© 2011 Check Point Software Technologies Ltd.

All rights reserved. This product and related documentation are protected by copyright and distributed under licensing restricting their use, copying, distribution, and decompilation. No part of this product or related documentation may be reproduced in any form or by any means without prior written authorization of Check Point. While every precaution has been taken in the preparation of this book, Check Point assumes no responsibility for errors or omissions. This publication and features described herein are subject to change without notice.

RESTRICTED RIGHTS LEGEND:

Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

TRADEMARKS:

Refer to the Copyright page (http://www.checkpoint.com/copyright.html) for a list of our trademarks.

Refer to the Third Party copyright notices (http://www.checkpoint.com/3rd_party_copyright.html) for a list of relevant copyrights and third-party licenses.
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.

Latest Documentation
The latest version of this document is at: http://supportcontent.checkpoint.com/documentation_download?ID=11877
For additional technical information, visit the Check Point Support Center (http://supportcenter.checkpoint.com).

Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/9/2011</td>
<td>First release of this document</td>
</tr>
</tbody>
</table>

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 How To Use ADP SecureXL on IPSO ).
# Contents

- Important Information ................................................................. 3
- How To Configure ADP & SecureXL on IPSO ........................................... 5
  - Objective .................................................................................. 5
  - Supported Versions ..................................................................... 5
  - Supported OS ........................................................................... 5
  - Supported Appliances ................................................................. 5
- Before You Start ............................................................................... 6
  - Requirements ............................................................................. 6
  - Related Documentation and Assumed Knowledge ......................... 6
  - Impact on the Environment and Warnings ....................................... 6
- Background Information .................................................................. 6
  - What is ADP? .............................................................................. 6
  - Key ADP Components .................................................................. 6
  - ADP Communication between IPSO and Firewall-1 ......................... 7
  - What is SecureXL? ..................................................................... 7
  - How SecureXL Accelerates Network Traffic .................................... 7
  - Templates .................................................................................... 10
  - When Templates are Created ....................................................... 10
  - When Templates are used to Accelerate Traffic .............................. 10
  - Why Templates are Disabled ....................................................... 10
  - What ADP Accelerates ................................................................ 11
  - Limitations ................................................................................ 12
  - Firewall Rule Limitations .......................................................... 12
  - ADP Limitations ......................................................................... 12
  - Traffic Limitations ..................................................................... 12
  - Environment Limitations ............................................................ 13
  - Advanced SecureXL Configurations (IPSO 6.x) ................................. 13
    - Fast Expiry .............................................................................. 13
    - How Fast Expiry Works ............................................................ 14
- Procedures ......................................................................................... 14
  - Determining the Status of SecureXL .............................................. 15
  - Viewing Additional SecureXL Statistics ........................................ 15
  - Enabling Fast-Expire ................................................................... 16
  - DNS ............................................................................................ 16
  - DHCP ......................................................................................... 16
  - TFTP .......................................................................................... 16
  - Identifying & Monitoring Memory Leaks in ADP .............................. 17
  - Identifying & Troubleshooting ADP Hung Conditions ..................... 17
    - Information Required for Troubleshooting these Conditions ........... 18
  - Installing a Replacement ADP (kana) Kernel .................................... 19
How To Configure ADP & SecureXL on IPSO

Objective

This document is intended to give an overview of Accelerated Data Path (ADP) technology, Secure XL security performance architecture, and how to use them to accelerate network traffic. It then lists some of the main procedures required to achieve this acceleration.

Supported Versions

- IPSO 3.8 and later

Supported OS

- IPSO 3.8 and later

Supported Appliances

- All IP appliances
Before You Start

Requirements

- IPSO 3.8 and later
- FW-1 NG R55P or later

Related Documentation and Assumed Knowledge

- This document is for advanced users who are familiar with the UNIX command line.

Impact on the Environment and Warnings

- Changes to SecureXL should always be carried out during planned downtime, as connections may be lost.

Background Information

In this section:

What is ADP? 6
What is SecureXL? 7
Templates 10
What ADP Accelerates 11
Limitations 12
Advanced SecureXL Configurations (IPSO 6.x) 13

What is ADP?

Accelerated Data Path (ADP) is a key technology for security and high-performance networks. ADP relies on specific hardware and software elements to achieve acceleration of network throughput and connection rates. The main function of ADP is to forward all packet sizes at the highest possible rate.

Key ADP Components

The key components of ADP are network processors (NPs), network interface controllers, and the operating system (OS) module specific to the NPs that handle the fast-path forwarding. An ADP subsystem comprises a network processor and its OS module.

HyperTransport allows communication between the ADP subsystems on multi-ADP units. The HyperTransport is a bidirectional serial/parallel high-bandwidth, low-latency bus. It connects the NPs, and transfers data between them inside a device for situations when packets on one ADP subsystem are to be transmitted to a port connected to another Check Point ADP subsystem.

For optimal port usage and highest performance, it is generally best to have traffic enter and leave the ports associated with the same ADP card, rather than going through both network processors through the HyperTransport.
ADP Communication between IPSO and Firewall-1

IPSO and Firewall-1 reside within the control processor (CP). The ADP subsystems and the control processor communicate with each other over the Peripheral Component Interconnect (PCI).

The PCI specifies a bus for attaching peripheral devices to the motherboard. The ADP subsystem retrieves its OS module from IPSO in the control processor.

What is SecureXL?

SecureXL is the Check Point security performance architecture of VPN-1/Firewall-1 and IP Appliance. The architecture offloads many security intensive operations to highly optimized IPSO code running on network processor hardware to perform operations, such as TCP state negotiation, packet forwarding, network address translation (NAT), VPN cryptography, anti-spoofing, routing, and accounting. Optimized IPSO code placed at the hardware interrupt level or in a network processor reduces the overhead involved in performing these security operations, thus greatly reduces the overhead.

SecureXL must be enabled for the ADP technology to function properly. Unlike Firewall Flows, SecureXL extends this acceleration to firewall traffic connection rate and to encrypted VPN traffic throughput as well.

How SecureXL Accelerates Network Traffic

SecureXL accelerates Firewall-1 and VPN-1 performance by remembering certain attributes of connections that have already been validated by the Firewall/VPN application.

Upon validation, Firewall-1 stores these attributes in its connection table and offloads the connections to the SecureXL API. Thereafter, validation of related packets and connections is delegated to IPSO across the SecureXL API. In turn, the SecureXL API on IPSO converts the connections into bidirectional flows and offloads the flows to the ADPs. SecureXL also stores the connections in its connection table. The connections in the SecureXL connection table and in the firewall connection table should be the same. The bidirectional flows (client to server, and server to client) in the ADPs are stored in the flows table. By using the notion of “flows” to uniquely identify traffic belonging to specific firewall connections, the ADPs are able to communicate with the firewall about connections that should be allowed. Caching this state in the ADP allows subsequent packets of those connections to bypass firewall processing. These packets can then be forwarded directly through the ADP to significantly improve firewall forwarding performance. The flows may not always be bidirectional: multicast flows will be unidirectional. Both of these approaches involve substantially less computing overhead than required by the firewall/VPN application itself.

SecureXL makes use of the infrastructure provided by, and operates on top of, firewall flows. SecureXL is not mutually exclusive to firewall flows, but actually needs firewall flows mode to be operational in order to be used. Also, IPSO’s slow path is not used with SecureXL.
First packet through the firewall follows the following path:

1. Input packet
2. Ingress pre-processing
3. Ingress firewall processing
4. Ingress post-processing
5. Forwarding
6. Egress pre-processing
7. Egress firewall processing
8. Egress post-processing
9. Output packet

Once the first packet is validated by the firewall, the information is offloaded to the SecureXL API.
Subsequent packets are accelerated by SecureXL based on the information offloaded by the firewall about the first packet and take the following path:

1. Input packet
2. SecureXL processing
3. Output packet

This allows faster processing of the packet from the ingress to egress.

For example, take a communication between a Web Client and a Web Server. The TCP connection establishment is initiated by the Web Client, which sends an HTTP request. The Web Server responds by sending the HTTP component (text or graphic). Each of the following HTTP request packets from the Web Client that requests an HTTP component from the Web Server has the same source address, destination address, destination port (80), and protocol (HTTP). Only the source port, assigned by the Web Client’s operating system, one per connection, differs in order to create unique socket addresses at the Client for each HTTP request/component (via separate TCP connections for each component). HTTP responses travelling in the other direction, from the Web Server, have the same source address, destination address, source port (80), and protocol (HTTP). Only the destination port differs (it has been assigned by the Client operating system to that connection).

SecureXL introduced in the above flow, will monitor and take a note of the first HTTP request packet from the Web Client and wait for the firewall to authorize the request. Once a connection involving a flow to port 80 is approved by the Firewall for the Web Client (resulting from the first HTTP request) a template is created and stored by SecureXL. All subsequent packets carrying those additional requests can share that same template “approval”. Establishing these subsequent connections does not involve a round trip to the Firewall for validation, and hence, these packets are processed much more quickly through the Firewall. This methodology for connection acceleration is called SecureXL templates.
Templates

Templates are another mechanism IPSO uses to help accelerate network traffic by looking at four attributes for a match.

The attributes are:

- **SrcAddr**: Source Address
- **Proto**: Protocol
- **DestAddr**: Destination Address
- **DestPort**: Destination Port - (the SrcPort is masked out).

Templates are stored in the SecureXL connections table, like any other cached connection state, but new connections matching these Templates are assigned a new connection state entry; the firewall process does not need to be consulted for this acceleration to take place.

When Templates are Created

The first packet that comes into the ingress port of the device usually goes up to the firewall for processing since no connections have been validated.

After the firewall processes and validates the packet and stores the connections in the connection table, it offloads the connections to the SecureXL API. At the same time, the firewall also creates a template through the SecureXL API.

When Templates are used to Accelerate Traffic

When the SecureXL API does not find a match in its connection lookup, it does a template lookup to decide whether the traffic can be accelerated. If SecureXL finds a match in a template lookup, it means a previous connection to the server by the same client has been validated by the firewall, and subsequent packets from the client can be allowed through the accelerated path.

Therefore, packets matching a template in IPSO will not be forwarded up to the firewall. Instead SecureXL API will add the connections in its connection table, convert the connections to bi-directional flows, and offload the flows to the ADPs for faster processing.

Why Templates are Disabled

Certain characteristics of a policy can (and will) disable templating of traffic. These items will adversely affect the acceleration of traffic matched against rules following the specific rule which disables the templates.

The following command may be run to view the rule which is disabling the templating feature:

```
IPSO[admin]# fwaccel stat
Accelerator Status : on
Templates : disabled by FireWall-1 starting from rule [rule number]
```

Characteristics of a rule will which disables templating of traffic matched against rules following the specific rule are as follows:

- Rules with the following objects:
  - Time object.
  - Port range object (SPORT range only, Services with DPORT range should not disable templates)
  - Dynamic object
  - Domain object
- Rules with "complex" services (e.g. services that have anything specified in the "Match" field, or "Enable reply from any port" of their "Advanced" section).
- Rules with RPC/DCOM/DCE-RPC services.
- Rules with client authentication or session authentication.
What ADP Accelerates

Connection acceleration is a SecureXL function. Connections that use ADP are accelerated by SecureXL and are processed by network processors instead of the main CPU.

ADP with SecureXL enabled, accelerates the following protocols and environments:

**Throughput Acceleration:**
- TCP traffic
- UDP traffic (unicast)
- IPSEC VPN traffic
- Higher layer protocol traffic transported over TCP or UDP
- Multicast forwarding
  - Only multicast forwarding is accelerated. Multicast protocols (PIM, etc.) are handled in IPSO in the motherboard Control Processor.

**Connection Rate Acceleration:**
- Unencrypted TCP traffic
- Unencrypted UDP traffic (unicast)
- Unencrypted higher layer protocol traffic transported over TCP or UDP (unicast)
Limitations

In this section:

Firewall Rule Limitations  
ADP Limitations  
Traffic Limitations  
Environment Limitations

Firewall Rule Limitations

The following rule properties present in the security policy will disable connection-rate acceleration for all traffic (throughput acceleration is not inhibited by the presence of rules with these properties):

- Service with a port number range, or as type "other", RPC, DCOM, or DCE-RPC, etc.
- Service with "enable reply from any port" checked
- Source or destination is a domain or a dynamic object
- Time object associated with the rule
- Client or session authentication involved with the rule
- SYN Defender (the entire 3-way handshake must be supervised by the Firewall-1 application, slightly reducing the effect of connection-rate acceleration – most significant performance impact on short duration connections)

The following rule properties present in the security policy will disable throughput and connection-rate acceleration for all traffic:

- Rules with action "encrypt" on an interface that does not support cryptography
- Rules where the source and destination of the rule is the gateway itself
- Rules where the service has an INSPECT handler (e.g. FTP control connection)
- Rules with Security Servers or services with resources
- Rules with user authentication
- Rules for non-TCP/UDP/GRE/ESP connections

ADP Limitations

Certain security policy rules and rule properties invoke extensive algorithms that are not replicated across the SecureXL API. SecureXL would not necessarily enable significant acceleration even if they were replicated because of their complexity relative to application overhead.

For optimum performance, the security policy should be designed, where possible, avoiding these rules and rule properties.

Traffic Limitations

The following traffic is not throughput or connection-rate accelerated by SecureXL:

- Multicast protocols
- Directed broadcast traffic
- Traffic across an Access Control List-enabled interface
- Traffic whose Protocol field in the IP header is not TCP or UDP (e.g. ICMP, IGRP, etc)
- IPv6 traffic
- VPN encryption algorithms that are not supported by the hardware
- IP compression enabled for VPN traffic

The following traffic is not connection-rate accelerated by SecureXL:

- VPN
- Complex connections such as FTP, H.323…
- Non-TCP/UDP connections

**Environment Limitations**

The presence of the following network traffic elements will disable SecureXL connection-rate acceleration:

- NAT
- FTP
- VPN Traffic

Network processors are programmable, so these challenges could all potentially be addressed through software and do not require new hardware in the future.

**Advanced SecureXL Configurations (IPSO 6.x)**

The IPSO software architecture allows additional tweaking of the SecureXL API.

**Fast Expiry**

Under certain circumstances the firewall experiences dropped packets when sending a relatively low rate of UDP (mainly DNS) traffic through the firewall. These drops are seen on even high-end platforms at very low data rates (< 50K pps). Similar problems are seen with TFTP connections. The packet drops can manifest as:

- drops in the software interrupt queues on ADP and non-ADP platforms
- in_qdrops in the internal Ethernet devices on ADP platforms
- in_qdrops in the external Ethernet devices on non-ADP platforms

On IPSO 6.x, this problem is caused by the build-up of UDP connections in the connection table, and the subsequent overhead in managing and deleting them.

Unlike TCP, UDP does not have an explicit teardown mechanism as part of the protocol, so the firewall removes them using an idle timer. The smallest value that can be configured for the timer is 10 seconds, so even short-lived UDP connections stay in the table for at least that long. So at even fairly modest connection rates, there can be hundreds of thousands of UDP connections in the table. The firewall periodically scans the connection table to determine whether the UDP connections need to be timed out. It sends a status request to IPSO for each UDP connection that may be idle. When the number of UDP connections is high, the overhead of processing them all monopolizes the system to the point where packets are dropped in the driver or in the software queues. The main source of contention appears to be for the NOKFW_MGMT_LOCK, which is required by both the code passing packets into the firewall, and the timer routine servicing the existing UDP connections.

This issue was first seen on IPSO 4.x. It was decided that IPSO would proactively delete UDP connections known to be complete, and notify the firewall of the deletion via the autoexpiry mechanism. This approach would reduce the number of UDP connections in the firewall's connection table, and therefore reduce the overhead of managing them. The fast-expire mechanism was added to SecureXL to implement this behavior.
How Fast Expiry Works

The fast-expire mechanism works as follows:

1. The user configures fast-expire entries, which describe the kind of traffic IPSO proactively expires. Each entry contains the IP proto and ports to match against new connections, and the data packet limit or idle time limit used to expire the connection. If the traffic type has data connections, the entry contains packet limit and idle time limit for the data connections.
2. When a connection is created, if it matches a fast-expire entry it is tagged as a fast-expire connection, and inherits the packet/time limits from the matching entry.
3. On an ADP system, fast-expire connections are not offloaded to the ADP, because the overhead of offloading the connection is greater than the benefits of doing so.
4. Each subsequent packet received for the connection decrements the packet count (if it exists), or restarts the idle timer (if running).
5. If the packet count reaches 0, or the idle timer expires, the connection is deleted via the autoexpiry mechanism used for TCP. If a connection has related connections, then the control connection will not be deleted until the data connections have been deleted.

It should be noted that this mechanism only reduces the overhead of UDP connections in the connection table; it does not eliminate it. After enabling this feature, you will still see packet drops at rates well below the published performance figures of TCP or UDP traffic for the platform.

Fast-expiry is enabled for each protocol using ipsctl commands. These commands should be added to /var/etc/rc.local, so that they are executed every time the system starts. The commands are quite different from those in IPSO 4.x. Instead of hard-coding the protocols for which fast-expiry can be enabled, these commands allow entries to be enabled for any UDP protocol.

Procedures

In this section:

- Determining the Status of SecureXL
- Viewing Additional SecureXL Statistics
- Enabling Fast-Expire
- Identifying & Monitoring Memory Leaks in ADP
- Identifying & Troubleshooting ADP Hung Conditions
- Installing a Replacement ADP (kana) Kernel
Determining the Status of SecureXL

The command... `fwaccel stat` ...can be used to determine the status of SecureXL.

You receive the following output when SecureXL is enabled:

```
IPSO[admin]# fwaccel stat
Accelerator Status : on
Templates : enabled
Accelerator Features : Accounting, NAT, Cryptography, Routing, HasClock, Templates, VirtualDefrag, GenerateIcmp, IdleDetection, Sequencing, TcpStateDetect, AutoExpire, VSX, DelayedNotif, McastRouting, BridgeRouting, WireMode
Cryptography Features : Tunnel, UDPEncapsulation, MD5, SHA1, NULL, 3DES, DES, AES-128, ESP, GRETunnel, DynamicVPN, NatTraversal, EncRouting
```

However, when SecureXL is disabled the output of the same command will be:

```
IPSO[admin]# fwaccel stat
Accelerator Status : off
```

```
Accelerator Features : Accounting, NAT, Cryptography, Routing, HasClock, Templates, VirtualDefrag, GenerateIcmp, IdleDetection, Sequencing, TcpStateDetect, AutoExpire, DelayedNotif, McastRouting, WireMode
Cryptography Features Mask : not available
```

Viewing Additional SecureXL Statistics

The following commands provide additional details about SecureXL statistics:

```plaintext
IPSO[admin]# ipsctl –i net:sxl:stats
```

- **Adds** – Total number of connections added
- **Add_updates** – adds that were converted to updates
- **Autoexpire_adds** – number of connection with autoexpiry
- **Add_fails** – limits, memory, duplicates
- **Delayed_total** – total delayed notifications connections
- **Deleted** – auto-expired and notified.
- **Handled** – delayed auto expired and notified.
- **Deletes** – Total deleted connections (includes autoexpired)
- **Drops_above_limit** – concurrent capacity reached.
- **Delete_fails** – connection entry not found/wrong instance.
- **F2F_conns** – current F2F connections.
- **Max_limit** – limit on Max concurrent connections.
- **Offloads** – number of connections offloaded to ADP.
- **Update_fails** – no entry/wrong instance.
- **Updates** – total number of updates on connections.

To show the number of flows created/deleted, run:

```plaintext
IPSO[admin]# ipsctl –i net:ip:flow:stats
```
Enabling Fast-Expire

The following command is used to create fast-expire entries:

```bash
ipsctl -c net:sxl:fastexpire:entry:<name>
```

The following variables can be set in the entry:

- **ipproto**: (1 – 255) Matches the IP protocol of the connection.
- **sport**: (1-65535) Matches the source port of the connection. A value of 0 matches any.
- **pktlim**: Number of packets seen on the connection before it is deleted.
- **timelim**: Interval, in milliseconds, after which the connection is deleted if no packets are seen.
- **desc_pktlim**: Packet limit for data connections.
- **desc_timelim**: Time limit for data connections.
- **flags:bidir**: (0 or 1) If set to 1, this entry will match connections in either direction (i.e. port values will match either source or destination ports).
- **flags:enabled**: (0 or 1) No connections will match this entry unless this variable is set to 1.

Below are some examples of how to implement this feature:

**DNS**
```bash
ipsctl -c net:sxl:fastexpire:entry:dns
ipsctl -w net:sxl:fastexpire:entry:dns:ipproto 17
ipsctl -w net:sxl:fastexpire:entry:dns: pktlim 1
ipsctl -w net:sxl:fastexpire:entry:dns:flags:enabled 1
```

**DHCP**
```bash
ipsctl -c net:sxl:fastexpire:entry:dhcp
ipsctl -w net:sxl:fastexpire:entry:dhcp:ipproto 17
ipsctl -w net:sxl:fastexpire:entry:dhcp:dport 68
ipsctl -w net:sxl:fastexpire:entry:dhcp: pktlim 0
ipsctl -w net:sxl:fastexpire:entry:dhcp:timelim 1000
ipsctl -w net:sxl:fastexpire:entry:dhcp:flags:enabled 1
```

**TFTP**
```bash
ipsctl -c net:sxl:fastexpire:entry:tftp
ipsctl -w net:sxl:fastexpire:entry:tftp:flags:bidir 1
ipsctl -w net:sxl:fastexpire:entry:tftp:ipproto 17
ipsctl -w net:sxl:fastexpire:entry:tftp:sport 69
ipsctl -w net:sxl:fastexpire:entry:tftp:dport 0
ipsctl -w net:sxl:fastexpire:entry:tftp: pktlim 0
ipsctl -w net:sxl:fastexpire:entry:tftp:timelim 1000
ipsctl -w net:sxl:fastexpire:entry:tftp:flags:enabled 1
```

**NOTE:** You should only enable fast-expiry for protocols causing packet drops on the system. Enabling the feature unnecessarily will cause performance degradation for other protocols.
Identifying & Monitoring Memory Leaks in ADP

Memory is statically allocated at bootup for connections, SAs, routes, mbufs. Connection memory leaks will lead to failures and eventual panic of ADP firmware.

This can be monitored using the command:

IPSO[admin]# ipsctl net:dev:adp:ipsctl:slot:[slot number]:nflow:stats

> Look for conn_free and existing

Mbuf memory leaks can be monitored using:

IPSO[admin]# ipsctl -a net:dev:adp:ipsctl:slot:[slot number]:kern:mbuf:stats

> Look for fmbufs and fmbufsfree

IPSO[admin]# ipsctl -a net:dev:adp:ipsctl:slot:[slot number]:kern:mbuf:cpu:[slot number]

> Look for fmbufs and fmbufsfree

> Mostly used for multicast related traffic when fanout is involved

SA memory leaks will lead to failures and eventual panic of ADP firmware. This can be monitored using:

IPSO[admin]# ipsctl -a net:dev:adp:ipsctl:slot:[slot number]:nsa:stats

> Look for alloc and free

Route memory leaks will lead to connection add failures and can be monitored using:

IPSO[admin]# ipsctl -a net:dev:adp:ipsctl:slot:[slot number]:nrt:stats

> Look for alloced and freed

Nexthops and other elements allocate memory dynamically can be monitored using:

IPSO[admin]# ipsctl -a net:dev:adp:ipsctl:slot:[slot number]:nnh:stats

> Look for malloc and free

Overall ADP memory can be monitored using:

IPSO[admin]# ipsctl -a net:dev:adp:ipsctl:slot:[slot number]:kern:kmem

> Look for free and total

Identifying & Troubleshooting ADP Hung Conditions

ADP hung conditions can be identified when:

- Packets are not getting forwarded
- Loss of connectivity
- VRRP/Cluster transitions
- Console is not responsive
Information Required for Troubleshooting these Conditions

Since the ADP is hung, the information from the IPSO level will be limited. One of the best pieces of information to gather is to get a forced core from the system in that hung state.

If the console is responsive, then force crash the system using the command:

IPSO[admin]# ipsctl -w -t string kern:force_crash nullptr (not while in ssh)

If the console is not responsive, you will need to break into the debugger:

- Send escape sequence... This is different for different console software. For example: for HyperTerminal it is CTRL+Break.
- Enter the key strokes... ddb ...This will not be displayed on screen. However, if this works you will be dropped into the db> prompt

From the db> prompt collect the output of the following:

- ps - Lists currently running processes
- t - Runs a backtrace
- panic - Will force a panic and reboot the system. Sometimes you will need to do this twice.

If ADP is hung, in most cases ADP should have dumped core and panicked IPSO automatically. In cases where you sense that ADP is hung, you can force crash ADP as well using:

IPSO[admin]# ipsctl -w net:dev:adp:ipsctl:slot:[slot #]:kern:debug:force_crash active

This will generate ADP core file and IPSO core file. It will take some time after the command is executed for IPSO to reboot as ADP’s dump needs to be transported to IPSO. The ADP crash file is located in its specific directory called kana_trace under the admin home directory.
Installing a Replacement ADP (kana) Kernel

You may be required to replace the ADP kernel (IP22xx). Since the ADP is an independent entity operating as an integral part of the system, replacing the ADP (or kana) kernel is similar to replacing the kernel on IPSO. However, the procedure is slightly different. This procedure is generally applied to IP22xx appliances.

The replacement kana kernel is obtained from Check Point support and the following should be taken into consideration:

- It is strongly recommended to revert back to the original kana kernel (firmware) prior to an IPSO upgrade.

To install a replacement kana kernel, follow this procedure:

1. Perform a complete system backup and move the file from the system
2. Move the replacement kana kernel to the IP22xx unit and verify md5
3. It is recommended to place the replacement kana kernel in a static location preferably in the admin home directory (/preserve/var/emhome/admin/)
4. Run the following command to update the kana kernel location:
   
   IPSO[admin]# dbset process:loadnp:arg:2 -f/preserve/var/emhome/admin/[new kana file]
   
   **Note:** Provided the replacement kana kernel is placed in the admin home directory
   
   IPSO[admin]# dbsave
   
   IPSO[admin]# sync;sync;reboot

Once the unit boots up you can confirm the change to the kana kernel by running:

IPSO[admin]# dbget process:loadnp:arg:2 -f/preserve/var/emhome/admin/[new kana file]

To revert back to the original kana kernel, follow the process below:

- IPSO[admin]# dbset process:loadnp:arg:2 -f/usr/firmware/kana.perf_g
- IPSO[admin]# dbsave
- IPSO[admin]# sync;sync;reboot