

# Nextra AppMinder User' s Guide

---

Version 5  
2<sup>nd</sup> Edition



# Contents

---

<b>Chapter 1 Introduction .....</b>	<b>2</b>
Using this book.....	2
Format Conventions.....	3
<b>Chapter 2 AppMinder Features and Capabilities .....</b>	<b>5</b>
Introduction to AppMinder .....	5
Multiple Broker hierarchies.....	9
Session security .....	10
Name service cleanup.....	10
<b>Chapter 3 Starting up AppMinder .....</b>	<b>13</b>
Before you start .....	13
Starting the AppMinder Monitor.....	14
Starting AppMinder Agent .....	17
Starting the Viewer.....	23
<b>Chapter 4 Using the AppMinder Viewer .....</b>	<b>25</b>
Connecting to a Monitor.....	25
Browsing for configurations.....	27
Updating configurations .....	31
Status Symbols.....	36
<b>Chapter 5 Using AppMinder' s command-line interface .....</b>	<b>39</b>
Overview .....	39
Starting and stopping services .....	40
Stopping AppMinder processes.....	42
<b>Chapter 6 Tutorial.....</b>	<b>43</b>
Useful functionalities .....	43

# Chapter 1 Introduction

---

This chapter covers how to use *AppMinder User's Guide*, who should use it, a quick rundown of topics, and a list of format conventions.

## Using this book

---

Welcome to the *AppMinder User's Guide*. This book combines discussions of important topics with step-by-step instructions to introduce you to some of the more complex features you can use when building or managing distributed applications. Take a minute to read these few pages—make sure you have the background that we assume you have and make sure you are reading the right book.

## Who should use it

---

This book is designed for administrators who use **AppMinder** to monitor three-tiered applications, and for developers who write programs using the **AppMinder** API.

## What you should already know

---

In general, you should have a basic understanding of client/server computing, and of the limitations inherent in two-tiered architectures. You should be familiar with the material in *Read Me First, Server Developer's Guide* and *Configuration & Security Guide* before delving into this manual.

## When to use it

---

The *AppMinder User's Guide* provides you with everything you should need to know about working with **AppMinder**. It is both a User's Guide and a Reference for **Appminder**.

## Topics

---

The following topics are covered in this book:

- **AppMinder** features and capabilities
- Starting **AppMinder** components
- Using **AppMinder**'s graphical user interface, the Viewer, to manage configurations

- Using **AppMinder's** command-line interface to manage configurations

## Format Conventions

---

### Text conventions

---

Understanding the conventions used in this manual will help you to learn how to use the utilities and to navigate the manual's structure.

Format	Explanation	Examples
terminal	Designates operating system or third-party utilities, file names, or constant values for variables.	kermit , telnet cust . def
<i>sub-text</i>	Designates text that represents many possible literal values; substitute your particular value here.	<i>server_c . pl</i> <i>-e environment_file</i>
<b>bold</b>	In body text, bold designates Entera utilities. In examples, bold highlights parts of the code.	<b>rpcmake</b> <b>AppMinder</b>
[brackets]	Designate optional text unless a vertical bar appears inside the brackets; in this case, one of the choices is required.	<i>[-d def_file]</i> [ NONE   ERROR   WARN   DEBUG ]

Paragraphs set off in the following manner are code examples:

```
#include <stdio.h>

main() {
    int i;
    printf("The number is %d", i);
}
```

### Symbols

---

The following symbols are used throughout the documentation to help you navigate the text.



### **Warning Message**

Indicates that you should pay special attention to the accompanying message. The message contains crucial information, without which you will not be able to continue properly.



### **Hint Message**

Indicates that the accompanying text, while it is not crucial information, does supply you with helpful instructions, depending on your situation.



### **Optional Message**

Indicates that the accompanying text is optional. The message may outline additional functionality or an alternate method, or detail a process step that may aid you in understanding a concept.



### **Debugging Tip**

Indicates that the accompanying text contains instructions on debugging the current step of your project. Debugging tips may be skipped if you use other successful debugging methods or if you choose not to debug (at your own risk).

## Chapter 2 AppMinder Features and Capabilities

---

This chapter introduces **AppMinder**'s features and capabilities. Please move on to "[Chapter 3 Starting up AppMinder](#)" after thoroughly understand this chapter.

Before you read this section, you should already be familiar with the information provided in *Server Developer's Guide*. Because **AppMinder**'s main function is to start and monitor distributed applications, you should be particularly comfortable with the structure of distributed applications, including multiple Broker hierarchies.

### Introduction to AppMinder

---

The **AppMinder** utility provides the following services for managing three-tiered distributed applications.

- It starts services:
  - servers
  - Brokers
- It checks services at regular intervals to ensure they are running;
- If a Broker goes down, **AppMinder** restarts the Broker, and reregisters all services that were registered with that Broker;
- If a server goes down, **AppMinder** restarts that server.

**AppMinder**'s main function is to make the start-up and maintenance of distributed applications more convenient. **AppMinder** does not have to be running in order for clients, servers and Brokers to communicate; it does, however, add considerable robustness to any distributed application.

### Overview of AppMinder components

---

To understand how to use **AppMinder**, you need to understand **AppMinder**'s components. **AppMinder** is a three-tiered distributed application consisting of the three shaded components shown in Figure 2.1.

In this model, you use the Viewer, a graphical user interface (GUI), to edit and monitor configurations. A *configuration* is a hierarchical listing of servers and Brokers that make up a Nextra application, or a collection of service groups. An *active configuration* is a configuration that is currently

being managed by **AppMinder**. A *standby configuration* is a configuration that is not currently being managed by **AppMinder**.

The Monitor receives requests from one or more Viewers, routes the requests to Agent, and sends server status information back to the Viewers.

Agent carry out requests received from the Monitor to start and stop servers. The components use remote procedure calls (RPCs) to communicate with each other.

The following sections describe these three components in more detail.

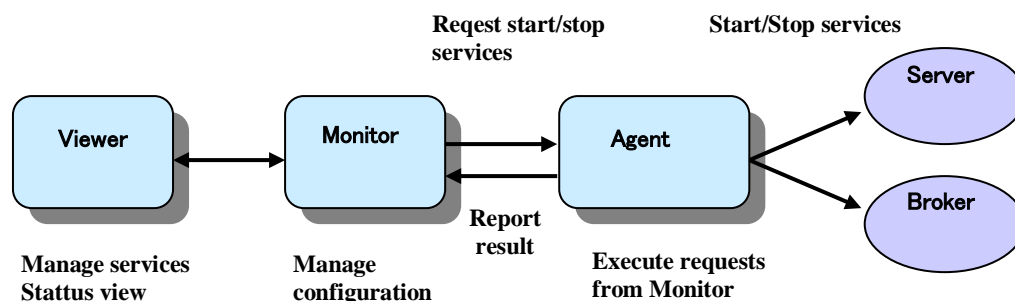


Figure 2.1: AppMinder components

## Viewer

---

**AppMinder's** Viewer makes managing your application easy, since you can immediately see what is running. The Viewer runs on Windows only and lets you edit standby configurations and monitor the status of the active configuration.

The Viewer uses Nextra RPC to communicate with Nextra Monitors.

## Monitor

---

The Monitor manages multiple configurations simultaneously, and lets you edit standby configurations.

The Monitor maintains an in-memory configuration database that contains records for all services and server instances of all its open configurations.

## Agent

---

The Monitor uses Agent to start, stop, and restart servers on any local or remote machine. An Agent is a process that runs continually in the background, waiting for a Monitor process to contact it and request an action.

### **Agent's local configuration database (appmagt.cdb)**



When you direct **AppMinder** to start or stop a service, on any host, the Monitor sends the request to the **AppMinder** Agent running on that host. The Agent then executes the request. Each Agent maintains a local configuration database that contains records for all services and server instances running on the Agent's host. Never attempt to manipulate the Agent's local configuration database directly.

Moreover, the Agent requires to have the configuration database when running off with `-recover` options.

## **Summary of features**

This section briefly describes **AppMinder's** features.

### **Internal security**

- **Session security.**

The Monitor and the agent always use encrypted RPCs to communicate with each other.

### **Configuration files**

You can use **AppMinder's** Viewer to create and debug the distributed application hierarchy interactively. Once completed, you can save the hierarchy configuration as a configuration file (you provide the name). You can then load this configuration file at any time to re-create the desired setup of servers and Brokers quickly and easily. **AppMinder** also lets you lock a configuration with a password so that other users can only read the configuration file.

### **Viewer shutdown**

Because **AppMinder** is a three-tiered client-server application, where the business logic is separated from the presentation, you can quit the

Viewer after you finish configuration operations, and let the Monitor and Agent manage configurations in the background.

## **Multiple Broker hierarchies**

---

In any Nextra configuration, there must always be one master Broker which keeps track of all the servers. The master Broker can also keep track of other Brokers, called “Sub-Brokers.” Each Sub-Broker may, in turn, keep track of its own servers and Brokers.

For more information about **AppMinder** monitoring multiple Brokers, see “[Multiple Broker hierarchies](#)”.

## **Error checking**

---

**AppMinder** provides several types of logging capabilities. When you start the Monitor, you can specify the name of a log file where the Monitor writes information about its operations and errors. Likewise, you can name a log file where the agent records its operations and errors. You can also name log files for the Monitor and Agent to record information on the RPC communication between the two components. Using **AppMinder**'s Viewer interface, you can view the log or error file for a particular service or server instance at any time.

## **Callable API**

---

To be available in the future versions.

## **Recovering orphan servers**

---

When an Agent stops running, it also stops all servers that it manages. However, under certain circumstances it is possible for *orphan servers* to remain running after the Agent stops. **AppMinder** lets you start an Agent with a recover option that tells the Agent, upon restarting, to look for orphan servers. If the agent finds orphan servers, it resumes managing them, rather than unnecessarily starting new instances of the server.

## **Fine-tuning AppMinder**

---

The Monitor and Agent have many options that allow you to fine-tune their operations. You can set these options in initialization files that the Monitor and Agent use at startup.

## **Scheduling services**

---

**AppMinder** includes a facility that lets you schedule start and stop times, on a daily, weekly, or monthly basis, for services and server instances.

## Multiple Broker hierarchies

---

Nextra **AppMinder** supports multiple Broker hierarchies. There must always be one master Broker, which keeps track of other services, a *service* being defined as either a server or a Broker. Any Broker that registers with another Broker is called a *Sub-Broker*. Each Sub-Broker may, in turn, keep track of its own services; this hierarchy resembles a tree-like structure of services with the master Broker as the root. (Keep in mind that this tree is upside-down, similar to a family tree.)

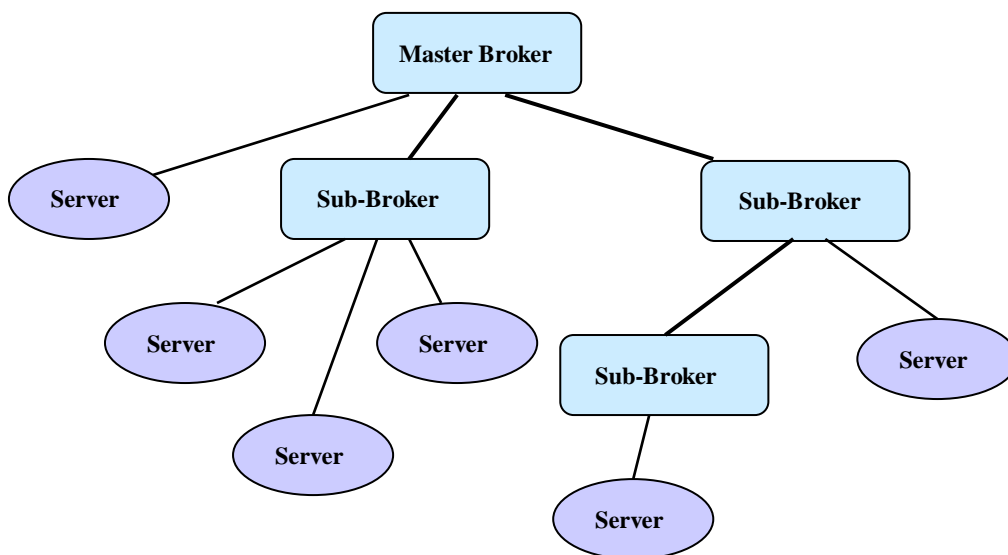


Figure 2.2: A Multiple Broker Hierarchy

### Advantages

---

This multiple Broker structure has two main advantages.

#### Minimize load

---

It minimizes the load on any single Broker by allowing more than one Broker to keep track of an application's services; this distributes the "brokering" duties among the branches of the tree.

Consider the following example. Company A has an application that consists of twenty-one servers. The application administrator can distribute the load of these servers by starting a master Broker with four Sub-Brokers. Each of the Sub-Brokers can then keep track of several servers—say 4 servers are tracked by Sub-Broker 1, 4 by Sub-Broker 2, and 6 and 7 by Sub-Brokers 3 and 4 respectively.

### **Robustness**

---

The second advantage is robustness. In a multiple Broker structure, if one of your many Sub-Brokers were to become unavailable for any reason, the Monitor still has access to other Sub-Brokers.

## **Session security**

---

The Monitor and Agent always use encrypted RPCs to communicate with each other. The Monitor and Agent both require that you supply a password to start them. The two passwords must match. The Monitor and Agent use the password to encrypt their RPC communication with each other.

By default, communication between the Monitor and the Viewer is not encrypted. However, you can start the Monitor in secure mode, by specifying `AMMON_SECURE=1` in the Monitor's initialization file, to enable encrypted communication between these two components.

## **Name service cleanup**

---

In addition to starting and stopping services and server instances, the **AppMinder** Agent constantly checks the state of its services and server instances, and it compares its list of services to the list maintained by the Broker. This section describes how the Agent manages services and server instances.

### **Agent startup**

---

When an Agent starts up, it initializes itself based on the command-line arguments and the initialization file settings that you supply. The Agent then tries to start the services and server instances listed in its

configuration database. If a configuration database does not exist, the Agent writes an empty configuration database to disk.

## Management cycle

---

After startup, the Agent loops through a management cycle. A cycle consists of the Agent checking each service in its list once, and performing the management operation appropriate for the service's state.

The following list describes how the Agent behaves in the management cycle.

1. **The Agent checks the state of the service at the top of its list.**
2. **If the service is in the PING state, the Agent attempts to ping it. If the ping fails, the Agent changes the service's state to FAILED, increments the start attempts counter, and sends an update state request to the Monitor.**
3. **If the service is in the STARTED state, the Agent checks the time elapsed since the service started.**

If the delay period has expired, the Agent changes the service's state to VERIFYING so that it knows to ping the service on its next loop through the management cycle, and sends an update state request to the Monitor.

4. **If the service is in the VERIFYING state, the agent queries the Broker for a list of all server instances that have that service's interface name on that host.**

The Agent retrieves the PID for each of those server instances, and compares the PIDs to the list of PIDs in the agent's list of services. If the Agent is unable to ping a server instance, and you have set `AMAGENT_CLEAN` to 1 in the Agent's initialization file, the Agent removes the server instance from the name service. The Agent sets the state of the service and all server instances whose PIDs are in the Agent's list to PINGED.

5. **If the service is in the NEW state, meaning that the service has just been added, the Agent initializes the service record and attempts to start the service.**

If the service starts, the Agent sets the service's state to STARTED. If an operating system error prevents the service from starting, the

Agent sets the service's state to `START_FAILED`, increments the start attempts counter, and sends an update state request to the Monitor.

- 6. If the service is in the `FAILED` state, meaning that the Agent failed to ping the service during its previous loop through the management cycle, the Agent checks the start attempts counter.**

If the number of start attempts equals the maximum number of attempts allowed, the Agent sets the service's state to `MAXED_OUT`, and sends an update state request to the Monitor. Otherwise, the Agent attempts to start the service, and sets the service's state to `STARTED`.

- 7. After checking the state of a service, the Agent checks for RPC requests from the Monitor.**

If there are any requests, the Agent processes one request, then returns to the management cycle and checks the state of the next service.

## Chapter 3 Starting up AppMinder

---

This chapter explains how to start up the **AppMinder** Monitor, Agent, and Viewer. This chapter also lists the **AppMinder** files that you need on your system, and the environment variables that you must set, to run **AppMinder**.

### Before you start

---

This section lists the **AppMinder** files that must reside on your system, and the environment variables that you must set to run **AppMinder**.

The file requirements vary by platform.

#### Windows

---

To run **AppMinder** on Windows NT, the following files must be in a directory that is part of your PATH:

File name	Description
<i>ammon.exe</i>	Monitor executable
<i>appmagt.exe</i>	Agent executable
<i>amlodsvr.exe</i>	load server command
<i>amuldsvr.exe</i>	unload server command
libamclient.dll	<b>AppMinder</b> client library
libamclnutl.dll	<b>AppMinder</b> client utility library
libamcmn.dll	<b>AppMinder</b> common library
amviewer.exe	Viewer executable
agtlaunch.exe	Launcher command
pmond.exe	Process monitor
librpc.dll	Nextra RPC library
appmmsg.cat	Message catalog

#### All UNIX platforms

---

To run **AppMinder** on any of the four supported UNIX platforms, the following files must be in a directory that is part of your PATH:

File name	Description
ammon	Monitor executable
appmagt	Agent executable
<i>amlodsvr</i>	load server command
<i>amuldsvr</i>	unload server command
appmmsg.cat	Message catalog

## Starting the AppMinder Monitor

This section shows how to start a Monitor. Before you start a Monitor, make sure that the name service (Broker) with which the Monitor is registered, is running. You must start a Monitor before you can issue **AppMinder** utility commands. To issue certain commands, you must also start an Agent. The command to start an **AppMinder** Monitor is:

```
ammon -c initialization_file [-i] [-p password ] [-v]
[-h] [-b broker_key][-m]
```

**Table 3.1: AppMinder Monitor command line options**

Option	Description
-c <i>initialization_file</i> <b>Required option</b>	text file that contains a series of attribute and value pairs, which control the behavior of the Monitor.
-i	generates a sample initialization file with the name <code>ammon.ini</code> . You can use this sample initialization file, which contains default values for all attributes, as a template.
-p <i>password</i>	password, which you must specify the first time you start the Monitor. For a Monitor and Agent to communicate with each other, they must have the same password. If you do not specify a password, the Monitor uses the password specified in the initialization file with the <code>AMMON_PASSWD</code> attributes. If the initialization file does not contain a password, you must specify a password on the command line. When you specify a password on the command line, the Monitor writes an encrypted version of that password to the initialization file. If the initialization file contains a password, the Monitor replaces it with an encrypted version of the

	password that you specify on the command line.
-v	shows the version information.
-h	shows command line option list.
-b	Not supported at the current version.
-m	runs in thread mode.  DCE_DROP_AGT_RPC environment variable can be used to improve the Monitor performance when running in thread mode. DCE_DROP_AGT_RPC environment variable is set 0 as default that means Monitor does not process a RPC request from the Agent if Monitor is in processing any other RPCs.

Table 3.2 lists the attributes for which you assign values in the initialization file. Although there is no restriction on naming initialization files, we recommend that you use a file extension of `.ini`. Having a common file extension makes it easier to perform troubleshooting. Do not include any blank lines in the initialization file. Assign values by using the following format:

For example: `AMMON_MODE=1`

**Table 3.2: Monitor initialization file settings**

Attributes	Value/Description	Default value
AMMON_ACTIVE	contains a list of configurations that the Monitor has loaded and is currently managing. You cannot edit this field.	no default
AMMON_AGTPORT	number of the port where the agent runs.	7001
AMMON_AGTTIMEOUT	number of seconds that the Monitor waits to receive communication from an Agent before the Monitor marks the Agent as failed and marks all services and server instances on that host as unreachable.	300
AMMON_ID	identifier that the Monitor	no default

	generates the first time it reads the initialization file. You cannot edit this field.	
AMMON_LOG	name of a log file where the Monitor writes status and error information.	monitor.log
AMMON_LOGLEVEL	controls the level of logging information that the Monitor writes to the log file that you specify with AMMON_LOG.  NORMAL=only errors and critical operations DEBUG=all operations	NORMAL
AMMON_LOGSIZE	controls the maximum size of the log file that you specify with AMMON_LOG.	1MB
AMMON_MODE	controls the Monitor server mode.  0=TCP	0
AMMON_PASSWD	encrypted Monitor password. You cannot edit this value in the initialization file. You can set this value by specifying a password with the <code>-p</code> command line option.	no default
AMMON_SERVERARGS	name of an environment file that the Monitor uses at startup.	-e ammon.env
AMMON_TRPCLOG	names the log file where the Monitor writes information about Monitor to Agent communications. Legacy attributesOnly DCE mode Monitors use this log file.	monitrtcp.log
AMMON_TRPCLOGLEVEL	controls the level of logging information. This field supports the same values that you can set with the	ERROR , DEBUG

	DCE_DEBUGLEVEL environment file attribute. See <i>Reference</i> for a complete description of these values.  NONE ERROR WARNING DEBUG	
--	---	--

## Starting AppMinder Agent

This section shows how to start an Agent.

The command to start an **AppMinder** Agent process is:

```
appmagt -c initialization_file [-pname <pipe name>] [-i] [-p password] [-recover] [-v] [-h] [-repair]
```

**Table 3.3: AppMinder Agent command line options**

Option	Description
-c <i>initialization_file</i> <b>Required option</b>	text file that contains a series of attribute and value pairs, which the agent uses to configure itself.
-p <i>pipe_name</i> <b>Windows only</b>	required if you want to run multiple appmagt on a single Windows platform. < <i>pipe_name</i> > can contain Alphabets, Numbers and _ (Underscore). No other characters are permitted.
-i	generates a sample initialization file with the name appmagt.ini. You can use this sample initialization file, which contains default values for all attributes, as a template.
-p <i>password</i>	password, which you must specify the first time you start the Agent. For a Monitor and Agent to communicate with each other, they must have the same password. If you do not specify a password, the Agent uses the password specified in the initialization file with the AMAGENT_PASSWORD attribute. If the initialization file does not contain a password, you must specify a password on the command line. When you specify a password on the

	command line, the Agent writes that password to the initialization file. If the initialization file contains a password, the Agent replaces it with the password that you specify on the command line.
-recover	This option directs the Agent to look for servers in its local configuration database that might have remained running when the Agent previously stopped running. Normally, when an Agent stops running, it also stops the servers that it manages. However, sometimes servers remain running even after the Agent stops running. When you specify -recover, if the Agent finds any running servers, it manages them; otherwise, it starts new servers. This option only works for UNIX version.
-repair	creates .cdb file from scratch. If already exists, then delete and create. If specify with -recover, then -recover overrides.
-v	shows the version information.
-h	shows command line option list.

Table 3.4 lists the attributes for which you assign values in the initialization file. Although there is no restriction on naming initialization files, we recommend that you use a file extension of `.ini`. Having a common file extension makes it easier to perform troubleshooting. Do not include any blank lines in the initialization file. Assign values by using the following format:

For example:

```
AMAGENT_PORT=7500
```

**Table 3.4: Agent initialization file settings**

Attributes	Value/Description	Default Value
AMAGENT_BROKER_SYNC_PERIOD	controls the frequency in second that Agent issues RPC against servers to reregister the location information to	3,600

	<p>the Broker.</p> <p>If smaller value specified than AMAGENT_MGTINTE RVAL, then AMAGENT_MGTINTE RVAL rules over this attribute.</p>	
AMAGENT_BURST_TIME_FOR_SEND	<p>The value, <i>duration</i>, for this attribute specifies the number of seconds that the Agent will send the services status information to Ammon after reached to the value with AMAGENT_MIN_UPD_FOR_SEND. In other words, this attribute turns effective when the value of AMAGENT_MIN_UPD_FOR_SEND attribute reached.</p> <p>Required to set the value smaller than AMMON_AGTTIMEOU T in the Monitor initialization file.</p>	60
AMAGENT_CLEAN	<p>controls whether Agent removes dead server entries from the naming service. A server is considered dead if the Agent is unable to ping it. Setting this option to 1 improves application performance.</p> <p>0=do not clean naming service. 1=clean naming service.</p>	1

<p>AMAGENT_DCE_CLN_TIMEOUT</p>	<p>The value, <i>duration</i>, for this attribute specifies the number of seconds that the Agent will wait for the return of an RPC sent to Broker/s Server. If the Agent is idle for <i>duration</i> seconds, then the Agent will close the connection to the Broker/Server, preventing it from hanging. The maximum value that <i>duration</i> can have is <i>LONG_MAX</i>, which is a platform dependent variable specified in <code>limits.h</code> as required by ANSI C.</p>	<p>5</p>
<p>AMAGENT_DCE_SVR_TIMEOUT</p>	<p>The value, <i>duration</i>, for this attribute specifies the number of seconds that the Agent will remain idle waiting for an RPC after the Ammon has connected to it. If no RPC arrives for <i>duration</i> seconds, then the Agent will close the connection to the Ammon, preventing it from hanging. The maximum value that <i>duration</i> can have is <i>LONG_MAX</i>, which is platform dependent variable specified in <code>limits.h</code> as required by ANSI C.</p>	<p>30</p>
<p>AMAGENT_CONFIGDB</p>	<p>name of local file that contains the Agent configuration database. If you do not specify an existing file, the Agent</p>	<p>appmagt.cdb</p>

	creates an empty configuration.	
AMAGENT_DEBUG	controls the level of logging information that the Agent writes to the log file that you assign to AMAGENT_LOG.  0=errors and critical operations only 1=all operations	0
AMAGENT_LOG	names the file to which the agent writes logging information. Set the level of logging information with the AMAGENT_DEBUG attribute.	appmagt.log
AMAGENT_LOGSIZE	controls the maximum size of the Agent log file.	1MB
AMAGENT_MIN_UPD_FOR_SEND	The minimum number of times the service status is to be updated by the Agent within its management cycle before the status is sent to ammon.	10000
AMAGENT_MIN_UPD_TIME_FOR_SEND	The value, <i>duration</i> , for this attribute specifies the number of seconds that the Agent will send the services status information to Ammon.  Required to set the value smaller than AMMON_AGTTIMEOUT in the Monitor initialization file.	120
AMAGENT_PASSWORD	an encrypted Agent password. You cannot edit this value in the initialization file. You	no default

	can set this value by specifying a password with the <code>-p</code> command line option.	
AMAGENT_PMOND_STOP_TIME	The value is the time in 4 digits in hhmm format to dictate Agent to restart the pmond sub-process at the time in the value. This attribute is only valid for Windows.  ex. 1315	0115
AMAGENT_PING_PERIOD	ping cycle to Broker and servers in seconds. Please NOT set lower than the value of AMAGENT_MGMTINTERVAL, otherwise it pings at the cycle set with AMAGENT_MGMTINTERVAL.	120
AMAGENT_MGMTINTERVAL	controls the frequency with which the Agent performs management operations. Value is in seconds.	120
AMAGENT_PORT	number of the port where the Agent runs. This must be set to the same number as AMMON_AGTPORT is set to in the Monitor initialization file.	7001
AMAGENT_SHUTDOWN *Obsolete	controls whether the Agent shuts down all servers it manages when it shuts itself down. If you set this to 0, be sure to specify <code>-recover</code> the next time	1

	<p>you start the agent to avoid orphan servers.</p> <p>0=do not shut down servers 1=shut down servers</p>	
AMAGENT_TRPCLOG	names the communications log file for TCP RPC.	agenttcp.log
AMAGENT_TRPCLOGLEVEL	<p>controls the level of logging information for TCP/RPC. This field supports the same values that you can set with the DCE_DEBUGLEVEL environment file attribute. See <i>Reference</i> for a complete description of these values.</p> <p>NONE ERROR WARNING DEBUG</p>	NONE , NONE

## Starting the Viewer

Before you start the Viewer, **AppMinder's** graphical user interface, be sure that the files listed in "Before you start" reside on your system, and that you have set the necessary environment variables. Also be sure to start any Monitors to which you wish to connect and an Agent if you want to load configurations, or start and stop services and server instances. Then enter the following command:

```
> amviewer [-h monitor_host] [-p monitor_port]
```

The main Viewer window and a transport security login dialog box appear on your screen.

However, given `-h` and `-p` options to the Viewer over start-up and accessed to an active configuration on the Monitor, then the Viewer does

not show any dialog box or interactive windows, but just show the all services on the Active status window.

## Chapter 4 Using the AppMinder Viewer

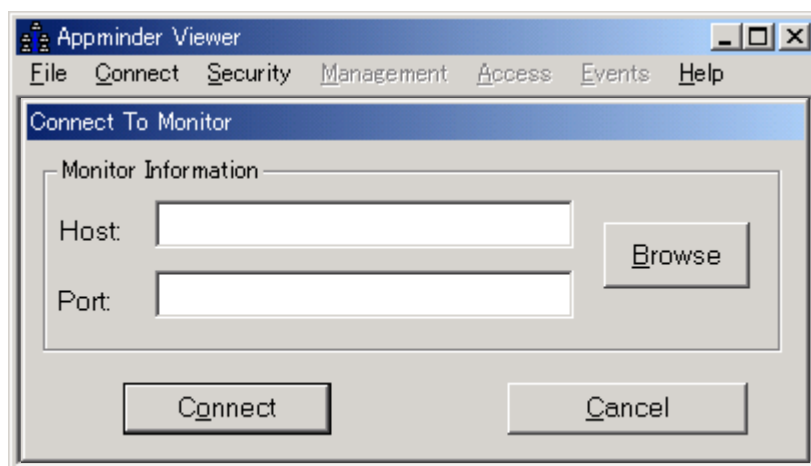
---

This chapter explains how to use the **AppMinder** GUI, known as the Viewer, to manage configurations.

### Connecting to a Monitor

---

When you start the Viewer, the Viewer's main window and the Transport Security dialog box appear on your screen.



**Figure 4.1: Connect To Monitor Dialog Box**

If you do not know the Monitor's host and port, click on the Browse button. A File Selection dialog box, similar to the one in Figure 4.2 appears.

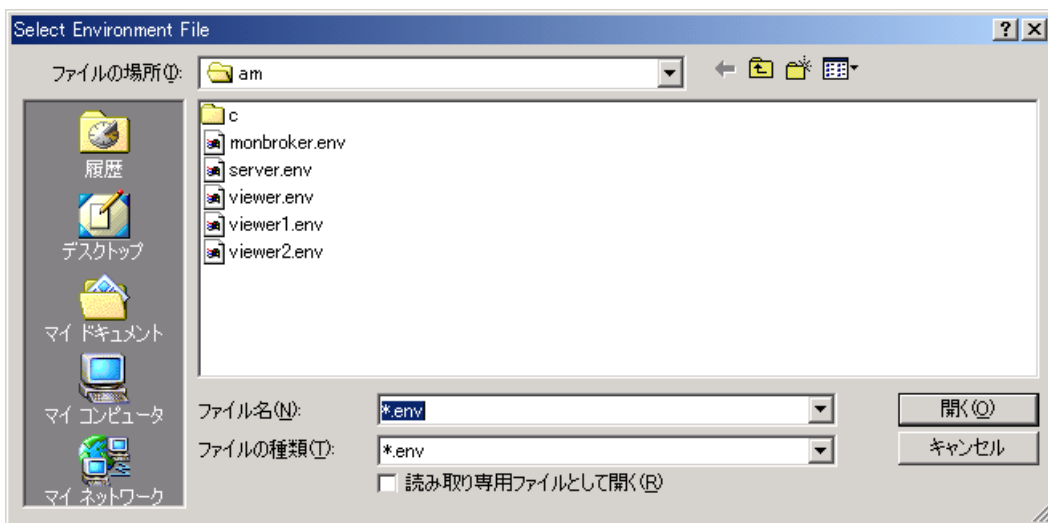


Figure 4.2: File Selection Window

The File Selection Window lets you use a filter to search the file system for Viewer's environment file. After you select the file and click the OK button, the Monitor Selection Window, as shown in Figure 4.3 appears on your screen.

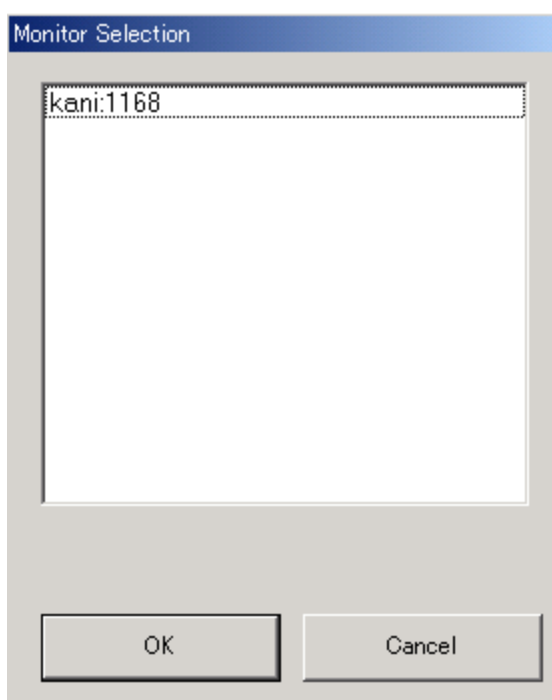


Figure 4.3: Monitor Selection Window

Select the host and port number of the Monitor to which you want to connect, and click the OK button. The host and port number appear in the

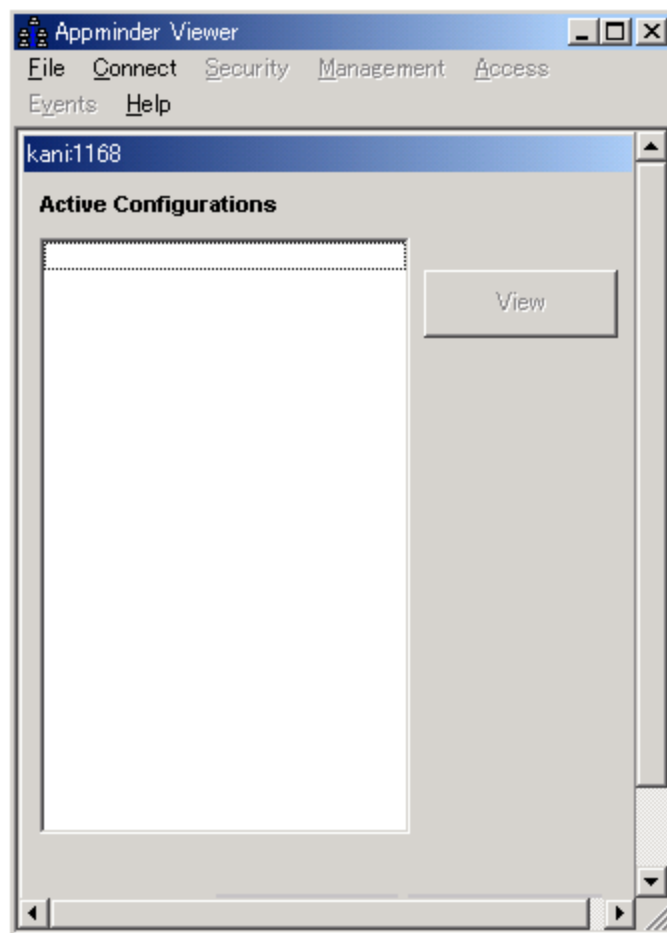
Monitor Connect Dialog Box. Click the Connect button to connect to the Monitor that you have selected.

## Browsing for configurations

---

When you connect to a Monitor, the main Viewer window automatically lists all active configurations for that Monitor, as Figure 4.4 shows.

No Monitor listed when no Monitor is active



**Figure 4.4: Main Viewer Window with Active Configurations**

The following tables describe the options available from the main Viewer window pull down menus.

### File menu

---

**Table 4.1: File menu options**

Option	Description
New	creates a new empty configuration file.
Open...	displays the Remote File selection window, which allows you to browse the file system for configuration files.
Save	writes the configuration file to disk, saving any changes that you made to an active or standby configuration. Note: when you add or remove a service or server instance from an active configuration, the change takes place immediately; however, <b>AppMinder</b> does not write the change to disk unless you save the configuration file before you close it. If you edit an active configuration file, the changes do not take effect immediately. You must save the changes, unload the configuration, and then load the configuration before the changes take effect.
Save As	saves the configuration file under a different name.
Close	closes a configuration file. Note: closing an active configuration file does not change the state of the configuration.
Exit	shuts down the Viewer.

## Connect menu

---

**Table 4.2: Connect menu options**

Option	Description
Open Connection To...	displays a dialog box where you specify the host and port number of the Monitor with which you wish to connect.
Disconnect	ends the Viewer session with the Monitor. Note: disconnecting from the Monitor does not affect active configurations. Configurations remain active as long as the Monitor and agents that manage them are running.

## Security menu

---

**Table 4.3: Security menu options**

Option	Description
--------	-------------

Set Monitor Password	displays dialog box where you specify a password for session security.
----------------------	--

## Management menu

---

**Table 4.4: Management menu options**

Option	Description
Begin Managing	Load the configuration listed in the Service List and make it active.
Stop Managing	Stop active configuration.

## Event menu

---

**Table 4.5: Event processing menu options**

Option	Description
Binding List	displays a list of event-handler servers currently registered with the Monitor for the configuration

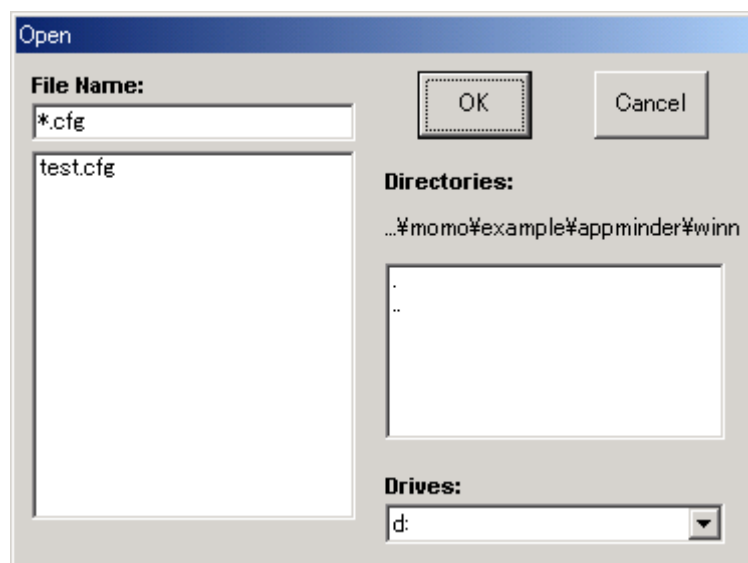
## Help menu

---

**Table 4.6: Help menu options**

Option	Description
About AM Viewer	Shows the version.

To see a list of all configurations, active and standby, select Open from the File pull down menu. The Remote File selection window appears, as shown in Figure 4.5. You can search for configuration files in a particular directory, and, if your file system supports disk drive letters, you can search for configuration files by disk drive letter.



**Figure 4.5: Remote File selection window**

When you find the configuration that you want to view, select it and click the OK button. The Remote File selection window goes away, and the name of the configuration file appears in the main Viewer window.

To see a hierarchical tree view of the configuration file, double-click on the configuration file name in the main Viewer window. For Nextra configurations, such as the one shown in Figure 4.6, the hierarchy tree shows Brokers, Sub-Brokers, and server instances.

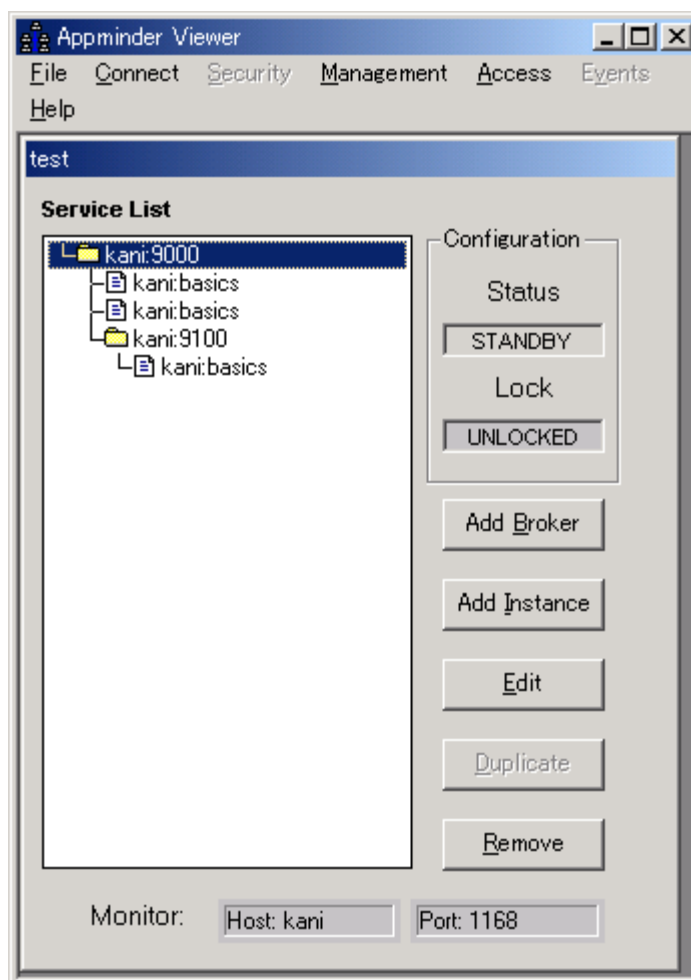


Figure 4.6: Hierarchy of services for a configuration

## Updating configurations

---

This section describes the different ways that you can update configurations through the Viewer. Specifically, this section describes:

- using buttons on main Viewer window
- using buttons on Active status window
- editing a configuration

### Main Viewer window operations

---

As Figure 4.6 shows, when you double-click on the service list name in the main Viewer window, the Viewer displays all of the configuration's services and server instances in a hierarchical tree. To perform operations on specific services or server instances, highlight the service or server

instance in the tree, and click on the appropriate button to the right in the main Viewer window.

When you add or remove a service or server instance from an active configuration, the Monitor does not save that change to disk. To write the new configuration file to disk, choose Save or Save As from the File pull down menu before you close the configuration.

### **Add Broker**

---

To add Broker to a configuration, click on “Add Broker” button. If the configuration is active, the Viewer lists the new Broker instance in the Active status window, and the Agent attempts to start the Broker instance.

### **Add Instance**

---

To add server instance to a configuration, click on “Add Instance” button. A dialog box pops up where you can specify the new server instance to be registered with that Broker or service group. If the configuration is active, the Viewer lists the new server instance in the Active status window, and the Agent attempts to start the server instance.

### **Remove**

---

To remove a server or server instance, highlight it in the tree, and click on the Remove button. When you remove a service, the Agent stops and removes the service and all server instances registered to that service. For Nextra configurations, the Agent also stops and removes all Sub-Brokers of the Broker. If the configuration is active, you can see the state of the service or server instance change to UNLOADED in the Active status window.

### **Load and Unload**

---

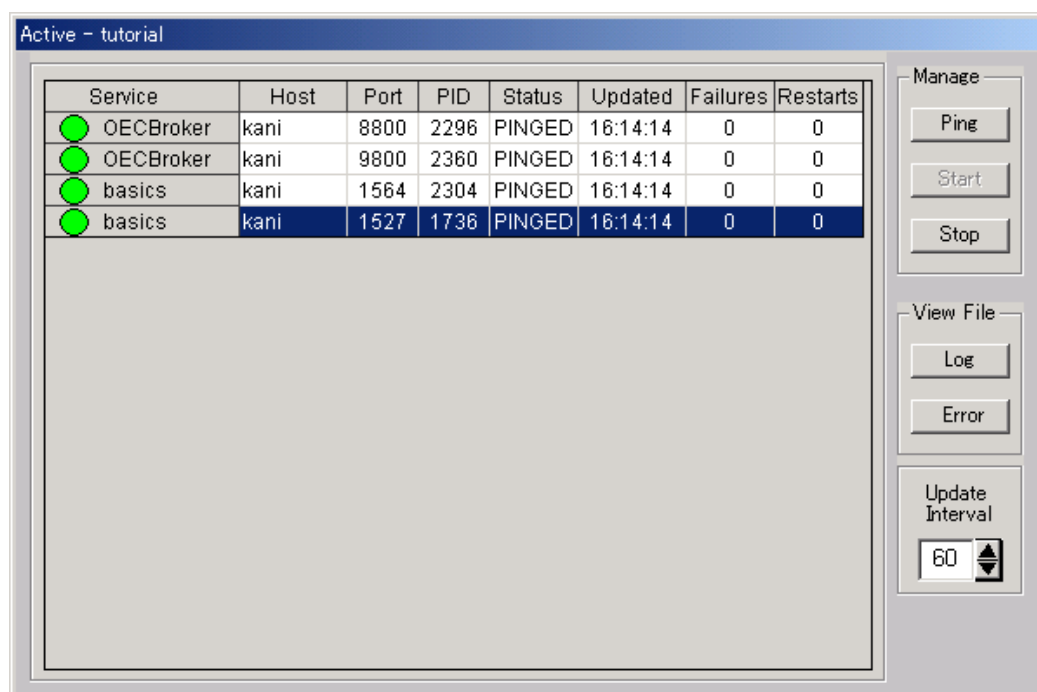
Click on the Load button to start all of the services and server instances in the configuration. Loading the configuration changes its state from standby to active. You must first load a configuration before you can stop and restart specific services and server instances in that configuration.

After you load a configuration, you can close the configuration, disconnect from the Monitor, and exit from the Viewer, and the Monitor, Agent, and configuration remain running.

Click on the Unload button to stop all services and server instances in the configuration. Unloading a configuration changes its state from active to standby.

## Active status window operations

When you first connect to a Monitor, the main Viewer window lists all active configurations for that Monitor. The only button available at that point is the View button. Highlight the configuration that you wish to check, and click on View. The Viewer opens an Active status window adjacent to the main Viewer window. The Active status window displays the status of every service and server instance in the configuration. Figure 4.7 shows an Active status window.



**Figure 4.7: Active status window**

The Monitor sends the latest status information to the Viewer, and the Viewer displays the new information in the Active status window. By default, the Monitor sends status updates to the Viewer every 60 seconds. Although you can change this setting with the Update Interval button, we recommend that you do not lower the number because doing so increases network traffic and can degrade performance.

To ping, start, or stop a service or server instance, highlight the service or server instance in the grid, then click the appropriate button at the right side of the window.

When you start or stop a Broker, the agent starts or stops:

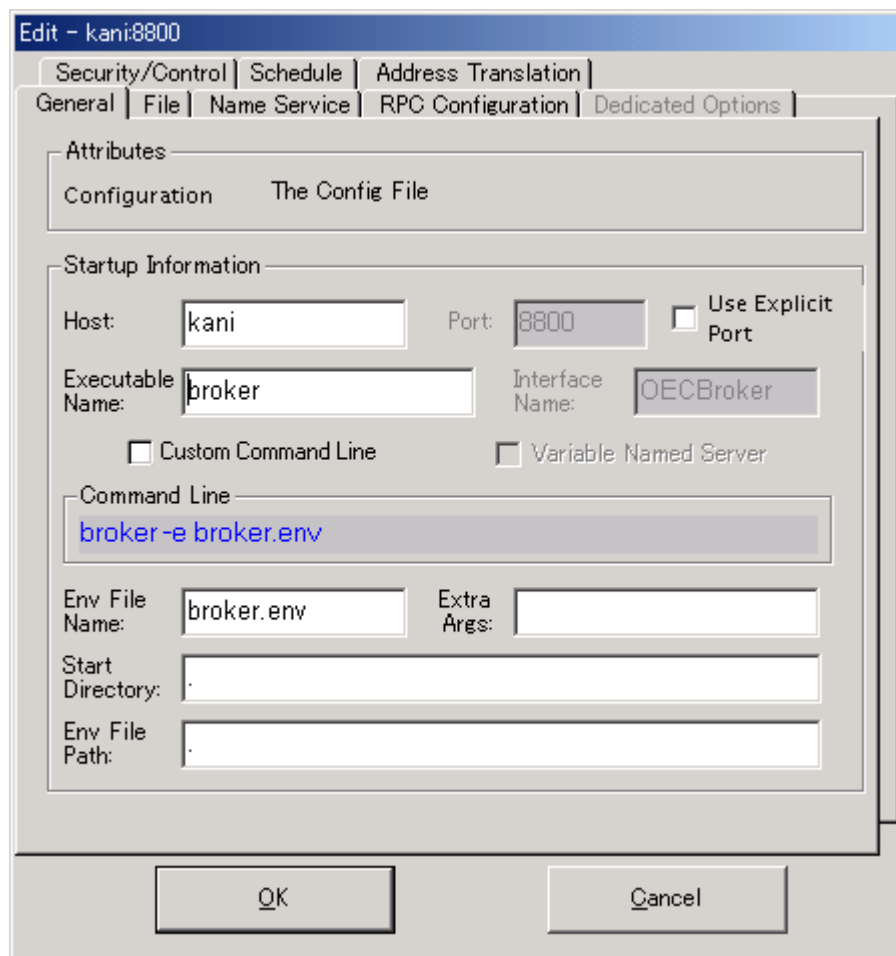
- the Broker
- all server instances registered with the Broker
- all of the Broker's Sub-Brokers
- all of the server instances registered with the Sub-Brokers

## **Editing a configuration**

---

To edit a service or a server instance in a configuration, highlight it in the hierarchy tree, and click on the Edit button in the main Viewer window. This causes the General Edit Details window, as shown in Figure 4.8, to appear adjacent to the main Viewer window. Use the General Edit Details window to specify information needed to start the service or server instance. The example in Figure 4.8 includes the command for starting a Broker.

If you click on the Detail page field at the top of the window and hold the mouse button down, a drop-down menu that lists the names of six other Detail pages appears. Table 4.8 describes all of the Edit Detail pages.



**Figure 4.8: General Edit Details window**

**Table 4.8: Edit Detail pages**

Detail page	Purpose
General	specifies information needed to start service or server instance.
File	specifies name and location of service's or server instance's environment file; error file; and log file and level of information to write to log.
Name Service	specifies the name service that the service or server instance uses. For Nextra, you can specify a Broker list and an extended Broker list.
RPC	sets attributes, such as packet size and timeout values, that control the RPC communication between components.

Dedicated Server	sets dedicated server options, such as maximum number of clients.
Security/Control	sets Nextra transport security, and sets control information such as ping delay time and maximum number of restarts.
Schedule	lets you schedule start and stop times for services and server instances.
Address Translation	Please refer to the Environment file attribute "DCE_TRANSCRIPT" in <i>Reference</i> manual.




Many of the attributes that you can set in the Edit Detail pages correspond to environment file attributes. See *Reference* for details on the effects of setting these attributes.






If you edit an active configuration, those changes do not take effect immediately. You must unload the configuration, and then load it, before the changes take effect.

## Status Symbols

Table 4.9 describes all Status Symbols appear on the Viewer.

**Table 4.9: Status symbols descriptions**

Symbols	Events	Value in Hex	Descriptions
	AM_NEW	0x000001	A new entry to the DB, but not started yet.
	AM_STARTING	0x000002	Processing to start the process.
	AM_STARTED	0x000003	The process has started normally.
	AM_RESTART	0x000010	The process has been set to restart.
	AM_VERIFYING	0x000400	Processing to acquire the process information.
	AM_BUSY	0x001000	RPC failed to respond.

	AM_DEDMAXED	0x002000	The dedicated server has been maxed out.
	AM_REREGISTER	0x040000	Some failure in the server process. It needs to reregister to the Broker.
	AM_PINGING	0x000100	AppMinder is pinging the process.
	AM_PINGED	0x000200	Succeeded to ping the process.
	AM_VERIFIED	0x000800	Verified that the process has been started off by the Agent.
	AM_STOP	0x080000	The process has been stopped. The status would be transferred to hibernation mode.
	AM_IDLE	0x100000	The process is in hibernation.
	AM_STARTFAIL	0x000008	Failed to start the process.
	AM_MAXED	0x000020	Reached to the max start.
	AM_LOST	0x008000	Unable to reach the host machine where the process was running.
	AM_FAILED	0x003000	Failed to ping the process.
	AM_OVERLOADED	0x010000	Overload on the server process.
	AM_UNREACHABLE	0x400000	The Monitor is unable to reach the Agent.
	AM_UNMANAGED	0x000040	The process is NOT the scope of AppMinder.
	AM_ADOPTED	0x000080	The process has been adopted by AppMinder.

	AM_BACKUP	0x020000	Backup process.
	AM_REMOVE	0x200000	The process has been unloaded or removed out of the configuration file (.cfg).

## Chapter 5 Using AppMinder's command-line interface

---

This chapter explains how to use the **AppMinder** command-line interface to create and manage your application configurations.

### Overview

---

This section provides a brief overview of the types of **AppMinder** commands available and the order in which to use them. This section also describes the arguments that are common to all **AppMinder** commands.

### Common arguments and environment variables

---

This section describes arguments that are common to all of the **AppMinder** commands, and it lists environment variables that you can use to reduce the number of arguments that you specify on the command line.

Table 5.1 lists the common arguments. The remaining sections of this chapter that show how to use the commands described only non common arguments.

**Table 5.1: Common command-line arguments**

Argument	Description
-h <i>monitor_host</i>	identifies the name of the host machine or IP address on which the Monitor runs.
-p <i>user_password</i>	identifies the password for the user. Specify this argument only if you want to run secure TCP RPC communication.
-m <i>monitor_password</i>	identifies the password for the Monitor. You must specify this argument if you start the Monitor in secure mode (AMMON_SECURE=1 in Monitor initialization file).
-e <i>environment_file</i>	identifies the name of the environment file.
-c <i>drive_path_and_file</i>	identifies the disk drive letter, if necessary, path, and name of the configuration file. Although there is no restriction on naming configuration files, we recommend using a file

	extension of <code>.cfg</code> . Using a common file extension makes it easier to perform troubleshooting.
<code>-?</code>	displays a help message for the command.
<code>-v</code>	shows the version information.

To reduce the number of arguments that you specify on the command line, you can set the following environment variables for the arguments listed in Table 5.1.

- `APPMONHOST=monitor_host`
- `APPMUSRPWD=user_password`
- `APPENVFILE=environment_file`
- `APPMCFGFILE=configuration_file`

## Starting and stopping services

---

This section describes how to use commands to perform the following operations:

- start all services and server instances by loading a configuration
- stop all services and server instances by unloading a configuration
- start one service or server instance
- stop one service or server instance

Because many of the arguments are common to all commands, the following sections describe only the non common arguments for each command. See [“Common arguments and environment variables”](#) for a description of the arguments used by all commands.

### Loading a configuration

---

To start all services and server instances in a configuration, or to start a particular service or server instance, issue the `amlodsvr` command as follows:

```
amlodsvr -h monitor_host -p user_password -m
monitor_password -e environment_file -c drive_path_and_file [-i
all | instance=uuid| service=uuid]
```

Parameter	Description
<code>-i</code>	Specify <code>all</code> to start all Brokers and server

	<p>instances in a configuration.</p> <p><code>-i service=<i>uuid</i></code>          Required to specify a Broker's <i>uuid</i>. It will load the Broker and services underneath.</p> <p>* Find <i>uuid</i> in the configuration file.</p>
--	--

You must start the Agent or Agents that start the configuration's services and server instances before issuing this command.

If you specify `all`, for a TCP configuration file, `amloadsvr` starts all Brokers and server instances.

You cannot start a specific instance of a service or server without first loading the entire configuration. After you load the configuration, you can stop specific instances by issuing the `amuldsvr` command, and then restart those specific instances by issuing `amuldsvr` with the `-i service=` option.

Loading a configuration changes the configuration state from standby to active. Once the configuration is active, you can stop and start specific services and server instances, and ping server instances and Agents.

Return value	Description
0	No error.
else	Return RPC error number described in "Nextra Error Message" in <i>Trouble Shooting Guide</i> .

## Unloading a configuration

To stop all services and server instances in a configuration, or to stop a particular service or server instance, issue the `amuldsvr` command as follows:

```
amuldsvr -h monitor_host -p user_password -m
monitor_password -u user_name -e environment_file -c
drive_path_and_file [-l all | instance=uuid service=uuid]
```

Parameter	Description
<code>-i</code>	Specify <code>all</code> to stop all Brokers and server in a

	configuration.  <code>-i service=<i>uuid</i></code> Required to specify a Broker's <i>uuid</i> . It will unload the Broker and services underneath.  * Find <i>uuid</i> in the configuration file.
--	---

If you specify `all`, for configuration file, `amuldsvr` stops all Brokers and server instance.

Unloading a configuration changes the configuration state from active to standby.

Return value	Description
0	No error.
else	Return RPC error number described in "Nextra Error Message" in <i>Trouble Shooting Guide</i> .

## Stopping AppMinder processes

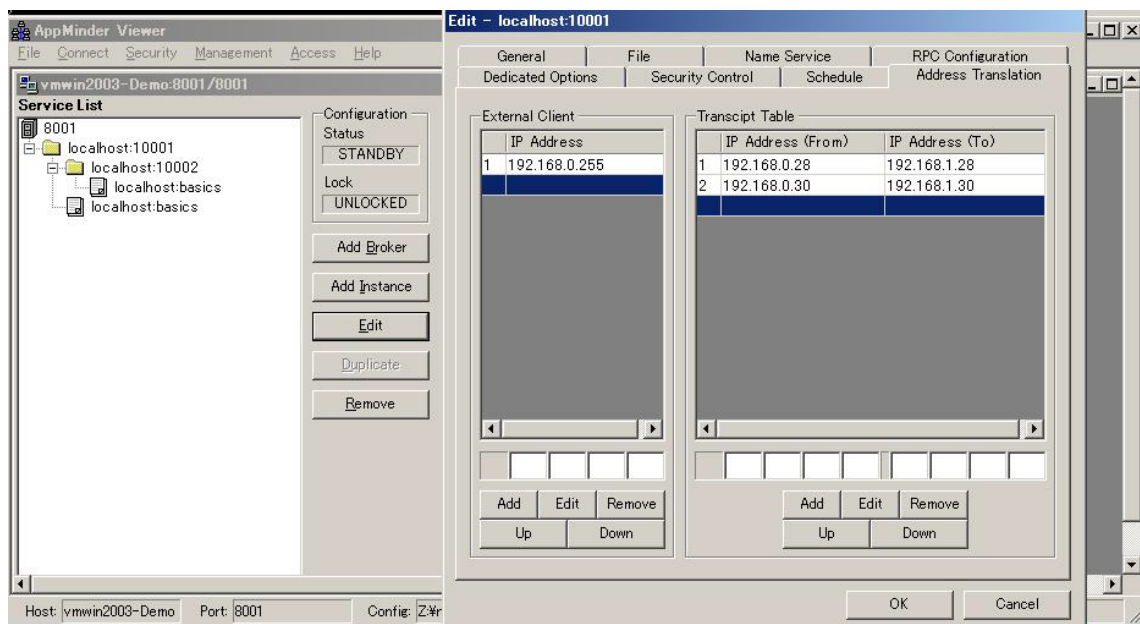
When you stop **AppMinder** processes, you may use a `kill -9` command. However, **AppMinder** cannot trap a `kill -9` signal, and the contents of the configuration file will be random. Always use a soft kill command to stop AppMinder processes.

# Chapter 6 Tutorial

## Useful functionalities

### Address Translation

This feature is suitable to use when the PC clients are located outside of the firewalls set along with NAT (Network Address Translation). Or the server IP address is visible to the PC clients, but the IP address is different from the IP address used by Nextra services.



**Figure 6.13: Address Translation Tab**

Figure 6.13 shows the Tab for Broker running at localhost : 10001. When requested by a client addressed 192.168.0.255 which is between 192.168.0.1 and 192.168.0.254 and the client requested for a Nextra server addressed 192.168.0.28, then the IP address 192.168.0.28 would be translated to 192.168.1.28.

Specified like that, then the requested client will acquire the service location information which contains the translated IP address as well as the port number and invoke the Nextra server.

### Specify server process's TCP/IP Port number

If you would like to run and monitor a Nextra server running at a specific port through AppMinder, please use this feature: click “Edit” button, tick “Use Explicit Port” and specify the port number you prefer.

## Nextra AppMinder User's Guide

---

2011/3/9	Agent's local configuration database (appmagt.cdb)
2008/10/10	v5 2 <sup>nd</sup> Edition
2007/6/28	amviewer [-h <i>monitor_host</i> ] [-p <i>monitor_port</i> ] and Useful functionalities added.
2007/4/17	3 <sup>rd</sup> Edition
2007/3/1	Agent performance improvement and extra attributes were added.
2006/11/13	Extra option -pname <pipe name> to the Agent.
2004/9/1	AMAGENT_MGMTINTERVAL changed from 40 to 120.
2003/5/25	2 <sup>nd</sup> Edition
2003/4/18	1 <sup>st</sup> Edition

Authors: Inspire International Inc.

---

Copyright © 1998-2011 Inspire International Inc.  
Printed in Japan