SCRIPTING

VULCAN SCRIPTING

Specification

This is the Scripting Specification for Layer 4-7.

Last updated: 2018-11-08

Introduction

The Xena L47 Scripting Commands are very similar to the L23 Scripting Commands. In a few cases the commands are identical, however many new L47 specific commands have been created.

Commands are logically grouped in a hierarchy. At the top level we have a Chassis. Currently there are two different L47 Chassis: 'XenaAppliance' and 'XenaScale'. For 'XenaAppliance' and 'XenaScale', the entire Chassis is considered one L47 Module, however in the future a chassis may have several L47 modules.

A L47 Module have several Ports (currently between 1 and 12). Like on L23, each Port must be reserved before it can be configured and traffic can be started, allowing multiple users to work with the L47 product at the same time.

In addition to Ports a L47 Module contains a number of Packet Engines (PE), which generates and handles the TCP traffic. As default each port is allocated one PE, however more PEs can be allocated to a Port increasing the performance on that port. Packet Engines are a shared resource between the Ports on a module. Currently 'Xena Appliance' contains 5 PEs and 'Xena Scale' contains 2 groups of 14 PEs.
The Stream concept in L23 scripting has been replaced by Connection Groups (CG). A Connection Group specifies a number of TCP connections (1 to 2 million per PE). Several Connection Groups can be configured on a Port (currently up to 200).

A Connection Group has a configured Load Profile, which defines the ramp-up start time along with the durations of the ramp-up, steady-state and ramp-down periods. A Connection Group is configured with an Application Type and an Application Scenario. The Application Type defines the type of data transmitted by the TCP connections, and the Application Scenario defines the data flow between Servers and Clients.
By combining several Connection Groups on a Port, it is possible to create a mixture of different traffic types and scenarios, and to create complex resulting load profiles.

L47 Scripting Commands are divided into groups, that deals with the different resources and aspects of the L47 products:

In terms of command syntax, there are four groups of scripting commands:

- Chassis Commands
- Module Commands
- Port Commands
- Connection Group Commands

However, to make it easier to get an overview of all the scripting commands, some of the module, port and connection group commands are separated in a number of other groups in this document.

- Packet Engine Commands
- Traffic Command
- Statistics Commands
- Capture Commands

Table of contents

Command Syntax

A Scripting Command can either be a get or a set command.

The general format of a get command is:
```
<addressing> <command_name> <index> ?
```

where <addressing> specifies module and port numbers as appropriate and <index> specifies for example connection group number.

The response to a get command has the format:
```
<addressing> <command_name> <index> <parameters>
```

where <parameters> are the values of the parameters requested by the get command

The general format of a set command is:
```
<addressing> <command_name> <index> <parameters>
```

where <parameters> are the values of the parameters set by the command. A few set commands has no parameters as they are action commands rather than commands to configure parameters (e.g. a command to clear counters).

The response to a set command has the format:
```
'<status>'
```

where '<status>' can have the following values.

<OK>	ok
<NOCONNECTIONS>	chassis has no available user connections
<NOTLOGGEDON>	no command can be submitted before logon
<NOTRESERVED>	parameter cannot be set because resource not reserved
<NOTWRITABLE>	parameter is read-only
<NOTREADABLE>	parameter is write-only
<NOTVALID>	operation not valid in current state
<BADCOMMAND>	invalid command
<BADPARAMETER>	invalid parameter code
<BADMODULE>	invalid module index
<BADPORT>	invalid port index
<BADINDEX>	invalid connection group index
<BADSIZE>	invalid size of data
<BADVALUE>	invalid value of data
<FAILED>	failed to perform operation
<MEMORYFAILURE>	failed to allocate memory

<NOLICPE>	no free PE licence
<NOLICPORT>	no free port licence

Command Naming

Chassis commands

Chassis commands has the format:

```
C_XXX
```

Examples:

```
C_LOGON “xena”
C_OWNER ?
```

Module commands

Module commands has the format

```
<module> M_XXX
```

or

```
<module> M4_XXX
```

for L47 specific module commands, where <module> specifies the module number.

Examples:

```
0 M_SERIALNO ?
0 M4_SYSTEMID ?
```

Port commands

Port commands have the format:

```
<module>/<port> P_XXX
```

or

```
<module>/<port> P4_XXX
```

for L47 specific module commands, where <module> specifies the module number and <port> specifies the port number.

Examples:

```
1/0 P_SPEED ?
1/0 P4_TRAFFIC on
```

Connection Group commands

Connection Group commands have the format

```
<module>/<port> P4G_XXX [index]
```

where <module> specifies the module number, <port> specifies the port number and index specifies the connection group number.

Examples:

```
1/0 P4G_CREATE [0]
1/0 P4G_CLIENT_RANGE [0] ?
```

Packet Engine commands

Packet Engine commands is a collection of port and module commands, and have the format
`<module> M4E_XXX`
or
`<module>/<port> P4E_XXX`
where <module> specifies the module number and <port> specifies the port number.

Examples:

```
0 M4E_MODE ?
1/0 P4E_ALLOCATE 2
```

Multi-User

Multiple users can operate on the Chassis simultaneously. To ensure smooth operation, access restrictions apply.

All parameters can be read by anyone as long as that user has logged on using the C_LOGON command. In order to set parameters on a Chassis, Module, or Port, the corresponding resource must be reserved by that user. Users are identified by name, which is configured using the C_OWNER command. Reservation state query and reservations can be done using the C/M/P_RESERVATION and C/M/P_RESERVEDBY commands.

Example:
Logging on to a chassis and reserving port 0 and 1 is done by the following commands.

```
C_LOGON “xena”
C_OWNER “JohnDoe”

1/0 P_RESERVATION reserve
1/1 P_RESERVATION reserve
```

Resources reserved by a user can be released by the same user using the command

```
1/0 P_RESERVATION release
```

Resources can also be released by other users using the command

```
1/0 P_RESERVATION relinquish
```

A description of what a resource is used for can be given using C/P_COMMENT.

Example:

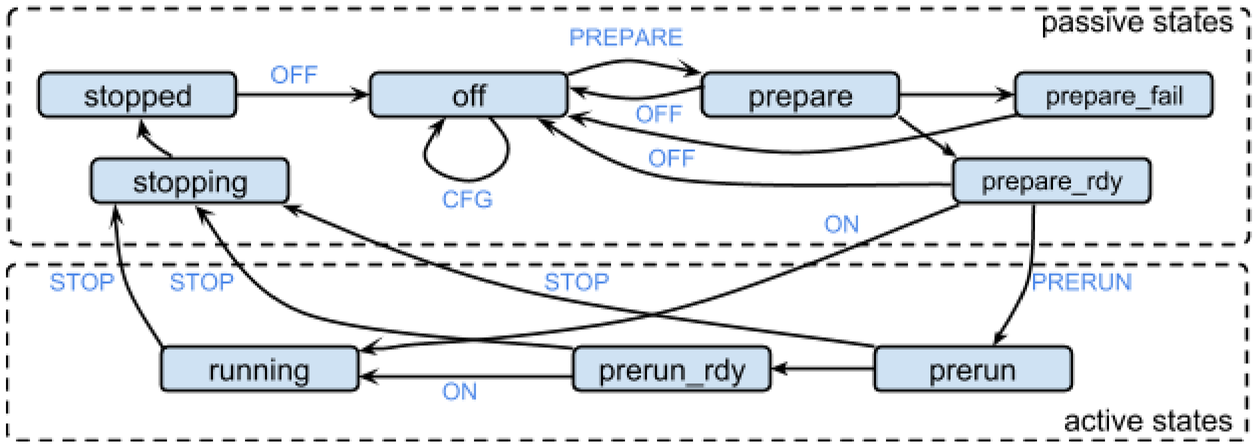
```
C_COMMENT “Live demo chassis. Resources can be relinquished without warning”
```

or

```
1/0 P_COMMENT “Port used until 5pm sat oct. 5, after which it can be relinquished”
1/1 P_COMMENT “Port used until 5pm sat oct. 5, after which it can be relinquished”
```

L47 Port States

The Xena L47 test execution engine has seven states: *off*, *prepare*, *prepare_rdy*, *prerun*, *prerun_rdy*, *running* and *stopped*. Traffic is generated in the *prerun* and *running* states only, and configuration of parameters is only valid in state *off* except for a few runtime options. Port traffic commands can be given with P4_TRAFFIC and port state queried by P4_STATE.



off - default state. Entered from *stopped* or *prepare* on 'OFF' command. This is the only state that allows configuration commands. 'P4_RESET' is also considered a configuration command. Upon entering *off* state**,** some internal 'house cleaning' is done. For example: freeing TCP Connections, clearing test specific counters etc.

prepare - this state is entered from state *off* on 'PREPARE' command. Here internal data structures relevant for the test configuration are created. When done the state changes to *prepare_rdy* or *prepare_fail* and,, a 'P4_STATE prepare_rdy' or 'P4_STATE prepare_fail' notification is sent to all users logged on to the chassis.

prepare_rdy- entered automatically after activities in *prepare* have completed successfully.

prepare_fail - entered automatically from *prepare*, if an error occurs. An error could for example be failure to load a configured replay_file.

prerun - entered from *prepare_ready* on 'PRERUN' command. If enabled, this is where ARP and NDP requests are sent. When done the state changes to *prerun_rdy* and a 'P4_STATE prerun_rdy' notification is sent to all users logged on to the chassis.

prerun_rdy - entered automatically after activities in *prerun* has completed.

running - entered either from *prepare_ready* or *prerun_ready* on 'ON' command. This is where TCP connections are established, payload is generated and connections are closed again.

stopping - entered from *running*, *prerun_ready* or *prerun* on “STOP” command. Stops Rx/Tx traffic. In *stopping* state, post-test data are calculated and captured packets are saved to files.

stopped - entered automatically after activities in *stopping* are complete. This is where we can read post-test statistics and extract captured packets.

Connection Groups

A Connection Group (CG) is the basic building block when creating L47 traffic. A Connection Group consists of a number of TCP connections - between one and millions. The CG has a role, which is either *client* or *server*. In order to create TCP connections between two ports on a L47 Chassis, two matching CGs must be configured - one on each port - one configured as *client* and the other configured as *server*.
The number of connections in a CG, is defined by the server range and the client range. A server/client range is a number of TCP connection endpoints defined by a number of IP addresses and a number of TCP ports. A server/client range is configured by specifying a start IP address, a number of IP addresses, a start TCP port and a number of TCP addresses. The number of clients are the number of client IP addresses times the number of client TCP ports, and the same goes for the number of servers. The number of TCP connections in a CG is the number of clients times the number of servers, that is TCP connections are created from all clients in the CG to all servers in the CG.

Example:
A Connection Group containing 100 clients on port 0 and 10 servers on port 1 can be configured the following way:

```
1/0 P4G_CREATE [0]
1/1 P4G_CREATE [0]
1/0 P4G_ROLE [0] CLIENT
1/1 P4G_ROLE [0] SERVER

1/0 P4G_CLIENT_RANGE [0] 10.0.1.1 10 5000 10
1/0 P4G_SERVER_RANGE [0] 10.0.2.1 10 80 1
1/1 P4G_CLIENT_RANGE [0] 10.0.1.1 10 5000 10
1/1 P4G_SERVER_RANGE [0] 10.0.2.1 10 80 1
```

NOTE: Connection Group index must start from 0.

Now Port 0 contains 100 clients - 10 different TCP ports on 10 different IP addresses, and Port 1 contains 10 servers - 1 TCP port on 10 different IP addresses. When starting traffic on Port 0 and 1, 1000 TCP connections will be established - from all clients to all servers.
NOTE: When configuring a CG, both client AND server range must be configured on both CGs - that is, the server CG must also know the client range and vice versa.

The Connection Group must be configured with a Load Profile, which is an envelope over the TCP connections lifetime. The connections in a connection group goes through three phases. The Load Profile defines a start time and a duration of each of these phases. During the ramp-up phase connections are established at a rate defined by the number of connections divided by the ramp-up duration. During the steady-state phase connections may transmit and receive payload data, depending on the configuration of test application and test scenario for the CG. During the ramp-down phase connections are closed at a rate defined by the number of connections divided by the ramp-up duration, if they were not already closed as a result of the traffic scenario configured.

Example:
The 1000 connections configured above will be ramped up in 1 second - starting immediately - will live for 10 seconds, and will be ramped down in 2 seconds with the following configuration:

```
1/0 P4G_LP_TIME_SCALE [0] SECONDS
1/1 P4G_LP_TIME_SCALE [0] SECONDS

1/0 P4G_LP_SHAPE [0] 0 1 10 2
1/1 P4G_LP_SHAPE [0] 0 1 10 2
```

NOTE: Just like client and server range, both the client and server CG must be configured with the Load Profile.

Next the CG must be configured with a *test application*, which defines what kind of traffic is transported in the TCP payload. Currently there are two kinds of test applications:

NONE, which means that no payload is sent on the TCP connections. This test application is suitable for a test, where the only purpose is to measure TCP connection open and close rates.

RAW, which means that the TCP connections transmits and receives user defined raw data. The contents of the raw TCP payload can be configured using P4G_RAW_PAYLOAD command. Raw TCP payload can also be specified as random and incrementing data.

Using test application RAW, the CG must also be configured with a test scenario, which defines the data flow between the TCP client and server. Currently the following test scenarios can be configured: Download, upload, both and echo.

Example:
The Connection Group defined above is configured to transmit random payload data from the servers to the clients after the clients have transmitted (and the servers received) a download request, with the following commands:

```
1/0 P4G_TEST_APPLICATION [0] RAW
1/1 P4G_TEST_APPLICATION [0] RAW

1/0 P4G_RAW_TEST_SCENARIO [0] DOWNLOAD
1/1 P4G_RAW_TEST_SCENARIO [0] DOWNLOAD

1/0 P4G_RAW_HAS_DOWNLOAD_REQ [0] YES
1/1 P4G_RAW_HAS_DOWNLOAD_REQ [0] YES

1/0 P4G_RAW_PAYLOAD_TYPE [0] RANDOM
```

```
1/1 P4G_RAW_PAYLOAD_TYPE [0] RANDOM
```

By combining several Connection Groups on a port, it is possible to create more complex traffic scenarios and more complex Load Profile shapes than the individual Connection Group allows. Examples of combinations of CGs can be found in the Scripting Examples section.

Data Types

- Integer (`I`): decimal integer, in the 32-bit range, e.g.: `1234567`.
- Long (`L`): decimal integer, in the 64-bit range, e.g.: `123456789123`.
- Byte (`B`): decimal integer, in the 8-bit range, e.g.: `123`.
- Hex (`H`): two hexadecimal digits prefixed by `0x`, e.g.: `0xF7`.
- String (`S`): printable 7-bit ASCII characters enclosed in `”`, e.g.: `”A string”`. Characters with values outside the 32-126 range and the `”` character itself are specified by their decimal value, outside the quotation marks and separated by commas, e.g.: `”A line”,13,10,”and the next line”`.
- Owner (`O`): a short string used to identify an owner, used for reservation.
- Address (`A`): a dot-separated IP address, e.g.: `192.168.1.200`.

Chassis Commands

The chassis commands deal with basic information about, and configuration of, the chassis itself (rather than its modules and test ports), as well as overall control of the scripting session.

All of the chassis command names have the form ***C_xxx*** and use neither a module index nor a port index.

The following chassis commands are supported by Vulcan chassis, and are all identical to the commands supported by Valkyrie:

C_LOGON	**C_MODEL**	**C_NAME**
C_LOGOFF	**C_SERIALNO**	**C_COMMENT**
C_OWNER	**C_VERSIONNO**	**C_PASSWORD**
C_RESERVATION	**C_PORTCOUNTS**	**C_TIMEOUT**
C_RESERVEDBY	**C_CONFIG**	
C_DOWN	**C_INFO**	

Module Commands

The module parameter names all have the form M_xxx and require a module index before the parameter name.

The following module commands supported by the L47 chassis are identical to the commands supported by the L23 products:

M_RESERVATION
M_RESERVEDBY
M_MODEL
M_SERIALNO
M_VERSIONNO
M_PORTCOUNT

The following module license commands are currently only supported by the L47 products:

M_LICENSE_DEMO_INFO
M_LICENSE_MAINTENANCE_INFO
M_LICENSE_CWB_INFO
M_LICENSE_UPDATE
M_LICENSE_UPDATE_INFO
M_LICENSE_LIST_BSON
M_LICENSE_ONLINE

The following L47 specific module commands are available:

M4_VERSIONNO
M4_SYSTEMID
M4_SYSTEM_STATUS
M4_CAPTURE_SIZE
M4_TIME

License commands
M4_LICENSE_INFO
Deprecated license commands
M4_LIC_PES
M4_LIC_PORTS_1G
M4_LIC_PORTS_10G
M4_LIC_PORTS_40G

Replay file commands

M4_REPLAY_PARSE_START
M4_REPLAY_PARSE_STOP
M4_REPLAY_PARSE_STATE
M4_REPLAY_PARSER_PARAMS
M4_CAPTURE_FILE_LIST
M4_CAPTURE_FILE_LIST_BSON
M4_REPLAY_FILE_LIST
M4_REPLAY_FILE_LIST_BSON
M4_REPLAY_FILE_DELETE
M4_CAPTURE_FILE_DELETE

TLS commands:
M4_TLS_CIPHER_SUITES

M_LICENSE_DEMO_INFO

Returns info about the demo status of the module.

demo coded byte, specifies if this is a demo module or not

● NOT_DEMO	(0)	This is not a demo module (but a regular module)
● DEMO	(1)	This is a demo module

valid coded byte, if this is a demo module, specifies if the demo license is valid

● INVALID	(0)	The demo licence is invalid (expired)
● VALID	(1)	The demo licence is valid

permanent coded byte, if this is a demo module and the demo license is valid, specifies if the demo license is permanent

● NOT_PERMANENT	(0)	The demo license is not permanent
● PERMANENT	(1)	The demo license is permanent

expiration long integer, if this is a demo module and the demo license is valid and not permanent, specifies the expiration date of the demo license - in seconds since Jan 1, 1970.

Summary get only, value type: B, B, B, L

Example get:

```
1 M_LICENSE_DEMO_INFO ?
1 M_LICENSE_DEMO_INFO  DEMO VALID NON_PERMANENT 1541030399
```

M_LICENSE_MAINTENANCE_INFO

Returns info about the maintenance license status for the module.

valid coded byte, specifies if the maintenance license is valid

● INVALID	(0)	The maintenance licence is invalid (expired)
● VALID	(1)	The maintenance licence is valid

permanent coded byte, if the maintenance license is valid, specifies if the maintenance license is permanent

● NOT_PERMANENT	(0)	The maintenance license is not permanent
● PERMANENT	(1)	The maintenance license is permanent

expiration long integer, if the maintenance license is valid and not permanent, specifies the expiration date of the maintenance license - in seconds since Jan 1, 1970.

Summary get only, value type: B, B, B, L

Example get:

```
1 M_LICENSE_MAINTENANCE_INFO ?
1 M_LICENSE_MAINTENANCE_INFO VALID NON_PERMANENT 1541030399
```

M_LICENSE_CWB_DETECTED

Returns it clock-windback is detected.

detected	coded byte, specifies if clock-windback is detected	
● NO	(0)	Clock-windback has not been detected
● YES	(1)	Clock-windback has been detected

Summary get only, value type: B

Example get:

```
1 M_LICENSE_CWB_DETECTED ?
1 M_LICENSE_CWB_DETECTED NO
```

If clock-windback has been detected the chassis is locked and no reservations of ports can be performed. To recover from clock-windback, set the system time correct (via the M4_SYSTEM_TIME command) and perform a license update (via the M_LICENSE_UPDATE command)

M_LICENSE_UPDATE

This command instructs the chassis to update its local license information from FlexNet Operations.
The chassis can be configured in on-line and off-line mode (by the M_LICENSE_ONLINE command)
In on-line mode, the chassis sends a capability request to FlexNet Operations and receives a capability response.
In offline mode a capability response must be downloaded from FlexNet Operations and uploaded to the chassis (in /xbin/lic47/)
The capability response is parsed and the license info is stored locally in trusted storage.
A capability response has a lifetime of one day.
The result of the license update operation can be retrieved by M_LICENSE_UPDATE_STATUS.

Summary set only, value type: -

Example get:

```
1 M_LICENSE_UPDATE
```

M_LICENSE_UPDATE_STATUS

Returns the status of the latest license update operations.

update _state	coded byte, specifies the state of the license update procedure	
● NONE	(0)	Default state.
● UPDATING	(1)	License update is in progress
● UPDATE_SUCCESS	(2)	License update succeeded
● UPDATE_FAIL	(3)	License update failed
last_update	long integer, time for the last update request - in seconds since Jan 1, 1979	
last_success	long integer, time for the last successful update - in seconds since Jan 1, 1979	
last_fail	long integer, time for the last failed update - in seconds since Jan 1, 1979	
Info	string, info about the last license update operation - reason for failed update.	

Summary get only, value type: B, L, L, L, S

Example get:

```
1 M_LICENSE_UPDATE_STATUS ?
1 M_LICENSE_UPDATE_STATUS UPDATE_SUCCESS 1537275653 1537275659 0 "License info updated"
```

M_LICENSE_LIST_BSON <bson document>

Returns a list of locally stored licenses - formatted as a BSON document.
The current format of the BSON document is:

```
key: num_features              type:    INT32 val:              3
key: feature_list              type:    ARRAY
  key: 0                         type:   DOCUMENT
       key: source                    type:     UTF8 val: TRUSTED STORAGE
       key: name                      type:     UTF8 val: L47_Packet_Engine
       key: version                   type:     UTF8 val: 1.0
       key: count                     type:     INT32 val:              4
       key: available                 type:     INT32 val:              4
       key: issued                    type:     INT64 val:     1522886400
       key: start                     type:     INT64 val:     1521590400
       key: expire                    type:     INT64 val:              0
       key: permanent                 type:     INT32 val:              1
       key: valid                     type:     INT32 val:              1
       key: valid_str                 type:     UTF8 val:
  key: 1                         type:   DOCUMENT
       key: source                    type:     UTF8 val: TRUSTED STORAGE
       key: name                      type:     UTF8 val: L47_Port_Type_1
       key: version                   type:        UTF8 val: 1.0
       key: count                     type:     INT32 val:              2
       key: available                 type:     INT32 val:              2
       key: issued                    type:     INT64 val:     1522886400
       key: start                     type:     INT64 val:     1521590400
       key: expire                    type:     INT64 val:              0
       key: permanent                 type:     INT32 val:              1
       key: valid                     type:     INT32 val:              1
       key: valid_str                 type:     UTF8 val:
  key: 2                         type:   DOCUMENT
       key: source                    type:     UTF8 val: TRUSTED STORAGE
       key: name                      type:     UTF8 val: Maintenance
       key: version                   type:     UTF8 val: 1.0
       key: count                     type:     INT32 val:              1
       key: available                 type:     INT32 val:              0
       key: issued                    type:     INT64 val:     1522886400
       key: start                     type:     INT64 val:     1522800000
       key: expire                    type:     INT64 val:     1554508799
       key: permanent                 type:     INT32 val:              0
       key: valid                     type:     INT32 val:              1
       key: valid_str                 type:     UTF8 val:
```

M_LICENSE_ONLINE

Configures the chassis in on-line or off-line mode. The on-line mode configuration defines two different license update procedures as descreibed for the M_LICENSE_UPDATE command.
In on-line mode the license update procedure requires access to the Internet. In off-line mode the license update procedure can be performed without access to the Internet.

M4_VERSIONNO version

Returns a version string containing a combination of information regarding the software version and the build environment. The first part of the string is the software build version.

version:	string

Summary get only, value type: S

Example get:

```
        1 M4_VERSIONNO ?
        1 M4_VERSIONNO  "0.1.0
2014-12-17-055000[xena47hp:cu][3.14.4-200.fc20.x86_64]c95923b"
```

In this case the software version is 0.1.0.

M4_SYSTEMID sys_id

Returns the L47 modules System ID, which is an identification of the module HW. The licence installed on a L47 module is only valid for one particular System ID, and must match the L47 module System ID.

sys_id:	string, the System ID of the L47 module.

Example get:

```
        1 M4_SYSTEMID ?
        1 M4_SYSTEMID “7ea0f7d001cea4e79a110aa8c78222fd12665326acbcaf13282f37472bc596ec”
```

M4_SYSTEM_STATUS status

Returns the status of the L47 module.

NOTE: If the licence installed on the L47 module is not valid for the HW, this is the only valid module command possible.

status	string, “OK” or “System ID is incorrect. The L47 Module cannot be used”

Summary get only, value type: S

Example get:

```
1 M4_SYSTEM_STATUS ?
1 M4_SYSTEM_STATUS “OK”
```

M4_CAPTURE_SIZE

See section ‘Capture Commands’

M4_SYSTEM_TIME time

Sets or returns the modules system time in UTC.

year	integer, the year
month	integer, the month
day	integer, the day of the month
hour	integer, the hour
minute	integer, the minute
second	integer, the second

Summary get/set, value type: I, I, I, I, I, I

Example get:

```
1 M4_SYSTEM_TIME ?
1 M4_SYSTEM_TIME 2018 9 19 7 25 00
```

M4_TIME time

Returns the module time in msec.

time	long integer, the current time (mSec since module restart)

Summary get only, value type: L

Example get:

```
1 M4_TIME ?
1 M4_TIME 757327825
```

M4_LICENSE_INFO info...
Returns number of available and free PE licenses.
Only ‘available’ number of PEs can simultaneously be assigned to reserved ports.

PEs available:	integer, number of PEs that are licensed on the module, and can be used simultaneously.
PEs free:	integer, number of free PE licenses on the module
1G available:	integer, number of 1G licenses on the module, that can be used simultaneously.
1G free:	integer, number of free 1G licenses on the module
10G available:	integer, number of 10G licensed on the module, that can be used simultaneously.
10G free:	integer, number of free 10G licenses on the module
25G available:	integer, number of 25G licenses on the module, that can be used simultaneously.
25G free:	integer, number of free 25 port licenses on the module
40G available:	integer, number of 40G licenses on the module, that can be used simultaneously.
40G free:	integer, number of free 40G port licenses on the module

Summary get only, value type: I, I, I, I, I, I, I, I, I, I

Example get:

```
1 M4_LICENSE_INFO ?
1 M4_LICENSE_INFO 28 20 4 2 0 0 4 4 0 0
```

In this case the module contains 28 PE licenses, of which 8 currently is in use (and hence 20 free).
The module contains 4 licenses for 1G of which 2 currently is in use (and hence 2 free)
The module contains 0 licenses for 10G.
The module contains 4 licenses for 25G of which 0 currently is in use (and hence 4 free)
The module contains 0 licenses for 40G.

M4_LIC_PES (deprecated)

M4_LIC_PORTS_1G (deprecated)

M4_LIC_PORTS_10G (deprecated)

M4_LIC_PORTS_40G (deprecated)

M4_REPLAY_PARSE_START file_name
Command to start parsing an uploaded Capture File (in PCAP format) intended for use in a replay test scenario. The result of the parsing - if successful - is a Replay File (in BSON format) with the same name as the Capture File, which can be used as parameter to P4G_REPLAY_FILE_NAME command. If parsing is

unsuccessful, a Replay File is created containing the parse result. The M4_REPLAY_FILE_INFO_BSON command can be used to get information about a Replay File - including the parse result.
PCAP Capture Files can be uploaded to the L47 chassis using FTP. The 'root' location of Capture Files uploaded manually by the user is /var/ftp/pub/replay/pcap/. Three subdirectories exists: cache/, user/ and xena/. cache / and xena/ is used by XenaConnect, and user/ is intended for manual upload and parsing of Capture Files.
A similar directory structure is present for Replay Files generated by the parsing, and the 'root' location is /var/ftp/pub/replay/bson/.

file_name:	string, filename (including relative path and excluding the '.pcap' extension).

Summary set only, value type: S

Example set:

```
1 M4_REPLAY_PARSE_START “user/capture2”
```

M4_REPLAY_PARSE_STOP

Command to stop parsing a Capture File. Parsing of very large Capture Files may take several seconds, and may be aborted using this command.

No parameters

Summary set only

Example set:

```
1 M4_REPLAY_PARSE_STOP
```

M4_REPLAY_PARSE_STATE state

Only one Capture File can be parsed at a time. This command returns the state of the parser, which can be PARSING or OFF. M4_REPLAY_PARSE_START command is only accepted when the parser state is OFF.

status coded byte, state of the replay parser

● OFF	(0)	Parser is OFF and ready to receive a START command
● PARSING	(1)	Parster is busy, and not ready to receive a START command

Summary get only, value type: B

Example get:

```
1 M4_REPLAY_PARSE_STATE ?
1 M4_REPLAY_PARSE_STATE “OFF”
```

M4_REPLAY_PARSER_PARAMS params
Configuration of parameters for the parsing of pcap files.

tcp_port	integer, server TCP Port of dummy TCP connection inserted in UDP only replay files

Summary set, value type: I

Example set:

```
1 M4_REPLAY_PARSER_PARAMS 8080
```

M4_CAPTURE_FILE_LIST file_list
Returns a list of Capture Files ('.pcap' files) in the 'user' Capture File directory (/var/ftp/pub/replay/pcap/user/).

file_list	string, comma separated list of filenames excluding the '.pcap' extension.

Summary get only, value type: string

Example get:

```
1 M4_CAPTURE_FILE_LIST ?
1 M4_CAPTURE_FILE_LIST "capture,capture1,capture10,capture11,capture12"
```

M4_CAPTURE_FILE_LIST_BSON file_list
Works as M4_CAPTURE_FILE_LIST, but returns the file list formatted as a BSON document.

M4_REPLAY_FILE_LIST file_list
Returns a list of Replay Files ('.bson' files) in the 'user' Replay File directory (/var/ftp/pub/replay/bson/user/).

file_list	string, comma separated list of filenames excluding the '.bson' extension..

Summary get only, value type: string

Example get:

```
1 M4_REPLAY_FILE_LIST ?
1 M4_REPLAY_FILE_LIST "capture,capture1,capture10,capture11,capture12"
```

M4_REPLAY_FILE_LIST_BSON file_list

Works as M4_REPLAY_FILE_LIST, but returns the file list formatted as a BSON document.

M4_CAPTURE_FILE_DELETE file_name

Command to delete a Capture File ('.pcap' file) in the Capture File directory (/var/ftp/pub/replay/pcap/). For information about the location and directory structure for the Capture Files, see: M4_REPLAY_PARSE_START

file_name:	string, file name (including relative path and excluding the '.pcap' extension).

Summary set only, value type: S

Example set:

```
1 M4_CAPTURE_FILE_DELETE "user/capture2"
```

M4_REPLAY_FILE_DELETE file_name

Command to delete a Replay File ('.bson' file) in the Replay File directory (/var/ftp/pub/replay/bson/). For information about the location and directory structure for the Replay Files, see: M4_REPLAY_PARSE_START

file_name:	string, file name (including relative path and excluding the '.bson' extension).

Summary set only, value type: S

Example set:

```
1 M4_REPLAY_FILE_DELETE "user/capture2"
```

M4_TLS_CIPHER_SUITES cipher_suites

Returns a list of supported TLS Cipher Suites.

cipher_suites	hexdata, list of IANA values of supported cipher suites

Summary get only, value type H*

Example get

```
1 M4_TLS_CIPHER_SUITES 0x00040005000A0016002F...
```

The supported cipher suites are:

00 04	TLS_RSA_WITH_RC4_128_MD5
00 05	TLS_RSA_WITH_RC4_128_SHA
00 0A	TLS_RSA_WITH_3DES_EDE_CBC_SHA
00 16	TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
00 2F	TLS_RSA_WITH_AES_128_CBC_SHA
….	

Port Commands

The port command names all have the form P_xxx and require both a module index and a port index before the parameter name.

In general, port parameters can only be changed while pot state is *off*

The following port commands supported by the L47 chassis are identical to the commands supported by the L23 products:

P_RESERVATION
P_RESERVEDBY
P_RESET
P_INTERFACE
P_SPEEDSELECTION (deprecated)
P_SPEED
P_RECEIVESYNC
P_COMMENT

The following L47 specific module commands are available:

P4_PCI_INFO
P4_FW_VER
P4_DEV_NAME
P4_PORT_TYPE
P4_CLEAR
P4_VLAN_OFFLOAD
P4_ARP_CONFIG
P4_NDP_CONFIG
P4_CAPABILITIES
P4_SPEEDSELECTION
P4_MAX_PACKET_RATE
P4_APTITUDES
P4_LICENSE_INFO

In addition to the above port commands, the sections Traffic Commands, Packet Engine Commands, Statistics Commands and Capture Commands also contains port based commands.

P4_PCI_INFO vendor_id device_id sub_vendor_id sub_device_id rev

Report the ports PCI info

vendor_id:	integer, PCI Vendor ID
device_id:	integer, PCI Device ID
sub_vendor_id:	integer, PCI Subsystem Vendor ID
sub_device_id:	integer, PCI Subsystem Device ID
rev:	integer, Revision

Summary get only

Example:

```
    1/0 P4_PCI_INFO ?
    1/0 P4_PCI_INFO 0x00008086 0x00001572 0x00008086 0x00000007 0
```

(this is a port on a Intel X710 NIC)

P4_FW_VER major minor

Report the firmware version of the port (NIC)

major:	integer, Major firmware version
minor:	integer, Minor firmware version

Summary get only

Example:

```
    1/0 P4_FW_VER ?
    1/0 P4_FW_VER 4 22
```

P4_DEV_NAME name

Report the name of the device (NIC) on which the port is located.

name:	string, Name of the device (NIC) on which the port is located

Summary get only

Example:

```
    1/0 P4_DEV_NAME ?
```

```
1/0 P4_DEV_NAME “Intel X710”
```

P4_PORT_TYPE type_number type_string

Report the port type. The different possible ports are divided into types.

type_number: integer, enumerated port type

- 1 1G/10G
- 2 1G/2.5G/5G/10G
- 3 1G/10G/25G
- 4 40G

type_string: string, textual representation of the port type

Summary get only

Example:

```
1/0 P4_LICENSE_PORT_TYPE ?
1/0 P4_LICENSE_PORT_TYPE 2 “1G/2.5G/5G/10G”
```

P_RESET

Command to:

- Sets the Port State to OFF
- Set all port configuration parameters to default values
- Delete all configured Connection Groups for the port
- Clear all counters

No parameters

Summary set only

Example set:

```
1/0 P_RESET
```

P4_CLEAR

Command to:

- Sets the Port State to OFF
- Delete all configured Connection Groups for the port

No parameters

Summary set only

Example set:

```
        1/0 P_CLEAR
```

P4_ARP_CONFIG rate rto retries

Configure the value of the arp request transmission rate, retransmission timeout and max. retries.

rate:	integer, ARP Request transmission rate (requests / sec) - must be larger than 0
rto:	integer, ARP Request retransmission timeout [ms] - must be larger than 0
retries:	byte, Max. ARP Request retransmission retries

Summary set, value type: I, I, B

Example set:

```
        1/0 P4_ARP_CONFIG  1000  1000  3
```

P4_NDP_CONFIG rate rto retries

Configure the value of the NDP Neighbor Solicitation transmission rate, retransmission timeout and max. retries.

rate:	integer, NDP Neighbor Solicitation transmission rate (requests / sec) - must be larger than 0
rto:	integer, NDP Neighbor Solicitation retransmission timeout [ms] - must be larger than 0
retries:	byte, Max. NDP Neighbor Solicitation retransmission retries

Summary set, value type: I, I, B

Example set:

```
        1/0 P4_NDP_CONFIG  1000  4000  3
```

P4_VLAN_OFFLOAD offload

Specifies if 802.1Q VLAN tag should be inserted and stripped by the Ethernet device. If VLAN Offload is switched ON, VLAN tags will not be present in frames captured by the L47 Server.

offload: coded byte, specifies if VLAN Offload is switched ON

- ON (1) (802.1Q VLAN tags is inserted and stripped by Ethernet device - default)
- OFF (0) (802.1Q VLAN tags is inserted and stripped by L47 Server)

Summary get/set, value types: B

Example, set:

```
        1/0 P4_VLAN_OFFLOAD ON
```

P4_CAPABILITIES auto 1g 2.5g 5g 10g 25g 40g 50g 100g

Report the speeds supported by the port.

auto:	byte, autoneg supported
1g:	byte, 1G speed supported
2.5g:	byte, 2.5G speed supported
5g:	byte, 5G speed supported
10g:	byte, 10G speed supported
25g:	byte, 25G speed supported
40g:	byte, 40G speed supported
50g:	byte, 50G speed supported
100g:	byte, 100G speed supported

Summary get only

Example:

```
        1/0 P4_CAPABILITIES ?
        1/0 P4_CAPABILITIES 0 1 1 1 1 0 0 0 0
```

(this port supports both 1G, 2.5G, 5G and 10G operation, but not AUTO)

P4_SPEEDSELECTION speed

Sets the port speed. The selected speed must be one of the speeds supported by the port, which can be retrieved with P4_CAPABILITIES.

speed: coded byte, specifies the speed of the port

- AUTO (0) Auto negotiation
- F1G (1) 1G
- F2_5G (2) 2.5G
- F5G (3) 5G
- F10G (4) 10G
- F25G (5) 25G
- F40G (6) 40G
- F50G (7) 50G
- F100G (8) 100G

Summary get/set, value type: B

Example set:

```
        1/0 P4_SPEEDSELECTION F25G
```

P4_MAX_PACKET_RATE mode rate window

Specifies the maximum number of packets per second allowed to be transmitted on the port.

mode	coded byte, specifies the mode of the max. pps mechanism	
● AUTOMATIC	(0)	pps rate is automatically set to a safe number
● MANUAL	(1)	pps rate is pacified by the pps parameter
rate	integer, max. number of packets per second to transmit on this port	
window	integer, time window [us] to measure the pps rate	

NOTE: Currently AUTOMATIC is not implemented. Setting the mode to AUTOMATIC will effectively disable max. Packet rate limitation.

Summary get/set, value type: B, I, I

Example set:

```
        1/0 P4_MAX_PACKET_RATE MANUAL 2000000 10
```

(sets the max. pps rate to 2 million packets per second, measured over a period of 10 usec.)

P4_APTITUDES <bson document>

Returns the ports aptitudes - i.e. what is possible to configure on the port in terms of features and performance.

Current format of the bson document:

```
key: chassis                  type:      INT32 val:               2
key: tcp_udp                  type:   DOCUMENT
  key: cc                       type:      INT32 val:         4000000
key: tls                      type:   DOCUMENT
  key: supported                type:       BOOL val:            true
  key: cc                       type:      INT32 val:          200000

Where chassis has the following meaning:
0: CHASSIS_TYPE_UNKNOWN
1: CHASSIS_TYPE_APPLIANCE
2: CHASSIS_TYPE_BAY
```

```
3: CHASSIS_TYPE_COMPACT
4: CHASSIS_TYPE_SAFIRE
```

Summary get only, value type: B []

P4_LICENSE_INFO info...

Returns information on the license assigned to the port - if any.

present coded byte, specifies if a license is assigned to the port

- NOT_PRESENT (0) No license is assigned to the port, subsequent parameters are N/A
- PRESENT (1) A license is assigned to the port

speed: coded byte, if a license is assigned to the port, specifies the speed of the license

- AUTO (0) Auto negotiation
- F1G (1) 1G
- F2_5G (2) 2.5G
- F5G (3) 5G
- F10G (4) 10G
- F25G (5) 25G
- F40G (6) 40G
- F50G (7) 50G
- F100G (8) 100G

permanent coded byte, if a license is assigned to the port, specifies if the license is permanent

- NOT_PERMANENT (0) The license is not permanent
- PERMANENT (1) The license is permanent

expiration long integer, if a license is assigned to the port and it is not permanent, specifies the expiration date of the license - in seconds since Jan 1, 1970.

Connection Group Commands

Connection Group Commands deal with configuration of TCP connections and are specific to L47. The commands have the form P4G_XXX and require a module index and port index before the command name, and a connection group index after the command name.

The following Connection Group Commands are available:

General commands:
P4G_INDICES
P4G_CREATE
P4G_DELETE
P4G_ENABLE
P4G_COMMENT

Test definition commands:
P4G_ROLE
P4G_IP_VERSION
P4G_CLIENT_RANGE
P4G_SERVER_RANGE
P4G_IPV6_CLIENT_RANGE
P4G_IPV6_SERVER_RANGE
P4G_L4_PROTOCOL
P4G_LP_TIME_SCALE
P4G_LP_SHAPE
P4G_TEST_APPLICATION

Raw test application configuration commands:
P4G_RAW_TEST_SCENARIO
P4G_RAW_PAYLOAD_TYPE
P4G_RAW_PAYLOAD_TOTAL_LEN
P4G_RAW_PAYLOAD
P4G_RAW_PAYLOAD_REPEAT_LEN
P4G_RAW_HAS_DOWNLOAD_REQ
P4G_RAW_DOWNLOAD_REQUEST
P4G_RAW_REQUEST_REPEAT
P4G_RAW_RX_PAYLOAD_LEN
P4G_RAW_CLOSE_CONN
P4G_RAW_UTILIZATION
P4G_RAW_TX_DURING_RAMP
P4G_RAW_TX_TIME_OFFSET
P4G_RAW_BURSTY_TX
P4G_RAW_BURSTY_CONF

P4G_RAW_CONN_INCARNATION
P4G_RAW_CONN_REPETITIONS
P4G_RAW_CONN_LIFETIME

Replay test application configuration commands
P4G_REPLAY_FILE_INDICES
P4G_REPLAY_FILE_NAME
P4G_REPLAY_FILE_CLEAR
P4G_REPLAY_UTILIZATION
P4G_REPLAY_USER_INCARNATION
P4G_REPLAY_USER_REPETITIONS

L2 configuration commands:
P4G_NAT
P4G_VLAN_ENABLE
P4G_VLAN_TCI
P4G_L2_CLIENT_MAC
P4G_L2_SERVER_MAC
P4G_L2_USE_ADDRESS_RES
P4G_L2_USE_GW
P4G_L2_GW
P4G_L2_IPV6_GW

IPv4 and IPv6 protocol configuration commands:
P4G_IP_DS_TYPE
P4G_IP_DS_VALUE
P4G_IP_DS_MASK
P4G_IP_DS_STEP
P4G_IP_DS_MINMAX
P4G_IPV6_TRAFFIC_CLASS
P4G_IPv6_FLOW_LABEL

TCP protocol configuration commands:
P4G_TCP_MSS_TYPE
P4G_TCP_MSS_MINMAX
P4G_TCP_MSS_VALUE
P4G_TCP_WINDOW_SIZE
P4G_TCP_WINDOW_SCALING
P4G_TCP_DUP_THRES
P4G_TCP_SYN_RTO
P4G_TCP_RTO
P4G_TCP_RTO_MINMAX
P4G_TCP_CONGESTION_MODE

UDP protocol configuration commands:

P4G_UDP_PACKET_SIZE_TYPE
P4G_UDP_PACKET_SIZE_MINMAX
P4G_UDP_PACKET_SIZE_VALUE

TLS protocol configuration commands:
P4G_TLS_ENABLE
P4G_TLS_CLOSE_NOTIFY
P4G_TLS_CIPHER_SUITES
P4G_TLS_MAX_RECORD_SIZE
P4G_TLS_CERTIFICATE_FILE_NAME
P4G_TLS_PRIVATE_KEY_FILENAME
P4G_TLS_DHPARAMS_FILENAME
P4G_TLS_SERVER_NAME
P4G_TLS_PROTOCOL_VER
P4G_TLS_MIN_REQ_PROTOCOL_VER

General commands

P4G_INDICES gid gid …

The full list of connection groups on this port. These are the sub-index that are used for the parameters that specify TCP connection behavior.

gid: integer, an index identifying a connection group.

Summary get only, value types: I*

Example, get:

```
1/0 P4G_INDICES ?
1/0 P4G_INDICES 0 5 9
```

P4G_CREATE [gid]

Creates an empty connection group with the specified sub-index value.

gid: integer, the sub-index value of the connection group to create.

Summary, set only, connection group index

Example, set:

```
1/0 P4G_CREATE [0]
```

P4G_DELETE [gid]

Deletes a connection group with the specified sub-index value.

gid: integer, the sub-index value of the connection group to delete.

Summary, set only, connection group index

Example, set:

```
1/0 P4G_DELETE [0]
```

P4G_ENABLE [gid] status

Enable/disable/suppress a previously created connection group with the specified sub-index value.

gid:	integer, the sub-index value of the connection group to create.
status:	coded byte, specifies the state of the connection group

● DISABLED	(0)	(CG will not be used when traffic is started)
● ENABLED	(1)	(CG will be used when traffic is started - default)
● SUPPRESS	(2)	not yet implemented

Summary, get/set, connection group index, value type: B

Example, set:

```
1/0 P4G_ENABLE [0] DISABLED
```

P4G_COMMENT [gid] comment

The description of a connection group.

gid:	integer, the sub-index value of the connection group.
comment:	string, the description of the connection group

Summary, get/set, connection group index, value type: S

Example, set:

```
1/0 P4G_COMMENT [0] “1M clients and 1 server, CPS=1M”
```

Test definition commands

P4G_ROLE [gid] role

Specifies the client or server role for this connection group. A server passively waits for the clients to establish connections.

gid:	integer, the sub-index value of the connection group.
role:	coded byte, specifies the role of the connection group

- CLIENT(0) (the connection group will actively open connections)
- SERVER (1) (the connection group will listen for connections)

Summary, get/set, connection group index, role value type: B

Example, set:

```
1/0 P4G_ROLE [0] CLIENT
1/1 P4G_ROLE [1] SERVER
```

P4G_IP_VERSION [gid] version

Specifies either IPv4 or IPv6

gid:	integer, the sub-index value of the connection group.
ipaddress:	coded byte

- IPV4 (4)
- IPV6 (6)

Summary, get/set, connection group index, value type: B

Example, set:

```
1/0 P4G_IP_VERSION [0] IPv6
```

P4G_CLIENT_RANGE [gid] ipaddress nbaddrs port nbports maxnbaddrs

Specifies a number of client sockets (ip address, port number)

gid:	integer, the sub-index value of the connection group.
ipaddress:	address, the start ip address of the address range
nbaddrs:	integer, the number of ip addresses
port:	integer, the start tcp port number, of the port range
nbports:	integer, the number of tcp ports

maxnbaddrs:	integer, the maximum number of ip addresses that this connection group will use, when connection incarnation is set to REINCARNATE

Summary, get/set, connection group index, value type: A, I, I, I, I

Example, set:

```
1/0 P4G_CLIENT_RANGE [0] 10.0.2.1 1000 32000 1000 65535
```

P4G_SERVER_RANGE [gid] ipaddress nbaddrs port nbports

Specifies a number of server sockets (ip address, port number)

gid:	integer, the sub-index value of the connection group.
ipaddress:	address, the start ip address of the address range
nbaddrs:	integer, the number of ip addresses
port:	integer, the start tcp port number, of the port range
nbports:	integer, the number if tcp ports

Summary, get/set, connection group index, value type: A, I, I, I

Example, set:

```
1/1 P4G_SERVER_RANGE [0] 10.0.0.1 1 80 1
```

P4G_IPV6_CLIENT_RANGE [gid] ipv6address nbaddrs port nbports maxnbaddrs

Specifies a number of client sockets (ipv6 address, port number)

gid:	integer, the sub-index value of the connection group.
ipv6address:	hexbytes, the start ip address of the address range
nbaddrs:	integer, the number of ip addresses
port:	integer, the start tcp port number, of the port range
nbports:	integer, the number of tcp ports
maxnbaddrs:	long integer, the maximum number of ip addresses that this connection group will use, when connection incarnation is set to REINCARNATE

Summary, get/set, connection group index, value type: H*16, I, I, I, L

Example, set:

```
1/0 P4G_IPV6_CLIENT_RANGE [0] 0xfffe0000000000000000000000000001 1000 32000 1000 65535
```

P4G_IPV6_SERVER_RANGE [gid] ipv6address nbaddrs port nbports
Specifies a number of server sockets (ipv6 address, port number)

gid:	integer, the sub-index value of the connection group.
ipv6address:	hexbytes, the start ip address of the address range
nbaddrs:	integer, the number of ip addresses
port:	integer, the start tcp port number, of the port range
nbports:	integer, the number of tcp ports

Summary, get/set, connection group index, value type: H*16, I, I, I

Example, set:

```
1/1 P4G_IPV6_SERVER_RANGE [0] 0xfffe0000000000000000000000000002 1 80 1
```

P4G_L4_PROTOCOL [gid] protocol
Specifies either TCP or UDP as layer 4 protocol

gid:	integer, the sub-index value of the connection group.
protocol:	coded byte

- TCP (0)
- UDP (1)

Summary, get/set, connection group index, value type: B

Example, set:

```
1/0 P4G_L4_PROTOCOL [0] TCP
```

P4G_LP_TIME_SCALE [gid] timescale
Specifies the time scale of the load profile.

gid:	integer, the sub-index value of the connection group.
timescale:	coded byte, specifying the time scale

- MSECS(0)
- SECONDS (1)
- MINUTES (2)
- HOURS (3)

Summary, get/set, connection group index, value type: B

Example,

```
        1/0 P4G_LP_TIME_SCALE [0] msecs
```

P4G_LP_SHAPE [gid] t0 t1 t2 t3

Specifies a load profile time durations. Time is measured from the beginning of the test (P4G_TRAFFIC on).

gid:	integer, the sub-index value of the connection group.
t0:	integer, ramp-up start time
t1:	integer, ramp-up duration
t2:	integer, traffic duration
t3:	integer, ramp-down duration

Summary, get/set, connection group index, value type: I, I, I, I

Example,
Assuming timescale is in seconds: Ramp up begins at 10s and ramp up is done in 10s. Ramp down begins after 10s of traffic and is done in 30s. Total running time: 60s.

```
        1/1 P4G_LP_SHAPE [0] 10 20 10 30
```

P4G_TEST_APPLICATION [gid] application

Configure the application layer behavior. This command affects whether tcp payload is generated. ‘NONE’ means that tcp connections are created according to the client and server ranges, and ramped up/down as specified in the load profile. But no payload is transmitted. ‘RAW’ also generates tcp payload.

gid:	integer, the sub-index value of the connection group.
application:	coded byte, application behavior

● NONE	(0)	(do not generate tcp payload)
● RAW	(1)	(generate raw TCP payload according to configuration)
● REPLAY	(2)	(replays TCP/UDP traffic from a Replay File)

Summary get/set, connection group index, value type: B

Example get/set:

```
        1/0 P4G_TEST_APPLICATION  [0]  RAW
```

Raw test application configuration commands

P4G_RAW_TEST_SCENARIO [gid] scenario

Configure the traffic scenario for RAW mode

gid:	integer, the sub-index value of the connection group.
scenario:	coded byte, traffic scenario

● DOWNLOAD	(0)	(server transmits payload to client)
● UPLOAD	(1)	(client transmits payload to server)
● BOTH	(2)	(payload is transmitted in both directions)
● ECHO	(3)	(client transmits payload to server, server echoes the payload back)

Summary get/set, connection group index, value type: B

Example set:

```
1/0 P4G_RAW_TEST_SCENARIO  [0]  UPLOAD
```

P4G_RAW_PAYLOAD_TYPE [gid] type

Specify the payload content.

gid:	integer, the sub-index value of the connection group.
type:	coded byte, payload generation method

● FIXED	(0)	(payload has a fixed value - as specified by the P4G_RAW_PAYLOAD command)
● INCREMENT	(1)	(payload consist of incrementing bytes)
● RANDOM	(2)	(payload consists of pseudo random bytes with a repeat cycle of 1MBytes)
● LONGRANDOM	(3)	(payload consists of pseudo random bytes with a repeat cycle of 4GBytes)

Summary get/set, connection group index, value type: B

Example set:

```
1/0 P4G_RAW_PAYLOAD_TYPE  [0]  FIXED
```

P4G_RAW_PAYLOAD_TOTAL_LEN [gid] mode length

Configure the total amount of payload to transmit on one connection.

gid:	integer, the sub-index value of the connection group.
mode:	coded byte

- INFINITE (0) (generates payload as long as test is running)
- FINITE (1) (stop generating payload after *length* bytes)

length:	long integer, length of tcp payload

Summary get/set, connection group index, value type: B, L

Example set:

```
1/0 P4G_RAW_PAYLOAD_TOTAL_LEN  [0]  FINITE 4294967296
```

P4G_RAW_PAYLOAD [gid] offset length hexdata

Specify payload as hex bytes. This command can be called several times to build a custom payload.

gid:	integer, the sub-index value of the connection group.
offset:	integer, the offset in the payload buffer where data is to be written
length:	integer, number of bytes to write
hexdata:	hex bytes, payload

Summary get/set, connection group index, value type: I, I, H*

Example set:

```
1/0 P4G_RAW_PAYLOAD [0] 0 4 0x01020304
1/0 P4G_RAW_PAYLOAD [0] 4 4 0x05060708
```

P4G_RAW_PAYLOAD_REPEAT_LEN [gid] length

Specify the length of the payload defined by one or more P4G_RAW_PAYLOAD commands. P4G_RAW_PAYLOAD_REPEAT_LEN number of bytes will be repeated, until P4G_RAW_PAYLOAD_TOTAL_LEN bytes has been transmitted on the connection.

gid:	integer, the sub-index value of the connection group.
length:	integer, length of custom payload

Summary get/set, connection group index, value type: I

Example set:
REQ

```
1/0 P4G_RAW_PAYLOAD_REPEAT_LEN [0] 8
```

P4G_RAW_HAS_DOWNLOAD_REQ [gid] request

Specify whether server waits for a request from the client before it starts transmitting.
NOTE: This parameter is N/A when P4G_L4_PROTOCOL is configured as UDP

gid:	integer, the sub-index value of the connection group.
request:	coded byte, expect request

- NO (0) (do not wait for a request before transmitting)
- YES (1) (wait for a request before transmitting)

Summary get/set, connection group index, value type: B

Example set:

```
1/0 P4G_RAW_HAS_DOWNLOAD_REQ [0] YES
```

P4G_RAW_DOWNLOAD_REQUEST [gid] length hexdata

Specify content of the download request send by the client and expected by the server as hex bytes.
NOTE: This parameter is N/A when P4G_L4_PROTOCOL is configured as UDP

gid:	integer, the sub-index value of the connection group.
length:	integer, number of bytes to write
hexdata:	hex bytes, request content

Summary get/set, connection group index, value type: I, H*

Example set:

```
1/0 P4G_RAW_DOWNLOAD_REQUEST [0] 17 0x58656e612072617720646f776e6c6f6164
```

P4G_RAW_REQUEST_REPEAT [gid] mode repeat

Specify number of request/response transactions to perform - if P4G_RAW_HAS_DOWNLOAD_REQ is set to Yes.
NOTE: This parameter is N/A when P4G_L4_PROTOCOL is configured as UDP

gid:	integer, the sub-index value of the connection group.
mode:	coded byte

- INFINITE (0) (repeats request/response transactions as long as test is running)
- FINITE (1) (stop generating request/response transactions after *repeat* cycles)

repeat: integer, number of request/response transactions to perform , if mode is FINITE.

Summary get/set, connection group index, value type: B, I

Example set:

```
        1/0 P4G_RAW_REQUEST_REPEAT [0] FINITE 100
```

P4G_RAW_RX_PAYLOAD_LEN [gid] mode length

Specify the length of the payload the Client should expect to receive before sending the next download request to the Server. Should be configured identical to the P4G_RAW_PAYLOAD_TOTAL_LEN for the Server. If mode is set to INFINITE,effectively no request/response repetitions will be performed.
NOTE: This parameter is N/A when P4G_L4_PROTOCOL is configured as UDP

gid:	integer, the sub-index value of the connection group.
mode:	coded byte

- INFINITE (0) (expects payload as long as test is running)
- FINITE (1) (expects *length* bytes of payload)

length:	long integer, number of payload bytes the client should receive before sending the next request, if mode is FINITE.

Summary get/set, connection group index, value type: B, L

Example set:

```
        1/0 P4G_RAW_RX_PAYLOAD_LEN  [0] FINITE 4096
```

P4G_RAW_CLOSE_CONN [gid] close

Specify whether to close tcp connection when all payload has been transmitted or keep the connection until ramp-down. In RAW_TEST_SCENARIO DOWNLOAD, the server can close the connection, when all payload has been transmitted, and in RAW_TEST_SCENARIO UPLOAD, the client can close the connection, when all payload has been transmitted. In any case, both server and client connection group must be configured with the same value of this parameter.
In RAW_TEST_SCENARIO BOTH (bidirectional), this parameter is N/A and will be ignored.
In a transaction scenario (P4G_RAW_HAS_DOWNLOAD_REQ is set to YES) both client and server can close the connection, when the last transaction has completed. When P4G_RAW_CONN_INCARNATION is set to IMMORTAL of REINCARNATE, and this parameter is set to NONE, connections will be closed after 'connection lifetime' (set by P4G_RAW_CONN_LIFETIME).
NOTE: This parameter is N/A when P4G_L4_PROTOCOL is configured as UDP

gid:	integer, the sub-index value of the connection group.
close:	coded byte, close connection

- NONE (0) (keep the connection open after last byte is transmitted)
- CLIENT (1) (client closes the connection after last byte is receiver/transmitted)

- SERVER (2) (server closes the connection after last byte is transmitted)

Summary get/set, connection group index, value type: B

Example get/set:

```
        1/0 P4G_RAW_CLOSE_CONN [0] CLIENT
```

P4G_RAW_UTILIZATION [gid] utilization

Specify the link layer bandwidth utilization for all the generated traffic from the specified Raw Connection Group.

gid:	integer, the sub-index value of the connection group.
utilization:	integer, utilization specified in ppm.

Summary get/set, connection group index, value type: I

Example get/set:
100%

```
        1/0 P4G_RAW_UTILIZATION [0] 1000000
```

1%

```
        1/0 P4G_RAW_UTILIZATION [0] 10000
```

P4G_RAW_TX_DURING_RAMP [gid] ramp-up ramp-down

Specify if TCP payload transmission should take place during ramp-up and ramp-down.

NOTE: For UDP connections payload transmission will always take place during ramp-up and ramp-down, and this parameter is therefore N/A

gid: integer, the sub-index value of the connection group.
ramp-up: coded byte, close connection

- NO (0) (Start payload transmission after ramp-up has completed)
- YES (1) (Start payload transmission as soon as each connection is established)

ramp-down: coded byte, close connection

- NO (0) (Stop payload transmission before ramp-down is started)
- YES (1) (Payload transmission is stopped when each connection is closed)

Summary get/set, connection group index, value type: B, B

Example set:

```
        1/0 P4G_RAW_TX_DURING_RAMP [0] YES YES
```

P4G_RAW_TX_TIME_OFFSET [gid] start stop

Specify a time offset to the transmit start and stop time, if P4G_TX_DURING_RAMP is set to NO for ramp-up and ramp-down respectively.

gid:	integer, the sub-index value of the connection group.
start:	integer, specify time in ms from ramp-up has completed to start of payload transmit.
stop:	integer, specify time in ms from stop of payload transmit to start of ramp-down.

Summary get/set, connection group index, value type: I, I

Example set:

```
1/0 P4G_RAW_TX_TIME_OFFSET [0] 100 100
```

P4G_RAW_BURSTY_TX [gid] bursty

Enables or disables bursty transmission.

gid:	integer, the sub-index value of the connection group.
bursty:	coded byte, do bursty transmission:

- OFF (0) (bursty transmission is disabled)
- ON (1) (bursty transmission is enabled)

Summary get/set, connection group index, value type: B

Example set:

```
1/0 P4G_RAW_BURSTY_TX [0] ON
```

P4G_RAW_BURSTY_CONF [gid] active inactive

Specifies active and inactive period of bursty transmission in milliseconds. The burst period starts with the active part.

gid:	integer, the sub-index value of the connection group.
active:	integer, specifies the duration in ms of the active part of the burst period.
inactive:	integer, specifies the duration in ms of the inactive part of the burst period.

Summary get/set, connection group index, value type: I, I

Example set:

```
1/0 P4G_RAW_BURSTY_CONF [0] 50 200
```

P4G_RAW_CONN_INCARNATION [gid] mode

Defines the lifecycle of a connection, and how new connections should be established as old connections are closed.

mode coded byte, defines the lifecycle of connections

- ONCE (0) connections are established during the ramp-up phase and not closed until the ramp-down phase of the load profile. That is, each configured connection only exists once.
- IMMORTAL (1) connections are established during the ramp-up phase of the load profile, and

  are closed after the configured lifetime (configured by P4G_RAW_CONN_LIFETIME). As connections closes, new connections are established, attempting to keep the concurrent number of established connections constant. A new connection will have the same IP address as the connection the replaces, but will have a new TCP port number. This will simulate that the user (defined by the client IP address) is living on, and creates new connections as old connections closes.
- REINCARNATE (2) connections are established during the ramp-up phase of the load profile, and

  are closed after the configured lifetime (configured by P4G_RAW_CONN_LIFETIME). As connections closes, new connections are established, attempting to keep the concurrent number of established connections constant. A new connection will have the same TCP port number as the connection the replaces, but will have a new IP address. This will simulate that the user (defined by the client IP address) cease to exist, and new users appears as old users dies.

Summary get/set, connection group index, value type: B

Example set:

```
1/0 P4G_RAW_CONN_INCARNATION [0] IMMORTAL
```

P4G_RAW_CONN_LIFETIME [gid] timescale lifetime

Defines the lifetime of a connection, when P4G_RAW_CONN_INCARNATION is set to IMMORTAL or REINCARNATE.

gid:	integer, the sub-index value of the connection group.
timescale:	coded byte, specifying the time scale

- MSECS(0)
- SECONDS (1)
- MINUTES (2)
- HOURS (3)

lifetime integer, time from a connection is established until it will be closed.

Summary get/set, connection group index, value type: B, I

Example set:

```
1/0 P4G_RAW_CONN_LIFETIME [0] SECONDS 5
```

P4G_RAW_CONN_REPETITIONS [gid] mode repetitions

Defines how many times a new connection should be created, after an old connection has been closed, when P4G_RAW_CONN_INCARNATION is set to IMMORTAL or REINCARNATE.

gid: integer, the sub-index value of the connection group.
mode: coded byte

- INFINITE (0) new connections are generated as long as the load profile allows.
- FINITE (1) each configured connection is closed and re-established repetitions number of times.

repetitions: integer, number of repetitions

Summary get/set, connection group index, value type: B, I

Example set:

```
1/0 P4G_RAW_CONN_REPETITIONS [0] FINITE 100
```

Replay test application configuration commands

P4G_REPLAY_FILE_INDICES [gid] fid fid ...

Returns an index list of configured Replay Files for this Connection Group. These are the Replay File Index that are used for P4G_REPLAY_FILE_NAME and P4G_REPLAY_FILE_CLEAR commands.
More than one Replay File can be configured for a Connection Group. When configuring a Replay File for a Connection Group, it must have an index.

gid: integer, the sub-index value of the connection group.
fid: integer, index of a configured replay file

Summary get only, value types: I*

Example, get:

```
	1/0 P4G_REPLAY_FILE_INDICES [0] ?
	1/0 P4G_REPLAY_FILE_INDICES [0] 0 1 5
```

P4G_REPLAY_FILE_NAME [gid, fid] file_name

More than one Replay File can be configured for a Connection Group. When configuring a Replay File for a Connection Group, it must have an index. The indices at which Replay Files are configured does not have to be continuous.

gid	integer, the sub-index value of the connection group.
fid	integer, index of a configured replay file **(must be less than 50)**
file_name	string, file name (including relative path and excluding the '.bson' extension).

Summary get/set, value types: I, I, S

Example set:

```
	1/0 P4G_REPLAY_FILE_NAME [0, 1] "user/capture17"
```

P4G_REPLAY_FILE_CLEAR [gid, fid]

Clears a Replay File index, so no Replay File is configured for that index.

gid	integer, the sub-index value of the connection group.
fid	integer, index of the configured replay file to clear

Summary set only, value types: I, I

Example set:

```
	1/0 P4G_REPLAY_FILE_CLEAR [0, 1]
```

P4G_REPLAY_UTILIZATION [gid] utilization

Specify the link layer bandwidth utilization for all the generated traffic from the specified Replay Connection Group.

gid:	integer, the sub-index value of the connection group.
utilization:	integer, utilization specified in ppm.

Summary get/set, connection group index, value type: I

Example get/set:
100%
```
        1/0 P4G_REPLAY_UTILIZATION [0] 1000000
```
1%
```
        1/0 P4G_REPLAY_UTILIZATION [0] 10000
```

P4G_REPLAY_USER_INCARNATION [gid] mode

Defines the lifecycle of a user and its connections, and how new users should be established as old connections are closed.

mode coded byte, defines the lifecycle of connections

- ONCE (0) users are created and its connections are established during the ramp-up phase and not closed until the ramp-down phase of the load profile. That is, each configured user only exists once.
- IMMORTAL (1) users are created and its connections are established during the ramp-up phase of the load profile. Each connection is closed when all payload in the Replay File for that connection has been transmitted. A user is destroyed when all its connections are closed. As users are destroyed, new users are created, attempting to keep the concurrent number of users constant. A new user will have the same IP address as the user the replaces, but its connections will have new TCP/UDP port numbers. This will simulate that the user is living on, and creates new connections as old connections closes.
- REINCARNATE (2) users are created and its connections are established during the ramp-up phase of the load profile. Each connection is closed when all payload in the Replay File for that connection has been transmitted. A user is destroyed when all its connections are closed. As users are destroyed, new users are created, attempting to keep the concurrent number of users constant. A new user will have a different IP address than the user it replaces, but its connection will have the same TCP/UDP port numbers. This will simulate that the user cease to exist, and new users appears as old users dies.

Summary get/set, connection group index, value type: B

Example set:

```
        1/0 P4G_REPLAY_USER_INCARNATION [0] REINCARNATE
```

P4G_REPLAY_USER_REPETITIONS [gid] mode repetitions

Defines how many times a new user should be created, after an old user has been destroyed, when P4G_REPLAY_USER_INCARNATION is set to IMMORTAL or REINCARNATE.

gid: integer, the sub-index value of the connection group.
mode: coded byte

- INFINITE (0) new users are generated as long as the load profile allows.
- FINITE (1) each configured user is destroyed and re-created repetitions number of times.

repetitions: integer, number of repetitions

Summary get/set, connection group index, value type: B, I

Example set:

```
1/0 P4G_REPLAY_USER_REPETITIONS [0] FINITE 100
```

L2 configuration commands

P4G_NAT [gid] enable

Specify whether to support DUT Source NAT functionality. For TEST_APPLICATION RAW it is sufficient to enable NAT on the Server Connection Group, but for TEST_APPLICATION REPLAY, NAT should be enabled on both Client and Server Connection Groups.

gid:	integer, the sub-index value of the connection group.
enable:	coded byte, specifying whether to enable NAT

- OFF (0)
- ON (1)

Summary, get/set, connection group index, value type: B

Example, get/set:

```
1/0 P4G_NAT [0]  ON
```

P4G_VLAN_ENABLE [gid] enable

Specify whether to insert a VLAN tag header upon transmit.

gid:	integer, the sub-index value of the connection group.
enable:	coded byte, specifying whether to enable VLAN tag

- NO (0)
- YES (1)

Summary, get/set, connection group index, value type: B

Example, get/set:

```
1/0 P4G_VLAN_ENABLE [0]  YES
```

P4G_VLAN_TCI [gid] tci

Specify the VLAN TCI

gid:	integer, the sub-index value of the connection group.
tci:	hex bytes, specifying the 16 bit TCI

Summary, get/set, connection group index, value type: HH

Example, get/set:

```
1/0 P4G_VLAN_TCI [0] 0x1234
```

P4G_L2_CLIENT_MAC [gid] hwaddress embedip

Configure the client hardware address. This is either a single static mac address or an embedding of the four byte ipv4 address into the lower 4 bytes of the 6 byte mac address.

gid:	integer, the sub-index value of the connection group.
hwaddress:	hexdata, the 6 byte mac address specified as hexadecimal
embedip:	coded byte, whether to embed the ip address or not

- DONT_EMBED_IP (0)
- EMBED_IP (1)

Summary, get/set, connection group index, value type: HHHHHH, B

Example, set:

```
1/0 P4G_L2_CLIENT_MAC [0] 0x04F4BC000001 DONT_EMBED_IP
```

P4G_L2_SERVER_MAC

Configure the server hardware address. This is either a single static mac address or an embedding of the four byte ipv4 address into the lower 4 bytes of the 6 byte mac address.

gid:	integer, the sub-index value of the connection group.
hwaddress:	hexdata, the 6 byte mac address specified as hexadecimal
embedip:	coded byte, whether to embed the ip address or not

- DONT_EMBED_IP (0)
- EMBED_IP (1)

Summary, get/set, connection group index, value type: HHHHHH, B

Example, set:

```
1/1 P4G_L2_SERVER_MAC [0] 0x04F4BC000000 EMBED_IP
```

P4G_L2_USE_ADDRESS_RES [gid] usearp

Specify whether to use ARP and NDP to resolve hardware (mac) addresses in the pre_run phase.

gid:	integer, the sub-index value of the connection group.
usearp:	coded byte, specifying whether to use ARP and NDP or not

- NO (0)
- YES (1)

Summary, get/set, connection group index, value type: B

Example, get/set:

```
1/0 P4G_L2_USE_ADDRESS_RES [0]  YES
```

P4G_L2_USE_GW [gid] usegw

Specify whether to use a default gateway when resolving destination mac address.

gid:	integer, the sub-index value of the connection group.
usegw:	coded byte, specifying whether to use gateway or not

- NO (0)
- YES (1)

Summary, get/set, connection group index, value type: B

Example, get/set:

```
1/0 P4G_L2_USE_GW [0]  YES
```

P4G_L2_GW [gid] ipaddress macaddress

Specify a default gateway.

gid:	integer, the sub-index value of the connection group.
ipaddress:	address, ip address of gateway
macaddress:	hexdata, the six bytes mac address of the gateway

Summary, get/set, connection group index, value type: A, HHHHHH

Example, get/set:

```
1/0 P4G_L2_GW [0]  10.0.0.1  0x04F4BC000000
```

P4G_L2_IPV6_GW [gid] ipv6address macaddress

Specify a default gateway for ipv6.

gid:	integer, the sub-index value of the connection group
ipaddress:	hexdata, the 16 bytes of ipv6 address of gateway
macaddress:	hexdata, the six bytes mac address of the gateway

Summary, get/set, connection group index, value type: HHHHHHHHHHHHHHHH, HHHHHH

Example, get/set:

```
      1/0 P4G_L2_IPV6_GW [0]  0x200000aa000000000000000000000001  0x04F4BC000001
```

IP protocol configuration commands

P4G_IP_DS_TYPE [gid] type

Configure the value of the DS field of the IP header.

gid:	integer, the sub-index value of the connection group.
type:	coded byte, specifying how to fill out DS

- FIXED (0) (use fixed value for DS)
- INCREMENT (1) (use incrementing values for DS)
- RANDOM (2) (use pseudorandom values for DS)

Summary, get/set, connection group index, value type: B

Example, get/set:

```
1/0 P4G_IP_DS_TYPE [0] FIXED
```

P4G_IP_DS_VALUE [gid] dsvalue

Specify the (FIXED) value used for DS

gid:	integer, the sub-index value of the connection group.
dsvalue:	hexbyte, the fixed DS value to be used

Summary, get/set, connection group index, value type: H

Example get/set:

```
1/1 P4G_IP_DS_VALUE [0] 0x40
```

P4G_IP_DS_MASK [gid] dsmask

Specify a bit mask to be applied to the DS field. If the fixed value is *fixed*, the current (calculated) value is *curr*, and the mask is *mask*, then the effective DS will be calculated as follows:

(fixed AND (NOT mask)) OR (curr AND mask) or in C syntax
(fixed & (~mask)) | (curr & mask)

gid:	integer, the sub-index value of the connection group.
dsmask:	hexbyte, the DS mask to be used.

Summary get/set, connection group index, value type: H

Example get/set,

```
        1/0 P4G_IP_DS_MASK [0] 0xef
```

P4G_IP_DS_STEP [gid] step

Specifies the incrementing step size for the calculated part of the DS value.
Relevant when P4G_IP_DS_TYPE is set to INCREMENT.

gid:	integer, the sub-index value of the connection group.
step:	hexbyte, the incrementing step size for DS.

Summary get/set, connection group index, value type: H

Example get/set,

```
        1/0 P4G_IP_DS_STEP [0] 0x10
```

P4G_IP_DS_MINMAX [gid] min max

Configure the min and max value of the range for the calculated part of the DS value. Both values are included in the range.
Relevant when P4G_IP_DS_TYPE is set to INCREMENT or RANDOM.

gid:	integer, the sub-index value of the connection group.
min:	hexbyte, minimum value for the calculated part of DS
max:	hexbyte, maximum value for the calculated part of DS

Summary get/set, connection group index, value type: H, H

Example get/set:

```
       1/0 P4G_IP_DS_MINMAX  [0] 0x08 0xff
```

P4G_IPV6_TRAFFIC_CLASS [gid] tcval

Configure the value of the traffic class field of the IPv6 header.

gid:	integer, the sub-index value of the connection group
tcval:	hexbyte, value of the traffic class field

Summary, get/set, connection group index, value type: H

Example, get/set:

```
1/0 P4G_IPV6_TRAFFIC_CLASS [0] 0xaa
```

P4G_IPV6_FLOW_LABEL [gid] flval

Configure the value of the flow label field of the IPv6 header.

gid:	integer, the sub-index value of the connection group
flval:	hexbytes, value of the traffic class field (only lowest 20 bits are valid)

Summary, get/set, connection group index, value type: HHHH

Example, get/set:

```
1/0 P4G_IPV6_FLOW_LABEL [0] 0x00012345
```

TCP protocol configuration commands

P4G_TCP_MSS_TYPE [gid] msstype

Specifies the Maximum Segment Size (MSS) for a connection group. The MSS can either be fixed size identical for all connections in the connection group, incrementing or random. The individual MSS for a specific connection is always constant once the incrementing or random value has been created.

gid:	integer, the sub-index value of the connection group.
msstype:	coded byte, specifying how MSS is set

- FIXED (0) (MSS is constant)
- INCREMENT (1) (MSS varies from min to max in increments of 1)
- RANDOM (2) (MSS varies pseudo randomly between min and max)

Refer to P4G_TCP_MSS_MINMAX command for information on how to configure min and max values.

Summary get/set, connection group index, value type: B

Example get/set:

```
1/0 P4G_TCP_MSS_TYPE [0]  INCREMENT
```

P4G_TCP_MSS_MINMAX [gid] min max

Configure a minimum and maximum value of the range for MSS. Both values are included in the range.
Relevant when P4G_TCP_MSS_TYPE is set to INCREMENT or RANDOM.

gid:	integer, the sub-index value of the connection group.
min:	integer, minimum value of MSS
max:	integer, maximum value of MSS

Summary get/set, connection group index, value type: I, I

Example get/set:

```
      1/0 P4G_TCP_MSS_MINMAX  [0]  576  1400
```

P4G_TCP_MSS_VALUE [gid] mss

Configure the fixed MSS value.
Relevant when P4G_TCP_MSS_TYPE is set to FIXED.

gid:	integer, the sub-index value of the connection group.
mss:	integer, fixed value of MSS

Summary get/set, connection group index, value type: I

Example get/set:

```
      1/0 P4G_TCP_MSS_VALUE  [0]  1023
```

P4G_TCP_WINDOW_SIZE [gid] size

Configure the value of the tcp window.

gid:	integer, the sub-index value of the connection group.
size:	integer, window size in bytes

Summary get/set, connection group index, value type: I

Example get/set:

```
      1/0 P4G_TCP_WINDOW_SIZE  [0]  4096
```

P4G_TCP_WINDOW_SCALING [gid] enable factor

Enable window scaling for the connection group. Note to use windows scaling it need to be enabled in both the client and server connection group. .

gid:	integer, the sub-index value of the connection group.
enable:	code byte, specifying whether to enable window scaling or not

- NO (0)
- YES (1)

factor: integer, scales the window size by the power of two of the given factor.

- Default value is 0 and maximum value is 14 - ignored if window scaling is not enabled

Summary get/set, connection group index, value type: B, I

Example get/set:

```
1/0 P4G_TCP_TCP_WINDOW_SCALING  [0]  YES 1
```

P4G_TCP_DUP_THRES [gid] thres

Configure the value of the tcp duplicate ACK threshold.

gid:	integer, the sub-index value of the connection group.
thres:	byte, duplicate ACK threshold - must be larger than 0

Summary get/set, connection group index, value type: B

Example get/set:

```
1/0 P4G_TCP_DUP_THRES  [0]  3
```

P4G_TCP_SYN_RTO [gid] rto retries backoff

Configure the value of the tcp SYN retransmission timeout, max. retries and max. backoff.

gid:	integer, the sub-index value of the connection group.
rto:	integer, SYN retransmission timeout [ms] - must be larger than 0
retries:	byte, Max. SYN retransmission retries - must be larger than 0
backoff:	byte, Max. SYN retransmission backoff

Summary get/set, connection group index, value type: I, B, B

Example get/set:

```
1/0 P4G_TCP_SYN_RTO  [0]  2000  5  3
```

P4G_TCP_RTO [gid] type rto retries backoff

Configure the value of the tcp retransmission timeout, max. retries and max. backoff.

gid:	integer, the sub-index value of the connection group.
type:	coded byte, specifying RTO type

- STATIC (0) (RTO is constant as configured)
- DYNAMIC (1) (RTO is dynamic and depending on round trip time (RTT))

rto:	integer, retransmission timeout [ms] - must be larger than 0
retries:	byte, max. retransmission retries - must be larger than 0
backoff:	byte, max. retransmission backoff

Summary get/set, connection group index, value type: B, I, B, B

Example get/set:

```
1/0 P4G_TCP_RTO  [0]  STATIC  200  15  7
```

P4G_TCP_RTO_MINMAX [gid] min max

Configure the min and max value of the tcp retransmission timeout, when rto type is set to dynamic. If the calculated rto fall outside the interval, the value is clamped to the min or max value.

gid:	integer, the sub-index value of the connection group.
min:	integer, min retransmission timeout [us] - must be larger than 0 and less than max.
max:	integer, max retransmission timeout [us] - must be larger than 0 and greater than min.

Summary get/set, connection group index, value type: B, I, I, I

Example get/set:

```
1/0 P4G_TCP_RTO_MINMAX  [0] 200000 1200000
```

P4G_TCP_CONGESTION_MODE [gid] type

Configure the tcp congestion control algorithm.

gid:	integer, the sub-index value of the connection group.

type: coded byte, specifying congestion algorithm type

- NONE (0) (Disable congestion control)
- RENO (1) (Enable RENO congestion control algorithm, default value)
- NEW_RENO (2) (Enable New Reno congestion control algorithm)

Summary get/set, connection group index, value type: B

Example get/set:

```
    1/0 P4G_TCP_CONGESTION_MODE  [0]  NONE
```

UDP protocol configuration commands

P4G_UDP_PACKET_SIZE_TYPE [gid] type

Specifies the UDP Packet Size for a connection group. The Packet Size can either be fixed size identical for all connections in the connection group, incrementing or random. The individual Packet Size for a specific connection is always constant once the incrementing or random value has been created.

gid:	integer, the sub-index value of the connection group.
type:	coded byte, specifying how UDP Packet Size is set

- FIXED (0) (UDP Packet Size is constant)
- INCREMENT (1) (UDP Packet Size varies from min to max in increments of 1)
- RANDOM (2) (UDP Packet Size varies pseudo randomly between min and max)

Refer to P4G_UDP PACKET SSIZE_MINMAX command for information on how to configure min and max values.

Summary get/set, connection group index, value type: B

Example get/set:

```
    1/0 P4G_UDP_PACKET_SIZE_TYPE [0]  INCREMENT
```

P4G_UDP_PACKET_SIZE_MINMAX [gid] min max

Configure a minimum and maximum value of the range for UDP Packet Size. Both values are included in the range.
Relevant when P4G_UDP_PACKET_SIZE_TYPE is set to INCREMENT or RANDOM.

gid:	integer, the sub-index value of the connection group.
min:	integer, minimum value of UDP Packet Size
max:	integer, maximum value of UDP Packet Size

Summary get/set, connection group index, value type: I, I

Example get/set:

```
1/0 P4G_UDP_PACKET_SIZE_MINMAX  [0]  576  1400
```

P4G_UDP_PACKET_SIZE_VALUE [gid] size

Configure the fixed UDP Packet Size value.
Relevant when P4G_UDP_PACKET_SIZE_TYPE is set to FIXED.

gid:	integer, the sub-index value of the connection group.
size:	integer, fixed value of UDP Packet Size

Summary get/set, connection group index, value type: I

Example get/set:

```
1/0 P4G_UDP_PACKET_SIZE_VALUE  [0]  1023
```

TLS protocol configuration commands

P4G_TLS_ENABLE [gid] enable

Enable/Disable TLS

gid:	integer, the sub-index value of the connection group.
enable:	code byte, specifying whether to enable window scaling or not

- NO (0)
- YES (1)

Summary get/set, connection group index, value type: B

Example get/set

```
1/0 P4G_TLS_ENABLE  [0]  YES
```

P4G_TLS_CLOSE_NOTIFY [gid] enable

Enable/Disable TLS sending close notify alert on connection tear down

gid:	integer, the sub-index value of the connection group.
enable:	code byte, specifying whether to send close notify on connection tear down

- NO (0)
- YES (1)

Summary get/set, connection group index, value type: B

Example get/set

```
1/0 P4G_TLS_ENABLE  [0]  YES
```

P4G_TLS_CIPHER_SUITES [gid] ciphers

Configure the list of ciphers to announce in order of priorities.

gid:	integer, the sub-index value of the connection group.
ciphers:	double byte sequence of ciphers identified by theirs IANA number in order of priority.

Summary get/set, connection group index, value type: H*

Example get/set

```
1/0 P4G_TLS_CIPHER_SUITES  [0]  0xC012C013C014C027C028C030CCA8CCAA00040005000A
```

P4G_TLS_MAX_RECORD_SIZE [gid] size

Configure the maximum outgoing record size.

gid:	integer, the sub-index value of the connection group.
size:	integer, maximum outgoing record size in the interval]0;16384], default value 8087

Summary get/set, connection group index, value type: I

Example get/set

```
1/0 P4G_TLS_MAX_RECORD_SIZE [0] 8087
```

P4G_TLS_CERTIFICATE_FILENAME [gid] filename

Configure the certificate .

gid:	integer, the sub-index value of the connection group.
filename:	string, the filename of the certificate relative to the ftp tls folder

Summary set, connection group index, value type: S

Example set

```
1/0 P4G_TLS_CERTIFICATE_FILENAME [0] “user/certificate1.pem”
```

P4G_TLS_PRIVATE_KEY_FILENAME [gid] filename

Configure private key matching the certificate.

gid:	integer, the sub-index value of the connection group.
filename:	string, the filename of the private key relative to the ftp tls folder

Summary set, connection group index, value type: S

Example set

```
      1/0 P4G_TLS_PRIVATE_KEY_FILENAME [0] “user/private_key.pem”
```

P4G_TLS_DHPARAMS_FILENAME [gid] filename

Configure dhparams, if not set a default set will be used

gid:	integer, the sub-index value of the connection group.
filename:	string, the filename of the dhparams relative to the ftp tls folder

Summary set, connection group index, value type: S

Example set

```
      1/0 P4G_TLS_DHPARMS_FILENAME [0] “user/dhparams.pem”
```

P4G_TLS_SERVER_NAME [gid] server_name

Configure the server_name advertised by the client in the TLS SNI (Server Name Indication) extension. Both the client and server must be configured with the same server_name, as the server vill check the server_name in Client Hello message. If server_name is not configured (or configured blank), the SNI extension will not be inserted in the CLient Hello message.

gid:	integer, the sub-index value of the connection group.
server_name	string, server_name inserted in the SNI TLS extension

Summary set, connection group index, value type: S

Example set:

```
      1/0 P4G_TLS_SERVER_NAME [0] “www.xenanetworks.com”
```

P4G_TLS_PROTOCOL_VER [gid] tls_version

Configures the desired TLS protocol version. More specifically the tls_version configured is the protocol version advertised by the client in the Client Hello message, and the highest TLS protocol version accepted by the server. If the protocol_version in the Client Hello message is higher than the highest protocol version accepted by the server, the TLS Handshake will fail.

gid:	integer, the sub-index value of the connection group.
tls_version	coded byte, max. TLS protocol version

- SSLV3 (0)
- TLS10 (1)
- TLS11 (2)
- TLS12 (3)

Summary set, connection group index, value type: B

Example set:

```
1/0 P4G_TLS_PROTOCOL_VER [0] TLS11
```

P4G_TLS_MIN_REQ_PROTOCOL_VER [gid] tls_ver

Returns the minimum TLS protocol version required by the configured list of cipher suites.Each cipher suite has a minimum requires TLS protocol version that will support the cipher suite. The minimum required TLS protocol version for a list of cipher suites are the lowest minimum required TLS protocol version of all the cipher suites in the list.

gid:	integer, the sub-index value of the connection group.
tls_version	coded byte, min required TLS protocol version

- SSLV3 (0)
- TLS10 (1)
- TLS11 (2)
- TLS12 (3)

Summary get only.

Example get:

```
1/0 P4G_TLS_MIN_REQ_PROTOCOL_VER [0] ?
1/0 P4G_TLS_MIN_REQ_PROTOCOL_VER [0] TLS10
```

Packet Engine Commands

The following command are used to reserve and assign processing resources. The basic resource unit is called a Packet Engine (PE). PEs must be allocated (simple mode) or assigned (advanced mode), in order for them to service a specific port.

There are two modes of PE configuration: simple and advanced mode. The default is simple mode in which several PEs can collaborate on servicing one port, but each PE can service one and only one port. If it is required that a single PE must service more than one port, advanced mode must be configured. The system behavior defaults to simple mode. Typically there will be more PEs than ports, and therefore **simple mode will be sufficient for most users**.

When a port is not reserved, no PEs are allocated to that port. Therefore, after reserving a port, PEs must be allocated to the port. The 'available' number of PE licenses on the module will limit how many PEs that simultaneously can be allocated to reserved ports. The 'available' PEs can freely be allocated between the reserved ports, but a good default is to allocate be 2 PEs for a 10G port, and 8 PEs for a 40 G port.

The following Packet Engine commands are available:

General commands:
M4E_MODE
P4E_ALLOCATION_INFO

Simple Mode commands
P4E_AVAILABLE
P4E_ASSIGN

Advanced Mode commands
M4E_RESERVE
P4E_ALLOCATE

General commands

M4E_MODE mode
Select resource allocation mode.

mode:	coded byte	

- SIMPLE (0) (default)
- ADVANCED (1)

Summary get/set, value type: B

Example set

```
0 M4E_MODE SIMPLE
```

P4E_ALLOCATION_INFO available allocated
Display information about which PEs that are available for allocation/assignment and which are currently allocated/assigned to this port.

available:	64 bit mask of available PEs
allocated:	64 bit mask of PEs assigned to this port

Summary get only, value type: HHHHHHHH HHHHHHHH

Example get:

```
1/0 P4E_INFO 0x00000000000000F8 0x0000000000000008
```

Simple Mode commands

P4E_AVAILABLE available

Simple mode only: Report the number of PEs available for allocation.

available:	byte, total number of PEs that can be allocated to the port - including the PEs already allocated to the port.

NOTE: The number of available PE Licenses can limit the number of PEs that can be allocated to a port.

Summary get only, value type: B

Example get:

```
1/0 P4E_AVAILABLE 4
```

P4E_ALLOCATE allocate

Simple mode only: Allocate a number of PEs to this port.

allocate:	byte, the total number of PEs to allocate to this port - including the PEs already allocated to the port.

NOTE: The number of available PE Licenses can limit the number of PEs that can be allocated to a port.

Summary get/set, value type: B

Example get:

```
1/0 P4E_ALLOCATE 2
1/1 P4E_ALLOCATE 2
```

Advanced Mode commands

M4E_RESERVE mask

Advanced mode only: Reserve a number of PEs so they later can be assigned to specific ports.

mask:	hexadecimal, bitmask of PEs to reserve

Summary get/set, value type: HHHHHHHH

Example get/set:

```
0 M4_RESERVE 0x00000000000000ff
```

Reserve PEs 1 through 8.

P4E_ASSIGN mask

Advanced mode only: Assign previously reserved PEs to a port.

mask:	hex bytes, a bitmask specifying which PEs should be assigned to this port

NOTE: The number of available PE Licenses can limit the number of PEs that can be assigned to a port.

Summary get/set, value type: HHHHHHHH

Example get/set:

```
1/0 P4E_ASSIGN 0x0000000040001fff
```

Assign PEs 1 through 13 and 31 to port 0.

Traffic Commands

These are commands that start and stops traffic and reads the start of a port. Please refer to L47 Port State section for a description of the port traffic commands and states.

P4_TRAFFIC port_cmd

Gives a traffic state command to the port.

port_cmd: coded byte, the traffic state command issued to the port

● OFF	(0)	(sets port in state OFF, configuration is permitted in this state)
● ON	(1)	(begin ramp up according to the load profile)
● STOP	(2)	(stop all packet generating)
● PREPARE	(3)	(prepare the port for traffic generation)
● PRERUN	(4)	(execute prerun activities. E.g. MAC address resolution via ARP and NDP)

Summary set only, value type: B

Example set:

```
1/0 P4_TRAFFIC PREPARE
```

P4_STATE state

Display the current state

state: coded byte, specifying the current state for this port

● OFF	(0)
● PREPARE	(1)
● PREPARE_RDY	(2)
● PREPARE_FAIL	(3)
● PRERUN	(4)
● PRERUN_RDY	(5)
● RUNNING	(6)
● STOPPING	(7)
● STOPPED	(8)

Summary get only, value type: B

Example get:

```
1/0 P4_STATE ?
```

```
        1/0 P4_STATE RUNNING
```

P4_STATE_STATUS status

Returns status of the last port state change.
If port state has changed to PREPARE_FAIL, the status contains information about the reason for the fail.
Currently the status will be “OK” in all other states.

status	string, status for the last port state change

Summary get only, value type: S

Example get:

```
1/0 P4_STATE_STATUS ?
1/0 P4_STATE_STATUS “Port  0, Goup   1 - Unable to load replay file user/capture1 -
                          Parse result: PARSE_FAILED: Packets must not be truncated.
                          Packet 1 has capture length (128) != packet Length (1514)”
```

Statistics Commands

Both during and after test completion, a comprehensive number of statistics are available. Some statistics are available per port, and others per Connection Group.
There are two types of statistics:
Run-time, which can be read continuously during a test.
Post-test, which are collected and calculated when the test are stopped, and hence can only be read after test completion in *stopped* state.

The following statistics commands are available:

Port Statistics

Run-time statistics
P4_CLEAR_COUNTERS
P4_ETH_RX_COUNTERS
P4_ETH_TX_COUNTERS
P4_PORT_RX_COUNTERS
P4_PORT_TX_COUNTERS
P4_PORT_COUNTERS
P4_RX_PACKET_SIZE
P4_TX_PACKET_SIZE
P4_IPV4_RX_COUNTERS
P4_IPV4_TX_COUNTERS
P4_IPV4_COUNTERS
P4_IPV6_RX_COUNTERS
P4_IPV6_TX_COUNTERS
P4_IPV6_COUNTERS
P4_ARP_RX_COUNTERS
P4_ARP_TX_COUNTERS
P4_ARP_COUNTERS
P4_NDP_RX_COUNTERS
P4_NDP_TX_COUNTERS
P4_NDP_COUNTERS
P4_ICMP_RX_COUNTERS
P4_ICMP_TX_COUNTERS
P4_ICMP_COUNTERS
P4_TCP_RX_COUNTERS
P4_TCP_TX_COUNTERS
P4_TCP_COUNTERS
P4_UDP_RX_COUNTERS
P4_UDP_TX_COUNTERS
P4_UDP_COUNTERS

Post-test statistics
P4_RX_MTU
P4_TX_MTU

Connection Group Statistics

Run-time statistics

P4G_CLEAR_COUNTERS
P4G_TCP_STATE_CURRENT
P4G_TCP_STATE_TOTAL
P4G_TCP_STATE_RATE
P4G_TCP_RX_PAYLOAD_COUNTERS
P4G_TCP_TX_PAYLOAD_COUNTERS
P4G_TCP_RX_PACKET_COUNTERS
P4G_TCP_TX_PACKET_COUNTERS
P4G_TCP_RETRANSMIT_COUNTERS
P4G_TCP_ERROR_COUNTERS
P4G_UDP_STATE_CURRENT
P4G_UDP_STATE_TOTAL
P4G_UDP_STATE_RATE
P4G_UDP_RX_PAYLOAD_COUNTERS
P4G_UDP_TX_PAYLOAD_COUNTERS
P4G_UDP_RX_PACKET_COUNTERS
P4G_UDP_TX_PACKET_COUNTERS
P4G_TLS_ALERT_WARNING_COUNTERS
P4G_TLS_ALERT_FATAL_COUNTERS
P4G_TLS_STATE_CURRENT
P4G_TLS_STATE_TOTAL
P4G_TLS_STATE_RATE
P4G_TLS_RX_PAYLOAD_COUNTERS
P4G_TLS_TX_PAYLOAD_COUNTERS
P4G_APP_TRANSACTION_COUNTERS
P4G_USER_STATE_CURRENT
P4G_USER_STATE_TOTAL
P4G_USER_STATE_RATE

Post-test statistics

P4G_CLEAR_POST_STAT
P4G_TCP_ESTABLISH_HIST
P4G_TCP_CLOSE_HIST
P4G_TLS_HANDSHAKE_HIST
P4G_TIME_HIST_CONF
P4G_RECALC_TIME_HIST
P4G_TCP_RX_TOTAL_BYTES_HIST
P4G_TCP_RX_GOOD_BYTES_HIST
P4G_TCP_TX_TOTAL_BYTES_HIST
P4G_TCP_TX_GOOD_BYTES_HIST
P4G_UDP_RX_BYTES_HIST

P4G_UDP_TX_BYTES_HIST
P4G_TLS_RX_PAYLOAD_BYTES_HIST
P4G_TLS_TX_PAYLOAD_BYTES_HIST
P4G_PAYLOAD_HIST_CONF
P4G_RECALC_PAYLOAD_HIST
P4G_TRANSACTION_HIST
P4G_TRANSACTION_HIST_CONF
P4G_RECALC_TRANSACTION_HIST

Port Run-time statistics

P4_CLEAR_COUNTERS

Clears all run-time port counters.

Summary set only

Example:

```
1/0 P4_CLEAR_COUNTERS
```

P4_ETH_RX_COUNTERS time ref_time stats...

Return total port Ethernet receive statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
bit/s	long integer, bit/second of (layer 2) bytes received
packets/s	long, integer packets/second of received packets
bytes	long integer, total number of (layer 2) bytes received
packets	long integer, total number of packets received

Summary get only

Example, get:

```
1/0 P4_ETH_RX_COUNTERS ?
1/0 P4_ETH_RX_COUNTERS 3620000 3610000 1000000 82 9869960965 14880000
```

P4_ETH_TX_COUNTERS time ref_time stats...

Return total port Ethernet transmit statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
bit/s	long integer, bit/second of (layer 2) bytes transmitted
packets/s	long integer, packets/second of packets transmitted
bytes	long integer, total number of (layer 2) bytes transmitted
packets	long integer, total number of packets transmitted

Summary get only

Example, get:

```
1/0 P4_ETH_TX_COUNTERS ?
1/0 P4_ETH_TX_COUNTERS 3620000 3610000 1000000 82 9869960965 14880000
```

P4_ETH_COUNTERS time ref_time stats...

Return total port Ethernet statistics since last clear.

tx_errors	long integer, Tx errors
rx_errors	long integer, Rx errors
rx_missed	long integer, packets lost by the Ethernet driver due to rx queue overflow

Summary get only

Example get:

```
1/0 P4_ETH_COUNTERS ?
1/0 P4_ETH_COUNTERS 3620000 3610000 0 0 2173
```

P4_PORT_RX_COUNTERS time ref_time stats...

Return total port receive statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
vlan_pkt	long integer, total number of 802.1Q VLAN tagged packets received
bit/s	long integer, bit/second of (layer 1) bits received.
bytes	long integer, total number of (layer 1) bytes received.

Summary get only

Example, get:

```
1/0 P4_PORT_RX_COUNTERS ?
1/0 P4_PORT_RX_COUNTERS 3620000 3610000 14880000 1016260 17509143283
```

P4_PORT_TX_COUNTERS time ref_time stats...

Return total port transmit statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
vlan_pkt	long integer, total number of 802.1Q VLAN tagged packets transmitted
bit/s	long integer, bit/second of (layer 1) bits transmitted.
bytes	long integer, total number of (layer 1) bytes received.

Summary get only

Example, get:

```
1/0 P4_PORT_TX_COUNTERS ?
1/0 P4_PORT_TX_COUNTERS 3620000 3610000 14880000 1016260 17509143283
```

P4_PORT_COUNTERS time ref_time stats...

Return total port transmit statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
inval_eth	long integer, total number of invalid (e.g. short) Ethernet packets received
unknown_eth	long integer, total number of unknown or unsupported Ethernet packets received
vlan_err	long integer, total number of packets with mismatching vlan info received
pkt_rate_lim	long integer, number of times that number of packets transmitted has been limited by the maximum packet rate limiter.

Summary get only

Example, get:

```
1/0 P4_PORT_TX_COUNTERS ?
1/0 P4_PORT_TX_COUNTERS 3620000 3610000 14880000 1016260
```

P4_RX_PACKET_SIZE time ref_time bins...

Return histogram over received (layer 2) packets sizes in 100 bytes intervals.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
bins	16 x long integer, number of packets received with a (layer 2) size in the given interval.

Summary get only

Example, get:

```
1/0 P4_RX_PACKET_SIZE ?
1/0 P4_RX_PACKET_SIZE 3620000 3610000 1000 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2000
```

P4_TX_PACKET_SIZE time ref_time bins...

Return histogram over transmitted (layer 2) packets sizes in 100 bytes intervals.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
bins	16 x long integer, number of packets transmitted with a (layer 2) size in the given interval.

Summary get only

Example, get:

```
1/0 P4_RX_PACKET_SIZE ?
1/0 P4_RX_PACKET_SIZE 3620000 3610000 1000 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2000
```

P4_IPV4_RX_COUNTERS time ref_time stats...

Return total Port IPv4 protocol receive statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
ipv4_pkt	long integer, total number of IPv4 packets received

Summary get only

Example, get:

```
1/0 P4_IPV4_RX_COUNTERS ?
1/0 P4_IPV4_RX_COUNTERS 3620000 3610000 2000000
```

P4_IPV4_TX_COUNTERS time ref_time stats...

Return total Port IPv4 protocol transmit statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
ipv4_pkt	long integer, total number of IPv4 packets transmitted

Summary get only

Example, get:

```
1/0 P4_IPV4_TX_COUNTERS ?
1/0 P4_IPV4_TX_COUNTERS 3620000 3610000 2000000
```

P4_IPV4_COUNTERS time ref_time stats...

Return total Port IPv4 protocol statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
chksum_error	long integer, total number of ipv4 packets which ip header checksum error
invalid_ipv4	long integer, total number of ipv4 packets which are malformed
unknown_ipv4	long integer, total number of ipv4 packets with unknown protocol

Summary get only

Example, get:

```
1/0 P4_IPV4_COUNTERS ?
1/0 P4_IPV4_COUNTERS 3620000 3610000 0 0 0
```

P4_IPV6_RX_COUNTERS time ref_time stats...

Return total Port IPv6 protocol receive statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
ipv6_pkt	long integer, total number of IPv6 packets received

Summary get only

Example, get:

```
1/0 P4_IPV6_RX_COUNTERS ?
1/0 P4_IPV6_RX_COUNTERS 3620000 3610000 2000000
```

P4_IPV6_TX_COUNTERS time ref_time stats...

Return total Port IPv6 protocol transmit statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
ipv6_pkt	long integer, total number of IPv6 packets transmitted

Summary get only

Example, get:

```
1/0 P4_IPV6_TX_COUNTERS ?
1/0 P4_IPV6_TX_COUNTERS 3620000 3610000 2000000
```

P4_IPV6_COUNTERS time ref_time stats...

Return total Port IPv6 protocol statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
invalid_ipv6	long integer, total number of ipv6 packets which are malformed
unknown_ipv6	long integer, total number of ipv6 packets with unknown protocol

Summary get only

Example, get:

```
1/0 P4_IPV6_COUNTERS ?
1/0 P4_IPV6_COUNTERS 3620000 3610000 0 0
```

P4_ARP_RX_COUNTERS time ref_time stats...

Return total Port ARP protocol receive statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
arp_request	long integer, total number ARP Requests received
arp_reply	long integer, total number ARP Replies received

Summary get only

Example, get:

```
1/0 P4_ARP_RX_COUNTERS ?
1/0 P4_ARP_RX_COUNTERS 3620000 3610000 10000 0
```

P4_ARP_TX_COUNTERS time ref_time stats...

Return total Port ARP protocol transmit statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
arp_request	long integer, total number ARP Requests transmitted
arp_reply	long integer, total number ARP Replies transmitted

Summary get only

Example, get:

```
1/0 P4_ARP_TX_COUNTERS ?
1/0 P4_ARP_TX_COUNTERS 3620000 3610000 0 10000
```

P4_ARP_COUNTERS time ref_time stats...

Return total Port ARP protocol statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
invalid_arp	long integer, total number of invalid ARP packets received
arp_request_lookup_failure	long integer, number of ARP requests received that could not be resolved
Arp_reply_lookup_failure	long integer, number of ARP replies received that could not be resolved
arp_request_rtx	long integer, number of retransmitted ARP requests
arp_resolved	long integer, number of correct resolved IP addresses
arp_failed	long integer, number of IP address that was not resolved
arp_table_lookup_failure	long integer, number of dest IP addresses not found in the ARP table

Summary get only

Example, get:

```
1/0 P4_ARP_COUNTERS ?
1/0 P4_ARP_COUNTERS 3620000 3610000 0 0 0 234 1000 0 0
```

P4_NDP_RX_COUNTERS time ref_time stats...

Return total Port NDP protocol receive statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
ndp_request	long integer, total number NDP Requests received
ndp_reply	long integer, total number NDP Replies received

Summary get only

Example, get:

```
1/0 P4_NDP_RX_COUNTERS ?
1/0 P4_NDP_RX_COUNTERS 3620000 3610000 10000 0
```

P4_NDP_TX_COUNTERS time ref_time stats...

Return total Port NDP protocol transmit statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
ndp_request	long integer, total number NDP Requests transmitted
ndp_reply	long integer, total number NDP Replies transmitted

Summary get only

Example, get:

```
1/0 P4_NDP_TX_COUNTERS ?
1/0 P4_NDP_TX_COUNTERS 3620000 3610000 0 10000
```

P4_NDP_COUNTERS time ref_time stats...

Return total Port NDP protocol statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
invalid_ndp	long integer, total number of invalid NDP packets received
ndp_request_lookup_failure	long integer, number of NDP requests received that could not be resolved
ndp_reply_lookup_failure	long integer, number of NDP replies received that could not be resolved
ndp_request_rtx	long integer, number of retransmitted NDP requests
ndp_resolved	long integer, number of correct resolved IP addresses
ndp_failed	long integer, number of IP address that was not resolved
ndp_table_lookup_failure	long integer, number of dest IP addresses not found in the NDP table

Summary get only

Example, get:

```
1/0 P4_NDP_COUNTERS ?
1/0 P4_NDP_COUNTERS 3620000 3610000 0 0 0 234 1000 0 0
```

P4_ICMP_RX_COUNTERS time ref_time stats...

Return total Port ICMP protocol receive statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
icmp_echo_req	long integer, total number of ICMP Echo requests received
icmp_echo_rep	long integer, total number of ICMP Echo replies received
icmp_dest_unk	long integer, total number of ICMP Destination unknown received
icmp_time_exc	long integer, total number of ICMP Time exceeded received
icmpv6_pkt	long integer, total number of ICMPv6 packets received

Summary get only

Example, get:

```
1/0 P4_ICMP_RX_COUNTERS ?
1/0 P4_ICMP_RX_COUNTERS 3620000 3610000 1000 0 0 0 0
```

P4_ICMP_TX_COUNTERS time ref_time stats...

Return total Port ICMP protocol transmit statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
icmp_echo_req	long integer, total number of ICMP Echo requests transmitted
icmp_echo_rep	long integer, total number of ICMP Echo replies transmitted
icmp_dest_unk	long integer, total number of ICMP Destination unknown transmitted
icmp_time_exc	long integer, total number of ICMP Time exceeded transmitted
icmpv6_pkt	long integer, total number of ICMPv6 packets transmitted

Summary get only

Example, get:

```
1/0 P4_ICMP_TX_COUNTERS ?
1/0 P4_ICMP_TX_COUNTERS 3620000 3610000 0 1000 0 0 0 0
```

P4_ICMP_COUNTERS time ref_time stats...

Return total Port ICMP protocol statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
invalid_icmp	long integer, total number of unknown or invalid ICMP packets received
unknown_icmp	long integer, total number of unknown or unsupported ICMP packets received
invalid_icmpv6	long integer, total number of unknown or invalid ICMPv6 packets received
unknown_icmpv6	long integer, total number of unknown or unsupported ICMPv6 packets received

Summary get only

Example, get:

```
1/0 P4_ICMP_COUNTERS ?
1/0 P4_ICMP_COUNTERS 3620000 3610000 0 0 0 0
```

P4_TCP_RX_COUNTERS time ref_time stats...

Return total Port TCP protocol receive statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
tcp_pkt	long integer, total number of TCP packets received

Summary get only

Example, get:

```
1/0 P4_TCP_RX_COUNTERS ?
1/0 P4_TCP_RX_COUNTERS 3620000 3610000 2000000
```

P4_TCP_TX_COUNTERS time ref_time stats...

Return total Port TCP protocol transmit statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
tcp_pkt	long integer, total number of TCP packets transmitted

Summary get only

Example, get:

```
1/0 P4_TCP_TX_COUNTERS ?
1/0 P4_TCP_TX_COUNTERS 3620000 3610000 2000000
```

P4_TCP_COUNTERS time ref_time stats...

Return total Port TCP protocol statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
chksum_error	long integer, total number of tcp packets which tcp header checksum error
invalid_tcp	long integer, total number of TCP packets which are malformed
tcp_lookup_failure	long integer, number of TCP packets received that could not be resolved

Summary get only

Example, get:

```
1/0 P4_TCP_COUNTERS ?
1/0 P4_TCP_COUNTERS 3620000 3610000 0 0 0
```

P4_UDP_RX_COUNTERS time ref_time stats...

Return total Port UDP protocol receive statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
udp_pkt	long integer, total number of UDP packets received

Summary get only

Example, get:

```
1/0 P4_UDP_RX_COUNTERS ?
1/0 P4_UDP_RX_COUNTERS 3620000 3610000 2000000
```

P4_UDP_TX_COUNTERS time ref_time stats...

Return total Port UDP protocol transmit statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
udp_pkt	long integer, total number of UDP packets transmitted

Summary get only

Example, get:

```
1/0 P4_UDP_TX_COUNTERS ?
1/0 P4_UDP_TX_COUNTERS 3620000 3610000 2000000
```

P4_UDP_COUNTERS time ref_time stats...

Return total Port UDP protocol statistics since last clear.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
chksum_error	long integer, total number of udp packets which udp header checksum error
invalid_udp	long integer, total number of UDP packets which are malformed
udp_lookup_failure	long integer, number of UDP packets received that could not be resolved

Summary get only

Example, get:

```
1/0 P4_UDP_COUNTERS ?
1/0 P4_UDP_COUNTERS 3620000 3610000 0 0 0
```

Port Post-test statistics

P4_RX_MTU bins...

Return histogram over received (layer 3) packets sizes in 1 byte intervals. Each bin represents a packet size in the interval [576..1500] bytes.

bins	925 x byte, ‘1’ if any packets was received with the specified layer 3 size, otherwise ‘0’.

Summary get only

Example, get:

```
1/0 P4_RX_MTU ?
1/0 P4_RX_MTU 1 1 1 1 1 1 1 1 ... 1 1 0 0 0 0 0 0 0 0 0 0 0 0
```

P4_TX_MTU bins...

Return histogram over transmitted (layer 3) packets sizes in 1 byte intervals. Each bin represents a packet size in the interval [576..1500] bytes.

bins	925 x byte, ‘1’ if any packets was transmitted with the specified layer 3 size, otherwise ‘0’.

Summary get only

Example, get:

```
1/0 P4_TX_MTU ?
1/0 P4_TX_MTU 1 1 1 1 1 1 1 1 ... 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Connection Group Run-time statistics

P4G_CLEAR_COUNTERS [gid]

Clears all run-time statistics for the connection group.

gid	integer, the sub-index value of the connection group.

Summary, set only, connection group index

Example, set:

```
1/1 P4G_CLEAR_COUNTERS [0]
```

P4G_TCP_STATE_CURRENT [gid] time ref_time stats...

Returns a list of the current tcp state counters. The counters returned corresponds the the following tcp states: CLOSED , LISTEN, SYN_SENT, TCB_SYN_RCVD, ESTABLISHED, FIN_WAIT_1, FIN_WAIT_2, CLOSE_WAIT, CLOSING, LAST_ACK, TIME_WAIT

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
stats	long integer, the number of connections currently in this state

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_TCP_STATE_CURRENT [0] ?
1/0 P4G_TCP_STATE_CURRENT [0]  3620000 3610000 640 0 0 0 360 0 0 0 0 0 0
```

P4G_TCP_STATE_TOTAL [gid] time ref_time stats...

Returns a list of the total tcp state counters. The counters returned corresponds the the following tcp states: CLOSED , LISTEN, SYN_SENT, SYN_RCVD, ESTABLISHED, FIN_WAIT_1, FIN_WAIT_2, CLOSE_WAIT, CLOSING, LAST_ACK, TIME_WAIT

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
stats	long integer, the total number of connections that has entered this state

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_TCP_STATE_TOTAL [0] ?
1/0 P4G_TCP_STATE_TOTAL [0] 3620000 3610000 2000 0 2000 0 2000 2000 2000 0 0 0 0
```

P4G_TCP_STATE_RATE [gid] time ref_time stats...

Returns a list of the tcp state rates measured in connections/second. The counters returned corresponds the the following tcp state rates: CLOSED , LISTEN, SYN_SENT, TCB_SYN_RCVD, ESTABLISHED, FIN_WAIT_1, FIN_WAIT_2, CLOSE_WAIT, CLOSING, LAST_ACK, TIME_WAIT

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
ctr	long integer, the number of connections/second entering this state

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_TCP_STATE_RATE [0] ?
1/0 P4G_TCP_STATE_RATE [0] 3620000 3610000 2000 0 2000 0 2000 2000 2000 0 0 0 0
```

P4G_TCP_RX_PAYLOAD_COUNTERS [gid] time ref_time stats...

Returns a list of the tcp Rx payload counters.

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
total_bytes	long integer, number of total TCP payload bytes received
total_bytes/s	long integer, number of total TCP payload bytes/second received
good_bytes	long integer, number of good TCP payload bytes received
good_bytes/s	long integer, number of good TCP payload bytes/second received

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_TCP_RX_PAYLOAD_COUNTERS [0] ?
```

```
1/0 P4G_TCP_RX_PAYLOAD_COUNTERS [0] 3620000 3610000 262370760 11853542 262217460
11831645
```

P4G_TCP_TX_PAYLOAD_COUNTERS [gid] time ref_time stats...

Returns a list of the tcp Tx payload counters.

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
total_bytes	long integer, number of total TCP payload bytes transmitted
total_bytes/s	long integer, number of total TCP payload bytes/second transmitted
good_bytes	long integer, number of good TCP payload bytes transmitted
good_bytes/s	long integer, number of good TCP payload bytes/second transmitted

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_TCP_TX_PAYLOAD_COUNTERS [0] ?
1/0 P4G_TCP_TX_PAYLOAD_COUNTERS [0] 3620000 3610000 262435000 11860805 262230600
11831610
```

P4G_TCP_RX_PACKET_COUNTERS [gid] time ref_time stats...

Returns a list of the tcp Rx packet counters.

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
packets	long integer, number of TCP packets received
packets/s	long integer, number of TCP packets/second received

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_TCP_RX_PACKETS_COUNTERS [0] ?
1/0 P4G_TCP_RX_PACKETS_COUNTERS [0] 775862076 775770249 220382600 2399634
```

P4G_TCP_TX_PACKET_COUNTERS [gid] time ref_time stats...

Returns a list of the tcp Tx packet counters.

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
packets	long integer, number of TCP packets transmitted
packets/s	long integer, number of TCP packets/second transmitted

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_TCP_TX_PACKET_COUNTERS [0] ?
1/0 P4G_TCP_TX_PACKET_COUNTERS [0] 775857460 775770249 174419820 1999725
```

P4G_TCP_RTT_VALUE [gid] time ref_time l_rtt_sum l_rtt_no g_rtt_sum g_rtt_no

Returns values there can be used to calculate the RTT value of all connections in a connection group..

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
l_rtt_sum	long integer, accumulated RTT value (microseond) in previous 200 ms
l_no_rtt	long integer, number of RTT value accumulated in l_rtt_sum
g_rtt_sum	long integer, accumulated RTT value (microsecond) since start of test
g_rtt_no	long integer, number of RTT values accumulated in g_rtt_sum

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_TCP_RTT_VALUE [0] ?
1/0 P4G_TCP_RTT_VALUE [0] 3620000 3610000 5000 30 3000 10
```

P4G_TCP_RETRANSMIT_COUNTERS [gid] time ref_time stats...

Returns a list of tcp retransmission counters.

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
rx_dupack	long integer, number of duplicate ACK received

rx_ooseg	long integer, number of out-of-order segments received
fast_rtx	long integer, number of fast-retransmit events occurred
fast_rtx_seg	long integer, number of segments retransmitted during fast-retransmit
rto_rtx	long integer, number of timer based retransmit events occurred
syn_rtx	long integer, number of SYN retransmitted
fin_rtx	long integer, number of FIN retransmitted

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_TCP_RETRANSMIT_COUNTERS [0] ?
1/0 P4G_TCP_RETRANSMIT_COUNTERS [0] 3620000 3610000 108 0 36 144 0 0 0
```

P4G_TCP_ERROR_COUNTERS [gid] time ref_time stats...

Returns a list of TCP error counters.

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
rx_rst	long integer, number of TCP RESET received
tx_rst	long integer, number of TCP RESET transmitted
wnd_full	long integer, number of TCP window full encountered
max_syn_rtx	long integer, number of connections reset due to maximum number of SYN retransmits
max_rtx	long integer, number of connections reset due to maximum number of RTO retransmits
reset_local	long integer, number of connections reset locally by transmitting a TCP RESET
reset_peer	long integer, number of connections reset by peer
seg_not_send	long integer, number of TCP segments not send due to exhausted Tx resources
rx_zero_wnd	long integer, number of Zero Window ACKs received from the peer

Summary, get only, connection group index, value type: L*

Example get:

```
1/0 P4G_TCP_ERROR_COUNTERS [0] ?
1/0 P4G_TCP_ERROR_COUNTERS [0] 3620000 3610000 0 0 968 0 0 402 402 0 38
```

P4G_UDP_STATE_CURRENT [gid] time ref_time stats...

Returns a list of the current udp state counters. The counters returned corresponds the the following udp states:

- CLOSED The connection structure has been created, but has not been 'ramped up' yet.

- OPEN	The connection has been ‘ramped up’, and is ready to transmit or receive data
- ACTIVE.	The connection is actively transmitting data

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
stats	long integer, the number of connections currently in this state

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_UDP_STATE_CURRENT [0] ?
1/0 P4G_UDP_STATE_CURRENT [0]  3620000 3610000 320 0 680
```

P4G_UDP_STATE_TOTAL [gid] time ref_time stats...

Returns a list of the total udp state counters. The counters returned corresponds the the following udp states:

- CLOSED	The connection structure has been created, but has not been ‘ramped up’ yet.
- OPEN	The connection has been ‘ramped up’, and is ready to transmit or receive data
- ACTIVE.	The connection is actively transmitting data

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
stats	long integer, the total number of connections that has entered this state

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_UDP_STATE_TOTAL [0] ?
1/0 P4G_UDP_STATE_TOTAL [0] 3620000 3610000 1320 0 2000
```

P4G_UDP_STATE_RATE [gid] time ref_time stats...

Returns a list of the udp state rates measured in connections/second. The counters returned corresponds the the following udp state rates:

- CLOSED	The connection structure has been created, but has not been ‘ramped up’ yet.
- OPEN	The connection has been ‘ramped up’, and is ready to transmit or receive data
- ACTIVE.	The connection is actively transmitting data

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
ctr	long integer, the number of connections/second entering this state

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_UDP_STATE_RATE [0] ?
1/0 P4G_UDP_STATE_RATE [0] 3620000 3610000 0 0 100
```

P4G_UDP_RX_PAYLOAD_COUNTERS [gid] time ref_time stats...

Returns a list of the udp Rx payload counters.

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
bytes	long integer, number of UDP payload bytes received
bytes/s	long integer, number of UDP payload bytes/second received

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_UDP_RX_PAYLOAD_COUNTERS [0] ?
1/0 P4G_UDP_RX_PAYLOAD_COUNTERS [0] 3620000 3610000 262370760 11853542
```

P4G_UDP_TX_PAYLOAD_COUNTERS [gid] time ref_time stats...

Returns a list of the udp Rx payload counters.

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
bytes	long integer, number of UDP payload bytes transmitted
bytes/s	long integer, number of UDP payload bytes/second transmitted

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_UDP_TX_PAYLOAD_COUNTERS [0] ?
1/0 P4G_UDP_TX_PAYLOAD_COUNTERS [0] 3620000 3610000 262435000 11860805
```

P4G_UDP_RX_PACKET_COUNTERS [gid] time ref_time stats...

Returns a list of the udp Rx packet counters.

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
packets	long integer, number of UDP packets received
packets/s	long integer, number of UDP packets/second received

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_UDP_RX_PACKETS_COUNTERS [0] ?
1/0 P4G_UDP_RX_PACKETS_COUNTERS [0] 775862076 775770249 220382600 2399634
```

P4G_UDP_TX_PACKET_COUNTERS [gid] time ref_time stats...

Returns a list of the udp Tx packet counters.

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
packets	long integer, number of UDP packets transmitted
packets/s	long integer, number of UDP packets/second transmitted

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_UDP_TX_PACKET_COUNTERS [0] ?
1/0 P4G_UDP_TX_PACKET_COUNTERS [0] 775857460 775770249 174419820 1999725
```

P4G_TLS_STATE_CURRENT [gid] time ref_time stats...

Returns a list of the current TLS state counters. The counters returned corresponds the the following TLS states: TLS_INACTIVE, TLS_HANDSHAKING, TLS_HANDSHAKE_DONE, TLS_HANDSHAKE_FAILED, TLS_FAILED, TLS_INTERNAL_FAILED, TLS_CLOSE_NOTIFY, TLS_DONE

gid	integer, the sub-index value of the connection group.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
stats	long integer, the number of connections currently in this state

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_TLS_STATE_CURRENT [0] ?
1/0 P4G_TLS_STATE_CURRENT [0]  3610000 3620000 0 640 360 0 0 0 0 0
```

P4G_TLS_STATE_TOTAL [gid] time ref_time stats...

Returns a list of the total TLS state counters. The counters returned corresponds the the following TLS states: TLS_INACTIVE, TLS_HANDSHAKING, TLS_HANDSHAKE_DONE, TLS_HANDSHAKE_FAILED, TLS_FAILED, TLS_INTERNAL_FAILED, TLS_CLOSE_NOTIFY, TLS_DONE

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
stats	long integer, the total number of connections that has entered this state

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_TLS_STATE_TOTAL [0] ?
1/0 P4G_TLS_STATE_TOTAL [0]  3610000 3620000 0 2000 2000 0 0 0 0 2000
```

P4G_TLS_STATE_RATE [gid] time ref_time stats...

Returns a list of the TLS state rates measured in connections/second. The counters returned corresponds the the following TLS states: TLS_INACTIVE, TLS_HANDSHAKING, TLS_HANDSHAKE_DONE, TLS_HANDSHAKE_FAILED, TLS_FAILED, TLS_INTERNAL_FAILED, TLS_CLOSE_NOTIFY, TLS_DONE

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
stats	long integer, the number of connections/second entering this state

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_TLS_STATE_RATE [0] ?
1/0 P4G_TLS_STATE_RATE [0]  3610000 3620000 0 100 200 0 0 0 0 0
```

P4G_TLS_RX_PAYLOAD_COUNTERS [gid] time ref_time stats...

Returns a list of the TLS Rx payload counters.

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
rx_bytes	long integer, number of TLS payload bytes received
rx_bytes/s	long integer, number of TLS payload bytes/second received

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_TLS_RX_PAYLOAD_COUNTERS [0] ?
1/0 P4G_TLS_RX_PAYLOAD_COUNTERS [0] 3610000 3620000 262370760 11853542
```

P4G_TLS_TX_PAYLOAD_COUNTERS [gid] time ref_time stats...

Returns a list of the TLS Tx payload counters.

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
rx_bytes	long integer, number of TLS payload bytes transmitted
rx_bytes/s	long integer, number of TLS payload bytes/second transmitted

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_TLS_TX_PAYLOAD_COUNTERS [0] ?
1/0 P4G_TLS_TX_PAYLOAD_COUNTERS [0] 3610000 3620000 262370760 11853542
```

P4G_TLS_ALERT_WARNING_COUNTERS [gid] time ref_time stats...

Returns a list of TLS warning counters. The counters returned corresponds the the following TLS warnings: close_notify, unexpected_message, bad_record_mac, record_overflow, decompression_failure, handshake_failure, bad_certificate, unsupported_certificate, certificate_revoked, certificate_expired, certificate_unknown, illegal_parameter, unknown_ca, access_denied, decode_error, decrypt_error, protocol_version, insufficient_security, internal_error, user_canceled, no_renegotiation, unsupported_extension, unknown.

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
stats	long integer, the total number of warning received

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_TLS_ALERT_WARNING_COUNTERS [0] ?
1/0 P4G_TLS_ALERT_WARNING_COUNTERS [0] 3610000 3620000 500 0 0 …. 0 0
```

P4G_TLS_ALERT_FATAL_COUNTERS [gid] time ref_time stats...

Returns a list of TLS error counters. The counters returned corresponds the the following TLS warnings: close_notify, unexpected_message, bad_record_mac, record_overflow, decompression_failure, handshake_failure, bad_certificate, unsupported_certificate, certificate_revoked, certificate_expired, certificate_unknown, illegal_parameter, unknown_ca, access_denied, decode_error, decrypt_error, protocol_version, insufficient_security, internal_error, user_canceled, no_renegotiation, unsupported_extension, unknown.

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
stats	long integer, the total number of errors received

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_TLS_ALERT_FATAL_COUNTERS [0] ?
1/0 P4G_TLS_ALERT_FATAL_COUNTERS [0] 3610000 3620000 500 0 0 …. 0 0
```

P4G_APP_TRANSACTION_COUNTERS [gid] time ref_time stats...

Returns request/response transaction statistics.

gid	integer, the sub-index value of the connection group.

time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
transactions	long integer, number of completed request/response transactions
transaction/s	long integer, number of completed transactions/second

Summary, get only, connection group index, value type: L*

Example get:

```
1/0 P4G_APP_TRANSACTION_COUNTERS [0] ?
1/0 P4G_APP_TRANSACTION_COUNTERS [0] 3620000 3610000 35395765 674936
```

P4G_USER_STATE_CURRENT [gid] time ref_time stats...

Returns a list of the current user state counters. A user is identified by a Client IP address. The counters returned corresponds the the following user states:

- INIT The user has been created, but has no open connections yet
- ACTIVE The user has at least one open connection
- SUCCESS The user has successfully transmitted and received all payload
- FAILED The user has failed in transmitting or receiving all payload
- STOPPED The user has been stopped due to ramp-down
- INACTIVE All the users connection is closed, but the user has not been destroyed yet.

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
stats	long integer, the number of users currently in this state

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_USER_STATE_CURRENT [0] ?
1/0 P4G_USER_STATE_CURRENT [0]  3620000 3610000 320 680 0
```

P4G_USER_STATE_TOTAL [gid] time ref_time stats...

Returns a list of the total user state counters. A user is identified by a Client IP address. The counters returned corresponds the the following user states:

- INIT The user has been created, but has no open connections yet
- ACTIVE The user has at least one open connection
- SUCCESS The user has successfully transmitted and received all payload
- FAILED The user has failed in transmitting or receiving all payload
- STOPPED The user has been stopped due to ramp-down

- INACTIVE All the users connection is closed, but the user has not been destroyed yet.

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
stats	long integer, the total number of users that has entered this state

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_USER_STATE_TOTAL [0] ?
1/0 P4G_USER_STATE_TOTAL [0]  3620000 3610000 1000 680 0
```

P4G_USER_STATE_RATE [gid] time ref_time stats...

Returns a list of the user state rates measured in users/second. A user is identified by a Client IP address. The counters returned corresponds the the following user states:

INIT	The user has been created, but has no open connections yet
ACTIVE	The user has at least one open connection
SUCCESS	The user has successfully transmitted and received all payload
FAILED	The user has failed in transmitting or receiving all payload
STOPPED	The user has been stopped due to ramp-down
INACTIVE	All the users connection is closed, but the user has not been destroyed yet.

gid	integer, the sub-index value of the connection group.
time	long integer, the current time (mSec since module restart)
ref_time	long integer, reference time (mSec for P4_TRAFFIC on)
stats	long integer, the number of users/second entering this state

Summary, get only, connection group index, value type: L*

Example, get:

```
1/0 P4G_USER_STATE_RATE [0] ?
1/0 P4G_USER_STATE_RATE [0]  3620000 3610000 0 100 0
```

Connection Group Post-test statistics

P4G_CLEAR_POST_STAT

Clears all TCP Connection Group post-test statistics.

Summary set only

Example set:

```
1/0 P4G_CLEAR_POST_STAT [0]
```

P4G_TCP_ESTABLISH_HIST [gid] conn min max average start interval bins...

Returns a histogram over connection establish times, with start and interval values as configured by P4G_TIME_HIST_CONF.

gid	integer, the sub-index value of the connection group.
conn	integer,number of connections established.
min	long integer, minimum connection establish time in us.
max	long integer, maximum connection establish time in us.
average	long integer, average connection establish time in us.
start	long integer, start value of first histogram interval in us
interval	long integer, histogram interval size in us
bins	32 integer, number of connections with establish time within the given interval.

Summary get only.

Example get:

```
1/0 P4G_TCP_ESTABLISH_HIST [0] ?
```

P4G_TCP_CLOSE_HIST [gid] conn min max average start interval bins...

Returns a histogram over connection close times, with start and interval values as configured by P4G_TIME_HIST_CONF.

gid	integer, the sub-index value of the connection group.
conn	integer, number of connections closed.
min	long integer, minimum connection close time in us.
max	long integer, maximum connection close time in us.
average	long integer, average connection close time in us.

start	long integer, start value of first histogram interval in us
interval	long integer, histogram interval size in us
bins	32 integer, number of connections with close time within the given interval.

Summary get only.

Example get:

```
1/0 P4G_TCP_CLOSE_HIST [0] ?
```

P4G_TLS_HANDSHAKE_HIST [gid] conn min max average start interval bins...

Returns a histogram over TLS connection handshake times, with start and interval values as configured by P4G_TIME_HIST_CONF.

gid	integer, the sub-index value of the connection group.
conn	integer,number of connections established.
min	long integer, minimum connection handshake time in us.
max	long integer, maximum connection handshake time in us.
average	long integer, average connection handshake time in us.
start	long integer, start value of first histogram interval in us
interval	long integer, histogram interval size in us
bins	32 integer, number of connections with handshake time within the given interval.

Summary get only.

Example get:

```
1/0 P4G_TLS_HANDSHAKE_HIST [0] ?
```

P4G_TIME_HIST_CONF [gid] start interval

Sets the start value and the interval size for the time histograms (P4G_TCP_ESTABLISH_HIST and P4G_TCP_CLOSE_HIST).

gid	long integer, the sub-index value of the connection group.
start	long integer, start value of first histogram interval in us
interval	long integer, histogram interval size in us

Summary get/set, connection group index, value type: I, I

Example set:

```
1/0 P4G_TIME_HIST_CONF [0] 0 1000
```

P4G_RECALC_TIME_HIST

Recalculates connection time histograms (retrieved with: P4G_TCP_ESTABLISH_HIST and P4G_TCP_CLOSE_HIST). Used in case time histogram configuration has been changed (using P4G_TIME_HIST_CONF)

Summary set only

Example set:

```
1/0 P4G_RECALC_TIME_HIST [0]
```

P4G_TCP_RX_TOTAL_BYTES_HIST [gid] conn min max average start interval bins...

Returns a histogram over number of total TCP bytes received, with start and interval values as configured by P4G_PAYLOAD_HIST_CONF.

gid	integer, the sub-index value of the connection group.
conn	integer,number of connections.
min	long integer, minimum total TCP bytes received on a connection.
max	long integer, maximum total TCP bytes received on a connection.
average	long integer, average total TCP bytes received on a connection.
start	long integer, start value of first histogram interval in bytes
interval	long integer, histogram interval size in bytes
bins	32 integer, number of conn, that has received total TCP bytes within the given interval.

Summary get only.

Example get:

```
1/0 P4G_TCP_RX_TOTAL_BYTES_HIST [0] ?
```

P4G_TCP_RX_GOOD_BYTES_HIST [gid] conn min max average start interval bins...

Returns a histogram over number of good TCP bytes received, with start and interval values as configured by P4G_PAYLOAD_HIST_CONF.

gid	integer, the sub-index value of the connection group.

conn	integer,number of connections.
min	long integer, minimum good TCP bytes received on a connection.
max	long integer, maximum good TCP bytes received on a connection.
average	long integer, average good TCP bytes received on a connection.
start	long integer, start value of first histogram interval in bytes
interval	long integer, histogram interval size in bytes
bins	32 integer, number of conn, that has received good TCP bytes within the given interval.

Summary get only.

Example get:

```
1/0 P4G_TCP_RX_GOOD_BYTES_HIST [0] ?
```

P4G_TCP_TX_TOTAL_BYTES_HIST [gid] conn min max average start interval bins...

Returns a histogram over number of total TCP bytes transmitted, with start and interval values as configured by P4G_PAYLOAD_HIST_CONF.

gid	integer, the sub-index value of the connection group.
conn	integer,number of connections.
min	long integer, minimum total TCP bytes transmitted on a connection.
max	long integer, maximum total TCP bytes transmitted on a connection.
average	long integer, average total TCP bytes transmitted on a connection.
start	long integer, start value of first histogram interval in bytes
interval	long integer, histogram interval size in bytes
bins	32 integer, number of conn, that has transmitted total TCP bytes within the given interval.

Summary get only.

Example get:

```
1/0 P4G_TCP_TX_TOTAL_BYTES_HIST [0] ?
```

P4G_TCP_TX_GOOD_BYTES_HIST [gid] conn min max average start interval bins...

Returns a histogram over number of good TCP bytes transmitted, with start and interval values as configured by P4G_PAYLOAD_HIST_CONF.

gid	integer, the sub-index value of the connection group.
conn	integer,number of connections.

min	long integer, minimum good TCP bytes transmitted on a connection.
max	long integer, maximum good TCP bytes transmitted on a connection.
average	long integer, average good TCP bytes transmitted on a connection.
start	long integer, start value of first histogram interval in bytes
interval	long integer, histogram interval size in bytes
bins	32 integer, number of conn, that has transmitted good TCP bytes within the given interval.

Summary get only.

Example get:

```
1/0 P4G_TCP_TX_GOOD_BYTES_HIST [0] ?
```

P4G_UDP_RX_BYTES_HIST [gid] conn min max average start interval bins...

Returns a histogram over number of UDP bytes received, with start and interval values as configured by P4G_PAYLOAD_HIST_CONF.

gid	integer, the sub-index value of the connection group.
conn	integer,number of connections.
min	long integer, minimum UDP bytes received on a connection.
max	long integer, maximum UDP bytes received on a connection.
average	long integer, average UDP bytes received on a connection.
start	long integer, start value of first histogram interval in bytes
interval	long integer, histogram interval size in bytes
bins	32 integer, number of conn, that has received UDP bytes within the given interval.

Summary get only.

Example get:

```
1/0 P4G_UDP_RX_BYTES_HIST [0] ?
```

P4G_UDP_TX_BYTES_HIST [gid] conn min max average start interval bins...

Returns a histogram over number of UDP bytes transmitted, with start and interval values as configured by P4G_PAYLOAD_HIST_CONF.

gid	integer, the sub-index value of the connection group.
conn	integer,number of connections.
min	long integer, minimum UDP bytes transmitted on a connection.
max	long integer, maximum UDP bytes transmitted on a connection.
average	long integer, average UDP bytes transmitted on a connection.

start	long integer, start value of first histogram interval in bytes
interval	long integer, histogram interval size in bytes
bins	32 integer, number of conn, that has transmitted UDP bytes within the given interval.

Summary get only.

Example get:

```
1/0 P4G_UDP_TX_BYTES_HIST [0] ?
```

P4G_TLS_RX_PAYLOAD_BYTES_HIST [gid] conn min max average start interval bins...

Returns a histogram over number of TLS Payload bytes received, with start and interval values as configured by P4G_PAYLOAD_HIST_CONF.

gid	integer, the sub-index value of the connection group.
conn	integer,number of connections.
min	long integer, minimum TLS Payload bytes received on a connection.
max	long integer, maximum TLS Payload bytes received on a connection.
average	long integer, average TLS Payload bytes received on a connection.
start	long integer, start value of first histogram interval in bytes
interval	long integer, histogram interval size in bytes
bins	32 integer, number of conn, that has received TLS Payload bytes within the given interval.

Summary get only.

Example get:

```
1/0 P4G_TLS_RX_PAYLOAD_BYTES_HIST [0] ?
```

P4G_TLS_TX_PAYLOAD_BYTES_HIST [gid] conn min max average start interval bins...

Returns a histogram over number of TLS Payload bytes transmitted, with start and interval values as configured by P4G_PAYLOAD_HIST_CONF.

gid	integer, the sub-index value of the connection group.
conn	integer,number of connections.
min	long integer, minimum TLS Payload bytes transmitted on a connection.
max	long integer, maximum TLS Payload bytes transmitted on a connection.
average	long integer, average TLS Payload bytes transmitted on a connection.
start	long integer, start value of first histogram interval in bytes

interval	long integer, histogram interval size in bytes
bins	32 integer, number of conn, that has transmitted TLS Payload bytes within the given interval.

Summary get only.

Example get:

```
1/0 P4G_TLS_TX_PAYLOAD_BYTES_HIST [0] ?
```

P4G_PAYLOAD_HIST_CONF [gid] start interval

Sets the start value and the interval size for the payload histograms ().

gid	integer, the sub-index value of the connection group.
start	long integer, start value of first histogram interval in bytes
interval	long integer, histogram interval size in bytes

Summary get/set, connection group index, value type: I, I

Example set:

```
1/0 P4G_PAYLOAD_HIST_CONF [0] 0 1000000
```

P4G_RECALC_PAYLOAD_HIST

Recalculates connection payload histograms (retrieved with: P4G_TCP_RX_TOTAL_BYTES_HIST, P4G_TCP_RX_GOOD_BYTES_HIST, P4G_TCP_TX_TOTAL_BYTES_HIST and P4G_TCP_TX_GOOD_BYTES_HIST). Used in case payload histogram configuration has been changed (using P4G_PAYLOAD_HIST_CONF)

Summary set only

Example set:

```
1/0 P4G_RECALC_PAYLOAD_HIST [0]
```

P4G_APP_TRANSACTION_HIST [gid] conn min max average start interval bins...

Returns a histogram over completed request/response transactions per connection, with start and interval values as configured by P4G_TRANSACTION_HIST_CONF.

gid	integer, the sub-index value of the connection group.

conn	integer, number of connections.
min	long integer, minimum number of transactions per connection.
max	long integer, maximum number of transactions per connection.
average	long integer, average number of transactions per connection.
start	long integer, start value of first histogram interval
interval	long integer, histogram interval size
bins	32 integer, number of connections with number of transactions within the given interval.

Summary get only.

Example get:

```
1/0 P4G_APP_TRANSACTION_HIST [0] ?
```

P4G_TRANSACTION_HIST_CONF [gid] start interval

Sets the start value and the interval size for the transaction histogram (P4G_APP_TRANSACTION_HIST).

gid	integer, the sub-index value of the connection group.
start	long integer, start value of first histogram interval
interval	long integer, histogram interval size

Summary get/set, connection group index, value type: L, L

Example set:

```
1/0 P4G_TRANSACTION_HIST_CONF [0] 0 10
```

P4G_RECALC_TRANSACTION_HIST

Recalculates transaction histograms (retrieved with: P4G_APP_TRANSACTION_HIST). Used in case transaction histogram configuration has been changed (using P4G_TRANSACTION_HIST_CONF)

Summary set only

Example set:

```
1/0 P4G_RECALC_TRANSACTION_HIST [0]
```

Capture Commands

The L47 chassis is capable to trace received and transmitted packets on a port. Packet capture can be switched ON and OFF at any time (i.e. in all port states) with the P4_CAPTURE command.
During test captured packets are stored in memory. When the test is stopped, the captured packets are saved to a file, and they can be retrieved with the P4_CAPTURE_GET_FIRST/NEXT commands. Use M4_CAPTURE_SIZE to switch between capturing full length packets or truncate to 128 bytes.

Notes:

- Tx packets are captured by the SW before any padding are appended to the packet to fulfill the minimum Ethernet Frame Length requirements, and before FCS are calculated and appended. Hence captured packets may be shorter than 60 bytes.
- If tagged VLAN is enabled and VLAN-offload is also enabled, Tx packets are captured before VLAN tags are added, and Rx frames are captured after VLAN tags has been removed. Hence captured packets does not contain VLAN tags. If VLAN tags are desired in captured packets, VLAN-offload must be disabled.

P4_CAPTURE capture

Description
Starts and stops packet capture on this port.

Parameters

capture:	coded byte, specifying whether to capture traffic on this port

- OFF (0)
- ON (1)

Summary
get/set, value type: B

Example
set:
```
1/0 P4_CAPTURE ON
```

P4_CAPTURE_GET_FIRST index sec usec capture_len len frame

Returns the first captured frame on the port. Command is only valid when port is in state STOPPED

index:	integer, index of frame returned
sec:	integer, second value of frame capture timestamp
usec:	integer, usec value of frame capture timestamp
capture_len:	integer, length of captured portion of the frame

len:	integer, length of the frame
frame:	hexdata, the captured frame (capture_len bytes)

Summary get only, value type: I, I, I, I, I, H*

Example get:

```
1/0 P_CAPTURE_GET_FIRST ?
1/0 P_CAPTURE_GET_FIRST 0 6325709 706541 60 60 0xFFFFFFFFFFFF04F40A0002010806.....
```

P4_CAPTURE_GET_NEXT index sec usec capture_len len frame

Returns the next captured frame on the port. Command is only valid when port is in state STOPPED

index:	integer, index of frame returned
sec:	integer, second value of frame capture timestamp
usec:	integer, usec value of frame capture timestamp
capture_len:	integer, length of captured portion of the frame
len:	integer, length of the frame
frame:	hexdata, the captured frame (capture_len bytes)

Summary get only, value type: I, I, I, I, I, H*

Example get:

```
1/0 P_CAPTURE_GET_NEXT ?
1/0 P_CAPTURE_GET_NEXT 1 6325709 706551 42 42 0x04F40A00020104F40A0F00000806....
```

M4_CAPTURE_SIZE size

Specify whether to capture whole packets(large) or truncated packets. When truncated (small) is selected only the first 128 bytes of the packet are saved.

size:	coded byte, specifying whether to capture whole packets or truncated packets..

- FULL (0)
- SMALL (1) - default

Summary get/set, value type: B

Example get:

```
0 M4_CAPTURE_SIZE ?
0 M4_CAPTURE_SIZE SMALL
```