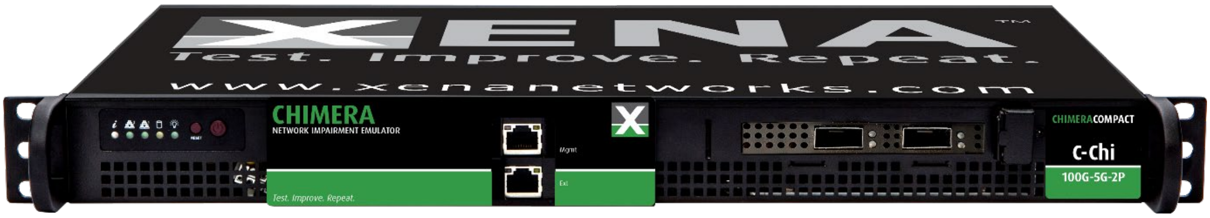


CHIMERA

NETWORK IMPAIRMENT EMULATOR

User Manual Rel. 86.0

May 2021

WWW.XENANETWORKS.COM

Table of Contents

Table of Figures

Table of tables

Table of Acronyms

Acronym	Meaning
BER	Bit Error Rate
CBS	Committed Burst Size
CIR	Committed Information Rate
CLI	Command Line Interface
DAC	Direct Attach Cable
DIP	Destination IP address
DMAC	Destination MAC address
DSCP	Differentiated Services Bode Point
ESMC	Ethernet Synchronization Message Channel
FCS	Frame Check Sequence
FID	Flow ID
IID	Impairment ID
LAN	Local Area Network
MAC	Media Access Control
MPLS	Multiprotocol Label Switching
N.A.	Not Available / Not Applicable
PCP	Priority Code Point
PCS	Physical Coding Sublayer
PMA	Physical Medium Attachment
Rx	Receive
SIP	Source IP address
SMAC	Source MAC address
TC	Traffic Class
TCP	Transmission Control Protocol
TPLD	Test Payload
TPID	Test Payload ID
Tx	Transmit
UDP	User Datagram Protocol
UI	User Interface
VID	VLAN ID
VLAN	Virtual LAN

1 Overview

Chimera is an Ethernet network emulator, which is inserted between two Ethernet ports to emulate a network, by applying impairments to the packets being forwarded between the ports.

This document explains the functionality of Chimera and how to configure it. The configuration is illustrated using both the User Interface (UI) named ValkyrieManager and the script commands. Chimera supports the general Xena script commands for ports and modules and a number of Chimera specific script commands for configuring impairments.

This document assumes that the user is logged into the Chimera impairment emulator with ValkyrieManager. If not, please refer to the “Chimera Quick Start Guide”.

1.1 Chimera script interface.

Chimera can be 100% controlled using scripting commands. I.e. all configuration and all statistics can be accessed via scripting.

This document includes examples on how to configure Chimera using script commands, by providing simple script examples for each of the described functions. For further details on the script commands supported by Chimera, please refer to “Xena script commands for Chimera”.

In the script command examples in this document, SYMBOLIC constants are used where possible. The constants are followed by the numeric number in parenthesis.

E.g.

```
PEF_IPV4SETTINGS [fid,0] ON (1) INCLUDE (1)
```

This includes the following SYMBOLIC constants:

- ON = 1
- INCLUDE = 1

The “fid” is the filter ID, which is explained in the following.

1.1.1 UI script client.

One way of setting up Chimera using script commands, is to use the “Script Client” built into the UI.

To open the script client, select the chassis in the UI and right click and select “Open Script Client”. This is illustrated in Figure 1.

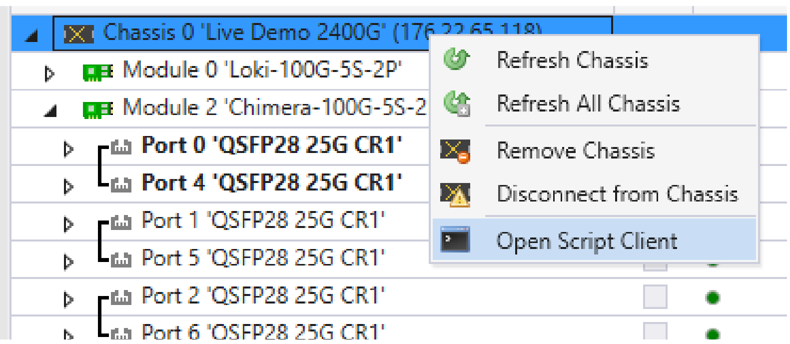


Figure 1: How to open "Script Client".

Once the script client is open, you can directly input script commands. The use of the script client is illustrated in Figure 2.

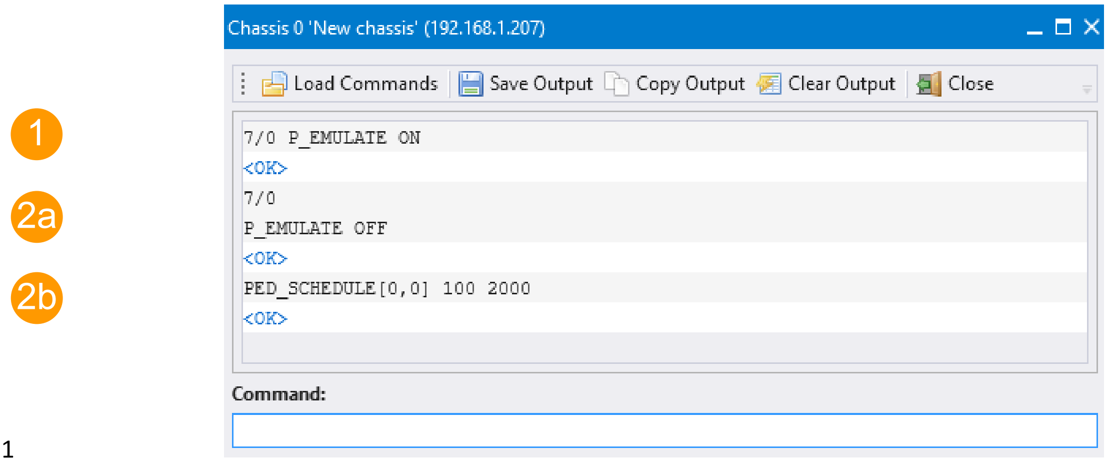


Figure 2: Using the "Script Client".

When using the script client to enter script commands, there are two ways to specify the module and port on which to apply the command:

1. Writing the <MODULE>/<PORT> in front of the command. (See Figure 2 – step 1, which illustrates how to apply the command to module 7 and port 0).
2. First, enter the <MODULE>/<PORT> (See Figure 2 – step 2a). Subsequently, enter the commands, which will then be executed on the module and port previously configured (See Figure 2 – step 2b).

Because the module and port will depend on the chassis configuration, all script commands in the following will be listed without the module and port values. I.e., to use the script command examples directly in the script client, the user is required to configure the <MODULE> / <PORT> in advance as illustrated in Figure 2 – step 2a.

1.2 Ethernet packet forwarding

Chimera is an impairment module, which works as a “bump-in-a-wire” forwarding Ethernet packets of sizes from 56 bytes to 12288 bytes. The ports are divided into fixed port pairs, and packets are forwarded between the ports in the port pair. The port pairs for different speeds are illustrated in Table 1.

Speed	Port pairs		
100G / 40G	Port 0	↔	Port 1
50G	Port 0	↔	Port 2
	Port 1	↔	Port 3
25G / 10G	Port 0	↔	Port 4
	Port 1	↔	Port 5
	Port 2	↔	Port 6
	Port 3	↔	Port 7

Table 1: Chimera port pairs.

The port pairs as seen in ValkyrieManager are illustrated in Figure 3.

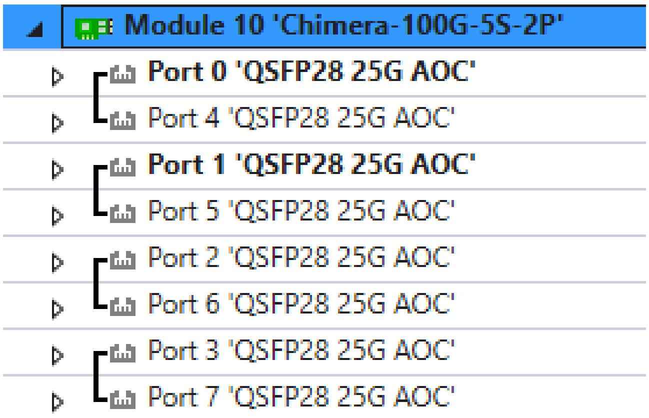


Figure 3: Chimera port pairs in ValkyrieManager.

1.3 Chimera impairment pipeline

Chimera implements up to 8 flows per port. Each flow implements a separate impairment pipeline, where the packets of that flow can be impaired independently of other flows.

Packets received on a port in Chimera will pass through the active flow filters. If the packet matches a filter, the packet is mapped to the corresponding flow and passes through the corresponding impairment pipeline.

The impairment pipeline includes the following configurable impairments:

- Policing
- Drop
- Misorder
- Latency / Jitter
- Corruption
- Duplication
- Shaping

Notice that the impairment pipeline includes a delay block. This delay block is responsible for generating both a fixed latency and a variable latency in terms of jitter. Throughout this document, the delay block is referred to as the “latency / jitter” impairment block.

The flow filters and corresponding impairment pipelines are illustrated in Figure 4.

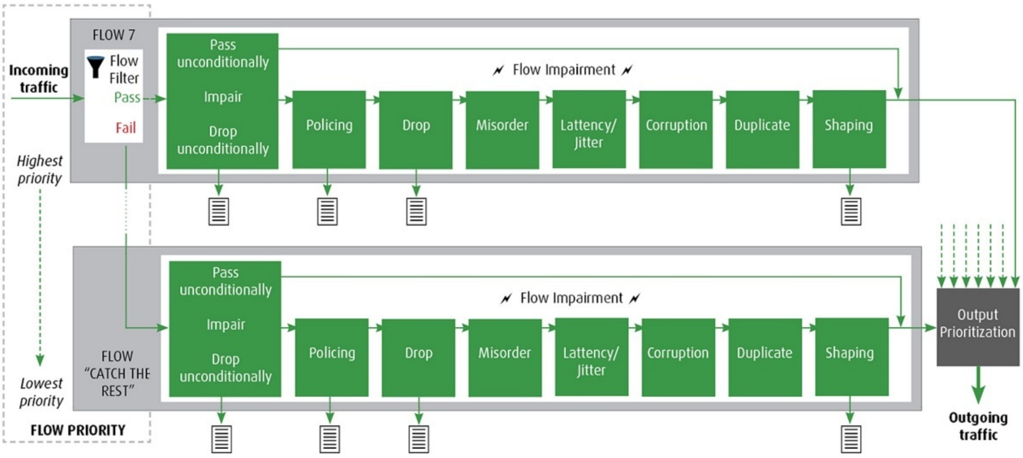


Figure 4: Chimera impairment pipeline.

Each impairment in each flow impairment pipeline can be individually configured. Notice that Figure 4 illustrates that packets may be dropped as a consequence of the following impairments:

- Policing
- Drop
- Shaping

After the flow specific impairment pipeline, all packets destined for a given output port are merged into a common packet stream and forwarded to wire.

2 Latency / jitter explained

The architecture of the Chimera delay block puts certain limits on the minimum and maximum latency that can be configured in the latency / jitter impairment described in section 10.6.

This section describes these limits, along with timing configuration parameters and the timing accuracy that can be expected from the latency / jitter impairment, depending on the configuration. Besides the latency of the active emulator core described in this section, notice the emulator bypass mode described in section 3.2.

The minimum latency that can be configured for any latency distribution is described in section 2.2. Regarding the maximum latency that can be configured for any latency distribution, there are two limits to be aware of:

- Lossless Latency:
  The maximum latency that can be guaranteed without the risk of packet loss at wire speed.
- Lossy Latency (Reduced Bandwidth Latency):
  The maximum latency supported by Chimera. At this latency, there will be loss at wire speed. It is possible to calculate a reduced BW which can be supported without loss. Sending packets at a higher bandwidth than the guaranteed BW will eventually result in packet loss. See section 2.4 for details on reduced BW.

When configuring a latency distribution, it is possible to configure the maximum delay to the lossy latency limit. However, when configuring a maximum above the lossless latency limit, sending packets at a higher BW than the guaranteed reduced rate may result in packets being lost.

Maximum Lossless Latency is described in section 2.3, while Reduced Bandwidth Latency is described in section 2.4.

Finally, section 2.5 describes the latency / jitter accuracy that can be expected depending on configuration.

2.1 Extended Timing Mode.

The latency / jitter impairment can operate in either "Normal Timing Mode" or "Extended Timing Mode". Normal timing mode will allow high precision latency and jitter, with a maximum configurable latency of 1.9 sec.

Extended timing mode allows configuring latencies up to 19.5 sec. at the expense of the latency and jitter precision. The minimum configurable latency is unaffected by the setting of extended timing mode.

The timing mode is configured at the module level as illustrated in Figure 5, and will apply to the entire Chimera module.

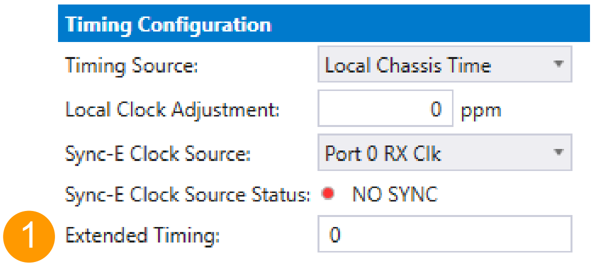


Figure 5: Configuring "Extended Timing Mode".

Notice: Changing the timing mode will reset all configured latency / jitter parameters in the entire module to default values, including those configured for custom distributions. Hence, modifying the timing mode will require all timing values in the module to be reconfigured.

<u>To configure extended timing mode:</u>

1) Go to the "Module → Resource Properties" tab and select the required timing mode (e.g. off).

This will set the extended timing mode to off and cause all timing parameters in the module to be reset to default values.

<u>Script configuration example:</u>

The example below illustrates how to turn off the extended timing mode.

```
M_LATENCYMODE NORMAL (0)
```

Due to decreased latency / jitter precision, it is recommended that users enable extended timing mode only when the increased maximum delay is required for your testing.

2.2 Minimal latency

Chimera supports the following minimum latency depending on port speed:

Speed	Min. latency
100G (FEC / no FEC)	7.0 us
50 G	7.0 us
40 G	7.0 us
25 G (no FEC)	7.2 us
25 G (FEC)	7.2 us
10 G	13 us

Table 2: Chimera minimum latency.

The minimum latency is significantly increased for 10G due to the store and forward delay of a 10K Ethernet packet at 10G.

Notice that the minimum latency is unaffected by the setting of the timing mode described in section 2.1.

2.3 Lossless latency

Due to the amount of memory needed to support latency, there is an upper limit to the latency which can be supported without loss. The maximum latency without loss which can be configured for a flow depends on the number of ports and flows currently active on the port.

Speed	# of active flows on port	Max lossless lat.
100G, 50G, 25G	1	160 ms
	2	80 ms
	3, 4	40 ms
	5-8	20 ms
40G	1	400 ms
	2	200 ms
	3,4	100 ms
	5-8	50 ms
10G	1	192 ms
	2	96 ms
	3,4	48 ms
	5-8	24 ms

Table 3: Chimera maximum latency (lossless).

The total amount of “latency / jitter” memory inside Chimera is constant. This memory is divided equally between the active flows, as is reflected in Table 3, which illustrates how the maximum lossless delay depends on the number of active flows.

The distribution of memory among active flows mentioned above, requires re-allocating the memory, when the number of active flows is modified. If traffic is running through the filters, when memory is re-allocated, packets will be lost on all active flows.

To avoid packet loss due to memory re-allocation, enable all required filters, before starting the traffic. Subsequently modifying the filters will not result in any packet loss.

2.4 Lossy latency (Reduced Bandwidth Latency)

Chimera supports latencies above what is listed in Table 3, but in such cases, it can only be guaranteed to be lossless at a reduced bandwidth given by:

$$ReducedBW(Gb/s) = \frac{LossLessLatency * Speed(Gb/s)}{ConfiguredLatency}$$

Where the “LosslessLatency” is taken from Table 3 and “ConfiguredLatency” is the latency currently configured for the flow (> LosslessLatency).

In case the average data rate on the flow exceeds the reduced bandwidth, packets will be dropped.

The maximum latency which can be configured for reduced bandwidth = 1.9 sec (normal timing mode) / 19.5 sec. (extended timing mode). See section 2.1 for details.

2.5 Latency and multiple flows

When only the default flow is configured on a port, the uncertainty on the configured latency is +/- 50 ns.

When multiple flows are configured on a port, there will be an added latency due to the fact that packets from multiple flows need to be merged onto the same physical link at the Chimera output port. This is a basic property of Ethernet. In this case, the added latency will depend on the number of flows configured on the port and the maximum packet size on the active flows.

When adding a 2nd flow (flow #1) to the port, the packets of the default flow risk waiting to be merged into the common output packet stream due to transmission of a packet on flow #1. Worst case, this is a maximum size packet (= 10K bytes), in which case 802 ns (for 100 G) will be added to the latency of the packet of the default flow (see Table 4).

For every flow added to the port, the packets of a given flow risk waiting another maximum packet size to be merged at the output. The influence of the multiple flows is very random, but worst case, a packet scheduled to be sent on a port with 8 flows configured will experience an increased delay of 7 x maximum packet delay.

The worst case added latencies in cases with 8 flows on a port are listed in Table 4.

Speed	Delay of 10K pkt (ns)	Max added delay (us)
100G	802	5.6
50G	1,603	11.2
40G	2,004	14.0
25G	3,206	22.4
10G	8,016	56.1

Table 4: Chimera maximum added latency.

3 Module settings

This section describes the settings that will affect the entire module, as opposed to port level or flow level settings.

3.1 SyncE

Chimera implements a single clock domain for clocking all Tx ports. The source clock can be configured to be an internally generated clock or a recovered clock from one of the active Rx ports.

To successfully synchronize the Tx ports to a recovered Rx clock, the internal circuitry must be able to lock on the configured Rx clock. Chimera implements a locking signal which indicates whether Chimera is currently locked to a Rx port. This “Rx lock” signal MUST be ON before you can trust that the Tx ports are running of the configured Rx port.

If the Rx lock signal returns NOVALIDTXCLK, it implies that Chimera could not lock to the configured Rx port clock, in which case it will fall back to running off the internally generated clock.

Notice that when configured to run off the internal clock (Module Local Clock), the Rx lock signal will always return NOVALIDTXCLK.

Parameter	Legal values	Comments	Step size
Clk source:	0 → 9	0; (=MODULELOCALCLOCK)	1
		1; Not supported	
		2; (=P0RXCLK) - All speeds	
		3; (=P1RXCLK) - All speeds	

4; (=P2RXCLK) - 50G / 25G / 10G
5; (=P3RXCLK) - 50G / 25G / 10G
6; (=P4RXCLK) - 25G / 10G
7; (=P5RXCLK) - 25G / 10G
8; (=P6RXCLK) - 25G / 10G
9 (=P7RXCLK) - 25G / 10G

Value = 1 is not supported and if so configured, the status will return: NOVALIDTXCLK.

You can only configure one of the available ports as clock source. The number of available ports depends on the selected speed mode. See Table 1 for valid ports depending on port speed.

Selecting a port which does not exist for the selected speed mode will cause the internally generated clock to be selected as Tx clock source.

Figure 6 illustrates how to configure SyncE in the UI (Select “Module → Resource Properties” tab).

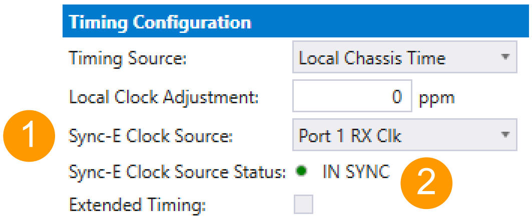


Figure 6: Configuring SyncE.

<u>To configure “SyncE”:</u>

1) Select the required clock source from the dropdown menu.
2) “IN SYNC” indicates if Chimera was able to lock to the selected input Rx port. (“IN SYNC” is not valid when selecting module local clock.)

The example above illustrates how to lock Tx output to the recovered clock from Rx port 1. Further the green light, indicates that the module was able to recover the configured clock.

<u>Script configuration example:</u>

The example below illustrates how to configure the same example using script commands.

```
M_TXCLOCKSOURCE P1RXCLK
```

The example below illustrates how to query if Chimera successfully locked to the configured Rx clock.

```
M_TXCLOCKSTATUS ?
OK
```

Notice, that the SyncE implementation described above, implies that Chimera is not Ethernet Synchronization Message Channel (ESMC) message aware and that all ESMC messages will pass transparently through Chimera if not explicitly configured for impairment using a flow filter.

3.2 Emulator bypass

It is possible to completely bypass the emulator core by directly connecting the input ports to the output ports for minimum latency. Setting the bypass mode, will affect all the ports of the module.

The emulator bypass mode is a convenient way of inactivating Chimera in the test setup, without physically removing the cables.

While in bypass mode, Chimera can be configured and statistics are updated, but this will have no effect on the output traffic. I.e. Chimera settings / statistics must be completely disregarded for the duration of the bypass.

The constant latency introduced by Chimera when in bypass mode, is listed in Table 5, for different port speeds and FEC settings. In addition to the constant latency listed in Table 5, Chimera will introduce a jitter of ± 50 ns.

	Port Speed						
	10G	25G	25G (FEC)	40G	50G	100G	100G (FEC)
Latency (ns)	1,150	600	1,250	1,000	550	600	1,400

Table 5: Chimera latency delay in emulator bypass mode.

Figure 7 illustrates how to configure emulator bypass mode in the UI (Select “Module → Resource Properties” tab).

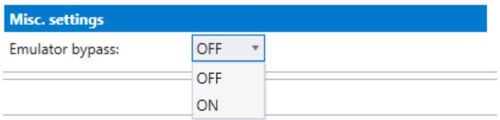


Figure 7: Configuring emulator bypass node.

<u>Script configuration example:</u>
The example below illustrates how to set the emulator bypass mode.

```
M_EMULBYPASS ON
```

4 Port settings

This section describes settings which will affect the entire port, i.e. it will affect all the flows defined for the selected port.

4.1 Reed-Solomon Forward Error Correction (RS-FEC)

Chimera ports support RS-FEC for 25G and 100G speeds.

To configure RS-FEC on a port, select the port in the UI and go to the “Resource Properties” → “PCS/PMA Config & Status” tab as illustrated in Figure 8 and click the “Enable RS-FEC” option.

(Note that “Link Training” and “Autoneg” is currently not supported for Chimera.)

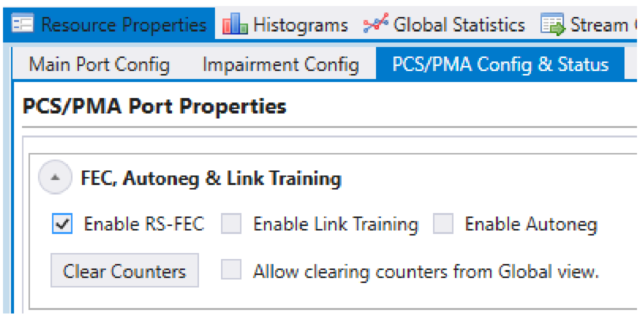


Figure 8: Enable RS-FEC in the UI.

Script configuration example:

The example below illustrates how to enable RS-FEC on a port.

```
PP_PHYAUTONEG 1 0 0 0 0
```

4.2 Test Payload (TPLD) size

The Xena Valkyrie traffic generators support inserting a Test Payload (TPLD) into the transmitted packets (see [Xena Test Payload](https://support.xenanetworks.com/hc/en-us/articles/115003944231-Xena-Test-Payload)[1]). The TPLD contains meta data, which can be used by the Xena receiving device to provide miscellaneous statistics.

When Chimera is connected to a Valkyrie traffic generator, Chimera can use the TPLD in the incoming packets for flow filtering (see section 7.5).

The TPLD supports 2 sizes:

- Default (20 bytes)
- Micro (6 bytes)

To use the TPLD for filtering in Chimera, it must be configured for the same TPLD format, as the transmitting Valkyrie traffic generator.

Figure 9 illustrates how to configure the TPLD size for a selected port.

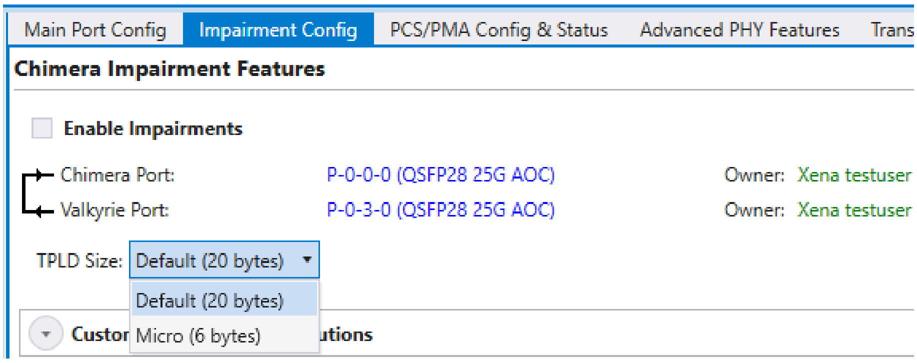


Figure 9: TPLD format configuration.

Notice that this setting is common to all flow filters on this port.

Script configuration example:

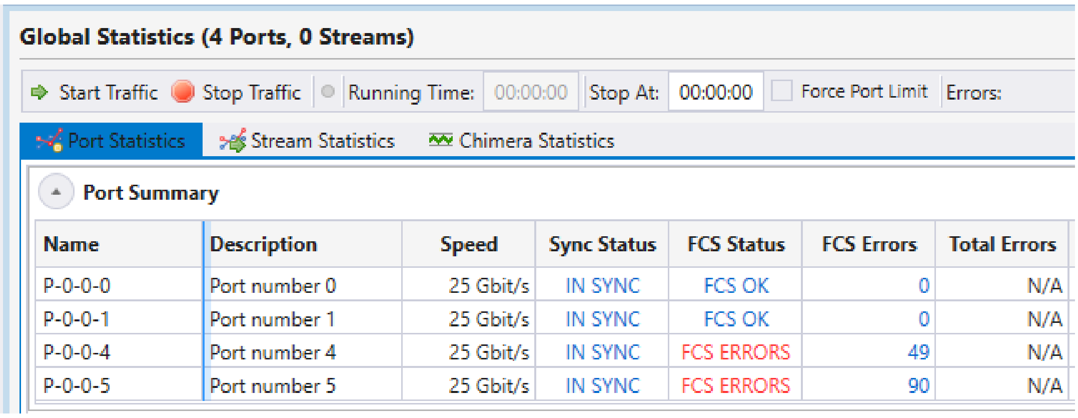
The example below illustrates how to configure the TPLD size on a port.

```
PE_TPLDMODE MICRO
```

4.3 FCS error mode

When packets with an FCS error is received on a Chimera port, they are counted by the port statistics as illustrated in Figure 1Figure 10.

[1] https://support.xenanetworks.com/hc/en-us/articles/115003944231-Xena-Test-Payload



Global Statistics (4 Ports, 0 Streams)

Start Traffic | Stop Traffic | Running Time: 00:00:00 | Stop At: 00:00:00 | Force Port Limit | Errors:

Port Statistics | Stream Statistics | Chimera Statistics

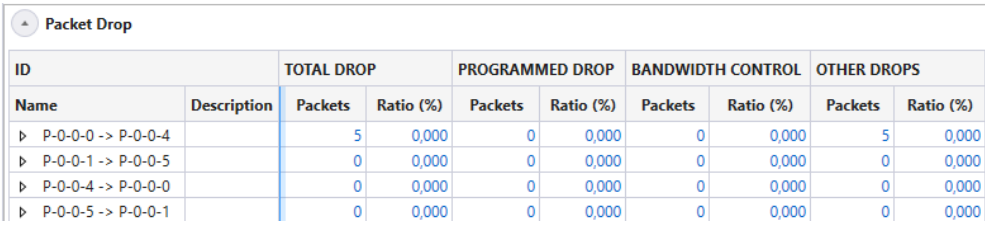
Port Summary

Name	Description	Speed	Sync Status	FCS Status	FCS Errors	Total Errors
P-0-0-0	Port number 0	25 Gbit/s	IN SYNC	FCS OK	0	N/A
P-0-0-1	Port number 1	25 Gbit/s	IN SYNC	FCS OK	0	N/A
P-0-0-4	Port number 4	25 Gbit/s	IN SYNC	FCS ERRORS	49	N/A
P-0-0-5	Port number 5	25 Gbit/s	IN SYNC	FCS ERRORS	90	N/A

Figure 10: Chimera FCS errors port statistics.

<u>Chimera supports two FCS error modes:</u>

- Pass mode.
  In this mode FCS errored packets are processed by Chimera as any other packet. I.e. the flow filter is a applied and the packet is subject to flow impairment and forwarded onto the output port.
- Discard mode.
  In this mode FCS errored packets are filtered by the flow filters and mapped to the corresponding impairment flow, where they are discarded and counted as "OTHER DROPS".



Packet Drop

ID		TOTAL DROP		PROGRAMMED DROP		BANDWIDTH CONTROL		OTHER DROPS	
Name	Description	Packets	Ratio (%)	Packets	Ratio (%)	Packets	Ratio (%)	Packets	Ratio (%)
P-0-0-0 -> P-0-0-4		5	0,000	0	0,000	0	0,000	5	0,000
P-0-0-1 -> P-0-0-5		0	0,000	0	0,000	0	0,000	0	0,000
P-0-0-4 -> P-0-0-0		0	0,000	0	0,000	0	0,000	0	0,000
P-0-0-5 -> P-0-0-1		0	0,000	0	0,000	0	0,000	0	0,000

Figure 11 illustrates how to configure the FCS error mode for a selected port.

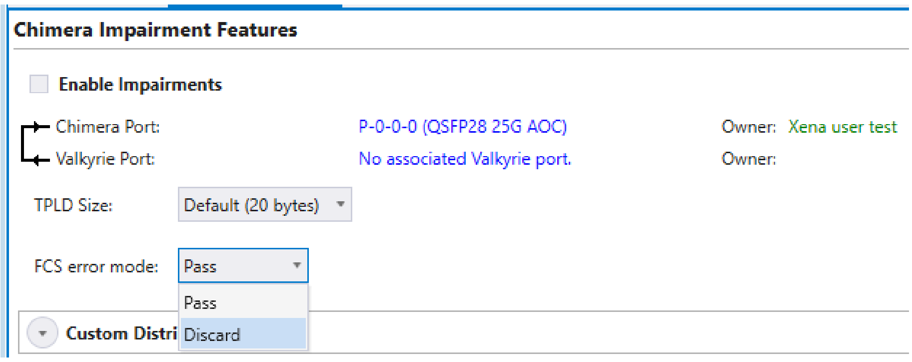


Figure 11: Chimera FCS errors port statistics.

<u>Script configuration example:</u>

The example below illustrates how to configure the FCS error mode on a port.

```
PE_FCSDROP ON
```

4.4 Link Flap

Chimera can be configured to emulate that the physical link is down or unstable. This feature is called "Link Flap". Link flap is implemented in 2 ways: "Logical Link Flap" and "Physical Link Flap".

Notice that link flap is configured at a port level and will affect all flows configured for the selected port.

Note that logical link flap and PMA error pulse inject (see section 4.5) are mutually exclusive.

4.4.1 Logical "Link Flap"

Logical link flap is implemented by scrambling the Tx PCS encoding to prevent the peer port from getting a link. I.e. it is not implemented by turning the physical transmitter on or off.

Logical link flap works for both electrical cables (DAC cables) and optical cables.

Logical link flap is configured under the "Main Port Config" tab as illustrated in Figure 12.

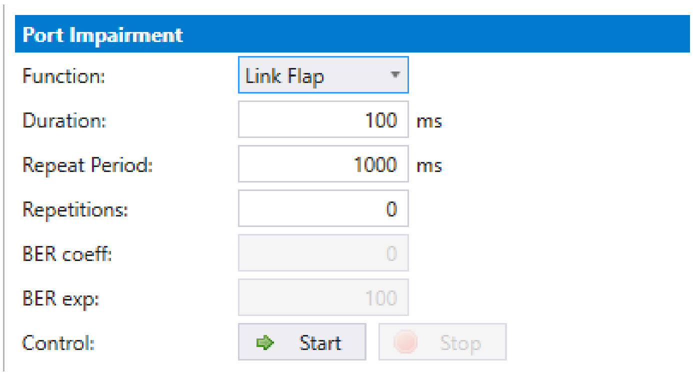


Figure 12: Configuration of "Logical Link Flap".

Logical link flap supports a repetitious pattern, where the link is taken down for a period ("Duration") and then brought up again. This is repeated after a configurable amount of time ("Repeat Period"). The flapping is repeated a configurable number of times or continuously ("Repetitions").

Pressing "Start" will start the configured link flap, pressing "Stop" will stop any ongoing link flapping.

Logical link flap is configured as follows:

Parameter	Description
Duration:	Duration of the link flap.
Repeat Period:	Period after which to restart link flap.
Repetitions	How many times to restart the link flap.

(For valid parameter ranges please refer to the script command description.)

Script configuration example:

The example below illustrates how to configure a link flap pattern, which will bring down the link for 120 ms and repeat this every 1.2 sec. This will be repeated 2346 times.

```
PP_LINKFLAP_PARAMS 120 1200 2346
PP_LINKFLAP_ENABLE 1
```

4.4.2 Optical "Link Flap".

To simulate the event of the optical link going down, it is possible to manually turn the optical transmitter off and on.

Optical link flap only works for optical cables, i.e. it will not work for e.g. DAC cables. Optical link flap does not support repetitious patterns as described above for logical link flap.

Optical link flap is configured on the “Main Port Config” tab as illustrated in Figure 13.

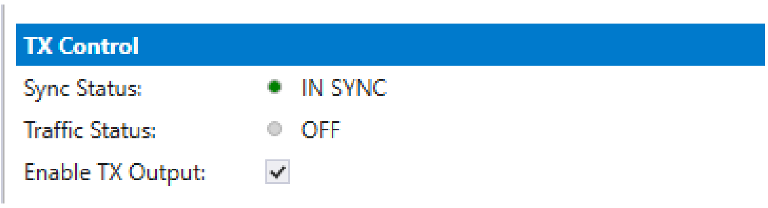


Figure 13: Configuration of “Optical Link Flap”.

Use “Enable Tx Output” to turn the optical transmitter off / on.

<u>Script configuration example:</u>
The example below illustrates how to turn the optical transmitter on and off.

```
P_TXENABLE OFF
P_TXENABLE ON
```

4.5 PMA error pulse injection

“PMA error pulse” allows the user to insert pulses of bit errors onto the link. If FEC is enabled, PMA errors are injected after the addition of the FEC bits, so that at the receiving end, FEC will correct as many of the PMA errors as possible.

Notice that PMA error pulse is configured at a port level and will affect all flows configured for that port. For BER insertion on a specific flow, see section 11.1.6.

Logical link flap (see section 4.4.1) and PMA error pulse inject are mutually exclusive.

PMA errors can be inserted with a fixed distance dependent on the selected port speed. The supported distances between two adjacent PMA errors and the corresponding BER for all speeds are listed in Table 6, where “n” is an integer number.

Speed	Supported PMA error distance	Supported PMA bit error rate
25G / 10G	n * 256 bits	0.39 % / n
50G	n * 512 bits	0.20 % / n
40G / 100G	n * 1024 bits	0.10 % / n

Table 6: Minimum distance between PMA errors.

When PMA pulse error injection is configured, the actual BER applied to the link is rounded to the value of ‘n’ which is closest to the configured value.

PMA error pulse injection is configured under the “Main Port Config” tab as illustrated in Figure 14.

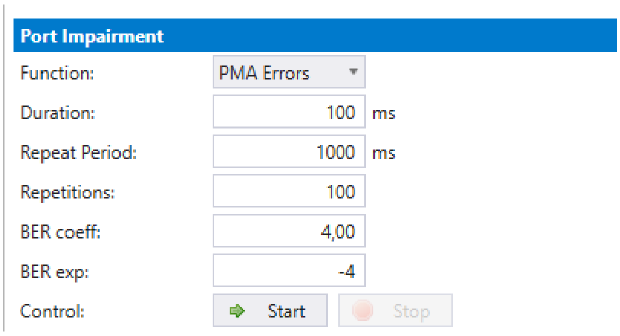


Figure 14: Chimera PMA error pulse injection.

It is possible to configure the length of the error pulse (“Duration”) and the BER during the pulse (“BER coeff” and “BER exp”). The burst is repeated after a programmable period (“Repeat Period”). The bursts will be repeated a configurable number of times (“Repetitions”).

Pressing “Start” will start the configured PMA error pulse, pressing “Stop” will stop any ongoing PMA error injection.

PMA error pulse inject is configured as follows:

Parameter	Legal values
Duration:	Duration of the PMA error pulse.
Repeat Period:	Period after which to restart the PMA error pulse.
BER Coeff	BER coefficient.
BER Exp	BER exponent.
Repetitions	How many times to restart the PMA error pulse.

(For valid parameter ranges, please refer to the script command description.)

The BER during error pulses is calculated as follows:

$$BER = \frac{coeff}{100} * 10^{exp}$$

Notice that the actual BER is rounded to the values listed in Table 6.

Script configuration example:
The example below illustrates how to configure a PMA error pulse inject pattern, which apply PMA errors for 430 ms and repeat this every 2.430 sec. BER = 2.34 * 10^{-12}. This will be repeated 2346 times.

```
PP_PMAERRPUL_PARAMS 430 2430 234 -12 2346
PP_PMAERRPUL_ENABLE 1
```

5 Chimera packet flows

When an Ethernet packet enters Chimera, it is assigned to a flow. Impairments are configured independently for every flow.

Every Rx port has a default flow. Additional flows can be added to the Rx port by configuring a corresponding flow filter. A maximum of 7 flow filters can be configured on every Rx port. Packets which match the flow filter, will be mapped to the corresponding flow. The flow filters are prioritized so packets are first matched against the filter of flow #7, subsequently flow filter #6 and finally filter #1.

Packets which do not match any flow filter, will be assigned to the port default flow (= flow #0)

This results in a total of maximum 8 flows for every port independent of port speed.

The Chimera flows as seen in ValkyrieManager are illustrated in Figure 15.

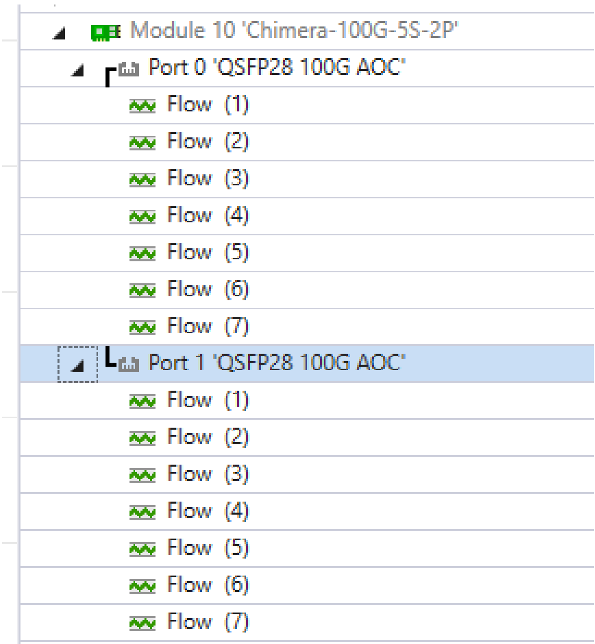


Figure 15: Chimera flow filters in Valkyrie Manager

Once the packet is mapped to a flow, it will pass through the associated impairment pipeline (see section 1.3). At the output of the impairment pipeline, packets from different flows will be merged into a common packet flow, which is transmitted to the output port.

This is illustrated in Figure 16.

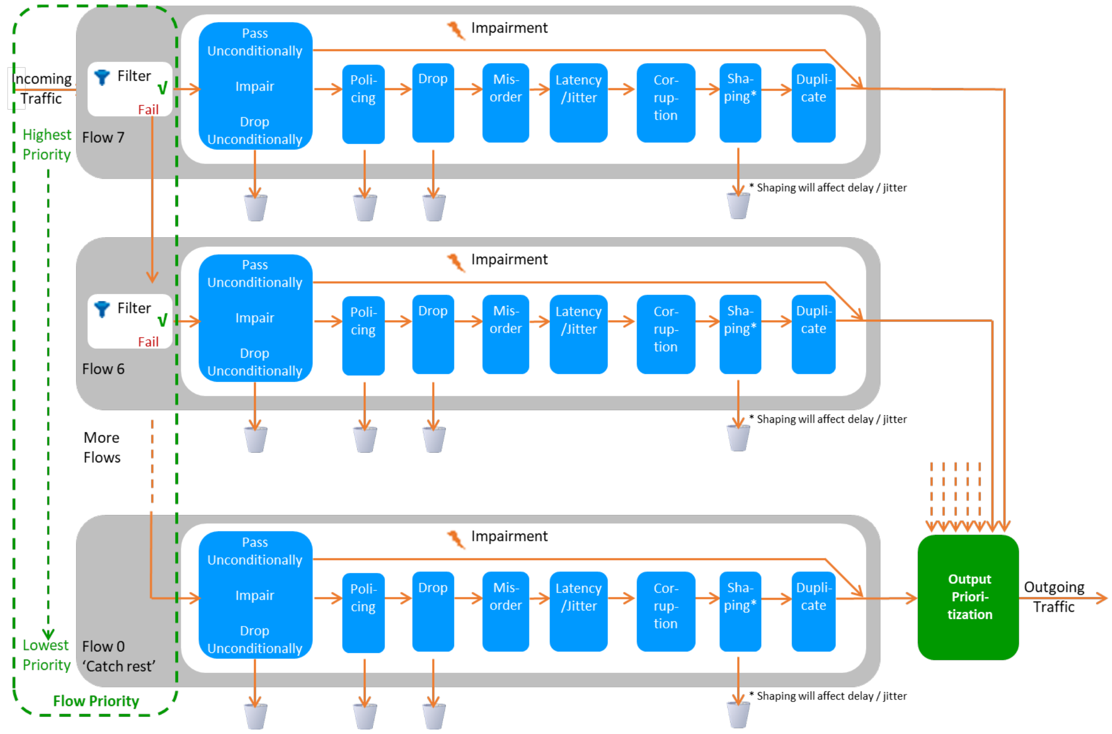


Figure 16: Chimera packet flow and impairments.

6 Configuring flow filters

As described in section 5, flows in Chimera are defined by the flow filters.

If a packet matches a given filter, the packet is mapped to the corresponding flow.

Notice, that modifying the number of active flow filters while traffic is running through Chimera, will result in packet drops, as described in section 2.3.

6.1 Updating flow filter registers

Flow filters can be updated during runtime with traffic applied to the input ports.

To guarantee that filtering is always coherent, Chimera implements two sets of registers in the flow filters:

- Working registers: used for flow filtering.
- Shadow registers: used for updating flow filters.

All registers in the flow filters have both a "working register" and a "shadow register".

Shadow registers can be written and read, while working registers can only be read.

Applying the script command "PEF_APPLY" will transfer all the shadow register values to the working registers instantaneously for all flow filter settings, including all sub-filters in basic mode (see section 7), so flow filters are always coherent. This allows updating the shadow registers, without the risk of using intermediate filtering values.

The following example illustrates how to specify filter and register type when accessing the flow filter registers using script commands.

```
PEF_ENABLE[fid,filter_type] ON
```

- *fid*: indicates the filter to configure (filter ID: 1 → 7)
- *filter_type*: indicates shadow (0) or working (1)

As described above, it is never legal to write to a register with filter_type = working (1).

6.2 Flow filtering modes

The flow filters can be configured in two different modes:

- Basic mode – a simple way of configuring a limited number of networking protocols.
- Extended mode – allows configuring of any networking protocol within the first 128 bytes.

Notice that “Extended mode” and “Basic mode” are mutually exclusive, since they are both using the same FPGA filters.

“Basic mode” is selected as default. To select “Extended mode” in the UI, click “Extended mode” as illustrated in Figure 17.

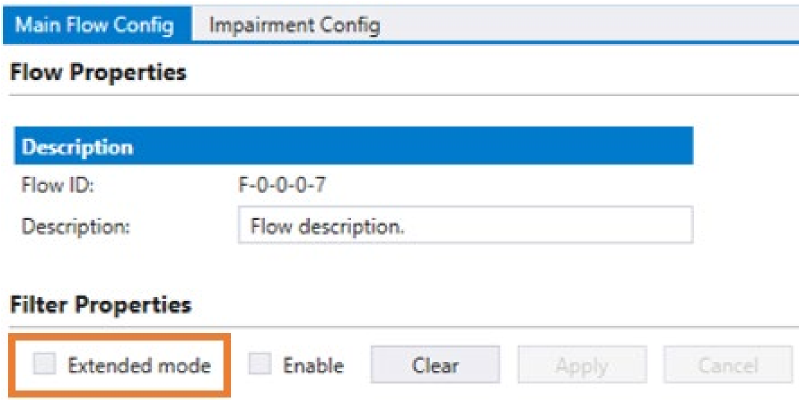


Figure 17: Configure flow filtering mode in UI.

Script configuration example:

The example below illustrates how to configure “Extended mode” flow filtering.

```
PEF_MODE [fid,0] EXTENDED
PEF_APPLY[fid,0]
```

Basic mode is described in section 7, while Extended mode is described section 8.

If you are new to networking protocols and encapsulation, you can start with “Basic Mode”, however if you have some experience with networking, it is recommended to always use “Extended Mode”.

7 Flow filters – Basic mode

In Basic mode, the flow filters are composed of multiple sub-filters, which match against different protocol layers. Sub-filters are named after the protocol layer at which they are applied.

The configuration options available in basic mode are illustrated in Figure 18.

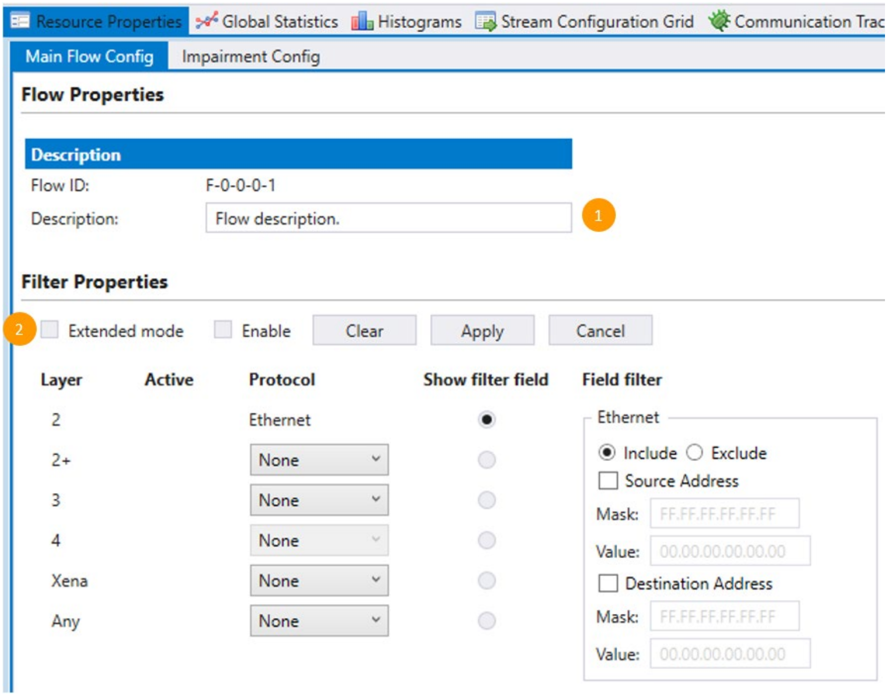


Figure 18: Flow sub-filters.

Figure 18 illustrates that it is possible to enter a flow description, which will be used in the UI to identify statistics (See Figure 18 1).

To enter basic mode you must de-select "Extended mode" (See Figure 18 2).

Script configuration example:

The example below illustrates how to configure "Basic mode" filtering.

```
PEF_MODE [fid,0] BASIC
PEF_APPLY[fid,0]
```

In Basic mode, the sub-filters are used to define the encapsulation of packets, even when they are not used for filtering.

E.g. defining "1 VLAN" at sub-filter 2+ (as illustrated in Figure 18) specifies that packets must have 1 VLAN, even when no fields in the VLAN are used for matching.

Flow filters can be defined on a port without enabling impairments. Simply configure the desired flow sub-filters, click "Enable" and then "Apply". If this is done, flow statistics will be updated, but no impairments are active.

To activate impairments at the flow level, the impairments must first be enabled at the port level (see Figure 38). If impairments are not enabled at the port level, all flow impairments are inactive.

All sub-filters allow specifying whether packets that match the corresponding sub-filter will be mapped to the flow ("Include" – option)or whether packets that do not match the sub-filter are mapped to the flow ("exclude" – option).

Basic mode implements the shadow- and working registers as described in section 6.1.

7.1 Ethernet sub-filter

A sub-filter can be applied at the Ethernet layer. The filter includes the following fields:

- Ethernet Destination MAC address (DMAC).
- Ethernet Source MAC address (SMAC).

The UI Ethernet sub-filter options are illustrated in Figure 19.

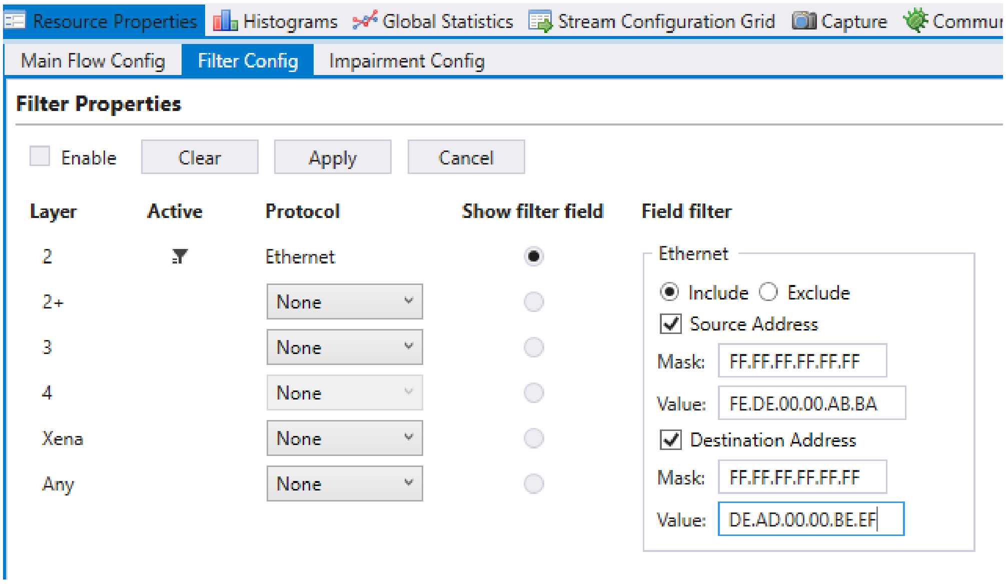


Figure 19: Ethernet sub-filter.

To match against SMAC or DMAC, check the boxes “Source Address” or “Destination Address” respectively.

If none of these checkboxes are checked, the sub-filter will not be used for matching. However, the remaining sub-filters will always assume that packets are Ethernet packets, including DMAC, SMAC and Ethertype.

When matching against DMAC / SMAC, a 48 bit mask is configured to identify which bits in the MAC address are used for matching. Filtering bits configured = 1 will be used for matching.

Script configuration example:

The example below illustrates how to filter for packets with DMAC = 0xFEDE0000ABBA and SMAC = 0xDEAD0000BEEF (all bits used for matching).

```
PEF_ETHSETTINGS[fid,0] AND INCLUDE
PEF_ETHSRCADDR [fid,0] ON 0xFEDE0000ABBA 0xffffffffffff
PEF_ETHDESTADDR[fid,0] ON 0xDEAD0000BEEF 0xffffffffffff
PEF_ENABLE     [fid,0] ON
PEF_APPLY      [fid]
```

7.2 Layer 2+ sub-filter

The Layer 2+ sub-filter allows the user to specify 1 VLAN, 2 VLANs or an MPLS label after the Ethernet header described in section 7.1.

7.2.1 1 VLAN Tag

This sub-filter allows filtering based on a single VLAN tag. The filter includes the following fields:

- VLAN ID (VID)
- VLAN PCP bits.

Selecting "1 VLAN Tag" causes the flow filter to verify that the TPID is 0x8100, in addition to any VID / PCP matching configured.

The available UI configuration options for "1 VLAN Tag" are illustrated in Figure 20.

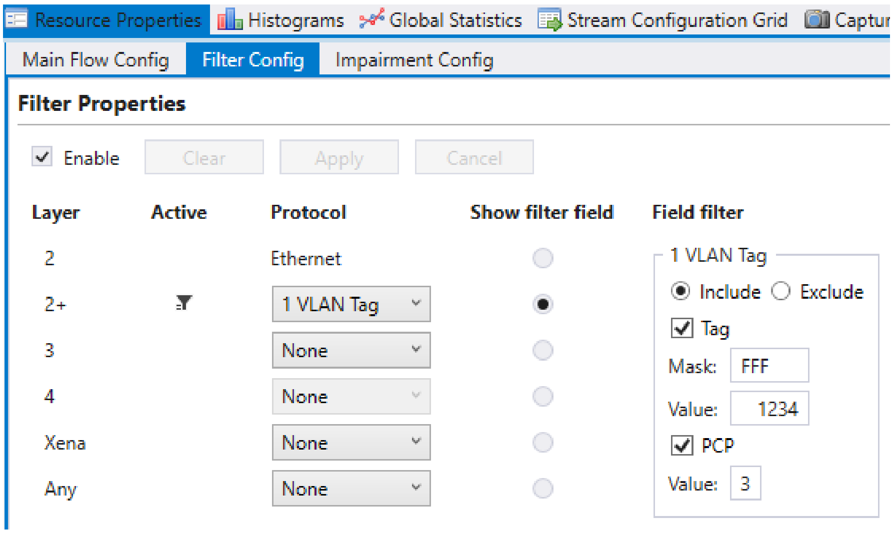


Figure 20: Layer 2+ sub-filter (1 VLAN)

To match against the VID, check the "Tag" checkbox and to match against the PCP bits, check the "PCP" checkbox.

The VID matching includes a mask to indicate that only selected bits (mask bit = 1) are used for matching. The PCP value is always matched against all 3 PCP bits in the UI.

If none of the checkboxes "Tag" or "PCP" are checked, no filtering is done on the VID / PCP values, but selecting the "1 VLAN Tag" option will indicate that packets mapped to this flow by higher layer sub-filters must have a single VLAN present. This includes checking TPID = 0x8100.

<u>Script configuration example:</u>
The example below illustrates how to filter packets with VID = 1234 and PCP = 3 (all bits used for matching).

```
PEF_L2PUSE        [fid,0]               VLAN1 (1)
PEF_VLANSETTINGS  [fid, 0]              AND (1) INCLUDE (1)
PEF_VLANTAG       [fid, 0, VLAN1 (0)]   ON (1)  1234  0xFFF
PEF_VLANPCP       [fid, 0, VLAN1 (0)]   ON (1)  3     0x7
PEF_ENABLE        [fid,0]               ON
PEF_APPLY         [fid]
```

7.2.2 2 VLAN Tags

This sub-filter allows filtering based on a 2 VLAN tags. The filter includes the following fields:

- Inner VLAN VID.
- Inner VLAN PCP bits.

- Outer VLAN VID.
- Outer VLAN PCP bits.

Selecting “2 VLAN Tags” causes the flow filter to verify that the TPID of the inner VLAN is 0x8100 and the TPID of the outer VLAN is 0x88A8, in addition to any VID / PCP checking configured.

The available UI configuration options for “2 VLAN Tags” are illustrated in Figure 21.

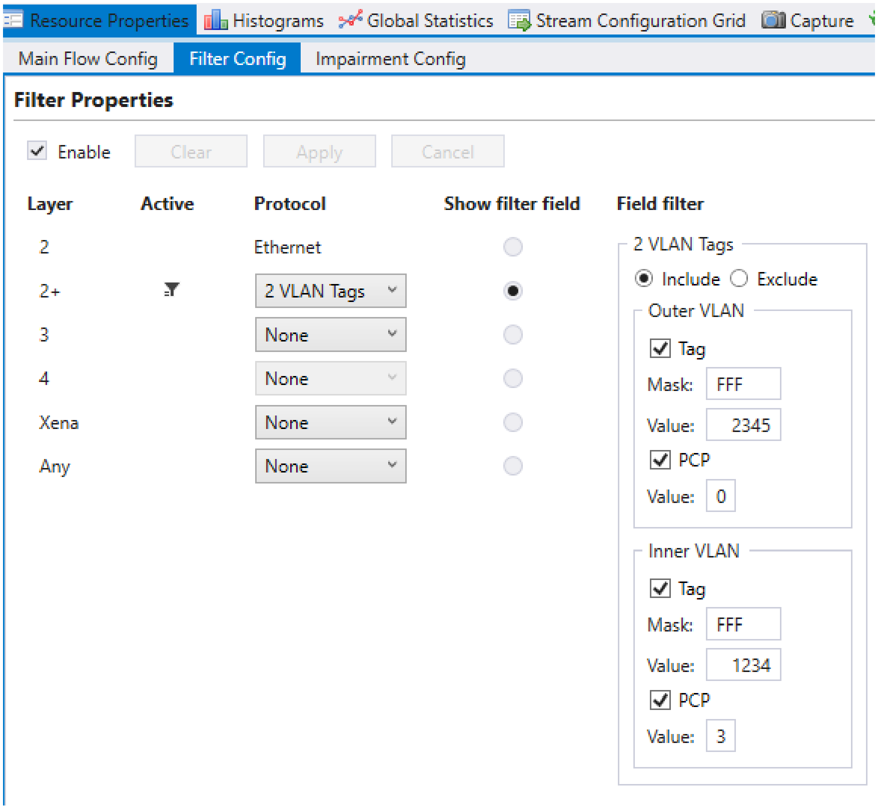


Figure 21: Layer 2+ sub-filter (2 VLANs)

To match against the inner and / or outer VID, check the corresponding “Tag” checkbox and to match against the inner and / or outer PCP bits, check the corresponding “PCP” checkbox.

The VID matching includes a mask to indicate that only selected bits (mask bit = 1) are used for matching. The PCP value is always matched against all 3 PCP bits in the UI.

If none of the checkboxes “Tag” or “PCP” are checked, no filtering is done on the VID / PCP values, but selecting the “2 VLAN Tags” option will indicate that packets mapped to this flow by higher layer sub-filters must have two VLANs tags present. This includes the TPID matching described above.

<u>Script configuration example:</u>
The example below illustrates how to filter for packets with outer VID = 2345, outer PCP = 0, inner VID = 1234 and inner PCP = 3 (all bits used for matching).

```
PEF_L2PUSE          [fid,0]                   VLAN2 (2)
PEF_VLANSETTINGS    [fid, 0]                  AND (1)  INCLUDE (1)
PEF_VLANTAG         [fid, 0, VLAN2 (1)]       ON (1)   2345 0xFFF
PEF_VLANPCP         [fid, 0, VLAN2 (1)]       ON (1)   0    0x7
PEF_VLANTAG         [fid, 0, VLAN1 (0)]       ON (1)   1234 0xFFF
PEF_VLANPCP         [fid, 0, VLAN1 (0)]       ON (1)   3    0x7
PEF_ENABLE          [fid,0]                   ON
PEF_APPLY           [fid]
```

7.2.3 MPLS

This sub-filter allows filtering based on a MPLS label. The filter includes the following fields:

- MPLS label.
- MPLS traffic class (TC) bits.

The available UI configuration options for MPLS are illustrated in Figure 22.

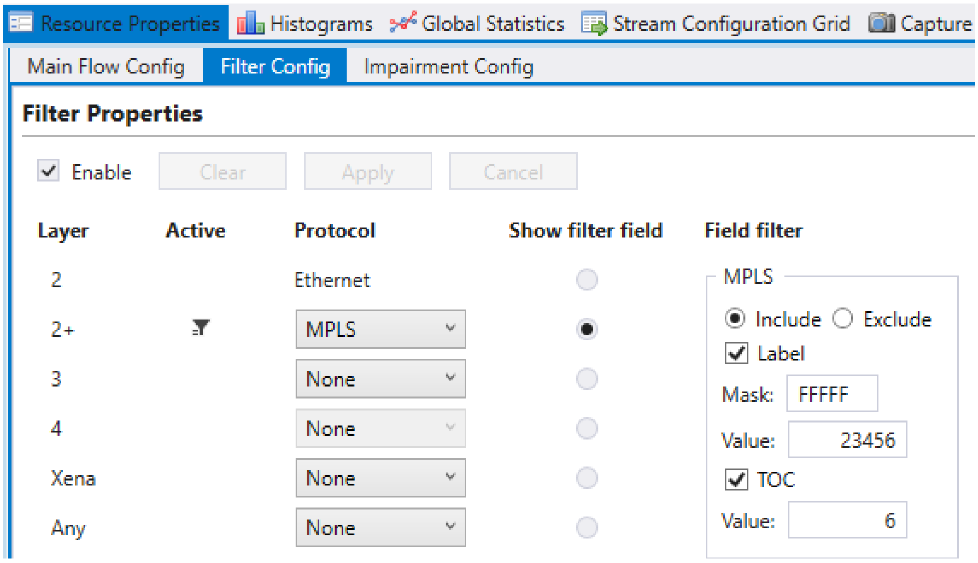


Figure 22: Layer 2+ sub-filter (MPLS)

To match against the MPLS label, check the corresponding "Label" checkbox and to match against the Traffic Class (TC) bits, check the "TOC" checkbox.

The label matching includes a mask to indicate that only selected bits (mask bit = 1) are used for matching. The TC value is always matched against all 3 TC bits in the UI.

If none of the checkboxes "Label" or "Exp/ToC" are checked, no filtering is done on the label / TC values, but selecting the "MPLS" option will indicate that packets mapped to this flow by higher layer sub-filters must have a MPLS label (32 bits) present.

<u>Script configuration example:</u>
The example below illustrates how to filter for packets with label = 23456 and TC = 6 (all bits used for matching).

```
PEF_L2PUSE         [fid,0] MPLS (3)
PEF_MPLSSETTINGS   [fid,0] AND (1) INCLUDE (1)
PEF_MPLSLABEL      [fid,0] ON (1) 23456 0xFFFFF
PEF_MPLSTOC        [fid,0] ON (1) 6 0x07
PEF_ENABLE         [fid,0] ON
PEF_APPLY          [fid]
```

7.3 Layer 3 sub-filter

The Layer 3 sub-filter allows the user to specify an IPv4 or an IPv6 header after the Ethernet header described in section 7.1 and the layer 2+ encapsulation described in section 7.2.

There is no explicit selection between the IPv4 and the IPv6 filtering in the script commands, but if both are configured, only the later will be active.

7.3.1 IPv4

This sub-filter allows filtering based on the following IPv4 fields:

- Destination IP address (DIP)
- Source IP address (SIP)
- IPv4 DSCP

The available UI configuration options for IPv4 are illustrated in Figure 23.

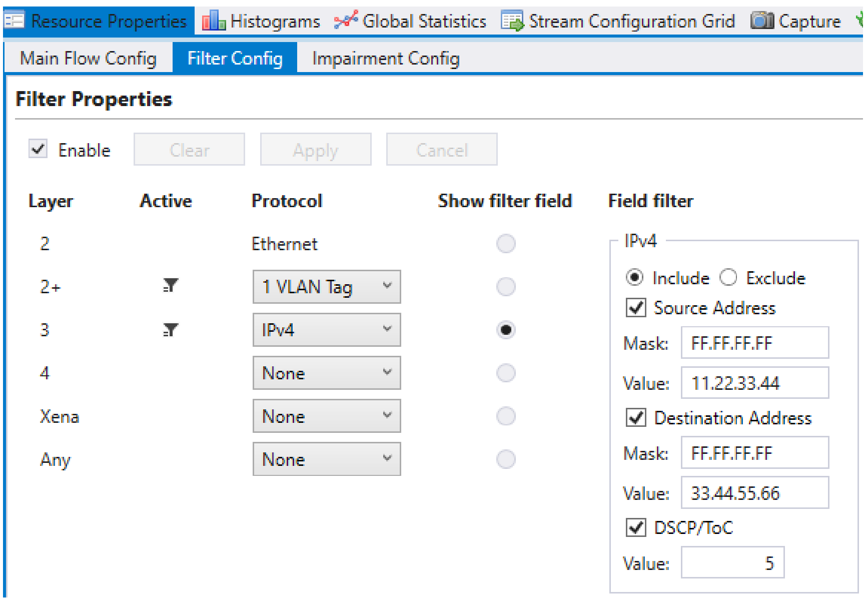


Figure 23: Layer 3 sub-filter (IPv4)

To match against IPv4 SIP, DIP or “DSCP/ToC”, check the corresponding checkboxes.

The DIP and SIP matching includes a mask to indicate that only selected bits (mask bit = 1) are used for matching. The DSCP value is always matched against all 6 bits in the UI.

If noneof the checkboxes “SIP”, “DIP” or “DSCP/ToC” are checked, no filtering is done at the IPv4 layer, but selecting the IPv4 option will indicate that packets mapped to this flow by higher layer sub-filters must have a IPv4 header present.

Script configuration example:

The example below illustrates how to filter for packets with SIP = 11.22.33.44, DIP = 33.44.55.66 and DSCP = 5 (all bits used for matching).

```
PEF_L3USE          [fid,0] IP4 (1)
PEF_IPV4SETTINGS   [fid,0] ON (1) INCLUDE (1)
PEF_IPV4SRCADDR    [fid,0] ON (1) 11.22.33.44 0xFFFFFFFF
PEF_IPV4DESTADDR   [fid,0] ON (1) 33.44.55.66 0xFFFFFFFF
PEF_IPV4DSCP       [fid,0] ON (1) 0x14 0xFC[2]
PEF_ENABLE         [fid,0] ON
PEF_APPLY          [fid]
```

7.3.2 IPv6

This sub-filter allows filtering based on the following IPv6 fields:

- Destination IP address (DIP)
- Source IP address (SIP)
- IPv6 DSCP

The available UI configuration options for IPv6 are illustrated in Figure 24.

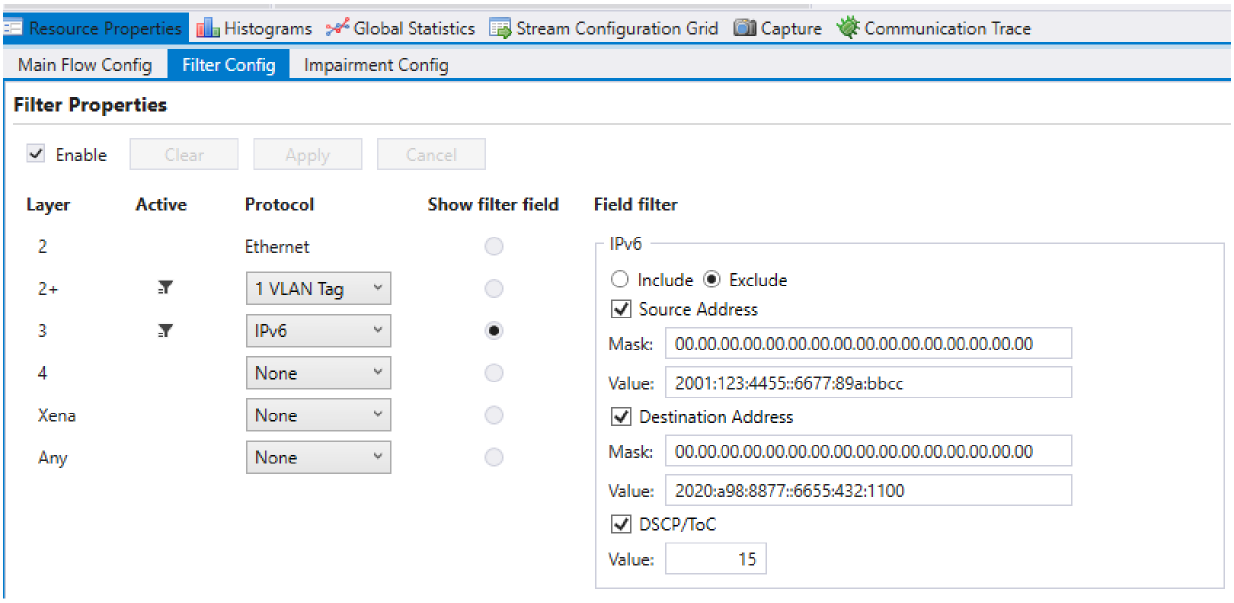


Figure 24: Layer 3 sub-filter (IPv6)

To match against IPv6 SIP, DIP or DSCP/ToC, check the corresponding checkboxes.

The DIP and SIP matching includes a mask to indicate that only selected bits (mask bit = 1) are used for matching. The DSCP value is always matched against all 6 bits in the UI.

If none of the checkboxes SIP, DIP or DSCP are checked, no filtering is done at the IPv6 layer, but selecting the IPv6 option will indicate that packets mapped to this flow by higher layer sub-filters must have a IPv6 header present.

[2] For details on how to configure DSCP value and mask, please refer to script command documentation.

Script configuration example:

The example below illustrates how to filter for packets with SIP = 2001:123:4455::6677:89A:BBCC, DIP = 2020:a98:8877::6655:432:1100 and DSCP = 15 (all bits used for matching).

```
PEF_L3USE         [fid,0] IP6 (2)
PEF_IPV6SETTINGS  [fid,0] ON (1) INCLUDE (1)
PEF_IPV6SRCADDR   [fid,0] ON (1)
                  0x200101234455000000006677089ABBCC
                  0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
PEF_IPV6DESTADDR  [fid,0] ON (1)
                  0x20200a98887700000000665504321100
                  0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
PEF_IPV6TC        [fid,0] ON (1) 0x3C 0xFC3
PEF_ENABLE        [fid,0] ON
PEF_APPLY         [fid]
```

7.4 Layer 4 sub-filter

The Layer 4 sub-filter allows the user to specify a UDP or a TCP header after the Ethernet header described in section 7.1, the layer 2+ encapsulation described in section 7.2 and layer 3 sub-filtering described in section 7.3.

Notice that the UDP / TCP port configuration is only available if a IPv4/IPv6 header was configured at layer 3.

There is no explicit selection between the UDP port and the TCP port filtering in the script commands, but if both are configured, only the later will be active.

7.4.1 TCP

This sub-filter allows filtering based on the following TCP fields:

- TCP source port.
- TCP destination port.

The available UI configuration options for TCP filtering are illustrated in Figure 25.

[3] For details on how to configure DSCP value and mask, please refer to script command documentation.

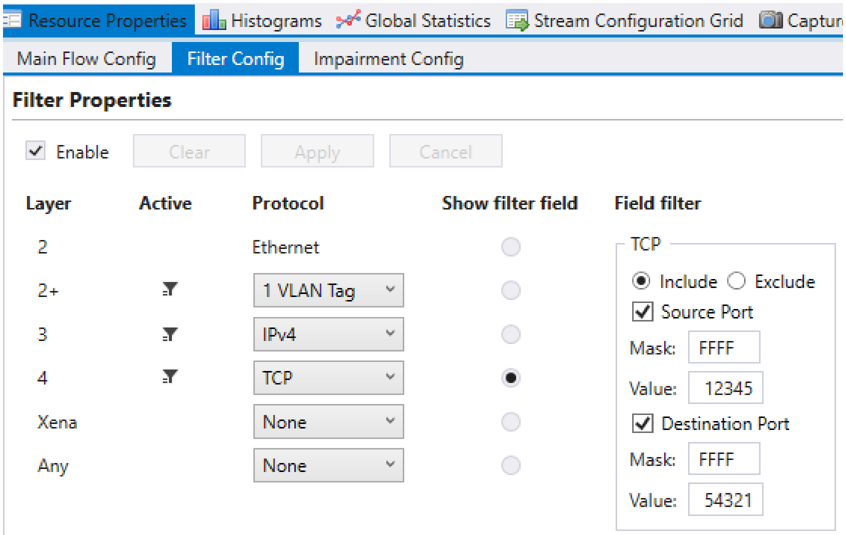


Figure 25: Layer 4 sub-filter (TCP).

To match against the TCP source port or the destination port, check the corresponding checkboxes.

Both the TCP source port and destination port matching includes a mask to indicate that only selected bits (mask bit = 1) are used for matching.

If none of the checkboxes “Source Port” or “Destination Port” are checked, no filtering is done at the TCP layer, but selecting the “TCP” option will indicate that packets mapped to this flow by other sub-filters must have a TCP header present.

<u>Script configuration example:</u>

The example below illustrates how to filter for packets with TCP source port = 12345 and destination port = 54321 (all bits used for matching).

```
PEF_TCPSETTINGS [fid,0] ON (1) INCLUDE (1)
PEF_TCPSRCPORT  [fid,0] ON (1) 12345 0xFFFF
PEF_TCPDESTPORT [fid,0] ON (1) 54321 0xFFFF
PEF_ENABLE      [fid,0] ON (1)
PEF_APPLY       [fid]
```

7.4.2 UDP

This sub-filter allows filtering based on the following UDP fields:

- UDP source port.
- UDP destination port.

The available UI configuration options for UDP filtering are illustrated in Figure 26.

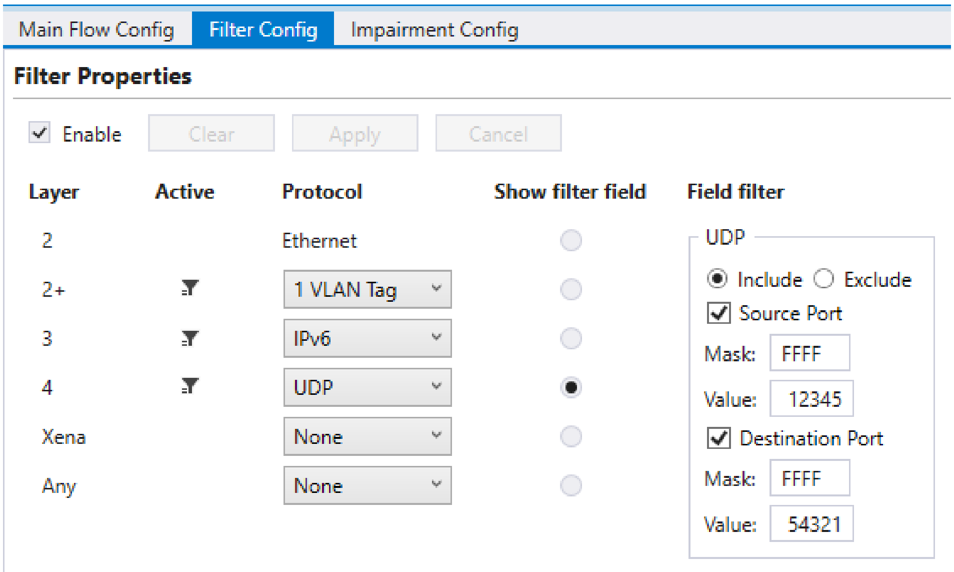


Figure 26: Layer 4 sub-filter (UDP)

To match against the UDP source port or the destination port, check the corresponding checkboxes.

Both the UDP source port and destination port matching include a mask to indicate that only selected bits (mask bit = 1) are used for matching.

If none of the checkboxes “Source Port” or “Destination Port” are checked, no filtering is done at the UDP layer, but selecting the UDP option will indicate that packets mapped to this flow by higher layer sub-filters must have a UDP header present.

Script configuration example:
The example below illustrates how to filter for packets with UCP source port = 12345 and destination port = 54321 (all bits used for matching).

```
PEF_UDPSETTINGS [fid,0] ON (1) INCLUDE (1)
PEF_UDPSRCPORT  [fid,0] ON (1) 12345 0xFFFF
PEF_UDPDESTPORT [fid,0] ON (1) 54321 0xFFFF
PEF_ENABLE      [fid,0] ON (1)
PEF_APPLY       [fid]
```

7.5 TPLD sub-filter

When using a Valkyrie traffic generator, it is possible to insert a Test Payload (TPLD) into the Tx packets. The TPLD includes a Test Identifier (TID), which can be used for flow filtering in Chimera. Notice that the configured Chimera TPLD size must be the same as the TPLD size configured on the traffic generator (see section 4.2).

The flow filters support matching against 16 TPLD TID values.

The available UI configuration options for TPLD TID filtering are illustrated in Figure 27.

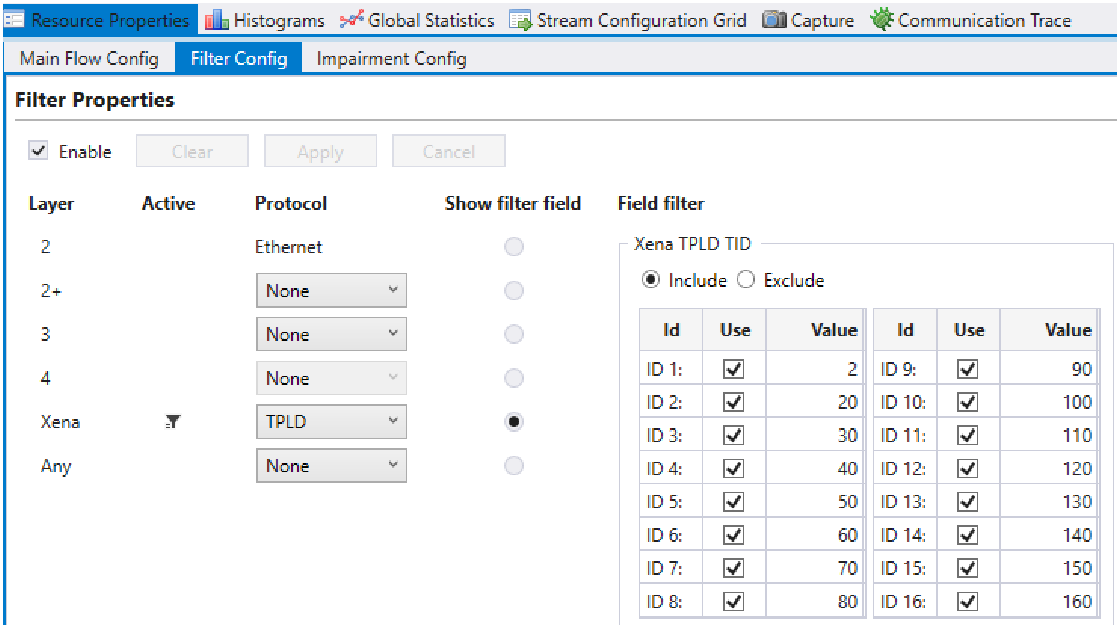


Figure 27: TLPD sub-filter.

To filter based on the TPLD TID, check the “Use” checkbox and fill in the required TID value. Valid values for the TPLD TID are 0 → 2015 for default size (1023 for Micro) and must match what is configured in the Valkyrie traffic generator.

<u>Script configuration example:</u>

Configuring multiple TPLD TID values is done by configuring a TID value for multiple indices (Index 0 – 15). The example below illustrates how to filter for packets with TPLD TPID = 987 (index = 5).

```
PEF_TPLDSETTINGS [fid,0]   AND (1) INCLUDE (1)
PEF_TPLDCONFIG   [fid,0,5] ON (1) 987
PEF_ENABLE       [fid,0]   ON (1)
PEF_APPLY        [fid]
```

7.6 Any field sub-filter

This filter can match 6 consecutive bytes at a configurable offset within the incoming packets. Furthermore, there is a mask to indicate which bits are to be used for matching.

Notice that in addition to the byte match configured here, the packet must match any encapsulation defined in sections 7.1 to 7.4.

The available UI configuration options for “Any Field” filtering are illustrated in Figure 28.

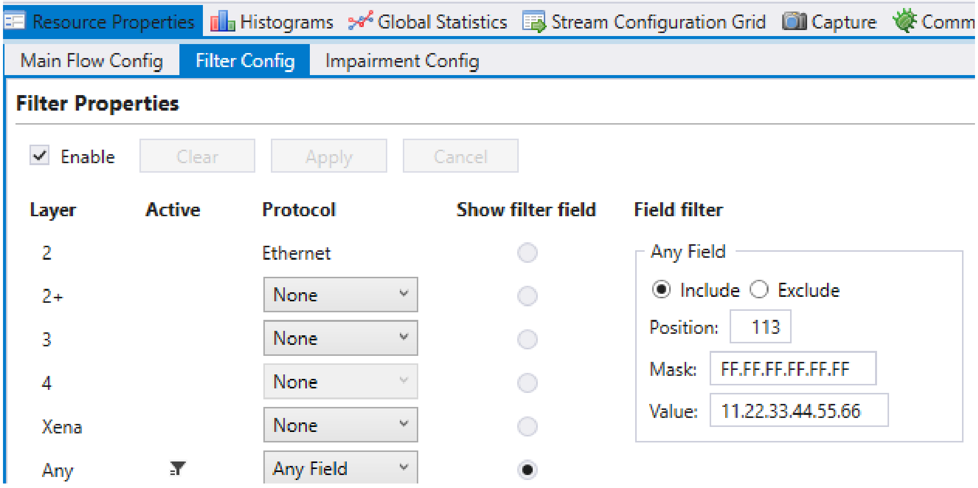


Figure 28: “Any Field” filtering.

Parameter	Legal values.	Comments	Step size
Position:	0 → 122	Byte offset in packet	1

Script configuration example:

The example below illustrates how to filter for packets including a 6 bytes value of 0x112233445566 starting at a byte offset of 113 (All bits used for matching).

```
PEF_ANYSETTINGS [fid,0] AND (1) INCLUDE (1)
PEF_ANYCONFIG   [fid,0] 113 0x112233445566 0xFFFFFFFFFFFF
PEF_ENABLE      [fid,0] ON (1)
PEF_APPLY       [fid]
```

8 Flow filters – Extended mode

Extended filtering mode allows the user to filter on any pattern within the first 128 bytes of the packet.

The filtering is done by specifying a “filter value” of 128 bytes and “filter mask” of 128 bytes.

The extended filter is illustrated in Figure 29.

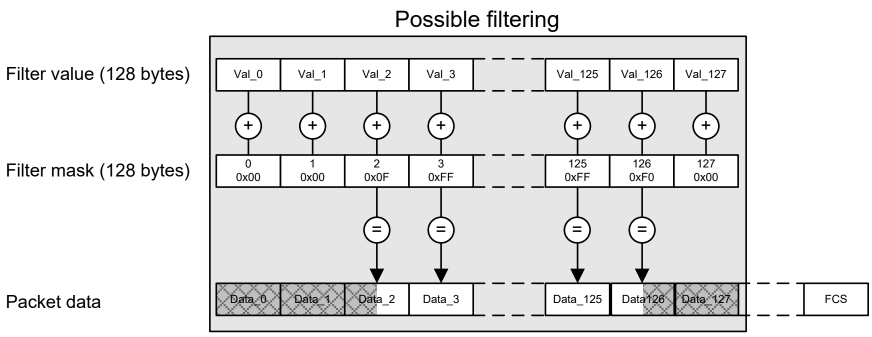


Figure 29: Extended Filtering mode.

Figure 29 illustrates that if the filter mask of a given byte is non-zero (bytes 2, 3, 125 and 126), the corresponding filter value byte is matched against the corresponding byte in the incoming packet at the bit positions indicated by a ‘1’ in the mask bit.

If the mask byte is zero (bytes 0, 1, 127), the corresponding byte in the incoming packet is ignored in the flow filter.

If an incoming packet is shorter that the specified filter, it will not match.

8.1 UI configuration

To configure a flow filter using extended filtering, select the relevant flow and go to the tab “Resource Properties” → “Main Flow Config” and select “Extended mode”, which will bring up the UI interface illustrated in Figure 30.

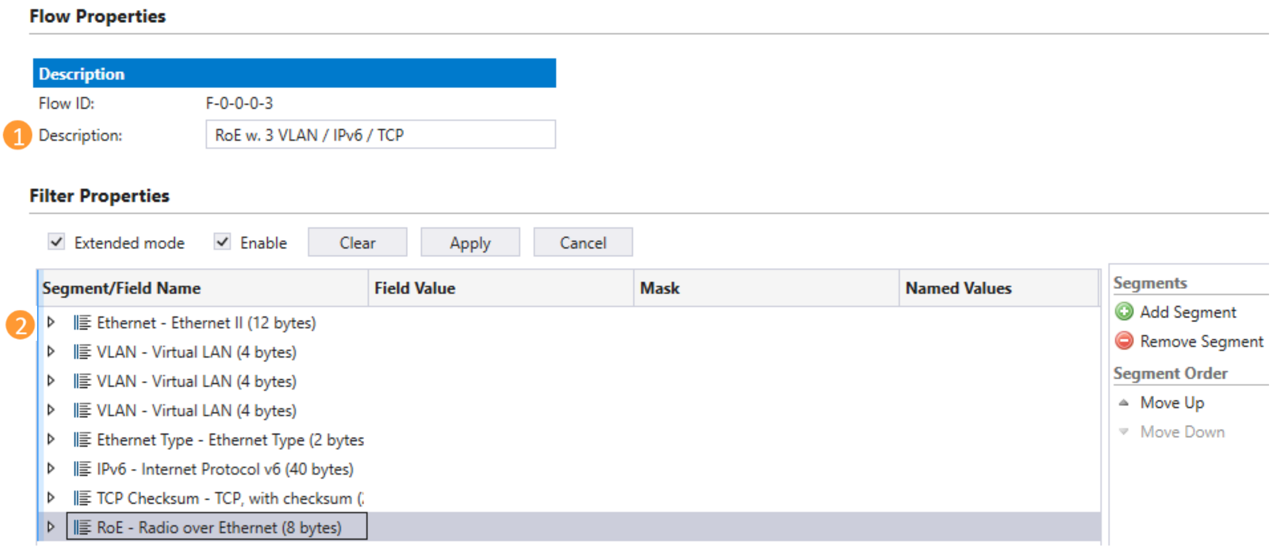


Figure 30: Extended filtering.

At the top it is possible to assign a descriptive text to the flow (1), which is used to identify the flow in the statistics tabs.

The protocol segment list illustrated in Figure 30 (2) can be manipuated using the “Add segment” and “Remove segment” buttons. To add a protocol layer use the “Add Segment” button and select relevant protocols from the predefined protocol list illustrated in Figure 31.

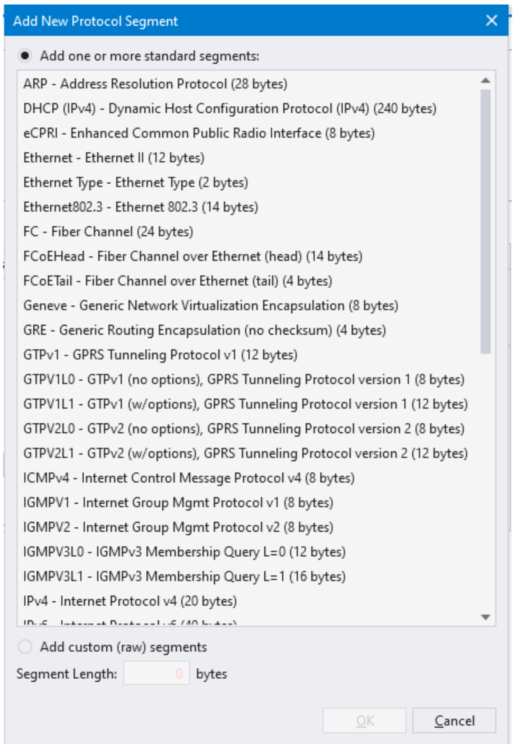
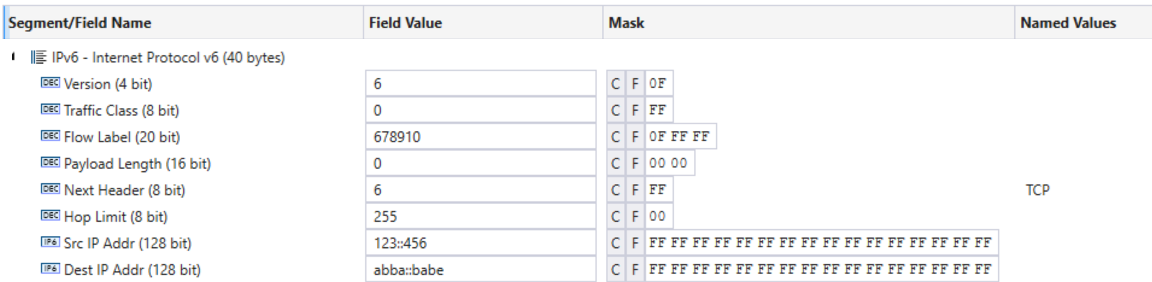


Figure 31: Extended filtering protocols.

To add custom protocol segments use “Add custom (raw) segments” at the bottom of Figure 31, by simply supplying the length of the segment in bytes. Notice that the total sum of the protocols can not exceed 128 bytes.

For each of the selected protocols, it is configurable which bits in the packet to match and which values to match against. An example of how to configure the IPv6 encapsulation is illustrated in Figure 32.



Segment/Field Name	Field Value	Mask	Named Values
IPv6 - Internet Protocol v6 (40 bytes)			
Version (4 bit)	6	C F 0F	
Traffic Class (8 bit)	0	C F FF	
Flow Label (20 bit)	678910	C F 0F FF FF	
Payload Length (16 bit)	0	C F 00 00	
Next Header (8 bit)	6	C F FF	TCP
Hop Limit (8 bit)	255	C F 00	
Src IP Addr (128 bit)	123::456	C F FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	
Dest IP Addr (128 bit)	abba::babe	C F FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	

Figure 32: IPv6 filter configuration.

Once the filter is complete, be sure to click “Enable” and press the “Apply” button, illustrated in Figure 30.

For info on how to configure extended filtering using scripting, see section 8.2.

8.2 Scripting

The bytes in the filter value and filter mask are structured as a list of protocols. Each protocol requires a certain number of bytes to be specified in the filter value and filter mask. Each protocol in the protocol list is identified using a protocol index, reflecting the number of the protocol in the list. The first protocol will be assigned protocol index = 1.

Figure 33 illustrates an example of a protocol list including the protocols names, protocol indices (in parenthesis) and the length of each protocol in bytes.

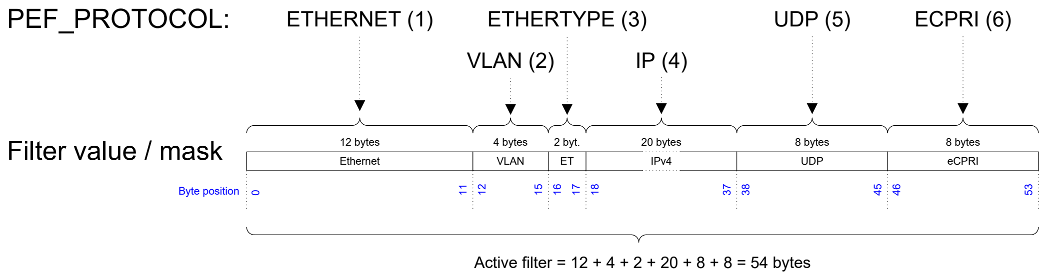


Figure 33: Flow filter protocol specification.

The protocol list shown here contains the protocols listed in Table 7.

Protocol	Protocol index	Length (bytes)
ETHERNET	1	12
VLAN	2	4
ETHERTYPE	3	2
IPv4	4	20
UDP	5	8
eCPRI	6	8

Table 7: Example of a protocol list.

The filter protocol list is specified using the script command PEF_PROTOCOL and takes as argument all the protocols to be included in the current filter. Notice, that the protocol list must include ETHERNET at protocol index = 1, to indicate that the incoming packets are Ethernet packets. In addition to the predefined protocols, it is possible to define “custom protocols” [4].

Script configuration example:

The example below illustrates how to select extended filtering mode and configure the protocol list illustrated in Figure 33.

```
PEF_MODE      [fid,0] EXTENDED
PEF_PROTOCOL[fid,0] ETHERNET VLAN ETHERTYPE IP UDP ECPRI
PEF_APPLY     [fid]
```

The combined length of the protocols configured using PEF_PROTOCOL defines the length of the active filter in bytes. Referring to the example from Figure 33, this amounts to 54 active filter bytes. This implies that the first 54 bytes of the filter value and the filter mask must be configured. The remaining filter mask bytes will automatically be set to zero. Notice that the active filter can never be configured to be more than 128 bytes.

Once the protocol list has been defined, the protocol index implicitly assigned to every protocol, can be used to set the protocol filter value and protocol filter mask, using PEF_VALUE and PEF_MASK respectively[4].

[4] For further details on script commands please refer to “Xena script commands for Chimera”.

Using the protocol index as input to PEF_VALUE and PEF_MASK only the value of that protocol is modified. The protocol value must reflect the byte values in the protocol being referenced. Figure 34 illustrates the configuration of the 8 bytes in the UDP protocol.

UDP source port		UDP destination port		UDP length		UDP checksum	
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

Figure 34: Configuring UDP protocol segment.

<u>Script configuration example:</u>

The example below illustrates how to configure the UDP (protocol index = 5) protocol filtering from the example in Figure 33 with “UDP source Port = 0x1122” and “UDP length = 0x3344”, while “UDP destination port” and “UDP checksum” are ignored.

```
PEF_VALUE[fid,0,5] 0x1122000033440000
PEF_MASK [fid,0,5] 0xFFFF0000FFFF0000
PEF_APPLY[fid]
```

Notice that the filter values in red are not used for filtering, due to the filter mask bit = ‘0’.

Protocol index 0 has a special significance. It is used to work on the entire active filter value and filter mask as a single array of bytes.

Once the protocol list has been defined, you can use index 0 to define the entire filter value / mask or use the individual protocol indices to configure the protocols individually. I.e. using protocol index 0 for the example in Figure 33 will modify all 54 bytes of the active filter value / mask.

9 Impairment configuration overview

The impairments currently supported in Chimera are:

- Ingress policing
- Packet drop
- Misordering
- Packet corruption (FCS/IP/TCP/UDP)
- Packet duplication
- Latency / jitter
- Egress shaper

Chimera supports a variety of distributions that can be used to apply the supported impairments. The distribution for a given impairment determines how often the impairment is applied to the flow in terms of time or number of packets between the impairments.

The complete set of distributions and which impairments they support is described in detail in section 11.

Furthermore, the impairments can be turned on and off automatically using a scheduler. The scheduler allows the impairment to be applied in 2 modes:

- Continuous[5]

[5] Notice that “Accumulate and burst” and “Random Burst” are not continuous by nature and implement a modified scheduler function. See section 12 for details.

- Repeated Pattern

When configured for “Continuous Mode”, the impairment will be applied continuously to the flow until turned off by the user.

When configured for “Repeated Pattern Mode”, the scheduler allows specifying a “Duration” and a “Repeat period”. The scheduler will (re-)start the impairment at intervals equal to the configured repeat period. When started, the impairment will be on for a period equal to the configured duration after which it is turned off. This pattern is repeated until the impairment is turned off by the user.

The repeat pattern configuration is illustrated in Figure 35.

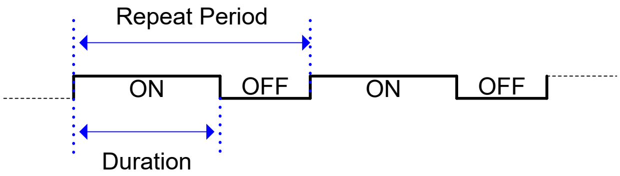


Figure 35: Scheduler ON / OFF function.

The details of the scheduling function are described in section 12.

9.1 Integration with Valkyrie traffic generators.

When using Chimera together with a Valkyrie traffic generator, the UI supports visually associating the connected Valkyrie and Chimera ports for better overview and ease of configuration.

Figure 36 illustrates how to associate two Valkyrie and Chimera ports.

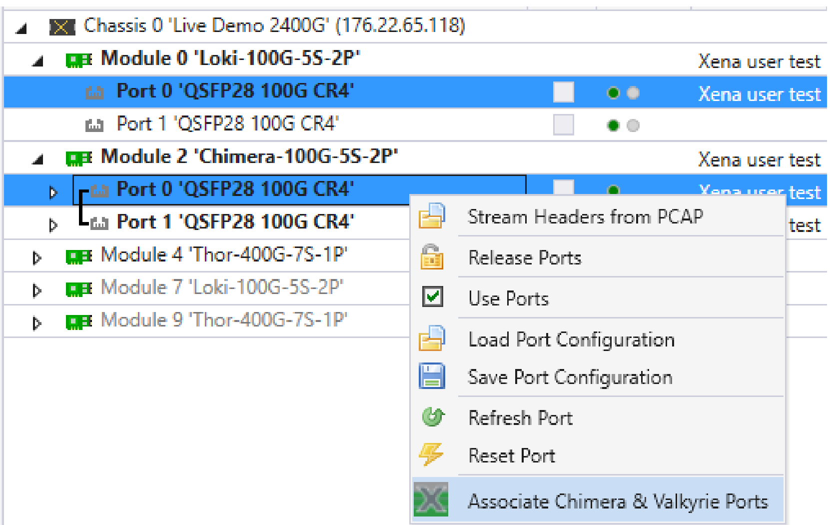


Figure 36: Associating the Valkyrie and Chimera ports in the UI.

<u>To associate Valkyrie and Chimera ports:</u>

1) Select the two ports to be associated using the <CTRL> button.
2) Right click with the mouse on one of the ports.
3) Select “Associate Chimera & Valkyrie Ports”.

Selecting the Valkyrie port in the UI, you can now see the associated Chimera port and access the Chimera port impairment configuration. This is illustrated in Figure 37.

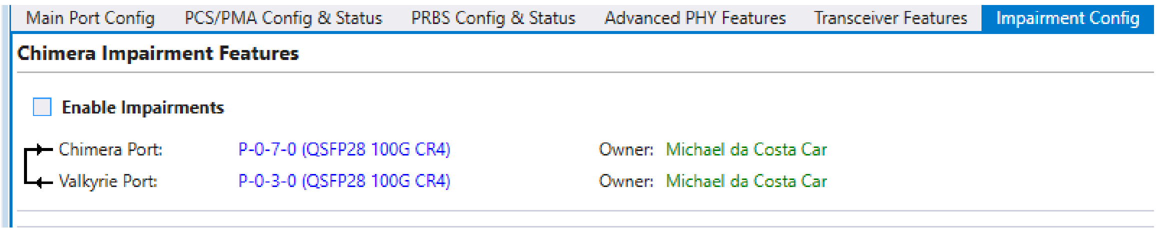


Figure 37: Configuring the Chimera port impairment from the Valkyrie port.

Notice that the port association is an UI property only. It is not available with script commands.

The following sub-sections describe the configuration of impairments using script commands. Users that solely use the UI can go to section 10.

9.2 Script configuration of impairments

When using script commands to configure impairments, the script commands are grouped into 3 groups:

- Impairment configuration
- Scheduler configuration
- Distribution configuration

In addition to the configuration, the impairment can be turned on and off with the given configuration.

To configure the different impairments, each impairment is assigned an 'impairment ID' (iid) as illustrated in Table 8.

Impairment ID	Impairment Name
0	DROP
1	MISORDER
2	DELAY / JITTER
3	DUPLICATION
4	CORRUPTION
5	POLICER
6	SHAPER

Table 8: Impairment IDs

The impairment ID is used in combination with the filter ID (fid) in the scheduler command and the "distribution commands" to configure the impairments in each flow. The fid / iid addressing is illustrated in Table 9.

Flow to configure	Impairment to configure	How to address: [fid, iid]
0	Drop	[fid=0, iid=0]
3	Corruption	[fid=3, iid=4]
7	Misordering	[fid=7, iid=1]

Table 9: How to configure impairments using [fid,iid]

The impairment configuration is described in the following sub-sections, followed by two configuration examples in sub-section 9.2.4.

9.2.1 Impairment configuration

Some impairments need configuration of the impairment event itself. E.g. for packet corruption, it is required to configure at which level (FCS/IP/TCP/UDP) the corruption takes place, while for drop, there is nothing to configure, because when a drop event occurs, the packet is simply dropped.

Table 10 lists the impairments which have an associated impairment configuration script command.

Impairment Name	Configuration command
Drop	N.A.
Misordering	PE_MISORDER[fid]
Delay / Jitter	N.A.
Duplication	N.A.
Corruption	PE_CORRUPT[fid]
Policer	PE_BANDPOLICER[fid]
Shaper	PE_BANDSHAPER[fid]

Table 10: Impairment configuration commands.

The commands all take a fid as input. The details of the impairment script commands are described in section 10.

9.2.2 Scheduler configuration

A single scheduler command is defined for all impairments.

```
PED_SCHEDULE[fid,iid]
```

The schedule command takes fid and iid to identify the impairment to configure. The scheduling command is described in detail in section 12.

9.2.3 Distribution configuration

The distribution commands are used to configure when to apply the selected impairment to the packets in the flow. The supported distribution commands are listed in Table 11.

Distribution Name	Distribution Configuration Command
Off	PED_OFF[fid, iid]
Constant Latency	PED_CONST[fid, iid]
Accumulate & Burst	PED_ACCBURST[fid, iid]
Step	PED_STEP[fid, iid]
Fixed probability	PED_FIXED[fid, iid]
Random probability	PED_RANDOM[fid, iid]
Fixed burst	PED_FIXEDBURST[fid, iid]
Random Burst	PED_RANDOMBURST[fid, iid]
Gilbert-Elliot	PED_GE[fid, iid]
Bit Error Rate	PED_BER[fid, iid]
Uniform	PED_UNI[fid, iid]
Gaussian (Normal)	PED_GAUSS[fid, iid]
Poisson	PED_POISSON[fid, iid]
Gamma	PED_GAMMA[fid, iid]
Custom	PED_CUSTOM[fid, iid]

Table 11: Distribution configuration commands

Notice the PED_OFF distribution. This is the default distribution at power up. It contains no impairment configuration and the impairment is turned off. Assigning the PED_OFF distribution to an impairment will clear all impairment configuration and turn off the impairment.

The distribution commands take fid and iid to identify the impairment to configure. Depending on the type of impairment, the distribution commands will either specify the number of packets between applying the impairment (inter-packet) or the delay applied to each packet (latency / jitter).

The details of the distribution commands are described in section 11.

9.2.4 Configuration example

This sub-section contains two examples of how to configure impairments using script commands.

<u>FCS corruption with fixed drop probability:</u>

This example illustrates how to configure a fixed drop probability of 5.4321 % at the Ethernet FCS layer with a duration of 1 sec. and a repeat period of 1.5 sec.

This will be configured for fid = 0 (Port default flow).

The corruption iid = 4 is found in Table 8.

- <u>Impairment configuration</u>
  Configure impairment corruption at the Ethernet FCS level

```
PE_CORRUPT[0] ETH
```

- <u>Scheduler configuration</u>
  Configure duration = 1 sec and repeat period = 1.5 sec

```
PED_SCHEDULE[0,4] 1000 1500
```

- <u>Distribution configuration</u>
  Configure fixed distance drop probability of 5.4321 %

```
PED_FIXED[0,4] 54321
```

- <u>Turn on the impairment.</u>

```
PED_ENABLE[0,4] ON
```

- <u>Turn off impairment.</u>

```
PED_ENABLE[0,4] OFF
```

<u>Gaussian Jitter:</u>

This example illustrates how to configure a Gaussian jitter distribution with an average = 50 us and standard deviation = 2 us.

This will be configured for fid = 7 (Highest priority flow filter).

The latency / jitter iid = 2 is found in Table 8.

- <u>Impairment configuration</u>
  There is no impairment configuration for latency / jitter. (See Table 10).

- <u>Scheduler configuration</u>
  The scheduling command is generally not supported for latency / jitter impairments (see Table 14).

- <u>Distribution configuration</u>
  Configure Gaussian distribution with average delay = 50 us and std dev = 2 us

```
PED_GAUSS[7,2] 50000 2000
```

- Turn on the impairment.

```
PED_ENABLE[7,2] ON
```

- Turn off impairment.

```
PED_ENABLE[7,2] OFF
```

10 Flow Impairments

This section describes the impairments which are configured on a per flow basis. For a definition of flows, see section 5.

As described in section 9, each impairment can be assigned a set of distributions and a scheduler. This section will focus on the Chimera impairments, including examples of how to configure selected distributions and the scheduler. However, for an elaborate description of the distributions available for each impairment, see section 11. For an elaborate description of the scheduler, see section 12.

To enable flow impairments, the impairments must be enabled on the port level. This is illustrated in Figure 38.

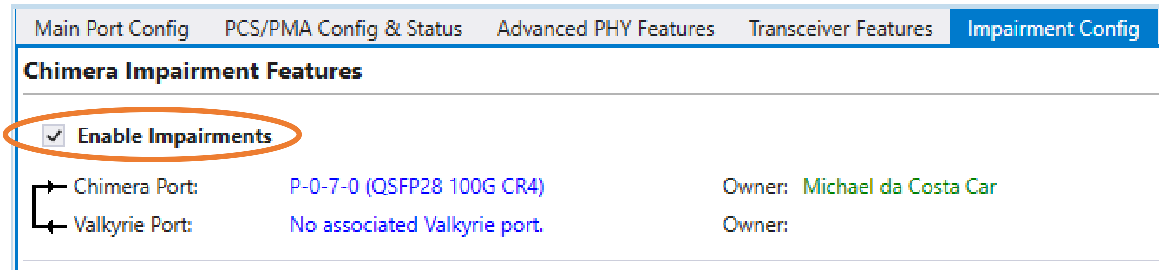


Figure 38: How to enable impairments.

If impairments are configured at the flow level but not enabled at the port level, they will not have any effect on the flow.

Script configuration example:
The following example illustrates how to enable flow impairments on a port.

```
P_EMULATE ON
```

To configure the impairments for a given flow, first select the flow to configure using UI. Figure 39 illustrates how to select flow 0 (Port default flow).

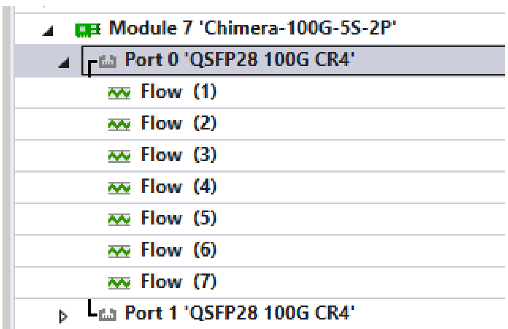


Figure 39: How to select flow to configure in the UI.

Then, expand the impairment window for the impairment to configure. The available impairment windows are illustrated in Figure 40.

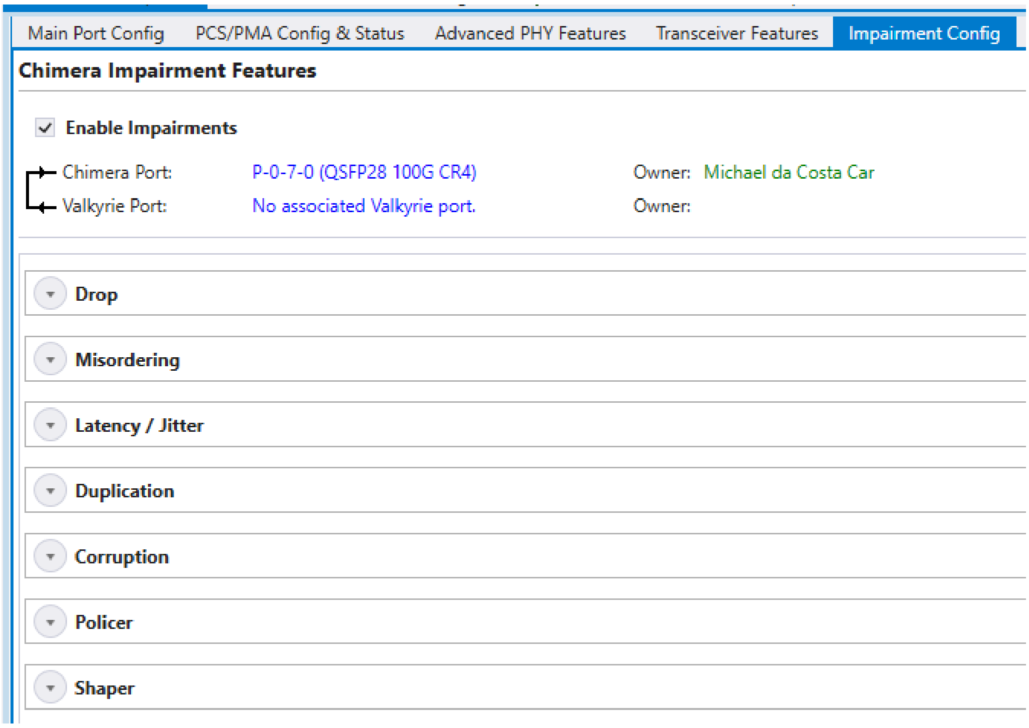


Figure 40: How to select impairment to configure in the UI.

The configuration of each impairment is explained in the following sub-sections.

10.1 Packet duplication (iid = 3)

Packet duplication will duplicate a packet, so the packet is transmitted twice in the Ethernet packet flow. The duplicate packet is inserted right after the original packet.

Notice that packet duplication is located after the shapers, just before the Tx port in the impairment pipeline. I.e., enabling packet duplication will add packets to the packet flow after the shapers, hereby increasing the BW compared to what was configured in the shaper. For further details on shapers, see section 10.5.

Configuration of duplication using the UI is illustrated in Figure 41.

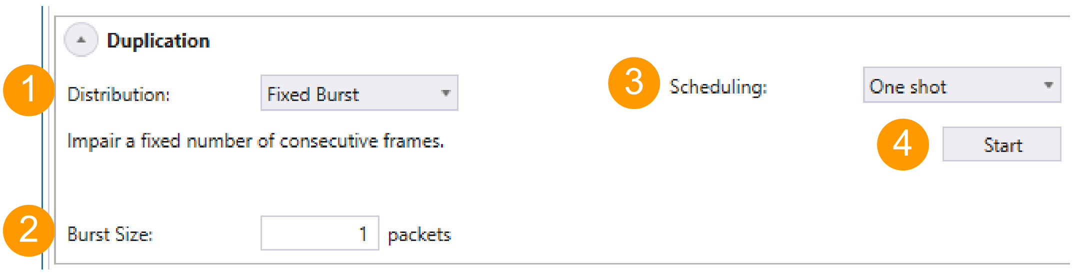


Figure 41: Duplication configuration in the UI.

(There is no configuration of the impairment for duplication).

<u>To configure duplication:</u>

1) Select the relevant distribution from the distribution dropdown menu (e.g. Fixed Burst).

2) Supply the parameters to configure the selected distribution (e.g. Burst Size = 1).
3) Configure the scheduler (e.g. One shot).
4) Press "Start" to activate the impairment.

The configuration shown in Figure 41 will cause the next packet to be duplicated, implementing a burst of 1 duplicated packet.

Note: For every impairment, distribution "Fixed Burst" can be used to manually insert a number of consecutively impairments. Simply configure the burst size and press start.

<u>Script configuration example:</u>
The example below illustrates how to duplicate a single packet (Schedule = One shot).

```
PED_SCHEDULE[fid,3]      1    0
PED_FIXEDBURST[fid,3]    1
PED_ENABLE[fid,3]        ON
```

10.2 Packet drop (iid = 0)

Packet drop will cause packets to be removed from the Ethernet packet flow.

Configuration of drop using the UI is illustrated in Figure 42.

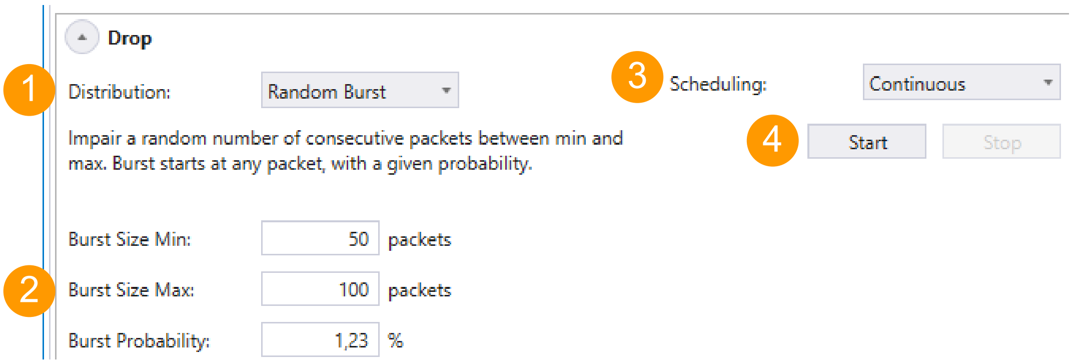


Figure 42: Drop configuration in the UI.

(There is no configuration of the impairment for Drop).

<u>To configure Drop:</u>
1) Select the relevant distribution from the distribution dropdown menu (e.g. Random Burst).
2) Supply the parameters to configure the selected distribution.
(e.g. Burst Size Min = 50, Burst Size Max = 100 and Burst Probability = 1.23 %.)
3) Configure the scheduler (e.g. Continuous)
4) Press "Start" to activate the impairment.

The configuration shown in Figure 41 will cause a drop burst to start at any packet with a probability of 1.23 %. The size of each burst will be selected randomly between 50 packets and 100 packets.

5) To stop dropping packets press "stop".

<u>Script configuration example:</u>
The example below illustrates how to configure the same configuration using script commands.

```
PED_SCHEDULE[fid,0]         1  0
PED_RANDOMBURST[fid,0]      50 100 1230
PED_ENABLE[fid,0]           ON
```

10.3 Misordering (iid = 1)

Misordering causes packets to be taken out of the Ethernet packet flow and delayed for a configurable number of packets, after which they are re-inserted into the packet flow. The number of packets that the packet is delayed is referred to as the “Misorder Depth”.

At any point in time, only a single packet can be in queue to be re-inserted. As a result, the following limitation applies to the values of probability and depth.

> The values configured for “Probability” and “Depth” must comply to the following constraint:
>
> Probability in % * (Depth+1) <= 100%.

The number of distributions which support misordering are limited to:

- Fixed Burst with burst size = 1
- Fixed Rate

Configuration of misordering using the UI is illustrated in Figure 43.

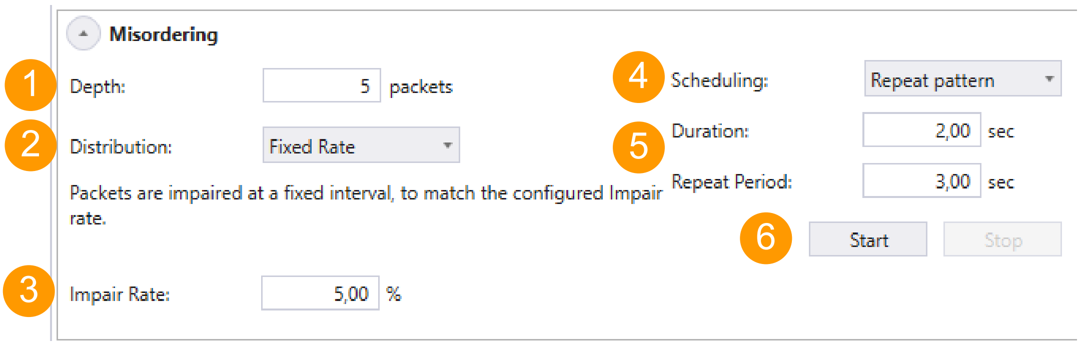


Figure 43: Misorder configuration in the UI.

<u>To configure misordering:</u>

1) Configure the misordering depth.
2) Select the relevant distribution from the distribution dropdown menu[6] (e.g. Fixed Rate).
3) Supply the parameters to configure the selected distribution (e.g. Impair Rate = 5%)
4) Select the relevant scheduler function from the dropdown menu (e.g. Repeat Pattern)
5) Configure the selected scheduler function (e.g. Duration = 2 sec, Repeat Period = 3 sec)
6) Press “Start” to activate the impairment.

The configuration above will cause 5% of the packets to be extracted from the packet flow and re-inserted after 5 packets.

[6] Only Fixed Burst (Burst Size = 1) and Fixed Rate support misordering. See section 11 for details.

<u>Script configuration example:</u>

The example below illustrates how to configure misordering as illustrated in Figure 43.

```
PE_MISORDER   [fid]    5
PED_SCHEDULE [fid,1] 2000 3000
PED_FIXED    [fid,1] 50000
PED_ENABLE   [fid,1] ON
```

10.4 Corruption (iid = 4)

Chimera supports packet corruption at the following protocol layers:

- Ethernet FCS
- IP
- TCP
- UDP

Corruption is done by altering a bit in the checksum for the configured protocol. Furthermore, when corruption is done at IP / TCP / UDP level, the Ethernet FCS is corrected, so the checksum error only appears at the configured level.

Note: when configuring corruption at IP / TCP / UDP level, the flow filter must include the selected layer in the flow filter (see section 6).

I.e., if corruption is configured at the UDP level, the flow filter must include all relevant protocols:

- Ethernet
- (optionally) VLAN(s) / MPLS
- IPv4 / IPv6
- UDP

This implies that IP / TCP / UDP corruption is not supported for the port default flow (flow = 0), because this flow has no filter.

Configuration of UDP checksum corruption using the UI is illustrated in Figure 44.

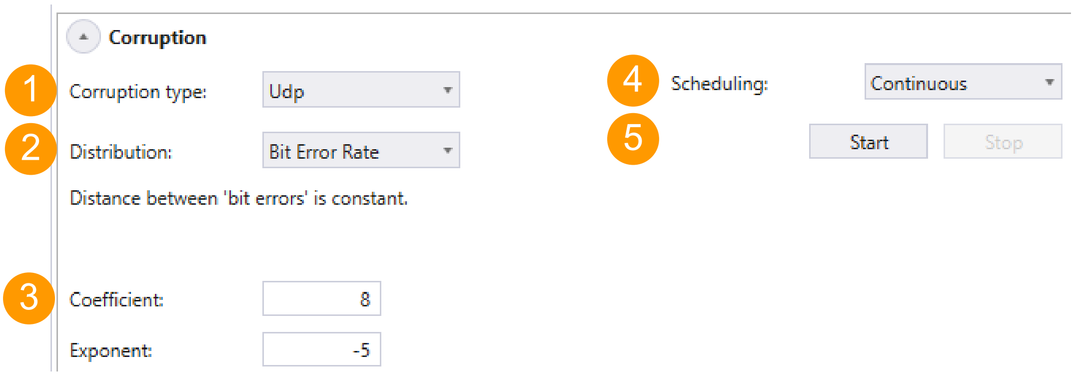


Figure 44: UDP checksum configuration in the UI.

<u>To configure UDP checksum corruption:</u>

1) Configure the protocol layer to corrupt (e.g. UDP).
2) Select the relevant distribution from the distribution dropdown menu (e.g. Bit Error Rate).
3) Supply the parameters to configure the selected distribution

(e.g. Coefficient = 8, Exponent = -5)

4) Select the relevant scheduler function from the dropdown menu (e.g. Continuous).
5) Press “Start” to activate the impairment.

The configuration above will cause an UDP checksum error to be inserted into the packet flow for every $8 * 10^{-5}$ bits of packet data.

<u>Script configuration example:</u>

The example below illustrates how to configure corruption as illustrated in Figure 44.

```
PE_CORRUPT   [fid[7]]    UDP (3)
PED_SCHEDULE [fid[7],4]  1   0
PED_BER      [fid[7],4]  8   -5
PED_ENABLE   [fid,0]     ON
```

10.5 Flow BW control

Chimera implements policers at every flow input, which will drop all packets that exceed the configured bandwidth (CIR) and burst size (CBS).

Likewise, Chimera implements shapers at the output, which will shape the outgoing traffic to a configurable bandwidth (CIR) and burst size (CBS). If excess packets are available, there is a buffer of configurable size (“Buffer size”) for storing excess packets. If the buffer overflows due to shaper BW limiting, packets will be dropped.

Policers and shapers implement a leaky bucket and update the current fill level. The fill level is reduced with the rate specified by CIR and increased with the packet size when a packet is forwarded. If the bucket fill level is above the CBS when a packet arrives at the policer / shaper, the packet will be dropped (policer) or held back (shaper). If the fill level is below the CBS, the packet will be forwarded, and the size of the packet is added to the fill level.

It is configurable whether the policer and shaper will be applied at layer 1 or layer 2. When configured for layer 2, only the Ethernet packets starting with the DMAC and ending with the FCS are counted as part of the traffic. When shaping at layer 1 the minimum IPG (= 12 bytes) and the preamble (= 8 bytes) are considered part of the traffic.

When configuring the parameters of the policer and shaper, the following limits apply:

Parameter	Legal values.	Comments	Step size
CIR:	0 → 1,000,000	Value is multiplied by 100 kbps	1 (= 100 kbps)
CBS:	0 → 2,097,152	Bytes	1 byte
Buffer size[8]*:*	0 → 2,097,152	Bytes	128 bytes

Furthermore, the CBS must be configured >= maximum packet size in flow + 64 bytes to function correctly.

Policers and shapers do not support any impairment distributions or scheduler functions.

[7] IP / TCP / UDP corruption not supported for fid = 0 (Default flow).

[8] Shaper only.

10.5.1 Ingress policers (iid = 5)

Configuration of the ingress policer using the UI is illustrated in Figure 45.

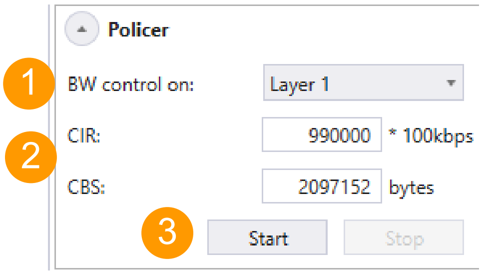


Figure 45: Configuring flow policer in the UI.

<u>To configure ingress policer:</u>

1) Select at which layer to implement the policer from dropdown menu (e.g. Layer 1).
2) Supply the parameters to configure policer
   (e.g. CIR = 9.9 Gbps, CBS = 2 Mbyte)
3) Press “Start” to activate the policer.

<u>Script configuration example:</u>

The example below illustrates how to configure the policer as illustrated in Figure 45.

```
PE_BANDPOLICER [fid] ON L1 990000 2097152
```

10.5.2 Egress shapers (iid = 6)

Notice that the shapers are located before the packet duplication in the impairment pipeline (see Figure 16). I.e., if packet duplication is configured, duplicate packets are added to the flow after the shaper, and the resulting output bandwidth will be higher than the one configured in the shaper.

Furthermore, the amount of memory allocated for the shaper buffer will be taken from the buffer used for generating latency, so when memory is allocated for shaper buffering, the guaranteed lossless latency listed in Table 3 will decrease accordingly.

Configuration of the shaper using the UI is illustrated in Figure 46.

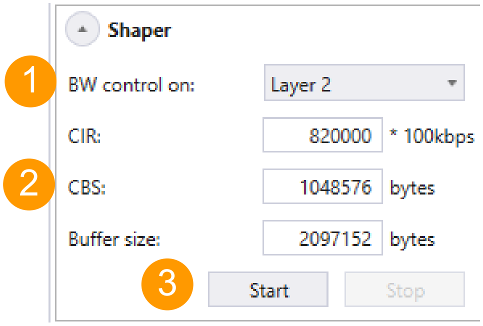


Figure 46: Configuring flow shapers.

<u>To configure egress shaper:</u>

1) Select at which layer to implement the shaper from dropdown menu (e.g. Layer 2).
2) Supply the parameters to configure the shaper
   (e.g. CIR = 8.2 Gbps, CBS = 1 Mbyte, Buffer Size = 2 Mbyte)
3) Press “Start” to activate the impairment.

Script configuration example:

The example below illustrates how to configure the shaper as illustrated in Figure 46.

```
PE_BANDSHAPER [fid] ON L2 820000 1048576 2097152
```

10.6 Latency / Jitter (iid = 2)

The latency / jitter impairment differs significantly from the other impairments described above, because it affects the delay of each packet. As a consequence, the distributions which can be assigned to the latency / jitter impairment define latencies rather than the distance in packets between two impaired packets.

Figure 47 illustrates how to configure a flow for a Gaussian jitter distribution with an average delay of 50 us and a standard deviation of 2.5 us.

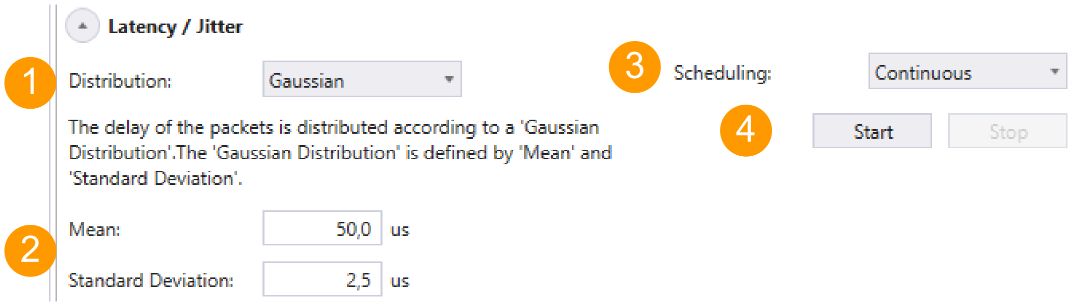


Figure 47: Latency jitter configuration in the UI.

(There is no configuration of the impairment for latency / jitter).

To configure latency / jitter:

1) Select the relevant distribution from the distribution dropdown menu (e.g. Gaussian).
2) Supply the parameters to configure the selected distribution (e.g. Mean = 50.0 us, Standard Deviation = 2.5 us.).
3) Configure the scheduler (e.g. Continuous).
4) Press “Start” to activate the impairment.

The configuration shown in Figure 47 will cause a Gaussian jitter distribution to be applied to the packets on the flow.

5) To stop dropping packets, press “stop”.

Script configuration example:

The example below illustrates how to configure the same configuration using script commands.

```
PED_GAUSS [fid,2] 50000 2500
PED_ENABLE[fid,0] ON
```

11 Impairments distributions

As stated above, Chimera supports a very flexible distribution scheme with a variety of distributions which can be applied depending on the impairment type.

Overall, Chimera supports two different ways of implementing the distribution:

- Inter-packet: The distribution determines the distance between the impaired packets in terms of packets. For example, the number of packets between two dropped packet could be 10.

This kind of distribution applies to the following impairments:
    - Drop
    - Corruption
    - Duplication
    - Misordering
- Latency: The distribution determines the time the packet is delayed in Chimera. If it is not a constant latency, this type of distribution specifies a combination of fixed latency and jitter. E.g., the jitter distribution for the packets in flow could follow a Gaussian distribution.
  This kind of distribution applies to the following impairments:
    - Latency / jitter

The policers and shapers do not support assignment of a distribution function.

Furthermore, some distribution functions are implemented as logic functions, while others are implemented using table look up to approximate mathematical functions. When implementing an inter-packet distribution, the table will contain 512 entries, while the table will contain 1024 entries for latency distributions.

The distributions supported in Chimera for each impairment are illustrated in Table 12.

"✓" : supported
"×" : not supported

Impairments / Distributions	Drop	Corruption – (FCS / IP/TCP/UDP)	Duplication	Misordering	Latency / Jitter	Scheduler
Off	✓	✓	✓	✓	✓	**N.A.**
Logic based						
Manual	✓	✓	✓	✓	×	**N.A.**
Fixed burst	✓	✓	✓	×	×	**One shot / Repeat**
Accumulate & Burst	×	×	×	×	✓	
Fixed Rate	✓	✓	✓	✓	×	**Continuous / Repeat Pattern**
BER	✓	✓	✓	×	×	
Random Rate	✓	✓	✓	×	×	
Gilbert-Elliot	✓	✓	✓	×	×	
Random burst	✓	✓	✓	×	×	
Constant	×	×	×	×	✓	**Continuous**
Table lookup (inter packet: 512 / latency: 1024 samples)						
Uniform	✓	✓	✓	×	✓	**Continuous / Repeat Pattern (latency not supported)**
Gamma	✓	✓	✓	×	✓	
Gaussian	✓	✓	✓	×	✓	
Poisson	✓	✓	✓	×	✓	
Step	×	×	×	×	✓	
Custom	✓	✓	✓	×	✓	

Table 12: Impairment distributions supported in Chimera

The following sub-sections will describe each distribution in detail and provide script examples of how to configure.

11.1 Logic based distributions

This sub-section contains a description of the logical distributions.

11.1.1 Off distribution

This distribution is default at power-up and contains no configuration of any impairment parameters. If assigned to an impairment it will turn off the impairment and clear all impairment configuration.

11.1.2 Manual injection

It is often convenient during testing to manually introduce a limited number of impairments. To do this, the user can configure the fixed burst described in section 11.1.3.

11.1.3 Fixed burst

The "Fixed burst", when triggered, will impair a number of consecutive packets specified by the "Burst Size".

An example of how to configure fixed burst for drop is illustrated in Figure 48.

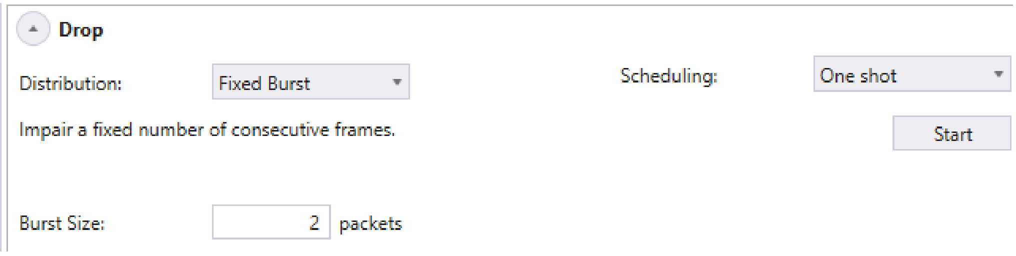


Figure 48: Configuring "Fixed Burst" distribution in the UI.

Distribution parameters:

- Burst Size: Specifies the number of consecutive packets to impair

(For valid parameter ranges, please refer to the script command description.)

This configuration will cause 2 consecutive packets to be dropped when pressing "Start".

Script configuration example:

The example below illustrates how to configure the configuration above using script commands.

```
PED_SCHEDULE    [fid,0] 1 0
PED_FIXEDBURST  [fid,0] 2
```

For fixed burst configured for one shot, there is a special command to determine whether there is a burst pending or not.

```
PED_ONESHOTSTATUS[fid,2] ?
```

This command will return the value of the register which is used to trigger a fixed burst. In case a 1 is returned, there is a burst pending. The next packet on the flow will trigger the burst.

11.1.4 Random burst

"Random Burst" implements bursts of random size. The burst is triggered randomly based on a configurable per packet probability and subsequently impair a random number of consecutive packets chosen between minimum burst size ("Burst Min") and maximum burst size ("Burst Max").

An example of how to configure fixed burst for corruption is illustrated in Figure 49.

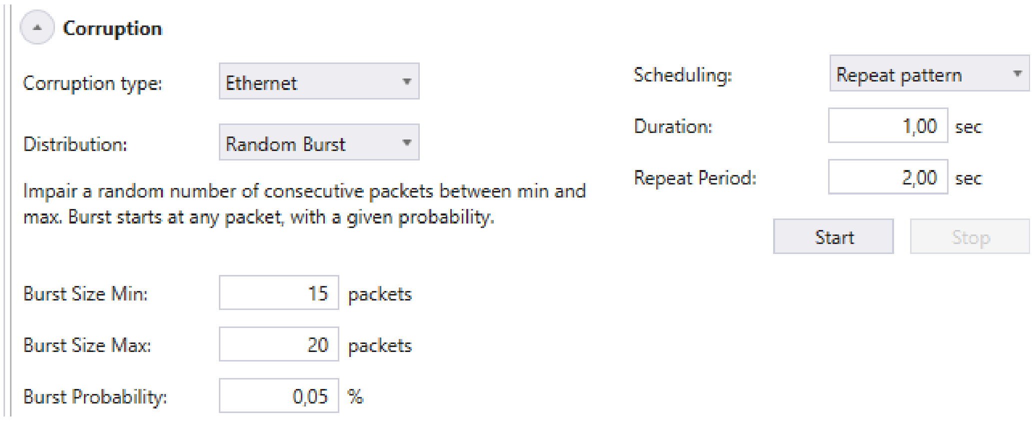


Figure 49: Configuring "Random Burst" distribution in the UI.

<u>Distribution parameters:</u>

- Burst Size Min: Specifies the minimum burst size.
- Burst Size Max: Specifies the maximum burst size.
- Burst Probability: Specifies the probability that a burst will start at any given packet.

(For valid parameter ranges, please refer to the script command description.)

In the example above, every packet has a 0.05 % probability of starting a burst. The burst size will be randomly chosen between 15 and 20 packets. The impairment will be restarted every 2.0 sec and turned off after 1.0 sec.

<u>Script configuration example:</u>
The example below illustrates how to configure the same using script commands.

```
PED_SCHEDULE    [fid,4] 100 200
PED_RANDOMBURST[fid,4] 15 20 500
```

11.1.5 Fixed rate

"Fixed Rate" will impair a configurable fraction of the packets in a predictable way, with nearly equal distance between impairments[9].

An example of how to configure fixed rate for duplication is illustrated in Figure 50.

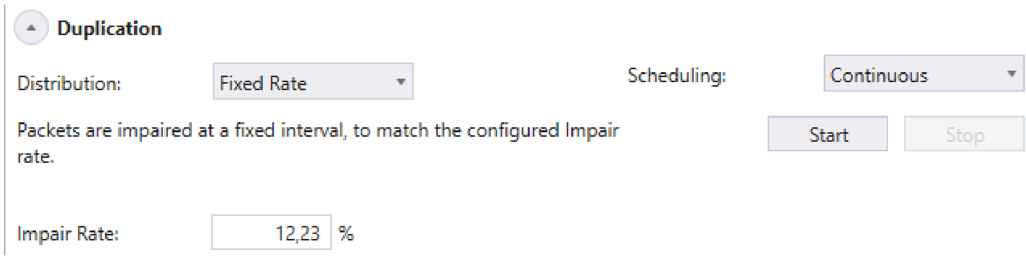


Figure 50: Configuring "Fixed Rate" distribution in the UI.

[9] The distance between impairments is adjusted to match the configured rate in a predictable repeatable pattern.

Distribution parameters:

- Impair rate: Fraction of packets to impair.

(For valid parameter ranges, please refer to the script command description.)

In the example above, 12.23% of the packets will be duplicated. Duplication is done with (nearly) equal distance between duplicated packets in a predictable manner to match the configured impair rate.

Script configuration example:
The example below illustrates how to configure the same configuration using script commands.

```
PED_SCHEDULE[fid,3] 1 0
PED_FIXED   [fid,3] 122300
```

11.1.6 Bit Error Rate (BER)

"Bit Error Rate" will impair the packets of a flow equivalent to a configured BER. E.g., if configured for a BER of $5*10^{-8}$, an impairment will be applied for every $0.2*10^{8}$ bits on the flow. The impairments are applied in a predictable way, with nearly equal distance between impairments.

An example of how to configure bit error rate for drop is illustrated in Figure 51.

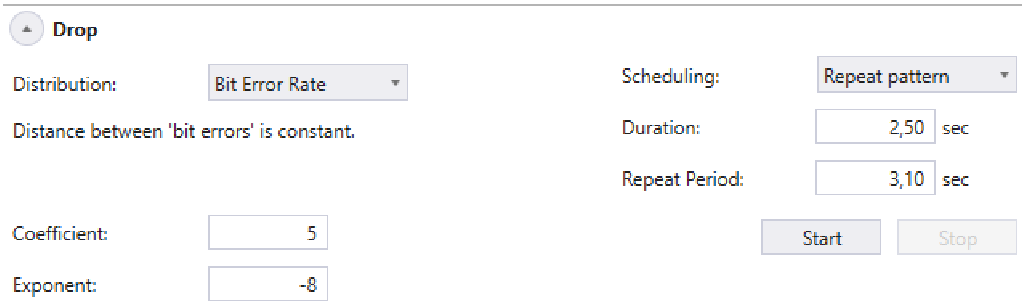


Figure 51: Configuring "Bit Error Rate" distribution.

Distribution parameters:

- Coefficient: The mantissa of the configured BER.
- Exponent: The exponent of the configured BER.

$$BER = Coefficient * 10^{Exponent}$$

(For valid parameter ranges, please refer to the script command description.)

In the example above, one packet will be dropped for every $0.2*10^{8}$ bits on the flow, equivalent to a BER of $5*10^{-8}$. The impairment will be restarted every 3.1 sec. and turned off after a duration of 2.5 sec.

Script configuration example:
The example below illustrates how to configure the same configuration using script commands.

```
PED_SCHEDULE[fid,0] 250 310
PED_BER     [fid,0] 5 -8
```

11.1.7 Random Rate

"Random Rate" will impair a configurable fraction of the packets based on a per packet drop probability, i.e. unlike fixed rate, the impairment pattern is stochastic with an average equal to the configured "Impair Probability".

An example of how to configure random rate for duplication is illustrated in Figure 52.

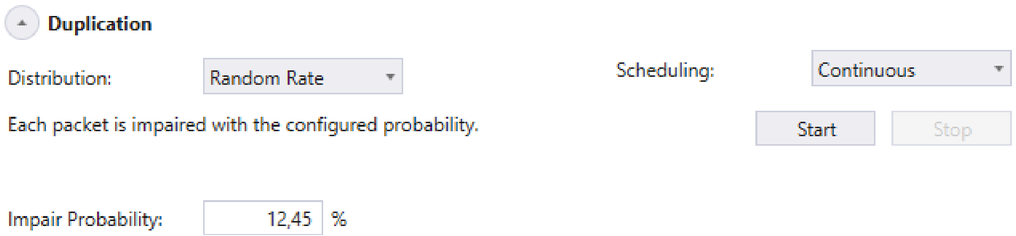


Figure 52: Configuring "Random Rate" distribution in the UI.

<u>Distribution parameters:</u>

- Impair rate: Fraction of packets to impair.

(For valid parameter ranges, please refer to the script command description.)

In the example above, 12.45% of the packets will be duplicated. Duplication is done based on per packet probability.

<u>Script configuration example:</u>

The example below illustrates how to configure the same configuration using script commands.

```
PED_SCHEDULE[fid,3] 1 0
PED_RANDOM   [fid,3] 124500
```

11.1.8 Gilbert-Elliot

The Gilbert-Elliot distribution defines two states, each with a separate packet impairment probability:

- Good state
- Bad state

In any of the two states, there is a certain probability that the system will transition to the other state. The Gilbert-Elliot algorithm is illustrated in Figure 53.

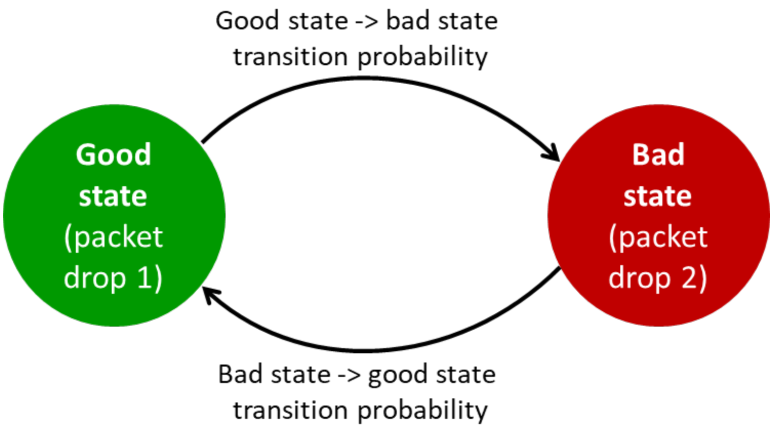


Figure 53: Gilbert-Elliot two states.

When the system is in the “Good State” (“State 1”) ,there is a configurable impairment probability (“Drop 1”) and there is a configurable probability (“Transfer prob 1”) to transition to the “Bad State” (“State 2”). Likewise, when in the “Bad State”, there is a configurable impairment probability (“Drop 2”) and a configurable probability to transition to the “Good State” (“Transfer prob 2”).

An example of how to configure “Gilbert-Elliot” for TCP corruption is illustrated in Figure 54.

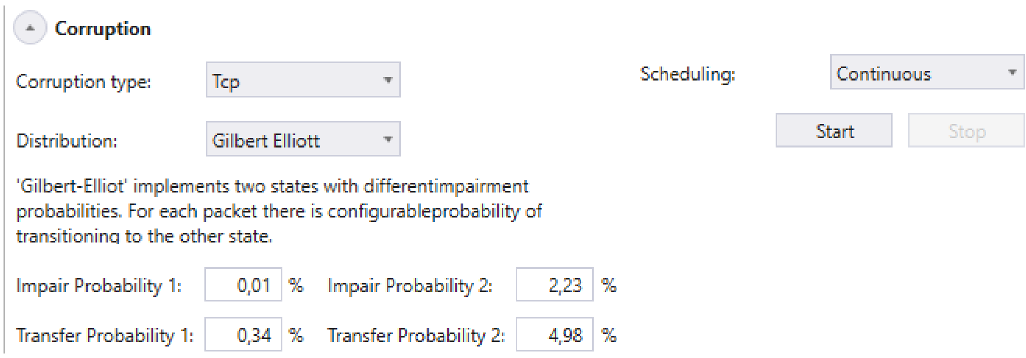


Figure 54: Configuring "Gilbert-Elliot" distribution in the UI.

<u>Distribution parameters:</u>

- Impair Probability 1: The per packet impairment probability in state 1.
- Transfer Probability 1: The per packet probability of moving to state 2 from state 1.
- Impair Probability 2: The per packet impairment probability in state 2.
- Transfer Probability 2: The per packet probability of moving to state 1 from state 2.

(For valid parameter ranges, please refer to the script command description.)

In the example above, the probability of TCP corruption when in state 1 is 0.01 %, while the probability of transferring to state 2 is 0.34 %. When in state 2, the probability of TCP corruption when is 0.01 %, while the probability of transferring to state 1 is 0.34 %

<u>Script configuration example:</u>
The example below illustrates how to configure the same configuration using script commands.

```
PE_CORRUPT   [fid]    TCP
PED_GE       [fid,4]  100 3400 22300 49800
PED_SCHEDULE[fid,4]  1 0
```

11.1.9 Accumulate & Burst

Chimera allows simulating temporary congestion in a network using the “accumulate and burst” distribution. For a configurable period (“Burst Delay”), packets are collected in a buffer, rather than forwarded to the output port. After this period of time, all the buffered packets are forwarded to the output as fast as possible, thus creating a burst. Once buffered packets have been transmitted from the buffer, packets will be forwarded with minimum latency.

The packet accumulation is triggered by the first packet received on the flow after the distribution was enabled.

An example of how to configure “Accumulate & Burst” for latency / jitter corruption is illustrated in Figure 55.

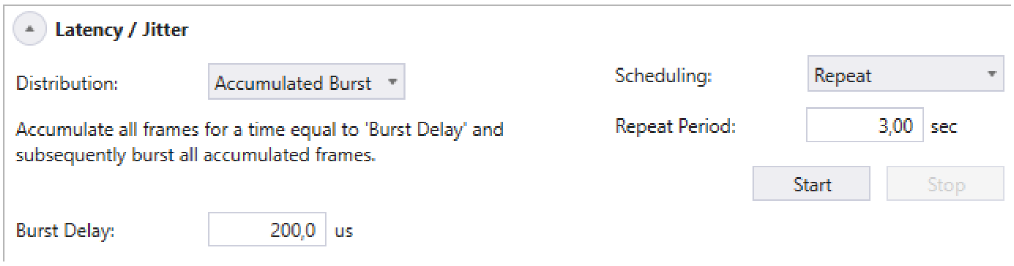


Figure 55: Configuring "Accumulate and Burst" in the UI.

<u>Distribution parameters:</u>

- Burst Delay: Specifies the duration of the packet accumulation after receiving the first packet.

(For valid parameter ranges, please refer to the script command description.)

In the example above, packets are accumulated for 200 us, and subsequently they are sent then to the output as fast as possible. The accumulation is re-triggered every 3.0 sec.

<u>Script configuration example:</u>

The example below illustrates how to configure the same configuration using script commands.

```
PED_SCHEDULE[fid,2] 1 300
PED_ACCBURST[fid,2] 200000
```

For accumulate & burst configured for repeat, there is a special command to determine whether an accumulate & burst event is pending, or whether it has been triggered.

```
PED_ONESHOTSTATUS[fid,2] ?
```

This command will return the value of the register which is used to trigger an accumulate & burst event. In case a 1 is returned, there is an event pending. The next packet on the flow will trigger the accumulate & burst.

11.1.10 Constant Delay

“Constant Latency” will apply a constant latency to all packets in the flow.

Configuring constant latency in the UI is illustrated in Figure 56.

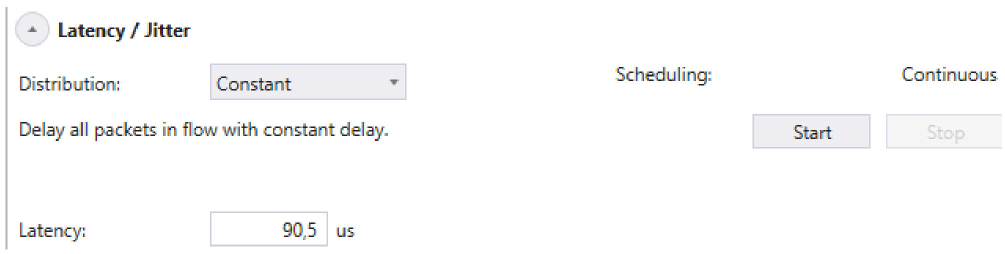


Figure 56: Configuring "Constant Latency" in the UI.

<u>Distribution parameters:</u>

- Latency: Specifies the constant latency to be applied to all packets.

(For valid parameter ranges, please refer to the script command description.)

In the example above, all packets are delayed for 90.5 us.

<u>Script configuration example:</u>
The example below illustrates how to configure the same configuration using script commands.

```
PED_CONST[fid,2] 90500
```

11.2 Table based distributions

This sub-section describes the distributions which are implemented using a table lookup to approximate a mathematical function.

<u>Each table-based distribution exists in 2 flavors:</u>

- Inter-packet distribution: The distribution describes the distance in packets between impairments. This is implemented with 512 table values.
- Latency distribution: This distribution describes the delay applied to the packets. This is implemented with 1024 table values.

This implies that when a table-based distribution is applied to the latency / jitter impairment, the table will contain 1024 values, while for all other impairments, it will contain 512 values.

The maximum data values that can be programmed into the table based distributions are listed in Table 13.

Entry type	Maximum value	
Inter-packet	262,143 packets	
Latency / jitter	30 ms	(Normal timing mode)
	302 ms	(Extended timing mode)

Table 13: Custom distribution maximum data values.

Note: When applying a table-based distribution to latency / jitter, Chimera can only adjust the jitter within the existing IPG of the incoming packets. This implies that packet misordering cannot happen due to jitter. If sending packets with a smaller IPG than the latency specified in a distribution, the distribution function will not be applied as intended.

11.2.1 Uniform

The “Uniform Distribution” will randomly select the distance between impairments from a configured interval defined by a minimum (“min”) and a maximum value (“max”).

Figure 57 illustrates how to configure a uniform distribution for drop.

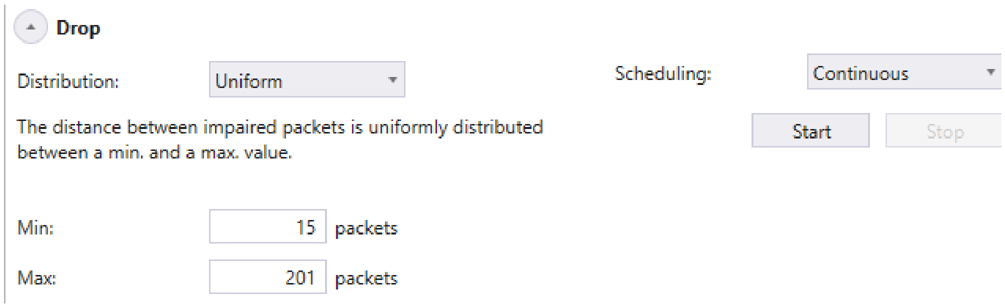


Figure 57: Configuring "Uniform distribution".

<u>Distribution parameters:</u>

- Min: Specifies the minimum number of packets/latency for the uniform distribution.
- Max: Specifies the maximum number of packets/latency for the uniform distribution.

(For valid parameter ranges, please refer to the script command description.)

In the example above, the distance between drops will be chosen randomly in the interval between 15 packets and 201 packets.

<u>Script configuration example:</u>

The example below illustrates how to configure the same configuration using script commands.

```
PED_SCHEDULE[fid,0] 1 0
PED_UNI      [fid,0] 15 201
```

11.2.2 Gaussian

The Gaussian (Normal) distribution implements an approximation of the mathematical function, which is defined by a mean value (μ) and a standard deviation (σ). The Gaussian distribution is illustrated in Figure 58. When a flow is configured for Gaussian Jitter, the mean latency of packets is equal to the configured mean latency, and the deviations of single packets from the mean will be according to the Gaussian distribution.

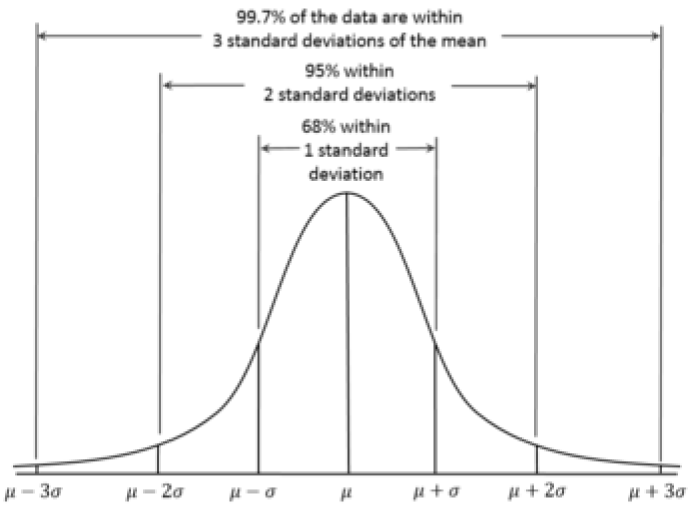


Figure 58: Gaussian distribution.

Chimera limits the Gaussian function to the following latency interval:

$$\mu - 3 \times \sigma <= \text{simulated values} <= \mu + 3 \times \sigma$$

Figure 59 illustrates how to configure Gaussian distribution for latency / jitter.

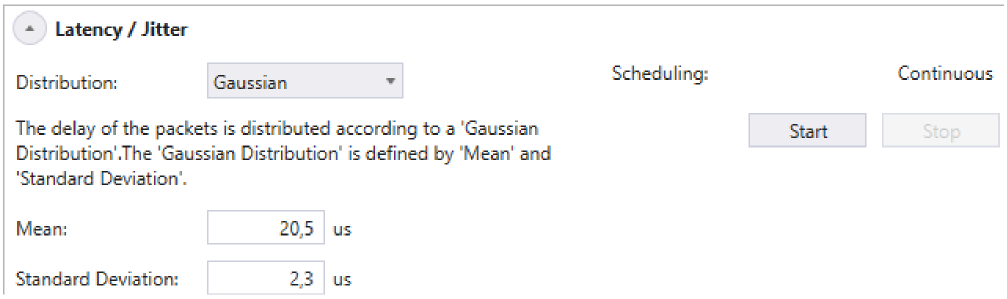


Figure 59: Configuring "Gaussian jitter" in the UI.

<u>Distribution parameters:</u>

- Mean: Specifies the mean value for the Gaussian distribution.
- Standard Deviation: Specifies the standard deviation for the Gaussian distribution.

(For valid parameter ranges, please refer to the script command description.)

In the example above, a jitter with a Gaussian distribution with Mean = 20.5 us and Standard Deviation = 2.3 us is applied continuously to the flow.

<u>Script configuration example:</u>

The example below illustrates how to configure the same configuration using script commands.

```
PED_GAUSS[fid,2] 20500 2300
```

11.2.3 Gamma

The Gamma distribution approximates the mathematical function which is defined by a Shape parameter (κ) and the Scale parameter (θ). The Gamma distribution is illustrated in Figure 60 for different values of the Shape and Scale parameters. When a flow is configured for Gamma Latency / Jitter, the mean latency of packets is equal to the configured mean latency (see below), and the deviations of single packets from the mean will be according to the Gamma distribution.

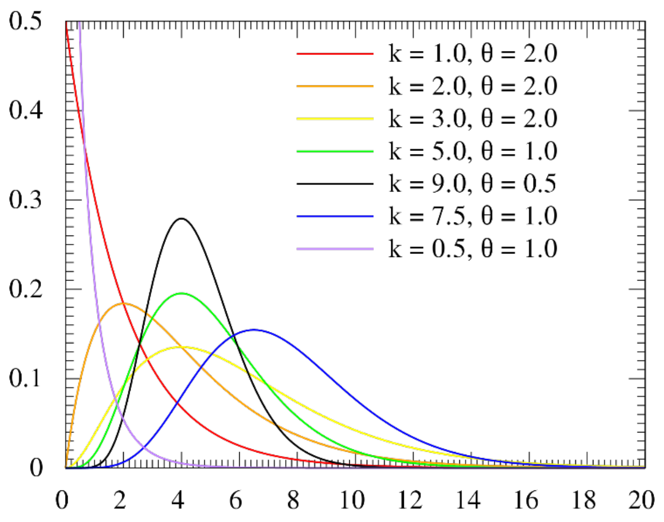


Figure 60: Gamma distribution.

Chimera limits the Gamma function to the following latency interval:

$$\mu - 4 \times \sigma <= \text{simulated values} <= \mu + 4 \times \sigma$$

Where:

$\sigma = \sqrt{\kappa * \theta^2}$ (standard deviation)

$\mu = \kappa * \theta$ (mean value)

Figure 63 illustrates how to configure Gamma distribution for Latency / Jitter.

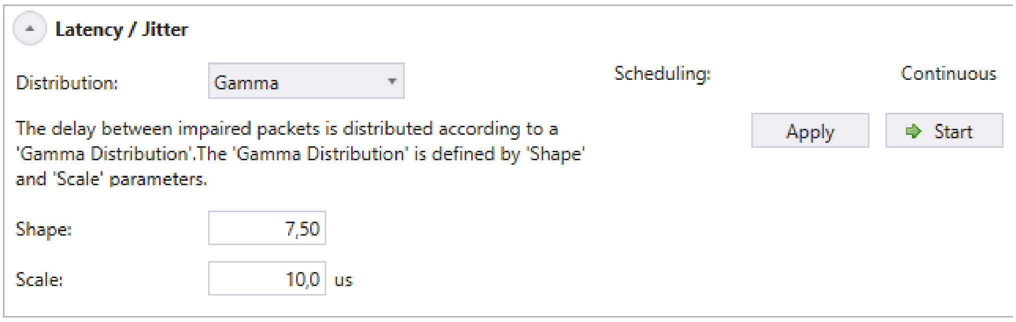


Figure 61: Configuring "Gamma distribution" in the UI.

<u>Distribution parameters:</u>

- Shape (κ): Gamma distribution Shape parameter.
- Scale (θ): Gamma distribution Scale parameter

(For valid parameter ranges, please refer to the script command description.)

In the example above, Latency / Jitter is configured with a Shape parameter of 7.5 and a Scale parameter of 10.0 us. For Latency / Jitter, it is not possible to configure a scheduler function.

<u>Script configuration example:</u>

The example below illustrates how to configure the same configuration using script commands.

```
PED_GAMMA [fid,2] 750 10000
```

11.2.4 Poisson

The Poisson distribution approximates the mathematical function which is defined by a mean value (λ). The Poisson distribution is illustrated in Figure 62. When a flow is configured for poisson jitter, the mean latency of packets is equal to the configured mean latency, and the deviations of single packets from the mean will be according to the Poisson distribution.

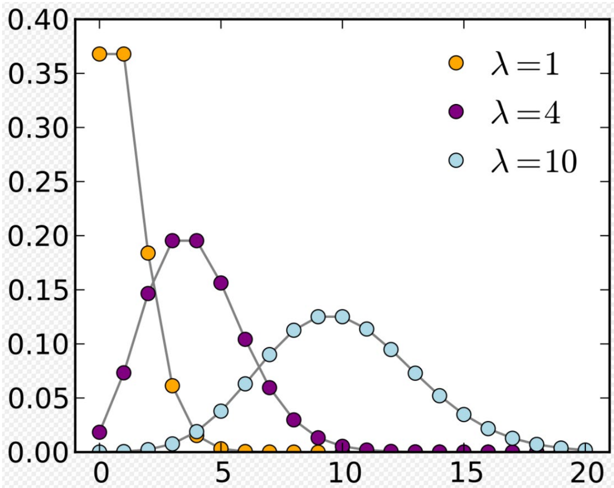


Figure 62: Poisson distribution.

Chimera limits the Poisson function to the following latency interval:

μ - 3 x σ <= simulated values <= μ + 3 x σ

Where:

$\sigma = \sqrt{\lambda}$ (standard deviation)

μ = λ (Mean value).

Figure 63 illustrates how to configure Poisson distribution for duplication.

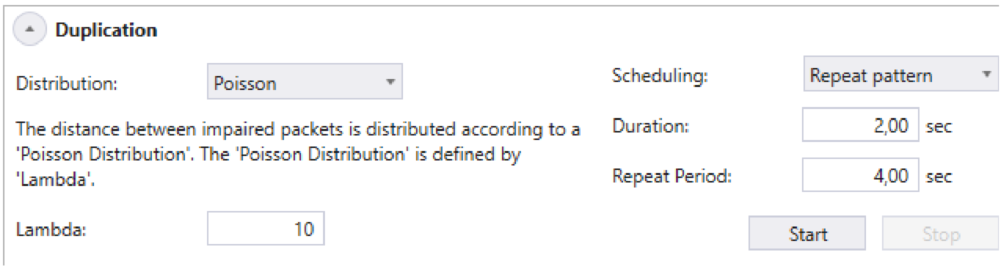


Figure 63: Configuring "Poisson distribution" in the UI.

Distribution parameters:

- Lambda (λ): Specifies the mean value for the Poisson distribution.

(For valid parameter ranges, please refer to the script command description.)

In the example above, duplication with a packet spacing defined by a Poisson distribution with Mean = 10 is applied to the flow for 2.0 sec and then stopped. This will be repeated every 4.0 sec.

Script configuration example:

The example below illustrates how to configure the same configuration using script commands.

```
PED_SCHEDULE[fid,3] 2000 4000
PED_POISSON [fid,3] 10
```

11.2.5 Step

The “Step Distribution” will apply an impairment to a flow, randomly altering between two configurable values. The step distribution is only applicable to latency / jitter.

Figure 64 illustrates how to configure step distribution for Latency / Jitter.

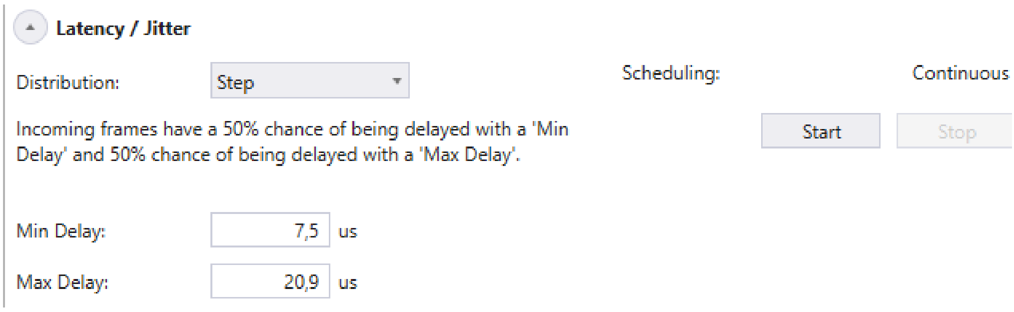


Figure 64: Configuring “Step Distribution” for latency / jitter in the UI.

Distribution parameters:

- Min Delay: Specifies the minimum delay.
- Max Delay: Specifies the maximum delay.

(For valid parameter ranges, please refer to the script command description.)

In the example above, packets will randomly be delayed by either 7.5 us or 20.9 us.

Script configuration example:
The example below illustrates how to configure the same configuration using script commands.

```
PED_STEP[fid,2] 7500 20900
```

11.2.6 Custom Distribution

In addition to the pre-defined distributions described above, Chimera supports the definition of “Custom Distributions”. Custom distributions are table-based distributions which are defined per port. They are identified by a Custom ID (cust_id), which identifies each custom distribution on that port. Chimera supports up to 40 custom distributions per port (cust_id: 1-40). Once the custom distribution is defined, it can be applied to any of the impairments in the impairment pipeline.

A custom distribution is a table-based distribution, where the user can supply the values in the table. Furthermore, the user can configure whether the values in the table should be applied in a predictable order, reading out table index 0, 1, 2 … 511/1023 → 0, 1, 2 …, or whether the values are applied in a random order.

Finally, the user can supply a “Custom Name” for every custom distribution to make it easier to navigate within the distributions defined.

The custom distributions will support 512 table entries for inter-packet distributions and 1024 values for latency / jitter distributions. As a result, only custom distributions with 1024 entries may be assigned to latency / jitter, while custom distributions with 512 entries can be assigned to all other impairments except for misordering, which does not support custom distributions.

Custom distributions are defined using the script command: PEC_VAL.

The PEC_VAL has the following parameters:

- Linear: Determines whether table values are chosen randomly or in predictable order 0 → highest row → 0 etc.
- Symmetric: Reserved for future use – must be set to OFF (0).
- Num_entries: This indicates whether the distribution is an “inter-packet” distribution (=512 entries) or a latency / jitter distribution (=1024 entries).
- 512 or 1024 data values according to num_entries.

The UI supports managing the custom distributions using the “Custom Distributions Library” found on the port impairment tab. The custom distributions library provides an overview of the custom distributions defined for each port and can be used to create, delete and export custom distributions to a file for later use.

The custom distributions library is illustrated in Figure 65.

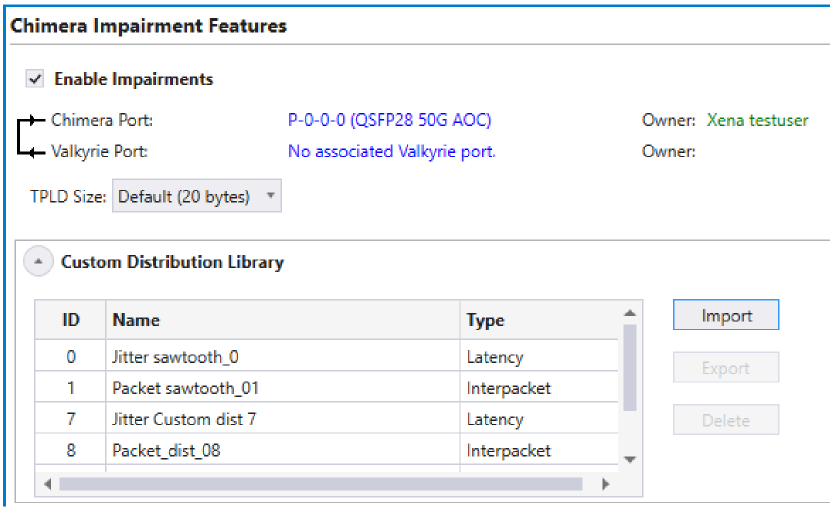


Figure 65: Custom Distributions Library.

The custom distribution library lists which custom distributions are on the selected port and hence which distributions will appear in the custom distribution list, when assigning a custom distribution to an impairment.

The “Type” column indicates whether the distribution can be assigned to latency/jitter or one of the other impairments (inter-packet).

Files used to define custom distributions have the *.xpc extension and must contain the following commands:

- PEC_VAL[cust_id]
- PEC_COMMENT[cust_id]

It is optional whether the *.xpc file contains the [MODULE]/[PORT] in front of the commands. If these values are present, the custom distribution will be defined for the port, indicated by the [PORT] parameter. If [MODULE] / [PORT] is not defined, the custom distribution will be defined for the port currently selected in the UI.

Custom distribution configuration example (inter-packet):
The example below illustrates how to configure a custom distribution for “inter-packet” with a cust_id = 5 and with linear property = ON. Furthermore, it will assign a name to the distribution: “Sample inter-packet distribution”.

```
PEC_VAL[5] ON OFF 512 <DATA_0> <DATA_1> ... <DATA_511>
PEC_COMMENT[5] "Sample inter-packet distribution"
```

Once the "inter-packet" Custom Distribution has been defined, it is possible to assign it to an impairment. Figure 66 illustrates how to use the distribution created above with drop.

Figure 66: Assigning "Custom Distribution" to drop.

<u>To configure custom distribution for drop:</u>

1) Select "Custom Distribution" from the distribution dropdown menu.
2) Select the required custom distribution from the custom distribution list (e.g. #5).
3) Configure the scheduler (e.g. Continuous)
4) Press "Start" to activate the Custom Distribution.

<u>Script configuration example:</u>

The example below illustrates how to configure the same configuration using script commands.

```
PED_SCHEDULE[fid,0] 1 0
PED_CUST     [fid,0] 5
```

<u>Custom distribution configuration example (latency / jitter):</u>

The example below illustrates how to configure a custom distribution for latency / jitter with a cust_id = 13 and with linear property = OFF. Furthermore, it will assign a name to the distribution: "Sample latency distribution".

```
PEC_VAL[13] OFF OFF 1024 <DATA_0> <DATA_1> ... <DATA_1023>
PEC_COMMENT[13] "Sample latency distribution"
```

Once the latency / jitter custom distribution has been defined, it is possible to assign it to an impairment. Figure 67 illustrates how to use the distribution created above with latency / jitter.

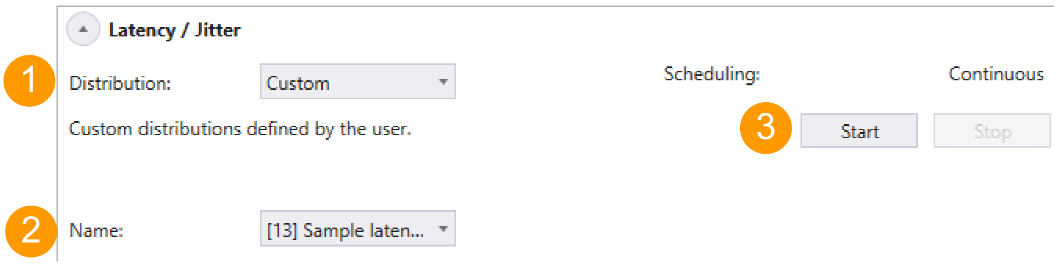


Figure 67: Assigning "Custom Distribution" to latency / jitter in the UI.

<u>To configure custom distribution for latency / jitter:</u>

1) Select "Custom Distribution" from the distribution dropdown menu.
2) Select the required custom distribution from the custom distribution list (e.g. #13).
3) Press "Start" to activate the custom distribution.
   (Latency / jitter does not support a scheduler for custom distributions).

<u>Script configuration example:</u>
The example below illustrates how to configure the same configuration using script commands.

```
PED_CUST[fid,2] 13
```

12 Scheduler functions

As described in section 9, Chimera supports automatically turning the impairments on and off with the configured distribution using a per impairment scheduler.

The scheduler is configured depending on the distribution type which is applied to the impairment. There are 2 types of distributions:

- Continuous distributions
- Burst distributions

Table 14 illustrates which distributions are "Bursty" and which are "Continuous".

Continuous	Bursty
• Random Burst • Fixed Rate • Random Rate • Bit Error Rate • Gilbert-Elliot • Uniform • Gamma • Gaussian • Poisson • Step • Custom distribution	• Fixed burst • Accumulate & Burst

Table 14: Distributions overview

Notice that for latency / jitter, only the Accumulate & Burst supports a scheduler. If other distributions are applied to latency / jitter, only continuous mode is supported (see Table 12).

12.1 Continuous distributions

For continuous distributions, the scheduler can work in 2 modes:

- <u>Continuous:</u> When started, the impairment is applied continuously with the configured distribution until it is manually stopped.

  The example below illustrates how to configure the scheduler in continuous mode for drop with fixed rate distribution of 1.23%.

```
PED_SCHEDULE[fid,0] 1 0
PED_FIXED    [fid,0] 12300
```

- <u>Repeat pattern:</u> When started, the impairment is applied with the configured distribution in a repeated pattern. First it will be applied for a configurable duration and then turned off. It will be restarted for every repeat period.

This is illustrated in Figure 68.

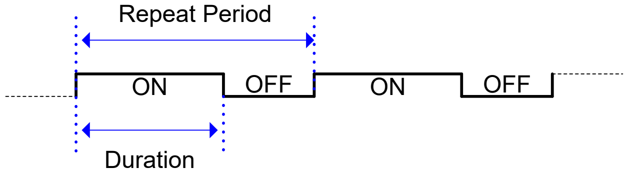


Figure 68: Scheduler function "Repeat Pattern".

The example below illustrates how to configure the scheduler for duration = 1.20 sec and a repeat period = 5.2 sec, applying drop with random rate distribution of 3.3421 %.

```
PED_SCHEDULE[fid,0] 1200 5200
PED_RANDOM   [fid,0] 33421
```

12.2 Bursty distributions

The bursty distributions are characterized by being bursty by nature, i.e. they will automatically terminate if not restarted. E.g., a fixed burst of 8 packets will automatically stop after dropping 8 packets.

For bursty distributions, the scheduler can work in 2 modes.

- One-Shot: When started, the impairment will be applied for the duration of the burst. When the burst terminates, the impairment is turned off.
  The example below illustrates how to configure the scheduler for in shot mode for drop with a fixed burst distribution of 10 packets.

```
PED_SCHEDULE   [fid,0] 1 0
PED_FIXEDBURST[fid,0] 10
```

- Repeat Burst: When started, the impairment will be applied for the duration of the burst. The burst will be restarted every Repeat period.
  This is illustrated in Figure 69.

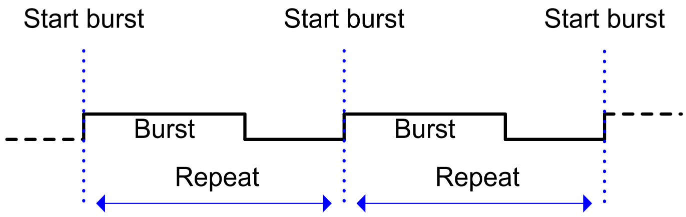


Figure 69: Scheduler function "Repeat Burst".

The example below illustrates how to configure the scheduler to restart the "Accumulate & Burst" every 2.36 sec with a Burst Delay of 0.654 sec.

```
PED_SCHEDULE[fid,2] 1 2360
PED_ACCBURST[fid,2] 654000
```