Check Point
SOFTWARE TECHNOLOGIES LTD.

15 May 2017

# PERFORMANCE TUNING

# R80.10

Administration Guide



Check Point
INFINITY

# Important Information

**Latest Software**

We recommend that you install the most recent software release to stay up-to-date with the latest functional improvements, stability fixes, security enhancements and protection against new and evolving attacks.

**Check Point R80.10**

For more about this release, see the R80.10 home page http://supportcontent.checkpoint.com/solutions?id=sk111841.

**Latest Version of this Document**

Download the latest version of this document http://downloads.checkpoint.com/dc/download.htm?ID=54765.

To learn more, visit the Check Point Support Center http://supportcenter.checkpoint.com.

**Feedback**

Check Point is engaged in a continuous effort to improve its documentation.

Please help us by sending your comments mailto:cp_techpub_feedback@checkpoint.com?subject=Feedback on Performance Tuning R80.10 Administration Guide.

**Searching in Multiple PDFs**

To search for text in all the R80.10 PDF documents, download and extract the complete R80.10 documentation package http://downloads.checkpoint.com/dc/download.htm?ID=54846.
Use **Shift-Control-F** in Adobe Reader or Foxit reader.

Revision History

| Date | Description |
|---|---|
| 15 May 2017 | First release of this document |

# Contents

# Terms

***Affinity***

The assignment of a specified process, Firewall instance, VSX Virtual System, interface or IRQ with one or more CPU cores.

***CoreXL***

A performance-enhancing technology for Security Gateways on multi-core processing platforms.

***Firewall Instance***

On a Security Gateway with CoreXL enabled, the Firewall kernel is copied multiple times. Each replicated copy, or firewall instance, runs on one processing core. These instances handle traffic at the same time and each instance is a complete and independent inspection kernel.

***IPv4***

Internet Protocol Version 4 IP address. A 32-bit number - 4 sets of numbers, each set can be from 0 - 255.

***IPv6***

Internet Protocol Version 6 IP address. 128-bit number - 8 sets of hexadecimal numbers, each set can be from 0 - ffff.

***IRQ Affinity***

A state of binding an IRQ to one or more CPUs.

***Multi-queue***

An acceleration feature that lets you assign more than one packet queue and CPU to an interface.

***Rx Queue***

Receive packet queue

***SND***

Secure Network Distributer. Part of SecureXL and CoreXL, it processes and helps to accelerate network traffic. In SecureXL, it distributes traffic to the accelerated or slow path. In CoreXL, it processes traffic on a specified Firewall instance.

***Traffic***

The flow of data between network resources.

***Tx queue***

Transmit packet queue

# Performance Pack

*In This Section:*



## About this Guide

This guide explains how to use the R80.10 SmartConsole to manage R80.10 and lower Security Gateways.

## Introduction to Performance Pack

Performance Pack is a software acceleration product installed on Security Gateways. Performance Pack uses SecureXL technology and other innovative network acceleration techniques to deliver wire-speed performance for Security Gateways.

- Performance Pack is automatically installed when you run the *First Time Wizard*.
- To enable/disable Performance Pack, run: `cpconfig`

### Supported Features

These security functions are enhanced by Performance Pack:

- Access control
- Encryption
- NAT
- Accounting and logging
- Connection/session rate
- General security checks
- IPS features
- CIFS resources
- ClusterXL High Availability and Load Sharing
- TCP Sequence Verification
- Dynamic VPN
- Anti-Spoofing verifications
- Passive streaming
- Drop rate

## Preparing the Performance Pack

For optimal performance, configure the BIOS and NICs for Performance Pack.

### *BIOS Settings*

- If your BIOS supports CPU clock setting, make sure that the BIOS is set to the actual CPU speed.
- For Hyper-threading, see sk93000 http://supportcontent.checkpoint.com/solutions?id=sk93000.

### *Network Interface Cards*

- If you are using a motherboard with multiple PCI or PCI-X buses, make sure that each Network Interface Card is installed in a slot connected to a *different* bus.
- If you are using more than two Network Interface Cards in a system with only two 64 bit/66 Mhz PCI buses, make sure that the least-used cards are installed in slots connected to the *same* bus.

For an updated list of certified Network Interface Cards, see Certified Network Interfaces (http://www.checkpoint.com/services/techsupport/hcl/#nic/).

**Note** - Performance Pack is automatically disabled on PPTP and PPPoE interfaces

# Command Line

## fwaccel

**Description** Lets you dynamically enable or disable acceleration for IPv4 traffic while a Security Gateway is running. The fwaccel6 has the same functionality for IPv6 traffic. The default setting is determined by the setting configured with cpconfig. This setting reverts to the default after reboot.

Works with the IPv4 kernel.

**Syntax**

```
fwaccel [on|off|stat|stats|conns|templates]
```

### Parameters

| Parameter | Description |
|---|---|
| on | Starts acceleration |
| off | Stops acceleration |
| stat | Shows the acceleration device status and the status of the Connection Templates on the local Security Gateway. |
| stats | Shows acceleration statistics. |
| stats -s | Shows more summarized statistics. |

| | |
|---|---|
| stats -d | Shows dropped packet statistics. |
| conns | Shows all connections. |
| conns -s | Shows the number of connections defined in the accelerator. |
| conns -m *max_entries* | Limits the number of connections displayed by the conns command to the number entered in the variable **max_entries**. |
| templates | Shows all connection templates. |
| templates -m max_entries | Limits the number of templates displayed by the templates command to the number entered in the variable **max_entries**. |
| templates -s | Shows the number of templates currently defined in the accelerator. |

## fwaccel6

**Description** Lets you enable or disable acceleration dynamically while a Security Gateway is running. The default setting is determined by the setting configured using `cpconfig`. This setting goes back to the default after reboot.

Works with the IPv6 kernel.

**Syntax** `fwaccel6 [on|off|stat|stats|conns|templates]`

### Parameters

| Parameter | Explanation |
|---|---|
| `on` | Starts IPv6 acceleration. |
| `off` | Stops IPv6 acceleration. |
| `stat` | Shows the acceleration device status and the status of the Connection Templates on the local Security Gateway. |
| `stats` | Shows summary acceleration statistics. |
| `stats -s` | Shows detailed summarized statistics. |
| `conns` | Shows all IPv6 connections. |
| `conns -s` | Shows the number of IPv6 connections currently defined in the accelerator. |
| `conns -m <max_entries>` | Lowers the number of IPv6 connections shown by the `conns` command to the number entered in the variable `max_entries`. |
| `templates` | Shows all IPv6 connection templates. |

| | |
|---|---|
| `templates -m max_entries` | Lowers the number of templates shown by the `templates` command to the number entered in the variable `max_entries`. |
| `templates -s` | Shows the number of templates currently defined for the accelerator. |

## *Example: fwaccel6 stat*

Description The fwaccel6 stat command displays the acceleration device status and the status of the Connection Templates on the local Security Gateway.

Example fwaccel6 stat -all

Output

```
Accelerator Status : on
Accept Templates : enabled
Accelerator Features : Accounting, NAT, Routing, HasClock, Templates,
 Synchronous, IdleDetection, Sequencing, TcpStateDetect, AutoExpire,
 DelayedNotif, TcpStateDetectV2, CPLS, WireMode, DropTemplates
```

## *Example: fwaccel6 templates*

**Description** The `fwaccel6 templates` command displays all the connection templates

**Example** `fwaccel6 templates`

**Output**

```
Source SPort Destination DPort PR Flags LCT DLY C2S i/f S2C i/f
---------------------------------------- -----
 ---------------------------------------- ----- -- ------- ----
9999:b:0:0:0:0:0:10 * 9999:b:0:0:0:0:0:20 10000 17 ....... 15 0
Lan5/Lan1 Lan1/Lan5
```

## *Example: fwaccel6 stats*

**Description** The `fwaccel6 stats` command displays acceleration statistics

**Example** `fwaccel6 stats`

**Output**

```
Name                  Value    Name                Value
--------------------  ------   ----------------    -----
Accelerated Path
-----------------------------------------------------------
accel packets          2        accel bytes            96
conns created         11        conns deleted           7
C total conns          0        C templates             0
C TCP  conns           0        C delayed TCP conns     0
C non TCP conns        4        C delayed nonTCP con    4
conns from templates   0        temporary conns         0
nat conns              0        dropped packets         0
dropped bytes          0        nat templates           0
port alloc templates   0        conns from nat tmpl     0
port alloc conns       0        conns auto expired      0

Accelerated VPN Path
-----------------------------------------------------------
C crypt conns          0        enc bytes               0
dec bytes              0        ESP enc pkts            0
ESP enc err            0        ESP dec pkts            0
ESP dec err            0        ESP other err           0
AH enc pkts            0        AH enc err              0
AH dec pkts            0        AH dec err              0
AH other err           0        espudp enc pkts         0
espudp enc err         0        espudp dec pkts         0
espudp dec err         0        espudp other err        0

Medium Path
-----------------------------------------------------------
PXL packets            0       PXL async packets        0
PXL bytes              0       C PXL conns              0
C PXL templates        0
```

## fwaccel stats and fwaccel6 stats

The `fwaccel stats` and `fwaccel6 stats` commands show performance statistics. This information can help you understand traffic behavior and help investigate performance issues.

| Statistic parameter | Explanation |
|---|---|
| conns created | Number of created connections |
| conns deleted | Number of deleted connections |
| temporary conns | Number of temporary connections |
| templates | Number of templates currently handled |
| nat conns | Number of NAT connections |
| accel packets | Number of accelerated packets |
| accel bytes | Number of accelerated traffic bytes |
| F2F packets | Number of packets handled by the VPN kernel in slow-path |
| ESP enc pkts | Number of ESP encrypted packets |

| Statistic parameter | Explanation |
|---|---|
| ESP enc err | Number of ESP encrypted errors |
| ESP dec pkts | Number of ESP decrypted packets |
| ESP dec err | Number of ESP decrypted errors |
| ESP other err | Number of ESP other general errors |
| espudp enc pkts | Not in use |
| espudp enc err | Not in use |
| espudp dec pkts | Not in use |
| espudp dec err | Not in use |
| espudp other err | Not in use |
| AH enc pkts | Not in use |
| AH enc err | Not in use |
| AH dec pkts | Not in use |
| AH dec err | Not in use |
| AH other err | Not in use |
| memory used | Not in use |
| free memory | Not in use |
| acct update interval | Accounting update interval in seconds |
| current total conns | Number of connections currently handled |
| TCP violations | Number of packets which are in violation of the TCP state |
| conns from templates | Number of connections created from templates |
| TCP conns | Number of TCP connections currently handled |
| delayed TCP conns | Number of delayed TCP connections currently handled |
| non TCP conns | Number of non TCP connections currently handled |
| delayed nonTCP conns | Number of delayed non TCP connections currently handled |

| Statistic parameter | Explanation |
|---|---|
| F2F conns | Number of connections currently handled by the VPN kernel in slow-path |
| F2F bytes | Number of traffic bytes handled by the VPN kernel in slow-path |
| crypt conns | Number of encrypted connections currently handled |
| enc bytes | Number of encrypted traffic bytes |
| dec bytes | Number of decrypted traffic bytes |
| partial conns | Number of partial connections currently handled |
| anticipated conns | Number of anticipated connections currently handled |
| dropped packets | Number of dropped packets |
| dropped bytes | Number of dropped traffic bytes |
| nat templates | Not in use |
| port alloc templates | Not in use |
| conns from nat tmpl | Not in use |
| port alloc conns | Not in use |
| port alloc f2f | Not in use |
| PXL templates | Number of PXL templates |
| PXL conns | Number of PXL connections |
| PXL packets | Number of PXL packets |
| PXL bytes | Number of PXL traffic bytes |
| PXL async packets | Number of PXL packets handled asynchronously |

## cpconfig

Check Point products are configured using the **cpconfig** utility. This utility shows the configuration options of the installed configuration and products. You can use **cpconfig** to enable or disable Performance Pack. When you select an acceleration setting, the setting remains configured until you change it.

For an alternative method to enable or disable acceleration, see fwaccel (on page 9).

Run: **cpconfig**

A menu shows **Enable/Disable Check Point SecureXL**.

## sim affinity

**Description** The **sim affinity** utility controls various Performance Pack driver features.

Affinity is a general term for binding Network Interface Card (NIC) interrupts to processors. By default, Affinity is not set to the NIC interrupts. Therefore, each NIC is handled by all processors. For optimal network performance, make sure each NIC is individually bound to one processor.

**Syntax**

```
sim affinity [-a|-s|-l]
```

### Parameters

| Parameter | Description |
|---|---|
| **-a** | **Automatic Mode** — (default) Affinity is determined by analysis of the load on each NIC. If a NIC is not activated, Affinity is not set. NIC load is analyzed every 60 seconds. |
| **-s** | **Manual Mode** — Configure Affinity settings for each interface: the processor numbers (separated by space) that handle this interface, or **all**. In Manual Mode, periodic NIC analysis is disabled. |
| **-l** | See Affinity settings. |

## proc entries

**Description** Performance Pack supports **proc** entries. These read-only entries show data about Performance Pack. The proc entries are in **/proc/ppk**.

**Syntax**

```
cat /proc/ppk/[conf|ifs|statistics|drop_statistics]
```

### Parameters

| Parameter | Description |
|---|---|
| conf | Shows Performance Pack configuration. |
| ifs | Shows the interfaces to which Performance Pack is attached. |
| statistics | Shows general Performance Pack statistics. |
| drop_statistics | Shows Performance Pack dropped packet statistics. |

# Performance Tuning and Measurement

## Setting the Maximum Concurrent Connections

To set the number of maximum concurrent connections:

1. In SmartConsole, open the **Gateway Object Properties** window.
2. In the **Optimizations** tab, make sure that the **Calculate connections hash table size and memory pool** option is set to **Automatically**.
3. Set the desired amount of concurrent connections in the **Maximum Concurrent Connections** field.

## Increasing the Number of Concurrent Connections

You can increase the actual number of concurrent connections by reducing the timeout of TCP and UDP sessions:

- TCP end timeout determines the amount of time a TCP connection will stay in the Firewall connection table after a TCP session has ended.
- UDP virtual session timeout determines the amount of time a UDP connection will stay in the Firewall connection table after the last UDP packet was seen by the gateway.

By reducing the above values, the capacity of actual TCP and UDP connections is increased.

## SecureXL Templates

Verify that templates are not disabled using the **fwaccel stat** command.

For further information regarding SecureXL Templates, see sk32578 http://supportcontent.checkpoint.com/solutions?id=sk32578.

## SecureXL NAT templates

Using SecureXL Templates for NAT traffic lets you achieve a high session rate for NAT traffic. SecureXL NAT Templates are supported in cluster in High Availability, VRRP, and Load Sharing modes.

For more, see: sk71200 http://supportcontent.checkpoint.com/solutions?id=sk71200 .

## Delayed Notification

In the ClusterXL configuration, the Delayed Notification feature is disabled by default. Enabling this feature improves performance (at the cost of connections' redundancy, which can be tuned using delayed notifications expiration timeout).

The **fwaccel stats** command indicates the number of delayed connections.

The **fwaccel templates** command indicates the delayed time for each template under the DLY entry.

# Connection Templates

Connection templates are generated from active connections according to the policy rules. The connection template feature accelerates the speed at which a connection is established by matching a new connection to a set of attributes. When a new connection matches the template, connections are established without performing a rule match and therefore are accelerated. Connection templates are generated from active connections according to policy rules. Currently, connection template acceleration is performed only on connections with the same destination port.

Examples:

- A connection from 10.0.0.1/2000 to 11.0.0.1/80 — established through Firewall and then accelerated.
- A connection from 10.0.0.1/2001 to 11.0.0.1/80 — fully accelerated (including connection establishment).
- A connection from 10.0.0.1/8000 to 11.0.0.1/80 — fully accelerated (including connection establishment).

HTTP GET requests to specific server will be accelerated since the connection has the same source IP address.

## *Restrictions*

In general, Connection Templates will be created only for plain UDP or TCP connections. The following restrictions apply for Connection Template generation:

Global restrictions:

- SYN Defender — Connection Templates for TCP connections will not be created
- VPN connections
- Complex connections (H323, FTP, SQL)
- NetQuotas
- ISN Spoofing

If the Rule Base contains a rule regarding one of these components, the Connection Templates will be disabled for connections matching this rule, and for all of the following rules:

- Security Server connections.
- Services of type "other" with a match expression.

  Exceptions - these services will not disable accept templates:
  - `traceroute`
  - `dhcp-request`
  - `dhcp-reply`
- User/Client/Session Authentication actions.
- Services of type RPC/DCERPC/DCOM.

When installing a policy containing restricted rules, you will receive console messages indicating that Connection Templates will not be created due to the rules that have been defined. The warnings should be used as a recommendation that will assist you to fine-tune your policy in order to optimize performance.

### *Testing*

To verify that connection templates are enabled, use the **fwaccel stat** command. To verify that connection templates are generated, use **fwaccel templates**. This should be done while traffic is running, in order to obtain a list of currently defined templates.

## Delayed Synchronization

The synchronization mechanism guarantees High Availability. In a cluster configuration, if one cluster member fails, the other recognizes the connection failure and takes over, so the user does not experience any connectivity issue. However, there is an overhead per synchronized operation, which can occasionally cause a system slow-down when there are short sessions.

Delayed synchronization is a mechanism based upon the duration of the connection, with the duration itself used to determine whether or not to perform synchronization. A time range can be defined per service. The time range indicates that connections terminated before a specified expiration time will not be synchronized. As a result, synchronized traffic is reduced and overall performance increases. Delayed Synchronization is performed only for connections matching a connection template.

**Note** - Delayed synchronization is disabled if the log or account are enabled

Currently, delayed synchronization is allowed only for services of type HTTP or None.

To configure delayed synchronization:

1. In SmartConsole, open the **Object Explorer** and click **Services**.
2. Open an existing TCP service, or create a new TCP service.
3. In the **TCP** window > **General** view, select **HTTP** or **None** from the **Protocol** list.
4. In the **Advanced** view, select **Synchronize connection on cluster**.
5. Select **Start synchronizing**, and then configure the **Seconds after connection initiation** parameter (in seconds).

## Multi-Core Systems

Running Performance Pack on multi-core systems may require more advanced configurations to account for core affinity and IRQ behavior. For more information, see sk33250 http://supportcontent.checkpoint.com/solutions?id=sk33250.

# Performance Measurement

There are various ways to monitor and measure the performance of a Security Gateway.

## *TCP State and Benchmarking*

Some testing applications (SmartBits or Chariot) generate invalid TCP sequences. The Security Gateway TCP state check detects these faulty sequences, and drops the packets. As a result, the benchmark fails. Because these TCP sequences are invalid, they can affect overall Firewall performance.

To disable this type of TCP state check:

1. In SmartConsole, click **Manage & Settings** > **Blades** > **Inspection Settings** > **General**.
2. In the **Inspection Settings** window, search for and double-click **Sequence Verifier**.
3. In the **IPS** tab, select **Protections** > **By Protocol** > **Network Security** > **TCP** > **Sequence Verifier**.
4. In the **Sequence Verifier** window, click **Edit**.
5. Make sure that the Main Action is set to **Inactive**.

   If the protection is not set to **Inactive**, select **Override with Action** and then select **Inactive** from the list.
6. Install Policy to apply the changes**.**

## *Non-accelerated traffic analysis*

Use the **fwaccel stats** command to verify the amount of non-accelerated traffic compared to accelerated traffic.

To understand the possible reasons for the non-accelerated traffic, run these:

- **fwaccel stats -p** command shows statistics on different reasons for non-accelerated traffic.
- For deeper analysis, enable this kernel debug:
  - *fwaccel dbg + offload*
  - *sim dbg + f2f*
  - *fw ctl zdebug*
- When you stop the debug process, run these commands to reset debug flags:
  - *fwaccel dbg resetall*
  - *sim dbg resetall*

## *Performance Troubleshooting*

Additional CLI commands, such as **ethtool**, are available to monitor the performance of the gateway. For a list of these commands and explanation of their usage, see sk33781 http://supportcontent.checkpoint.com/solutions?id=sk33781.

# CoreXL Administration

*In This Section:*



CoreXL is a performance-enhancing technology for Security Gateways on multi-core processing platforms. CoreXL enhances Security Gateway performance by enabling the processing cores to concurrently perform multiple tasks.

CoreXL provides almost linear scalability of performance, according to the number of processing cores on a single machine. The increase in performance is achieved without requiring any changes to management or to network topology.

CoreXL joins ClusterXL Load Sharing and SecureXL as part of Check Point's fully complementary family of traffic acceleration technologies.

On a Security Gateway with CoreXL enabled, the Firewall kernel is replicated multiple times. Each replicated copy, or instance, runs on one processing core. These instances handle traffic concurrently, and each instance is a complete and independent inspection kernel. When CoreXL is enabled, all the kernel instances in the Security Gateway process traffic through the same interfaces and apply the same security policy.

## Unsupported Features

CoreXL does not support these Check Point Suite features:

- Check Point QoS (Quality of Service) for R77 and below.
- IPv6 on IPSO
- Overlapping NAT

To enable a non-supported feature:

1. From the Security Gateway CLI, run `cpconfig`.
2. Select `Disable Check Point CoreXL`.
3. Enter `y` to confirm.
4. Reboot the Security Gateway.

# Default Configuration of CoreXL

When you enable CoreXL, the number of kernel instances is based on the total number of CPU cores.

| Number of Cores | Number of Kernel Instances |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 4 | 3 |
| 6-20 | Number of cores, minus 2 |
| More than 20 | Number of cores, minus 4 -- but no more than 40 |

The default affinity setting for all interfaces is automatic when Performance Pack is installed. See Processing Core Allocation (on page 23). Traffic from all interfaces is directed to the core running the Secure Network Distributor (SND).

# CoreXL for IPv6

R80.10 and higher supports multiple cores for IPv6 traffic. For each firewall kernel instance that works with IPv4 traffic, there is a corresponding firewall kernel instance that also works with IPv6 traffic. Both instances run on the same core.

To check the status of CoreXL on your Security Gateway, run:
`fw6 ctl multik stat`.

The `fw6 ctl multik stat` (multi-kernel statistics) command shows IPv6 information for each kernel instance. The state and processing core number of each instance is displayed, along with:

- The number of connections currently running.
- The peak number of concurrent connections the instance has used since its inception.

# Configuring IPv4 and IPv6 Firewall Instances

After IPv6 support is enabled on the gateway, you can configure the gateway processing cores to run different combinations of IPv4 and IPv6 firewall kernel instances.

- The number of IPv4 instances range from a minimum of two to a number equal to the maximum number of cores on the gateway.

  By default, the number of IPv6 firewall instances is set to two.

  When SMT (Hyper-Threading) is enabled (by default on 15000 & 23000 appliances), the default number of IPv6 instances is 4.
- The number of IPv6 instances range from a minimum of two to a number equal to the number of IPv4 instances.

  The number of IPv6 instances cannot exceed the number of IPv4 instances.
- The total number of IPv4 and IPv6 instances cannot exceed: 40.

To configure the number of IPv6 firewall instances:

1. Open a command prompt on the gateway.
2. Run: `cpconfig`.

   The configuration menu shows.
3. Enter option **8: Check Point CoreXL**.

```
 Configure Check Point CoreXL...
==========================
CoreXL is currently enabled with 3 firewall instances and 2 IPv6 firewall instances.

(1) Change the number of firewall instances
(2) Change the number of IPv6 firewall instances
(3) Disable Check Point CoreXL

(4) Exit
```

   The **Configuring Check Point CoreXL** menu shows how many IPv4 and IPv6 firewall instances are running on the processing cores.
4. Enter option **2: Change the number of IPv6 firewall instances**.

   The menu shows how many cores are available on the gateway.
5. Enter the *total* number of IPv6 firewall instances to run.

   You can only select a number from within the range shown.
6. Reboot the gateway.

   **Note** - In a clustered deployment, changing the number of kernel instances should be treated as a version upgrade.

**Example:**

A gateway that has four cores and is running three IPv4 instances of the firewall kernel and two IPv6 instances of the firewall kernel can be represented like this:

| Core | Firewall instances | IPv6 Firewall instances |
|---|---|---|
| CPU 0 | | |
| CPU 1 | fw4_2 | |
| CPU 2 | fw4_1 | fw6_1 |
| CPU 3 | fw4_0 | fw6_0 |
| | **3 instances of IPv4** | **2 instances of IPv6** |

- The minimum allowed number of IPv4 instances is two and the maximum four
- The minimum allowed number of IPv6 instances is two and the maximum is three

To increase the number of IPv6 instances to four, you must first increase the number of IPv4 firewall instances to the maximum of four:

```
How many firewall instances would you like to enable (2 to 4)[3] ? 4

CoreXL was enabled successfully with 4 firewall instances.
Important: This change will take effect after reboot.
```

The gateway now looks like this:

| Core | Firewall instances | IPv6 Firewall instances |
|---|---|---|
| CPU 0 | fw4_3 | |
| CPU 1 | fw4_2 | |
| CPU 2 | fw4_1 | fw6_1 |
| CPU 3 | fw4_0 | fw6_0 |
| | **4 instances of IPv4** | **2 instances of IPv6** |

Increase the number of IPv6 instances to four:

```
 How many IPv6 firewall instances would you like to enable (2 to 4)[2] ? 4

CoreXL was enabled successfully with 3 IPv6 firewall instances.
Important: This change will take effect after reboot.
```

The gateway now looks like this:

| Core | Firewall instances | IPv6 Firewall instances |
|---|---|---|
| CPU 0 | fw4_3 | fw6_3 |
| CPU 1 | fw4_2 | fw6_2 |
| CPU 2 | fw4_1 | fw6_1 |
| CPU 3 | fw4_0 | fw6_0 |
| | **4 instances of IPv4** | **4 instances of IPv6** |

# Performance Tuning

The following sections are relevant only for Gaia.

## Processing Core Allocation

The CoreXL software architecture includes the Secure Network Distributor (SND). The SND is responsible for:

- Processing incoming traffic from the network interfaces
- Securely accelerating authorized packets (if Performance Pack is running)
- Distributing non-accelerated packets among kernel instances.

Traffic entering network interface cards (NICs) is directed to a processing core running the SND. The association of a particular interface with a processing core is called the interface's *affinity* with that core. This affinity causes the interface's traffic to be directed to that core and the SND to run on that core. Setting a kernel instance or a process to run on a particular core is called the instance's or process's *affinity* with that core.

The default affinity setting for all interfaces is Automatic. Automatic affinity means that if Performance Pack is running, the affinity for each interface is automatically reset every 60 seconds, and balanced between available cores. If Performance Pack is not running, the default affinities of all interfaces are with one available core. In both cases, any processing core running a kernel instance, or defined as the affinity for another process, is considered unavailable and will not be set as the affinity for any interface.

In some cases, which are discussed in the following sections, it may be advisable to change the distribution of kernel instances, the SND, and other processes, among the processing cores. This is done by changing the affinities of different NICs (interfaces) and/or processes. However, to ensure CoreXL's efficiency, all interface traffic must be directed to cores not running kernel instances. Therefore, if you change affinities of interfaces or other processes, you will need to accordingly set the number of kernel instances and ensure that the instances run on other processing cores.

Under normal circumstances, it is not recommended for the SND and an instance to share a core. However, it is necessary for the SND and an instance to share a core when using a machine with exactly two cores.

## Allocating Processing Cores

In certain cases, it may be advisable to change the distribution of kernel instances, the SND, and other processes, among the processing cores. This section discusses these cases.

Before planning core allocation, make sure you have read the Processing Core Allocation (on page 23).

### *Adding Processing Cores to the Hardware*

Increasing the number of processing cores on the hardware platform does not automatically increase the number of kernel instances. If the number of kernel instances is not increased, CoreXL does not utilize some of the processing cores. After upgrading the hardware, increase the number of kernel instances with `cpconfig`.

Reinstalling the gateway will change the number of kernel instances if you have upgraded the hardware to an increased number of processing cores, or if the number of processing cores stays the same but the number of kernel instances was previously manually changed from the default. Run `cpconfig` from the Security Gateway command line to reconfigure the number of kernel instances.

In a clustered deployment, changing the number of kernel instances (such as by reinstalling CoreXL) should be treated as a version upgrade. Follow the instructions in the *Installation and Upgrade Guide* http://downloads.checkpoint.com/dc/download.htm?ID=54829.

See the "Upgrading ClusterXL Deployments" chapter, and perform either a Minimal Effort Upgrade (using network downtime) or a Zero Downtime Upgrade (no downtime, but active connections may be lost), substituting the instance number change for the version upgrade in the procedure. A Full Connectivity Upgrade cannot be performed when changing the number of kernel instances in a clustered environment.

## *Allocating an Additional Core to the SND*

In some cases, the default configuration of instances and the SND will not be optimal. If the SND is slowing the traffic, and your platform contains enough cores that you can afford to reduce the number of kernel instances, you may want to allocate an additional core to the SND. This is likely to occur especially if much of the traffic is of the type accelerated by Performance Pack, as Performance Pack fully accelerates packets when none of the Software Blades (except for VPN) is enabled. In this case, the task load of the SND may be disproportionate to that of the kernel instances.

To check if the SND is slowing down the traffic:

1. Identify the processing core to which the interfaces are directing traffic using **fw ctl affinity -l -r**.
2. Under heavy traffic conditions, run the **top** command on the CoreXL gateway and check the values for the different cores under the 'idle' column.

It is recommended to allocate an additional core to the SND only if all of the following conditions are met:

- Your platform has at least eight processing cores.
- The 'idle' value for the core currently running the SND is in the 0%-5% range.
- The sum of the 'idle' values for the cores running kernel instances is significantly higher than 100%.

If any of the above conditions are not met, the default configuration of one processing core allocated to the SND is sufficient, and no further configuration is necessary.

Allocating an additional processing core to the SND requires performing the following two stages in the order that they appear:

1. Reduce the number of kernel instances using **cpconfig**.
2. Set interface affinities to the remaining cores, as detailed below.
3. Reboot to implement the new configuration.

### *Setting Interface Affinities*

Check which cores are running the kernel instances. See also Allocating Processing Cores (on page 24). Allocate the remaining cores to the SND by setting interface affinities to the cores. The correct method of defining interface affinities depends on whether or not Performance Pack is running, as described in the following sections.

- When Performance Pack is Running

  If Performance Pack is running, interface affinities are handled by using the Performance Pack **sim affinity** command.

  The default **sim affinity** setting is Automatic. In the Performance Pack Automatic mode, interface affinities are automatically distributed among cores that are not running kernel instances and that are not set as the affinity for any other process.

  In most cases, you do not need to change the **sim affinity** setting.

- Setting Interface Affinities when Performance Pack is not Running

  If Performance Pack is not running, interface affinities are loaded at boot from a configuration text file called **fwaffinity.conf**, located under: **$FWDIR/conf** . In the text file, lines beginning with the letter **i** define interface affinities.

  If Performance Pack is running, interface affinities are defined by **sim affinity** settings, and lines beginning with **i** in **fwaffinity.conf** are ignored.

If you are allocating only one processing core to the SND, it is best to have that core selected automatically by leaving the default interface affinity set to automatic, and having no explicit core affinities for any interfaces. To do this, make sure **fwaffinity.conf** contains the following line:

**i default auto**

In addition, make sure that **fwaffinity.conf** contains no other lines beginning with **i**, so that no explicit interface affinities are defined. All interface traffic will be directed to the remaining core.

If you are allocating two processing cores to the SND, you need to explicitly set interface affinities to the two remaining cores. If you have multiple interfaces, you need to decide which interfaces to set for each of the two cores. Try to achieve a balance of expected traffic between the cores (you can later check the balance by using the **top** command).

To explicitly set interface affinities, when Performance Pack is not running:

1. Set the affinity for each interface by editing **fwaffinity.conf**. The file should contain one line beginning with **i** for each interface. Each of these lines should follow the following syntax:

   **i <interfacename> <cpuid>**

   where **<interfacename>** is the interface name, and **<cpuid>** is the number of the processing core to be set as the affinity of that interface.

   For example, if you want the traffic from **eth0** and **eth1** to go to core **#0**, and the traffic from **eth2** to go to core **#1**, create the following lines in **fwaffinity.conf**:

   **i eth0 0**

   **i eth1 0**

   **i eth2 1**

   Alternatively, you can choose to explicitly define interface affinities for only one processing core, and define the other core as the default affinity for the remaining interfaces, by using the word **default** for **<interfacename>**.

   In the case described in the previous example, the lines in **fwaffinity.conf** would be:

   **i eth2 1**

   **i default 0**
2. Run $FWDIR/scripts/fwaffinity_apply for the **fwaffinity.conf** settings to take effect.

The affinity of virtual interfaces can be set using their physical interface(s).

## *Allocating a Core for Heavy Logging*

If the gateway is performing heavy logging, it may be advisable to allocate a processing core to the **fwd** daemon, which performs the logging. Like adding a core for the SND, this too will reduce the number of cores available for kernel instances.

To allocate a processing core to the **fwd** daemon, you need to do two things:

1. Reduce the number of kernel instances using **cpconfig**.
2. Set the **fwd** daemon affinity, as detailed below.

### *Setting the fwd Daemon Affinity*

Check which processing cores are running the kernel instances and which cores are handling interface traffic using **fw ctl affinity -l -r**. Allocate the remaining core to the **fwd** daemon by setting the **fwd** daemon affinity to that core.

**Note**: Avoiding the processing core or cores that are running the SND is important only if these cores are explicitly defined as affinities of interfaces. If interface affinities are set to Automatic, any core that is not running a kernel instance can be used for the fwd daemon, and interface traffic will be automatically diverted to other cores.

Affinities for Check Point daemons (such as the **fwd** daemon), if set, are loaded at boot from the **fwaffinity.conf** configuration text file located at: **$FWDIR/conf** . Edit the file by adding the following line:

**n fwd <cpuid>**

where **<cpuid>** is the number of the processing core to be set as the affinity of the **fwd** daemon. For example, to set core **#2** as the affinity of the **fwd** daemon, add to the file:

**n fwd 2**

Reboot for the **fwaffinity.conf** settings to take effect.

# Configuring CoreXL

To enable/disable CoreXL:

1. Log in to the Security Gateway.
2. Run `cpconfig`
3. Select `Configure Check Point CoreXL`.
4. Enable or disable CoreXL.
5. Reboot the Security Gateway.

To configure the number of instances:

1. Run `cpconfig`
2. Select `Configure Check Point CoreXL`.
3. If CoreXL is enabled, enter the number of firewall instances.
   If CoreXL is disabled, enable CoreXL and then set the number of firewall instances.

Reboot the gateway.

**Note -** In a clustered deployment, changing the number of kernel instances should be treated as a version upgrade.

# Command Line Reference

## Affinity Settings

Affinity settings controlled by the *fwaffinity_apply* script file, which executes automatically at boot. When you make a change to affinity settings, the settings will not take effect until you either reboot or manually execute the *fwaffinity_apply* script.

*fwaffinity_apply* executes affinity definitions according to the information in the *fwaffinity.conf* text file. To change affinity settings, edit the text file.

**Note** - If Performance Pack is running, interface affinities are only defined by the Performance Pack *sim affinity* command. The *fwaffinity.conf* interface affinity settings are ignored.

## fwaffinity.conf

*fwaffinity.conf* is located in the *$FWDIR/conf* directory.

**Syntax**

Each line in the text file uses the same format: *<type> <id> <cpu>*

| Data | Values | Description |
|---|---|---|
| **<type>** | **i** | interface |
| | **n** | Check Point daemon |
| | **k** | kernel instance |
| **<id>** | interface name | if **<type>** = **i** |
| | daemon name | if **<type>** = **n** |
| | instance number | if **<type>** = **k** |
| | **default** | interfaces that are not specified in another line |
| **<cpuid>** | <number> | number(s) of processing core(s) to be set as the affinity |
| | **all** | all processing cores are available to the interface traffic, daemon or kernel instance |
| | **ignore** | no specified affinity (useful for excluding an interface from a **default** setting) |
| | **auto** | Automatic mode See also Processing Core Allocation (on page 23). |

**Note** - Interfaces that share an IRQ cannot have different cores as their affinities, including when one interface is included in the **default** affinity setting. Either set both interfaces to the same affinity, or use **ignore** for one of them. To view the IRQs of all interfaces, run: *fw ctl affinity -l -v -a* .

## fwaffinty_apply

*fwaffinity_apply* is located in the **$FWDIR/scripts** directory. Use the following syntax to execute the command: *$FWDIR/scripts/fwaffinity_apply <option>*

where *<option>* is one of the following parameters:

| Parameter | Description |
|---|---|
| **-q** | Quiet mode - print only error messages. |
| **-t <type>** | Only apply affinity for the specified type. |
| -f | Sets interface affinity even if automatic affinity is active. |

## fw ctl affinity

The *fw ctl affinity* command controls affinity settings. However, *fw ctl affinity* settings will not persist through a restart of the Security Gateway.

To set affinities, execute *fw ctl affinity -s.*

To list existing affinities, execute *fw ctl affinity -l.*

### *fw ctl affinity -s*

Use this command to set affinities.

*fw ctl affinity -s* settings are not persistent through a restart of the Security Gateway. If you want the settings to be persistent, either use *sim affinity* or edit the *fwaffinity.conf* configuration file.

To set interface affinities, you should use *fw ctl affinity* only if Performance Pack is not running. If Performance Pack is running, you should set affinities by using the Performance Pack *sim affinity* command. These settings will be persistent. If the Performance Pack *sim affinity* is set to Automatic mode (even if Performance Pack was subsequently disabled), you will not be able to set interface affinities by using *fw ctl affinity -s*.

#### *Syntax*

```
fw ctl affinity -s <proc_selection> <cpuid>
```

<proc_selection> is one of the following parameters:

| Parameter | Description |
|---|---|
| -p <pid> | Sets affinity for a particular process, where <pid> is the process ID#. |
| -n <cpdname> | Sets affinity for a Check Point daemon, where <cpdname> is the Check Point daemon name (for example: **fwd**). |
| -k <instance> | Sets affinity for a kernel instance, where <instance> is the instance's number. |
| -i <interfacename> | Sets affinity for an interface, where <interfacename> is the interface name (for example: *eth0*). |

<cpuid> should be a processing core number or a list of processing core numbers. To have no affinity to any specific processing core, <cpuid> should be**: all**.

**Note** - Setting an Interface Affinity will set the affinities of all interfaces sharing the same IRQ to the same processing core.
To view the IRQs of all interfaces, run: `fw ctl affinity -l -v -a`

***Example***

To set kernel instance #3 to run on processing core #5, run:

```
fw ctl affinity -s -k 3 5
```

## *fw ctl affinity -l*

Use this command to list existing affinities. For an explanation of kernel, daemon and interface affinities, see CoreXL Administration (on page 20).

***Syntax***

```
fw ctl affinity -l [<proc_selection>] [<listtype>]
```

If <proc_selection> is omitted, `fw ctl affinity -l` lists affinities of all Check Point daemons, kernel instances and interfaces. Otherwise, <proc_selection> is one of the following parameters:

| Parameter | Description |
|---|---|
| `-p <pid>` | Displays the affinity of a particular process, where <pid> is the process ID#. |
| `-n <cpdname>` | Displays the affinity of a Check Point daemon, where <cpdname> is the Check Point daemon name (for example: *fwd*). |
| `-k <instance>` | Displays the affinity of a kernel instance, where <instance> is the instance's number. |
| `-i <interfacename>` | Displays the affinity of an interface, where <interfacename> is the interface name (for example: *eth0*). |

If <listtype> is omitted, `fw ctl affinity -l` lists items with specific affinities, and their affinities. Otherwise, <listtype> is one or more of the following parameters:

| Parameter | Description |
|---|---|
| `-a` | All: includes items without specific affinities. |
| `-r` | Reverse: lists each processing core and the items that have it as their affinity. |
| `-v` | Verbose: list includes additional information. |

***Example***

To list complete affinity information for all Check Point daemons, kernel instances and interfaces, including items without specific affinities, and with additional information, run:

*fw ctl affinity -l -a -v*

## fw ctl multik stat

The `fw ctl multik stat` and `fw6 ctl multik stat` (multi-kernel statistics) commands show information for each kernel instance. The state and processing core number of each instance is displayed, along with:

- The number of connections currently being handled.
- The peak number of concurrent connections the instance has handled since its inception.

# Multi-Queue

*In This Section:*



## Introduction to Multiple Traffic Queues

By default, each network interface has one traffic queue handled by one CPU. You cannot use more CPUs for acceleration than the number of interfaces handling traffic. Multi-Queue lets you configure more than one traffic queue for each network interface. For each interface, more than one CPU is used for acceleration.

### Multi-Queue Requirements and Limitations

- Multi-Queue is not supported on single core computers.
- Network interfaces must support Multi-Queue
- The number of queues is limited by the number of CPUs and the type of interface driver:

| Driver type | Maximum number of rx queues |
|---|---|
| lgb | 4 |
| lxgbe | 16 |
| mlx5_core | 10 |

### Deciding if Multi-Queue is needed

This section will help you decide if you can benefit from configuring Multi-Queue. We recommend that you do these steps before configuring Multi-Queue:

- Make sure that SecureXL is enabled
- Examine the CPU roles allocation
- Examine CPU Utilization
- Decide if more CPUs can be allocated to the SND
- Make sure that network interfaces support Multi-Queue

### Making sure that SecureXL is enabled

1. On the Security Gateway, run: `fwaccel stat`
2. Examine the **Accelerator Status** value:

```
 [Expert@gw-30123d:0]# fwaccel stat
Accelerator Status : on
Accept Templates   : enabled
Drop Templates     : disabled
NAT Templates      : disabled by user

Accelerator Features : Accounting, NAT, Cryptography, Routing,
                       HasClock, Templates, Synchronous, IdleDetection,
                       Sequencing, TcpStateDetect, AutoExpire,
                       DelayedNotif, TcpStateDetectV2, CPLS, WireMode,
                       DropTemplates, NatTemplates, Streaming,
                       MultiFW, AntiSpoofing, DoS Defender, ViolationStats,
                       Nac, AsychronicNotif, ERDOS
Cryptography Features : Tunnel, UDPEncapsulation, MD5, SHA1, NULL,
                        3DES, DES, CAST, CAST-40, AES-128, AES-256,
                        ESP, LinkSelection, DynamicVPN, NatTraversal,
                        EncRouting, AES-XCBC, SHA256
```

SecureXL is enabled if the value of this field is: *on*.

**Note** - Multi-Queue is relevant only if SecureXL is enabled.

### Examining the CPU roles allocation

To see the CPU roles allocation, run: `fw ctl affinity –l`

This command shows the CPU affinity of the interfaces, which assigns SND CPUs. It also shows the CoreXL firewall instances CPU affinity. For example, if you run the command on a Security Gateway:

```
[Expert@gw-30123d:0]# fw ctl affinity -l
Mgmt: CPU 0
eth1-05: CPU 0
eth1-06: CPU 1
fw_0: CPU 5
fw_1: CPU 4
fw_2: CPU 3
fw_3: CPU 2
```

In this example:

- The SND is running on CPU 0 and CPU1
- CoreXL firewall instances are running on CPUs 2-5

If you run the command on a VSX gateway:

```
 [Expert@gw-30123d:0]# fw ctl affinity -l
Mgmt: CPU 0
eth1-05: CPU 0
eth1-06: CPU 1
VS_0 fwk: CPU 2 3 4 5
VS_1 fwk: CPU 2 3 4 5
```

In this example:

- The SND is running on CPU 0-1
- CoreXL firewall instances (part of fwk processes) of all the Virtual System are running on CPUs 2-5.

## Examining CPU Utilization

1. On the Security Gateway, run: `top`.
2. Press 1 to toggle the SMP view.

   This shows the usage and idle percentage for each CPU. For example:

```
 top - 18:02:33 up 28 days,  1:18,  1 user, load average: 1.22, 1.38, 1.48
Tasks:  137 total,  3 running,  134 sleping,  0 stopped,  0 zombie

Cpu0 : 2.0%us,  0.0%sy,  0.0%ni,  42.7%id,  5.9%wa,  0.0%hi,  49.4%si,  0.0%st
Cpu1 : 0.0%us,  1.0%sy,  0.0%ni,  55.2%id,  0.0%wa,  0.0%hi,  43.8%si,  0.0%st
Cpu2 : 2.0%us,  2.0%sy,  0.0%ni,  45.5%id,  0.0%wa,  4.0%hi,  46.5%si,  0.0%st
Cpu3 : 1.0%us,  2.0%sy,  0.0%ni,  74.5%id,  0.0%wa,  0.0%hi,  22.5%si,  0.0%st
Cpu4 : 5.0%us,  1.0%sy,  0.0%ni,  42.6%id,  0.0%wa,  0.0%hi,  51.5%si,  0.0%st

Mem:  12224020k total, 70005820k used,   5218200k free, 273536k buffers
Swap: 14707496k total,        0k used,  14707496k free, 484340k cached

 PID USER   PR  NI  VIRT  RES  SHR  S  %CPU  %MEM   TIME+   COMMAND
3301 root   15   0     0    0    0  R    17   0.0  2747:04  [fw_worker_3]
3326 root   15   0     0    0    0  R    16   0.0  2593:35  [fw_worker_0]
```

In this example:

- SND CPUs (CPU0 and CPU1) are approximately 30% idle
- CoreXL firewall instances CPUs are approximately 70% idle

## Deciding if more CPUs can be allocated to the SND

If you have more network interfaces handling traffic than CPUs assigned to the SND , you can allocate more CPUs for SND. For example, if you have the following network interfaces:

- eth1-04 – connected to an internal network
- eth1-05 – connected to an internal network
- eth1-06 – connected to the DMZ
- eth1-07 – connected to the external network

And running `fw ctl affinity -l` shows this IRQ affinity:

```
[Expert@gw-30123d:0]# fw ctl affinity -l
Mgmt: CPU 0
eth1-04: CPU 1
eth1-05: CPU 0
eth1-06: CPU 1
eth1-07: CPU 0
fw_0: CPU 5
fw_1: CPU 4
fw_2: CPU 3
fw_3: CPU 2
```

You can use the Sim affinity utility to change an interface's IRQ affinity to use more CPUs for the SND. You can do this:

- Even before the Multi-Queue feature is activated
- If you have more network interfaces handling traffic than CPUs assigned to the SND

## Making sure that the network interfaces support Multi-Queue

Multi-Queue is supported only on network cards that use **igb** (1Gb), **ixgbe** (10Gb), or **mlx5_core** (40Gb) drivers. Before upgrading these drivers, make sure that the latest version supports Multi-Queue.

| Gateway type | Expansion Card Model |
|---|---|
| Security Appliance | Multi-Queue is supported on these expansion cards for 4000, 12000, and 21000 appliances:<br>• CPAC-ACC-4-1C<br>• CPAC-ACC-4-1F<br>• CPAC-ACC-8-1C<br>• CPAC-ACC-2-10F<br>• CPAC-ACC-4-10F |
| Security Appliance | Multi-Queue is supported on this expansion card for 5000, 13000, and 23000 appliances: CPAC-2-40F-B |
| IP appliance | The XMC 1Gb card is supported on:<br>• IP1280<br>• IP2450 |
| Open server | Network cards that use **igb** (1Gb), **ixgbe** (10Gb), or **mlx5_core** (40Gb) drivers |

- To view which driver an interface is using, run: `ethtool -i <interface name>`.
- When installing a new interface that uses the **igb** or **ixgbe** driver, run: `cpmq reconfigure` and reboot.

## Recommendation

We recommend configuring Multi-Queue when:

- CPU load for SND is high (idle is less than 20%) and
- CPU load for CoreXL firewall instances are low (idle is greater than 50%)
- You cannot assign more CPUs to the SND by changing interface IRQ affinity

# Basic Multi-Queue Configuration

The *cpmq* utility is used to view or change the current Multi-Queue configuration.

## Configuring Multi-Queue

The `cpmq set` command lets you configure Multi-Queue on supported interfaces.

### To configure Multi-Queue:

- On the gateway, run: `cpmq set`

  This command:
  - Shows all supported interfaces that are active
  - Lets you change the Multi-Queue configuration for each interface.

  Network interfaces that are down are not in the output.

  **Note** -
  - Multi-Queue lets you configure a maximum of five interfaces
  - You must reboot the gateway after changing the Multi-Queue configuration

## Querying the current Multi-Queue configuration

The *cpmq get* command shows the Multi-Queue status of supported interfaces.

### To see the Multi-Queue configuration:

Run: `cpmq get [-a]`

The `-a` option shows the Multi-Queue configuration for all supported interfaces (both active and inactive). For example:

```
[Expert@gw-30123d:0]# cpmq get -a

Active igb interfaces:
eth1-05 [On]
eth1-06 [Off]
eth1-01 [Off]
eth1-03 [Off]
eth1-04 [On]

Non active igb interfaces:
eth1-02 [Off]
```

**Status messages**

| Status | Meaning |
|---|---|
| On | Multi-Queue is enabled on the interface. |
| Off | Multi-Queue is disabled on the interface. |
| Pending On | Multi-Queue currently disabled. Multi-Queue will be enabled on this interface only after rebooting the gateway.<br>**Note**: *Pending on* can also indicate bad configuration or system errors. For more, see the section on troubleshooting (on page 43). |
| Pending Off | Multi-Queue enabled. Multi-Queue will be disabled on this interface only after rebooting the gateway. |

In this example:

- Two interfaces are up with Multi-Queue enabled
  (eth1-05, eth1-04)
- Three interfaces are up with Multi-Queue disabled
  (eth1-06, eth1-01, eth1-03)
- One interface that supports Multi-Queue is down
  (eth1-02)

Running the command without the `-a` option shows the active interfaces only.

# Multi-Queue Administration

There are two main roles for CPUs applicable to SecureXL and CoreXL:

- SecureXL and CoreXL dispatcher CPU (the SND - Secure Network Distributor)
  You can manually configure this using the `sim affinity -s` command.
- CoreXL firewall instance CPU
  You can manually configure this using the `fw ctl affinity` command.

For best performance, the same CPU should not work in both roles. During installation, a default CPU role configuration is set. For example, on a twelve core computer, the two CPUs with the lowest CPU ID are set as SNDs and the ten CPUs with the highest CPU IDs are set as CoreXL firewall instances.

Without Multi-Queue, the number of CPUs allocated to the SND is limited by the number of network interfaces handling the traffic. Since each interface has one traffic queue, each queue can be handled by only one CPU at a time. This means that the SND can use only one CPU at a time per network interface.

When most of the traffic is accelerated, the CPU load for SND can be very high while the CPU load for CoreXL firewall instances can be very low. This is an inefficient utilization of CPU capacity.

Multi-Queue lets you configure more than one traffic queue for each supported network interface, so that more than one SND CPU can handle the traffic of a single network interface at a time. This balances the load efficiently between SND CPUs and CoreXL firewall instances CPUs.

# Advanced Multi-Queue settings

Advanced Multi-Queue settings include:

- Controlling the number of queues
- IRQ Affinity
- Viewing CPU Utilization

## Controlling the number of queues

Controlling the number of queues depends on the driver type:

| Driver type | Queues | Recommended number of rx queues |
|---|---|---|
| **ixgbe** | • When configuring Multi-Queue for an **ixgbe** interface, an RxTx queue is created per CPU. You can control the number of active rx queues using `rx_num`.<br>• All tx queues are active. | 16 |
| **igb** | When configuring Multi-Queue for an **igb** interface, the number of tx and rx queues is calculated by the number of active rx queues. | 4 |
| **mlx5_core** | When configuring Multi-Queue for an **mlx5_core** interface, the number of tx and rx queues is calculated by the number of active rx queues with a maximum queue value set to 10 | 10 |

- By default on a Security Gateway, the number of active rx queues is calculated by:
  *active rx queues = Number of CPUs – number of CoreXL firewall instances*
- By default on a VSX gateway, the number of active rx queues is calculated by:
  *active rx queues = the lowest CPU ID that an fwk process is assigned to*

### To control the number of active rx queues:

Run: `cpmq set rx_num <igb/ixgbe/mlx5_core> <number of active rx queues>`

This command overrides the default value.

### To view the number of active rx queues:

Run: `cpmq get rx_num <igb/ixgbe/mlx5_core>`

### To return to the recommended number of rx queues:

On a Security Gateway, the number of active queues changes automatically when you change the number of CoreXL firewall instances (using `cpconfig`). This number of active queues does not change if you configure the number of rx queues manually.

Run: `cpmq set rx_num <igb/ixgbe/mlx5_core> default`

## IRQ Affinity

The IRQ affinity of the queues is set automatically when the operating system boots, as shown (rx_num set to 3):

rxtx-0 -> CPU 0

rxtx-1 -> CPU 1

rxtx-2 -> CPU 2

and so on. This is also true in cases where rx and tx queues are assigned with a separated IRQ:

rx-0 -> CPU 0

tx-0 -> CPU 0

rx-1 -> CPU 1

tx-1 -> CPU 1

and so on.

- You cannot use the `sim affinity` or the `fw ctl affinity` commands to change and query the IRQ affinity for Multi-Queue interfaces.
- You can reset the affinity of Multi-Queue IRQs by running: `cpmq set affinity`
- You can view the affinity of Multi-Queue IRQs by running: `cpmq get -v`

**Important** - Do not change the IRQ affinity of queues manually. Changing the IRQ affinity of the queues manually can affect performance.

## Viewing CPU Utilization

1. Find the CPUs assigned to Multi-Queue IRQs by running: `cpmq get -v`. For example:

```
[Expert@gw-426e82:0]# cpmq get  -v

Active mlx5_core interfaces:
eth2-01 [On]

Active i40e interfaces:
eth5-01 [On]
eth5-02 [Off]

Active ixgbe interfaces:
eth4-01 [On]
eth4-02 [On]

Active igb interfaces:
Mgmt [On]

The rx_num for mlx5_core is: 10 (default)
The rx_num for i40e is: 10
The rx_num for ixgbe is: 16 (default)
The rx_num for igb is: 2

multi-queue affinity for mlx5_core interfaces:
CPU | TX | Vector                        | RX Bytes
-------------------------------------------------------------
0   | 0  | eth2-01-0 (211)               | 0
1   | 2  | eth2-01-2 (227)               | 0
2   | 4  | eth2-01-4 (52)                | 0
3   | 6  | eth2-01-6 (68)                | 0
4   | 8  | eth2-01-8 (84)                | 0
5   | 10 |                               |
```

```
multi-queue affinity for i40e interfaces:
CPU | TX | Vector                        | RX Bytes
-------------------------------------------------------------
0   | 0  | i40e-eth5-01-TxRx-0 (99)     | 0
1   | 2  | i40e-eth5-01-TxRx-2 (115)    | 0
2   | 4  | i40e-eth5-01-TxRx-4 (131)    | 0
3   | 6  | i40e-eth5-01-TxRx-6 (147)    | 0
4   | 8  | i40e-eth5-01-TxRx-8 (163)    | 0
5   | 0  |                              |

multi-queue affinity for ixgbe interfaces:
CPU | TX | Vector                        | RX Bytes
-------------------------------------------------------------
0   | 0  | eth4-01-TxRx-0 (156)         | 0
    |    | eth4-02-TxRx-0 (157)         |
1   | 2  | eth4-01-TxRx-2 (172)         | 0
    |    | eth4-02-TxRx-2 (173)         |
2   | 4  | eth4-01-TxRx-4 (188)         | 0
    |    | eth4-02-TxRx-4 (189)         |
3   | 6  | eth4-01-TxRx-6 (204)         | 0
    |    | eth4-02-TxRx-6 (205)         |
4   | 8  | eth4-01-TxRx-8 (220)         | 0
    |    | eth4-02-TxRx-8 (221)         |
5   | 10 | eth4-01-TxRx-10 (236)        | 0
    |    | eth4-02-TxRx-10 (237)        |
6   | 12 | eth4-01-TxRx-12 (61)         | 0
    |    | eth4-02-TxRx-12 (62)         |
7   | 14 | eth4-01-TxRx-14 (77)         | 0
    |    | eth4-02-TxRx-14 (78)         |
```

2. Run: `top`
3. Press 1 to toggle to the SMP view.

```
top - 18:02:33 up 28 days,  1:18,  1 user, load average: 1.22, 1.38, 1.48
Tasks:  137 total,  3 running,  134 sleeping,  0 stopped,  0 zombie

Cpu0 : 2.0%us, 0.0%sy, 0.0%ni, 42.7%id, 5.9%wa, 0.0%hi, 49.4%si, 0.0%st
Cpu1 : 0.0%us, 1.0%sy, 0.0%ni, 55.2%id, 0.0%wa, 0.0%hi, 43.8%si, 0.0%st
Cpu2 : 2.0%us, 2.0%sy, 0.0%ni, 45.5%id, 0.0%wa, 4.0%hi, 46.5%si, 0.0%st
Cpu3 : 1.0%us, 2.0%sy, 0.0%ni, 74.5%id, 0.0%wa, 0.0%hi, 22.5%si, 0.0%st
Cpu4 : 5.0%us, 1.0%sy, 0.0%ni, 42.6%id, 0.0%wa, 0.0%hi, 51.5%si, 0.0%st

Mem:  12224020k total, 70005820k used,   5218200k free, 273536k buffers
Swap: 14707496k total,        0k used,  14707496k free, 484340k cached

 PID USER   PR  NI  VIRT  RES  SHR  S  %CPU  %MEM   TIME+   COMMAND
3301 root   15   0     0    0    0  R    17   0.0  2747:04  [fw_worker_3]
3326 root   15   0     0    0    0  R    16   0.0  2593:35  [fw_worker_0]
```

In the above example, CPU utilization of Multi-Queue CPUs is approximately 50%, as CPU0 and CPU1 are handling the queues (as shown in step 1).

For more information, use the `-vv` (very verbose) option. For example:

```
 [Expert@gw-426e82:0]# cpmq get -vv

Active i40e interfaces:
eth5-01 [On]
eth5-02 [Off]

Active ixgbe interfaces:
eth4-01 [On]
eth4-02 [On]

Active igb interfaces:
```

```
Mgmt [On]

The rx_num for i40e is: 10
The rx_num for ixgbe is: 16 (default)
The rx_num for igb is: 2

multi-queue affinity for i40e interfaces:
CPU | TX | Vector                      | RX Packets          | RX Bytes
-------------------------------------------------------------------------
0   | 0  | i40e-eth5-01-TxRx-0 (220)   | 0                   | 0
1   | 2  | i40e-eth5-01-TxRx-2 (236)   | 0                   | 0
2   | 4  | i40e-eth5-01-TxRx-4 (61)    | 0                   | 0
3   | 6  | i40e-eth5-01-TxRx-6 (77)    | 0                   | 0
4   | 8  | i40e-eth5-01-TxRx-8 (93)    | 0                   | 0
5   | 0  |                             |                     |

multi-queue affinity for ixgbe interfaces:
CPU | TX | Vector                      | RX Packets          | RX Bytes
-------------------------------------------------------------------------
0   | 0  | eth4-01-TxRx-0 (234)        | 0                   | 0
    |    | eth4-02-TxRx-0 (187)        |                     |
1   | 2  | eth4-01-TxRx-2 (59)         | 0                   | 0
    |    | eth4-02-TxRx-2 (203)        |                     |
2   | 4  | eth4-01-TxRx-4 (75)         | 0                   | 0
    |    | eth4-02-TxRx-4 (219)        |                     |
3   | 6  | eth4-01-TxRx-6 (91)         | 0                   | 0
    |    | eth4-02-TxRx-6 (235)        |                     |
4   | 8  | eth4-01-TxRx-8 (107)        | 0                   | 0
    |    | eth4-02-TxRx-8 (60)         |                     |
5   | 10 | eth4-01-TxRx-10 (123)       | 0                   | 0
    |    | eth4-02-TxRx-10 (76)        |                     |
6   | 12 | eth4-01-TxRx-12 (139)       | 0                   | 0
    |    | eth4-02-TxRx-12 (92)        |                     |
7   | 14 | eth4-01-TxRx-14 (155)       | 0                   | 0
    |    | eth4-02-TxRx-14 (108)       |                     |


multi-queue affinity for igb interfaces:
CPU | TX | Vector                      | RX Packets          | RX Bytes
-------------------------------------------------------------------------
0   | 0  | Mgmt-TxRx-0 (172)           | 2752                | 176674
1   | 0  |                             |                     |
```

## Adding more Interfaces

Due to IRQ limitations, you can configure a maximum of five interfaces with Multi-Queue.

To add more interfaces, run: `cpmq set -f`

# Special Scenarios and Configurations

- **In Security Gateway mode**: Changing the number of CoreXL firewall instances when Multi-Queue is enabled on some or all interfaces

  For best performance, the default number of active rx queues is calculated by:

  *Number of active rx queues = number of CPUs – number of CoreXL firewall instances*

  This configuration is set automatically when configuring Multi-Queue. When changing the number of instances, the number of active rx queues will change automatically if it was not set manually.

- **In VSX mode**: changing the number of CPUs that the `fwk` processes are assigned to

- The default number of active rx queues is calculated by:

  *Number of active rx queues = the lowest CPU ID that an fwk process is assigned to*

  For example:

```
 [Expert@gw-30123d:0]# fw ctl affinity -l
Mgmt: CPU 0
eth1-05: CPU 0
eth1-06: CPU 1
VS_0 fwk: CPU 2 3 4 5
VS_1 fwk: CPU 2 3 4 5
```

  In this example

  - The number of active rx queues is set to 2.
  - This configuration is set automatically when configuring Multi-Queue.
  - It will not automatically update when changing the affinity of the Virtual System. When changing the affinity of the Virtual System, make sure to follow the instructions in Advanced Multi-Queue settings (on page 38).

### The effects of changing the status of a Multi-Queue enabled interface

- **Changing the status to DOWN**

  The Multi-Queue configuration is saved when you change the status of an interface to down.

  Since the number of interfaces with Multi-Queue enabled is limited to five, you may need to disable Multi-Queue on an interface after changing its status to down to enable Multi-Queue on other interfaces.

- **To disable Multi-Queue on non-active interfaces**:

  a) Activate an interface.

  b) Disable the Multi-Queue using the `cpmq set` command.

  c) Deactivate the interface.

- **Changing the status to UP**

  You must reset the IRQ affinity for Multi-Queue interfaces if, *in this order*, you:

  - Enabled Multi-Queue on the interface
  - Changed the status of the interface to down
  - Rebooted the gateway
  - Changed the interface status to up.

This problem does not occur if you are running automatic sim affinity (`sim affinity -a`). Automatic sim affinity runs by default, and has to be manually canceled using the `sim affinity -s` command.

To set the static affinity of Multi-Queue interfaces again, run: `cpmq set affinity`.

### Adding a network interface

- When adding a network interface card to a gateway that uses **igb** or **ixgbe** drivers, the Multi-Queue configuration can change due to interface indexing. If you add a network interface card to a gateway that uses **igb** or **ixgbe** drivers make sure to run Multi-Queue configuration again or run: `cpmq reconfigure`.
- If a reconfiguration change is required, you will be prompted to reboot the computer.

### Changing the affinity of CoreXL firewall instances

- For best performance, we recommend that you do not assign both SND and a CoreXL firewall instance to the same CPU.
- When changing the affinity of the CoreXL firewall instances to a CPU assigned with one of the Multi-Queue queues, we recommend that you reconfigure the number of active rx queues following this rule:

  *Active rx queues = the lowest CPU number that a CoreXL firewall instance is assigned to*
- You can configure the number of active rx queues by running:
  `cpmq set rx_num <igb/ixgbe> <value/default>`

# Troubleshooting

- **After reboot, the wrong interfaces are configured for Multi-Queue**

  This can happen after changing the physical interfaces on the gateway. To solve this issue:

  a) Run: `cpmq reconfigure`

  b) Reboot.

  Or configure Multi-Queue again.
- **After configuring Multi-Queue and rebooting the gateway, some of the configured interfaces are shown as** *down*. **These interfaces were** *up* **before the gateway reboot. The** `cpmq get –a` **command shows the interface status as** *Pending on*.

  This can happen when not enough IRQs are available on the gateway. To resolve this issue do one of these:
  - Disable some of the interfaces configured for Multi-Queue
  - Manually reduce the number of active rx queues (`rx_num`) using the `cpmq set rx_num` command, and reboot the gateway

- **When changing the status of interfaces, all the interface IRQs are assigned to CPU 0 or to all of the CPUs**

  This can happen when an interface status is changed to UP after the automatic affinity procedure runs (the affinity procedure runs automatically during boot).

  To solve this issue, run: `cpmq set affinity`

  This problem does not occur if you are running automatic sim affinity (`sim affinity -s`). Automatic sim affinity runs by default, and has to be manually canceled using the `sim affinity -s` command.

- **In VSX mode, an fwk process runs on the same CPU as some of the interface queues**

  This can happen when the affinity of the Virtual System was manually changed but Multi-Queue was not reconfigured accordingly.

  To solve this issue, configure the number of active rx queues manually or run: `cpmq reconfigure` and reboot.

- **In Security Gateway mode – after changing the number of instances Multi-Queue is disabled on all interfaces**

  When changing the number of CoreXL firewall instances, the number of active rx queues automatically changes according to this rule (if not configured manually):

  *Active rx queues = Number of CPUs – number of CoreXL firewall instances*

  **For R77**

  If the number of instances is equal to the number of CPUs, or if the difference between the number of CPUs and the number of CoreXL firewall instances is 1, Multi-Queue will be disabled. To solve this issue, configure the number of active rx queues manually by running:

  `cpmq set rx_num <igb/ixgbe> <value>`

  **For R77.10 and R77.20**

  If the difference between the number of CPUs and the number of CoreXL firewall instances is 1, Multi-Queue will be disabled. To solve this issue, configure the number of active rx queues manually by running:

  `cpmq set rx_num <igb/ixgbe> <value>`

  **For R77.30 and Higher**

  If the difference between the number of CPUs and the number of CoreXL firewall instances is 1, Multi-Queue will be disabled. To solve this issue, configure the number of active rx queues manually by running:

  `cpmq set rx_num <igb/ixgbe/mlx5_core> <value>`

  **Note -** Only **R77.30** and higher supports *mlx5_core* drivers.