

## IDP Series Operating System Overview

The following topics explain the features of the IDP Series operating system:

- IDP Series Multicore Architecture on page 1
- Auto-Recovery Feature on page 2
- Flow Bypass Feature on page 3
- Key Processes on page 3

### IDP Series Multicore Architecture

The IDP Series operating system separates control plane and data plane processes, making the IDP Series appliances resilient to periods of heavy load, such as a denial-of-service attack (DoS attack) or distributed denial-of-service attack (DDoS attack). In high-end models, control plane and data plane processes run on separate CPU, bolstering resiliency and performance. Figure 1 shows the processes that run on each core CPU for IDP Series models.

**Figure 1: IDP Multicore Architecture**

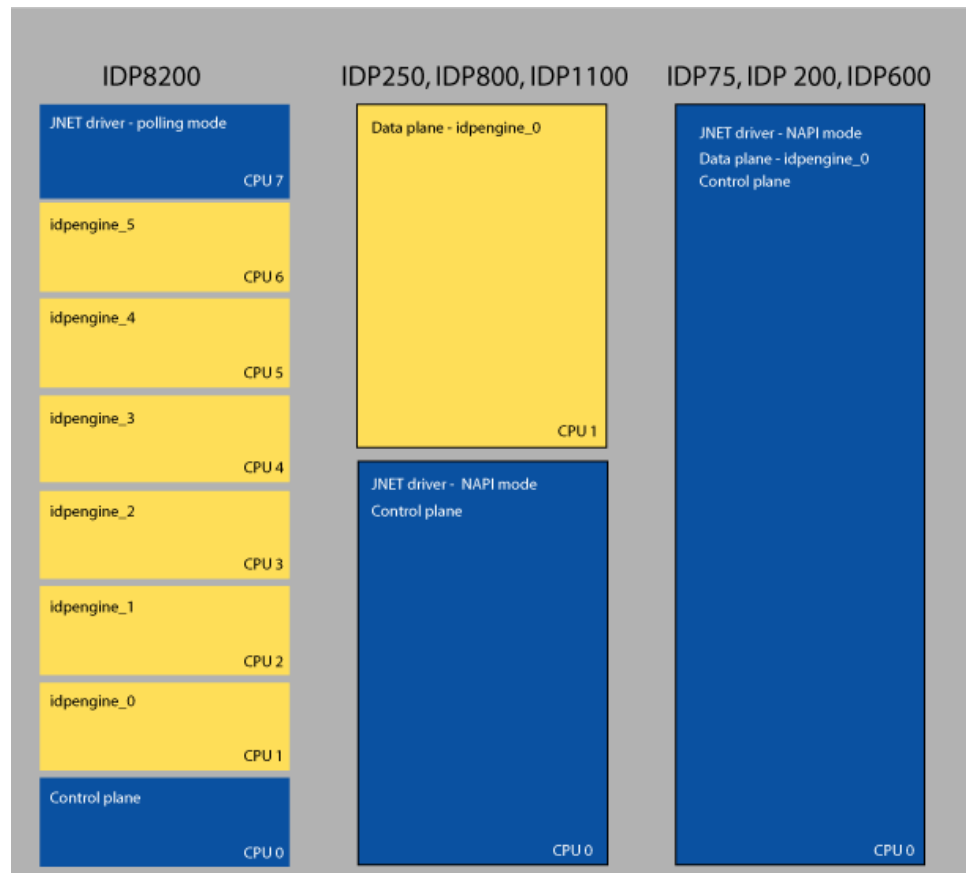


Table 1 describes how CPUs are dedicated on IDP Series models.

**Table 1: IDP Multicore Architecture**

Model	Multicore Architecture
IDP8200	<p>The IDP8200 appliance has eight core CPUs:</p> <ul style="list-style-type: none"> <li>■ One CPU is dedicated to control plane processes, including configuration and logging processes.</li> <li>■ One CPU is dedicated to the JNET driver processes, which handle traffic transmission and buffering. The JNET driver runs in polling mode.</li> <li>■ The remaining CPUs are dedicated to data plane processes, including the IDP engine processes that inspect traffic and take action against attacks.</li> </ul> <p><b>NOTE:</b> When you use the <code>scio</code> or <code>sctop</code> utilities to monitor IDP processes, you can specify the <code>-c CPU</code> option to display statistics related to processing on a particular IDP engine. For example, to display CPU utilization for CPU 2, the syntax is <code>scio -c 2 idp-cpu-utilization</code>. Without the <code>-c</code> option, <code>scio</code> displays an aggregate for all IDP engines.</p>
IDP1100, IDP800, IDP250	<p>The IDP1100, IDP800, and IDP250 appliances have two core CPUs:</p> <ul style="list-style-type: none"> <li>■ One CPU is used for both control plane and JNET driver processes. The JNET driver runs in NAPI mode: when traffic is low, the JNET driver operates in interrupt mode; when traffic is high, the JNET driver switches to polling mode.</li> <li>■ The second CPU is dedicated to data plane processes.</li> </ul>
IDP600, IDP200, IDP75	<p>The IDP600, IDP200, and IDP75 appliances have one core CPU. The JNET driver runs in NAPI mode: when traffic is low, the JNET driver operates in interrupt mode; when traffic is high, the JNET driver switches to polling mode.</p>

## Auto-Recovery Feature

The auto-recovery feature monitors the status of individual IDP engines. In the event an IDP engine is terminated, the auto-recovery daemon logs the event, attempts to restart the IDP engine, and logs recovery. The auto-recovery feature is enabled by default.

The auto-recovery feature bolsters resiliency of IDP Series appliances by enabling restart per IDP engine. In case of failure by an IDP engine, the other IDP engines in the data plane do not need to be restarted. If an IDP engine becomes overloaded or terminates, the JNET driver buffers packets for the active sessions while the IDP engine restarts. If the IDP engine does not restart after six attempts, IDP may enter internal bypass (if enabled).

The auto-recovery feature is complemented by the bypass under congestion feature. When the IDP engine recovers, it processes the buffered packets. If the buffer becomes large enough, it triggers bypass under congestion instead of resulting in subsequent failure.



**NOTE:** The auto-recovery process ensures the IDP engine restarts with the same device configuration, feature configuration, and security policy that were in place before the restart. However, because the application identification feature uses the first packets of a session as a key to determining the application, the auto-recovery process cannot reliably identify the application for buffered sessions. As a result, in processing buffered traffic, the application identification feature is unavailable and application rate limiting cannot be applied. In addition, the latest interval of application volume tracking data is discarded.

## Flow Bypass Feature

The flow bypass feature prevents the IDP appliance from becoming a point of failure when the network is congested. With flow bypass enabled, when the IDP system packet receive queue reaches a rising threshold that you specify, the IDP engine marks the flow as a bypass flow and passes it through, uninspected. The IDP appliance passes through subsequent flows until the IDP system receive queue falls below a reset threshold that you also specify. On IDP8200, which has multiple IDP engines, the flow bypass feature operates per IDP engine; that is, when the IDP packet receive queue for an individual IDP engine reaches its rising threshold, the individual IDP engine takes the flow bypass action and other IDP engines continue to inspect flows.

The flow bypass feature is disabled by default. It is intended for use in networks that prioritize availability over security.

For procedures for enabling the flow bypass feature, configuring its thresholds, and monitoring the feature, see the *IDP Administration Guide*.

## Key Processes

Table 2 describes the key processes.

**Table 2: Processes**

Process	Description
agent	Manages communication between the IDP appliance and Network and Security Manager.
idpengine	Performs packet processing, including decapsulation or decryption, defragmentation, reassembly, inspection, and rule actions.
idpHMD	Generates SNMP alerts when thresholds are crossed for tracked resources on the device. Responds to SNMP poll requests. Resources are CPU, memory, hard disk space, and session count.
idpLogReader	Gathers logs generated by idpengine and stores the information in a local log database. The agent process forwards the data to NSM, where you can view records.
pkid	Inspects SSL traffic, if SSL inspection is turned on.
profiler	Gathers information about hosts and applications in your network. Profiler stores the information in the Profiler database. The agent process forwards the data to NSM, where you can view records.
sciod	Handles policy push, information retrieval, Profiler status, and so on.

**Table 2: Processes** (continued)

Process	Description
shad	Used by the standalone high availability feature. The standalone high availability feature is not supported in IDP 5.0.

**Related Topics** The following related topic is included in the *IDP Concepts and Examples Guide*:

- IDP Series Network Interfaces Overview

The following related topics are included in the *IDP Administration Guide*:

- Viewing Auto-Recovery Logs
- Enabling the Flow Bypass Feature

---

Published: 2010-01-13