

# Call Information Logging, CL

OPERATIONAL DIRECTIONS



## NOTICE

The information contained in this document is believed to be accurate in all respects but is not warranted by Mitel Networks™ Corporation (MITEL®). Mitel makes no warranty of any kind with regards to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The information is subject to change without notice and should not be construed in any way as a commitment by Mitel or any of its affiliates or subsidiaries. Mitel and its affiliates and subsidiaries assume no responsibility for any errors or omissions in this document. Revisions of this document or new editions of it may be issued to incorporate such changes.

No part of this document can be reproduced or transmitted in any form or by any means - electronic or mechanical - for any purpose without written permission from Mitel Networks Corporation.

## TRADEMARKS

The trademarks, service marks, logos and graphics (collectively "Trademarks") appearing on Mitel's Internet sites or in its publications are registered and unregistered trademarks of Mitel Networks Corporation (MNC) or its subsidiaries (collectively "Mitel") or others. Use of the Trademarks is prohibited without the express consent from Mitel. Please contact our legal department at [legal@mitel.com](mailto:legal@mitel.com) for additional information. For a list of the worldwide Mitel Networks Corporation registered trademarks, please refer to the website: <http://www.mitel.com/trademarks>.

© Copyright 2016, Mitel Networks Corporation

All rights reserved

# 1

## GENERAL

The Call Information Logging (CIL) feature registers all the calls made in an MX-ONE™ system.

The Call Information data contains logging information of internal calls, and incoming and outgoing external calls. Also, calls abandoned during queuing or ringing will be logged.

The CIL feature helps customers to analyze, and thus get control over, their phone costs. Detailed information of calls can be obtained that permits the customer to correctly charge these calls to the responsible parties.

The CIL data can be sent through the network port of a local Network Interface Card (NIC) to a remote application with an established TCP/IP connection.

Using TCP/IP, V.24, or a parallel port, the CIL data can be logged to either an SQL database or to a file on a server. It can then be handled by some post-processing application.

## 1.1

### GLOSSARY AND ACRONYMS

For a complete list of abbreviations and glossary, see the description for *ACRONYMS, ABBREVIATIONS AND GLOSSARY*.

# 2

## PREREQUISITES

-

## 3

## PROCEDURE

### 3.1

### GENERAL

The CIL feature can have up to 10 active output connections individually configured and activated. They can be configured to have any of the following (major) types: SQL, file, TCP/IP stream, or V24 stream.

The type SQL, which is the primary, recommended type for MX-ONE, currently only supports the subtype PostgreSQL. Other database engines (such as DB2, Oracle, and Sybase) may be added in the future.

The type file, TCP/IP, and V24 stream support a number of formats as subtypes. These include XML, comma-separated, general, and all the formats from MD110. The general format uses a script language, which allows the user to define almost any format at runtime.

There is also a none output subtype, which can be used if alarm functionality is needed without any real storage of data.

The CIL feature may be used in the following ways:

- CIL output from a TCP/IP connection to a database or another server
- CIL output to the local or network file system
- CIL output forwarded to another LIM

It is recommended to set up one fixed local storage in every LIM, combined with a central storage in a separate system.

For example configurations, see figure 1 A CIL Configuration on page 5 and figure 2 CIL Configuration with forwarding from LIM 2 to LIM 1 on page 6.

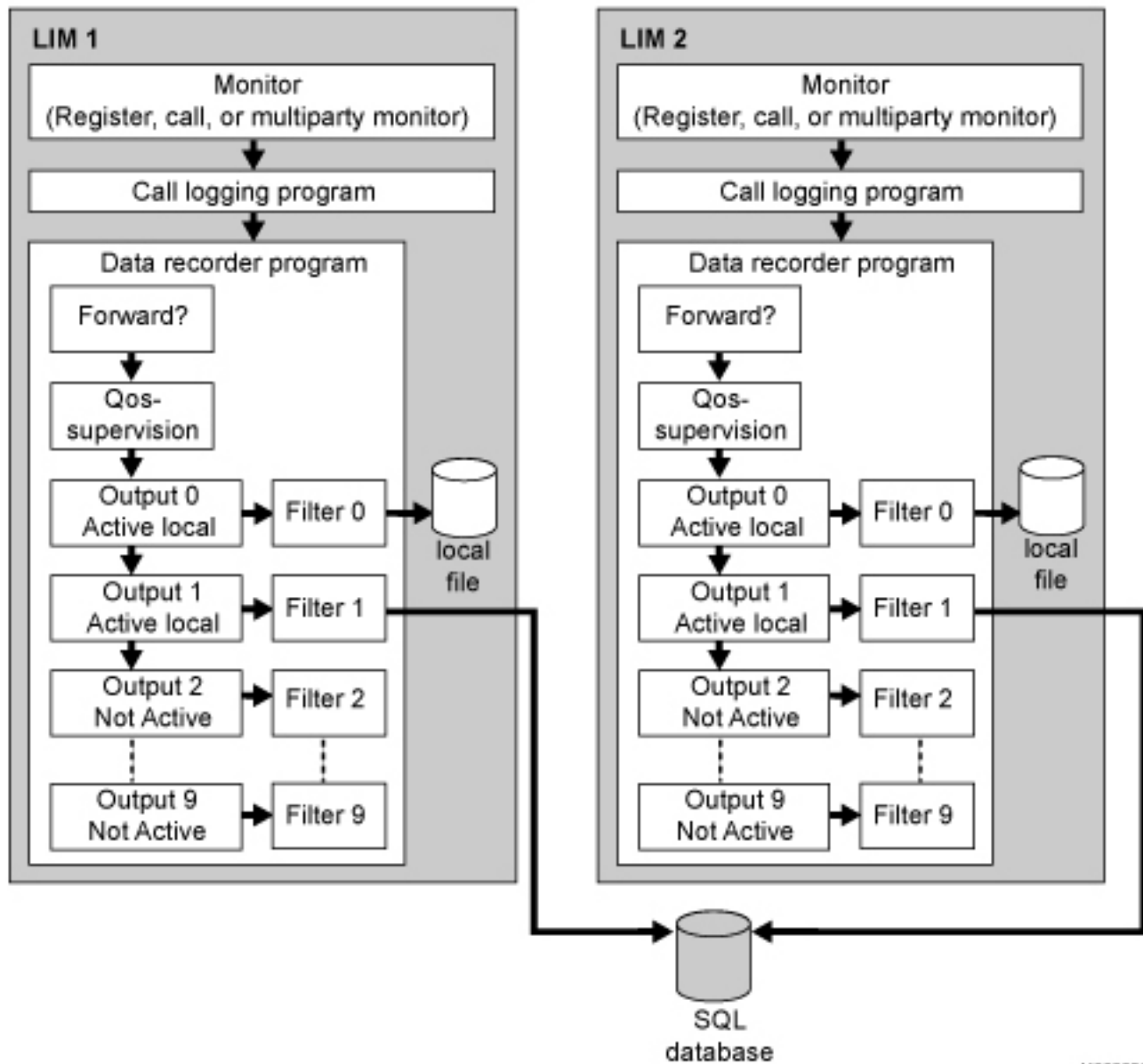


Figure 1: A CIL Configuration

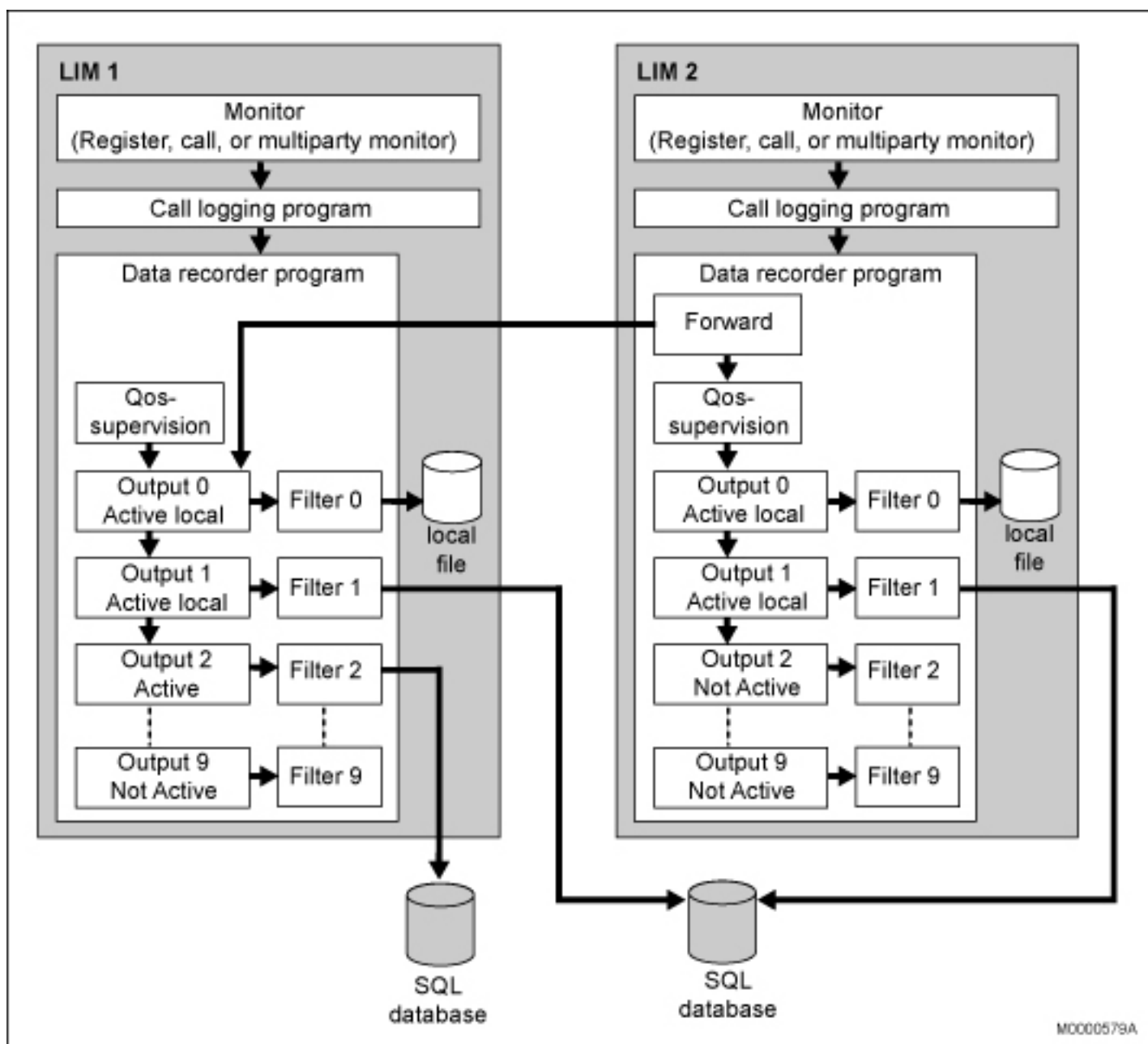


Figure 2: CIL Configuration with forwarding from LIM 2 to LIM 1

### 3.2

### INITIATING CIL

The procedure for initiating the CIL feature is as follows:

1. Initiate an output together with an output connection. The connection may be a local, or network file, or a TCP/IP connection.
2. Initiate the call criteria for an output.
3. Optionally, initiate or set the condition code strings, number data handling, and filtering options.
4. Activate the CIL feature.

### 3.3 REMOVING CIL

The procedure for removing the CIL feature and the equipment associated with it is as follows:

1. Deactivate the output.
2. Optionally, remove the call criteria for the output.

### 3.4 LOGGING TO FILE

The type file is stored daily in the file system, locally or over Network File System (NFS), for one week before being overwritten. To each file name the system adds a "daynumber" that reflects the day it was produced (for example, callData.1.xml callData.2.xml 1= Monday, 2=Tuesday).

Logging to file is recommended as backup in case the logging to a database or Management Center fails due to network problems. The commands *callinfo\_file\_to\_file*, *callinfo\_sql\_to\_file*, and *callinfo\_file\_to\_sql* can be used to repair the database log.

If the CIL files will be stored for more than six days, use a script file in a cron job to copy the files to a location where there is sufficient disk space. A template script file for this kind of copying is supplied as */etc/opt/eri\_sn/template\_scripts/callinfo\_copy\_script.sh*. This template script file includes instructions (as comments) on how to modify the template script file, and how to run it automatically every night.

Note that CIL files can occupy a large amount of disk space. Make sure to only save CIL data to disk in a way that does not exhaust your storage.

## 4 EXECUTION

### 4.1 CONFIGURE AND ACTIVATE THE CIL OUTPUT CONNECTION

#### 4.1.1 CONFIGURE THE OUTPUT CONNECTION AND START CIL OUTPUT

##### 4.1.1.1 *Execution*

- Key the command `callinfo_output_set` to define the CIL output configuration . This sets output destination, formatting rules, speed, and so on.
- Key the command `callinfo_status_set -state on` to activate the CIL output .
- Key the command `callinfo_status_print` to verify the result .

**Note:** Each command is entered as one line when entered manually.

##### 4.1.1.2 *Example 1, Basic Call Logging to SQL*

The example shows the recommended basic setup, using local backup, with central billing system. The local storage is used as a backup, if the connection to the central storage is faulty.

How to set up a local storage in every LIM combined with a central storage in a separate system:

- Enter commands  

```
>callinfo_output_set -type file -subtype xml -dbname /var/opt/eri_sn/call_logging/logfile -lim all -output 0 -local
```

```
>callinfo_status_set -output 0 -lim all -state on
```

How to set up the central storage in SQL for the billing system:

Enter commands

```
>callinfo_output_set -type sql -subtype postgresSQL -dbname smdr -server ebcl195.mynet.com:0 -lim all -output 1
```

##### 4.1.1.3 *Example 2, Basic Call Logging to a central NFS storage*

How to setup a basic logging to a central NFS storage, where the local storage is used as a backup, if the connection to the central storage is faulty:

- Enter commands  

```
>callinfo_output_set -type file -subtype XML -dbname /var/opt/eri_sn/call_logging/logfile -lim all -output 0 -local
```

```
>callinfo_status_set -output 0 -lim all -state on
```

How to set up the central storage in NFS for the billing system:

Enter commands



```
>callinfo_output_set -type asyncfile -subtype fp15 -dbname /call_logging/central_1 -lim 1 -output 1 -format "utc"
```

```
>callinfo_status_set -lim all -forward 1 -state on
```

## 4.2 CHANGE THE OUTPUT CRITERIA

**Note:** Consider testing the result of an intended change by defining an additional output to a separate file. You can, for example, define an extra output with comma-separated format. Then take the created file, and use the `callinfo_file_to_file` command to trim your filter to provide the desired output.

For more information, see chapter 4.8 Test the CIL Output for TCP/IP on page 13.

### 4.2.1 PREREQUISITES

The current CIL output criteria are unsatisfactory.

### 4.2.2 EXECUTION

- Key the command `callinfo_output_change` to initiate the changed output criteria.
- Key the command `callinfo_status_print` to verify the result.

**Table 1 Change output**

		Measure/Question	Observation/Comment
<p><b>Flow</b></p> <pre> graph TD     START([START]) --&gt; D1{1}     D1 -- Y --&gt; P2[2]     D1 -- N --&gt; D3{3}     P2 --&gt; D3     D3 -- Y --&gt; P4[4]     D3 -- N --&gt; P5[5]     P4 --&gt; P5     P5 --&gt; P6[6]     P6 --&gt; D7{7}     D7 -- Y --&gt; P8[8]     D7 -- N --&gt; STOP([STOP])     P8 --&gt; STOP           </pre>	1	Is the current flexible output format to be modified?	
	2	Determine the output number to change.	Key the command <i>callinfo_status_print</i> .
	3	Is the output active?	Key the command <i>callinfo_status_print</i> .
	4	Deactivate the output.	Key the command <i>callinfo_status_set -state off</i> . A CIL output can also be modified while it is active. The deactivation is optional.
	5	Define the displayed fields for the chosen output format.	Key the command <i>callinfo_output_set</i> (with -subtype and -format parameters).
	6	Verify the results by printing the output format.	Key the command <i>callinfo_status_print</i> .
	7	Was the output deactivated in step 5?	Key the command <i>callinfo_status_print</i> .
	8	Activate the output.	Key the command <i>callinfo_status_set</i> (with -state on)

## 4.2.3

## PRINT THE OUTPUT CRITERIA

## 4.2.3.1

*Execution*

Key the command *callinfo\_status\_print* to print the criteria.

## 4.3 REMOVE THE CIL OUTPUT CONNECTION

### 4.3.1 REMOVE THE OUTPUT CONNECTION

- Deactivate the CIL output.
- Optionally, remove the call criteria for the output.

#### 4.3.1.1 *Execution*

- Key the command `callinfo_status_set -state off` to deactivate the output connection .
- Key the command `callinfo_status_print` to verify the result .

For more details, see the command description for *CALL INFORMATION LOGGING, CL* .

## 4.4 DIALED NUMBER MASKING

### 4.4.1 SET THE MASKING OF DIALED NUMBERS IN THE CIL OUTPUT

This feature makes it possible to truncate or mask dialed numbers, for example, for reasons of integrity protection. The last digits of the dialed number will be removed from the CIL output.

#### 4.4.1.1 *Execution*

Key the command `callinfo_mask_set` to specify how to treat the dialed number regarding masking.

### 4.4.2 PRINT THE DIALED NUMBER MODIFICATION DATA

#### 4.4.2.1 *Execution*

Key the command `callinfo_mask_print` to print the current setting of number masking.

## 4.5 HEARTBEAT

### 4.5.1 SELECT THE HEARTBEAT FUNCTION

This feature makes it possible to periodically send information every 15 minutes from, for example, an exchange to a service center. If the exchange should go down, the service center can take necessary measures without losing too much valuable data.

It is not possible to initiate this facility if the local or network file system is used as a CIL output device.

## 4.5.1.1

*Execution*

Use the `-heartbeat` option to the command `callinfo_output_set` to select the heartbeat function.

A printout of the year, month, date, and exchange identity is generated to the ordinary CIL output.

## 4.6

## CONDITION CODE CHANGE AND PRINT

## 4.6.1

## CONDITION CODE CHANGE

The condition code is a key element in CIL data. It designates the type of call that was made or the telephony service that was granted. Condition codes can be customized by setting customer-defined text strings to meet specific demands. This is only available for flexible formats.

## 4.6.1.1

*Execution*

- Key the command `callinfo_condcode_set` to change condition code characters .
- Key the command `callinfo_condcode_print` to verify the result .

## 4.6.2

## PRINT INFORMATION ABOUT CONDITION CODES

## 4.6.2.1

*Execution*

Key the command `callinfo_condcode_print` to see how condition codes are presented for each output format .

## 4.6.3

## PRINT INFORMATION ABOUT OUTPUT FORMATS

## 4.6.3.1

*Execution*

Key the command `callinfo_output_info` to see information about output formats . This command shows the currently available condition code types, subtypes, and formats, plus some examples.

## 4.6.4

## SET CUSTOMER-SPECIFIC TEXT STRINGS FOR CONDITION CODES

## 4.6.4.1

*Execution*

- Key the command `callinfo_condcode_set -string` to customize text strings to be printed in the CIL output .
- Key the command `callinfo_condcode_set -restore` to revert to the default format (CC3) .

## 4.7 POST PROCESSING OF DATA FROM LOCALLY STORED FILES

This section applies when the central storage has been lost and you want to update from a local copy.

### 4.7.1 SET FILTER CRITERIA AND TRANSFER CIL DATA

The CIL output data can be post-processed internally in the MX-ONE Service Node by defining filter criteria and transferring the CIL data from one output format to another. The transfer types can be file to SQL database, SQL database to file, or file to file.

#### 4.7.1.1 *Prerequisites*

- CIL data has been logged and stored in the specified source format.
- The destination output connection must have been defined.

#### 4.7.1.2 *Execution*

- Depending of source and destination format, enter one of the following commands:
  - Key the command *callinfo\_file\_to\_sql* to set filtering criteria and transfer (possibly filtered) CIL data from file to SQL database .
  - Key the command *callinfo\_sql\_to\_file* to set filtering criteria and transfer (possibly filtered) CIL data from SQL database to file .
  - Key the command *callinfo\_file\_to\_file* to set filtering criteria and transfer (possibly filtered) CIL data from file to file .

### 4.7.2 PRINT THE FILTERING SETTINGS

#### 4.7.2.1 *Execution*

Key the command *callinfo\_status\_print* to see the current filtering settings .

## 4.8 TEST THE CIL OUTPUT FOR TCP/IP

### 4.8.1 PRINT THE RECEIVED CIL DATA ON A TCP PORT

To verify that the CIL output has been correctly set up, there is a function which simulates a TCP/IP server.

#### 4.8.1.1 *Prerequisites*

The CIL output function shall be initiated and active.

#### 4.8.1.2

##### *Execution*

Key the command *callinfo\_tcp\_print* to start listening to a specified TCP port, and print the received data.

## 5 OUTPUT FORMATS

### 5.1 DEFINE AN OUTPUT FORMAT

Up to 10 outputs can be defined per LIM, and each can have its own format. With flexible formats, it is possible to omit the option of logging abandoned and failed calls. The steps to change an existing format are the same as for defining a format for the first time.

#### 5.1.1 FORMAT STRINGS, GENERAL FORMAT

For a thorough description, see the interworking description for *Call Information Logging, Quality Logging*, section Using a Script as Both Formatter and Filter.

The General format has no predefined format string. Instead all formatting of the General format is build by the user using the **-format** option. The predefined formats (FP15, MDFP15, ASB501 and ASBUMDFP15) have a predefined format string. For the predefined formats the argument to the **-format** option is intelligently merged with the predefined format string.

With a condition string to the **-format** option the General format and the predefined formats (FP15, MDFP15, ASB501, and ASBUMDFP15) can be filtered to store only explicitly defined data. The format strings can contain conditions, written inside square brackets, that limit what is logged. For example, only log calls that last more than 10 seconds would give: **-format "[duration >= 10]"**.

With the **[head]** condition is stated what will be recorded only the first time. In the previous example, 'duration' is a predefined variable. For a list of these variables, see the interworking description for *Call Information Logging*, on Field Names in the section Using a Script as Both Formatter and Filter.

What data to log can be stated with predefined variables, if they are to be left or right justified, and the min and max number of digits. These terms are written within curly braces. If we, for calls with valid account codes, would want to log the dialed number, right justified, with 15 digits, and access code1 left justified with 3 digits, this could be written as **-format "[accountcodevalid == 1] : {dialednumber R 15 15} {access-Code1 L 3 3}"**. The variables are case insensitive. Different statements within the format string are separated by semicolons.

As seen in the interworking description there are a number of substitutions available, like **{newline}** to output a new line in the log.

If **{exit}** is specified after a condition and the condition is true, processing of the call is abandoned at **{exit}**. That means that all output specified before **{exit}** will be output (if the conditions are true), but what is specified after **{exit}** will be ignored. After "executing" an **{exit}** further statements will not be processed for that call.

**Note:** The `callinfo_output_set` command should preferably be written as a command file, which is "sourced" to run from an mdsh shell. When entered as a sourced file, line continuation through the backslash character, "\", is supported. It also works to put the command on one line, when sourced in mdsh.

In the following examples, where the backslash character is shown, the use of sourced files are assumed.

## 5.1.2

## EXAMPLE 1

Create an output format for the following conditions:

"A connected to B", "A connected to B but dialed C", "A connected to B but dialed C duration D"

The dialed number is only recorded if it is different from the connected number. The duration is only recorded if both the call is longer than 10 seconds and the dialed number is different from the connected one.

```
callinfo_output_set -type file -subtype general \
-dbname /var/log/smdr/ex_2 -output 1 -lim all \
-format "{callingNumber L 2 10} connected to {connectedNumber L 2 10} ; \
[dialedNumber != connectedNumber] : but dialed {dialedNumber L 2 10} ; \
[duration > 10] : \
duration {duration L 2 20} ; {newline} ; "
```

```
callinfo_status_set -state on -lim all -output 1
```

As seen in the above example arbitrary text ("connected to ", "but dialed ", and "duration ") can be entered in the format string to enhance the readability of the logged data.

An alternative implementation would be to use {exit} to terminate recordings with wrong format.

```
callinfo_output_set -type file -subtype general \
-dbname /var/log/smdr/ex_2 -output 1 -lim all \
-format "{callingNumber L 2 10} connected to {connectedNumber L 2 10} ; \
[dialedNumber ==connectedNumber] : {newline} {exit} ; \
but dialed {dialedNumber L 2 10} ; \
[duration <= 10] : {newline} {exit} ; duration {duration L 2 20} ; {newline} ; "
```

```
callinfo_status_set -state on -lim all -output 1
```

## 5.1.3

## EXAMPLE 2

This example shows a UK logging format with output over TCP to the server at 192.168.0.1 on port 9876.

```
callinfo_output_set -lim all -output 0 -type tcp -subtype general \
-server 192.168.0.1 -port 9876 -local -utc -eol CRNL \
-format "{stoptime md110date L 4 4} {stoptime md110time L 4 4} \
{stoptime second 0R 2 2} {duration md110duration L 5 5} ; \
[ taxpulses != 0 ] { taxpulses R 4 4 } ; [ taxpulses == 0 ] : ; \
{conditionCodeUserDefined L 3 3} {accesscode1 R 5 5} {accesscode2 R 5 5} \
{dialednumber R 20 20} {callingnumber L 20 20} {accountcode L 15 15} \
{cilcode L 6 6} {queueTimeCounter 0R 2 2} {ringTimeCounter 0R 2 2} \
{ogTrnkId R 9 9} {incTrnkId R 9 9} {connectedNumber R 16 16} {newline} ; "
```



```
callinfo_status_set -state on -output 0 -lim all
```

## 5.2 FIXED OUTPUT FORMAT

The fixed output format is a predefined output format *F* that cannot be altered. The fixed output formats can only be used for CIL data, not for VoIP QoS logging. For further information, see the command description for *CALL INFORMATION LOGGING, CL*.

### 5.2.1 EXECUTION

- Key the command *callinfo\_output\_set* -format to choose a fixed output format.
- Key the command *callinfo\_status\_print* to check the result .

### 5.2.2 EXAMPLE 1

The customer wants to do call logging in the MDFP15 format to the file */var/log/smdr/mdfp15* but would like to log only external calls (with *accessCode1 == 00*) that are at least 10 seconds long. Time data shall be presented in local time.

```
callinfo_output_set -type file -subtype mdfp15 -dbname \
/var/log/smdr/mdfp15 -output 1 -lim all -localtime \
-format "[duration < 10] :{exit}; [accessCode1 != 00] :{exit}; "
```

```
callinfo_status_set -state on -lim all -output 1
```

### 5.2.3 EXAMPLE 2

The customer wants to do call logging in the ASB501 format to the file */var/log/smdr/asb501*, but to reduce the amount of data, discard calls to and from numbers in the range from 5900 to 5999. Time data shall be presented in local time. Output 2 will be used.

```
callinfo_output_set -type file -subtype asb501 \
-database /var/log/smdr/asb501 -output 2 -lim all -localtime \
-format "[callingNumber begins 59] :{exit}; \
[dialedNumber begins 59] :{exit};"
```

```
callinfo_status_set -state on -lim all -output 2
```

## 5.2.4

## USE THE GENERAL FORMAT, TO CHANGE PREDEFINED FORMATS

To change a format that is almost right, create a new general format, using the existing format as a template.

- Use the command *callinfo\_format\_print -subtype* to print the existing format. Perform the needed modifications.
- Open the new format string in an output file to test the change. Hint: perform 'tail -f' on the file, to output as it is written to the file.

**Note:** A delay may be noticed if the file system caches the data prior to output to file. Try making more than one call to flush the cache or wait for the cache flush every second minute.

An existing file from a local fixed format copy, XML, comma separated, or SQL, can also be used. One of the commands *callinfo\_file\_to\_file*, or *callinfo\_sql\_to\_file* shall be used. The resulting file can be examined and the format string can be adjusted without making calls. This method is faster if complicated calls have to be made to confirm the format changes.

To test the format, enter the following command (on one line).

```
>callinfo_file_to_file -insubtype xml
-infilename /var/smdr/data.1.xml -outsubtype asb501
-outfilename /var/opt/eri_sn/call_logging/conv -format "xxx"
```

Where "xxx" is the format to be tested

## 5.2.5

## EXAMPLE 2

Create an output like FP15, but add the 3 letter condition code last on each line. If it is a normal internal call, though, write "normal internal" instead of the condition code.

Get the format string for FP15. Do a copy and paste from the output of the command *callinfo\_format\_print -subtype* fp15.

You will get the format string (on one line):

```
" [head] : {currentTime month 0R 2 2} {currentTime day 0R 2 2} {newline} ; [isMobileLogging != 1] :
```

```
{stoptime md110time L 4 4} {duration md110durationDecMinute L 5 5} {condition-code1character L 2 2}
```

```
{accesscode1 L 3 3} {accesscode2 L 3 3} {dialednumber R 15 15}
```

```
{callingnumber L 4 4} ; [isMobileLogging != 1] [accountcodevalid != 1] : {cilCode R 15 15} {newline} ;
```

```
[isMobileLogging != 1] [accountcodevalid == 1] : {accountcode R 15 15} {newline}; "
```

Writing "normal internal" can be done in two ways. The first way is to globally redefine the user defined condition code string "conditionCodeUserDefined". This is the same as the 3 letter condition code, unless already changed. To do this use the following command:

```
callinfo_condcode_set -code 8 -string "normal internal"
```

Then use the `callinfo_output_set` command with the format string as above, where the user defined condition code is added, and activate the logging with the `callinfo_status_set` command.

```
callinfo_output_set -lim all -output 1 -type file -subtype general -dbname \
/var/log/smdr/ex_3 -format " [head] : {currentTime month 0R 2 2} \
{currentTime day 0R 2 2} {newline} ; [isMobileLogging != 1] : \
{stoptime md110time L 4 4} {duration md110durationDecMinute L 5 5} \
{conditioncode1character L 2 2} {accesscode1 L 3 3} {accesscode2 L 3 3} \
{dialednumber R 15 15} {callingnumber L 4 4} ; \
[isMobileLogging != 1] [accountcodevalid != 1] : {cilCode R 15 15} {newline} ; \
[isMobileLogging != 1][accountcodevalid == 1] : \
{accountcode R 15 15} {conditionCodeUserDefined L 1 20} {newline}; "
```

```
callinfo_status_set -state on -lim all -output 1
```

If the "conditionCodeUserDefined" is used for something else the format string can be written using a test on the condition code, where the "normal internal" text is written if the condition is true.

```
callinfo_output_set -lim all -output 1 -type file -subtype general -dbname \
/var/log/smdr/ex_3 -format " [head] : {currentTime month 0R 2 2} \
{currentTime day 0R 2 2} {newline} ; [isMobileLogging != 1] : \
{stoptime md110time L 4 4} {duration md110durationDecMinute L 5 5} \
{conditioncode1character L 2 2} {accesscode1 L 3 3} {accesscode2 L 3 3} \
{dialednumber R 15 15} {callingnumber L 4 4} ; \
[isMobileLogging != 1] [accountcodevalid != 1] : {cilCode R 15 15} {newline} ; \
[isMobileLogging != 1] [accountcodevalid == 1] : {accountcode R 15 15} ; \
[conditionCode != 8] : {conditionCode3Character L 3 3} ; \
[conditionCode == 8] : normal internal ; {newline} ; "
```

```
callinfo_status_set -state on -lim all -output 1
```

## 6

## TERMINATION

If exchange data has been altered and no more commands are to be entered, a dump to backup media must be performed, see operational directions for *ADMINISTRATOR USERS GUIDE*.

## 7

## FREQUENTLY ASKED QUESTIONS

Below are the answers to the most frequently asked questions about call information logging.

- How many storage files can a system have?  
Each LIM can store up to 10 outputs (streams), where each output is independent. So basically the system can have 10 different outputs per LIM, each with a different format (totally 1240 outputs).
- Can several formats be active simultaneously?  
Yes, all output streams can have its own format. The same data can basically be printed in 10 different ways per LIM.
- Where are the files normally stored?  
The path to where the files are stored is /var/opt/eri\_sn/call\_logging.
- Is it possible to get only one output for the whole system?  
Each LIM can forward its input to other LIMs. Then the receiving LIM can be the connection point to the central billing system.
- Why does the commaseparated look so strange?  
The intension is to create a format that is easy to read, by a computer, where each column carries specific data. Each call is one row and each data in the output uses one column. If a file is imported into Microsoft Excel, the initial comment can be used to identify the data fields as the first rows can be fixed to be the heading.
- Can an individual file be stored in a central place?  
Yes, but this is not recommended because the post processing gets more complex with more files and more commands are needed. Enter the following commands to create one set of files per LIM in the central storage:  
  

```
>callinfo_output_set -type file -subtype xml -dbname /var/opt/eri_sn/call_logging/logfile -lim all -output 0 -local
```

```
>callinfo_status_set -output 0 -lim 1 -state on
```

```
>callinfo_output_set -type asyncfile -subtype fp15 -dbname /call_logging/central_1 -server storage.mynet.com:0 -lim 1-output 1 -format "utc"
```

```
>callinfo_status_set -output 1 -lim 1 -state on
```

```
>callinfo_output_set -type asyncfile -subtype fp15 -dbname /call_logging/central_2 -server storage.mynet.com:0 -lim 2 -output 1 -format "utc"
```

```
>callinfo_status_set -output 1 -lim 2-state on
```
- Why are the files deleted after 6 days? Can data be stored more then 6 days?  
The setup is to prevent the files to grow indefinitely. This way the directory size is controlled and no human intervention is (normally) needed to prevent the disks to be filled up. To store the data more than 6 days, there is a template file that can be started from the cron service each day, under /usr/etc/opt/eri\_sn/templates/callinfo\_copy\_script.sh.
- Can more than one central storage be set up?  
Yes, all outputs can be setup as central storage, but it is recommended to store at least one local copy.

- How can the data, sent on TCP to the central storage, be tested?

The command **callinfo\_tcp\_print** is creating a passive server that will listen to the incoming datastream and present it on the screen. To see the output, make a copy of the output data and paste it directly into the server where the program is running. For example, enter command:

```
>callinfo_tcp_print -server localhost -port 1023
```

- How is a V24 (rs232) output set up?

**Note:** User "eri\_sn\_d" must be reconfigured. For more details, see User Account Management (66/15431-ANF90114).

A 9 pin - 25pin null modem shall be used, if the system is connected to a 25 pin dumb terminal for testing. The following example is using bitrate = 9600, flow control = none, bits = 8, stopbit = 1, no parity, control signal = DTR/RTS. For example, enter commands:

```
>callinfo_output_set -output 1 -type v24 -subtype asb501 -lim all -dbname /dev/ttyS0 -bitrate 9600 -parity no -flowcontrol hw -databits 8 -eol CRNL -record call -format local
```

```
>callinfo_status_set -state on -lim all -output 1
```

- How is the data sent to a line printer (centronix)?

The data is sent to a line printer by entering commands:

```
>callinfo_output_set -output 1 -type v24 -subtype asb501 -lim all -dbname /dev/lp0 -noinit -eol CRNL -record call -format local
```

```
>callinfo_status_set -state on -lim all -output 1
```

- How is the naming of the files done?

The file name is built up from the -dbname, supplied on the command line, by adding a week day number and an extension. The extension is usually .dat or .xml. The week day extension is 0 for Sunday, 1 for Monday ... and 6 for Saturday.

For example, enter commands:

```
>callinfo_output_set -output 0 -type file -subtype xml -dbname /var/opt/eri_sn/call_logging/foobar -lim all
```

```
>callinfo_status_set -lim 1 -output 0 -state on
```

If the commands above are entered on a Tuesday, the resulting output file name would be "var/opt/eri\_sn/call\_logging/foobar.2.xml".