` operation can have any of the following parameters in any order:

```
originalcalledpartnumber <number comparator>
    callingpartynumber <number comparator>
    redirectingpartynumber <number comparator>
    redircount <integer> redirreason <integer>
    destroutingaddr <number comparator>
    callingpartyscategory <integer> | callingpartyscategory <category>
    genericnumbers numberqualifier <qualifier> <number comparator>
...numberqualifier <qualifier> <number comparator>
```

You must include the `destroutingaddr` parameter. The `<category>` parameter is one of:

```
unknowncategory
operatorfrench
operatorenglish
operatorgerman
operatorrussian
operatorspanish
ordinarycallingsubscriber
callingsubscriberwithpriority
datacall
testcall
payphone
```

The `<qualifier>` parameter is either an integer value or one of the following:

```
additionalCalledNumber
additionalConnectedNumber
additionalCallingNumber
additionalOriginalCalledNumber
additionalRedirectingNumber
additionalRedirectionNumber
```

connectGprs

This operation is available for GPRS handling. When establishing a PDP context, it modifies the Access Point Name. This operation has the following format:

```
connectGprs
gprsAccessPointName <string>
[ gprsPdPid <integer> ]
```

connecttoresource

On receipt from the GSM SCF, this operation connects the IP to the incoming call. This operation has the following format:

```
connecttoresource
[ legid <integer> ]
[ address <digits> ]
```

If no address is specified, the received operation must have `none` indicated for its `resourceAddress` tag. If an address is specified, it must match the address in the `ipRoutingAddress` tag.

continue

This operation requests the SSF to proceed with call processing at the detection point (DP) where it previously suspended call processing to wait for instructions from the SCF. This operation has the following format and no parameters:

```
continue
```

continueGprs

This operation is available for GPRS handling. It requests the GPRS SSF to proceed with the GPRS session or context processing at the detection point (DP) where it previously suspended processing to wait for instructions from the GSM SCF. It has the following format:

```
continueGprs [ gprsPdPid <integer> ]
```

The release cause must be between 0 and 255 inclusive.

continueSms

This operation requests the SMS SSF to proceed with processing at the detection point (DP) where it previously suspended processing to wait for instructions from the GSM SCF. It is available for CAMEL handling and it has no parameters.

```
continueSms
```

continuewithargument

This operation requests the GSM SSF to proceed with call processing at the detection point (DP) at which it previously suspended call processing to wait for instructions from the GSM service control function. It also provides additional service-related information to the called party or the calling party while call processing proceeds.Parameters that are provided in the operation replace the corresponding signalling parameters in the call control function (CCF) and are used in subsequent call processing. Parameters that are not replaced by the operation retain their value in the CCF for subsequent call

processing. This operation is available for INAP level 2 (CS-2) handling and has the following format:

```
continuewithargument [<parameters>]
```

The `continuewithargument` operation can have the following parameters:

```
legId <integer>
cap4CallSegmentId
cap4LegId <integer>
```

Use either the `legId` parameter or use the `cap4CallSegmentId` and the `cap4LegId` parameters. For CAP4 protocols, use the `cap4CallSegmentId` and the `cap4LegId` parameters. The `legId` parameter is not valid for CAP4 protocols.

cs1InitiateCallAttempt

This operation requests the SSF to initiate a CS1 call attempt. Although it can produce errors, it has no returned result. It has the following format:

```
cs1IniateCallAttempt [<parameters>]
```

A `cs1InitiateCallAttempt` operation can have any of the following parameters, in any order:

```
callingpartynumber ,number comparator.
callingpartyscategory <integer> | callingpartyscategory <category>
destroutingaddr <number comparator>
```

disconnectforwardconnection

This operation is used in two cases: 1) To disconnect a connection to a specialized resource function (SRF) and 2) to clear a connection to an assisting SSF. In the first case, it disconnects a forward connection from the SSF. In the second case, it disconnects the temporary connection between the initiating SSF and the assisting SSF and between the assisting SSF and its associated SRF. The operation has the following format with no parameters:

```
disconnectforwardconnection
```

disconnectforwardconnectionwithargument

This operation is available for INAP level 2 (CS-2) handling.

```
disconnectforwardconnectionwithargument partytodisconnect legid <integer>
```

The `partytodisconnect` parameter with the `legid` variant is the only supported parameter for this operation.

disconnectleg

This operation is available for INAP level 2 (CS-2) handling. It requests the GSM SSF to release a leg associated with the call. Other legs are retained.

```
disconnectleg reason <cause comparator> [legid <legid> | ( <integer> )]
```

Please see the description of "releasecall" for the possible values for the `reason` parameter. See the description of "eventreportbcsm" for a description of the possible values for `legid`.

establishtemporaryconnection

This operation creates a connection to a resource for a limited period of time to play an announcement or collect information and so on. It has the following format and parameter:

```
establishtemporaryconnection address <digits>
```

The `address` parameter, which is a string in double quotes, is mandatory.

furnishcharginginformation

This operation requests the SSF to generate or register a call record or to include some information in the default call record.This operation has no parameters:

```
furnishcharginginformation
```

> **Note:**
>
> Increasing the output level causes the parameters of the received operation to be written out even though the **slpit** utility does not check them.

mergecallsegments

This operation is available for INAP level 2 (CS-2) handling. It has the following format and mandatory parameters

```
mergecallsegments sourcecallsegment <integer> targetcallsegment <integer>
```

moForwardSmResult

This operation is available for MAP handling. It has no parameters:

```
moForardSmResult
```

mtForwardSmResult

This operation is available for MAP handling. It has no parameters:

```
mtForwardSmResult
```

playannouncement

This operation is used for in-band interaction with an analog user or for interaction with an Integrated Services Digital Network (ISDN) user. It has the following format and parameters:

```
playannouncement
     [connectedparty <integer>]
     annid <integer comparator> [, <integer comparator>...]
     [variableparts <parts>]
```

If you include the `variableparts` parameter, you must include one or more of the part value specifiers:

```
price <integer> <integer>
digits <digits>
```

```
time <integer> <integer>
date <integer> <integer> <integer>
integer <integer>
```

These correspond to the obvious sub-tags in the `variableparts` parameter of the outgoing operation. The `price` specifier should have two integer parameters giving dollars (or big currency unit) and cents (or little currency unit) respectively. You specify the `time` value in hours and minutes and the `date` value as day of the month, month number (1-12), and year (0-99).

prearrangedend

This operation has no parameters. It expects the other side of the dialog to send the fake TCAP primitive `TCAP_PRE_END`.

```
prearrangedend
```

promptandcollectuserinformation

This operation interacts with a user to collect information. It has the following format and parameters:

```
promptandcollectuserinformation
     annid <integer comparator> [, <integer comparator>...]
     [minnumberofdigits <integer>]
     maxnumberofdigits <integer>
     [<digit parameter>...]
     [variableparts <part> [<part>...]]
```

You can use the following specifiers for `digit parameter` in any order:

```
endofreplydigit <digits>
canceldigit <digits>
startdigit <digits>
interdigittimeout <integer>
firstdigittimeout <integer>
```

If you include the `variableparts` parameter, you must specify one or more of the following `part` value specifiers. See the description of "playannouncement" for more information.

readyForSMResult

This operation is available for MAP handling. It has no parameters:

```
readyForSMResult
```

releasecall

This operation causes the SCF to terminate an existing call at any phase for all parties. It has the following format and one parameter:

```
releasecall reason <cause comparator>
```

You can specify an integer for the `reason` parameter or one of the following names:

```
unalloc_num
norm_call_clr
user_busy
no_user_resp
no_answer
call_rejected
```

```
num_changed
out_of_order
inval_num_fmt
normal
temp_failure
```

releaseGprs

This operation causes the GSM service control function (SCF) to terminate an existing GPRS Session or PDP context at any phase. It is available for GPRS handling and it has the following format:

```
releaseGprs gprsReleaseCause <integer> [ gprsPdPid <integer> ]
```

releaseSms

This operation causes the GSM (SCF) to terminate a short message submission attempt or short message delivery attempt and is allowed only within a control relationship. It is available for CAMEL handling and has the following parameter:

```
releaseSms reason <integer comparitor>
```

reportSMSDeliveryStatusResult

This operation is available for MAP handling:

```
reportSMDeliveryStatusResult [<parameters>]
```

A `reportSMSDeliveryStatusResult` operation must have the following parameter:

```
msisdn <number expression>
```

requestreportbcsmevent

This operation causes the SSF to monitor for call-related BCSM events such as busy or no-answer and notify the SCF when one is detected. This operation has the following format:

```
requestreportbcsmevent [<event>...]
```

A `requestreportbcsmevent` must have one or more event descriptions:

```
eventtypebcsm <type> [monitormode <mode>] [legid <legid> | ( <integer> )]
   [dbspecificcriteria <criteria>]
```

The value off `legid` must be one of the following:

```
[sendingsideid] <legtype>
[receivingsideid] <legtype>
```

The value of criteria must be one of the following:

```
numberofdigits <integer comparator>
applicationtimer <integer comparator>
```

For example:

```
eventTypeBCSM oNoAnswer (2) dpspecificcriteria applicationTimer 20
```

requestReportGprsEvent

Causes the GSM SCF to request the GPRS SSF to monitor for a GPRS session event or a PDP context event, such as establish or detach, and to notify the GSM SCF when one is detected.You can request monitoring of more than one event in a single operation but each one will be reported in a separate `EventReportGPRS` operation.This operation has the following format:

```
RequestReportGprsEvent gprsEventType <number> [ gprsPdPid <integer> ]
```

requestReportSmsEvent

Causes the GSM service control function to request the SMS SSF to monitor for a short message related event such as failure, delivery, or submission, and to notify the GSM SCF when it detects one. You can request monitoring of more than one event with a single operation but each event will be reported in a separate `EventReportSMS` operation. This operation is available for CAMEL handling and has the following format.

```
eventTypeSms [ smsFailure | smsSubmitted ]
```

resettimer

This operation causes the SCF to refresh the `tSSF` application timer to avoid the `tSSF` time-out at the service SSF. This operation supports only one parameter, `timervalue`. You cannot specify the timer ID and it is not checked in the received operation. It defaults to `tSSF`.

```
resettimer timervalue <integer>
```

returnError

The operation has the following format:

```
returnError|tcapError [invokeID <integer>] errorCode <integer>
```

The `returnError` operation and the `tcapError` parameter operation are synonymous. If you specify `invokeID`, it must match the response. Otherwise, the returned `invokeID` is not checked. In the **slpit** script, calls start with an `invokeID` of 0 and the value is increased by 1 for each subsequent call.

sendcharginginformation

This operation instructs the SSF on the charging information to be sent. No parameters are supported for this operation and the received content is not validated.

```
sendcharginginformation
```

sendRoutingInfoForSmResult

This operation is available for MAP handling and has the following format and parameters:

```
sendRoutingInfoForSmResult [ imsi <number comparitor> ] [ nnn <number comparitor>]
```

sendRoutingInformationResult

This operation is available for MAP handling and has the following format and parameters:

```
sendRoutingInformationResult [ imsi <number comparitor> ]
     [ nnn <number comparitor> ]
```

splitleg

This operation causes the GSM SCF to request the GSM SSF to separate one party from the source call segment and place it in a new target call segment. This operation is available for INAP level 2 (CS-2) handling and has the following format and parameters.

```
splitleg legtobesplit <integer> newcallsegment <integer>
```

tcapError

This operation has the following format and parameters:

```
returnError|tcapError [invokeID <integer>] errorCode <integer>
```

The `returnError` parameter and `tcapError` parameter are synonymous. If you specify `invokeID`, it must match the response. Otherwise, the returned `invokeID` is not checked. In the **slpit** script, calls start with an `invokeID` of 0 and the value is increased by 1 for each subsequent call.

unstructuredSSResult

This operation is available for MAP handling and has the following format and parameters.

```
unstructuredSSResult ussdString <string>
```

The `ussdString` parameter is the expected parameter in an `unstructuredSSResult` operation. If you do not specify it, no check is performed. If a check is performed, the string must match the returned string, or the call will be counted in the failed call statistics.

Example Scripts

This section illustrates the call definition statements in a **slpit** script file for two sample calls: a standard point A to point B call and a call that plays an announcement.

A Standard Call

The statements in this example define a standard point A to point B call with one subsequent reservation in which the called party hangs up. This is not a CAMEL call.

```
define call atb_two_periods {
    DN ?= "39421234567"
    CLI ?= "3099440000"

    send {
      initialdp
        calledpartynumber DN
        callingpartynumber CLI
        callingpartyscategory 10
        locationnumber CLI
        eventtypebcsm analyzedinformation
    }

    receive {
      applycharging
        thresholdtime any -> threshold
```

```
    warningtime threshold - 10

  requestreportbcsmevent
    eventtypebcsm omidcall (2)

  requestreportbcsmevent
    eventtypebcsm oCalledPartyBusy (2)
    eventtypebcsm oNoanswer (2)
    eventtypebcsm oabandon (1)
    eventtypebcsm RouteselectFailure
    eventtypebcsm oDisconnect (2)
    eventtypebcsm oDisconnect (1)

  callinformationrequest
    callattemptElapsedTime
    callstopTime
    callConnectedElapsedTime
    calledaddress
    releaseCause

  connect
    destroutingaddr DN
}

talktime = threshold

wait 1.0

send {
  eventreportbcsm
    eventtypebcsm omidcall (2)
}
receive {
  applyCharging
    thresholdtime any -> threshold
    warningtime threshold - 10

  RequestReportBCsMEvent
    eventtypebcsm omidcall (2)
  continue
}

talktime = talktime + threshold

wait 1.0

send {
  eventreportbcsm
    eventtypebcsm odisconnect (2)

  applychargingreport
    thresholdtime 20
    endofcallindicator 1

  callinformationreport
    callattemptelapsedtime 10
    callstoptime "001002000000"
    callConnectedElapsedTime (talktime - 20) * 10
    calledaddress DN
    releasecause 31
}
```

```
    receive {
      releasecall
        reason 31
    }
}

startcall atb_two_periods using once
```

A Call that Plays an Announcement

The statements in this example define a simple call that requests assistance and plays an announcement.

```
define call assisting_ip_pa {
    SERVICE_NUMBER ?= "555801"

    send {
      initialdp
        calledpartynumber SERVICE_NUMBER
        callingpartynumber "40002000"
        callingpartyscategory 10
        locationnumber "40002000"
        eventtypebcsm analyzedinformation
    }

    receive {
      establishtemporaryconnection
        address "1234"
    }

    send {
      assistrequestinstructions
    }

    receive {
      playannouncement
        annid any
    }

// Might receive the abort any time after sending the SRR.
    allow abort assisting

    send {
      specializedresourcereport
    }

    receive {
      disconnectForwardConnection
    releasecall
      reason normal
    }

// The abort might not have arrived yet.
    abort open
}
```

3
Testing IP Interactions with the mipt Utility

This chapter describes how to use the Oracle Communications Convergent Charging Controller **mipt** test utility.

About the mipt Utility

The **mipt** utility generally tests the sending and receiving of messages over IP, or internet-based protocols, and you can use it to do high load testing. See "Overview of the mipt Utility" for an overview of **mipt**.

The **mipt** utility processes operations from an input text file rather than a real network. The input text file is called the *script file* and it is a file that you create. In the script file, you add commands and send and receive messages that define the message sequences that you want to test. You can include send and receive messages of the following protocols: Diameter, EMI, M3UA, RADIUS, SMPP, and SUA. The scripting language allows you to define the test sequences. See "Creating the mipt Script File" for more information.

Depending on the protocol, the **mipt** utility can act as an Application Service Provider (ASP), a Short Message Service Center (SMSC), a RADIUS client, a DIAMETER Credit Control Client, or a DIAMETER Credit Control Server.

See Appendix B, "Supported Protocol Fields for mipt," for the fields that **mipt** supports for each protocol.

You can run multiple instances of **mipt**, acting as ASPs or SMSCs, communicating with each other on the same machine.

Running the mipt Utility

The **mipt** utility is located in the following directory:

```
/IN/service_packages/TEST_TOOLS/bin
```

Command Syntax

Use the following command to run the **mipt** utility:

```
mipt {<option> ...} [<file> ... <file>]
```

You can specify the user name, password, host name, and port either on the command line or in the script file.

See "Command-Line Options" for explanations of the command line options.

When the utility is acting as an ASP you can specify the user name, password, host name, and port number of the SMSC in the script with a statement like the following:

```
connect smpp_asp tcp smschost smscport bind_receiver system_id=username
    password=user_password
```

See "Specifying the Test Sequence" for additional information.

If you specify values both on the command line and in the script, the values you specify in the script override the values you specify on the command line.

Command-Line Options

The **mipt** utility takes the following command-line options:

-A
Accelerated time (for testing purposes only). Causes **mipt** to ignore any times specified in the script and perform operations as fast as it can

-D <var>=<string>
Assigns a string to a variable.

<file>
You must specify at least one script file on the command line. You can give your script files any name. If you specify multiple script files, **mipt** runs them in parallel. You can specify the script file name as -, in which case **mipt** reads the script from standard input. For example:

```
mipt {<options>} -<<EOF
```

This would be followed by lines of script like you would find in a script file and would be terminated by typing EOF.
See "Creating the mipt Script File" for more information.

-N <var>=<number>
Assigns a number to a variable.

-P <password>
Specifies the user's password on the host computer.

-U <user>
Specifies the user's login name on the host computer.

-V
Displays **mipt** version number and details about how and when **mipt** was created.

-d
Prints the script file after parsing it, rather than running it.

-e
Exits on error.

-l [<log options>]
Sets logging options. See "Logging Options" for a list of logging options.

-h <host>
Specifies the name of the host short message service center (SMSC) computer.

-n <protocol>
Specifies the network protocol to use if none is specified in the script file.

-p <port>
Specifies the port number on the host SMSC computer.

-q
Suppresses output to the command window other than errors, stats, and warnings.

-u latin
Sets EMI string processing to latin-1 (ISO8859-1).

-u unicode
Sets EMI string processing to UTF-8 (default is latin-1).

-v
Triggers verbose mode, which displays the packets being sent and received, including the `smpp_bind_transceiver` packet that enables the SMSC to authenticate the connection. The `smpp_bind_transceiver` packet has the following fields:

Field Name	Default Value
smpp_command_status	0
smpp_sequence_number	1
smpp_password	"PASSWORD"
smpp_system_id	"mipt"
smpp_system_type	"mipt"
smpp_interface_version	0x50
smpp_addr_ton	1
smpp_addr_npi	1
smpp_address_range	""

Logging Options

You can specify the following logging options with the **-l** command-line option. The default options are `error`, `script`, `stats`, and `warning`.

You can turn off a default logging option by preceding it with a dash (-).

The logging options are:

all
Enable all logging options except for `binary` and `execute`.

binary
Log binary packet contents as they are sent and received.

control
Log control packets as they are sent and received.

data
Log data packet contents as they are sent and received.

error
Log error messages.

execute
Log **mipt** execution at a low level. Use only for development.

match
Trace progress of packet matching.

network
Log information about the network state.

-q
Specify `-q` as shorthand for `-all,error,stats,warning`.

script
Log script execution.

stats
Print statistics.

-v
Enable all logging options except for `binary` and `execute`.

warning
Log warning messages.

werror
Log an error on any warnings.

Creating the mipt Script File

The **mipt** script file is a text file that you create to define message sequences that you want to test. You can name the **mipt** script file whatever you like.

> **Note:**
>
> The following sections use SMPP examples, using **mipt** as an ASP or SMSC, except for those examples that are specific to other protocols. The syntax and concepts demonstrated in the SMPP examples, however, apply to all protocols.

Specifying the Test Sequence

A script consists of the following five basic statements that you specify to create a test sequence.

- `connect <node>`

  The `connect` statement specifies the role of the initiating node and initiates the connection with the receiving node.

> **Note:**
>
> You can also specify the host name, port name, user ID, and password in the connect statement, using the following format:
>
> `connect <node> <protocol> <host> <port> <ID> <password>`

- `accept <node>`

  The `accept` statement specifies the role of the receiving node and accepts a connection request.

- `send <message>`

  The `send` statement sends a message as defined by `<message>`.

- `receive <message>`

  The `receive` statement defines the expected short message response in `<message>`.

- `end`

  The `end` statement defines the end of a `repeat` block and also defines the end of the test message script and causes **mipt** to disconnect. Its absence results in a syntax error.

Reserved keywords

The mipt utility supports the following reserved keywords for the `connect` and `accept` statements:

- emi_asp
- emi_smsc
- m3ua_asp
- m3ua_sgp
- diameter_raw
- diameter_cooked
- diameter_agent
- smpp_asp
- smpp_smsc
- sua_asp
- sua_sgp

Using mipt as an ASP or SMSC

The **mipt** utility can run as either the ASP or the SMSC. The first line of the script specifies the role that **mipt** plays. For example, the following line indicates that **mipt** is acting as an ASP and is ready to send messages to an SMSC and receive messages from it.

```
connect smpp_asp
```

The following line specifies that **mipt** is acting as an SMSC and is ready to receive messages from ASPs.

```
accept smpp_smsc
```

The following example shows a sample test script with SMPP messages, one sent and one received. This script simulates an ASP that sends a message to an SMSC and waits for it to be accepted.

```
connect smpp_asp

send submit_sm
    source_addr = "0274022020"
    destination_addr = "0274022023"
    short_message = octets "Hello World"

receive submit_sm_resp
    command_status = 0        # Expect success

end
```

Sending Multiple Messages

Typically, an ASP script will keep the connection open and send multiple messages, as seen in the following script:

```
connect smpp_asp

send submit_sm
    source_addr = "0274022020"
    destination_addr = "0274022023"
    short_message = octets "Hello World"

receive submit_sm_resp
    command_status = 0              # Expect success.

send submit_sm
    source_addr = "0274022020"
    destination_addr = "0274022023"
    short_message = octets "I'm still here"

receive submit_sm_resp
    command_status = 0              # Expect success.

end
```

To handle multiple messages on the SMSC, you must create a loop. For example:

```
accept smpp_smsc
    repeat
        receive submit_sm
        send submit_sm_resp
    end
end
```

The `repeat` statement specifies that the set of statements before the first `end` statement will repeat indefinitely until a message causes the SMSC to disconnect from the ASP. The first `end` statement specifies the end of the statements to be repeated, that is, the end of the loop. The second `end` statement terminates the script.

See "Controlling Loops" for more information about controlling loops.

Rejecting Messages

You might not want to accept all messages. The following script segment rejects messages that contain the word *bad*:

```
accept smpp_smsc
    repeat
        receive submit_sm
            short_message = octets "bad"
        send submit_sm_resp
            command_status = 1 # Failure
```

Using Variables

You can use variables to save values while the script is running. The following SMSC script saves the `message_id` field in `query_sm` to a variable named `$MID` and then uses it to set `message_id` in the `query_sm_resp` operation:

```
    receive query_sm
        message_id -> $MID
    send  query_sm_resp
        message_id = $MID
        final_date = "20050718152345002"
        message_state = 1
        error_code = 0
```

> **Note:**
>
> The **mipt** utility has many reserved words and using a reserved word as a variable name will result in an error. If you prefix your variable name with a $, you can avoid conflicts with reserved words.

Controlling the Message Flow

The **mipt** utility provides several statements that enable you to control the flow of messages in the script.

Providing an Alternate Flow

The following script example uses the `or` statement to direct the flow to an alternate path if a message does not meet a specific condition:

```
accept smpp_smsc
    repeat
        receive submit_sm
            short_message = octets "bad"
        send submit_sm_resp
            command_status = 1 # Failure

    or
        receive submit_sm
        send submit_sm_resp
```

```
    end
end
```

Controlling the Processing Sequence

When you send two messages at the same time, the replies could arrive out of order. The following is an example of sending two messages:

```
send submit_sm
    source_addr = "0274022020"
    destination_addr = "0274022023"
    short_message = octets "bad"

send submit_sm
    source_addr = "0274022020"
    destination_addr = "0274022023"
    short_message = octets "good"

receive submit_sm_resp
    command_status = 0  # success

receive submit_sm_resp
    command_status != 0 # failure
```

In this case, you do not know the order in which the replies will come so either response could be handled incorrectly as a success or a failure.

You can use the `and` operation to send two messages at the same time and ensure that the results are processed in the correct sequence. In the following example, the `and` operation allows you to send two statements at the same time and associate the `receive` operations with the correct `send` operations:

```
begin
    send submit_sm
        source_addr = "0274022020"
        destination_addr = "0274022023"
        short_message = octets "bad"
    receive submit_sm_resp
        command_status != 0 # failure
and
    send submit_sm
        source_addr = "0274022020"
        destination_addr = "0274022023"
        short_message = octets "good"
     receive submit_sm_resp
         command_status = 0 # success
end
```

The `begin` and `end` keywords define the scope of the `and` operation. In this case, the first `receive` operation will match only the first `send` operation, and the second `receive` operation will match only the second `send` operation.

The SMSC should be able to respond to `query_sm` requests as well as `submit_sm` requests. The following example uses the `and` operation to run two loops in parallel to respond to both `query_sm` and `submit_sm` requests:

```
begin
    repeat
        receive query_sm
            message_id -> MID
```

```
        send  query_sm_resp
            message_id = MID
            final_date = "20050718152345002"
            message_state = 1
            error_code = 0
    end
and
    repeat
        receive submit_sm
        send submit_sm_resp
    end
end
```

Controlling Loops

You can add options to the `repeat` operation to specify a rate or time that controls how many times or how long a loop will continue. For example, **mipt** does not know when to stop the following loop, which continuously sends `enquire_link` operations from the ASP to the SMSC:

```
repeat
    send enquire_link
    receive enquire_link_resp
    sleep 1 # Wait a second
end
```

You can limit how long a loop will repeat by specifying a time limit after which it will stop. The following loop repeats for up to 10 seconds:

```
repeat for 10 seconds
    send enquire_link
    receive enquire_link_resp
    sleep 1 # Wait a second.
end
```

You can also repeat the loop at a particular pace by specifying a rate of hertz, which is one iteration per second. The following example repeats the loop at a uniform distribution of 2 hertz for a duration of 10 seconds:

```
repeat uniform 2 hertz for 10 seconds
    send enquire_link
    receive enquire_link_resp
end
```

You can also specify intervals instead of frequencies. In this example, `uniform 0.5` seconds means one iteration every 0.5 seconds, which is equal to 2 hertz:

```
repeat uniform 0.5 seconds for 10 seconds
    send enquire_link
    receive enquire_link_resp
end
```

You can also ramp the rate up and down gradually. This example ramps up the rate from 0 to 10 hertz over a period of 5 seconds and then holds it at 10 hertz for a period of 6 seconds:

```
repeat uniform (5 seconds) 10 hertz for 6 seconds
    send enquire_link
    receive enquire_link_resp
end
```

This example ramps up the rate and then switches rates twice:

```
repeat poisson 10 hertz (10 seconds) 20 hertz () 1000 hertz (100 iterations)
       1000 hertz () 1 hertz for 20 seconds
```

This statement has the following effect on the loop:

- Uses a Poisson distribution model
- Ramps up from 10 hertz to 20 hertz over 10 seconds
- Switches to 1000 hertz for an average of 100 iterations, 1 iteration every 0.001 second for about 0.1 seconds
- Switches back to 1 hertz
- Runs for a total of 20 seconds

The following example demonstrates the implicit flow control in a **mipt** script. It sends messages as fast as possible until an error is returned:

```
send submit_sm
    source_addr = "0274022020"
    destination_addr = "0274022023"
    short_message = octets "spam"

repeat
    receive submit_sm_resp
        command_status = 0 // OK
    send submit_sm
        source_addr = "0274022020"
        destination_addr = "0274022023"
        short_message = octets "spam"
end

receive submit_sm_resp
    command_status != 0 // OK
```

The fields and values within the `receive` section, such as `command_status = 0`, specify the expected values. If the actual value received does not match the expected value, **mipt** generates an `Unmatched packet` error, which would terminate the loop in this case.

Using the RADIUS Protocol

The following example of a **mipt** script for the RADIUS protocol simulates a short prepaid billing session. The `connect` statement specifies the RADIUS protocol. The name of the script file can be anything.

```
connect radius_raw udp 'radius-server-hostname' 1812 packet radius_secret =
octets "SECRET"

      # Initial request for quota allocation
      send access_request
         user_name = '0219393571'
         radius_correlation_id = '12345678'
         nas_identifier = '12345'
         PPAC = {
            PPAC__select_for_session = octets 00 00 00 10  #Duration accounting
         }

      # Assume successful response and a quota allocated
```

```
      receive access_accept

      # Wait 10 seconds before disconnecting
      sleep 10

      # User disconnects
      send access_request
         radius_correlation_id = '12345678'

         radius_service_type = 17 # 'Authorize only', i.e. this is an "on-line" Access-
Request

         PPAQ = {
            PPAQ__quota_identifier = 1
            PPAQ__duration_quota = 10
            PPAQ__update_reason = octets 00 06  # Client Service termination
         }

      # Acknowledgement from the radius server
      receive access_accept

   end
```

Using mipt as a Diameter Client or Server

You can send Diameter messages over either a transmission control protocol (TCP) transport or a stream control transmission protocol (SCTP) transport. This section describes how to create a connection as a Diameter client and how to listen for a connection as a Diameter server, as well as how to send an error message.

See "Supported Protocol Fields for mipt", for a list of fields that the **mipt** utility supports for the Diameter protocol.

Connecting as a Diameter Client

Use the following format of the connect message to create a connection as a Diameter client in raw mode:

```
connect diameter_raw <protocol> <hostname> <port>
```

For example, the following `connect` statements establish Diameter client connections for the TCP and SCTP protocols, respectively:

```
connect diameter_raw tcp my_host 3868
connect diameter_raw sctp my_host 3868
```

You can also create a connection using cooked or agent mode, which have the following formats:

```
connect diameter_cooked <protocol> <hostname> <port>
connect diameter_agent <protocol> <hostname> <port>
```

There is no difference between cooked and raw mode when **mipt** is running as a Diameter client. When **mipt** is running as a Diameter server and using cooked mode, it automatically responds to Device-Watchdog-Request messages so that you do not need to handle them explicitly in the script.

The agent mode is similar to cooked mode except that it automatically performs a basic Capabilities-Exchange at the beginning of the connection.

The following script example connects to a Diameter server, performs a basic Capabilities-Exchange, sends a Credit-Control-Request (CCR) message, and waits for the answer:

```
connect diameter_agent 'server-hostname' 3868

    send CCR
        end_to_end_identifier = 55
        session_id = 'session id oh yeah'
        origin_host = 'host.example.com'
        origin_realm = 'host.example.com'
        destination_realm = 'host.example.com'
        auth_application_id = 1
        service_context_id = 'service context id woohoo'
        cc_request_type = 0
        cc_request_number = 1
        cost_information = {
            unit_value = {
                value_digits = -1000
                exponent = -20
            }
            currency_code = 888
            cost_unit = 'Polish Zlotys'
        }

    receive CCA
        end_to_end_identifier = 55

 end
```

Accepting a Connection as a Diameter Server

When **mipt** is running as a Diameter server, use the following format to listen for and accept a connection:

```
accept diameter_raw <protocol> <port>
```

The value of `<protocol>` is either TCP or SCTP and `<port>` is the port number to which **mipt** is listening.

Sending a Diameter Error Message

You can send a Diameter error message by sending an answer message and explicitly setting the error bit in the `command_flags` field, as shown in the following example:

```
send CCA    command_flags = 0x20  # Error bit    session_id = 'session id oh
yeah'    error_message = 'Error #12345'
```

4
Testing Messaging with the SMSC Test Tool

This chapter describes how to configure and use the Oracle Communications Convergent Charging Controller **smsc** test tool.

About the smsc Test Tool

The **smsc** test tool emulates various parts of the short message service (SMS) messaging environment, including a Short Message Service Center (SMSC), Visitor Mobile Switching Center (VMSC), and Home Location register (HLR). You use the utility to test the sending and receiving of SMS messages. See "Overview of the smsc Test Tool" for an overview of **smsc**.

The **smsc** test tool attaches to the SLEE as a TCAP interface and handles both MAP and IS-41 incoming short message requests. It can simulate an SMSC by sending an MO ForwardSM operation, an HLR by sending an SendRoutingInfoForSM operation, or an MSC by sending an MT ForwardSM operation at MAP levels 1-3. The responses to each method are defined in a configuration file. Selection of functionality is automatic, depending on which type of operation the test tool receives.

Running the smsc Test Tool

The **smsc** test tool is located in the following directory:

```
/IN/service_packages/TEST_TOOLS/bin/smsc
```

The **smsc** test tool runs as a SLEE interface when you start the SLEE; therefore, starting the SLEE starts **smsc**. You must have configured the **smsc** test tool and configured the SLEE for **smsc** before **smsc** will run. See "Configuring SLEE for smsc" for information about configuring the SLEE.

The **smsc** test tool redirects its output to a log file. The log file is written to the /IN/service_packages/TEST_TOOLS/tmp/smsc.log file.

Before starting the SLEE, set the following environment variables for **smsc**:

- ESERV_CONFIG_FILE

  Optional. Set to the location of the **eserv.config** file, which contains the XMS configuration. Defaults to **/IN/service_packages/etc/eserv.config**.

- SMSC_CONFIG_FILE

  Set to the location of the **smsc.cfg** file, which defines the SMSC responses to incoming messages. The following example illustrates a setting of the SMSC_CONFIG_FILE environment variable:

  ```
  SMSC_CONFIG_FILE=/IN/service_packages/TEST_TOOLS/etc/smsc.cfg
  export SMSC_CONFIG_FILE
  ```

- DEBUG

  Set to `fred_smsc` to obtain useful information about what **smsc** is doing.

> **Note:**
>
> If this variable is set to `all`, every SLEE program writes out debugging information, which is usually not desirable due to the volume of information, most of which is not meaningful for **smsc**.

Configuring SLEE for smsc

To configure the Service Logic Execution Environment (SLEE) for SMSC, you must include the following parameters in the **slee.cfg** file:

- SERVICEKEY=INTEGER 42 SMSCINT
- INTERFACE=Timer timerIF /IN/service_packages/SLEE/bin EVENT
- INTERFACE=SMSCINT smsc </path/to/the_smsc> EVENT

The following example highlights these parameters in an excerpt from the **slee.cfg** file.

```
# Standard SLEE definitions...
# (MAXAPPLICATIONS, MAXDIALOGS, etc)
WATCHDOG=/IN/service_packages/SLEE/bin/ watchdog
WATCHDOGCYCLETIME=3000
SERVICEKEY=INTEGER 101 xmsIf
SERVICEKEY=INTEGER 42 SMSCINT
SERVICE=ACS 1 slee_acs ACS
INTERFACE=Timer timerIF /IN/service_packages/SLEE/bin EVENT
INTERFACE=SMSCINT smsc </path/to/the_smsc> EVENT
INTERFACE=xmsIf xmsTrigger.sh /IN/service_packages/XMS/bin EVENT
APPLICATION=slee_acs slee_acs /IN/service_packages/ACS/bin 1 1
```

Other parameters listed here, which are important to running SLEE, are described in *Service Logic Execution Environment Technical Guide*.

Configuring the smsc Test Tool

Configuring the **smsc** test tool consists of adding configuration parameters to the **smsc.cfg** file, which is in the **/IN/service_packages/TEST_TOOLS/etc** directory by default. You can specify a different location for the file by setting the `SMSC_CONFIG` environment variable.

You add parameters to the **smsc.cfg** file based on the testing that you want to do. Configuring the **smsc** test tool can include the following tasks:

- Configuring General Parameters
- Configuring for CAP3 GPRS
- Configuring for MAP
- Configuring for MAP as HLR
- Configuring for IS-41
- Configuring for IS-41 as HLR

Configuring General Parameters

The following parameters apply to the **smsc** test tool rather than a particular protocol.

Set the following parameters in the **smsc.cfg** file to specify a sleep time and a request delay range.

Protocol

Deprecated.

> **Note:**
>
> The Protocol parameter formerly specified the protocol ("map" or "is41") to be used. However, the **smsc** test tool currently determines the protocol based on the incoming message.

requestDelayRangeStart

The requestDelayRangeStart and requestDelayRangeEnd parameters specify a range in seconds by which responses are delayed. The delay is randomly selected from within the specified range.

Specifies the start of the delay range.

Allowed Values	Zero or a positive integer
Units:	Seconds
Default:	0
Example:	requestDelayRangeStart = 0

requestDelayRangeEnd

Specifies the end of the delay range.

Allowed Values:	Zero or a positive integer
Units:	Seconds
Default:	0
Example:	requestDelayRangeEnd = 0

sleepTime

Specifies the sleep time between event polls when idle.

Allowed Values:	Zero or a positive integer
Units:	milliseconds
Default:	200
Example:	sleepTime = 200

Configuring for CAP3 GPRS

You can use the SMSC test tool to perform one CAP3 GPRS operation, ActivityTestGPRS.

Use the following parameters in the **smsc.cfg** file to perform an ActivityTestGPRS operation for the CAP3 GPRS protocol. This operation tests whether a relationship exists between the gsmSCF and gprsSSF. If so, the gprsSSF replies.

CAP3DestinationReferenceRange

Optional. Specifies an array of Destination Reference patterns and associated behaviors.

```
CAP3DestinationReferenceRange = [
    {
         pattern = ""
         responses = [
             {
                  reply = true | false
                  delay = <seconds>
             }
             ...
         ]
    }
    ...
]
```

You set the following values in the CAP3DestinationReferenceRange parameter.

- **delay**

  Mandatory for an entry in the responses array of CAP3DestinationReferenceRange. Specifies in seconds the length of time to delay the response.

Allowed Values:	Zero or a positive integer
Unit:	seconds
Default:	30
Example	delay = 30

- **pattern**

Mandatory for entry in the CAP3DestinationReferenceRange array. Specifies a regular expression as understood by the UNIX grep command.

Allowed Values:	A string that specifies a pattern as understood by the UNIX grep command.
Default:	""
Example:	pattern = 95.* (matches any destination reference beginning with a 95. pattern. The * specifies a match for any value following the 95. value.)

- **reply**

  Mandatory for entry in the CAP3DestinationReferenceRange array. Specifies whether to reply to an ActivityTestGPRS operation sent by the gsmSCF.

Allowed Values:	true or false
Default:	true
Example:	reply = true

- **responses**

  Mandatory for an entry in the CAP3DestinationReferenceRange array. Specifies an array of responses, each of which specifies a reply of true or false and a delay in seconds.

Allowed Values:	An array in which each entry consists of a boolean value and a number of seconds.
Default:	[true, 30]
Example:	response = [{true, 30}, {false, 20}];

Configuring for MAP

To use the MAP protocol, set the appropriate parameters in the **smsc.cfg** file.

AbortRequest

Deprecated. Setting AbortRequest to true is equivalent to Action = ABORT.

Allowed Values:	true or false
Default:	false
Example:	AbortRequest = false

Action

Specifies the action to take when a message is received. These values override IgnoreRequest, AbortRequest, or FailResponse.

Allowed Values:	• "ACK", "ACKNOWLEDGE_REQUEST", 0 Respond to an SMSDeliveryPointToPoint message by acknowledging it as a success. • "IGNORE", "IGNORE_REQUEST", 1 Ignore the request as if it was not received. • "ABORT", "TCAP_ABORT", "ABORT_REQUEST", 2 Close the dialog with a TCAP ABORT message. • "NACK", "FAIL_REQUEST", 3 Send a TCAP ReturnError component. • "NOTIFY_REQUEST", 5 Respond to a SendRoutingInformation operation with a TCAP NOTICE message. • "MAP_FALLBACK", "MAP_FALLBACK_REQUEST", 6 Respond to the request with a TCAP ABORT message requesting to fall back to the MAP version specified by fallbackVersion. • "CANCEL", "CANCEL_REQUEST", "TCAP_CANCEL", 7 Respond to the request with a TCAP CANCEL message.
Default:	"ACKNOWLEDGE_REQUEST"
Example:	Action = "NACK"

altAction

Used instead of Action for the percentage of time specified by altPercentage.

Allowed Values:	• "ACK", "ACKNOWLEDGE_REQUEST", 0 Respond to an SMSDeliveryPointToPoint message by acknowledging it as a success. • "IGNORE", "IGNORE_REQUEST", 1 Ignore the request as if it was not received. • "ABORT", "TCAP_ABORT", "ABORT_REQUEST", 2 Close the dialog with a TCAP ABORT message. • "NACK", "FAIL_REQUEST", 3 Send a TCAP ReturnError component. • "NOTIFY_REQUEST", 5 Respond to a SendRoutingInformation operation with a TCAP NOTICE message. • "MAP_FALLBACK", "MAP_FALLBACK_REQUEST", 6

	Respond to the request with a TCAP ABORT message requesting to fall back to the MAP version specified by fallbackVersion. • "CANCEL", "CANCEL_REQUEST", "TCAP_CANCEL", 7 Respond to the request with a TCAP CANCEL message.
Default:	"ACKNOWLEDGE_REQUEST"
Example:	altAction = "ACKNOWLEDGE_REQUEST"

altPercentage

Specifies the percentage of time to use altAction instead of Action

Allowed Values:	An integer between 0 and 255
Default:	0
Example:	altPercentage = 0

FailOpid

The error code to put in a TCAP ReturnError.

Allowed Values:	Zero or a positive integer.
Default:	32 (sm-DeliveryFailure))
Example:	FailOpid= 32

FailCause

The value that specifies the cause in a ReturnError message when the error code is 32 (sm-DeliveryFailure).

Allowed Values	Zero or a positive integer. Sensible values: 0 (memoryCapacityExceeded) 1 (equipmentProtocolError) 2 (equipmentNotSM-Equipped) 3 (unknownServiceCenter) 4 (sc-Congestion) 5 (invalidSME-Address) 6 (subscriberNotSC-Subscriber)

Default:	32 (which is not meaningful)
Example:	FailCause = 0

FailResponse

Deprecated. Setting FailResponse to `true` is equivalent to Action = "NACK".

Allowed Values:	true or false
Default:	false
Example:	FailResponse = false

fallbackVersion

Specifies the value when Action is MAP_FALLBACK. This is the MAP version to put in the TCAP ABORT message when MAP fallback is requested.

Allowed Values:	1, 2, or 3
Default:	1
Example:	fallbackVersion = 1

> **Note:**
>
> While this chapter and the **smsc.cfg** file refer to MAP version 3, its official name is MAP phase 2+.

IgnoreRequest

Deprecated. Setting IgnoreRequest to true is equivalent to Action = "IGNORE".

Allowed Values:	true or false
Default:	false
Example:	IgnoreRequest = false

MapVersion

Specifies the MAP version to support. SMC or SMSC requests that use a higher version are aborted (TCAP_ABORT) and **smsc** requests a fallback to this version, regardless of the value of the Action parameter.

> **Note:**
>
> While this chapter and the **smsc.cfg** file refer to MAP version 3, its official name is MAP phase 2+.

Allowed Values:	1, 2, or 3
Default:	1
Example:	MapVersion = 3

moreMessages

If true, the **smsc** test tool sends the MAP result in a TCAP_CONTINUE message, leaving the TCAP dialog open. If false, the **smsc** test tool sends the MAP result in a TCAP_END message, ending the TCAP dialog.

Allowed Values:	true or false
Default:	true
Example:	moreMessages = false

RECEIVED_SMS_STATUS_REPORT

Specifies the Action and FailOpid to use when processing messages that contain an SMS Status Report.

Allowed Values:	true or false
Default:	true
Example:	moreMessages = false

SEND_USSD_MAP_ERROR_VALUE

The error code to put in a TCAP ReturnError message for a MAP UNSTRUCTURED_SS_NOTIFY or PROCESS_UNSTRUCTURED_SS operation.

Allowed Values:	Zero or a positive integer
Default:	1
Example:	SEND_USSD_MAP_ERROR_VALUE = 1

SEND_USSD_PROCESS_UNSTRUCTURED_SS_RESPONSE

Specifies the action to take on receipt of a MAP_PROCESS_UNSTRUCTURED_SS operation.

Allowed Values:	0 Send a TCAP ReturnError 1 Send a TCAP ReturnResult 2 Do nothing
Default:	1
Example:	SEND_USSD_PROCESS_UNSTRUCTURED_SS_RESPONSE = 1

SEND_USSD_SS_NOTIFY_RESPONSE

Specifies the action to take on receipt of a MAP UNSTRUCTURED_SS_NOTIFY operation.

Allowed Values:	true or false
Default:	true
Example:	moreMessages = false

TCAP_ABORT_cause

Specifies the cause (p-abortCause) to put in a TCAP ABORT message.

Allowed Values:	An integer between 0 and 255. Meanings for commonly used values include: • unrecognizedMessageType (0) • unrecognizedTransactionID (1) • baadlyFormatteTransactionPortion (2) • incorrectTransactionPortion (3) • resourceLimitation (4)
Default:	0
Example:	TCAP_ABORT_cause = 0

Example MAP Configuration

The following example illustrates the **smsc** configuration parameters for MAP in the **smsc.cfg** file:

```
# smsc.cfg
SMSC = {    ...    MAP = {        moreMessages = False        MapVersion =
3        IgnoreRequest = False        AbortRequest = False        FailResponse =
False        Action = "ACKNOWLEDGE_REQUEST"        TCAP_ABORT_cause = 0
```

```
altAction = "ACKNOWLEDGE_REQUEST"        altPercentage = 0        FailOpid = 32
FailCause = 32        fallbackVersion = 1        SEND_USSD_SS_NOTIFY_RESPONSE =
1        SEND_USSD_PROCESS_UNSTRUCTURED_SS_RESPONSE = 1
SEND_USSD_MAP_ERROR_VALUE = 1        RECEIVED_SMS_STATUS_REPORT = {            Action
= "ACKNOWLEDGE_REQUEST"            altAction = "ACKNOWLEDGE_REQUEST"
altPercentage = 0            FailOpid = 32            FailCause = 32        }
}
```

Configuring for MAP as HLR

When Protocol is set to MAP, you can also configure SMSC to respond as an HLR when it receives a SendRoutingInfoForSM message.

> **Note:**
>
> You can enable both SMS and HLR capabilities at the same time.

Use the following parameters to configure SMSC as an HLR.

Action

Specifies the action to take upon receipt of a lookup request. This action supersede the IgnoreRequest, AbortRequest, and FailResponse parameters.

Allowed Values:	• "ACK", "ACKNOWLEDGE_REQUEST", 0 Respond to an SMSDeliveryPointToPoint message by acknowledging it as a success. • "IGNORE", "IGNORE_REQUEST", 1 Ignore the request as if it was not received. • "ABORT", "TCAP_ABORT", "ABORT_REQUEST", 2 Close the dialog with a TCAP ABORT message. • "NACK", "FAIL_REQUEST", 3 Send a TCAP ReturnError component. • "NOTIFY_REQUEST", 5 Respond to a SendRoutingInformation operation with a TCAP NOTICE message. • "MAP_FALLBACK", "MAP_FALLBACK_REQUEST", 6 Respond to the request with a TCAP ABORT message requesting to fall back to the MAP version specified by fallbackVersion. • "CANCEL", "CANCEL_REQUEST", "TCAP_CANCEL", 7 Respond to the request with a TCAP CANCEL message.
Default:	"ACKNOWLEDGE_REQUEST"

Example:	Action = "ACKNOWLEDGE_REQUEST"

Address

These are the digits of the MSC number to put in SEND_ROUTING_INFORMATION_FOR_SM results.

Allowed Values:	A string of digits
Default:	empty (do not return an address
Example:	Address = 12345678

AddressNature

This is the nature address of the MSC number to put in SEND_ROUTING_INFORMATION_FOR_SM results.

Allowed Values:	An integer between 0 and 7 Meanings: 0 unknown 1 international number 2 national significant number 3 network specific number 4 subscriber number 5 reserved 6 abbreviated number 7 reserved for extension
Default:	0
Example:	AddressNature = 1

AddressPlan

This is the indicator of the numbering plan for the MSC number to put in SEND_ROUTING_INFORMATION_FOR_SM results.

Allowed Values:	An integer between 0 and 15 Meanings: 0 unknown 1 ISDN/Telephony Numbering plan (Rec CCITT E.164)

	2 spare 3 data numbering plan (CCITT rec X.121) 4 Telex numbering plan (CCITT rec F.69) 5 spare 6 land mobile numbering plan (CCITT Rec E.212) 7 spare 8 national numbering plan 9 private numbering plan 15 reserved for extension
Default:	0
Example:	AddressPlan = 1

canSendEmptyImsi

Controls what happens if the value of the IMSI parameter is an empty string (" "). If the value of canSendEmptyImsi is true, the IMSI is sent as an empty string. If the value of canSendEmptyImsi is false, the IMSI is constructed of the imsiPrefix value followed by the MSISDN.

Allowed Values:	true or false
Default:	false
Example:	canSendEmptyImsi = false

Denied

Deprecated. A value of true is equivalent to Action = "NACK".

Allowed Values:	true or false
Default:	false
Example:	Denied = false

Enabled

Deprecated. When true, enables the HLR functions and **smsc** responds to those messages. A value of false is equivalent to Action = "IGNORE".

Allowed Values:	true or false

Default:	true
Example:	Enabled = true

Error

Deprecated. When `true`, **smsc** responds to HLR messages with an error code.

Allowed Values:	true or false
Default:	true
Example:	moreMessages = false

ErrorCause

The cause of failure to include in the NACK response, if appropriate to the operation ID.

Allowed Values:	true or false
Default:	true
Example:	moreMessages = false

ErrorOpid

Deprecated. The error code to put in a TCAP ReturnError message.

Allowed Values:	0 or a positive integer
Default:	0, which is undefined and not meaningful.
Example:	ErrorOpid = 1 (unknownSubscriber)

FailOpid

The error code to put in a TCAP ReturnError message.

> **Note:**
>
> FailOpid overrides the deprecated ErrorOpid parameter.

Allowed Values:	0 or a positive integer

Default:	0, which is undefined and not meaningful.
Example:	FailOpid = 1 (unknownSubscriber)

fallbackVersion

If the action is FALLBACK_REQUEST, this value is the version specified in TCAP_ABORT, assuming that the logic for MapVersion does not apply.

Allowed Values:	true or false
Default:	true
Example:	moreMessages = false

GPRSNodeIndicator

The LocationInfoWithLMSI.gprsNodeIndicator value to put in a MAP2 SEND_ROUTING _INFORMATION_FOR_SM result.

Allowed Values:	true or false
Default:	false
Example:	GPRSNodeIndicator = true

hlrAltAction

Specifies the alternative action to take upon receipt of a lookup request. The hlrAltAction parameter is used instead of the Action parameter for the percentage of time specified by hlrAltPercentage.

Allowed Values:	• "ACK", "ACKNOWLEDGE_REQUEST", 0 Respond to an SMSDeliveryPointToPoint message by acknowledging it as a success. • "IGNORE", "IGNORE_REQUEST", 1 Ignore the request as if it was not received. • "ABORT", "TCAP_ABORT", "ABORT_REQUEST", 2 Close the dialog with a TCAP ABORT message. • "NACK", "FAIL_REQUEST", 3 Send a TCAP ReturnError component. • "NOTIFY_REQUEST", 5 Respond to a SendRoutingInformation operation with a TCAP NOTICE message.

	• "MAP_FALLBACK", "MAP_FALLBACK_REQUEST", 6 Respond to the request with a TCAP ABORT message requesting to fall back to the MAP version specified by fallbackVersion. • "CANCEL", "CANCEL_REQUEST", "TCAP_CANCEL", 7 Respond to the request with a TCAP CANCEL message.
Default:	"ACKNOWLEDGE_REQUEST"
Example:	hlrAltAction = "ACKNOWLEDGE_REQUEST"

hlrAltPercentage

Specifies as a percentage the number of times that **smsc** uses hlrAltAction instead of Action.

Allowed Values:	An integer between 0 and 255
Default:	0
Example:	hlrAltPercentage = 0

HLRSequence

Optional. This parameter is an array that enables you to configure a different responses for the MAP plugin when using HLR functionality for Status Reports and Submit messages. See "SMSC Sequence for MAP as HLR" for more information.

IMSI

The International Mobile Subscriber Identifier (IMSI) to put in the SEND_ROUTING_INFORMATION result or SEND_ROUTING_INFORMATION_FOR_SM result.

Allowed Values:	A string of exactly 16 hex digits
Default:	"" (empty string) See "canSendEmptyImsi" for related information on sending an empty string.
Example:	IMSI = "0123456789ABCDEF"

imsiPrefix

Specifies the prefix to turn the MSISDN into an IMSI, when the IMSI parameter is empty.

Allowed Values:	A string of between 0 and 15 hex digits
Default:	"" (empty)

Example:	imsiPrefix = "123"

ISC

MAP 3 only. Specifies whether to follow every ReturnError or ReturnResult message with an INFORM_SERVICE_CENTER operation.

Allowed Values:	true or false
Default:	false
Example:	ISC = false

ISCMwStatus

If non-zero, ISCMwStatus is included as the mw-Status field of the MAP INFORM_SERVICE_CENTER operation. Only the following six least-significant bits are valid:

Allowed Values:	An integer between 0 and 63. Only the following six least-significant bits are valid: 0x20 AddressNotIncluded 0x10 mnrf-Set (Mobile Not Reachable Flag) 0x08 mcef-Set (Memory Capacity Exceeded Flag) 0x04 mnrg-Set (Memory Not Reachable for GPRS)
Default:	0.
Example:	ISCMwStatus = 4

> **Note:**
>
> You can apply the logical OR operation to these bits. For example, applying a logical OR operation to 0x20 (01 0000) and 0x04 (00 0100) results in 0x24 (01 0100), which means address not included and memory not reachable for GPRS.

ISCSeparate

Specifies whether to send the INFORM_SERVICE_CENTER in a separate TCAP message from the TCAP ReturnError message.

Allowed Values:	true or false
Default:	false

Example:	ISCSeparate= false

ISCStoredMsisdnNoa, ISCStoredMsisdnNpi, ISCStoredMsisdn

If present, is included as the storedMSISDN field of the ISC operation.

Allowed Values:	true or false
Default:	true
Example:	moreMessages = false

LMSI

Specifies the LMSI to put in the SEND_ROUTING_INFORMATION_FOR_SM results.

Allowed Values:	A string of exactly 8 hex digits
Default:	empty (do not return an LMSI)
Example:	LMSI = "12345678"

MapVersion

Specifies the MAP version to use. If the incoming MAP version is higher, SMSC sends a TCAP ABORT message to fall back to this version, regardless of the version specified.

> **Note:**
>
> While this chapter and the **smsc.cfg** file refer to MAP version 3, its official name is MAP phase 2+.

Allowed Values:	1, 2, or 3
Default:	1
Example:	MapVersion = 1

SGSN

Specifies the digits of the SGSN number to put in SEND_ROUTING_INFORMATION_FOR_SM results.

Allowed Values:	A string of digits

Default:	empty (do not return an address)
Example:	SGSN= "12345678"

SGSNNature

Specifies the nature address of the Service GPRS Support Node (SGSN) number to put in SEND_ROUTING_INFORMATION_FOR_SM results.On behalf of mobile subscribers, SGSN manages access to network resources and enacts the packet scheduling policy. It also initiates the Packet Data Protocol (PDP) context with the gateway GPRS support node (GGSN).

Allowed Values:	An integer between 0 and 7 Meanings: 0 unknown 1 international number 2 national significant number 3 network specific number 4 subscriber number 5 reserved 6 abbreviated number 7 reserved for extension
Default:	0
Example:	SGSNNature = 1

SGSNPlan

Specifies the indicator of the numbering plan of the SGSN number to put in SEND_ROUTING_INFORMATION_FOR_SM results.

Allowed Values:	An integer between 0 and 15 Meanings: 0 unknown 1 ISDN/Telephony Numbering Plan (Rec CCITT E.164) 2 spare 3 data numbering plan (CCITT Rec x.121) 4 telex numbering plan (CCITT Rec F.69 5 spare

	6 land mobile numbering plan (CCITT Rec E.212) 7 spare 8 national numbering plan 9 private numbering plan 15 reserved for extension
Default:	0
Example:	SGSNPlan = 1

Example MAP as HLR Configuration

The following example illustrates the **smsc** configuration parameters for MAP when **smsc** acts as an HLR:

```
# smsc.cfg
SMSC = {
    ...
    MAP = {
        ...
        RECEIVED_SMS_STATUS_REPORT = {
            ...
        }
# This section describes how the SMSC handles SEND-ROUTING-INFORMATION
# and SEND_ROUTING-INFORMATION-FOR-SM requests.
        HLR = {

            Enabled = True
            Denied = False
            Action = "ACKNOWLEDGE_REQUEST"
            MapVersion = 1
            ErrorOpid = 1 # unknownSubscriber
            FailOpid = 1 # unknownSubscriber
            ISC = False
            ISCSeparate = False
            TCAP_ABORT_cause = 0
            ISCStoredMsisdnNoa = 1
            ISCStoredMsisdnNpi = 1
            ISCStoredMsisdn = "12345678"
            ISCMwStatus
            IMSI = "0123456789ABCDEF"
            canSendEmptyImsi = False
            GPRSNodeIndicator
            imsiPrefix = "123"
            AddressNature = 1
            AddressPlan = 1
            Address = "12345678"
            SGSNNature = 1
            SGSNPlan = 1
            SGSN = "12345678"
            LMSI = "12345678"
        }
    }
}
```

SMSC Sequence for MAP as HLR

When Protocol is set to MAP, you can add an HLRSequence parameter to configure different responses for the MAP plugin when using HLR functionality for Status Reports and Submit messages. HLRSequence is an array where each array element represents a response to a SEND_ROUTING_INFORMATION or SEND_ROUTING_INFORMATION_FOR_SM request.

Each parameter available for HLR configuration is also available for HLRSequence.

The following rules apply to use of the HLRSequence parameter:

- If HLRSequence has one response, it is used for all responses.
- If HLRSequence has more than one response, the first request will be handled according to the first array element, the second request will be handled according to the second array element, and so on. When the **smsc** test tool reaches the end of the array, it loops back to the beginning of the array.
- If HLRSequence is absent or is present but has no responses, HLR is used for all responses.

The following example illustrates the use of the HLRSequence parameter:

```
SMSC = {
    ...
    MAP = {
       ...
        RECEIVED_SMS_STATUS_REPORT = {
            ...
        }
        # This section describes how the SMSC should handle
        # SEND-ROUTING-INFORMATION and SEND_ROUTING-INFORMATION-FOR-SM requests.
        HLR = {
           ...
        }
        hlrAltAction = "ACKNOWLEDGE_REQUEST"
        hlrAltPercentage = 0
        HLRSequence = [
            {
            # Each of these sections has the same format as the SMSC.MAP.HLR
            # section above.
            }
            ...
            {
            }
        ]
        ...
    }
    ...
}
```

Map SendUSSDNotification

The SEND_USSD_SS_NOTIFY_RESPONSE and SEND_USSD_MAP_ERROR_VALUE parameters configure the response to the Advanced Control Services (ACS) for a MAP-UNSTRUCURED-SS-NOTIFY message.

To send a valid response, which is the default, set SEND_USSD_SS__NOTIFY_RESPONSE to 1:

SEND_USSD_SS_NOTIFY_RESPONSE=1

To send a MAP ERROR response, set SEND_USSD_SS_NOTIFY_RESPONSE to 0:

SEND_USSD_SS_NOTIFY_RESPONSE=0

> **Note:**
>
> Use SEND_USSD_MAP_ERROR_VALUE=*value* to set the error code value (1 - 44) as defined in **Map2Types.h**.

Set SEND_USSD_SS_NOTIFY_RESPONSE to 2 to indicate that no response is returned:

SEND_USSD_SS_NOTIFY_RESPONSE=2

The following example illustrates the SEND_USSD_SS_NOTIFY_RESPONSE and SEND_USSD_MAP_ERROR_VALUE parameters:

```
AP = {
    ...
    SEND_USSD_SS_NOTIFY_RESPONSE=0
    SEND_USSD_MAP_ERROR_VALUE=1
}
```

XMS tcapInterfaceName

The tcapInterfaceName parameter specifies the name of the interface that XMS uses to communicate with the SMSC. The parameter value is the same name that is specified as the SMSC interface by the INTERFACE parameter in the **SLEE.cfg** file.

The following example illustrates the tcapInterfaceName parameter in the **smsc.cfg** file:

```
XMS = {
xmsTrigger = {
    plugins = [
    # MAP plugin
    {
        lib = "xmsiMap.so"
        SSN = 8
        prefix = "485"
        pluginId = 1
        config = {
            contextKey = 1234
            tcapInterfaceName = "SMSCINT"
            GT = "5114406267"
            PC = 55
            SSN = 8
        }
    }
}
}
```

Configuring for IS-41

When the Protocol parameter in the **smsc.cfg** file is set to IS-41, **smsc** accepts SMSDeliveryPointToPoint requests and responds to them based on the following IS-41 parameters.

Use the following parameters to configure **smsc** for the IS-41 protocol.

ACK

Deprecated in favor of Action.

If the value is `true`, respond with a positive value for SMSDeliveryPointToPointResult, but one without an SMS_CauseCode value.

Allowed Values:	true or false
Default:	false
Example:	ACK = false

Action

Specifies the action to take upon receipt of a message, which supersedes the ACK, NACK, TCAP_ABORT, and Ignore items. You can also use the strings "ack", "nack", "abort", and "ignore".

Allowed Values:	• "ACK", "ACKNOWLEDGE_REQUEST", 0 Respond to an SMSDeliveryPointToPoint message by acknowledging it as a success. • "IGNORE", "IGNORE_REQUEST", 1 Ignore the request as if it was not received. • "ABORT", "TCAP_ABORT", "ABORT_REQUEST", 2 Close the dialog with a TCAP ABORT message. • "NACK", "FAIL_REQUEST", 3 Respond to the request with a ReturnResult message but with a cause value to indicate failure. See "SMS CauseCode Mapping" for cause values. "TCAP_U_ERROR", 4 Send a TCAP ReturnError component. • "NOTIFY_REQUEST", 5 Respond to a SendRoutingInformation operation with a TCAP NOTICE message.
Default:	"ACKNOWLEDGE_REQUEST"

Example:	Action = "ACKNOWLEDGE_REQUEST"

altAction

Specifies the action to take upon receipt of a message, which supersedes the ACK, NACK, TCAP_ABORT, and Ignore items. You can also use the strings "ack", "nack", "abort", and "ignore".

Allowed Values:	• "ACK", "ACKNOWLEDGE_REQUEST", 0 Respond to an SMSDeliveryPointToPoint message by acknowledging it as a success. • "IGNORE", "IGNORE_REQUEST", 1 Ignore the request as if it was not received. • "ABORT", "TCAP_ABORT", "ABORT_REQUEST", 2 Meaning: Close the dialog with a TCAP ABORT message. • "NACK", "FAIL_REQUEST", 3 Meaning: Respond to the request with a ReturnResult message but with a cause value to indicate failure. See "SMS CauseCode Mapping" for cause values. "TCAP_U_ERROR", 4 Meaning: Send a TCAP ReturnError component. • "NOTIFY_REQUEST", 5 Meaning: Respond to a SendRoutingInformation operation with a TCAP NOTICE message.
Default:	"ACKNOWLEDGE_REQUEST"
Example:	altAction = "ACKNOWLEDGE_REQUEST"

altPercentage

Specifies the percentage of time to use altAction instead of Action.

Allowed Values:	An integer between 0 and 255
Default:	0
Example:	altPercentage = 0

defaultDestPC

The point code to put in the SCCP Called Party Address of TCAP messages containing IS-41 operations.

Allowed Values:	An integer between 0 and 16383
Default:	3
Example:	defaultDestPC = 2001

defaultDestSSN

The subsystem number to put in the SCCP Called Party Address of TCAP messages containing IS-41 operations.

Allowed Values:	An integer between 0 and 255
Default:	3
Example:	defaultDestSSN = 8

Ignore

Deprecated in favor of Action. You can also use the string "ignore".

If the value is true, ignore the request as if it was not received.

Allowed Values:	true or false
Default:	false
Example:	Ignore = false

NACK

Deprecated in favor of Action.

If the value is `true`, respond with a negative value for SMSDeliveryPointToPointResult by including an SMS_CauseCode value.

Allowed Values:	true or false
Default:	false
Example:	NACK = false

NACK_StatusCode

The action to put in the status code when the action is FAIL_REQUEST.

Allowed Values:	Zero or a positive integer. Meaningful values:

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

Default:	3
Example:	NACK_StatusCode = 3

RECEIVED_SMS_STATUS_REPORT

The parameters in this section have the same meaning as their counterparts in the IS41 section. They apply, however, only to the action that the **smsc** test tool takes upon receiving a RECEIVED_SMS_STATUS_REPORT message. You can include the following parameters in the RECEIVED_SMS_STATUS_REPORT section:

- Action
- altAction
- altPercentage
- NackStatusCode
- TCAP_ABORT
- TCAP_Notice_cause
- TCAP_U_ERROR_code

> **Note:**
>
> If any of these parameters is not present, the value from its counterpart IS-41 parameter is used instead.

For example:

```
 IS41 = {
        ...
        RECEIVED_SMS_STATUS_REPORT = {
            Action = "ACKNOWLEDGE_REQUEST"
            altAction = "ACKNOWLEDGE_REQUEST
            altPercentage = 0
            TCAP_ABORT_cause = 0
            TCAP_NOTICE_cause = 0
            TCAP_U_ERROR_code = 0
            NACK_StatusCode = 3
        }
...
}
```

TCAP_ABORT

Deprecated in favor of Action. You can also use the string "abort".

If the value is `true`, respond by sending a TCAP_ABORT and closing the dialog.

Allowed Values:	true or false
Default:	false

Example:	TCAP_ABORT = false

TCAP_ABORT_cause

If TCAP_ABORT is `true`, the value of TCAP_ABORT_cause is the numeric code for the cause of the abort.

Allowed Values:	An integer between 0 and 255 Meaningful Values: 0 Unrecognized Packet Type 1 Unrecognized Trans ID 2 Badly Structured Trans Portion 3 Incorrect Trans Portion 4 Resource Unavailable 5 Permission To Release Problem 6 Unrecognized dialogue portion ID 7 Badly structured dialogue portion 8 Missing dialogue portion 9 Inconsistent dialogue portion
Default:	0
Example:	TCAP_ABORT_cause = 0

TCAP_NOTICE_cause

The cause to put in a TCAP_NOTICE message.

Allowed Values:	Zero or a positive integer. Meaningful values: 0 no translation for an address of such nature 1 no translation for this specific address 2 subsystem congestion 3 subsystem failure 4 unequipped user 5 MTP failure 6 network congestion 7 unqualified

	8 error in message transport (Note) 9 error in local processing (Note) 10 destination cannot perform reassembly (Note) 11 SCCP failure 12 hop counter violation 13 segmentation not supported 14 segmentation failure
Default:	0
Example:	TCAP_NOTICE_cause = 0

TCAP_U_ERROR

Deprecated in favor of Action.

Allowed Values:	true or false
Default:	false
Example:	TCAP_U_ERROR = false

TCAP_U_ERROR_code

The error code to put in a TCAP ReturnError

Allowed Values:	Zero or a positive integer.
Default:	0
Example:	TCAP_U_ERROR_code = 0

Configuring for IS-41 as HLR

When Protocol is set to IS41, you can configure **smsc** to respond when it is sent an SMSRequest.

> **Note:**
>
> SMS and HLR functionality is independent of each other.

The **smsc** response is based on the following parameters:

AccessDenied

Deprecated. When true, this parameter is equivalent to Action = "NACK". which responds to the request with a ReturnResult and a cause value that indicates failure.

Allowed Values:	true or false
Default:	False
Example:	AccessDenied = False

AccessDeniedReason

Specifies the reason access was denied when the action is FAIL_REQUEST.

Allowed Values:	0 NotUsed1 Denied2 Postponed3 Unavailable4 Invalid
Default:	0
Example:	AccessDeniedReason = 0

Action

Specifies the action to take upon receipt of a message. The Action parameter supersedes the Enabled and AccessDenied items.

Allowed Values:	• "ACK", "ACKNOWLEDGE_REQUEST", 0 Respond to an SMSDeliveryPointToPoint message by acknowledging it as a success. • "IGNORE", "IGNORE_REQUEST", 1 Ignore the request as if it was not received. • "ABORT", "TCAP_ABORT", "ABORT_REQUEST", 2 Meaning: Close the dialog with a TCAP ABORT message. • "NACK", "FAIL_REQUEST", 3 Meaning: Respond to the request with a ReturnResult message but with a cause value to indicate failure. See "TCAP_U_ERROR", 4 Meaning: Send a TCAP ReturnError component. • "NOTIFY_REQUEST", 5 Meaning: Respond to a SendRoutingInformation operation with a TCAP NOTICE message.
Default:	"ACKNOWLEDGE_REQUEST"

Example:	Action = "ACKNOWLEDGE_REQUEST"

Enabled

Deprecated. Enables HLR capabilities when `true`. When false, is equivalent to Action = "IGNORE".

Allowed Values:	true or false
Default:	true
Example:	Enabled= true

esnManufacturerCode

The manufacturer code of the mobile device's Electronic Serial Number (ESN). This is returned in the SMSRequestResult message if an ESN was not present in the SMSRequest message.

Allowed Values:	An integer between 0 and 255
Default:	0
Example:	esnManufacturerCode = 123

esnSerialNumber

The mobile device's Electronic Serial Number (ESN), which is returned in the SMSRequestResult message if an ESN was not present in the SMSRequest message.

Allowed Values:	An integer between 0 and 16777215
Default:	0
Example:	esnSerialNumber = 12345678

hlrAltAction

Specifies the alternative action to take upon receipt of a lookup request. The hlrAltAction parameter is used instead of the Action parameter the percentage of time specified by hlrAltPercentage.

Allowed Values:	• "ACK", "ACKNOWLEDGE_REQUEST", 0 Respond to an SMSDeliveryPointToPoint message by acknowledging it as a success. • "IGNORE", "IGNORE_REQUEST", 1

	Ignore the request as if it was not received. • "ABORT", "TCAP_ABORT", "ABORT_REQUEST", 2 Close the dialog with a TCAP ABORT message. • "NACK", "FAIL_REQUEST", 3 Respond to the request with a ReturnResult message but with a cause value that indicates failure. See "SMS CauseCode Mapping" for a list of cause codes. "TCAP_U_ERROR", 4 Send a TCAP ReturnError component. • "NOTIFY_REQUEST", 5 Respond to a SendRoutingInformation operation with a TCAP NOTICE message.
Default:	"ACKNOWLEDGE_REQUEST"
Example:	hlrAltAction = "ACKNOWLEDGE_REQUEST"

hlrPercentage

Specifies the percentage of time to use the hlrAltAction parameter instead of the Action parameter.

Allowed Values:	An integer between 0 and 255.
Default:	0
Example:	hlrAltPercentage

HLRSequence

Optional. This parameter is an array that enables you to configure a different responses for the MAP plugin when using HLR functionality for Status Reports and Submit messages. See "SMSC Sequence for MAP as IS-41" for more information.

mobileIdentificationNumber

The mobile identification number (MIN), which is returned in the SMSRequestResult message if the MDN was present in the SMSRequest message.

Allowed Values:	A string of exactly ten digits
Default:	"1234567890"
Example:	mobileIdentificationNumber = "1234567890"

NatureOfNumber

Specifies whether the routing address is national or international.

Allowed Values:	"national" or "international" (case sensitive).
Default:	"national"
Example:	NatureOfNumber = "national"

NumberPlan

Specifies the number plan portion of the routing address, which is the address of the MSC at which the mobile device currently can be reached.

Allowed Values:	An integer between 0 and 255 Meaning (from standard IS-41D): 0 Unknown or not applicable. 1 ISDN Numbering (not used in this Standard). 2 Telephony Numbering (ITU-T Rec. E.164, E.163). 3 Data Numbering (ITU-T Rec. X.121) (not used in this Standard). 4 Telex Numbering (ITU-T Rec. F.69) (not used in this Standard). 5 Maritime Mobile Numbering (not used in this Standard). 6 Land Mobile Numbering (ITU-T Rec. E.212) 7 Private Numbering Plan (service provider defined). 13 ANSI SS7 Point Code (PC) and SubsystemNumber (SSN). 14 Internet Protocol (IP) Address. 15 Reserved for extension. X Other values are reserved.
Default:	2 (Telephony Numbering)
Example:	NumberPlan = 2

returnMIN

Specifies whether to include the MIN in the SMSRequestResult message.

Allowed Values:	true or false
Default:	false

Example:	returnMIN = false

RoutingAddress

The global address of the MSC at which the mobile device currently can be reached.

Allowed Values:	A string of digits.
Default:	0
Example:	RoutingAddress = "12345678"

TCAP_ABORT_cause

If TCAP_ABORT is `true`, the value of TCAP_ABORT_cause is the numeric code for the cause of the abort.

Allowed Values:	An integer between 0 and 255 Meaningful Values: 0 Unrecognized Packet Type 1 Unrecognized Trans ID 2 Badly Structured Trans Portion 3 Incorrect Trans Portion 4 Resource Unavailable 5 Permission To Release Problem 6 Unrecognized dialogue portion ID 7 Badly structured dialogue portion 8 Missing dialogue portion 9 Inconsistent dialogue portion
Default:	0
Example:	TCAP_ABORT_cause = 0

TCAP_NOTICE_cause

The cause to put in a TCAP_NOTICE message.

Allowed Values:	An integer between 0 and 255. Meaningful values: 0 no translation for an address of such nature

	1 no translation for this specific address 2 subsystem congestion 3 subsystem failure 4 unequipped user 5 MTP failure 6 network congestion 7 unqualified 8 error in message transport (Note) 9 error in local processing (Note) 10 destination cannot perform reassembly (Note) 11 SCCP failure 12 hop counter violation 13 segmentation not supported 14 segmentation failure
Default:	0
Example:	TCAP_NOTICE_cause = 0

SMSC Sequence for MAP as IS-41

When Protocol is set to IS41, you can add an HLRSequence parameter to configure different responses for the MAP plugin when using HLR functionality for Status Reports and Submit messages. HLRSequence is an array where each array element represents a response.

Each parameter available for HLR configuration is also available for HLRSequence.

The following rules apply to use of the HLRSequence parameter:

- If HLRSequence has one response, it is used for all responses.
- If HLRSequence has more than one response, the first request will be handled according to the first array element, the second request will be handled according to the second array element, and so on. When the **smsc** test tool reaches the end of the array, it loops back to the beginning of the array..
- If HLRSequence is absent or is present but has no responses, HLR is used for all responses.

The following example illustrates the use of the HLRSequence parameter for MAP as IS-41:

```
# smsc.cfg
SMSC = {
    Protocol = "IS41"
    IS41 = {
        Action = "ACK"
        HLRSequence = [
            {
```

```
                Action = 0
                NatureOfNumber = "international"
                RoutingAddress = "6449393400"
            }
            {
                Action = "NACK"
                AccessDeniedReason = 2
                NatureOfNumber = "international"
                RoutingAddress = "6449393400"
            }
        ]
    }
}
```

Specifying AccessDeniedReason Values

Use the values in Table 4-1 to specify values for the AccessDeniedReason parameter.

Table 4-1 AccessDeniedReason Values

Value	Meaning
0	NotUsed
1	Denied
2	Postponed
3	Unavailable
4	Invalid

tcapInterfaceServiceKey for XMS

The tcapInterfaceServiceKey parameter specifies the service key that XMS uses to communicate with the **smsc** test tool. This service key must match the `SERVICEKEY` parameter for the SMSC interface in the **SLEE.cfg** file.

The following example shows the tcapInterfaceServiceKey parameter in the **smsc.cfg** file:

```
# eserv.config
XMS = {
xmsTrigger = {
plugins = [
# IS41 plugin
{
lib = "libxmsiIS41.so"
pluginId = 1
config = {
# ....
TDMA = {
xmsPointCode = 200
tcapInterfaceServiceKey = 42
# ....
}
CDMA = {
xmsPointCode = 201
tcapInterfaceServiceKey = 42
}
```

```
}
} ] } }
```

SMS CauseCode Mapping

Use the values in Table 4-2 to specify the SMS CauseCode values.

Table 4-2 SMS CauseCode Values

Numeric Value	String Value
0	AddressVacant
1	AddressTranslationFailure
2	NetworkResourceShortage
3	NetworkFailure
4	InvalidTeleserviceID
5	OtherNetworkProblem
6-31	Reserved, treated as OtherNetworkProblem
32	NoPageResponse
33	DestinationBusy
34	NoAcknowledgement
35	DestinationResourceShortage
36	SMSDeliveryPostponed
37	DestinationOutOfService
38	DestinationNoLongerAtThisAddress
39	OtherTerminalProblem
40-47	Reserved, treated as OtherTerminalProblem
48-63	Reserved, treated as SMSDeliveryPostponed
64	RadioInterfaceResourceShortage
65	RadioInterfaceIncompatibility
66	OtherRadioInterfaceProblem
67-95	Reserved, treated as OtherRadioInterfaceProblem
96	EncodingProblem
97	SMSOriginationDenied
98	SMSTerminationDenied
99	SupplementaryServiceNotSupported
100	SMSNotSupported
101	Reserved
102	MissingExpectedParameter
103	MissingMandatoryParameter
104	UnrecognizedParameterValue
105	UnexpectedParameter Value
106	UserDataSizeError

Table 4-2 (Cont.) SMS CauseCode Values

Numeric Value	String Value
107	OtherGeneralProblems

A

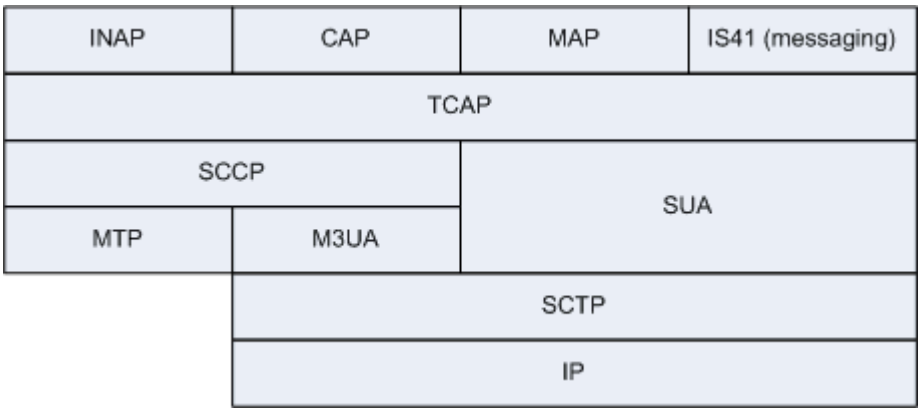
About the SS7 Protocol Suite

The SS7 protocol suite is a set of telephony signaling protocols that are used to establish and terminate telephone calls on public switched telephone networks. The SS7 protocol suite provides additional services as well, including number translation, local number portability, prepaid billing mechanisms, short message service (SMS), and a variety of other services.

Each protocol within a suite usually has a particular purpose. Such modularization makes design and assessment of the protocols easier. Because each protocol module usually communicates with two others, they are ordinarily considered as layers in a protocol stack. The lowest-layer protocol performs the low-level, physical interaction with the network hardware. Higher layers add more features.

Figure A-1 illustrates the SS7 protocol suite:

Figure A-1 SS7 Protocol Suite



The INAP Protocol

The INAP protocol is the signalling protocol that is used in Intelligent Networks (INs). INAP was developed by the International Telecommunications Union (ITU), and is recognized as an international standard. The functionality of INAP has been defined and implemented by the ITU in segments called capability sets. The first version was Capability Set 1 (CS-1) and Capability Set 2 (CS-2) is currently available.

INAP communicates between a service switching point (SSP), network media resources (intelligent peripherals), and a centralized network database called a service control point (SCP). The SCP encompasses operator or third-party-derived service logic programs and data.

The CAP Protocol

The CAMEL Application Part (CAP) protocol is a signalling protocol in the IN architecture and is layered on top of the TCAP protocol. It makes possible the implementation of carrier-grade,

value added services like unified messaging services, prepaid services, fraud control, and Freephone (800 number calls) in both the Global System for Mobile Communication (GSM) voice and General Packet Radio Service (GPRS) data networks. CAMEL is a means of adding intelligent applications to mobile networks. It builds upon established practices in the fixed-line telephony business that are generally considered part of the INAP CS-2 protocol.

The MAP Protocol

The Mobile Application Part (MAP) protocol supplies an application layer for nodes in the following networks:

- GSM (mobile) networks
- Universal Mobile Telecommunications System (UMTS) networks
- GPRS networks

The nodes in these networks use the MAP protocol to communicate with each other so they can provide services to mobile phone users. These services include mobility services such as location management to support roaming, call handling, SMS for text messaging, packet data protocol (PDP) services for GPRS, and operation and maintenance, as well as other services.

The IS-41 Protocol

The Interim Standard 41 protocol enables mobile, cellular telecommunications operations between different networks. It is similar to GSM and supports capabilities such as handover between networks, roaming authentication, and SMS delivery. It includes the Visitor Location Register (VLR) and Home Location Register (HLR) databases.

The TCAP Protocol

The TCAP protocol provides a presentation layer that facilitates the distribution of intelligent network services. The presentation layer deals with data format, operating system compatibility, and encapsulating data to send over the network. Fundamentally, TCAP simplifies simultaneous communications between subsystems on the same machines by using transaction IDs to associate multiple messages with a particular transaction.

In intelligent networks TCAP transports INAP and in mobile phone networks it transports MAP. See "Using TCAP Primitives" for more information.

The SCCP Protocol

The Signaling Connection Control Part (SCCP) protocol is a routing protocol that routes TCAP messages to their proper database. SCCP provides connectionless and connection-oriented network services. SCCP provides subsystem numbers that enable messages to be addressed to specific applications or subsystems at signaling points. SCCP is the transport layer for TCAP-based services such as calling card, local number portability, wireless roaming, personal communications services (PCS), and freephone (800 numbers).

The M3UA Protocol

M3UA stands for Message Transfer Part Level 3 (MTP3) User Adaptation Layer. The M3UA protocol enables the SS7 protocol User Part SCCP, as well as others, to run over internet protocol instead of telephony equipment. The M3UA protocol is generally transmitted by using the services of Stream Control Transmission Protocol (SCTP).

The SUA Protocol

SUA stands for the SCCP User Adaptation layer. The SUA protocol facilitates the transfer of SCCP user messages, such as TCAP, between the signalling gateway and the application server process (ASP).

The SCTP Protocol

The Stream Control Transmission Protocol (SCTP) is a transport-layer protocol that delivers in-sequence messages. It performs path selection and provides fail-over support for duplicated paths in the network.

The SCTP protocol was originally designed to transport telephony over the internet, but it has evolved to have other purposes as well.

The Internet Protocol

The Internet Protocol (IP) provides routing for data packets from source to destination hosts based on IP addresses. It facilitates the internetworking that constitutes the internet and defines structures that enclose data and add the source and destination addresses. Because it is often used together with the Transport Control Protocol, it is frequently referred to as TCP/IP. It runs on top of data link interfaces such as Ethernet and Wi-Fi, operating at layer 3 of the OSI model, which is the network layer. The network layer provides routing and switching functionality to transmit data between nodes.

B

Supported Protocol Fields for mipt

This appendix lists the fields supported by the **mipt** utility for each of the protocols that you can use:

Supported Fields for the Diameter Protocol

Following is a list of the Diameter protocol fields that the **mipt** utility supports:

3gpp_abort_cause
3gpp_acceptable_service_info
3gpp_access_network_charging_address
3gpp_access_network_charging_identifier
3gpp_access_network_charging_identifier_gx
3gpp_access_network_charging_identifier_value
3gpp_access_network_information
3gpp_account_expiration
3gpp_accumulated_cost
3gpp_adaptations
3gpp_additional_content_information
3gpp_additional_mbms_trace_info
3gpp_additional_type_information
3gpp_addressee_type
3gpp_address_data
3gpp_address_domain
3gpp_address_type
3gpp_af_application_identifier
3gpp_af_charging_identifier
3gpp_af_correlation_information
3gpp_af_signalling_protocol
3gpp_allocation_retention_priority
3gpp_alternate_charged_party_address

3gpp_alternative_apn
3gpp_an_gw_address
3gpp_aoc_cost_information
3gpp_aoc_format
3gpp_aoc_information
3gpp_aoc_request_type
3gpp_aoc_service
3gpp_aoc_service_obligatory_type
3gpp_aoc_service_type
3gpp_aoc_subscription_information
3gpp_apn_aggregated_max_bitrate_dl
3gpp_apn_aggregated_max_bitrate_ul
3gpp_application_provided_called_party_address
3gpp_application_server
3gpp_application_server_information
3gpp_application_service_provider_identity
3gpp_applic_id
3gpp_associated_party_address
3gpp_associated_uri
3gpp_authorised_qos
3gpp_aux_applic_info
3gpp_base_time_interval
3gpp_bearer_control_mode
3gpp_bearer_identifier
3gpp_bearer_operation
3gpp_bearer_service
3gpp_bearer_usage
3gpp_billing_information
3gpp_called_asserted_identity
3gpp_called_party_address
3gpp_calling_party_address
3gpp_carrier_select_routing_information
3gpp_cause_code
3gpp_cg_address
3gpp_change_condition
3gpp_change_time
3gpp_charged_party
3gpp_charging_characteristics_selection_mode
3gpp_charging_correlation_indicator
3gpp_charging_id
3gpp_charging_rule_base_name
3gpp_charging_rule_definition
3gpp_charging_rule_install
3gpp_charging_rule_name
3gpp_charging_rule_remove
3gpp_charging_rule_report
3gpp_class_identifier
3gpp_client_address
3gpp_cn_ip_multicast_distribution

3gpp_coa_information
3gpp_coa_ip_address
3gpp_codec_data
3gpp_content_class
3gpp_content_disposition
3gpp_content_length
3gpp_content_size
3gpp_content_type
3gpp_csg_access_mode
3gpp_csg_information_reporting
3gpp_csg_membership_indication
3gpp_cug_information
3gpp_current_tariff
3gpp_data_coding_scheme
3gpp_dcd_information
3gpp_default_eps_bearer_qos
3gpp_deferred_location_event_type
3gpp_delivery_report
3gpp_delivery_report_requested
3gpp_diagnostics
3gpp_domain_name
3gpp_drm_content
3gpp_dynamic_address_flag
3gpp_dynamic_address_flag_extension
3gpp_early_media_description
3gpp_envelope
3gpp_envelope_end_time
3gpp_envelope_reporting
3gpp_envelope_start_time
3gpp_event
3gpp_event_charging_timestamps
3gpp_event_report_indication
3gpp_event_trigger
3gpp_event_type
3gpp_experimental_result_code
3gpp_expires
3gpp_file_repair_supported
3gpp_flows
3gpp_flow_description
3gpp_flow_direction
3gpp_flow_information
3gpp_flow_label
3gpp_flow_number
3gpp_flow_status
3gpp_flow_usage
3gpp_ggsn_address
3gpp_guaranteed_bitrate_dl
3gpp_guaranteed_bitrate_ul
3gpp_imsi

3gpp_imsi_unauthenticated_flag
3gpp_ims_application_reference_identifier
3gpp_ims_charging_identifier
3gpp_ims_communication_service_identifier
3gpp_ims_information
3gpp_im_information
3gpp_incoming_trunk_group_id
3gpp_incremental_cost
3gpp_initial_ims_charging_identifier
3gpp_initial_recipient_address
3gpp_interface_id
3gpp_interface_port
3gpp_interface_text
3gpp_interface_type
3gpp_inter_operator_identifier
3gpp_ip_can_type
3gpp_ip_realm_default_indication
3gpp_lcs_client_dialed_by_ms
3gpp_lcs_client_external_id
3gpp_lcs_client_id
3gpp_lcs_client_name
3gpp_lcs_client_type
3gpp_lcs_data_coding_scheme
3gpp_lcs_format_indicator
3gpp_lcs_information
3gpp_lcs_name_string
3gpp_lcs_requestor_id
3gpp_lcs_requestor_id_string
3gpp_local_gw_inserted_indication
3gpp_local_sequence_number
3gpp_location_estimate
3gpp_location_estimate_type
3gpp_location_type
3gpp_low_balance_indication
3gpp_low_priority_indicator
3gpp_maximum_bandwidth
3gpp_max_bandwidth_ul
3gpp_max_requested_bandwidth_dl
3gpp_max_supported_bandwidth_dl
3gpp_max_supported_bandwidth_ul
3gpp_mbms_2g_3g_indicator
3gpp_mbms_access_indicator
3gpp_mbms_bmsc_ssm_ipv6_address
3gpp_mbms_bmsc_ssm_ip_address
3gpp_mbms_bmsc_ssm_udp_port
3gpp_mbms_counting_information
3gpp_mbms_flow_identifier
3gpp_mbms_ggsn_address
3gpp_mbms_ggsn_ipv6_address

3gpp_mbms_gw_address
3gpp_mbms_gw_ssm_ipv6_address
3gpp_mbms_gw_ssm_ip_address
3gpp_mbms_gw_udp_port
3gpp_mbms_gw_udp_port_indicator
3gpp_mbms_hc_indicator
3gpp_mbms_information
3gpp_mbms_service_area
3gpp_mbms_service_type
3gpp_mbms_session_duration
3gpp_mbms_session_identity
3gpp_mbms_session_repetition_number
3gpp_mbms_startstop_indication
3gpp_mbms_time_to_data_transfer
3gpp_mbms_user_data_mode_indication
3gpp_mbms_user_service_type
3gpp_media_component_description
3gpp_media_component_number
3gpp_media_initiator_flag
3gpp_media_initiator_party
3gpp_media_sub_component
3gpp_media_type
3gpp_message_body
3gpp_message_class
3gpp_message_id
3gpp_message_size
3gpp_message_type
3gpp_metering_method
3gpp_min_requested_bandwidth_dl
3gpp_min_requested_bandwidth_ul
3gpp_mm10_recipient_address
3gpp_mmbox_storage_requested
3gpp_mms_information
3gpp_mmtel_information
3gpp_mm_content_type
3gpp_monitoring_key
3gpp_mps_identifier
3gpp_msisdn
3gpp_network_request_support
3gpp_next_tariff
3gpp_node_functionality
3gpp_node_id
3gpp_number_of_diversions
3gpp_number_of_messages_sent
3gpp_number_of_participants
3gpp_number_of_received_talk_bursts
3gpp_number_of_talk_bursts
3gpp_number_portability_routing_information
3gpp_offline

3gpp_offline_charging
3gpp_online
3gpp_online_charging_flag
3gpp_originating_interface
3gpp_originating_ioi
3gpp_originator
3gpp_originator_address
3gpp_originator_received_address
3gpp_originator_sccp_address
3gpp_outgoing_session_id
3gpp_outgoing_trunk_group_id
3gpp_packet_filter_content
3gpp_packet_filter_identifier
3gpp_packet_filter_information
3gpp_packet_filter_operation
3gpp_packet_filter_usage
3gpp_participants_involved
3gpp_participant_access_priority
3gpp_participant_action_type
3gpp_participant_group
3gpp_pcc_rule_status
3gpp_pdg_address
3gpp_pdg_charging_id
3gpp_pdn_connection_charging_id
3gpp_pdn_connection_id
3gpp_pdp_address
3gpp_pdp_address_prefix_length
3gpp_pdp_context_type
3gpp_poc_change_condition
3gpp_poc_change_time
3gpp_poc_controlling_address
3gpp_poc_event_type
3gpp_poc_group_name
3gpp_poc_information
3gpp_poc_server_role
3gpp_poc_session_id
3gpp_poc_session_initiation_type
3gpp_poc_session_type
3gpp_poc_user_role
3gpp_poc_user_role_ids
3gpp_poc_user_role_info_units
3gpp_positioning_data
3gpp_precedence
3gpp_preemption_vulnerability
3gpp_preferred_aoc_currency
3gpp_pre_emption_capability
3gpp_priority
3gpp_priority_level
3gpp_ps_append_free_format_data

3gpp_ps_free_format_data
3gpp_ps_furnish_charging_information
3gpp_ps_information
3gpp_qos_class_identifier
3gpp_qos_information
3gpp_qos_negotiation
3gpp_qos_rule_base_name
3gpp_qos_rule_definition
3gpp_qos_rule_install
3gpp_qos_rule_name
3gpp_qos_rule_remove
3gpp_qos_rule_report
3gpp_qos_upgrade
3gpp_quota_consumption_time
3gpp_quota_holding_time
3gpp_rai
3gpp_rate_element
3gpp_rat_type
3gpp_read_reply
3gpp_read_reply_report_requested
3gpp_real_time_tariff_information
3gpp_reason_code
3gpp_received_talk_burst_time
3gpp_received_talk_burst_volume
3gpp_recipient_address
3gpp_recipient_info
3gpp_recipient_received_address
3gpp_recipient_sccp_address
3gpp_refund_information
3gpp_remaining_balance
3gpp_reply_applic_id
3gpp_reply_path_requested
3gpp_reporting_level
3gpp_reporting_reason
3gpp_requested_party_address
3gpp_required_mbms_bearer_capabilities
3gpp_resource_allocation_notification
3gpp_result_recipient_address
3gpp_revalidation_time
3gpp_role_of_node
3gpp_routeing_address
3gpp_routeing_address_resolution
3gpp_routing_filter
3gpp_routing_ip_address
3gpp_routing_rule_definition
3gpp_routing_rule_identifier
3gpp_routing_rule_install
3gpp_routing_rule_remove
3gpp_rr_bandwidth

3gpp_rs_bandwidth
3gpp_rule_activation_time
3gpp_rule_deactivation_time
3gpp_rule_failure_code
3gpp_scale_factor
3gpp_sdp_answer_timestamp
3gpp_sdp_media_component
3gpp_sdp_media_description
3gpp_sdp_media_name
3gpp_sdp_offer_timestamp
3gpp_sdp_session_description
3gpp_sdp_timestamps
3gpp_sdp_type
3gpp_security_parameter_index
3gpp_sender_address
3gpp_sender_visibility
3gpp_sequence_number
3gpp_served_party_ip_address
3gpp_served_user_identity
3gpp_service_data_container
3gpp_service_generic_information
3gpp_service_id
3gpp_service_information
3gpp_service_info_status
3gpp_service_key
3gpp_service_mode
3gpp_service_specific_data
3gpp_service_specific_info
3gpp_service_specific_type
3gpp_service_type
3gpp_service_urn
3gpp_serving_node_type
3gpp_session_linking_indicator
3gpp_session_release_cause
3gpp_sgsn_address
3gpp_sgsn_mcc_mnc
3gpp_sgw_address
3gpp_sgw_change
3gpp_sip_forking_indication
3gpp_sip_method
3gpp_sip_request_timestamp
3gpp_sip_request_timestamp_fraction
3gpp_sip_response_timestamp
3gpp_sip_response_timestamp_fraction
3gpp_smsc_address
3gpp_sms_information
3gpp_sms_node
3gpp_sm_discharge_time
3gpp_sm_message_type

3gpp_sm_protocol_id
3gpp_sm_service_type
3gpp_sm_status
3gpp_sm_user_data_header
3gpp_specific_action
3gpp_sponsored_connectivity_data
3gpp_sponsor_identity
3gpp_start_time
3gpp_status
3gpp_status_code
3gpp_status_text
3gpp_stop_time
3gpp_submission_time
3gpp_subscriber_role
3gpp_supplementary_service
3gpp_talk_burst_exchange
3gpp_talk_burst_time
3gpp_talk_burst_volume
3gpp_tariff_information
3gpp_tariff_xml
3gpp_terminating_ioi
3gpp_tft_filter
3gpp_tft_packet_filter_information
3gpp_time_first_usage
3gpp_time_last_usage
3gpp_time_quota_mechanism
3gpp_time_quota_threshold
3gpp_time_quota_type
3gpp_time_stamps
3gpp_time_usage
3gpp_tmgi
3gpp_token_text
3gpp_tos_traffic_class
3gpp_traffic_data_volumes
3gpp_transcoder_inserted_indication
3gpp_trigger
3gpp_trigger_event
3gpp_trigger_type
3gpp_trunk_group_id
3gpp_tunnel_header_filter
3gpp_tunnel_header_length
3gpp_tunnel_information
3gpp_type_number
3gpp_unit_cost
3gpp_unit_quota_threshold
3gpp_usage_monitoring_information
3gpp_usage_monitoring_level
3gpp_usage_monitoring_report
3gpp_usage_monitoring_support

3gpp_user_csg_information
3gpp_user_location_information
3gpp_user_participating_type
3gpp_user_session_id
3gpp_vasp_id
3gpp_vas_id
3gpp_volume_quota_threshold
3gpp_wag_address
3gpp_wag_plmn_id
3gpp_wlan_information
3gpp_wlan_radio_container
3gpp_wlan_session_id
3gpp_wlan_technology
3gpp_wlan_ue_local_ipaddress
abort_session_answer
abort_session_request
ACA
ACCOUNTING
accounting_answer
accounting_realtime_required
accounting_record_number
accounting_record_type
accounting_request
accounting_session_id
accounting_sub_session_id
acct_application_id
acct_interim_interval
acct_multi_session_id
ACR
ALLOW_SERVICE
ALL_APPLICATION
ALL_HOST
ALL_REALM
ALL_SESSION
ALL_USER
application_id
art_AUTHORIZE_AUTHENTICATE
art_AUTHORIZE_ONLY
ASA
ASR
AUTHENTICATE_ONLY
authorization_lifetime
auth_application_id
auth_grace_period
auth_request_type
auth_session_state
balanceExpiry
balanceInfo
balanceLimitType

balanceMaxCredit
balanceType
balanceUnit
balanceUserValue
balanceValue
BUSY
capabilities_exchange_answer
capabilities_exchange_request
CCA
CCR
cc_correlation_id
cc_input_octets
cc_money
cc_output_octets
cc_request_number
cc_request_type
cc_service_specific_units
cc_session_failover
cc_sub_session_id
cc_time
cc_total_octets
cc_unit_type
CEA
CER
charge
chargeBalanceType
chargeBalanceUnit
chargeBalanceValue
chargeInfo
chargingStartTimestamp
CHECK_BALANCE
check_balance_result
class
command_flags
CONTINUE
cost_information
cost_unit
CREDIT_AUTHORIZATION
credit_control
credit_control_answer
credit_control_failure_handling
credit_control_request
currency_code
DELIVER_AND_GRANT
destination_host
destination_realm
device_watchdog_answer
device_watchdog_request
DIRECT_DEBITING

direct_debiting_failure_handling
disconnect_cause
disconnect_peer_answer
disconnect_peer_request
DONT_CACHE
DO_NOT_WANT_TO_TALK_TO_YOU
DPA
DPR
DWA
DWR
ELECTION_LOST
end_to_end_identifier
END_USER_IMSI
END_USER_NAI
END_USER_PRIVATE
END_USER_SIP_URI
ENOUGH_CREDIT
error_message
error_reporting_host
esg_address
esg_diamident
esg_diamuri
esg_enumerated
esg_grouped
esg_grouped_1
esg_grouped_2
esg_grouped_3
esg_grouped_4
esg_grouped_5
esg_integer32
esg_integer32_1
esg_integer32_2
esg_integer32_3
esg_integer64
esg_ipfiltrrule
esg_octetstring
esg_time
esg_unsigned32
esg_unsigned64
esg_utf8string
esg_utf8string_1
esg_utf8string_2
esg_utf8string_3
EVENT_RECORD
EVENT_REQUEST
event_timestamp
experimental_result
experimental_result_code
exponent

failed_avp
FAILOVER_NOT_SUPPORTED
FAILOVER_SUPPORTED
final_unit_action
final_unit_indication
firmware_revision
granted_service_unit
GRANT_AND_LOSE
GRANT_AND_STORE
g_s_u_pool_identifier
g_s_u_pool_reference
hop_by_hop_identifier
host_ip_address
IMEISV
inband_security_id
INITIAL_REQUEST
INPUT_OCTETS
INTERIM_RECORD
MAC
message_process_answer
message_process_request
Mobile_IP
MONEY
MPA
MPR
multiple_services_credit_control
multiple_services_indicator
MULTIPLE_SERVICES_NOT_SUPPORTED
MULTIPLE_SERVICES_SUPPORTED
multi_round_time_out
NASREQ
NO_CREDIT
NO_INBAND_SECURITY
NO_STATE_MAINTAINED
origin_host
origin_realm
origin_state_id
OUTPUT_OCTETS
PRICE_ENQUIRY
product_name
proxy_host
proxy_info
proxy_state
RAA
RAR
rart_AUTHORIZE_AUTHENTICATE
rart_AUTHORIZE_ONLY
rating_group
REALM_AND_APPLICATION

REBOOTING
REDIRECT
redirect_address_type
redirect_host
redirect_host_usage
redirect_max_cache_time
redirect_server
redirect_server_address
REFUND_ACCOUNT
REFUSE_SERVICE
Relay
requested_action
requested_service_unit
restriction_filter_rule
RESTRICT_ACCESS
result_code
RETRY_AND_TERMINATE
RE_AUTH
RE_AUTHORIZATION
re_auth_answer
re_auth_request
re_auth_request_type
route_record
sb_STR
service_context_id
service_identifier
service_parameter_info
service_parameter_type
service_parameter_value
SERVICE_SPECIFIC_UNITS
session_binding
session_id
session_server_failover
session_termination_answer
session_termination_request
session_timeout
STA
START_RECORD
STATE_MAINTAINED
STOP_RECORD
STR
subscription_id
subscription_id_data
subscription_id_type
supported_vendor_id
tariff_change_usage
tariff_time_change
TERMINATE
TERMINATE_OR_BUFFER

termination_cause
TERMINATION_REQUEST
TIME
TLS
topUpAmount
topUpValueDigits
topUpVoucherId
topUpVoucherNumber
topUpVoucherType
TOTAL_OCTETS
TRY_AGAIN
TRY_AGAIN_ALLOW_SERVICE
UNIT_AFTER_TARIFF_CHANGE
UNIT_BEFORE_TARIFF_CHANGE
UNIT_INDETERMINATE
unit_value
UPDATE_REQUEST
URL
used_service_unit
user_equipment_info
user_equipment_info_type
user_equipment_info_value
user_name
validity_time
value_digits
vendor_id
vendor_specific_application_id
voucherBalance
voucherInfo
voucherInfoBalanceExpiryExtension
voucherInfoBalanceExpiryExtensionPolicy
voucherInfoBalanceExpiryExtensionType
voucherInfoBalanceType
voucherInfoMissingBalancePolicy
voucherInfoNewBucket
voucherInfoReplaceBalance
voucherInfoValue
voucherInfoVoucher
voucherInfoWalletExpiryExtension
voucherInfoWalletExpiryExtensionPolicy
voucherInfoWalletExpiryExtensionType
voucherTypeName
walletActivationDate
walletExpiry
walletInfo
walletLastAccessed
walletMaxConcurrent
walletState
walletSysCurrency

walletUserCurrency

Base AVP Diameter Fields

Table B-1 lists separately the base AVP fields that the **mipt** utility supports for the Diameter protocol.

Table B-1 Base AVP Diameter Fields

Base AVP Diameter Field Name	MIPT Field Name
Acct-Interim-Interval	acct_Interim_interval
Accounting-Realtime-Required	accounting_realtime_required
Acct-Multi-Session-Id	acct_multi_session_id
Accounting-Record-Number	accounting_record_number
Accounting-Record-Type	accounting_record_type
Accounting-Session-Id	accounting_session_id
Accounting-Sub-Session-Id	accounting_sub_session_id
Acct-Application-Id	acct_application_id
Auth-Application-Id	auth_application_id
Auth-Request-Type	auth_request_type
Authorization-Lifetime	authorization_lifetime
Auth-Grace-Period	auth_grace_period
Auth-Session-State	auth_session_state
Re-Auth-Request-Type	re_auth_request_type
Class	class
Destination-Host	destination_host
Destination-Realm	destination_realm
Disconnect-Cause	disconnect_cause
Error-Message	error_message
Error-Reporting-Host	error_reporting_host
Event-Timestamp	event_timestamp
Experimental-Result	experimental_result
Experimental-Result-Code	experimental_result_code
Failed-AVP	failed_avp
Firmware-Revision	firmware_revision
Host-IP-Address	host_ip_address
Inband-Security-Id	inband_security_id
Multi-Round-Time-Out	multi_round_time_out
Origin-Host	origin_host
Origin-Realm	origin_realm
Origin-State-Id	origin_state_id
Product-Name	product_name

Table B-1 (Cont.) Base AVP Diameter Fields

Base AVP Diameter Field Name	MIPT Field Name
Proxy-Host	proxy_host
Proxy-Info	proxy_info
Proxy-State	proxy_state
Redirect-Host	redirect_host
Redirect-Host-Usage	redirect_host_usage
Redirect-Max-Cache-Time	redirect_max_cache_time
Result-Code	result_code
Route-Record	route_record
Session-Id	session_id
Session-Timeout	session_timeout
Session-Binding	session_binding
Session-Server-Failover	session_server_failover
Supported-Vendor-Id	supported_vendor_id
Termination-Cause	termination_cause
User-Name	user_name
Vendor-Id	vendor_id
Vendor-Specific-Application-Id	vendor_specific_application_id
CC-Correlation-Id	cc_correlation_id
CC-Input-Octets	cc_input_octets
CC-Money	cc_money
CC-Output-Octets	cc_output_octets
CC-Request-Number	cc_request_number
CC-Request-Type	cc_request_type
CC-Service-Specific-Units	cc_service_specific_units
CC-Session-Failover	cc_session_failover
CC-Sub-Session-Id	cc_sub_session_id
CC-Time	cc_time
CC-Total-Octets	cc_total_octets
CC-Unit-Type	cc_unit_type
Check-Balance-Result	check_balance_result
Cost-Information	cost_information
Cost-Unit	cost_unit
Credit-Control	credit_control
Credit-Control-Failure-Handling	credit_control_failure_handling
Currency-Code	currency_code
Direct-Debiting-Failure-Handling	direct_debiting_failure_handling
Exponent	exponent

Table B-1 (Cont.) Base AVP Diameter Fields

Base AVP Diameter Field Name	MIPT Field Name
Final-Unit-Action	final_unit_action
Final-Unit-Indication	final_unit_indication
Granted-Service-Unit	granted_service_unit
G-S-U-Pool-Identifier	g_s_u_pool_identifier
G-S-U-Pool-Reference	g_s_u_pool_reference
Multiple-Services-Credit-Control	multiple_services_credit_control
Multiple-Services-Indicator	multiple_services_indicator
Rating-Group	rating_group
Redirect-Address-Type	redirect_address_type
Redirect-Server	redirect_server
Redirect-Server-Address	redirect_server_address
Requested-Action	requested_action
Requested-Service-Unit	requested_service_unit
Restriction-Filter-Rule	restriction_filter_rule
Service-Context-Id	service_context_id
Service-Identifier	service_identifier
Service-Parameter-Info	service_parameter_info
Service-Parameter-Type	service_parameter_type
Service-Parameter-Value	service_parameter_value
Subscription-Id	subscription_id
Subscription-Id-Data	subscription_id_data
Subscription-Id-Type	subscription_id_type
Tariff-Change-Usage	tariff_change_usage
Tariff-Time-Change	tariff_time_change
Unit-Value	unit_value
Used-Service-Unit	used_service_unit
User-Equipment-Info	user_equipment_info
User-Equipment-Info-Type	user_equipment_info_type
User-Equipment-Info-Value	user_equipment_info_value
Value-Digits	value_digits
Validity-Time	validity_time

Vendor-Specific Diameter Fields

Table B-2 lists the vendor-specific fields that the **mipt** utility supports for the Diameter protocol.

Table B-2 Vendor-Specific Diameter Fields

Vendor-Specific Diameter Field Name	MIPT Field Name	Vendor ID
Abort-Cause	3gpp_abort_cause	10415
Acceptable-Service-Info	3gpp_acceptable_service_info	10415
Access-Network-Charging-Address	3gpp_access_network_charging_address	10415
Access-Network-Charging-Identifier	3gpp_access_network_charging_identifier	10415
Access-Network-Charging-Identifier-Gx	3gpp_access_network_charging_identifier_gx	10415
Access-Network-Charging-Identifier-Value	3gpp_access_network_charging_identifier_value	10415
Access-Network-Information	3gpp_access_network_information	10415
Account-Expiration	3gpp_account_expiration	10415
Accumulated-Cost	3gpp_accumulated_cost	10415
Adaptations	3gpp_adaptations	10415
Additional-Content-Information	3gpp_additional_content_information	10415
Additional-MBMS-Trace-Info	3gpp_additional_mbms_trace_info	10415
Additional-Type-Information	3gpp_additional_type_information	10415
Address-Data	3gpp_address_data	10415
Address-Domain	3gpp_address_domain	10415
Address-Type	3gpp_address_type	10415
Addressee-Type	3gpp_addressee_type	10415
AF-Application-Identifier	3gpp_af_application_identifier	10415
AF-Charging-Identifier	3gpp_af_charging_identifier	10415
AF-Correlation-Information	3gpp_af_correlation_information	10415
AF-Signalling-Protocol	3gpp_af_signalling_protocol	10415
Allocation-Retention-Priority	3gpp_allocation_retention_priority	10415
Alternate-Charged-Party-Address	3gpp_alternate_charged_party_address	10415
Alternative-APN	3gpp_alternative_apn	10415
AN-GW-Address	3gpp_an_gw_address	10415
AoC-Cost-Information	3gpp_aoc_cost_information	10415
AoC-Format	3gpp_aoc_format	10415
AoC-Information	3gpp_aoc_information	10415
AoC-Request-Type	3gpp_aoc_request_type	10415
AoC-Service	3gpp_aoc_service	10415
AoC-Service-Obligatory-Type	3gpp_aoc_service_obligatory_type	10415
AoC-Service-Type	3gpp_aoc_service_type	10415
AoC-Subscription-Information	3gpp_aoc_subscription_information	10415
APN-Aggregated-Max-Bitrate-DL	3gpp_apn_aggregated_max_bitrate_dl	10415
APN-Aggregated-Max-Bitrate-UL	3gpp_apn_aggregated_max_bitrate_ul	10415

Table B-2 (Cont.) Vendor-Specific Diameter Fields

Vendor-Specific Diameter Field Name	MIPT Field Name	Vendor ID
Applic-Id	3gpp_applic_id	10415
Application-Provided-Called-Party-Address	3gpp_application_provided_called_party_address	10415
Application-Server	3gpp_application_server	10415
Application-Server-Information	3gpp_application_server_information	10415
Application-Service-Provider-Identity	3gpp_application_service_provider_identity	10415
Associated-Party-Address	3gpp_associated_party_address	10415
Associated-URI	3gpp_associated_uri	10415
Authorised-QoS	3gpp_authorised_qos	10415
Aux-Applic-Info	3gpp_aux_applic_info	10415
Base-Time-Interval	3gpp_base_time_interval	10415
Bearer-Control-Mode	3gpp_bearer_control_mode	10415
Bearer-Identifier	3gpp_bearer_identifier	10415
Bearer-Operation	3gpp_bearer_operation	10415
Bearer-Service	3gpp_bearer_service	10415
Bearer-Usage	3gpp_bearer_usage	10415
Billing-Information	3gpp_billing_information	10415
Called-Asserted-Identity	3gpp_called_asserted_identity	10415
Called-Party-Address	3gpp_called_party_address	10415
Calling-Party-Address	3gpp_calling_party_address	10415
Carrier-Select-Routing-Information	3gpp_carrier_select_routing_information	10415
Cause-Code	3gpp_cause_code	10415
CG-Address	3gpp_cg_address	10415
Change-Condition	3gpp_change_condition	10415
Change-Time	3gpp_change_time	10415
Charged-Party	3gpp_charged_party	10415
Charging-Characteristics-Selection-Mode	3gpp_charging_characteristics_selection_mode	10415
Charging-Correlation-Indicator	3gpp_charging_correlation_indicator	10415
Charging-Rule-Base-Name	3gpp_charging_rule_base_name	10415
Charging-Rule-Definition	3gpp_charging_rule_definition	10415
Charging-Rule-Install	3gpp_charging_rule_install	10415
Charging-Rule-Name	3gpp_charging_rule_name	10415
Charging-Rule-Remove	3gpp_charging_rule_remove	10415
Charging-Rule-Report	3gpp_charging_rule_report	10415
Class-Identifier	3gpp_class_identifier	10415
Client-Address	3gpp_client_address	10415

Table B-2 (Cont.) Vendor-Specific Diameter Fields

Vendor-Specific Diameter Field Name	MIPT Field Name	Vendor ID
CN-IP-Multicast-Distribution	3gpp_cn_ip_multicast_distribution	10415
CoA-Information	3gpp_coa_information	10415
CoA-IP-Address	3gpp_coa_ip_address	10415
Codec-Data	3gpp_codec_data	10415
Content-Class	3gpp_content_class	10415
Content-Disposition	3gpp_content_disposition	10415
Content-Length	3gpp_content_length	10415
Content-Size	3gpp_content_size	10415
Content-Type	3gpp_content_type	10415
CSG-Access-Mode	3gpp_csg_access_mode	10415
CSG-Information-Reporting	3gpp_csg_information_reporting	10415
CSG-Membership-Indication	3gpp_csg_membership_indication	10415
CUG-Information	3gpp_cug_information	10415
Current-Tariff	3gpp_current_tariff	10415
Data-Coding-Scheme	3gpp_data_coding_scheme	10415
DCD-Information	3gpp_dcd_information	10415
Default-EPS-Bearer-QoS	3gpp_default_eps_bearer_qos	10415
Deferred-Location-Event-Type	3gpp_deferred_location_event_type	10415
Delivery-Report	3gpp_delivery_report	10415
Delivery-Report-Requested	3gpp_delivery_report_requested	10415
Diagnostics	3gpp_diagnostics	10415
Domain-Name	3gpp_domain_name	10415
DRM-Content	3gpp_drm_content	10415
Dynamic-Address-Flag	3gpp_dynamic_address_flag	10415
Dynamic-Address-Flag-Extension	3gpp_dynamic_address_flag_extension	10415
Early-Media-Description	3gpp_early_media_description	10415
Envelope	3gpp_envelope	10415
Envelope-End-Time	3gpp_envelope_end_time	10415
Envelope-Reporting	3gpp_envelope_reporting	10415
Envelope-Start-Time	3gpp_envelope_start_time	10415
Event	3gpp_event	10415
Event-Charging-TimeStamps	3gpp_event_charging_timestamps	10415
Event-Report-Indication	3gpp_event_report_indication	10415
Event-Trigger	3gpp_event_trigger	10415
Event-Type	3gpp_event_type	10415
Experimental-Result-Code	3gpp_experimental_result_code	10415
Expires	3gpp_expires	10415

Table B-2 (Cont.) Vendor-Specific Diameter Fields

Vendor-Specific Diameter Field Name	MIPT Field Name	Vendor ID
File-Repair-Supported	3gpp_file_repair_supported	10415
Flow-Description	3gpp_flow_description	10415
Flow-Direction	3gpp_flow_direction	10415
Flow-Information	3gpp_flow_information	10415
Flow-Label	3gpp_flow_label	10415
Flow-Number	3gpp_flow_number	10415
Flow-Status	3gpp_flow_status	10415
Flow-Usage	3gpp_flow_usage	10415
Flows	3gpp_flows	10415
GGSN-Address	3gpp_ggsn_address	10415
Guaranteed-Bitrate-DL	3gpp_guaranteed_bitrate_dl	10415
Guaranteed-Bitrate-UL	3gpp_guaranteed_bitrate_ul	10415
IM-Information	3gpp_im_information	10415
IMS-Application-Reference-Identifier	3gpp_ims_application_reference_identifier	10415
IMS-Charging-Identifier	3gpp_ims_charging_identifier	10415
IMS-Communication-Service-Identifier	3gpp_ims_communication_service_identifier	10415
IMS-Information	3gpp_ims_information	10415
IMSI-Unauthenticated-Flag	3gpp_imsi_unauthenticated_flag	10415
Incoming-Trunk-Group-Id	3gpp_incoming_trunk_group_id	10415
Incremental-Cost	3gpp_incremental_cost	10415
Initial-IMS-Charging-Identifier	3gpp_initial_ims_charging_identifier	10415
Initial-Recipient-Address	3gpp_initial_recipient_address	10415
Inter-Operator-Identifier	3gpp_inter_operator_identifier	10415
Interface-Id	3gpp_interface_id	10415
Interface-Port	3gpp_interface_port	10415
Interface-Text	3gpp_interface_text	10415
Interface-Type	3gpp_interface_type	10415
IP-CAN-Type	3gpp_ip_can_type	10415
IP-Realm-Default-Indication	3gpp_ip_realm_default_indication	10415
LCS-Client-Dialed-By-MS	3gpp_lcs_client_dialed_by_ms	10415
LCS-Client-External-Id	3gpp_lcs_client_external_id	10415
LCS-Client-Id	3gpp_lcs_client_id	10415
LCS-Client-Name	3gpp_lcs_client_name	10415
LCS-Client-Type	3gpp_lcs_client_type	10415
LCS-Data-Coding-Scheme	3gpp_lcs_data_coding_scheme	10415

Table B-2 (Cont.) Vendor-Specific Diameter Fields

Vendor-Specific Diameter Field Name	MIPT Field Name	Vendor ID
LCS-Format-Indicator	3gpp_lcs_format_indicator	10415
LCS-Information	3gpp_lcs_information	10415
LCS-Name-String	3gpp_lcs_name_string	10415
LCS-Requestor-Id	3gpp_lcs_requestor_id	10415
LCS-Requestor-Id-String	3gpp_lcs_requestor_id_string	10415
Local-GW-Inserted-Indication	3gpp_local_gw_inserted_indication	10415
Local-Sequence-Number	3gpp_local_sequence_number	10415
Location-Estimate	3gpp_location_estimate	10415
Location-Estimate-Type	3gpp_location_estimate_type	10415
Location-Type	3gpp_location_type	10415
Low-Balance-Indication	3gpp_low_balance_indication	10415
Low-Priority-Indicator	3gpp_low_priority_indicator	10415
Max-Bandwidth-UL	3gpp_max_bandwidth_ul	10415
Max-Requested-Bandwidth-DL	3gpp_max_requested_bandwidth_dl	10415
Max-Supported-Bandwidth-DL	3gpp_max_supported_bandwidth_dl	10415
Max-Supported-Bandwidth-UL	3gpp_max_supported_bandwidth_ul	10415
Maximum-Bandwidth	3gpp_maximum_bandwidth	10415
MBMS-2G-3G-Indicator	3gpp_mbms_2g_3g_indicator	10415
MBMS-Access-Indicator	3gpp_mbms_access_indicator	10415
MBMS-BMSC-SSM-IP-Address	3gpp_mbms_bmsc_ssm_ip_address	10415
MBMS-BMSC-SSM-IPv6-Address	3gpp_mbms_bmsc_ssm_ipv6_address	10415
MBMS-BMSC-SSM-UDP-Port	3gpp_mbms_bmsc_ssm_udp_port	10415
MBMS-Counting-Information	3gpp_mbms_counting_information	10415
MBMS-Flow-Identifier	3gpp_mbms_flow_identifier	10415
MBMS-GGSN-Address	3gpp_mbms_ggsn_address	10415
MBMS-GGSN-IPv6-Address	3gpp_mbms_ggsn_ipv6_address	10415
MBMS-GW-Address	3gpp_mbms_gw_address	10415
MBMS-GW-SSM-IP-Address	3gpp_mbms_gw_ssm_ip_address	10415
MBMS-GW-SSM-IPv6-Address	3gpp_mbms_gw_ssm_ipv6_address	10415
MBMS-GW-UDP-Port	3gpp_mbms_gw_udp_port	10415
MBMS-GW-UDP-Port-Indicator	3gpp_mbms_gw_udp_port_indicator	10415
MBMS-HC-Indicator	3gpp_mbms_hc_indicator	10415
MBMS-Information	3gpp_mbms_information	10415
MBMS-Service-Area	3gpp_mbms_service_area	10415
MBMS-Service-Type	3gpp_mbms_service_type	10415
MBMS-Session-Duration	3gpp_mbms_session_duration	10415
MBMS-Session-Identity	3gpp_mbms_session_identity	10415

Table B-2 (Cont.) Vendor-Specific Diameter Fields

Vendor-Specific Diameter Field Name	MIPT Field Name	Vendor ID
MBMS-Session-Repetition-Number	3gpp_mbms_session_repetition_number	10415
MBMS-StartStop-Indication	3gpp_mbms_startstop_indication	10415
MBMS-Time-To-Data-Transfer	3gpp_mbms_time_to_data_transfer	10415
MBMS-User-Data-Mode-Indication	3gpp_mbms_user_data_mode_indication	10415
MBMS-User-Service-Type	3gpp_mbms_user_service_type	10415
Media-Component-Description	3gpp_media_component_description	10415
Media-Component-Number	3gpp_media_component_number	10415
Media-Initiator-Flag	3gpp_media_initiator_flag	10415
Media-Initiator-Party	3gpp_media_initiator_party	10415
Media-Sub-Component	3gpp_media_sub_component	10415
Media-Type	3gpp_media_type	10415
Message-Body	3gpp_message_body	10415
Message-Class	3gpp_message_class	10415
Message-Id	3gpp_message_id	10415
Message-Size	3gpp_message_size	10415
Message-Type	3gpp_message_type	10415
Metering-Method	3gpp_metering_method	10415
Min-Requested-Bandwidth-DL	3gpp_min_requested_bandwidth_dl	10415
Min-Requested-Bandwidth-UL	3gpp_min_requested_bandwidth_ul	10415
MM10-Recipient-Address	3gpp_mm10_recipient_address	10415
MM-Content-Type	3gpp_mm_content_type	10415
MMBox-Storage-Requested	3gpp_mmbox_storage_requested	10415
MMS-Information	3gpp_mms_information	10415
MMTel-Information	3gpp_mmtel_information	10415
Monitoring-Key	3gpp_monitoring_key	10415
MPS-Identifier	3gpp_mps_identifier	10415
Network-Request-Support	3gpp_network_request_support	10415
Next-Tariff	3gpp_next_tariff	10415
Node-Functionality	3gpp_node_functionality	10415
Node-Id	3gpp_node_id	10415
Number-Of-Diversions	3gpp_number_of_diversions	10415
Number-Of-Messages-Sent	3gpp_number_of_messages_sent	10415
Number-Of-Participants	3gpp_number_of_participants	10415
Number-Of-Received-Talk-Bursts	3gpp_number_of_received_talk_bursts	10415
Number-Of-Talk-Bursts	3gpp_number_of_talk_bursts	10415
Number-Portability-Routing-Information	3gpp_number_portability_routing_information	10415

Table B-2 (Cont.) Vendor-Specific Diameter Fields

Vendor-Specific Diameter Field Name	MIPT Field Name	Vendor ID
Offline	3gpp_offline	10415
Offline-Charging	3gpp_offline_charging	10415
Online	3gpp_online	10415
Online-Charging-Flag	3gpp_online_charging_flag	10415
Originating-Interface	3gpp_originating_interface	10415
Originating-IOI	3gpp_originating_ioi	10415
Originator	3gpp_originator	10415
Originator-Address	3gpp_originator_address	10415
Originator-Received-Address	3gpp_originator_received_address	10415
Originator-SCCP-Address	3gpp_originator_sccp_address	10415
Outgoing-Session-Id	3gpp_outgoing_session_id	10415
Outgoing-Trunk-Group-Id	3gpp_outgoing_trunk_group_id	10415
Packet-Filter-Content	3gpp_packet_filter_content	10415
Packet-Filter-Identifier	3gpp_packet_filter_identifier	10415
Packet-Filter-Information	3gpp_packet_filter_information	10415
Packet-Filter-Operation	3gpp_packet_filter_operation	10415
Packet-Filter-Usage	3gpp_packet_filter_usage	10415
Participant-Access-Priority	3gpp_participant_access_priority	10415
Participant-Action-Type	3gpp_participant_action_type	10415
Participant-Group	3gpp_participant_group	10415
Participants-Involved	3gpp_participants_involved	10415
PCC-Rule-Status	3gpp_pcc_rule_status	10415
PDG-Address	3gpp_pdg_address	10415
PDG-Charging-Id	3gpp_pdg_charging_id	10415
PDN-Connection-Charging-Id	3gpp_pdn_connection_charging_id	10415
PDN-Connection-Id	3gpp_pdn_connection_id	10415
PDP-Address	3gpp_pdp_address	10415
PDP-Address-Prefix-Length	3gpp_pdp_address_prefix_length	10415
PDP-Context-Type	3gpp_pdp_context_type	10415
PoC-Change-Condition	3gpp_poc_change_condition	10415
PoC-Change-Time	3gpp_poc_change_time	10415
PoC-Controlling-Address	3gpp_poc_controlling_address	10415
PoC-Event-Type	3gpp_poc_event_type	10415
PoC-Group-Name	3gpp_poc_group_name	10415
PoC-Information	3gpp_poc_information	10415
PoC-Server-Role	3gpp_poc_server_role	10415
PoC-Session-Id	3gpp_poc_session_id	10415

Table B-2 (Cont.) Vendor-Specific Diameter Fields

Vendor-Specific Diameter Field Name	MIPT Field Name	Vendor ID
PoC-Session-Initiation-Type	3gpp_poc_session_initiation_type	10415
PoC-Session-Type	3gpp_poc_session_type	10415
PoC-User-Role	3gpp_poc_user_role	10415
PoC-User-Role-IDs	3gpp_poc_user_role_ids	10415
Poc-User-Role-Info-Units	3gpp_poc_user_role_info_units	10415
Positioning-Data	3gpp_positioning_data	10415
Pre-Emption-Capability	3gpp_pre_emption_capability	10415
Precedence	3gpp_precedence	10415
Preemption-Vulnerability	3gpp_preemption_vulnerability	10415
Preferred-AoC-Currency	3gpp_preferred_aoc_currency	10415
Priority	3gpp_priority	10415
Priority-Level	3gpp_priority_level	10415
PS-Append-Free-Format-Data	3gpp_ps_append_free_format_data	10415
PS-Free-Format-Data	3gpp_ps_free_format_data	10415
PS-Furnish-Charging-Information	3gpp_ps_furnish_charging_information	10415
PS-Information	3gpp_ps_information	10415
QoS-Class-Identifier	3gpp_qos_class_identifier	10415
QoS-Information	3gpp_qos_information	10415
QoS-Negotiation	3gpp_qos_negotiation	10415
QoS-Rule-Base-Name	3gpp_qos_rule_base_name	10415
QoS-Rule-Definition	3gpp_qos_rule_definition	10415
QoS-Rule-Install	3gpp_qos_rule_install	10415
QoS-Rule-Name	3gpp_qos_rule_name	10415
QoS-Rule-Remove	3gpp_qos_rule_remove	10415
QoS-Rule-Report	3gpp_qos_rule_report	10415
QoS-Upgrade	3gpp_qos_upgrade	10415
Quota-Consumption-Time	3gpp_quota_consumption_time	10415
Quota-Holding-Time	3gpp_quota_holding_time	10415
RAI	3gpp_rai	10415
RAT-Type	3gpp_rat_type	10415
Rate-Element	3gpp_rate_element	10415
Read-Reply	3gpp_read_reply	10415
Read-Reply-Report-Requested	3gpp_read_reply_report_requested	10415
Real-Time-Tariff-Information	3gpp_real_time_tariff_information	10415
Reason-Code	3gpp_reason_code	10415
Received-Talk-Burst-Time	3gpp_received_talk_burst_time	10415
Received-Talk-Burst-Volume	3gpp_received_talk_burst_volume	10415

Table B-2 (Cont.) Vendor-Specific Diameter Fields

Vendor-Specific Diameter Field Name	MIPT Field Name	Vendor ID
Recipient-Address	3gpp_recipient_address	10415
Recipient-Info	3gpp_recipient_info	10415
Recipient-Received-Address	3gpp_recipient_received_address	10415
Recipient-SCCP-Address	3gpp_recipient_sccp_address	10415
Refund-Information	3gpp_refund_information	10415
Remaining-Balance	3gpp_remaining_balance	10415
Reply-Applic-Id	3gpp_reply_applic_id	10415
Reply-Path-Requested	3gpp_reply_path_requested	10415
Reporting-Level	3gpp_reporting_level	10415
Reporting-Reason	3gpp_reporting_reason	10415
Requested-Party-Address	3gpp_requested_party_address	10415
Required-MBMS-Bearer-Capabilities	3gpp_required_mbms_bearer_capabilities	10415
Resource-Allocation-Notification	3gpp_resource_allocation_notification	10415
Result-Recipient-Address	3gpp_result_recipient_address	10415
Revalidation-Time	3gpp_revalidation_time	10415
Role-Of-Node	3gpp_role_of_node	10415
Routeing-Address	3gpp_routeing_address	10415
Routeing-Address-Resolution	3gpp_routeing_address_resolution	10415
Routing-Filter	3gpp_routing_filter	10415
Routing-IP-Address	3gpp_routing_ip_address	10415
Routing-Rule-Definition	3gpp_routing_rule_definition	10415
Routing-Rule-Identifier	3gpp_routing_rule_identifier	10415
Routing-Rule-Install	3gpp_routing_rule_install	10415
Routing-Rule-Remove	3gpp_routing_rule_remove	10415
RR-Bandwidth	3gpp_rr_bandwidth	10415
RS-Bandwidth	3gpp_rs_bandwidth	10415
Rule-Activation-Time	3gpp_rule_activation_time	10415
Rule-Deactivation-Time	3gpp_rule_deactivation_time	10415
Rule-Failure-Code	3gpp_rule_failure_code	10415
Scale-Factor	3gpp_scale_factor	10415
SDP-Answer-Timestamp	3gpp_sdp_answer_timestamp	10415
SDP-Media-Component	3gpp_sdp_media_component	10415
SDP-Media-Description	3gpp_sdp_media_description	10415
SDP-Media-Name	3gpp_sdp_media_name	10415
SDP-Offer-Timestamp	3gpp_sdp_offer_timestamp	10415
SDP-Session-Description	3gpp_sdp_session_description	10415
SDP-TimeStamps	3gpp_sdp_timestamps	10415

Table B-2 (Cont.) Vendor-Specific Diameter Fields

Vendor-Specific Diameter Field Name	MIPT Field Name	Vendor ID
SDP-Type	3gpp_sdp_type	10415
Security-Parameter-Index	3gpp_security_parameter_index	10415
Sender-Address	3gpp_sender_address	10415
Sender-Visibility	3gpp_sender_visibility	10415
Sequence-Number	3gpp_sequence_number	10415
Served-Party-IP-Address	3gpp_served_party_ip_address	10415
Served-User-Identity	3gpp_served_user_identity	10415
Service-Data-Container	3gpp_service_data_container	10415
Service-Generic-Information	3gpp_service_generic_information	10415
Service-Id	3gpp_service_id	10415
Service-Info-Status	3gpp_service_info_status	10415
Service-Information	3gpp_service_information	10415
Service-Key	3gpp_service_key	10415
Service-Mode	3gpp_service_mode	10415
Service-Specific-Data	3gpp_service_specific_data	10415
Service-Specific-Info	3gpp_service_specific_info	10415
Service-Specific-Type	3gpp_service_specific_type	10415
Service-Type	3gpp_service_type	10415
Service-URN	3gpp_service_urn	10415
Serving-Node-Type	3gpp_serving_node_type	10415
Session-Linking-Indicator	3gpp_session_linking_indicator	10415
Session-Release-Cause	3gpp_session_release_cause	10415
SGSN-Address	3gpp_sgsn_address	10415
SGW-Address	3gpp_sgw_address	10415
SGW-Change	3gpp_sgw_change	10415
SIP-Forking-Indication	3gpp_sip_forking_indication	10415
SIP-Method	3gpp_sip_method	10415
SIP-Request-Timestamp	3gpp_sip_request_timestamp	10415
SIP-Request-Timestamp-Fraction	3gpp_sip_request_timestamp_fraction	10415
SIP-Response-Timestamp	3gpp_sip_response_timestamp	10415
SIP-Response-Timestamp-Fraction	3gpp_sip_response_timestamp_fraction	10415
SM-Discharge-Time	3gpp_sm_discharge_time	10415
SM-Message-Type	3gpp_sm_message_type	10415
SM-Protocol-Id	3gpp_sm_protocol_id	10415
SM-Service-Type	3gpp_sm_service_type	10415
SM-Status	3gpp_sm_status	10415
SM-User-Data-Header	3gpp_sm_user_data_header	10415

Table B-2 (Cont.) Vendor-Specific Diameter Fields

Vendor-Specific Diameter Field Name	MIPT Field Name	Vendor ID
SMS-Information	3gpp_sms_information	10415
SMS-Node	3gpp_sms_node	10415
SMSC-Address	3gpp_smsc_address	10415
Specific-Action	3gpp_specific_action	10415
Sponsor-Identity	3gpp_sponsor_identity	10415
Sponsored-Connectivity-Data	3gpp_sponsored_connectivity_data	10415
Start-Time	3gpp_start_time	10415
Status	3gpp_status	10415
Status-Code	3gpp_status_code	10415
Status-Text	3gpp_status_text	10415
Stop-Time	3gpp_stop_time	10415
Submission-Time	3gpp_submission_time	10415
Subscriber-Role	3gpp_subscriber_role	10415
Supplementary-Service	3gpp_supplementary_service	10415
Talk-Burst-Exchange	3gpp_talk_burst_exchange	10415
Talk-Burst-Time	3gpp_talk_burst_time	10415
Talk-Burst-Volume	3gpp_talk_burst_volume	10415
Tariff-Information	3gpp_tariff_information	10415
Tariff-XML	3gpp_tariff_xml	10415
Terminating-IOI	3gpp_terminating_ioi	10415
TFT-Filter	3gpp_tft_filter	10415
TFT-Packet-Filter-Information	3gpp_tft_packet_filter_information	10415
Time-First-Usage	3gpp_time_first_usage	10415
Time-Last-Usage	3gpp_time_last_usage	10415
Time-Quota-Mechanism	3gpp_time_quota_mechanism	10415
Time-Quota-Threshold	3gpp_time_quota_threshold	10415
Time-Quota-Type	3gpp_time_quota_type	10415
Time-Stamps	3gpp_time_stamps	10415
Time-Usage	3gpp_time_usage	10415
TMGI	3gpp_tmgi	10415
Token-Text	3gpp_token_text	10415
ToS-Traffic-Class	3gpp_tos_traffic_class	10415
Traffic-Data-Volumes	3gpp_traffic_data_volumes	10415
Transcoder-Inserted-Indication	3gpp_transcoder_inserted_indication	10415
Trigger	3gpp_trigger	10415
Trigger-Event	3gpp_trigger_event	10415
Trigger-Type	3gpp_trigger_type	10415

Table B-2 (Cont.) Vendor-Specific Diameter Fields

Vendor-Specific Diameter Field Name	MIPT Field Name	Vendor ID
Trunk-Group-Id	3gpp_trunk_group_id	10415
Tunnel-Header-Filter	3gpp_tunnel_header_filter	10415
Tunnel-Header-Length	3gpp_tunnel_header_length	10415
Tunnel-Information	3gpp_tunnel_information	10415
Type-Number	3gpp_type_number	10415
Unit-Cost	3gpp_unit_cost	10415
Unit-Quota-Threshold	3gpp_unit_quota_threshold	10415
Usage-Monitoring-Information	3gpp_usage_monitoring_information	10415
Usage-Monitoring-Level	3gpp_usage_monitoring_level	10415
Usage-Monitoring-Report	3gpp_usage_monitoring_report	10415
Usage-Monitoring-Support	3gpp_usage_monitoring_support	10415
User-CSG-Information	3gpp_user_csg_information	10415
User-Participating-Type	3gpp_user_participating_type	10415
User-Session-Id	3gpp_user_session_id	10415
VAS-Id	3gpp_vas_id	10415
VASP-Id	3gpp_vasp_id	10415
Volume-Quota-Threshold	3gpp_volume_quota_threshold	10415
WAG-Address	3gpp_wag_address	10415
WAG-PLMN-Id	3gpp_wag_plmn_id	10415
WLAN-Information	3gpp_wlan_information	10415
WLAN-Radio-Container	3gpp_wlan_radio_container	10415
WLAN-Session-Id	3gpp_wlan_session_id	10415
WLAN-Technology	3gpp_wlan_technology	10415
WLAN-UE-Local-IPAddress	3gpp_wlan_ue_local_ipaddress	10415
eServGlobal-Address	esg_address	16247
eServGlobal-DiamIdent	esg_diamident	16247
eServGlobal-DiamURI	esg_diamuri	16247
eServGlobal-Enumerated	esg_enumerated	16247
eServGlobal-Grouped	esg_grouped	16247
eServGlobal-Integer32	esg_integer32	16247
eServGlobal-Integer64	esg_integer64	16247
eServGlobal-IPFiltrRule	esg_ipfiltrrule	16247
eServGlobal-OctetString	esg_octetstring	16247
eServGlobal-Time	esg_time	16247
eServGlobal-Unsigned32	esg_unsigned32	16247
eServGlobal-Unsigned64	esg_unsigned64	16247
eServGlobal-UTF8String	esg_utf8string	16247

Table B-2 (Cont.) Vendor-Specific Diameter Fields

Vendor-Specific Diameter Field Name	MIPT Field Name	Vendor ID
eServGlobal-Grouped	esg_grouped_1	16247
eServGlobal-Grouped	esg_grouped_2	16247
eServGlobal-Grouped	esg_grouped_3	16247
eServGlobal-Grouped	esg_grouped_4	16247
eServGlobal-Grouped	esg_grouped_5	16247
eServGlobal-UTF8String	esg_utf8string_1	16247
eServGlobal-UTF8String	esg_utf8string_2	16247
eServGlobal-UTF8String	esg_utf8string_3	16247
eServGlobal-Integer32	esg_integer32_1	16247
eServGlobal-Integer32	esg_integer32_2	16247
eServGlobal-Integer32	esg_integer32_3	16247
Balance-Expiry	balanceExpiry	16247
Balance-MaxCredit	balanceMaxCredit	16247
Balance-LimitType	balanceLimitType	16247
Balance-Unit	balanceUnit	16247
Balance-Value	balanceValue	16247
Balance-Value	balanceUserValue	16247
Balance-Type	balanceType	16247
Balance-Information	balanceInfo	16247
Wallet-Information	walletInfo	16247
Wallet-Expiry	walletExpiry	16247
Wallet-State	walletState	16247
Wallet-LastAccessed	walletLastAccessed	16247
Wallet-ActivationDate	walletActivationDate	16247
Wallet-MaxConcurrent	walletMaxConcurrent	16247
Wallet-SystemCurrency	walletSysCurrency	16247
Wallet-SystemCurrency	walletUserCurrency	16247
Charge-Information	chargeInfo	16247
Charge	charge	16247
Charge-Balance-Type	chargeBalanceType	16247
Charge-Balance-Value	chargeBalanceValue	16247
Charge-Balance-Unit	chargeBalanceUnit	16247
Voucher-Information	voucherInfo	16247
Voucher-Info-Voucher	voucherInfoVoucher	16247
Voucher-Info-Wallet-Expiry-Extension	voucherInfoWalletExpiryExtension	16247
Voucher-Info-Wallet-Expiry-Extension-Policy	voucherInfoWalletExpiryExtensionPolicy	16247

Table B-2 (Cont.) Vendor-Specific Diameter Fields

Vendor-Specific Diameter Field Name	MIPT Field Name	Vendor ID
Voucher-Info-Wallet-Expiry-Extension-Type	voucherInfoWalletExpiryExtensionType	16247
Voucher-Balance	voucherBalance	16247
Voucher-Info-Balance-Type	voucherInfoBalanceType	16247
Voucher-Info-Value	voucherInfoValue	16247
Voucher-Info-Balance-Expiry-Extension	voucherInfoBalanceExpiryExtension	16247
Voucher-Info-Voucher-Expiry-Extension-Policy	voucherInfoBalanceExpiryExtensionPolicy	16247
Voucher-Info-Balance-Expiry-Extension-Type	voucherInfoBalanceExpiryExtensionType	16247
Voucher-Info-New-Bucket	voucherInfoNewBucket	16247
Voucher-Info-Missing-Balance-Policy	voucherInfoMissingBalancePolicy	16247
Voucher-Info-Replace-Balance	voucherInfoReplaceBalance	16247
Voucher-Type	voucherTypeName	16247
Charging-Start-Timestamp	chargingStartTimestamp	16247
Top-Up-Voucher-Number	topUpVoucherNumber	16247
Top-Up-Voucher-Id	topUpVoucherId	16247
Top-Up-Amount	topUpAmount	16247
Voucher-Value-Digits	topUpValueDigits	16247
Top-Up-Voucher-Type	topUpVoucherType	16247
3GPP-IMSI	3gpp_imsi	10415
3GPP-Charging-Id	3gpp_charging_id	10415
3GPP-SGSN-MCC-MNC	3gpp_sgsn_mcc_mnc	10415
MSISDN	3gpp_msisdn	10415
User-Location-Information	3gpp_user_location_information	10415
Fail-If-Below-Threshold	awcc_fail_if_below_threshold	16247
Charging-Start-Timestamp	awcc_charging_start_timestamp	16247
Product-Type	awcc_product_type	16247
Policy-Control	awcc_policy_control	16247
Calling-Party-Presentation-Indicator	awcc_calling_party_presentation_indicator	16247
Data-Source-System	awcc_data_source_system	16247
Data-Application-Id	awcc_data_application_id	16247
Data-Transaction-Id	awcc_data_transaction_id	16247
Balance-Information	awcc_balance_information	16247
Balance-Type-Id	awcc_balance_type_id	16247
Balance-Expire-Date	awcc_balance_expire_date	16247

Supported Fields for the EMI Protocol

Following is a list of the EMI protocol fields that the **mipt** utility supports:

AC
ACK
AdC
AMsg
call_input
call_input_ack
call_input_nack
call_input_with_supplementary_services
call_input_with_supplementary_services_ack
call_input_with_supplementary_services_nack
CPg
CSUM
DCs
DD
DDT
delete_message
delete_message_ack
delete_message_nack
delivery_notification
delivery_notification_ack
delivery_notification_nack
deliver_sm
deliver_sm_ack
deliver_sm_nack
DSCTS
Dst
EC
GAs
HPLMN
inquiry_message
inquiry_message_ack
inquiry_message_nack
LAdC
LEN
LNPI
LPID
LPR
LRAd
LRC
LRq
LTON
LUR
MCLs
MMS

modify_sm
modify_sm_ack
modify_sm_nack
ms_message_transfer
ms_message_transfer_ack
ms_message_transfer_nack
MT
mt_alert
mt_alert_ack
mt_alert_nack
multiple_address_call_input
multiple_address_call_input_ack
multiple_address_call_input_nack
MVP
NACK
NAd
NAdC
NB
NMsg
NPID
NPL
NPWD
NRq
NT
OAdC
ONPI
OPID
OT
OTOA
OTON
O_R
PID
PR
provisioning_actions
provisioning_actions_ack
provisioning_actions_nack
PWD
RAd
RAds
RC
RES1
RES2
RES4
RES5
response_delete_message
response_delete_message_ack
response_delete_message_nack
response_inquiry_message
response_inquiry_message_ack

response_inquiry_message_nack
RP
RPI
RPID
RPLy
Rsn
SCTS
session_management
session_management_ack
session_management_nack
SM
STYP
submit_sm
submit_sm_ack
submit_sm_nack
TMsg
TRN
UR
VERS
VP
XSer

Supported Fields of the M3UA Protocol

Following is a list of the M3UA protocol fields that the **mipt** utility supports:

affected_point_code
ASPAC
ASPAC_ACK
ASPDN
ASPDN_ACK
ASPIA
ASPIA_ACK
ASPUP
ASPUP_ACK
asp_active
asp_active_acknowledgement
asp_down
asp_down_acknowledgement
asp_identifier
asp_inactive
asp_inactive_acknowledgement
asp_up
asp_up_acknowledgement
BEAT
BEAT_ACK
concerned_destination
congestion_indications
control_word

correlation_id
DATA
DAUD
DAVA
deregistration_request
deregistration_response
deregistration_result
deregistration_status
DEREG_REQ
DEREG_RSP
destination_available
destination_point_code
destination_restricted
destination_state_audit
destination_unavailable
destination_user_part_unavailable
diagnostic_info
DRST
DUNA
DUPU
ERR
error
error_code
heartbeat
heartbeat_acknowledgement
heartbeat_data
info_string
local_routing_key_identifier
network_appearance
notify
NTFY
originating_point_code
originating_point_code_list
payload_data
protocol_data
registration_request
registration_response
registration_result
registration_status
REG_REQ
REG_RSP
routing_context
routing_key
SCON
service_indicators
signalling_congestion
status
traffic_mode_type
user_cause

Supported Fields of the RADIUS Protocol

Following is a list of the RADIUS protocol fields that the **mipt** utility supports:

3GGP_CAMEL_CHARGING_INFO
3GGP_CG_ADDRESS
3GGP_GGSN_ADDRESS
3GGP_IMEISV
3GGP_MS_TIMEZONE
3GGP_SGSN_ADDRESS
3GGP_USER_LOCATION_INFO
3GPP_CAMEL_CHARGING_INFO
3GPP_CG_ADDRESS
3GPP_CG_IPV6_ADDRESS
3GPP_CHARGING_CHARACTERISTICS
3GPP_CHARGING_ID
3GPP_GGSN_ADDRESS
3GPP_GGSN_IPV6_ADDRESS
3GPP_GGSN_MCC_MNC
3GPP_GPRS_QOS_PROFILE
3GPP_IMEISV
3GPP_IMSI
3GPP_IMSI_MCC_MNC
3GPP_IPV6_DNS_SERVER
3GPP_MS_TIMEZONE
3GPP_NSAPI
3GPP_PDP_TYPE
3GPP_QOS_PROFILE
3GPP_RAT_TYPE
3GPP_SELECTION_MODE
3GPP_SESSION_STOP_INDICATOR
3GPP_SGSN_ADDRESS
3GPP_SGSN_IPV6_ADDRESS
3GPP_SGSN_MCC_MNC
3GPP_USER_LOCATION_INFO
access_accept
access_challenge
access_reject
access_request
accounting_container
accounting_request
accounting_response
accounting_stop_triggered_by_active_stop_indication
acct_authentic
acct_delay_time
acct_input_gigawords
acct_input_octets
acct_input_packets

acct_interim_interval
acct_link_count
acct_multi_session_id
acct_output_gigawords
acct_output_octets
acct_output_packets
acct_session_id
acct_session_time
acct_status_type
acct_terminate_cause
active_time
airlink_priority
air_priority
allowed_differentiated_services_marking
allowed_differentiated_services_marking__class
allowed_differentiated_services_marking__max
allowed_differentiated_services_marking__reverse_tunnel
allowed_persistent_tfts
always_on
always_ON
arap_challenge_response
arap_features
arap_password
arap_security
arap_security_data
arap_zone_access
authenticator
bad_frame_count
bad_ppp_frame_count
beginning_session
begin_session
bsid
BSID
callback_id
callback_number
called_station_id
calling_station_id
chap_challenge
chap_password
CISCO_STRING
class
coa_ack
coa_nak
coa_request
compulsory_tunnel_indicator
comp_flag
configuration_token
connect_info
correlation_id

dcch_frame_size
DFSIZE
differentiated_services_class_option
disconnectreason
disconnect_ack
disconnect_nak
disconnect_request
dns_update_capability
dns_update_required
eap_message
error_cause
esn
ESN
event_timestamp
FA_CoA
fch_frame_size
FDRC
FFSIZE
filter_id
foreign_agent_address
forward_dcch_mux_option
forward_dcch_rc
forward_fch_mux_option
forward_fch_rc
forward_pdch_rc
forward_traffic_type
framed_appletalk_link
framed_appletalk_network
framed_appletalk_zone
framed_compression
framed_interface_id
framed_ipv6_pool
framed_ipv6_prefix
framed_ipv6_route
framed_ipx_network
framed_ip_address
framed_ip_netmask
framed_mtu
framed_pool
framed_protocol
framed_route
framed_routing
FRC
FTYPE
F_DCCH_MUX
F_FCH_MUX
F_PDCH_RC
HA_IP_addr
home_agent

identifier
idle_timeout
ike_pre_shared_secret_request
ip_host
ip_port
IP_QOS
ip_quality_of_service
IP_tech
ip_technology
keyid
login_ipv6_host
login_ip_host
login_lat_group
login_lat_node
login_lat_port
login_lat_service
login_service
login_tcp_port
message_authenticator
mip_lifetime
mip_lifetime__rrq_lifetime
mip_lifetime__used_lifetime
mn_aaa_removal_indication
mn_ha_shared_key
mn_ha_spi
MS_PRIMARY_DNS
MS_PRIMARY_NBNS
MS_SECONDARY_DNS
MS_SECONDARY_NBNS
nas_identifier
nas_ipv6_address
nas_ip_address
nas_port
nas_port_id
nas_port_type
number_of_active_transitions
number_of_hdlc_layer_octets_received
number_of_sdbs_originating
number_of_sdbs_terminating
NumSDB_input
NumSDB_output
num_active
num_bytes_received
password_retry
PCF
port_limit
PPAC
PPAC__AiC
PPAC__available_in_client

PPAC__selected_for_session
PPAC__SfS
PPAQ
PPAQ__DQ
PPAQ__DT
PPAQ__duration_quota
PPAQ__duration_threshold
PPAQ__pre_paid_server
PPAQ__QID
PPAQ__quota_identifier
PPAQ__update_reason
PPAQ__UR
PPAQ__volume_quota
PPAQ__volume_quota_overflow
PPAQ__volume_threshold
PPAQ__volume_threshold_overflow
PPAQ__VQ
PPAQ__VQO
PPAQ__VT
PPAQ__VTO
pre_paid_accounting_capability
pre_paid_accounting_quota
pre_paid_tariff_switching
pre_shared_secret
prompt
proxy_state
PTS
PTS__QID
PTS__quota_identifier
PTS__tariff_switch_interval
PTS__time_interval_after_tariff_switch_update
PTS__TITSU
PTS__TSI
PTS__volume_used_after_tariff_switch
PTS__volume_used_ats_overflow
PTS__VUATS
PTS__VUATSO
RDRC
reason_ind
release_indicator
remote_address_table_index
remote_address_table_index__qualifier
remote_address_table_index__table_index
remote_ipv4_address
remote_ipv4_address_octet_count
remote_ipv4_address_octet_count__address
remote_ipv4_address_octet_count__forward_octet_count
remote_ipv4_address_octet_count__forward_overflow
remote_ipv4_address_octet_count__mask

remote_ipv4_address_octet_count__reverse_octet_count
remote_ipv4_address_octet_count__reverse_overflow
remote_ipv4_address_octet_count__table_index
remote_ipv4_address__address
remote_ipv4_address__mask
remote_ipv4_address__qualifier
remote_ipv6_address
remote_ipv6_address_octet_count
remote_ipv6_address_octet_count__address
remote_ipv6_address_octet_count__forward_octet_count
remote_ipv6_address_octet_count__forward_overflow
remote_ipv6_address_octet_count__prefix_length
remote_ipv6_address_octet_count__reverse_octet_count
remote_ipv6_address_octet_count__reverse_overflow
remote_ipv6_address_octet_count__table_index
remote_ipv6_address__address
remote_ipv6_address__prefix_length
remote_ipv6_address__qualifier
reply_message
request_message_to_the_home_radius_server
reverse_dcch_mux_option
reverse_dcch_rc
reverse_fch_mux_option
reverse_fch_rc
reverse_tunnel_specification
rn_packet_data_inactivity_timer
RRC
R_DCCH_MUX
R_FCH_MUX
SDB_input_octets
sdb_octet_count_originating
sdb_octet_count_terminating
SDB_output_octet
secret
security_level
service_option
service_option_profile
service_reference_id
service_type
serving_pcf
session_cont
session_continue
session_termination_capability
session_timeout
sf_access_point_id
SF_API
SO
SR_ID
SR_ID__main_si_indicator

SR_ID__sr_id
state
STC
s_key
s_lifetime
termination_action
user_ID
user_name
user_password
user_zone
vendor_specific

Supported Vendor-Specific Fields of the RADIUS Protocol

Following is a list of vendor-specific fields of the RADIUS protocol that the **mipt** utility supports:

3GPP_CAMEL_CHARGING_INFO
3GPP_CG_ADDRESS
3GPP_CG_IPV6_ADDRESS
3GPP_CHARGING_CHARACTERISTICS
3GPP_CHARGING_ID
3GPP_GGSN_ADDRESS
3GPP_GGSN_IPV6_ADDRESS
3GPP_GGSN_MCC_MNC
3GPP_GPRS_QOS_PROFILE
3GPP_IMEISV
3GPP_IMSI
3GPP_IMSI_MCC_MNC
3GPP_IPV6_DNS_SERVER
3GPP_MS_TIMEZONE
3GPP_NSAPI
3GPP_PDP_TYPE
3GPP_RAT_TYPE
3GPP_SELECTION_MODE
3GPP_SESSION_STOP_INDICATOR
3GPP_SGSN_ADDRESS
3GPP_SGSN_IPV6_ADDRESS
3GPP_SGSN_MCC_MNC
3GPP_USER_LOCATION_INFO
accounting_container
accounting_stop_triggered_by_active_stop_indication
active_time
airlink_priority
allowed_differentiated_services_marking
allowed_differentiated_services_marking__class
allowed_differentiated_services_marking__max
allowed_differentiated_services_marking__reverse_tunnel
allowed_persistent_tfts
always_on

bad_ppp_frame_count
beginning_session
bsid
CISCO_STRING
compulsory_tunnel_indicator
correlation_id
dcch_frame_size
differentiated_services_class_option
disconnectreason
dns_update_capability
dns_update_required
esn
fch_frame_size
foreign_agent_address
forward_dcch_mux_option
forward_dcch_rc
forward_fch_mux_option
forward_fch_rc
forward_pdch_rc
forward_traffic_type
home_agent
ike_pre_shared_secret_request
ip_quality_of_service
ip_technology
keyid
mip_lifetime
mip_lifetime__rrq_lifetime
mip_lifetime__used_lifetime
mn_aaa_removal_indication
mn_ha_shared_key
mn_ha_spi
MS_PRIMARY_DNS
MS_PRIMARY_NBNS
MS_SECONDARY_DNS
MS_SECONDARY_NBNS
number_of_active_transitions
number_of_hdlc_layer_octets_received
number_of_sdbs_originating
number_of_sdbs_terminating
PPAC__AiC
PPAC__available_in_client
PPAC__selected_for_session
PPAC__SfS
PPAQ__DQ
PPAQ__DT
PPAQ__duration_quota
PPAQ__duration_threshold
PPAQ__pre_paid_server
PPAQ__QID

PPAQ__quota_identifier
PPAQ__update_reason
PPAQ__UR
PPAQ__volume_quota
PPAQ__volume_quota_overflow
PPAQ__volume_threshold
PPAQ__volume_threshold_overflow
PPAQ__VQ
PPAQ__VQO
PPAQ__VT
PPAQ__VTO
pre_paid_accounting_capability
pre_paid_accounting_quota
pre_paid_tariff_switching
pre_shared_secret
PTS__QID
PTS__quota_identifier
PTS__tariff_switch_interval
PTS__time_interval_after_tariff_switch_update
PTS__TITSU
PTS__TSI
PTS__volume_used_after_tariff_switch
PTS__volume_used_ats_overflow
PTS__VUATS
PTS__VUATSO
release_indicator
remote_address_table_index
remote_address_table_index__qualifier
remote_address_table_index__table_index
remote_ipv4_address
remote_ipv4_address_octet_count
remote_ipv4_address_octet_count__address
remote_ipv4_address_octet_count__forward_octet_count
remote_ipv4_address_octet_count__forward_overflow
remote_ipv4_address_octet_count__mask
remote_ipv4_address_octet_count__reverse_octet_count
remote_ipv4_address_octet_count__reverse_overflow
remote_ipv4_address_octet_count__table_index
remote_ipv4_address__address
remote_ipv4_address__mask
remote_ipv4_address__qualifier
remote_ipv6_address
remote_ipv6_address_octet_count
remote_ipv6_address_octet_count__address
remote_ipv6_address_octet_count__forward_octet_count
remote_ipv6_address_octet_count__forward_overflow
remote_ipv6_address_octet_count__prefix_length
remote_ipv6_address_octet_count__reverse_octet_count
remote_ipv6_address_octet_count__reverse_overflow

remote_ipv6_address_octet_count__table_index
remote_ipv6_address__address
remote_ipv6_address__prefix_length
remote_ipv6_address__qualifier
request_message_to_the_home_radius_server
reverse_dcch_mux_option
reverse_dcch_rc
reverse_fch_mux_option
reverse_fch_rc
reverse_tunnel_specification
rn_packet_data_inactivity_timer
sdb_octet_count_originating
sdb_octet_count_terminating
security_level
service_option
service_option_profile
service_reference_id
serving_pcf
session_continue
session_termination_capability
sf_access_point_id
SR_ID__main_si_indicator
SR_ID__sr_id
s_key
s_lifetime
user_zone

Supported Fields of the SMPP Protocol

Following is a list of the SMPP protocol fields that the **mipt** utility supports:

address_range
addr_npi
addr_ton
alert_notification
application_id
auth_acc
auth_acc_resp
bind_receiver
bind_receiver_resp
bind_transceiver
bind_transceiver_resp
bind_transmitter
bind_transmitter_resp
cancel_sm
cancel_sm_resp
command_id
command_status
data_coding

data_sm
data_sm_resp
deliver_sm
deliver_sm_resp
destination_addr
dest_account
dest_addr
dest_address
dest_addr_npi
dest_addr_ton
dest_imsi
dest_oper_id
enquire_link
enquire_link_resp
error_code
esme_addr
esme_addr_npi
esme_addr_ton
esm_class
fee_addr
fee_addr_npi
fee_addr_ton
fee_fixed
fee_flag
fee_ltd_msg_num
fee_single
fee_type
final_date
generic_nack
interface_version
is_time_message
message_id
message_length
message_pid
message_state
mo_msc_addr
mo_msc_addr_npi
mo_msc_addr_ton
mo_mt_flag
mt_msc_addr
mt_msc_addr_npi
mt_msc_addr_ton
notify_mode
no_unsuccess
number_of_dests
operation_result
original_group
orig_account
orig_imsi

outbind
password
priority_flag
protocol_id
protocol_version
query_sm
query_sm_resp
registered_delivery
replace_if_present_flag
replace_sm
replace_sm_resp
schedule_delivery_time
schedule_mode
send_result
sequence_number
service_id
service_type
short_message
smsc_addr
sm_default_msg_id
sm_length
sm_result_notify
sm_result_notify_resp
source_addr
source_addr_npi
source_addr_ton
source_oper_id
status_report_request
submit_multi
submit_multi_resp
submit_sm
submit_sm_resp
system_id
system_type
unbind
unbind_resp
unsuccess_sme
validity_period

Supported SMPP TLV Fields

Following is a list of the SMPP protocol TLV fields that the **mipt** utility supports:

tlv_additional_status_info_text
tlv_billing_identification
tlv_callback_num
tlv_callback_num_atag
tlv_callback_num_pres_ind
tlv_delivery_failure_reason
tlv_destination_port

tlv_dest_addr_np_country
tlv_dest_addr_np_information
tlv_dest_addr_np_resolution
tlv_dest_addr_subunit
tlv_dest_bearer_type
tlv_dest_network_id
tlv_dest_network_type
tlv_dest_node_id
tlv_dest_subaddress
tlv_dest_telematics_id
tlv_display_time
tlv_dpf_result
tlv_its_reply_type
tlv_its_session_info
tlv_language_indicator
tlv_message_payload
tlv_message_state
tlv_more_messages_to_send
tlv_ms_availability_status
tlv_ms_msg_wait_facilities
tlv_ms_validity
tlv_network_error_code
tlv_number_of_messages
tlv_payload_type
tlv_privacy_indicator
tlv_qos_time_to_live
tlv_receipted_message_id
tlv_sar_msg_ref_num
tlv_sar_segment_seqnum
tlv_sar_total_segments
tlv_sc_interface_version
tlv_set_dpf
tlv_sms_signal
tlv_source_addr_subunit
tlv_source_bearer_type
tlv_source_network_id
tlv_source_network_type
tlv_source_node_id
tlv_source_port
tlv_source_subaddress
tlv_source_telematics_id
tlv_user_message_reference
tlv_user_response_code
tlv_ussd_service_op

Supported Fields of the SUA Protocol

Following is a list of the SUA protocol fields that the **mipt** utility supports:

ACTIVE
ACTIVE_ACK
address_indicator
address_range
affected_point_code
asp_active
asp_active_ack
asp_capabilities
asp_down
asp_down_ack
asp_identifier
asp_inactive
asp_inactive_ack
asp_up
asp_up_ack
BEAT
BEAT_ACK
CLDR
CLDT
COAK
CODA
CODT
COERR
COIT
congestion_level
connectionless_data_response
connectionless_data_transfer
connection_acknowledge
connection_oriented_data_acknowledge
connection_oriented_data_transfer
connection_oriented_error
connection_oriented_inactivity_test
connection_refused
connection_request
CORE
COREF
correlation_id
credit
data
DAUD
DAVA
deregistration_request
deregistration_response
deregistration_result
deregistration_status
DEREG_REQ
DEREG_RSP
destination_address
destination_available

destination_reference_number
destination_restricted
destination_state_audit
destination_unavailable
destination_user_part_unavailable
diagnostic_info
DOWN
DOWN_ACK
drn_label
DRST
DUNA
DUPU
ERR
error
error_code
global_title
heartbeat
heartbeat_ack
heartbeat_data
hostname
importance
INACTIVE
INACTIVE_ACK
info_string
ipv4_address
ipv6_addresses
local_routing_key_identifier
message_priority
network_appearance
notify
NTFY
point_code
protocol_class
receive_sequence_number
registration_request
registration_response
registration_result
registration_status
REG_REQ
REG_RSP
RELCO
release_complete
release_request
RELRE
RESCO
reset_confirm
reset_request
RESRE
routing_context

routing_indicator
routing_key
sccp_cause
SCON
segmentation
sequence_control
sequence_number
signalling_congestion
smi
source_address
source_reference_number
ss7_hop_count
status
SUA Field Name
SUA Field Name
subsystem_number
Supported fields of the SUA Protocol
Supported Fields of the SUA Protocol
tid_label
traffic_mode_type
UP
UP_ACK
user_cause