Xena Layer 2-3 Scripting API

This manual describes the Xena Scripting API for the XenaBay and XenaCompact layer 2-3 test products.

As an alternative to using the XenaManager, you can interact with the testers using a command-line interface (CLI). This also allows the tester to be controlled from a scripting environment, and be part of a larger automation environment.

Detailed Script Parameters

To see the individual parameters, please visit the following pages:

Overview

Everything that can be done using the XenaManager can also be done through scripting, and in a nearly identical fashion. (Please keep this in mind even though most of the examples use the XenaManager for illustration.)

The commands are simple lines of text exchanged between a client and the Xena chassis. An example command to the chassis could be:

0/5 PS_RATEPPS [3] 500000

This goes to module 0, port 5, and sets stream 3's rate to 500000 packets per second. The chassis responds with:

<OK>

You would query for the current value this way:

0/5 PS_RATEPPS [3] ?

And the chassis would respond in exactly the same way that you set the value yourself:

0/5 PS_RATEPPS [3] 500000

Note that this is exactly the same syntax used when saving and loading port configurations to files on the client PC. This lets you use saved configurations as a starting point for scripting, and it also allows you to modify the saved files before loading them back into the chassis.

- Making a script connection
- Port configuration files
- Test cases

Scripting Explained

The chassis of the Xena test platform can be controlled in two separate fashions: in a point-and-click interactive style using XenaManager, and in a command-line-interface style using a text-based scripting interface.

The script commands are simple lines of text exchanged between a client and the Xena chassis. An example command to the chassis could be:

0/5 PS_RATEPPS [3] 500000

This goes to module 0, port 5, and sets stream 3's rate to 500000 packets per second. The chassis responds with:

<OK>

You would query for the current value this way:

0/5 PS_RATEPPS [3] ?

And the chassis would respond in exactly the same way that you set the value yourself:

0/5 PS_RATEPPS [3] 500000

This document contains a general overview of the scripting mechanism, followed by reference sections describing each group of scriptable parameters in detail. There are a few hundred parameters in total, but only a handful is required for typical simple tasks.

To set up basic traffic patterns and obtain traffic statistics, use the port parameters (starting with P_…), the stream parameters (starting with PS_…), and the transmit/receive statistics parameters (starting with PT_… and PR_…).

At the end of the overview sections there is a complete example of how to use a collection of script commands to define and execute some simple operations on a chassis.

Connecting to the Chassis

The chassis support multiple concurrent scripting sessions, just like they support multiple concurrent interactive Manager sessions. And like Manager sessions, scripting sessions interact with the chassis in its current state; establishing a scripting session does not in itself impact the chassis state.

In order to start a scripting session, you establish a TCP/IP connection with the chassis using port 22611, on the same IP address as when using XenaManager.

You then send lines of ASCII text to the chassis, terminated by CR/LF, and receive lines of ASCII text in response. The first command should be a logon with the valid password for the chassis, or the session will be terminated.

You can use any client platform that is able to establish a TCP/IP connection and send and receive lines of text through it. Typical client platforms include Tcl, Perl, Python, Java, Excel/VBA, and Telnet. You may use client-side functionality to execute script commands conditionally or repetitively.

REMEMBER: all lines sent to the chassis must be terminated by CR/LF. You will need to include these characters explicitly if the connection library of your platform does not do so automatically.

ALSO: please make sure to read all the responses send back to you from the chassis, so that the buffer is empty when the connection is eventually closed down.

Xena also provides a simple interactive scripting client application that runs on Windows and allows you to manually type commands to the chassis and see its responses.

Figure 1 The Xena Script Client used to change a chassis description.

To keep the session alive the client must show some activity every minute or so; else the chassis will assume that the client has stopped, and there are also routers that will kill a TCP session that is inactive for more than a few minutes. The simplest way is to

send an empty line, and the chassis will also respond with an empty line. The timeout interval can be changed with the C_TIMEOUT command, so that for instance C_TIMEOUT 99999 effectively disables the timeout.

Chassis resources must as always be reserved before they can be updated, whereas you can view any chassis, module, or port parameter as soon as the session is logged on. Reserving and releasing is done through the C/M/P_RESERVATION commands.

Before reservation, a user name must be provided for the scripting session using the C_OWNER command. If the chassis has resources reserved to this username they will automatically be granted to this session.

Any line starting with a semicolon is treated as a comment and ignored by the chassis, e.g.:

;This script implements the RFC2544 suite

Commands to the chassis are not case-sensitive, and replies from the chassis are in upper-case.

Relation to XenaManager

Anything you can do with XenaManager application you can also do via scripting, and the correspondence is quite straightforward. For example, just as the Manager's PORT PROPERTIES panel has a field for viewing and changing a port's minimum inter-frame gap, the scripting interface can view and change the P_INTERFRAMEGAP parameter for doing the same. The same applies to most of the other fields of the Manager's user interface.

However, there are a few areas where the Manager has more advanced functionality which is missing in the scripting interface. This does not limit what you can do, but the way you must do it is more primitive.

- Stream rates and capping. When you specify the rate of a stream using either a percentage, layer-2 Mbps, or packets per second, the Manager calculates the equivalent rates using the other two methods. It also checks that you do not exceed the available bandwidth for the port. This is not available through scripting: you just specify the rate using your method of choice, and you must take care not to exceed the available bandwidth.
- Protocol field editing. The Manager knows the field-by-field layout of various common protocols, and allows you to inspect and edit packet data at the field level. With scripting you just specify packet data as a sequence of hexadecimal bytes.
- Filter conditions. The Manager allows you to enter filter conditions as an easy-to-read Boolean expression on the various terms. With scripting you need to encode the condition using a set of bitmasks.

- Capture protocol decoding. The Manager inspects the raw bytes of captured packets in order to identify the protocols at the header of the packet. With scripting you must decode the packet data yourself if needed.

Also, the manager will disable the user-interface whenever a particular operation is not currently allowed; for instance trying to update the configuration of a port that has not been reserved, changing a parameter for an enabled stream while traffic is on, or changing a filter term used in the condition of an enabled filter. Attempting such things in a scripting session will instead lead to error status messages.

At a more fundamental level, the Manager supports the notion of a testbed containing multiple chassis. This is not applicable through scripting, since each scripting session runs through its own connection to a single chassis, and indeed the chassis are not aware of each other. Any cross-chassis control must be handled at the scripting client environment; in particular cross-chassis statistics such as packet loss.

In contrast, the scripting environment provides wild-carding across modules and ports, which is not available through the Manager.

Command Syntax

The scripting language contains similar syntax for setting and getting values of individual parameters of the chassis resources. Some parameters, like inter-frame gap, support both set and get; others, like physical port type, support only get; and a few, like injecting errors, support only set.

You change the value of a settable parameter using:

module/port parameter [*index*,…] *value value* …

Here, *module* and *port* are the numeric indices for a particular port (for chassis-level parameters neither of these are present, and for module-level parameters only *module* is present); *parameter* is one of the names specified later in this document in the reference sections; *index* is a possible sub-index of the parameter, for instance identifying a stream; and each *value* specifies a value appropriate for the particular parameter. All indices start at zero.

Values are specified using one of the following formats:

- 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.

Some parameters allow a variable number of values of a particular type (I*, B*, H*), and these are simply written with spaces in between. For hex values (H*), multiple bytes can be specified using a single0x prefix, e.g.: 0xF700ABCD2233.

Finally, certain parameters are actually integers, but use coded names for special numeric values to enhance readability, e.g.: (0=OFF,1=ON).

You query the current value of a gettable parameter using a very similar syntax:

module/port parameter [*index*,…] ?

The chassis responds with a line using identical syntax to the change-command, containing the current values. These responses can therefore be ‘replayed’ back to the chassis to re-establish the value from a previous query. This is actually the core of the load/save mechanism of the Xena Manager, as you can see by using an ordinary text editor to inspect the local files produced by save. You can also change the content if you want to; it is not interpreted by the Xena Manager.

Note that some queries, like P_INFO ? and P_CONFIG ?, are special in that they do not refer to one particular parameter, but rather to a collection of parameters. The response is multiple lines containing the current value of each of these parameters.

Status Messages

The set/change commands themselves simply produce a reply from the chassis of:

<OK>

In case something is unacceptable to the chassis, it may return one of the following status messages:

<NOTLOGGEDON> You have not issued a C_LOGON providing the chassis password.

<NOTRESERVED> You have not issued a x_RESERVATION for the resource you want to change.

<NOTWRITABLE> The parameter is read-only.

<NOTREADABLE> The parameter is write-only.

<NOTVALID> The operation is not valid in the current chassis state, e.g. because traffic is on.

<BADMODULE> The module index value is out of bounds.

<BADPORT> The port index value is out of bounds.

<BADINDEX> A parameter sub-index value is wrong.

<BADSIZE> The size of a data value is not appropriate.

<BADVALUE> A value is not appropriate.

<FAILED> An operation failed to produce a result.

In case of a plain syntax error, misspelled parameter, or an inappropriate use of module/port/indices, the chassis will return a line pointing out the column where the error was detected, e.g.:

```
0/5 PS_RATEPPS [] 5q00
---^
#Syntax error in column 24
```

Defaults and Wild-carding

The scripting environment provides you with optional default values for the module index and port index, allowing you to change and query parameters without providing the module and port index explicitly.

Default indices are enabled and disabled using the following short commands:

module/port Set default module and port to the specified values.

port Set default port to the specified value, retaining the default module.

-/- Disable the default module and port.

– Disable the default port, retaining the default module.

module/- Set the default module, and disable the default port.

? Show the current default module and port.

When a default module and port is provided, parameters that would otherwise require explicit module and port index values can be written without them, e.g.:

```
PS_RATEPPS [3] 500
^---
#Index error in column 1
0/5
PS_RATEPPS [3] 500
<OK>
```

Replies from the chassis will also use the current default values to suppress the explicit module and port indices when possible.

The scripting environment also provides wild-carding across modules and ports. Using an asterisk as a module or port index effectively makes the chassis execute the command for each value, e.g.:

```
0/* P_INTERFRAMEGAP 30
```

This sets the inter-frame gap for every port on module 0. It will generate an individual status response for each operation, and indeed some may succeed while others fail, for instance due to lack of reservation.

Wild-cards also work for queries. This will give you the inter-frame gap for each port of module 0:

```
0/* P_INTERFRAMEGAP ?
```

Wild-cards cannot be used as default values, but the default and wild-card mechanisms can be combined, for instance to use a default module together with a wild-carded port:

```
0/-
* P_INTERFRAMEGAP 30
```

Indeed, for chassis with a single module you will typically set it as the default module and then use only port indices.

Special Scripting Commands

The scripting environment provides a few commands that do not directly interact with the chassis state, but rather support the scripting process itself.

- SYNC. This command simply produces a reply of <SYNC>, which can be helpful when parsing and delimiting the lines returned from the chassis, in particular when using multi-parameter queries. You can also do SYNC ON, which will subsequently cause an automatic SYNC after each command. SYNC OFF disables this.
- WAIT n. This command waits for the specified number of seconds, up to 60, and then produces a reply of <RESUME>. This is a simple mechanism for inserting pauses into

scripts that are contained in a file and simply sent to the chassis line-by-line. Longer waits and more sophisticated automation require client-side functionality, which must also handle the keep-alives.

- HELP ?. This command gives you an overview of the built-in help function, useful when using the scripting environment interactively, as from the Xena Scripting Client.
- HELP ”*parameter*”. Gives you a brief overview of the required indices and values for the specified parameter. You are allowed to specify only a prefix of the parameter name, which will then give you the overview for each matching parameter, e.g.: HELP ”P_” for all port-level parameters. The summary of the required values uses the abbreviations for the various types introduced in the command syntax above, e.g.:B(0=OFF,1=ON) which means a single byte value where the two relevant values can be specified using coded names.

Sample Script Session

Below is an example of using the scripting commands to define and execute a simple test. A file containing these commands can simply be uploaded to a chassis using the XenaScripting Client.

You can download the file here:

samplescript

Output from Sample Session

Below you can download a file containing the output generated by the chassis when it receives the commands shown in the previous section. A dump like this can be obtained and saved using the XenaScriptClient.

You need to do a line-by-line correlation of the two lists in order to fully understand the output. Note that there are sections of blank lines in the output corresponding to the comment lines in the input.

sampleoutput

Chassis Parameters

The chassis parameters correspond to the **Chassis Resource Properties** panel of the XenaManager, and deal with basic information and configuration of the chassis itself (rather than its modules and test ports), as well as overall control of the scripting session.
The chassis parameter names all have the form **C_** and use neither a module index nor a port index.

C_LOGON password

Explanation	You log on to the chassis by setting the value of this parameter to the correct password for the chassis. All other commands will fail if the session has not been logged on.
Summary	set only, value type: S
Parameters	*password*: string, containing the correct password value.
Example (set)	C_LOGON ”xena”

C_OWNER username

Explanation	Identify the owner of the scripting session. The name can be any short quoted string up to eight characters long. This name will be used when reserving ports prior to updating their configuration. There is no authentication of the users, and the chassis does not have any actual user accounts. Multiple concurrent connections may use the same owner name, but only one connection can have any particular resource reserved at any given time. Until an owner is specified the chassis configuration can only be read. Once specified, the session can reserve ports for that owner, and will inherit any existing reservations for that owner retained at the chassis.
Summary	set and get, value type: O
Parameters	*username*: string, containing the name of the owner of this session.
Example	C_OWNER ”HH”

C_KEEPALIVE ticks

Explanation	You can request this value from the chassis, simply to let it (as well as and any routers and proxies between you) know that the connection is still valid.
Summary	get only, value type: I
Parameters	ticks: integer, an increasing number from the chassis.
Example	C_KEEPALIVE 1234

C_TIMEOUT seconds

Explanation	The maximum number of idle seconds allowed before the connection is timed out by the tester.
Summary	get/set only, value type: I
Parameters	*seconds*: integer, the maximum idle interval, default is 130 seconds.
Example	C_TIMEOUT 999

C_RESERVATION whattodo

Explanation	You set this parameter to reserve, release, or relinquish the chassis itself. The chassis must be reserved before any of the chassis-level parameters can be changed.The owner of the session must already have been specified. Reservation will fail if any modules or ports are reserved to other users.
Summary	set or get, value type: B
Parameters	*whattodo*: coded byte, containing the operation to perform: RELEASE RESERVE RELINQUISH The reservation parameters are slightly asymmetric with respect to set/get. When querying for the current reservation state, the chassis will use these values: RELEASED RESERVED_BY_YOU RESERVED_BY_OTHER
Example	C_RESERVATION RESERVE

C_RESERVEDBY username

Explanation	Identify the user who has the chassis reserved. The empty string if the chassis is not currently reserved.
Summary	get only, value type: O
Parameters	*username*: string, containing the name of the current owner of the chassis.
Example	C_RESERVEDBY ”HH”

C_LOGOFF

Explanation	Terminates the current scripting session. Courtesy only, the chassis will also handle disconnection at the TCP/IP level
Summary	set only.
Parameters	None
Example	C_LOGOFF

C_DOWN magic whatodo

Explanation	Shuts down the chassis, and either restarts it in a clean state or leaves it powered off.
Summary	set only, value types: I,B
Parameters	*magic*: integer, must be the special value -1480937026. *whattodo*: coded byte, what to do after shutting chassis down: RESTART POWEROFF
Example	C_DOWN -1480937026 RESTART

C_CAPABILITIES integer integer …

Explanation	A series of integer values specifying various internal limits of the chassis.
Summary	get only, value types: I*
Parameters	*integer*: integer, internally defined limit values.

Example	C_CAPABILITIES 1 25 …

C_MODEL model

Explanation	Obtains the specific model of this Xena chassis.
Summary	get only, value type: S
Parameters	model: string, the Xena model designation for the chassis.
Example	C_MODEL ”C1-M2SFP+”

C_SERIALNO serialno

Explanation	Obtains the unique serial number of this particular Xena chassis.
Summary	get only, value type: I
Parameters	*serialno*: integer, the serial number of this chassis.
Example	C_SERIALNO 12345678

C_VERSIONNO server driver

Explanation	Obtains the version numbers of the software installed on the chassis.
Summary	get only, value types: I,I
Parameters	*server*: integer, the server version number. *driver*: integer, the driver version number.
Example	C_VERSIONNO 200 30

C_PORTCOUNTS portcount portcount …

Explanation	Obtains the number of ports in each module slot of the chassis, and indirectly the number of slots and modules. Note: CFP modules return the number 8 which is the maximum number of 10G ports, but the actual number of ports can be configured dynamically using the M_CFPCONFIG command.

Summary	get only, value types: B*
Parameters	*portcount*: byte, the number of ports, typically 2 or 6, or 0 for an empty slot.
Example	C_PORTCOUNTS 2

C_PORTERRORS errorcount errorcount …

Explanation	Obtains the number of errors detected across all streams on each port of each test module of the chassis. The counts are ordered in sequence with those of the module in the lowest numbered chassis slot first. Empty slots are skipped, so that a chassis with a 6-port and a 2-port test module will return eight counts regardless of which slots they are in. Note: CFP modules return eight error counts since they can be configured as up to eight 10G ports. When in 100G and 40G mode only the first one or two counts are significant. Note: FCS errors are included, which leads to double-counting for streams detecting lost packets using the test payload mechanism.
Summary	get only, value types: L*
Parameters	*errorcount*: long, the total number of errors across all streams, and including FCS errors.
Example	C_PORTERRORS 0 0 0 7 0 123

C_TRAFFIC on/off moduleA portA ……. moduleX portX

Explanation	Starts or stops the traffic on a number of ports on the chassis simultaneously. The ports are identified by pairs of integers (module port) separated by white space.
Summary	
Parameters	*on/off*: byte, determines if traffic is stopped (off) or started (on). *moduleX portX*: two integers, specifies one port on a module, which should be stopped/started. Note: From Release 57.1, if any of the specified packet sizes cannot fit into the packet generator on any of the specified ports, this command will return FAILED and not start the traffic on those ports.
Example	(starts traffic on module/port 0/0, 3/1, 3/2 and 5/4)

	C_TRAFFIC ON 0 0 3 1 3 2 5 4

C_INFO ?

Explanation	Multi-parameter query, obtaining all the non-settable parameters for the chassis.
Summary	get only.
Parameters	None
Example	C_RESERVATION RESERVED_BY_YOU C_RESERVEDBY ”HH” C_PORTCOUNTS 2 C_MODEL ”C1-M2SFP+” C_SERIALNO 12345678 C_VERSIONNO 200 30

C_ALLPORTCAPS ?

Explanation	Multi-parameter query, obtaining the port capabilities for all ports of the chassis.
Summary	get only.
Parameters	None
Example	0/0 P_CAPABILITIES 1110 10000 … 0/1 P_CAPABILITIES 1000 5000 …

C_NAME chassisname

Explanation	The name of the chassis, as it appears at various places in the Manager user-interface. The name is also used to distinguish the various chassis contained within a testbed and in files containing the configuration for an entire test-case.
Summary	set and get, value type: S
Parameters	*chassisname*: string, containing the name of the chassis.
Example	C_NAME ”Lab ABC”

C_COMMENT comment

Explanation	The description of the chassis.
Summary	set and get, value type: S
Parameters	*comment*: string, containing the description of the chassis.
Example	C_COMMENT ”This chassis belongs to the XYZ project.”

C_PASSWORD password

Explanation	The password of the chassis, which must be provided when logging on to the chassis.
Summary	set and get, value type: S
Parameters	*password*: string, containing the password for the chassis.
Example	C_PASSWORD ”SeCrEt”

C_IPADDRESS address subnetmask gateway

Explanation	The network configuration parameters of the chassis management port.
Summary	set and get, value types: A,A,A
Parameters	*address*: address, the static IP address of the chassis. *subnetmask*: address, the subnet mask of the local network segment. *gateway*: address, the gateway of the local network segment.
Example	C_IPADDRESS 192.168.1.200 255.255.255.0 192.168.1.1

C_FLASH onoff

Explanation	Make all the test port LEDs flash on and off with a 1-second interval. This is helpful if you have multiple chassis mounted side by side and you need to identify a specific one.
Summary	set and get, value type: B
Parameters	*onoff*: coded byte, whether all test port LEDs are blinking: OFF ON

Example	C_FLASH OFF

C_CONFIG ?

Explanation	Multi-parameter query, obtaining all the settable parameters for the chassis.
Summary	get only.
Parameters	None
Example	C_NAME ”Lab ABC” C_COMMENT ”This chassis belongs to the XYZ project.” C_PASSWORD ”SeCrEt” C_IPADDRESS 192.168.1.200 255.255.255.0 192.168.1.1

C_INDICES session session …

Explanation	Obtains the session indices for all current sessions on the chassis.
Summary	get only, value types: I*
Parameters	None
Example	C_INDICES 0 1 7 9 13

C_STATSESSION [ses] typ adr own ops req rsp

Explanation	Obtains information and statistics for a particular session on the chassis.
Summary	get only, session index, value types: I,A,O,L,L,L
Parameters	*ses*: integer, session index. *typ*: coded integer, which kind of session: [MANAGER | SCRIPT] *adr*: address, client IP address. own: string, user name of the session. *ops*: long, number of operations done on session. *req*: long, number of bytes received by the chassis. *rsp*: long, number of bytes sent by the chassis.
Example	C_STATSESSION [7] SCRIPT 88.99.77.66 ”HH” 100 12345 23456

C_STATS ?

Explanation	Multi-parameter query, obtaining all the chassis-level statistics.
Summary	get only.
Parameters	None
Example	C_INDICES 7 13 C_STATSESSION [7] SCRIPT 88.99.77.66 ”HH” 100 12345 23456 C_STATSESSION [13] NATIVE 88.99.77.55 ”JVN” 111 22345 33456

C_FILESTART type size time mode chk name

Explanation	Initiates upload of a file to the chassis. This parameter should be followed by a sequence og C_FILEDATA parameters to provide the file content, and finally a C_FILEFINISHto commit the new file to the chassis.
Summary	set only, value types: HHHH,HHHH,HHHH,HHHH,HHHH,S
Parameters	*type*: little-endian integer as four hex bytes, the file type, should be 1. *size*: little-endian integer as four hex bytes, the number of bytes in the file. *time*: little-endian integer as four hex bytes, the Linux date+time of the file. *mode*: little-endian integer as four hex bytes, the Linux permisions of the file. *chk*: little-endian integer as four hex bytes, the checksum of the file. *name*: string, the name and location of the file, as a full path.
Example	C_FILESTART 0x01000000 … 0xF9B1A073 ”/xbin/xenaserver”

C_FILEDATA offset databytes

Explanation	Uploads a fragment of a file to the chassis.
Summary	set only, value types: I,H*
Parameters	*offset*: integer, the position within the file. databytes: hex bytes, the data content of a section of the file.
Example	C_FILEDATA 10240 0x11FC3344……43AB

C_FILEFINISH magic

Explanation	Completes upload of a file to the chassis. After validation it will replace any existing file with the same name.
Summary	set only, value type: I
Parameters	*magic*: integer, must be the special value -1480937026.
Example	C_FILEFINISH -1480937026

Module Parameters

This page describes the layer 2-3 module level scripting parameters.
The chassis parameters correspond to the **Module Resource Properties** panel of the XenaManager, and deal with basic information about, and configuration of, the module itself.
The chassis parameter names all have the form M_**<xxx>** and require a module index before the parameter name.

M_RESERVATION whattodo

Explanation	You set this parameter to reserve, release, or relinquish a module itself (as opposed to its ports). The module must be reserved before its hardware image can be upgraded. The owner of the session must already have been specified. Reservation will fail if the chassis or any ports are reserved to other users.
Summary	set or get, value type: B
Parameters	*whattodo*: coded byte, containing the operation to perform: [RELEASE (0) | RESERVE (1) | RELINQUISH (2)] Note: The reservation parameters are slightly asymmetric with respect to set/get. When querying for the current reservation state, the chassis will use these values: [RELEASED (0) | RESERVED_BY_YOU (1) | RESERVED_BY_OTHER (2)]
Example (set)	0 M_RESERVATION RELEASE

M_RESERVEDBY username

Explanation	Identify the user who has a module reserved. Returns an empty string if the module is not currently reserved by anyone.
Summary	get only, value type: O
Parameters	username: string, containing the name of the current owner of the module.
Example	0 M_RESERVEDBY ””

M_MODEL model

Explanation	Obtains the specific Xena model of a module. model: string, the Xena model designation for the module.
Summary	get only, value type: S
Parameters	model: Module model string
Example	0 M_MODEL ”M2SFP+”

M_SERIALNO serialno

Explanation	Obtains the unique serial number of a module.
Summary	get only, value type: I
Parameters	*serialno*: integer, the serial number of this module.
Example	0 M_SERIALNO 16703

M_VERSIONNO image

Explanation	Obtains the version numbers of the hardware image installed on a module.
Summary	get only, value types: I
Parameters	*image*: integer, the hardware image version number.
Example	0 M_VERSIONNO 170

M_TIMESYNC mode

Explanation	Control how the test module time-stamp clock is running, either freely in the chassis or locked to an external system time. Running with free chassis time allows nano-second precision measurements of latencies, but only when the transmitting and receiving ports are in the same chassis. Running with locked external time enables inter-chassis latency measurements, but can introduce small time discontinuities as the test module time is adjusted.
Summary	set and get, value type: B

Parameters	*mode*: coded byte, selecting the time sync mode: [CHASSIS | EXTERNAL | MODULE]
Example	M_TIMESYNC CHASSIS

M_STATUS temperature

Explanation	Get status readings for the test module itself.
Summary	get only, I*
Parameters	*temperature*: temperature of the main hardware chip, in degrees Celsius.
Example	0 M_STATUS 45

M_CFPTYPE info

Explanation	Get information about the CFP transceiver currently inserted into the cage of a CFP test module.
Summary	get only, B
Parameters	*info*: coded byte, specifying the CFP state: • NOTCFP (this is not a CFP-based test module) • NOTPRESENT (no transceiver, the CFP cage is empty) • NOTFLEXIBLE (transceiver present, supporting a fixed speed and port-count) • FLEXIBLE (transceiver present, supporting flexible speed and port-count)
Example	0 M_CFPTYPE FLEXIBLE

M_CFPCONFIG ports speed

Explanation	The current number of ports and their speed of a CFP test module. If the CFP type is NOTFLEXIBLE then it reflects the transceiver currently in the CFP cage. If the CFP type is FLEXIBLE(or NOTPRESENT) then the configuration can be changed explicitly. The following combinations are possible: 4x10G, 8x10G, 1x40G, 2x40G, and 1x100G.
Summary	set and get, value types: B,B

Parameters	*ports*: number of ports. speed: port speed, in Gbps.
Example	0 M_CFPCONFIG 2 40

M_INFO ?

Explanation	Multi-parameter query, obtaining all the non-settable parameters for a module.
Summary	get only.
Parameters	None
Example	0 M_RESERVATION RESERVED_BY_YOU 0 M_RESERVEDBY ”HH” 0 M_MODEL ”M2SFP+” 0 M_SERIALNO 16703 0 M_VERSIONNO 170 0 M_STATUS 45 0 M_CFPTYPE NOTCFP

M_CONFIG ?

Explanation	Multi-parameter query, obtaining all the settable parameters for a module.
Summary	get only.
Parameters	None
Example	

M_UPGRADE magic imagename

Explanation	Transfers a hardware image file from the chassis to a module. This image will take effect when the chassis is powered-on the next time. The transfer takes approximately 3 minutes, but no further action is required by the client.
Summary	set only, value types: I,S
Parameters	*magic*: integer, must be the special value -1480937026. *imagename*: string, the fully qualified name of a file previously uploaded to the chassis.

Example	0 M_UPGRADE -1480937026 ”/xbin/xenaimageXE_18”

M_UPGRADEPROGRESS progress

Explanation	Provides a value indicating the current stage of an on-going hardware image upgrade operation. This is for information only; the upgrade operation runs to completion by itself. The progress values are pushed to the client without it having to request them.
Summary	pushed (get) only, value type: I • 1-100: Erase completion percentage. • 101-200: Write completion percentage + 100. • 201-300: Verify completion percentage + 200. • 0: Failure.
Parameters	*progress*: integer, the current stage within the three phases: erase, write, verify.
Example	0 M_UPGRADEPROGRESS 277

M_CLOCKPPB ppb

Explanation	Makes small adjustment to the local clock of the test module, which drives the TX rate of the test ports.
Summary	set and get, value type: I
Parameters	*ppb*: adjustment from nominal value, in parts-per-billion, positive or negative.
Example	0 M_CLOCKPPB -200000

M_SMASTATUS status

Explanation	For test modules with SMA connectors, this returns the status of the SMA input.
Summary	get, value type: B
Parameters	*status*: coded byte, specifying the status of the SMA input: • OK (valid signal is received) • NO_VALID_SIGNAL (no valid signal is received
Example	0 M_SMASTATUS OK

M_SMAINPUT smain

Explanation	For test modules with SMA connectors, selects the function of the SMA input.
Summary	set and get, value type: B
Parameters	*smain*: coded byte, specifying the function: • NOTUSED • TX2MHZ (nominal 2.048 MHz reference clock for port TX rate) • TX10MHZ (nominal 10.0 MHz reference clock for port TX rate)
Example	0 M_SMAINPUT NOTUSED

M_SMAOUTPUT smaout

Explanation	For test modules with SMA connectors, selects the function of the SMA output.
Summary	set and get, value type: B
Parameters	*smaout*: coded byte, specifying the function: • DISABLED • PASSTHROUGH (replica of the SMA input signal) • P0SOF (start-of-frame pulse for port 0 TX) • P1SOF (start-of-frame pulse for port 1 TX) • REF2MHZ (nominal 2.048 MHz reference clock from TX port rate) • REF10MHZ (nominal 10.0 MHz reference clock from TX port rate) • REF156MHZ (nominal 156.25 MHz reference clock from TX port rate) • P0RXCLK (recovered clock from port 0 RX) P1RXCLK (recovered clock from port 1 RX)
Example	0 M_SMAOUTPUT DISABLED

M_TXCLOCKSOURCE txclk

Explanation	For test modules with advanced timing features, select what drives the port TX rates.
Summary	set and get, value type: B
Parameters	*txclk*: coded byte, specifying the function: • MODULELOCALCLOCK (default, the local oscillator of the test module)

	• SMAINPUT (clock derived from the reference on the SMA input) • P0RXCLK (SyncE, clock derived from RX clock of port 0) • P1RXCLK (SyncE, clock derived from RX clock of port 1)
Example	0 M_TXCLOCKSOURCE MODULELOCALCLOCK

M_TXCLOCKFILTER filter

Explanation	For test modules with advanced timing features, the loop bandwidth on the TX clock filter.
Summary	set and get, value type: B
Parameters	*filter*: coded byte, specifying the bandwidth: • BW103HZ (bandwidth of 103 Hz) • BW207HZ (bandwidth of 207 Hz) • BW416HZ (bandwidth of 416 Hz) • BW1683HZ (bandwidth of 1683 Hz) • BW7019HZ (bandwidth of 7019 Hz)
Example	0 M_TXCLOCKFILTER BW114HZ

M_TXCLOCKSTATUS status

Explanation	For test modules with advanced timing features, check whether a valid clock is present.
Summary	get only, value type: B
Parameters	*status*: coded byte, specifying the function: • OK • NOVALIDTXCLK (no valid clock, can be missing SMA input or no clock recovery from test port)
Example	0 M_TXCLOCKSTATUS OK

M_MEDIA media

Explanation	For the M1CFP4QSFP28CXP test module where one of several front ports (media types) can be selected as the input/output, this command sets the desired media type (front port).

Summary	get/set, value type: B
Parameters	*media*: coded byte, specifying the active front port: • CFP4 • QSFP28 • CXP
Example	0 M_MEDIA CFP4

Port Parameters

This page describes the layer 2-3 port level scripting parameters.
The chassis parameters correspond to the **Port Resource Properties** panel of the XenaManager, and deal with basic information and configuration of the test ports.
The port parameter 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 cannot be changed while traffic is on. Additionally, every stream must be disabled before changing parameters that affect the bandwidth of the port.

P_RESERVATION whattodo

Explanation	You set this parameter to reserve, release, or relinquish a port. The port must be reserved before any of its configuration can be changed, including streams, filters, capture, and datasets.The owner of the session must already have been specified. Reservation will fail if the chassis or module is reserved to other users.
Summary	set or get, value type: B
Parameters	*whattodo*: coded byte, containing the operation to perform: RELEASE RESERVE RELINQUISH The reservation parameters are slightly asymmetric with respect to set/get. When querying for the current reservation state, the chassis will use these values: RELEASED RESERVED_BY_YOU RESERVED_BY_OTHER
Example, set:	0/5 P_RESERVATION RESERVE Example, get: 0/5 P_RESERVATION RESERVED_BY_YOU

P_RESERVEDBY username

Explanation	Identify the user who has a port reserved. The empty string if the port is not currently reserved. Note that multiple connections can specify the same name with C_OWNER, but a resource can only be reserved to one connection. Therefore you cannot count on having the port just because it is reserved in your name. The port is reserved to this connection only if P_RESERVATIONreturns RESERVED_BY_YOU.
Summary	get only, value type: O
Parameters	*username*: string, containing the name of the current owner of the port.

Example, get:	0/5 P_RESERVEDBY ”HH”

P_RESET

Explanation	Reset port-level parameters to standard values, and delete all streams, filters, capture, and dataset definitions.
Summary	set only.
Parameters	None
Example, set:	0/1 P_RESET

P_CAPABILITIES integer integer …

Explanation	A series of integer values specifying various internal limits of a port. integer: integer, internally defined limit values.
Summary	get only, value types: I*
Parameters	*capability*: integer value
Example, get:	0/1 P_CAPABILITIES 1000 …

P_INTERFACE interface

Explanation	Obtains the physical interface type of a port. model: string, the name of the port’s physical interface.
Summary	get only, value type: S
Parameters	*interface*: string value describing the interface
Example, get:	0/5 P_INTERFACE ”SFP+ LR”

P_STATUS opticalpower

Explanation	Get status readings for the port itself.
Summary	get only, I*
Parameters	*opticalpower*: integer, received signal level for optical ports, in nanowatts, -1 when not available.
Example, get:	0/5 P_STATUS -1

P_INFO ?

Explanation	Multi-parameter query, obtaining all the non-settable parameters for a port. These parameters should not be included if the port configuration is saved and reloaded at a later time.
Summary	get only.
Parameters	None
Example, get:	0/5 P_RESERVATION RESERVED_BY_OTHERS 0/5 P_RESERVEDBY ”XX” 0/5 P_INTERFACE ”SFP+ LR” 0/5 P_SPEED 100 0/5 P_TRAFFIC ON 0/5 P_CAPTURE OFF

P_SPEEDSELECTION selection

Explanation	The speed mode for a port with an interface type supporting multiple speeds. Note: this is only a settable parameter for tri-speed ports. For CFP ports use the M_CFPCONFIG command at the module level.
Summary	set and get, value type: B
Parameters	*selection*: coded byte, containing the speed selection for the port: • AUTO (auto-negotiate) • F10M (10 Mbps) • F100M (100 Mbps) • F1G (1000 Mbps) • F10G (10000 Mbps) • F40G (40000 Mbps) • F100G (100000 Mbps) • F10MHDX (10 Mbps half duplex)

	• F100MHDX (100 Mbps half duplex) • F10M100M (10/100 Mbps) • F100M1G (100/1000 Mbps)
Example, set or get:	• 0/0 P_SPEEDSELECTION F100M •

P_MDIXMODE selection

Explanation	Selects the MDI/MDIX behaviour of copper interfaces (Currently supported on M6SFP and M2SFPT).
Summary	set and get, value type: B
Parameters	*selection*: • AUTO (auto-detect) • MDI (straight – default for host NICs when Auto MDI-X is not supported) • MDIX (crossed – default for switch/router interfaces when Auto MDI-X is not supported)
Example, set or get:	0/0 P_MDIXMODE MDI

P_SPEED mbps

Explanation	Obtains the current physical speed for a port's interface.
Summary	get only, value type: I
Parameters	*mbps*: integer, current speed in units of Mbps.
Example, get:	0/0 P_SPEED 100

P_AUTONEGSELECTION onoff

Explanation	Whether the port responds to incoming auto-negotiation requests. Only applicable to optical ports, which are fixed speed anyway.
Summary	set and get, value type: B
Parameters	*onoff*: coded byte, whether the port replies to auto-neg requests.OFF ON

Example, set or get:	0/0 P_AUTONEGSELECTION OFF

P_RECEIVESYNC syncstatus

Explanation	Obtains the current in-sync status for a port's receive interface.
Summary	get only, value type: B
Parameters	*syncstatus*: coded byte, current receive sync status: • NO_SYNC • IN_SYNC
Example, get:	0/5 P_RECEIVESYNC IN_SYNC

P_ERRORS errorcount

Explanation	Obtains the total number of errors detected across all streams on the port, including lost packets, misorder events, and payload errors. Note: FCS errors are included, which will typically lead to double-counting of lost packets.
Summary	get only, value types: L*
Parameters	*errorcount*: long, the total number of errors across all streams, and including FCS errors.
Example, get:	0/5 P_ERRORS 7

P_COMMENT comment

Explanation	The description of a port.
Summary	set and get, value type: S
Parameters	*comment*: string, containing the description of the port.
Example, set or get:	0/5 P_COMMENT "This port generates IPTV background traffic."

P_SPEEDREDUCTION ppm

Explanation	A speed-reduction applied to the transmit-side of a port, resulting in an effective traffic rate that is slightly lower than the rate of the physical interface. Speed reduction is effectuated by inserting short idle periods in the generated traffic pattern to consume part of the port's physical bandwidth. The port's clock-speed is not altered.
Summary	set and get, value type: I
Parameters	*ppm*: integer, specifying the speed reduction in units of parts-per-million.
Example, set or get:	0/3 P_SPEEDREDUCTION 100

P_INTERFRAMEGAP minbytes

Explanation	The mimimum gap between packets in the traffic generated for a port. The gap includes the Ethernet preamble.
Summary	set and get, value type: I
Parameters	*minbytes*: integer, specifying the minimum number of byte-times between generated packets.
Example, set or get:	0/3 P_INTERFRAMEGAP 20

P_MACADDRESS hexdata

Explanation	A 48-bit Ethernet MAC address specified for a port. This address is used as the default source MAC field in the header of generated traffic for the port, and is also used for support of the ARP protocol.
Summary	set and get, value type: HHHHHH
Parameters	*hexdata*: hex bytes, specifying the six bytes of the MAC address.
Example, set or get:	0/3 P_MACADDRESS 0x001122AABBCC

P_IPADDRESS address subnet gateway wild

Explanation	An IPv4 network configuration specified for a port.The address is used as the default source address field in the IP header of generated traffic, and the configuration is also used for support of the ARP and PING protocols.
Summary	set and get, value types: A,A,A,A
Parameters	*address*: address, the IP address of the port. *subnet*: address, the subnet mask of the local network segment for the port. *gateway*: address, the gateway of the local network segment for the port. *wild*: address, wildcards used for ARP and PING replies, must be 255 or 0.
Example, set or get:	0/3 P_IPADDRESS 10.0.0.123 255.255.255.0 10.0.0.1 0.0.0.255

P_ARPREPLY onoff

Explanation	Whether the port generates replies using the Address Resolution Protocol. The port can reply to incoming ARP requests by mapping the IP address specified for the port to the MAC address specified for the port. ARP reply generation is independent of whether traffic and capture is on for the port.
Summary	set and get, value type: B
Parameters	*onoff*: coded byte, whether the port replies to ARP requests: [OFF | ON]
Example, set or get:	0/0 P_ARPREPLY ON

P_PINGREPLY onoff

Explanation	Whether the port generates ping replies using the ICMP protocol. The port can reply to incoming ping requests to the IP address specified for the port. Ping reply generation is independent of whether traffic and capture is on for the port.
Summary	set and get, value type: B
Parameters	*onoff*: coded byte, whether the port replies to ping requests: [OFF | ON]
Example, set or get:	0/0 P_PINGREPLY OFF

P_ARPV6REPLY onoff

Explanation	Whether the port generates replies using the IPv6 Network Discovery Protocol.The port can reply to incoming NDP requests by mapping the IPv6 address specified for the port to the MAC address specified for the port. NDP reply generation is independent of whether traffic and capture is on for the port.
Summary	set and get, value type: B
Parameters	*onoff*: coded byte, whether the port replies to NDP requests. [OFF | ON]
Example, set or get:	0/0 P_ARPV6REPLY ON

P_PINGV6REPLY onoff

Explanation	Whether the port generates ping replies using the ICMP protocol received over IPv6. The port can reply to incoming ping requests to the IPv6 address specified for the port. Ping reply generation is independent of whether traffic and capture is on for the port.
Summary	set and get, value type: B
Parameters	*onoff*: coded byte, whether the port replies to ping requests: [OFF | ON]
Example, set or get:	0/0 P_PINGV6REPLY OFF

P_PAUSE onoff

Whether a port responds to incoming Ethernet PAUSE frames, by holding back outgoing traffic.

Summary	set and get, value type: B
Parameters	*onoff*: coded byte, whether PAUSE response is enabled: [OFF | ON]
Example, set or get:	0/0 P_PAUSE OFF

P_FLASH onoff

Explanation	Make the test port LED for a particular port flash on and off with a 1-second interval. This is helpful when you need to identify a specific port within a

	chassis.
Summary	set and get, value type: B
Parameters	*onoff*: coded byte, whether the test port LED is blinking: [OFF | ON]
Example, set or get:	0/0 P_FLASH ON

P_RANDOMSEED value

Explanation	A fixed seed value specified for a port. This value is used for a pseudo-random number generator used when generating traffic that requires random variation in packet length, payload, or modified fields.As long as no part of the port configuration is changed, the generated traffic patterns are reproducible when restarting traffic for the port. A specified seed value of -1 instead creates variation by using a new time-based seed value each time traffic generation is restarted.
Summary	set and get, value type: I
Parameters	*value*: integer, specifying a fixed seed value for the pseudo-random number generator. -1, new random sequence for each start.
Example, set or get:	0/3 P_RANDOMSEED 12345

P_AUTOTRAIN interval

Explanation	The interval between sending out training packets, allowing a switch to learn the port's MAC address. Layer-2 switches configure themselves automatically by detecting the source MAC addresses of packets received on each port. If a port only receives, and does not itself transmit test traffic, then the switch will never learn its MAC address. Also, if transmission is very rare the switch will age-out the learned MAC address. By setting the auto-train interval you instruct the port to send switch training packets, independent of whether the port is transmitting test traffic.
Summary	set and get, value type: I
Parameters	*interval*: integer, specifying the number of seconds between training packets. 0, disable training packets.

Example, set or get:	0/3 P_AUTOTAIN 60

P_LATENCYMODE mode

Explanation	Latency is measured by inserting a time-stamp in each packet when it is transmitted, and relating it to the time when the packet is received. There are three separate modes for calculating the latency: • Last-bit-out to last-bit-in, which measures basic bit-transit time, independent of packet length. • First-bit-out to last-bit-in, which adds the time taken to transmit the packet itself. • Last-bit-out to first-bit-in, which subtracts the time taken to transmit the packet itself. The same latency mode must be configured for the transmitting port and the receiving port; otherwise invalid measurements will result.
Summary	set and get, value type: B
Parameters	*mode*: coded byte, which calculation mode to use: • LAST2LAST • FIRST2LAST • LAST2FIRST
Example, set or get:	0/3 P_LATENCYMODE LAST2LAST

P_LATENCYOFFSET value

Explanation	An offset applied to the latency measurements performed for received traffic containing test payloads. This value affects the minimum, average, and maximum latency values obtained through the PR_TPLDLATENCY parameter.
Summary	set and get, value type: I
Parameters	*value*: integer, specifying the offset for the latency measurements.
Example, set or get:	0/3 P_LATENCYOFFSET 1238

P_MAXHEADERLENGTH value

Possible values: 128 (default), 256, 512, 1024

Explanation	The maximum number of header content bytes that can be freely specified for each generated stream. The remaining payload bytes of the packet are auto-generated.The default is 128 bytes. When a larger number is select there is a corresponding proportional reduction in the number of stream definitions that are available for the port.
Summary	set and get, value type: I
Parameters	*value*: integer, specifying the maximum number of header bytes.
Example, set or get:	0/3 P_MAXHEADERLENGTH 256

P_LOOPBACK loopmode

- NONE (normal non-looped operation)
- L1RX2TX (transmit byte-by-byte copy of the incoming packet) [1] [3]
- L2RX2TX (swap source and destination MAC addresses) [1] [3]
- TXON2RX (packet is also transmitted from the port) [2]
- TXOFF2RX (port's transmitter is idle) [2]

Explanation	The loop-back mode for a port. Ports can be configured to perform two different kinds of loop-back: • External RX-to-TX loop-back, where the received packets are re-transmitted immediately. The packets are still processed by the receive logic, and can be captured and analysed. • Internal TX-to-RX loop-back, where the transmitted packets are received directly by the port itself. This is mainly useful for testing the generated traffic patterns before actual use. [1] This mode is currently not available on 40G/100G ports. [2] TX2RX loop modes are done before the physical layer. Link sync, 40G/100G CAUI statistics and EEE statistics still reflect the state of the physical link. [3] In L2/L3 RX-to-TX loop mode, a port will only loop Ethernet frames if the destination MAC address (DMAC) corresponds with the MAC address of that particular port.
Summary	set and get, value type: B
Parameters	*loopmode*: coded byte, containing the loop-back mode for the port:
Example, set	0/5 P_LOOPBACK L2RX2TX

or get:	

P_CHECKSUM offset

Explanation	Controls an extra payload integrity checksum, which also covers the header protocols following the Ethernet header. It will therefore catch any modifications to the protocol fields (which should therefore not have modifiers on them).
Summary	set and get, value type: B
Parameters	*offset*: The offset in the packet where the calculation of the extra checksum is started from. Set to 0 to disable. Valid enable range is [8 .. 127].
Example, set or get:	0/0 P_CHECKSUM 8

P_GAPMONITOR start stop

Explanation	The gap-start and gap-stop criteria for the port's gap monitor.The gap monitor expects a steady stream of incoming packets, and detects larger-than-allowed gaps between them. Once a gap event is encountered it requires a certain number of consequtive packets below the threshold to end the event.
Summary	set and get, value type: I,I
Parameters	*start*: integer, the maximum allowed gap between packets, in microseconds. (0 – 13400 us) 0 = disable gap monitor. *stop*:integer, the minimum number of good packets required. (0 – 1000 packets) 0 = disable gap monitor.
Example, set or get:	0/3 P_GAPMONITOR 60000 5

P_AUTONEGSELECTION onoff

Explanation	Select whether an optical port should respond to incoming auto-neg requests.
Summary	set and get, value type: B
Parameters	*onoff*: coded byte, whether the extra checksum is included.OFF ON

Example, set or get:	0/0 P_AUTONEGSELECTION OFF

P_MULTICAST ipaddress operation seconds

Explanation	A multi-cast mode for a port. Ports can use the IGMP protocol to join or leave multi-cast groups, either on a one-off basis or repeatedly.
Summary	set and get, value type: A,B,B
Parameters	*ipaddress*: a multi-cast group address to join or leave *operation*: coded byte, specifying the operation: • OFF (stop repeated joins) • ON (start joining group repeatedly) • JOIN (do a single join operation) • LEAVE (do a single leave operation) *seconds*: the interval between repeated joins
Example, set or get:	0/5 P_MULTICAST 239.1.2.3 ON 30

P_TXMODE txmode

Explanation	The scheduling mode for outgoing traffic from the port, specifying how multiple logical streams are merged onto one physical port. There are three primary modes:Normal interleaved: The streams are treated independently, and are merged into a combined traffic pattern for the port, which honors each stream's ideal packet placements as well as possible. This is the default mode. Strict Uniform: This is a slight variation of normal interleaved scheduling, which emphasizes strict uniformity of the inter-packet-gaps as more important than hitting the stream rates absolutely precisely. Sequential: Each stream in turn contribute one or more packets, before continuing to the next stream, in a cyclical pattern. The count of packets for each stream is obtained from the PS_PACKETLIMIT parameter value for the stream. The individual rates for each stream are ignored, and instead the overall rate is determined at the port-level using the P_RATExxx parameters. This in turn determines the rates for each stream, taking into account their packet lengths and counts. The maximum number of packets in a cycle (i.e. the sum of PS_PACKETLIMIT for all enabled streams) is 500. If the packet number is larger than 500, <NOTVALID> will be returned when attempting to start the traffic (P_TRAFFIC ON).

Summary	set and get, value type: B
Parameters	*txmode*: coded byte, containing the loop-back mode for the port: • NORMAL (interleaved packet scheduling) • STRICTUNIFORM (strict uniform mode) • SEQUENTIAL (sequential packet scheduling)
Example, set or get:	0/5 P_TXMODE NORMAL

P_RATEFRACTION fraction

Explanation	The port-level rate of the traffic transmitted for a port in sequential tx mode, expressed in millionths of the effective rate for the port. The bandwidth consumption includes the inter-frame gaps, and does not depend on the length of the packets for the streams.
Summary	set and get, value type: I
Parameters	*fraction*: integer, port rate expressed as a value between 0..1000000.
Example, set or get:	0/1 P_RATEFRACTION 500000

P_RATEPPS pps

Explanation	The port-level rate of the traffic transmitted for a port in sequential tx mode, expressed in packets per second. The bandwidth consumption is heavily dependent on the length of the packets generated for the streams, and also on the inter-frame gap for the port.
Summary	set and get, value type: I
Parameters	*pps*: integer, port rate expressed as packets per second.
Example, set or get:	0/1 P_RATEPPS 300000

P_RATEL2BPS bps

Explanation	The port-level rate of the traffic transmitted for a port in sequential tx mode,

	expressed in units of bits per-second at layer-2, thus including the Ethernet header but excluding the inter-frame gap. The bandwidth consumption is somewhat dependent on the length of the packets generated for the stream, and also on the inter-frame gap for the port.
Summary	set and get, value type: L
Parameters	*bps*: long integer, port rate expressed as bits-per-second.
Example, set or get:	0/1 P_RATEL2BPS 800000000

P_RATE ?

Explanation	For a port in sequential tx mode, query the port-level rate of the traffic transmitted in the manner it was last expressed. The response is one of P_RATEFRACTION, P_RATEPPS, or P_RATEL2BPS.
Summary	get only.
Parameters	None
Example, get:	0/1 P_RATE ?

P_TXENABLE onoff

Explanation	Whether a port should enable its transmitter, or keep the outgoing link down.
Summary	set and get, value type: B
Parameters	*onoff*: coded byte, whether the transmitter is enabled. • OFF(0) • ON(1)
Example, set or get:	0/0 P_TXENABLE ON

P_TRAFFIC onoff

Explanation	Whether a port is transmitting packets. When on, the port generates a sequence of

	packets with contributions from each stream that is enabled. The streams are configured using the PS_xxx parameters. NOTE: From Release 57.1, if any of the specified packet sizes cannot fit into the packet generator, this command will return FAILED and not start the traffic.While traffic is on the streams for this port cannot be enabled or disabled, and the configuration of those streams that are enabled cannot be changed.
Summary	set and get, value type: B
Parameters	*onoff*: coded byte, whether traffic generation is active for this port:OFF ON
Example, set or get:	0/1 P_TRAFFIC ON

P_TXDELAY delayval

Explanation	Sets a variable delay from a traffic start command is received by the port until it starts transmitting. The delay is specified in multiples of 64 microseconds. Valid values are 0-31250 (0 to 2.000.000 microseconds).
Summary	set and get, value type: I
Parameters	*delayval*: integer, TX delay in multiples of 64 microseconds. (TX delay = delayval * 64 microseconds).
Example, set or get:	0/1 P_TXDELAY 31250

P_FAULTSIGNALING setting

Explanation	Sets the Remote/Local fault signaling behavior of the port (performed by the Reconciliation Sub-layer). By default, the port acts according to the standard, i.e. when receiving a bad signal, it transmits “Remote Fault indications” on the output and when receiving a “Remote Fault indication” from the far-side it will transmit IDLE sequences.
Summary	set and get, value type: B
Parameters	*setting*: coded byte:Setting can have four different values: • Normal (0): The port acts according to the standard as described above. • Force_Local (1): Port will continuously transmit “Local Fault indication” on the TX output (which is usually not allowed by the standard). • Force_Remote (2): Port will continuously transmit “Remote Fault indication” on the

	TX output. • Disabled (3) : Port will relay the traffic from the TX core regardless of what it receives on the input.
Example, set or get:	0/1 P_FAULTSIGNALING FORCE_REMOTE Note: Currently only available on M1CFP100, M2CFP40, M2QSFP+ and M1CFP4QSFP28CXP.

P_FAULTSTATUS status

Explanation	Shows if a local or remote fault is currently being detected by the Reconciliation Sub-layer of the port. **Note:** Currently only available on M1CFP100, M2CFP40, M2QSFP+ and M1CFP4QSFP28CXP.
Summary	get, value type: B
Parameters	*status*: coded byte: • OK (0): The port is receiving a valid Ethernet signal. • Local_FAULT (1): Port is receiving a bad signal (or no signal at all). • REMOTE_FAULT (2): Port is receiving a "Remote Fault indication" from its peer port.
Example, get:	0/1 P_FAULTSTATUS LOCAL_FAULT

P_TPLDMODE setting

Explanation	Sets the size of the Xena Test Payload (TPLD) used to track streams, perform latency measurements etc. Default is "Normal", which is a 20 byte TPLD. "Micro" is a condensed version, which is useful when generating very small packets with relatively long headers (like IPv6). It has the following characteristics compared to the "normal" TPLD. When the TPLDMODE is changed, it will affect ALL streams on the port. 1. Only 6 byte long. 2. Less accurate mechanism to separate Xena generated packets from other packet is the network – it is recommended not to have too much other traffic going into the receive Xena port, when micro TPLD is used. 3. No sequence checking (packet loss or packet misordering). The number of received packets for each stream can still be compared to the number of transmitted packets to detect packet loss once traffic has been stopped.

	Note: Currently not available on M6SFP, M2SFPT, M6RJ45+/M2RJ45+, M2CFP40, M1CFP100, M2SFP+4SFP
Summary	set and get, value type: B
Parameters	*setting*: coded byte: • Normal (0): Default 20B TPLD. • Micro (1): 6B micro TPLD
Example, set or get:	0/1 P_TPLDMODE MICRO

P_CAPTURE onoff

Explanation	Whether a port is capturing packets. When on, the port retains the received packets and makes them available for inspection. The capture criterias are configured using the PC_xxxparameters. While capture is on the capture parameters cannot be changed.
Summary	set and get, value type: B
Parameters	*onoff*: coded byte, whether capture is active for this port: OFF ON
Example, set or get:	0/1 P_CAPTURE OFF

P_XMITONE hexdata

Explanation	Transmits a single packet from a port, independent of the stream definitions, and independent of whether traffic is on. A valid Frame Check Sum is written into the final four bytes.
Summary	set only, value types: H*
Parameters	*hexdata*: hex bytes, the data content of the packet to be transmitted.
Example, set:	P_XMITONE 0x554433……48EE

P_LPENABLE onoff

Explanation	Enables/disables Energy Efficient Ethernet (EEE) on the port*.

Summary	set and get, value type: B
Parameters	*onoff*: coded byte, whether the EEE feature is activated or not. OFF (0), ON (1)
Example, set or get:	0/0 P_LPENABLE ON*EEE functionality is currently only supported on M2RJ45+ and M6RJ45+ modules.

P_LPTXMODE onoff

Explanation	Enables/disables the transmission of Low Power Idles (LPIs) on the port. When enabled, the transmit side of the port will automatically enter low-power mode (and leave) low-power mode in periods of low or no traffic. LPIs will **only** be transmitted if the Link Partner (receiving port) has advertised EEE capability for the selected port speed during EEE auto-negotiation.
Summary	set and get, value type: B
Parameters	*onoff*: coded byte, whether low power idles will be transmitted or not. OFF (0) ON (1)
Example, set or get:	0/0 P_LPTXMODE ON

P_LPSTATUS txh rxh txc rxc linkup

Explanation	Displays the EEE status as reported by the PHY.
Summary	set and get, value type: B, B, B, B, B
Parameters	*txh*: shows if there has been any recent change in the EEE state on the transmission side (either going into low power mode or leaving low power mode. — (0) no activity X (1) activityrxh: shows recent changes on the receive side (see “txh”). *txc*: shows the current EEE state of the transmitter (in low power or active) TX_ACTIVE (0) – actively transmitting TX_LPI (1) – In low power mode *rxc*: shows the current EEE state of the receiver (in low power or active) RX_ACTIVE (0) – actively receiving RX_LPI (1) – In low power mode *linkup*: shows if the link is up (seen from perspective of the the PHY’s PCS). LINK_DOWN (0) LINK_UP (1)
Example, get:	0/0 P_LPSTATUS TXH:– RXH:X TX_LPI RX_LPI LINK_UP

P_LPPARTNERAUTONEG speed0 …. speed 5

Explanation	Displays the EEE capabilities advertised during autonegotiation by the far side (link partner).0 = – , 1 = 100BASE-TX 0 = – , 1 = 1000BASE-T 0 = – , 1 = 10GBASE-T 0 = – , 1 = 100BASE-KX 0 = – , 1 = 10GBASE-KX4 0 = – , 1 = 10GBASE-KR
Summary	set and get, value type: B, B, B, B, B, B
Parameters	
Example, get:	0/0 P_LPPARTNERAUTONEG 100BASE-TX 1000BASE-T 10GBASE-T – – –

P_LPSNRMARGIN chA chB chC chD

Explanation	Displays the SNR margin on the four link channels (Channel A-D) as reported by the PHY. It is displayed in units of 0.1dB.
Summary	set and get, value type: I, I, I, I
Parameters	
Example, get:	0/0 P_LPSNRMARGIN 380 376 384 370

P_TXTIMELIMIT microseconds

Explanation	A port-level time-limit on how long it keeps transmitting when started. After the elapsed time traffic must be stopped and restarted. This complements the stream-level PS_PACKETLIMIT function. **Note:** This feature is currently not available on 40G/100G ports.
Summary	set and get, value type: L
Parameters	*microseconds*: long, time limit after which the port stops transmitting.
Example, set or get:	0/1 P_TXTIMELIMIT 60000000

P_TXTIME microseconds

Explanation	How long the port has been transmitting, the elapsed time since traffic was started.
Summary	get, value type: L
Parameters	*microseconds*: long, elapsed time since traffic was started.
Example, get:	0/1 P_TXTIME 37123456

P_XMITONETIME nanoseconds

Explanation	The time at which the latest packet was transmitted using the P_XMITONEcommand. The time reference is the same used by the time stamps of captured packets.
Summary	get, value type: L
Parameters	*nanoseconds*: long, the time at which packet was transmitted.
Example, get:	0/1 P_XMITONETIME 123456789

P_MIXWEIGHTS weights

Explanation	Allow changing the distribution of the MIX packet size length by specifying the percentage of each of the 16 possible frame sizes used in the MIX. The sum of the percentage values specified must be 100. The command will affect the mix-distribution for all streams on the port.The possible 16 frame sizes are: 56 (not valid for 40G/100G), 60, 64, 70, 78, 92, 256, 496, 512, 570, 576, 594, 1438, 1518, 9216, and 16360. **Note**: This command requires Xena server version 375 or higher.
Summary	set and get, value type: I* (16 values in total)
Parameters	
Example set:	P_MIXWEIGHTS 0 25 25 25 25 0 0 0 0 0 0 0 0 0 0 0

P_CONFIG ?

Explanation	Multi-parameter query, obtaining all the settable parameters for a port itself, but excluding streams, filters, etc.
Summary	get only.
Parameters	None
Example, get:	• 0/5 P_SPEEDSELECTION F100M • 0/5 P_COMMENT ”This port generates IPTV background traffic.” • 0/5 P_SPEEDREDUCTION 100 • 0/5 P_INTERFRAMEGAP 20 • 0/5 P_MACADDRESS 0x001122AABBCC • 0/5 P_IPADDRESS 10.0.0.123 255.255.255.0 10.0.0.1 0.0.0.255 • 0/5 P_ARPREPLY ON • 0/5 P_PINGREPLY OFF • 0/5 P_PAUSE OFF • 0/5 P_RANDOMSEED 12345 • 0/5 P_LATENCYMODE LAST2LAST • 0/5 P_LATENCYOFFSET 1238 • 0/5 P_LOOPBACK L2RX2TX • <etc>

P_FULLCONFIG ?

Explanation	Multi-parameter query, obtaining all the settable parameters for a port, including streams, filters, capture, and datasets. These parameters comprise the complete user-definable configuration for the port.
Summary	get only.
Parameters	None
Example, get:	• 0/5 P_SPEEDSELECTION F100M • 0/5 P_COMMENT ”This port generates IPTV background traffic.” • 0/5 P_SPEEDREDUCTION 100 • 0/5 P_INTERFRAMEGAP 20 • <etc>

Tags: Scripting

Stream Parameters

The stream parameters correspond to the STREAM DEFINITION panel of the XenaManager, and deal with configuration of the traffic streams transmitted from a port.
The stream parameter names all have the form PS_xxx and require both a module index and a port index, as well as a sub-index identifying a particular stream.

General Information

Enabling Traffic

Whether the port is actually transmitting packets is controlled both by the P_TRAFFIC parameter for the parent port and by the PS_ENABLE parameter for the stream.
While the parent port is transmitting, the parameters of any enabled stream cannot be changed.

Stream Test Payload Data (TPLD)

Each Xena test packet contains a special proprietary data area called the *Test Payload Data* (TPLD) which contains various information about the packet. The TPLD is located just before the Ethernet FCS and consist of the following sections:

Normal TPLD (20 or 22 bytes)

Field	Length	Explanation
Sequence Number	3 byte	Packet sequence number used for loss and misordering detection.
Timestamp	4 byte	Timestamp value used for latency measurements.
Test Payload ID (TID)	2 byte	Test payload identifier used to identify the sending stream.
Payload Integrity Offset	1 byte	Offset in packet from where to calculate payload integrity.
First Packet Flag	1 bit	Set if this is the first packet after traffic is started.
Checksum Enabled	1 bit	Set if payload integrity checksum is used.
<reserved>	10 bit	
Timestamp Decimals	4 bit	Additional decimals for the timestamp.
Checksum	8 byte	Payload integrity checksum.
Total TPLD Size	**20 bytes**	

If the **Payload Checksum Offset** option is enabled on the parent port then an additional 2 byte checksum field is inserted in the TPLD sequence – just before the sequence number. This increases the total size of the TPLD to 22 bytes.

Micro-TPLD (6 bytes)

Field	Length	Explanation
First Packet Flag	1 bit	Set if this is the first packet after traffic is started.
<reserved>	1 bit	
Test Payload ID (TID)	10 bit	Test payload identifier used to identify the sending stream.
Timestamp	28 bit	Timestamp value used for latency measurements.
Checksum	8 bit	Payload integrity checksum (CRC-8).

The selection between the default TPLD or the micro-TPLD is done on the parent port. It is thus not possible to use different TPLD types for streams on the same port.
The TPLD function can also be completely disabled for any given stream by setting the Test Payload ID (TID) value for the stream to the value -1 (or the empty value in the XenaManager-2G GUI).

Minimum Packet Size Considerations

The stream will generally accept any configuration and attempt to transmit packets according to the configuration. In order for the various Xena stream features to work correctly certain aspects about the minimum packet size used must be observed.
The minimum packet size must obviously be large enough to accommodate the defined protocol headers + the final Ethernet FCS field.
If the TPLD function explained above is enabled then each packet must also be able to contain the TPLD area (20, 22 or 6 bytes depending on the configuration).
If the stream payload type is set to **Incrementing** then an additional minimum payload area of 2 bytes is needed. Otherwise excessive payload errors will be reported. This is however not necessary if the **Payload Checksum Offset** option is enabled on the parent port as this will override the payload integrity check implied by the**Incrementing** payload type.

STREAM PARAMETERS

PS_INDICES sid sid …

The full list of which streams are defined for a port. These are the sub-index values that are used for the parameters defining the traffic patterns transmitted for the port. Setting the value of this parameter creates a new empty stream for each value that is

not already in use, and deletes each stream that is not mentioned in the list. The same can be accomplished one-stream-at-a-time using the PS_CREATE and PS_DELETE commands.
sid: integer, the sub-index of a stream definition for the port.
Summary: set and get, value types: I*
Example, set or get:

- 0/1 PS_INDICES 0 1 5

PS_CREATE [sid]

Creates an empty stream definition with the specified sub-index value.
sid: integer, the sub-index value of the stream definition to create.
Summary: set only, stream index.
Example, set:

- 0/1 PS_CREATE [5]

PS_DELETE [sid]

Deletes the stream definition with the specified sub-index value.
sid: integer, the sub-index value of the stream definition to delete.
Summary: set only, stream index.
Example, set:

- 0/1 PS_DELETE [5]

PS_ENABLE [sid] state

This property determines if a stream contributes outgoing packets for a port. The value can be toggled between ON and SUPPRESS while traffic is enabled at the port level. Streams in the OFF state cannot be set to any other value while traffic is enabled.
The sum of the rates of all enabled or suppressed streams must not exceed the effective port rate.
sid: integer, the sub-index value of the stream definition.
state: coded integer, specifies the state of the stream:

- OFF (0) (stream will not be used when port traffic is started)
- ON (1) (stream will be started when port traffic is started)
- SUPPRESS (2) (stream will not be started when port traffic is started but can be started afterwards)

Summary: set and get, stream index, value type: B
Example, set or get:
0/1 PS_ENABLE [5] ON

PS_PACKETLIMIT [sid] count

The number of packets that will be transmitted when traffic is started on a port. A value of -1 makes the stream transmit continuously.
sid: integer, the sub-index value of the stream definition.
count: integer, the number of packets.
-1 (disable packet limitation)
Summary: set and get, stream index, value type: I
Example, set or get:

- 0/1 PS_PACKETLIMIT [5] 25

PS_COMMENT [sid] comment

The description of a stream.
sid: integer, the sub-index value of the stream definition. comment: string, containing the description of the stream.
Summary: set and get, stream index, value type: S
Example, set or get:

- 0/1 PS_COMMENT [5] ”Stream for …”

PS_TPLDID [sid] tpldid

The identifier of the test payloads inserted into packets transmitted for a stream. A value of -1 disables test payloads for the stream.
Test payloads are inserted at the end of each packet, and contains time-stamp and sequence-number information. This allows the receiving port to provide error-checking and latency measurements, in addition to the basic counts and rate measurements provided for all traffic.
The test payload identifier furthermore allows the receiving port to distinguish multiple different streams, which may originate from multiple different chassis. Since test payloads are an inter-port and inter-chassis mechanism, the test payload identifier assignments should be planned globally across all the chassis and ports of the testbed.
sid: integer, the sub-index value of the stream definition.
tpldid: integer, the test payload identifier value.
-1 (disable test payloads)
Summary: set and get, stream index, value type: I
Example, set or get:

- 0/1 PS_TPLDID [5] 17

PS_INSERTFCS [sid] onoff

Whether a valid frame checksum is added to the packets of a stream.
sid: integer, the sub-index value of the stream definition.
onoff: coded integer, whether frame checksums are inserted: OFF, ON
Summary: set and get, stream index, value type: B
Example, set or get:

- 0/1 PS_INSERTFCS [5] ON

PS_ARPREQUEST [sid] macaddress

Generates an outgoing ARP request on the test port.
The packet header for the stream must contain an IP protocol segment, and the destination IP address is used in the ARP request. If there is a gateway IP address specified for the port and it is on a different subnet than the destination IP address in the packet header, then the gateway IP address is used instead.
The framing of the ARP request matches the packet header, including any VLAN protocol segments.
This script parameter does not generate an immediate result, but waits until an ARP reply is received on the test port. If no reply is received within a short timeout it returns <FAILED>.
macaddress: hex bytes, specifying the six bytes of the MAC address.
Summary: get only, value type: HHHHHH
Example, get:

- 0/3 PS_ARPREQUEST [5] 0x001122776655

PS_PINGREQUEST [sid] delay ttl

Generates an outgoing ping request using the ICMP protocol on the test port.
The packet header for the stream must contain an IP protocol segment, with valid source and destination IP addresses.
The framing of the ping request matches the packet header, including any VLAN protocol segments, and the destination MAC address must also be valid, possibly containing a value obtained with PS_ARPREQUEST.
This script parameter does not generate an immediate result, but waits until a ping reply is received on the test port. If no reply is received within a short timeout it returns <FAILED>.
delay: integer, the number of milliseconds for the ping reply to arrive. ttl: byte, the time-to-live value in the ping reply packet.
Summary: get only, value type: I,B
Example, get:

- 0/3 PS_PINGREQUEST [5] 12 128

PS_RATEFRACTION [sid] fraction

The rate of the traffic transmitted for a stream, expressed in millionths of the effective rate for the port.
The bandwidth consumption includes the inter-frame gap, and is independent of the length of the packets generated for the stream. The sum of the bandwidth consumption for all the enabled streams must not exceed the effective rate for the port.
Setting this parameter also instructs the Manager to attempt to keep the rate-percentage unchanged in case it has to cap stream rates. Getting it is only valid if the rate was last set using this parameter. sid: integer, the sub-index value of the stream definition.
fraction: integer, stream rate expressed as a value between 0..1000000.
Summary: set and get, stream index, value type: I
Example, set or get:

- 0/1 PS_RATEFRACTION [5] 500000

PS_RATEPPS [sid] pps

The rate of the traffic transmitted for a stream, expressed in packets per second.
The bandwidth consumption is heavily dependent on the length of the packets generated for the stream, and also on the inter-frame gap for the port. The sum of the bandwidth consumption for all the enabled streams must not exceed the effective rate for
the port.
Setting this parameter also instructs the Manager to attempt to keep the packets-per-second unchanged in case it has to cap stream rates. Getting it is only valid if the rate was last set using this parameter. sid: integer, the sub-index value of the stream definition.
pps: integer, stream rate expressed as packets per second.
Summary: set and get, stream index, value type: I
Example, set or get:

- 0/1 PS_RATEPPS [5] 300000

PS_RATEL2BPS [sid] bps

The rate of the traffic transmitted for a stream, expressed in units of bits-per-second at layer-2, thus including the Ethernet header but excluding the inter-frame gap. The bandwidth consumption is somewhat dependent on the length of the packets generated for the stream, and also on the inter-frame gap for the port.
The sum of the bandwidth consumption for all the enabled streams must not exceed the effective rate for the port. Setting this parameter also instructs the Manager to

attempt to keep the layer-2 bps rate unchanged in case it has to cap stream rates.
Getting it is only valid if the rate was last set using this parameter.
sid: integer, the sub-index value of the stream definition. bps: long integer, stream rate expressed as bits-per-second.
Summary: set and get, stream index, value type: L
Example, set or get:

- 0/1 PS_RATEL2BPS [5] 800000000

PS_RATE [sid]?

Query the rate of the traffic transmitted for a stream in the manner it was last expressed. The response is one of PS_RATEFRACTION, PS_RATEPPS, or PS_RATEL2BPS.
This is the rate that is highlighted in the Manager, and the one it attempts to keep unchanged in case it has to cap stream rates.
sid: integer, the sub-index value of the stream definition.
Summary: get only, stream index.
Example, get:

- 0/1 PS_RATE [5] ?

PS_BURST [sid] size density

The burstiness of the traffic transmitted for a stream, expressed in terms of the number of packets in each burst, and how densely they are packed together.
The burstiness does not affect the bandwidth consumed by the stream, only the spacing between the packets.
A density value of 100 means that the packets are packed tightly together, only spaced by the minimum inter-frame gap. A value of 0 means even, non-bursty, spacing. The exact spacing achieved depends on the other enabled streams of the port.
Note: The sum of all stream burst size values for a port may not exceed 500 packets.
sid: integer, the sub-index value of the stream definition.
size: integer, the number of packets lumped together in a burst.
density: integer, the percentage of the available spacing that is inserted between bursts.
Summary: set and get, stream index, value types: I,I
Example, set or get:

- 0/1 PS_BURST [5] 4 50

PS_PACKETHEADER [sid] hexdata

The first portion of the packet bytes that are transmitted for a stream. This starts with the 14 bytes of the Ethernet header, followed by any contained protocol segments. All packets transmitted for the stream start with this fixed header. Individual byte positions of the packet header may be varied on a packet-to-packet basis using modifiers.
The full packet comprises the header, the payload, an optional test payload, and the frame checksum.
The header data is specified as raw bytes, since the script environment does not know the field-by-field layout of the various protocol segments. But XenaManager does, so in practice you may use XenaManager's packet editor, and simply query for the resulting hex string at the script level.
sid: integer, the sub-index value of the stream definition. hexdata: hex bytes, the raw bytes comprising the packet header.
Summary: set and get, stream index, value types: H*
Example, set or get:

- 0/1 PS_PACKETHEADER [5] 0x02AABB…

PS_HEADERPROTOCOL [sid] segment segment …

This parameter will inform the Xena tester how to interpret the packet header byte sequence specified with PS_PACKETHEADER.
This is mainly for information purposes, and the stream will transmit the packet header bytes even if no protocol segments are specified. The Xena tester however support calculation of certain field values in hardware, such as the IP, TCP and UDP length and checksum fields. This allow the use of hardware modifiers for these protocol segments. In order for this function to work the Xena tester needs to know the type of each segment that precedes the segment where the hardware calculation is to be performed.
Refer to this page for more details on hardware-based calculation of protocol fields.
sid: integer, the sub-index value of the stream definition.
segment: coded integer, a number specifying a built-in protocol segment:

- ETHERNET (14 bytes)
- VLAN (4 bytes)
- ARP (28 bytes)
- IP (20 bytes)
- IPV6 (40 bytes)
- UDP (8 bytes)
- TCP (20 bytes)
- LLC (3 bytes)
- SNAP (5 bytes)
- GTP (20 bytes)

- ICMP (8 bytes)
- RTP (12 bytes)
- RTCP (4 bytes)
- STP (35 bytes)
- SCTP (12 bytes)
- MACCTRL (4 bytes)
- MPLS (4 bytes)
- PBBTAG (4 bytes)
- FCOE (14 bytes)
- FC (24 bytes)
- FCOETAIL > > (4 bytes)
- IGMP0 (12 bytes)
- IGMP1 (16 bytes)
- -n (n bytes raw segment)

Summary: set and get, stream index, value types: B*
Example, set or get:

- 0/1 PS_HEADERPROTOCOL [5] ETHERNET -4 IP UDP

PS_MODIFIERCOUNT [sid] count

The number of standard 16-bit modifiers active on the packet header of a stream.
Each modifier is specified using**PS_MODIFIER**.
sid: integer, the sub-index value of the stream definition.
count: integer, the number of modifiers for the stream.
Summary: set and get, stream index, value type: I
Example, set or get:
0/1 PS_MODIFIERCOUNT [5] 1

PS_MODIFIER [sid,mid] pos mask act rep

A packet modifier for a stream header. The headers of each packet transmitted for the stream will be varied according to the modifier specification.
This parameter requires two sub-indices, one for the stream and one for the modifier.
A modifier is positioned at a fixed place in the header, selects a number of consecutive bits starting from that position, and applies an action to those bits in each packet. Packets can be repeated so that a certain number of identical packets are transmitted before applying the next modification.
sid: integer, the sub-index value of the stream definition.
mid: integer, the sub-index value of the modifier.
pos: integer, the byte position from the start of the packet.
mask: four hex bytes, the mask specifying which bits to affect.
act: coded integer, which action to perform on the affected bits:

- INC (increment)
- DEC (decrement)
- RANDOM (random)

rep: integer, how many times to repeat each packet.
Summary: set and get, stream index, modifier index, value types: I,HHHH,I,I
Example, set or get:

- 0/1 PS_MODIFIER [5,0] 6 0x0FC00000 RANDOM 1

PS_MODIFIERRANGE [sid,mid] min step max

Range specification for a packet modifier for a stream header, specifying which values the modifier should take on.
This applies only to incrementing and decrementing modifiers; random modifiers always produce every possible bit pattern.
The range is specified as a three values: mix, step, and max, where max must be equal to min plus a multiple of step. Note that when "decrement" is specified in PS_MODIFIER as the action, the value sequence will begin with the max value instead of the min value and decrement from there: {max, max-1, max-2, …., min, max, max-1…}.
sid: integer, the sub-index value of the stream definition.
mid: integer, the sub-index value of the modifier.
min: integer, the minimum modifier value.
step: integer, the increment between modifier values.
max: integer, the maximum modifier value.
Summary: set and get, stream index, modifier index, value types: I,I,I
Example, set or get:

- 0/1 PS_MODIFIERRANGE [5,0] 100 10 200

PS_MODIFIEREXTCNT [sid] count

The number of extended 24-bit modifiers active on the packet header of a stream. Each modifier is specified using **PS_MODIFIEREXT**.
Note: The extended modifier feature works for 40G/100G ports only.
Note: This feature requires Xena server version 375 or higher.
sid: integer, the sub-index value of the stream definition.
count: integer, the number of modifiers for the stream.
Summary: set and get, stream index, value type: I
Example, set or get:
0/1 PS_MODIFIERCOUNT [5] 1

PS_MODIFIEREXT [sid,mid] pos mask act rep

An extended packet modifier for a stream header. The headers of each packet transmitted for the stream will be varied according to the modifier specification. The modifier acts on 24 bits and takes up the space for two 16-bit modifiers to do this.
Note: The extended modifier feature works for 40G/100G ports only.
Note: This feature requires Xena server version 375 or higher.
This parameter requires two sub-indices, one for the stream and one for the modifier. A modifier is positioned at a fixed place in the header, selects a number of consecutive bits starting from that position, and applies an action to those bits in each packet. Packets can be repeated so that a certain number of identical packets are transmitted before applying the next modification.
sid: integer, the sub-index value of the stream definition.
mid: integer, the sub-index value of the modifier.
pos: integer, the byte position from the start of the packet. Cannot be < 1 !
mask: four hex bytes, the mask specifying which bits to affect.
act: coded integer, which action to perform on the affected bits:

- INC (increment) – Note: This is the only value supported for now.

rep: integer, how many times to repeat each packet. Note: For now the only value supported is 1.
Summary: set and get, stream index, modifier index, value types: I,HHHHHHHH,I,I
Example, set or get:
0/1 PS_MODIFIER [5,0] 6 0xFFFFFF00 INC1

PS_PACKETLENGTH [sid] type min max

The length distribution of the packets transmitted for a stream.
The length of the packets transmitted for a stream can be varied from packet to packet, according to a choice of distributions within a specified min..max range.
The length of each packet is reflected in the size of the payload portion of the packet, whereas the header has constant length.
Length variation complements, and is independent of, the content variation produced by header modifiers.
*sid**:*** integer, the sub-index value of the stream definition.
type: coded integer, the kind of distribution:

- FIXED (all packets have min size)
- INCREMENTING (incrementing from min to max)
- BUTTERFLY (min, max, min+1, max-1, min+2, max-2, etc)
- RANDOM (random between min and max)
- MIX (a mixture of sizes between 56 and 1518, average 464 bytes)

min: integer, lower limit on the packet length.
max: integer, upper limit on the packet length.
Summary: set and get, stream index, value types: I,I,I

Example, set or get:

- 0/1 PS_PACKETLENGTH [5] BUTTERFLY 100 1500

PS_PAYLOAD [sid] type hexdata

The payload content of the packets transmitted for a stream.
The payload portion of a packet starts after the header and continues up until the test payload or the frame checksum. The payload may vary in length, and is filled with either an incrementing sequence of byte values, or a repeated multi-byte pattern.
sid: integer, the sub-index value of the stream definition.
type: coded integer, the kind of payload content:

- PATTERN (a pattern is repeated up through the packet)
- INCREMENTING (bytes are incremented up through the packet)
- PRBS (bytes are randomized from packet to packet)
- RANDOM (a random generated pattern)

hexdata: hex bytes, a pattern of bytes to be repeated. The maximum length of the pattern is 18 bytes. Only used if *type* is set to PATTERN.
Summary: set and get, stream index, value types: B,H*
Example, set or get:

- 0/1 PS_PAYLOAD [5] PATTERN 0xAABB00FFEE

PS_CONFIG [sid]?

Multi-parameter query, obtaining all the parameters for a specific stream.
sid: integer, the sub-index value of the stream definition.
Summary: get only, stream index.
Example, get:
0/1 PS_ENABLE [5] ON
0/1 PS_PACKETLIMIT [5] 25
…
…
0/1 PS_PAYLOAD [5] PATTERN 0xAABB00FFEE

PS_FULLCONFIG ?

Multi-parameter query, obtaining all parameters for all streams defined on a port.
Summary: get only.
Example, get:

- 0/1 PS_INDICES 0 1 5 0/1 PS_ENABLE [0] ON 0/1 PS_PACKETLIMIT [0] 25 . . 0/1 PS_PAYLOAD [0] PATTERN 0xAABB00FFEE 0/1 PS_ENABLE [1] OFF . . 0/1 PS_ENABLE [5] ON . .

PS_INJECTFCSERR [sid]

Force a frame checksum error in one of the packets currently being transmitted from a stream. This can aid in analysing the error-detection functionality of the system under test. Traffic must be on for the port, and the stream must be enabled.
sid: integer, the sub-index value of the stream definition.
Summary: set only, stream index.
Example, set:

- 0/1 PS_INJECTFCSERR [5]

PS_INJECTSEQERR [sid]

Force a sequence error by skipping a test payload sequence number in one of the packets currently being transmitted from a stream. This can aid in analyzing the error-detection functionality of the system under test. Traffic must be on for the port, and the stream must be enabled and include test payloads.
sid: integer, the sub-index value of the stream definition.
Summary: set only, stream index.
Example, set:

- 0/1 PS_INJECSEQSERR [5]

PS_INJECTMISERR [sid]

Force a misorder error by swapping the test payload sequence numbers in two of the packets currently being transmitted from a stream. This can aid in analysing the error-detection functionality of the system under test. Traffic must be on for the port, and the stream must be enabled and include test payloads.
sid: integer, the sub-index value of the stream definition.
Summary: set only, stream index.
Example, set:

- 0/1 PS_INJECTMISERR [5]

PS_INJECTPLDERR [sid]

Force a payload integrity error in one of the packets currently being transmitted from a stream.
Payload integrity validation is only available for incrementing payloads, and the error is created by changing a byte from the incrementing sequence. The packet will have a correct frame checksum, but the receiving Xena chassis will detect the invalid payload based on information in the test payload. Traffic must be on for the port, and the stream must be enabled and include test payloads.

sid: integer, the sub-index value of the stream definition.
Summary: set only, stream index.
Example, set:

- 0/1 PS_INJECTPLDERR [5]

PS_INJECTTPLDERR [sid]

Force a test payload error in one of the packets currently being transmitted from a stream.
This means that the test payload will not be recognized at the receiving port, so it will be counted as a no- test-payload packet, and there will be a lost packet for the stream. Traffic must be on for the port, and the stream must be enabled and include test payloads.
sid: integer, the sub-index value of the stream definition.
Summary: set only, stream index.
Example, set:

- 0/1 PS_INJECTTPLDERR [5]

Filter Parameters

The filter and term parameters correspond to the FILTER TERMS and FILTER DEFINITION panels of the Manager, and deal with configuration of the basic terms and conditions active on the received traffic of a port.
The filter and term parameter names all have the form **PM_<xxx>**, **PL_<xxx>** and **PF_<xxx>**, and require both a module index and a port index, as well as a sub-index identifying a particular match term, length term, or filter.
The match terms and length terms provide basic true/false indications for each packet received on the port, and each filter specifies a compound Boolean condition on these true/false values to determine if the filter as a whole is true/false.
While a filter is enabled, neither its condition nor the definition of each match term or length term used by the condition can be changed.

PM_INDICES mid mid …

The full list of which match terms are defined for a port. These are the sub-index values that are used for the parameters defining the content-based matching of packets received for the port.
Settting the value of this parameter creates a new empty match term for each value that is not already in use, and deletes each match term that is not mentioned in the list. The same can be accomplished onematch-term-at-a-time using the PM_CREATE and PM_DELETE commands.
mid: integer, the sub-index of a match term definition for the port.
Summary: set and get, value types: I*
Example, set or get:

- 0/1 PM_INDICES 1 3

PM_CREATE *[mid]*

Creates an empty match term definition with the specified sub-index value.
mid: integer, the sub-index value of the match term definition to create.
Summary: set only, match term index.
Example, set:

- 0/1 PM_CREATE [3]

PM_DELETE *[mid]*

Deletes the match term definition with the specified sub-index value. A macth term cannot be deleted while it is used in the condition of any filter for the port.
mid: integer, the sub-index value of the match term definition to delete.

Summary: set only, match term index.
Example, set:

- 0/1 PM_DELETE [3]

PM_PROTOCOL *[mid] segment segment …*

The protocol segments assumed on the packets received on the port.
This is mainly for information purposes, and helps you identify which portion of the packet header is being matched. The actual value definition the match position is specified with PM_POSITION.
mid: integer, the sub-index value of the match term definition.
segment: coded integer, a number specifying a built-in protocol segment: Uses the same coded values as the PS_HEADERPROTOCOLparameter.
Summary: set and get, match term index, value types: B*
Example, set or get:

- 0/1 PM_PROTOCOL [3] ETHERNET VLAN

PM_POSITION *[mid] byteoffset*

The position within each received packet where content matching begins for the port.
mid: integer, the sub-index value of the match term definition.
byteoffset: integer, offset from the start of the packet bytes.
Summary: set and get, match term index, value types: I
Example, set or get:

- 0/1 PM_POSITION [3] 14

PM_MATCH [mid] mask value

The value that must be found at the match term position for packets received on the port. The mask can make certain bit positions don’t-care.
mid: integer, the sub-index value of the match term definition.
mask: eight hex bytes, which bits are significant in the match operation.
value: eight hex bytes, the value that must be found for the match term to be true.
Summary: set and get, match term index, value types: HHHHHHHH,HHHHHHHH
Example, set or get:

- 0/1 PM_MATCH [3] 0x0FFF000000000000 0x0123000000000000

PM_CONFIG *[mid]?*

Multi-parameter query, obtaining all the parameters for a specific match term.

mid: integer, the sub-index value of the match term definition.
Summary: get only, match term index.
Example, get:

- 0/1 PM_PROTOCOL [3] ETHERNET VLAN 0/1 PM_POSITION [3] 14 0/1 PM_MATCH [3] 0x0FFF000000000000 0x0123000000000000

PM_FULLCONFIG *?*

Multi-parameter query, obtaining all parameters for all match terms defined on a port.
Summary: get only.
Example, get:

0/1 0/1	PM_INDICES PM_PROTOCOL	1 3 [1]	ETHERNET	IP	UDP
.					
.					
0/1 0/1	PM_PROTOCOL PM_POSITION	[3] [3]	ETHERNET 14	VLAN	

0/1 PM_MATCH [3] 0x0FFF000000000000 0x0123000000000000

PL_INDICES *lid lid …*

The full list of which length terms are defined for a port. These are the sub-index values that are used for the parameters defining the length-based matching of packets received for the port. Settting the value of this parameter creates a new empty length term for each value that is not already in use, and deletes each length term that is not mentioned in the list. The same can be accomplished one- length-term-at-a-time using the PL_CREATE and PL_DELETE commands.
lid: integer, the sub-index of a length term definition for the port.
Summary: set and get, value types: I*
Example, set or get:

- 0/1 PL_INDICES 0

PL_CREATE *[lid]*

Creates an empty length term definition with the specified sub-index value.
lid: integer, the sub-index value of the length term definition to create.
Summary: set only, length term index.
Example, set:

- 0/1 PL_CREATE [0]

PL_DELETE [lid]

Deletes the length term definition with the specified sub-index value. A length term cannot be deleted while it is used in the condition of any filter for the port.
l*id:* integer, the sub-index value of the length term definition to delete.
Summary: set only, length term index.
Example, set:

- 0/1 PL_DELETE [0]

PL_LENGTH *[lid] type size*

The specification for a length-based check that is applied on the packets received on the port.

lid:	**integer, the sub-index value of the length term definition.**	
type:	coded integer, whether to test for shorter-than or longer-than:	
	AT_MOST	(packet length must be less-than-or-equal to specified size)
	AT_LEAST	(packet length must be greater-than-or-equal to specified size)
size:	integer, the value to compare the packet length against.	

Summary: set and get, length term index, value types: I,I
Example, set or get:

- 0/1 PL_LENGTH [0] AT_MOST 100

PL_FULLCONFIG *?*

Multi-parameter query, obtaining all parameters for all length terms defined on a port.
Summary: get only.
Example, get:

- 0/1 PL_INDICES 0
- 0/1 PL_LENGTH [0] AT_MOST 100

PF_INDICES *fid fid …*

The full list of which filters are defined for a port. These are the sub-index values that are used for the parameters defining the compound conditions on the match/length terms operating on the packets received for the port.
Settting the value of this parameter creates a new empty filter for each value that is not already in use, and deletes each filter that is not mentioned in the list. The same

can be accomplished one-filter-at-a-time using the PF_CREATE and PF_DELETE commands.
fid: integer, the sub-index of a filter definition for the port.
Summary: set and get, value types: I*
Example, set or get:

- 0/1 PF_INDICES 0 3

PF_CREATE *[fid]*

Creates an empty filter definition with the specified sub-index value.
fid: integer, the sub-index value of the filter definition to create.
Summary: set only, filter index.
Example, set:

- 0/1 PF_CREATE [3]

PF_DELETE *[fid]*

Deletes the filter definition with the specified sub-index value.
fid: integer, the sub-index value of the filter definition to delete.
Summary: set only, filter index.
Example, set:

- 0/1 PF_DELETE [3]

PF_ENABLE *[fid] onoff*

Whether a filter is currently active on a port.
While a filter is enabled its condition cannot be changed, nor can any match term or length terms used by it.
fid: integer, the sub-index value of the filter definition.
onoff: coded integer, whether the filter is enabled: OFF ON
Summary: set and get, filter index, value types: B
Example, set or get:

- 0/1 PF_ENABLE [3] OFF

PF_COMMENT *[fid] comment*

The description of a filter.
fid: integer, the sub-index value of the filter definition.
comment: string, containing the description of the filter.
Summary: set and get, filter index, value types: S

Example, set or get:

- 0/1 PF_COMMENT [3] ”Filter for …”

PF_CONDITION *[fid] terms terms …*

The boolean condition on the terms specifying when the filter is satisfied. The condition uses a cannonical and-or–not expression on the match terms and length terms.
The condition is specified using a number of compound terms, each encoded as an integer value specifying an abitrary set of the match terms and length terms defined for the port. Each match/length term has a specific power-of-two value, and the set is encoded as the sum of the values for the contained terms:
Value for match term [mid] = 2 ^ mid Value for length term [lid] = 2 ^ (lid+16)
A compound term is true if all the match terms and length terms contained in it are true. This supports the and-part of the condition.
If some compound term is satisfied, the condition as a whole is true. This is the or-part of the condition.
The first few compound terms at the even positions (second, fourth, …) are inverted, and all the contained match terms and length terms must be false at the same time that the those of the preceding compound term are true. This is the not-part of the condition.
In practice, the simplest way to generate these encodings is to use the Manager, which supports Boolean expressions using the operators &, |, and ~, and simply query the chassis for the resulting script-level definition.
fid: integer, the sub-index value of the filter definition. terms: integer, encoding a compound term which is a set of the match terms and length terms.
Summary: set and get, filter index, value types: I*
Example, set or get:

- 0/1 PF_CONDITION [3]

PF_CONFIG *[fid]?*

Multi-parameter query, obtaining all the parameters for a specific filter.
fid: integer, the sub-index value of the filter definition.
Summary: get only, filter index.
Example, get:

- 0/1 PF_COMMENT [3] ”Filter for …” 0/1 PF_CONDITION [3] 0/1 PF_ENABLE [3] OFF

PF_FULLCONFIG *?*

Multi-parameter query, obtaining all parameters for all filters defined on a port.
Summary: get only.
Example, get:

- 0/1 PF_INDICES 0 3 0/1 PF_COMMENT [0] ”Filter for …” . . 0/1 PF_COMMENT [3] ”Filter for …” 0/1 PF_CONDITION [3] 0/1 PF_ENABLE [3] OFF

Capture Parameters

The capture parameters correspond to the Capture panel of the XenaManager, and deal with configuration of the capture criteria and inspection of the captured data from a port.
Whether the port is enabled for capturing packets is specified by the P_CAPTURE parameter. Captured packets are indexed starting from 0, and are stored in a buffer that is cleared before capture starts. While on, the capture configuration parameters cannot be changed.
The capture parameter names all have the form **PC_<xxx>** and require both a module index and a port index. The per-packet parameters also use a sub-index identifying a particular packet in the capture buffer.

PC_TRIGGER start filter1 stop filter2

Explanation	The criteria for when to start and stop the capture process for a port. Even when capture is enabled with P_CAPTURE, the actual capturing of packets can be delayed until a particular start criteria is met by a received packet. Likewise, a stop criteria can be specified, based on a received packet. If no explicit stop criteria is specified capture stops when the internal buffer runs full. In buffer overflow situations, if there is an explicit stop criteria then the latest packets will be retained (and the early ones discarded), and otherwise the earliest packets are retained (and the later ones discarded).
Summary	set and get, value types: I,I,I,I
Parameters	*start:* coded integer, the criteria for starting the actual packet capture: • ON (start immediately when capture is started) • FCSERR (start when receiving a packet containing a frame checksum error) • FILTER (start when receiving a packet satisfying a filter condition) • PLDERR (start when receiving a packet containing a packet payload error) *filter1:* integer, the index of a particular filter for the start criteria. *stop:* coded integer, the criteria for stopping the actual packet capture: • FULL (continue until the capture buffer runs full) • FCSERR (continue until receiving a packet with a frame checksum error) • FILTER (continue until receiving a packet satisfying a filter condition) • PLDERR (continue until receiving a packet with a packet payload error) *filter2:* integer, the index of a particular filter for the stop criteria.
Example	0/1 PC_TRIGGER FILTER 3 FULL 0

PC_KEEP which index bytes

Explanation	Which packets to keep once the start criteria has been triggered for a port. Also how big a portion of each packet to retain, saving space for more packets in the capture buffer.
Summary	set and get, value types: I,I,I
Parameters	*which*: coded integer, which general kind of packets to keep: • ALL (keep all packets between the start and stop trigger) • FCSERR (keep only those packets with frame checksum errors) • NOTPLD (keep only those packets without a test payload) • TPLD (keep only those packets with a test payload and specific id) • FILTER (keep only those packets satisfying a specific filter condition) *index*: integer, test payload id or filter index for which packets to keep. *bytes*: integer, how many bytes to keep in the buffer for of each packet. The value -1 means no limit on packet size.
Example	0/1 PC_KEEP TPLD 17 30

PC_STATS status packets starttime

Explanation	Obtains the number of packets currently in the capture buffer for a port. The count is reset to zero when capture is turned on.
Summary	get only, value types: L,L,L
Parameters	*status*: long integer, 1 if capture has been stopped because of overflow, 0 if still running. *packets*: long integer, the number of packets in the buffer. *starttime*: long integer, time when capture was started, in nano seconds since midnight 1/1/2010.
Example	0/1 PC_STATS 0 987 3453543453

PC_PACKET [cid] hexdata

Explanation	Obtains the raw bytes of a captured packet for a port. The packet data may be truncated if the PC_KEEPparameter specified a limit on the number of bytes kept.
Summary	get only, capture packet index, value types: H*

Parameters	*cid:* integer, the sub-index value of the captured packet. hexdata: hex bytes, the raw bytes kept for the packet.
Example	0/1 PC_PACKET [986] 0x00AA00CC…

PC_EXTRA [cid] time latency ifg length

Explanation	Obtains extra information about a captured packet for a port. The information comprises time of capture, latency, inter-frame gap, and original packet length. Latency is only valid for packets with test payloads and where the originating port is on the same module and therefore has the same clock.
Summary	get, capture packet index, value types: L,L,L,I
Parameters	*cid:* integer, the sub-index value of the captured packet. *time:* long integer, the number of nano-seconds since capture was started. *latency:* long integer, the number of nano-seconds since packet was transmitted. *ifg:* long integer, the number of byte-times since previous packet. *length:* integer, the real length of the packet on the wire.
Example	0/1 PC_EXTRA [986] 30000000 40000 100 555

PC_INFO [cid]?

Explanation	Multi-parameter query, obtaining all the information for a particular captured packet for a port.
Summary	get only, capture packet index.
Parameters	*cid:* integer, the sub-index value of the captured packet.
Example	0/1 PC_PACKET [986] 0x00AA00CC… 0/1 PC_EXTRA [986] 30000000 40000 100 555

PC_FULLCONFIG ?

Explanation	Multi-parameter query, obtaining all parameters of the capture configuration for a port. This does not include any captured data itself.
Summary	get only.
Parameters	None

Example	0/1 PC_TRIGGER FILTER 3 FULL 0 0/1 PC_KEEP TPLD 17 30

Statistics Parameters

The statistics parameters correspond to the Transmit Statistics and Receive Statistics panels of the XenaManager, and provide quantitative information about the transmitted and received packets on a port.
The statistics parameter names all have the form **PT_<xxx>** and **PR_<xxx>** and require both a module index and a port index. Those parameters dealing with a specific transmitted stream also have a sub-index, and so have the parameters dealing with a specific received test payload id and a specific filter.
All bit-and byte-level statistics are at layer-2, so they include the full Ethernet frame, and exclude the inter-frame gap and preamble.

PT_TOTAL bps pps bytes packets

Obtains statistics concerning all the packets transmitted on a port.
bps: long integer, number of bits transmitted in the last second.
pps: long integer, number of packets transmitted in the last second.
bytes: long integer, number of bytes transmitted since statistics were cleared.
packets: long integer, number of packets transmitted since statistics were cleared.
Summary: get only, value types: L,L,L,L
Example, get:
0/0 PT_TOTAL 8000000000 15000000 12345678987654 123456789876

PT_NOTPLD bps pps bytes packets

Obtains statistics concerning the packets without a test payload transmitted on a port.
bps: long integer, number of bits transmitted in the last second.
pps: long integer, number of packets transmitted in the last second.
bytes: long integer, number of bytes transmitted since statistics were cleared.
packets: long integer, number of packets transmitted since statistics were cleared.
Summary: get only, value types: L,L,L,L
Example, get:

- 0/0 PT_NOTPLD 800000 1000 1234567 12345

PT_EXTRA arptxreq arptxrsp pingtxreq pingtxrsp fcs seq mis pld tpld train

Obtains additional statistics for packets transmitted on a port.

arptxreq: long integer, number of ARP requests transmitted
arptxrsp: long integer, number of ARP responses transmitted
pingtxreq: long integer, number of PING requests transmitted
pingtxrsp: long integer, number of PING responses transmitted
fcs: long integer, number of FCS errors injected
seq: long integer, number of sequence mismatch errors injected
mis: long integer, number of packet misordering errors injected
pld: long integer, number of payload errors injected
tpld: long integer, number of payload integrity errors injected
train: long integer, number of MAC learning (training) packets transmitted
Summary: get only, value types: L*
Example, get:
0/0 PT_EXTRA 1 0 2 3 0 0 0 1 0 4

PT_STREAM [sid] bps pps bytes packets

Obtains statistics concerning the packets of a specific stream transmitted on a port.
sid: integer, the sub-index value of the stream definition.
bps: long integer, number of bits transmitted in the last second.
pps: long integer, number of packets transmitted in the last second. bytes: long integer, number of bytes transmitted since statistics were cleared.
packets: long integer, number of packets transmitted since statistics were cleared.
Summary: get only, stream index, value types: L,L,L,L
Example, get:

- 0/0 PT_STREAM [5] 800000000 1500000 1234567898765 12345678987

PT_ALL ?

Multi-parameter query, obtaining all the transmit statistics for a port.
Summary: get only.
Example, get:

- 0/0 PT_TOTAL 8000000000 15000000 12345678987654 123456789876
- 0/0 PT_NOTPLD 800000 1000 1234567 12345 0/0 PT_STREAM [0] 800000000 1500000 1234567898765 12345678987 0/0 PT_STREAM [1] 700000000 1300000 1134567898765 11345678987 0/0 PT_STREAM [5] 600000000 1100000 1034567898765 10345678987

PT_CLEAR

Clear all the transmit statistics for a port. The byte and packet counts will restart at zero.

Summary: set only.
Example, set:

- 0/0 PT_CLEAR

PR_TOTAL bps pps bytes packets

Obtains statistics concerning all the packets received on a port.
bps: long integer, number of bits received in the last second.
pps: long integer, number of packets received in the last second.
bytes: long integer, number of bytes received since statistics were cleared.
packets: long integer, number of packets received since statistics were cleared.
Summary: get only, value types: L,L,L,L
Example, get:

- 0/0 PR_TOTAL 8000000000 15000000 12345678987654 123456789876

PR_NOTPLD bps pps bytes packets

Obtains statistics concerning the packets without a test payload received on a port.
bps: long integer, number of bits received in the last second.
pps: long integer, number of packets received in the last second.
bytes: long integer, number of bytes received since statistics were cleared.
packets: long integer, number of packets received since statistics were cleared.
Summary: get only, value types: L,L,L,L
Example, get:

- 0/0 PR_NOTPLD 800000 1000 1234567 12345

PR_EXTRA *miscstats…*

Obtains statistics concerning special packets received on a port since receive statistics were cleared.

fcserrors:	long integer, number of packets with frame checksum errors.
pauseframes:	long integer, number of Ethernet pause frames.
arprequests:	long integer, number of ARP request packets received.
arpreplies:	long integer, number of ARP reply packets received.
pingrequests:	long integer, number of PING request packets received.
pingreplies:	long integer, number of PING reply packets received.
gapcount:	long integer, number of gap monitor gaps encountered.

gapduration:	long integer, combined duration of gap monitor gaps encountered, microseconds.

Summary: get only, value types: L*
Example, get:

- 0/0 PR_EXTRA 0 1230000

PR_TPLDS *tid tid* …

Obtain the set of test payload ids observed among the received packets since receive statistics were cleared. Traffic statistics for these test payload streams will have non-zero byte and packet count. tid: integer, identifier of test payload with non-zero packet count.
Summary: get only, value types: I*
Example, get:

- 0/0 PR_TPLDS 17 77

PR_TPLDTRAFFIC [tid] bps pps byt pac

Obtains traffic statistics concerning the packets with a particular test payload id received on a port.
tid: integer, the identifier of the test payload.
bps: long integer, number of bits received in the last second.
pps: long integer, number of packets received in the last second.
byt: long integer, number of bytes received since statistics were cleared.
pac: long integer, number of packets received since statistics were cleared.
Summary: get only, test payload id, value types: L,L,L,L
Example, get:

- 0/0 PR_TPLDTRAFFIC [17] 80000000 150000 123456789876 1234567898

PR_TPLDERRORS [tid] dummy seq mis pld

Obtains statistics concerning errors in the packets with a particular test payload id received on a port.
The error information is derived from analysing the various fields contained in the embedded test payloads of the received packets, independent of which chassis and port may have originated the packets.
Note that packet-lost statistics involve both a transmitting port and a receiving port, and in particular knowing which port originated the packets with a particular test payload identifier. This information requires knowledge of the global test environment, and is not supported at the port-level.

Index Values

tid	integer, the identifier of the test payload
Data Values	
dummy	long integer, not used
seq	long integer, number of non-incrementing-sequence-number events.
mis	long integer, number of swapped-sequence-number misorder events.
pld	long integer, number of packets with non-incrementing payload content.

Summary: get only, test payload id, value types: L,L,L,L
Example, get:

- 0/0 PR_TPLDERRORS [17] 0 1 0 0

PR_TPLDLATENCY [tid] min avg max avg1sec min1sec max1sec

Obtains statistics concerning the latency experienced by the packets with a particular test payload id received on a port. The values are adjusted by the port-level P_LATENCYOFFSET value.
A special value of -1 is returned if latency numbers are not applicable.
Latency is only meaningful when the clocks of the transmitter and receiver are synchronized. This requires the two ports to be on the same test module, and it requires knowledge of the global test environment to ensure that packets are in fact routed between these ports.

tid	integer, the identifier of the test payload
min	long integer, nanoseconds, minimum latency for test payload stream
avg	long integer, nanoseconds, average latency for test payload stream
max	long integer, nanoseconds, maximum latency for test payload stream
avg1sec	long integer, nanoseconds, average latency over last 1-second period
min1sec*	long integer, nanoseconds, minimum latency during last 1-second period
max1sec*	long integer, nanoseconds, maximum latency during last 1-second period

Summary: get only, test payload id, value types: L,L,L,L
Example, get:

- 0/0 PR_TPLDLATENCY [17] 2400 2900 3700 2800
- Note, the last two values were added in Release 57.0.

PR_TPLDJITTER *[tid] min avg max avg1sec min1sec max1sec*

Obtains statistics concerning the jitter experienced by the packets with a particular test payload id received on a port. The values are the difference in packet-to-packet latency, and the minimum will usually be zero.
A special value of -1 is returned if jitter numbers are not applicable. They are only available for TID values 0..31.

tid	integer, the identifier of the test payload
min	long integer, nanoseconds, minimum jitter for test payload stream
avg	long integer, nanoseconds, average jitter for test payload stream
max	long integer, nanoseconds, maximum jitter for test payload stream
avg1sec	long integer, nanoseconds, average jitter over last 1-second period
min1sec*	long integer, nanoseconds, minimum jitter during last 1-second period
max1sec*	long integer, nanoseconds, maximum jitter during last 1-second period

Summary: get only, test payload id, value types: L,L,L,L
Example, get:

- 0/0 PR_TPLDJITTER [17] 0 1234 2900 1345
- Note, the last two values were added in Release 57.0.

PR_FILTER *[fid] bps pps bytes packets*

Obtains statistics concerning the packets satisfying the condition of a particular filter for a port.
fid: integer, the sub-index of the filter definition.
bps: long integer, number of bits received in the last second.
pps: long integer, number of packets received in the last second.
bytes: long integer, number of bytes received since statistics were cleared.
packets: long integer, number of packets received since statistics were cleared.
Summary: get only, filter index, value types: L,L,L,L
Example, get:

- 0/0 PR_FILTER [3] 80000000 150000 123456789876 1234567898

PR_ALL *?*

Multi-parameter query, obtaining all the receive statistics for a port.
Summary: get only.

Example, get:

- 0/0 PR_TOTAL 8000000000 15000000 12345678987654 123456789876 0/0 PR_NOTPLD 800000 1000 1234567 12345 0/0 PR_EXTRA 0 123 0 0 0 0 0 0/0 PR_TPLDS 17 77 0/0 PR_TPLDTRAFFIC [17] 80000000 150000 123456789876 1234567898 0/0 PR_TPLDERRORS [17] 0 1 0 0 0/0 PR_TPLDLATENCY [17] 2400 2900 3700 0/0 PR_TPLDTRAFFIC [77] 80000000 150000 123456789876 1234567898 . . 0/0 PR_FILTER [3] 80000000 150000 123456789876 1234567898

PR_ALLERRORS ?

Multi-parameter query, obtaining all the test payload id error statistics for a port.
Summary: get only.
Example, get:

- 0/0 PR_TPLDS 17 77 0/0 PR_TPLDERRORS [17] 0 1 0 0 0/0 PR_TPLDERRORS [77] 2 3 0 7

PR_CALIBRATE

Calibrate the latency calculation for packets received on a port. The lowest detected latency value (across all TIDs) will be set as the new base.
Summary: set only.
Example, set:

- 0/1 PR_CALIBRATE

PR_CLEAR

Clear all the receive statistics for a port. The byte and packet counts will restart at zero.
Summary: set only.
Example, set:

- 0/0 PR_CLEAR