

The AppSense logo is positioned in the top right corner of the page. It features the word "AppSense" in a dark blue, sans-serif font, with a registered trademark symbol (®) to its upper right. The background of the top half of the page is a dynamic, abstract composition of flowing, translucent blue ribbons that create a sense of motion and depth.

AppSense®

# AppSense Performance Manager CPU Control

## Contents

<b>Glossary</b>	<b>2</b>
<b>Introduction</b>	<b>3</b>
<b>Application Groups and the Configuration Rulebase</b>	<b>4</b>
<b>Smart Scheduling</b>	<b>5</b>
<b>Shifting Table</b>	<b>5</b>
<b>Half-life Algorithm</b>	<b>6</b>
<b>Share Factors</b>	<b>6</b>
<b>Reservations</b>	<b>6</b>
<b>Soft Limits</b>	<b>7</b>
<b>CPU Affinity</b>	<b>7</b>
<b>CPU Application (Hard) Limits</b>	<b>7</b>
<b>Thread Throttling</b>	<b>8</b>
<b>Notes on Combining Features</b>	<b>9</b>
<b>Example 1 - Reservations and Soft Limits</b>	<b>9</b>
<b>Example 2 - Affinities and Reservations</b>	<b>10</b>
<b>Example 3 - Affinities and Reservations/Limits</b>	<b>10</b>
<b>Example 4- Combining Smart Scheduling with Thread Throttling or Hard Limits</b>	<b>10</b>

## Glossary

<b>TERM</b>	<b>MEANING</b>
<b>Base Priority</b>	Processing priority allocated to a process
<b>CPU</b>	Central Processing Unit
<b>GUI</b>	Graphical User Interface
<b>Kernel</b>	Central component of the OS, responsible for management of resources and communications between hardware and software components
<b>OS</b>	Operating system
<b>Shifting table</b>	Granular list of priorities to which processes can be assigned and move up or down the table

## Introduction

This document provides detail about AppSense Performance Manager features which are specific to the management of available CPU resources on a server or workstation. This functionality is divided into four main areas:

- > Smart Scheduling
- > CPU Affinity
- > CPU Application Limits
- > CPU Thread Throttling

## Application Groups and the Configuration Rulebase

Although application groups within AppSense Performance Manager are not part of the CPU control functionality, it is important to understand application groups and how their use influences the management of resources.

Resources are distributed by assigning the appropriate amount of CPU time to a particular application group within AppSense Performance Manager. This is managed using a rulebase, for which the default configuration for the product allocates the available CPU resource as follows:

User/Group	Process	Share Factor
NT Authority\System	<Any Application>	High (15)
BUILTIN\Administrators	<Any Application>	Normal (10)
<Other Users>	<Any Application>	Low (5)

When the performance of a given process is evaluated, it is compared with each rule in the rulebase, starting from the top. A process can only be matched to a single rule. The third and final rule acts as a 'catch-all', but is in fact not necessary, because AppSense Performance Manager contains an implicit rule not listed in the configuration, and which cannot be removed. This implicit rule ensures that all processes (regardless of user context) not matched to any rule in the active configuration will be assigned to a default application group, which is granted a share factor of 5. The third rule is present to ensure simple configurations and prevent confusion.

It is important to note that all processes assigned to an application group will then share the resources made available to the group. This sharing between processes within a group is equal and cannot be configured. Thus, when using the default configuration, CPU time allocated via a share factor of 5 will be divided equally between all processes which are not being run in the context of the local system account or a member of the local administrators group.

Note: This can be tested by creating 11 instances of autoflatline.exe (an in-house written utility that simply generates a 100% CPU load). Add ten instances to one application group, and one instance to another application group. Grant both groups a share factor of 1, move them to the top of the rulebase, save the configuration and start all 11 executables. The performance graph in Figure 1 demonstrates the difference in allocated CPU resource in this scenario.

Resources are, however, only consumed by the processes within an application group where required. As the operating system typically only requires a small amount of the available resources in order to function correctly, and local administrators rarely attempt to process resource-intensive tasks on user systems during core business hours, the third rule in the default configuration typically provides effective fair sharing of the available resource. However, where resource intensive applications are run in the context of system or a local administrator, it may be necessary to re-evaluate the resources allocated to users.

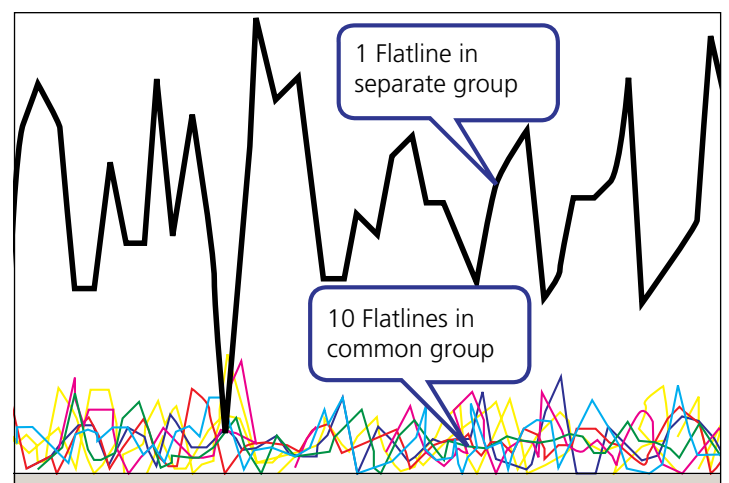


FIGURE 1

## Smart Scheduling

Smart Scheduling is the generic term for a technique which dynamically manipulates the base priorities allocated to running processes according to the active AppSense Performance Manager configuration. CPU resource is then accurately distributed to applications as required using three different features:

- > Share Factors
- > Soft Limits
- > Reservations

Smart Scheduling can be seen to be operational when the base priorities of running processes are changing dynamically. This manipulation of base priorities to achieve the desired performance profile is essentially the technique behind Smart Scheduling. The different features such as share factors and soft limits which can be used in combination allow the resources to be allocated exactly as required.

Smart Scheduling is achieved via implementation of two key components:

- > Shifting Table
- > Half-life algorithm

### Shifting Table

The Windows operating system implements a number of system priorities at which tasks can be executed, only some of which can be accessed by applications outside the OS itself. These accessible priorities are:

- > Shifting Table
- > Realtime
- > High
- > Above Normal
- > Normal
- > Below Normal
- > Low

Of these, applications are typically run at Normal priority by the Windows Scheduler. This grants equal priority to all processes running at the same priority. AppSense Performance Manager enhances this prioritization system to ensure resources are allocated effectively and as required. However, this produces the potential for application performance to be perceived as 'jerky' due to priority manipulation of a large number of competing applications across a small number of available priorities. To address this, the Smart Scheduler uses a 'shifting table' to derive 24 available base priorities within the freely available Windows base priorities to which applications can be assigned. The shifting table is illustrated in Figure 2.

```
Const DWORD dwshiftingTable [NUM_SHIFTING_BANDS +
1] [NUM_SHIFTING_CYCLES] =
{
0   TOP_P, TOP_P, TOP_P, TOP_P, TOP_P, TOP_P, TOP_P, TOP_P,
1   TOP_P, TOP_P, TOP_P, TOP_P, TOP_P, TOP_P, TOP_P, MID_P,
2   TOP_P, TOP_P, TOP_P, TOP_P, TOP_P, TOP_P, MID_P, MID_P,
3   TOP_P, TOP_P, TOP_P, TOP_P, TOP_P, TOP_P, MID_P, MID_P, MID_P,
4   TOP_P, TOP_P, TOP_P, TOP_P, MID_P, MID_P, MID_P, MID_P,
5   TOP_P, TOP_P, TOP_P, MID_P, MID_P, MID_P, MID_P, MID_P,
6   TOP_P, TOP_P, MID_P, MID_P, MID_P, MID_P, MID_P, MID_P,
7   TOP_P, MID_P, MID_P, MID_P, MID_P, MID_P, MID_P, MID_P,
8   MID_P, MID_P, MID_P, MID_P, MID_P, MID_P, MID_P, MID_P,
9   BEL_P, MID_P, MID_P, MID_P, MID_P, MID_P, MID_P, MID_P,
10  BEL_P, BEL_P, MID_P, MID_P, MID_P, MID_P, MID_P, MID_P,
11  BEL_P, BEL_P, BEL_P, MID_P, MID_P, MID_P, MID_P, MID_P,
12  BEL_P, BEL_P, BEL_P, BEL_P, MID_P, MID_P, MID_P, MID_P,
13  BEL_P, BEL_P, BEL_P, BEL_P, BEL_P, MID_P, MID_P, MID_P,
14  BEL_P, BEL_P, BEL_P, BEL_P, BEL_P, BEL_P, MID_P, MID_P,
15  BEL_P, BEL_P, BEL_P, BEL_P, BEL_P, BEL_P, BEL_P, MID_P,
16  BEL_P, BEL_P, BEL_P, BEL_P, BEL_P, BEL_P, BEL_P, BEL_P,
17  LOW_P, BEL_P, BEL_P, BEL_P, BEL_P, BEL_P, BEL_P, BEL_P,
18  LOW_P, LOW_P, BEL_P, BEL_P, BEL_P, BEL_P, BEL_P, BEL_P,
19  LOW_P, LOW_P, LOW_P, BEL_P, BEL_P, BEL_P, BEL_P, BEL_P,
20  LOW_P, LOW_P, LOW_P, LOW_P, LOW_P, BEL_P, BEL_P, BEL_P,
21  LOW_P, LOW_P, LOW_P, LOW_P, LOW_P, BEL_P, BEL_P, BEL_P,
22  LOW_P, LOW_P, LOW_P, LOW_P, LOW_P, LOW_P, BEL_P, BEL_P,
23  LOW_P, LOW_P, LOW_P, LOW_P, LOW_P, LOW_P, LOW_P, BEL_P,
24  LOW_P, LOW_P, LOW_P, LOW_P, LOW_P, LOW_P, LOW_P, LOW_P,
};
```

FIGURE 2

## Half-life Algorithm

When managing the performance of an application, it is important to adjust the allocated level of system resources sufficiently quickly to produce the desired response and maintain agreed levels of service, but it is equally vital that changes to a performance profile are implemented smoothly to prevent large and frequent changes in resources which would produce a quick-slow-quick-slow effect, adversely affecting performance. AppSense Performance Manager achieves this via implementation of a half-life algorithm, ensuring a smooth transition between current and required performance for each process. For example, if a process is currently consuming 40% of CPU time, and is allocated 20% of CPU time, then the Performance Manager agent is required to manipulate the Windows base priorities such that the process is consuming 20% when resource usage is next evaluated. This evaluation of CPU consumption takes place once per second. Base priorities can then be changed every 50 milliseconds, allowing any required changes in CPU consumption to be completed gradually over a one-second period using the 20 available increments. An example of this controlled change in resource use is illustrated in Figure 3.

### CHANGING CPU UTILIZATION VIA HALF-LIFE ALGORITHM

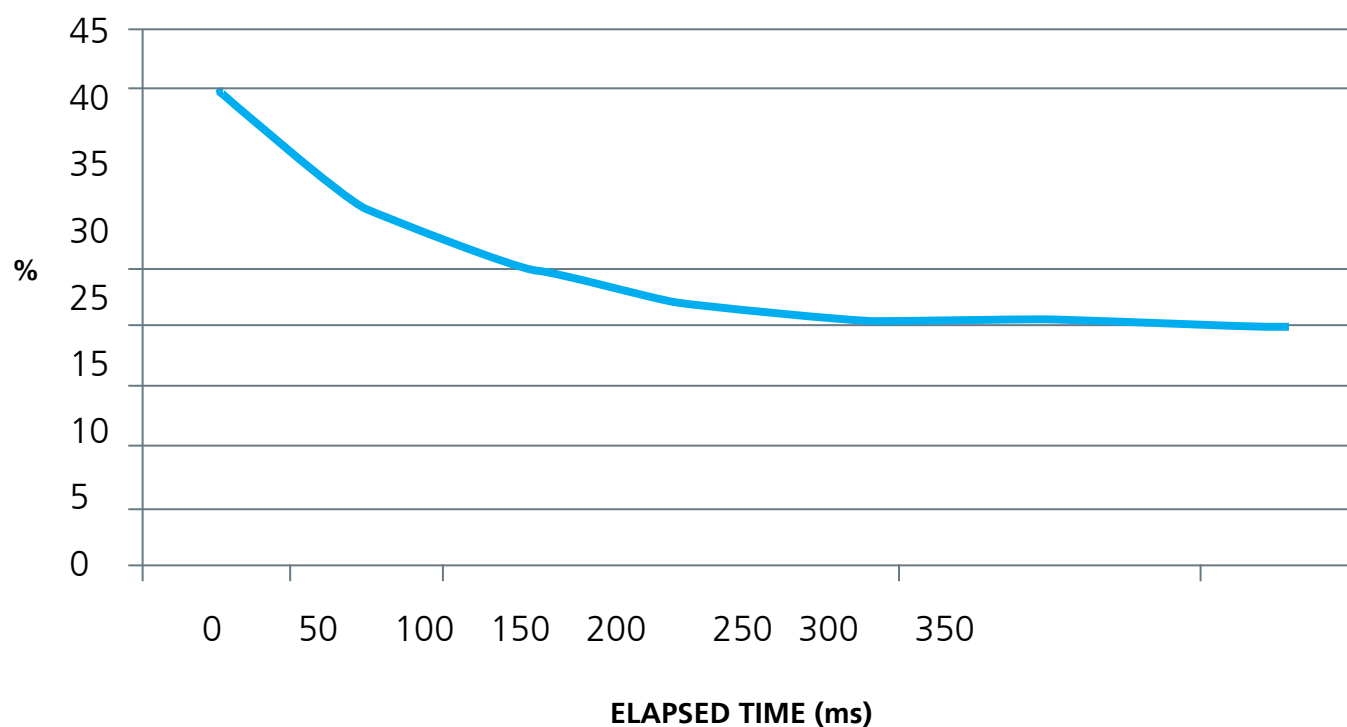


FIGURE 3

## Share Factors

The purpose of share factors is to provide a method of assigning relative levels of performance to processes running on a system. This is achieved by using the shifting table and half-life mechanism to adjust the process base priorities.

Share Factors are also used by the Soft Limit and Reservation features, so disabling Share Factors will disable these other components.

## Reservations

A Reservation means that while a process is consuming less than a specified amount of CPU time, that process will be assigned the highest possible base priority. This ensures that performance of critical tasks cannot be impacted regardless of other processes on the system.

When a process is consuming more than the Reservation level, it will be allocated base priorities based on the relevant share factor.

## Soft Limits

A Soft Limit is essentially the opposite of a Reservation, and is used to ensure that processes cannot consume more than a specified amount of available resources when there is contention for that resource. This is achieved by ensuring that when the CPU utilization of a process exceeds the designated limit, that process is assigned the lowest possible base priority. This means that where there is no competition for the available resource, the process will be permitted to consume the available CPU. This feature is useful for maintenance tasks that need to complete without impacting performance of critical processes or user experience. When a process is consuming less than the specified Soft Limit, it will be allocated base priorities based on the relevant share factor.

## CPU Affinity

CPU Affinities ensure that all threads for a particular process are bound to particular logical processors. Note that single core processors, individual cores in multi-core processors, and hyper threading modules are all perceived by the Windows kernel as logical processors.

Consequently, consideration should be given to the processing capacity of the specific logical processor(s) to which processes will be bound. This does not prevent other processes from also using the same logical processors.

## CPU Application (Hard) Limits

Although a Hard Limit sounds similar to a Soft Limit, it is not accomplished using Smart Scheduling technology. A Hard Limit means that all threads in a process are suspended, and subsequently resumed, for a specified period of time every second. For example, a Hard Limit of 20% will mean that the target process is prevented from accessing the processor for 800 milliseconds in every second. When the process is granted access to the processor, other active CPU control features will be enforced.

## Thread Throttling

Thread Throttling, a patented technology, predates the other CPU control functionality in the product and is used to prevent individual threads from monopolizing CPU resource. This works by suspending over-consuming threads for specified periods of time according to the active configuration. The default configuration is detailed in Table 1.

When CPU reaches (%)	For (sec)	Clamp by (%)	For (sec)	Do not clamp processes below (%)	Do not clamp threads below (%)
100	2	10	10	20	20

TABLE 1

This default configuration means that when the CPU is under 100% load for 2 seconds, Thread Throttling will be invoked to suspend, for 10 in every 100 milliseconds, all threads consuming more than 20% of the total CPU time. This will continue for 10 seconds, at which time performance will be re-evaluated. This process is detailed further in Figure 4.

Thread Throttling does not clamp process IDs 0 (Idle) or 4 (System), as these are critical to system operation. When threads are being clamped, but not actually suspended, they are still subject to the other active CPU control features.

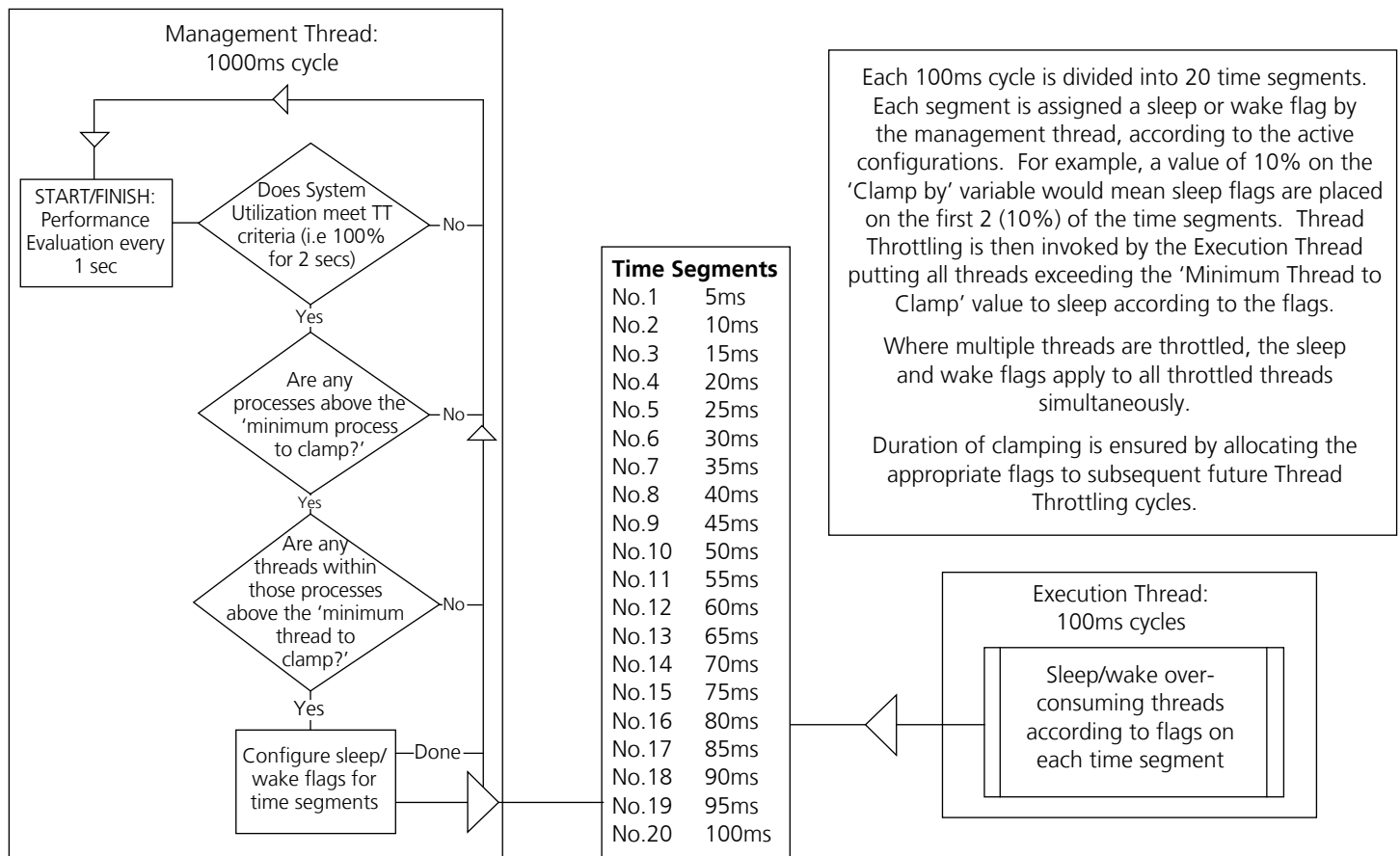


FIGURE 4

## Notes on Combining Features

The various CPU control features work well in combination, and allow a variety of scenarios to be catered for. A brief discussion of some examples follows.

### Example 1 - Reservations and Soft Limits

It may be appropriate to control a maintenance process which runs overnight when system utilization is otherwise low, but ensure that should the required task be incomplete when business hours commence, the workload is not permitted to adversely impact the user experience or performance of other server applications. Thus, the Smart Scheduling configuration in Table 2 may be appropriate.

Share Factor	Reservation (%)	Soft Limit (%)
1	20	30

TABLE 2

This would result in the target process being managed as follows:

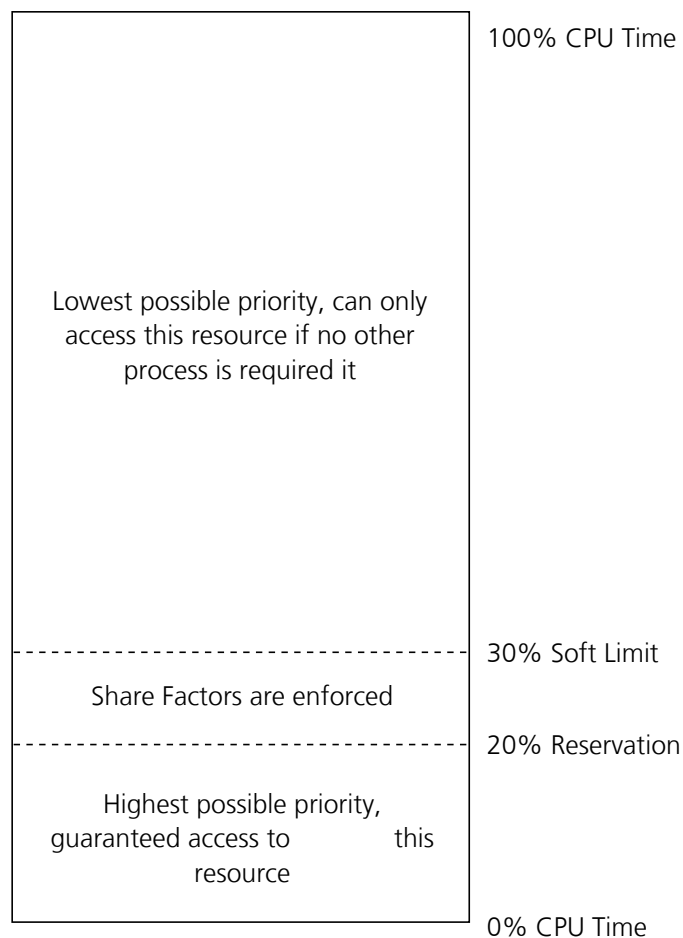


FIGURE 5

## Example 2 - Affinities and Reservations

It may be appropriate to control a maintenance process which runs overnight when system utilization is otherwise low, but ensure that should the required task be incomplete when business hours commence, the workload is not permitted to adversely impact the user experience or performance of other server applications. Thus, the Smart Scheduling configuration in Table 2 may be appropriate.

## Example 3 - Affinities and Reservations/Limits

Features applying control of a percentage of CPU resource (i.e. Reservations, Soft Limits, Hard Limits), apply to the CPUs available to the process at that time, so using Affinity to prevent access to particular logical CPUs will directly affect the implementation of a Reservation or guarantee. For example, when not using Affinity, limiting a process to 30% of 4 logical processors offers roughly double the processing power of the same limit when binding the process to just 2 cores.

## Example 4 - Combining Smart Scheduling with Thread Throttling or Hard Limits

Thread Throttling and Hard Limits are independent of other configuration aspects, so it is possible, for example, to apply a high share factor and a Hard Limit to a target process. This would have the effect of granting the process a high level of access to CPU time, even though for part of that time, the process would be prevented from accessing the CPU by the Hard Limit.

For more information on AppSense Performance Manager, please visit [www.appsense.com](http://www.appsense.com)

You can also evaluate AppSense software for 21 days by visiting [www.appsense.com/evaluate](http://www.appsense.com/evaluate)



[www.appsense.com](http://www.appsense.com)

## AppSense®

The information contained in this document ("the Material") is believed to be accurate at the time of printing, but no representation or warranty is given (express or implied) as to its accuracy, completeness or correctness. Neither AppSense nor the publisher accepts any liability whatsoever for any direct, indirect or consequential loss or damage arising in any way from any use of or reliance placed on this Material for any purpose.

© 2000-2008 APPSENSE INCORPORATED. ALL RIGHTS RESERVED

AppSense is a registered trademark of AppSense Inc. All other brands or product names are trademarks or registered trademarks of their respective companies.

