



# **AppGate Personal Firewall Policy Manager 1.2**

---

# **AppGate Personal Firewall Policy Manager 1.2**

Copyright © 2004-2008 AppGate Network Security AB

---

# Table of Contents

|  |    |
|--|----|
| 1. Overview .....  | 1  |
| 1.1. Typical usage .....                                       | 1  |
| 1.2. Policies .....  | 1  |
| 1.3. Protocol .....  | 2  |
| 2. Installation .....  | 3  |
| 2.1. Windows Installation .....                                | 3  |
| 2.1.1. AppGate Personal Firewall Packaging Tool .....          | 3  |
| 2.1.2. Unattended Installation .....                           | 3  |
| 2.2. Installation on other platforms .....                     | 3  |
| 2.3. Securing the installation .....                           | 4  |
| 2.3.1. File Access Rights .....                                | 4  |
| 3. Administration .....  | 5  |
| 3.1. Working with the Administration tool .....                | 5  |
| 3.2. Client type screen .....                                  | 5  |
| 3.3. Server screen .....                                       | 6  |
| 3.4. Selector screen .....                                     | 7  |
| 3.5. Policy screen .....                                       | 8  |
| 3.6. Firewall ruleset screen .....                             | 9  |
| 3.7. Current clients screen .....                              | 9  |
| 3.8. Log screen .....  | 10 |
| 3.9. License screen .....                                      | 10 |
| 3.10. Advanced administration .....                            | 10 |
| 3.10.1. How to enable verbose logging .....                    | 10 |
| 3.10.2. How the server configuration is found on startup ..... | 11 |
| 4. Ruleset Syntax .....  | 12 |
| 4.1. Summary of High-Level Rules .....                         | 12 |
| 4.2. Macros .....  | 12 |
| 4.3. Low-Level Rule Syntax .....                               | 13 |
| 4.4. High-Level Rule Expansion .....                           | 14 |
| 4.5. "opt" settings .....                                      | 17 |
| 4.6. ICMP types and codes .....                                | 17 |
| Glossary .....   | 19 |

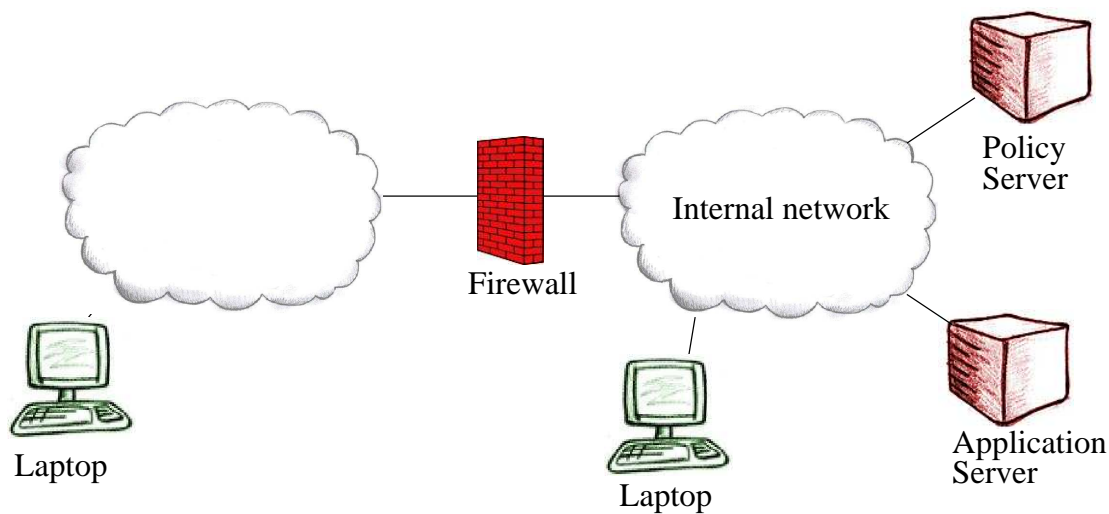
---

# Chapter 1. Overview

The AppGate Personal Firewall Policy Manager is a software for distributing policies to the AppGate Personal Firewall. It works by having one or more servers running the Policy Manager Software, these are known as *policy servers*. The users' Personal Firewalls queries the servers for which firewall rules they should use for the moment. All other configuration is also downloaded from the policy server. Different clients can get different policies depending on IP address and/or *client type*.

Each installation of AppGate Personal Firewall contains a default *policy* and a *client type*. The administration tool lets you create different installation packages with different default policies and type names. The policy included when installing such a package may later be replaced by the one provided by a policy server, but the client type never changes.

## 1.1. Typical usage



The above picture shows a typical usage case. The company network is protected by a corporate firewall. Some users have laptops which moves between the company network and the outside. The laptops should use different *rulesets* inside and outside the network. The internal servers should use a different *ruleset*.

Typically one would use the *client type* to select different *policies*. Laptops would be given one type and servers another. If one need different policies for different servers one can either use different types or select on IP address.

Each policy includes two rulesets: the default one which is used when the client can not talk to a policy server, and the active which is used when the client is in contact with a policy server. The laptop policy will have a fairly restrictive default ruleset but a more open active one. For the internal servers the active and default rulesets may be identical.

## 1.2. Policies

A policy consists of the following:

- A list of policy servers to query for new policies.
- An active ruleset. This ruleset is installed when the client first gets an answer from the policy server and is left installed until either the policy server does not answer a request, or the configuration on the server is updated or the network configuration changes.

- A default ruleset. This ruleset is used when the client can not reach any of the defined policy servers. This ruleset is always the one used at startup or network configuration change, it remains in use until the client has established connection with a policy server.

A policy server can contain multiple policies. When a client requests a policy the server checks its list of *selectors*. Each selector must match against both the client IP address and the client type (wild cards are available).

## 1.3. Protocol

The communication takes place over UDP on port 38203 (the port can be changed). The protocol is completely client driven and there are only two requests the client can send.

- `GET_POLICY` This requests the current policy for the client. The client includes its IP address and client type. The server replies with a signed message containing the policy the client should use as well as the *configuration identifier* identifying the current server configuration. This reply is unique for each `GET_POLICY` request.

The client will send a `GET_POLICY` request each times the network configuration changes or when the configuration identifier changes.

- `GET_CURRENT` The client uses this to get the current configuration identifiers from the server. If the configuration identifier has changed the client fetches a new policy with the `GET_POLICY`.

The time between `GET_CURRENT` requests, the *poll interval*, determines how quickly a new policy/ruleset is spread to the clients. The default interval is 15 minutes. Decreasing this time will put a greater load on the policy servers.

---

# Chapter 2. Installation

## 2.1. Windows Installation

The AppGate Policy Manager may be installed on Windows 2000 or later. When installing the Policy Manager on Windows a Sun Java 1.4.2 JRE will also be installed. The installation of this JRE is for the sole purpose of running the Policy Manager and will not affect any other Java installations or applications on the system.

The Policy Manager Daemon will install itself as a Windows NT service which will be started automatically when the computer starts.

The installation is started by running **agpolicy.exe**. The installer will prompt for a directory where the Policy Manager will be installed. By default it is `C:\Program Files\AppGate`. It is also possible to select to install the AppGate Personal Firewall Packaging tool.

### 2.1.1. AppGate Personal Firewall Packaging Tool

With this tool it is possible to make an executable installer for the AppGate Personal Firewall, which is pre-configured with a selected policy. This tool is only available when the Policy Manager is installed on a Windows platform.

### 2.1.2. Unattended Installation

It is possible to do an unattended installation of the AppGate Policy Manager by running **agpolicy.exe** in the following way:

```
agpolicy.exe /S [/D=installation directory]
```

## 2.2. Installation on other platforms

The AppGate policy Manager may be installed on any server that has a Java 1.4 SE (or later) Runtime Environment.

Observe that you should preserve the "conf" directory when upgrading.

The Policy Manager is installed by following these steps:

1. Unpack `agpolicy.tar.gz` in a directory of choice (e.g `/opt`).

```
# cd /opt ; gzcatt agpolicy.tar.gz | tar xf -
```

2. In the `bin` directory in the unpacked installation, there will be two shell scripts:

**agpadmin** Starts the Policy Manager Administration Tool.

**agpmd** Starts the Policy Manager Daemon.

These scripts has to be edited to use the correct path of the JRE by changing the following line:

```
JAVA_HOME=/usr/j2se
```

If the `JAVA_HOME` environment variable is already set, then it will be used.

3. In the `bin` directory in the unpacked installation, there is a shell script named **agpolicy.init**. This is a System V style init script that may be used to start the Policy Manger Daemon when the system is started. To use this, copy the file to the appropriate location for the operating system in question and then make it start at the desired run level.

For example, assume the Policy Manger is installed in the `/opt` on a server running Solaris and that we want to start the Policy Manger Daemon on run level 3. This would involve the following commands:

```
# cp /opt/agpolicy/bin/agpolicy.init /etc/init.d/agpolicy
# cd /etc/rc3.d
# ln -s ../init.d/agpolicy S99agpolicy
```

## 2.3. Securing the installation

It is vital that the Policy Server is adequately protected, to prevent the distribution of policies from being compromised.

The exact steps to securing a given platform varies, but a minimum would be:

- Ensure the system is updated with the latest patches.
- Turn of unnecessary services.
- Limit access to the system.

If it is a Windows system it is possible to use the AppGate Personal Firewall to further protect the machine. A suitable ruleset for a computer that only acts as a Policy Server is:

```
block-bad
allow-in port 38203 udp
block
```

### 2.3.1. File Access Rights

Since the Policy Manager operates on files it is important that all the files in the `conf` directory of the Policy Manager installation is writable to those and only those that should be able to administer the Policy Manager.

---

# Chapter 3. Administration

The policy manager daemon is controlled by a configuration file plus a directory containing the different rulesets. There is a graphical administration tool which helps generating the configuration. The daemon also generates a number of state and log-files which the administration tool reads to give feedback about the operational status.

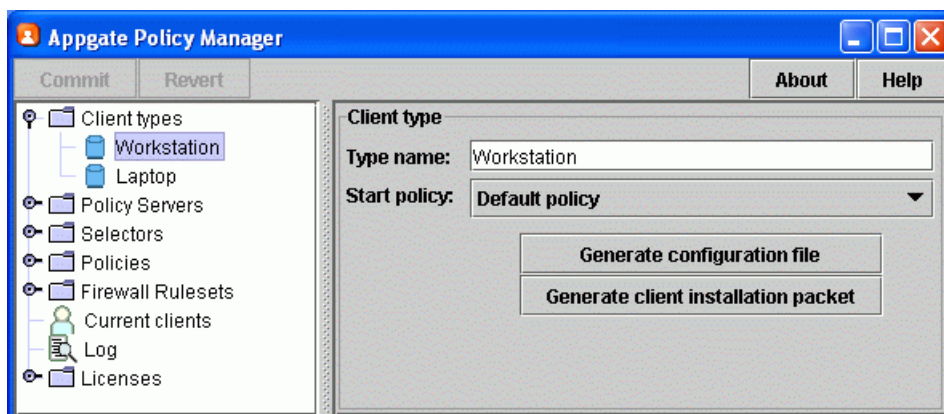
The daemon will check the configuration once every second and reloads the configuration as soon as it detects any changes. This means it is perfectly possible to do most administration by editing the files directly.

## 3.1. Working with the Administration tool

The Administration tool is not client-server based and must be run on the actual host that runs the Policy Manager. After installation of the Policy Manager may be launched. On Windows it should typically be found among the programs in the Start menu.

The Administration tool will display the contents of the configuration files and log files and allow for changes to be made. If any changes are made to the configuration the Commit should be pressed to make the changes be write and take effect. The Revert may be used to re read the content from the configuration files. It will reset any changes made in the GUI dialogs to match the running configuration.

## 3.2. Client type screen



This screen is where complete client configurations, and possibly installation packages, are created. A client configuration includes a client type as well as a policy. The policy may later be updated from a policy server but the type is never changed.

Each configuration is defined by the following attributes:

Type name            The type of the client. This is the value which later on will be checked against the selectors.

Start policy         The initial policy to install on the client.

Additionally there are two buttons to create configurations.

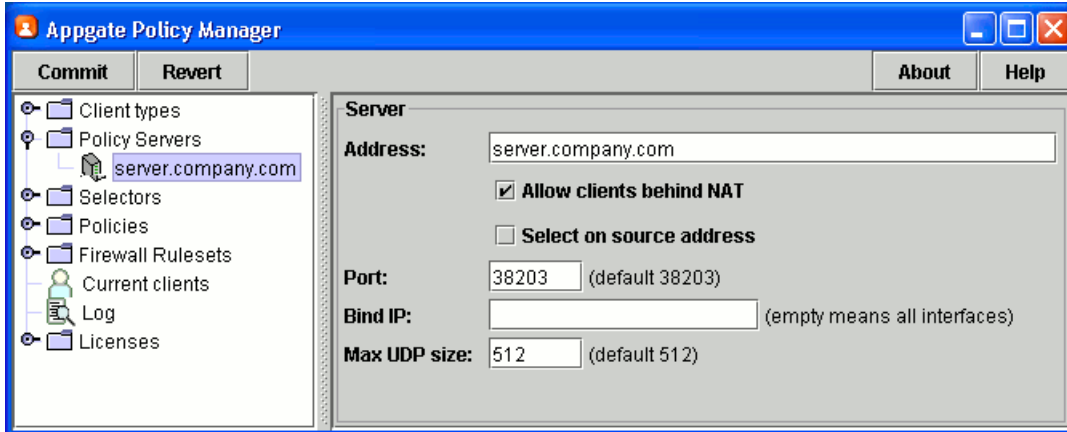
Generate configuration file         This is used to create a configuration file which can later be manually installed on clients. The administration tool will ask for the name of the file to generate.

Normally this configuration file should be installed as "C:\Windows\Ap-pGate Personal Firewall\policy.cfg" on the client.

Generate client installation package         This button is only enabled when the administration tool is running on a windows system. It will generate an installation package also containing a

configuration file. The tool will ask for a file to generate and one should typically give a file name ending with `.exe`. When the resulting file runs on a client PC it will install the firewall client software as well as the configuration file.

### 3.3. Server screen

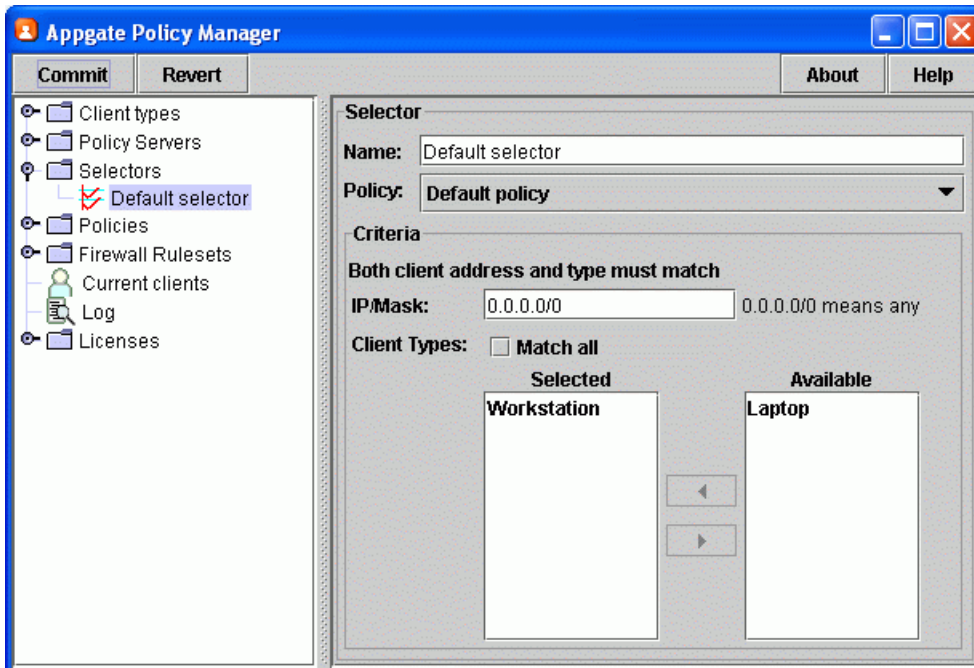


This is where policy servers are defined. The tree on the left lists all defined servers. The order is significant and can be changed by dragging the server with the mouse.

Each policy server is defined by the following attributes:

|                          |  |
|--------------------------|--|
| Address                  | The address of the server host. This is the address clients will use when trying to contact the host. It can be either a DNS name or an IP address.  |
| Allow clients behind NAT | Client requests include the IP address the client thinks it has. If this address differs from the source address of the packet the daemon assumes the client is located behind a NAT bridge.   |
| Select on source address | Client requests include the IP address the client thinks it has. Normally this address is used when selecting a policy, but if this attribute is set the request's source address is used instead.   |
| Port                     | Which port the server listens to.  |
| Bind IP                  | The address of the interface the daemon should listen to. The default value is empty which means that the daemon should listen to all interfaces.  |
| Max UDP Size             | Maximum size of UDP packets sent from the server. Replies which are bigger will be split into multiple smaller UDP packets. Experience shows that a lot of networks can not handle big UDP packets so the default is to split packets bigger than 512 bytes. |

## 3.4. Selector screen

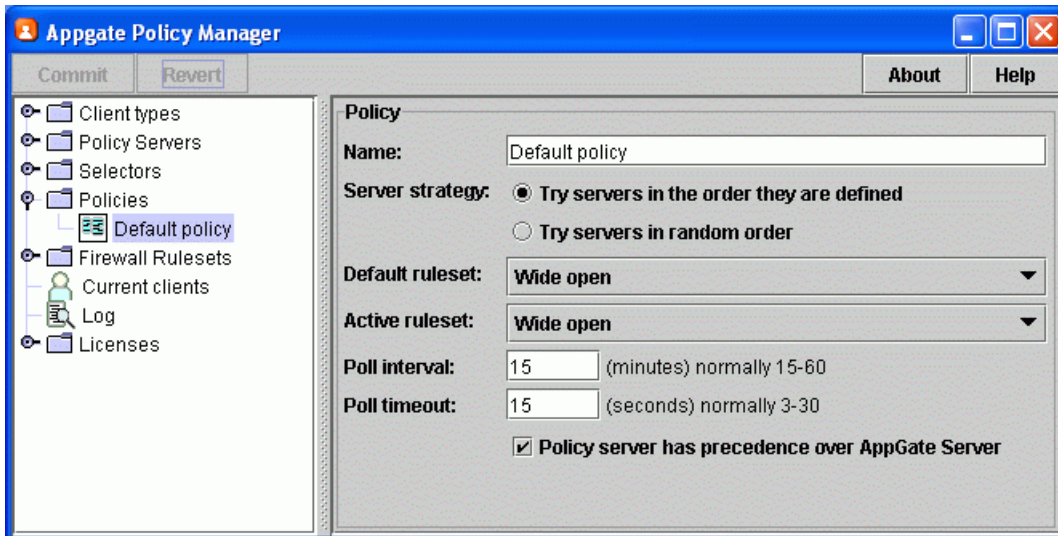


Selectors are the way policy servers determine which policy to give to which client. The tree on the left lists all defined selectors. The order is significant and can be changed by dragging the selector with the mouse. The server will go through the list of selectors from the top until one matches.

Each selector is defined by the following attributes:

|              |  |
|--------------|--|
| Name         | The name of the selector. This name is only used for administration.   |
| Policy       | The policy to put on clients that match this selector.   |
| IP/Mask      | The client IP must match this expression. The mask should be expressed as number of significant bits to match. A mask of 0 will match all clients. Which IP number is used depends on the attribute "Select on source address" attribute of the policy server. |
| Client types | The type of client this selector applies to. If Match all is checked then all client types matches, otherwise one those client types in the Selected list matches.   |

## 3.5. Policy screen

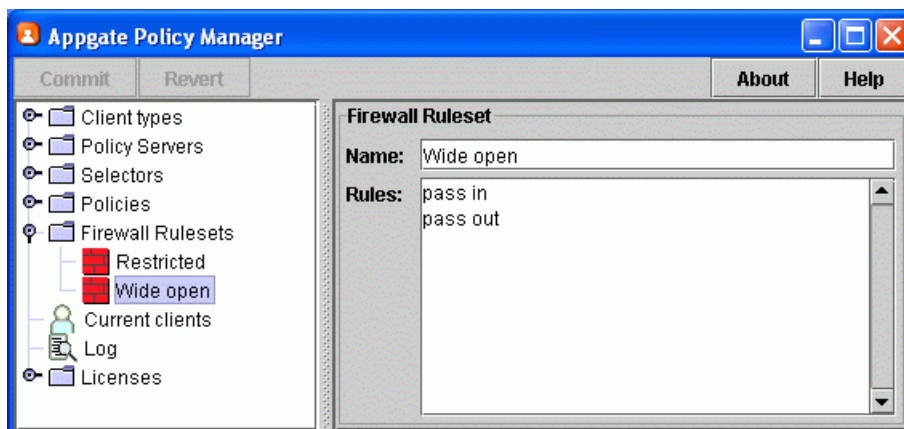


A policy is the configuration on a client which is updated from the policy server. It includes all the defined policy servers as well as the data referenced to on this screen.

Each policy is defined by the following attributes:

|  |   |
|--|---|
| Name   | The name of the policy. This name is only used for administration.  |
| Server strategy                                  | Controls in which order clients tries to talk to the different policy servers. Random order is useful for load balancing between different servers. Trying the policy servers in order is useful when you have different servers serving different subnets.                             |
| Default ruleset                                  | The name of the default ruleset. This ruleset will be used by the client when it can not talk to any policy server.   |
| Active ruleset                                   | The name of the active ruleset. The client will use this ruleset as long as it can talk to the policy server or until the policy changes.   |
| Poll interval                                    | How often the client should check back with the policy server. These checks will notice both if the configuration has been updated or that the client no longer can contact the policy server. beware that checks are also triggered whenever the client network configuration changes. |
| Poll timeout                                     | How long to wait for an answer from the policy server before assuming that it is unreachable.   |
| Policy server has precedence over AppGate server | If you use the AppGate Security Server then there might be a conflict because the AppGate security server may also want to mandate a policy on the client. This option controls if that is permitted or not.  |

## 3.6. Firewall ruleset screen

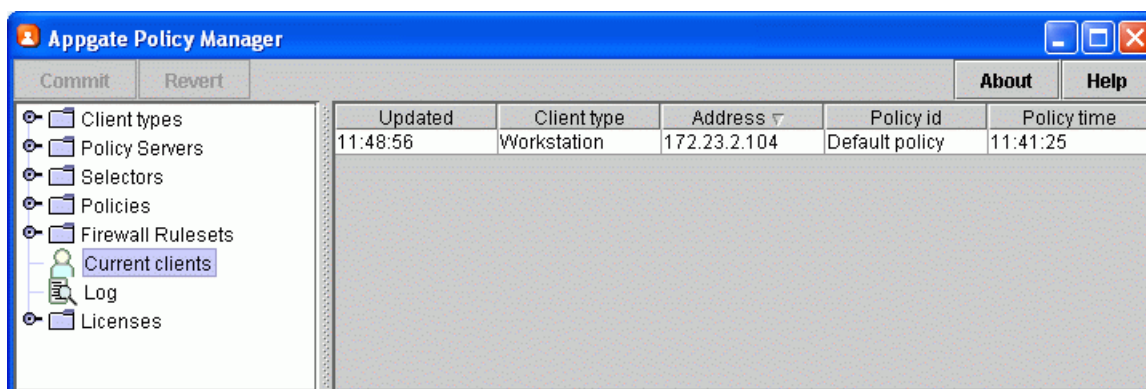


This is a collection of firewall rules to use. Only one ruleset is active at any single time.

Each ruleset is defined by the following attributes:

- Name** The name of the ruleset. This name is only used for administration.
- Rules** The actual firewall rules. For syntax see [Summary of High-Level Rules](#). The rule editor has error checking built in. If an error is found, the illegal text is colored with red. To find out how to fix the error, hover with the mouse cursor on the red text.

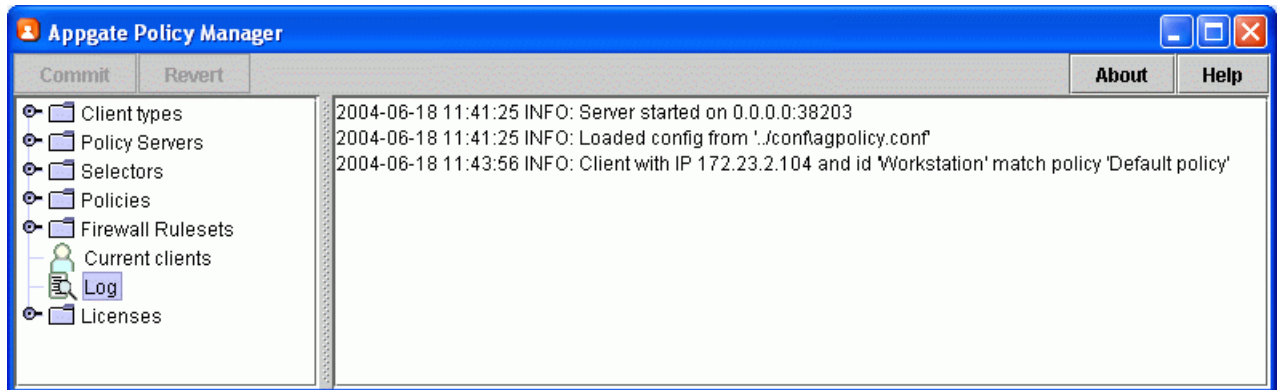
## 3.7. Current clients screen



This screen shows the clients which have contacted the server during the last two hours. The table contains the following columns.

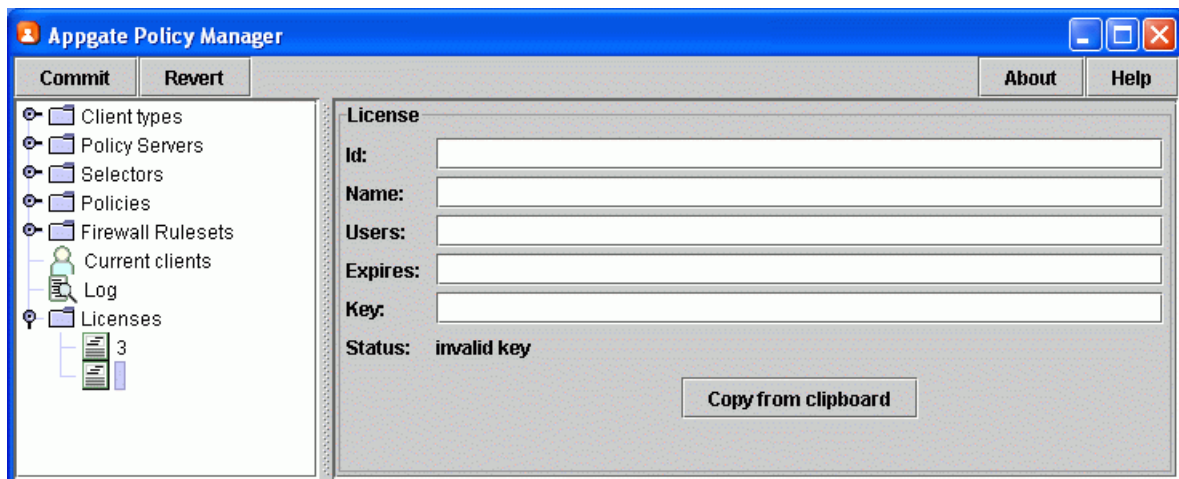
- Updated** When the server last received a packet from the server.
- Client type** The type of the client.
- Address** The IP address the client claims to have. If this differs from the source address of the packet then the source address are shown within [ ].
- Policy id** Which policy the client is running.
- Policy time** When the policy the client actually is using was last updated. This field lets you see if a client has picked up the latest changes to the policy.

## 3.8. Log screen



This screen shows log messages from the policy manager daemon.

## 3.9. License screen



The licensing is per concurrent users. All users who have connected within a certain period are counted. The server will not send answers to any new clients if all licenses have been used up.

The license data should be entered exactly as received from AppGate. Case and punctuation is significant.

It is possible to copy the license data to the license screen by using the system clipboard. First copy the license data to the clipboard and then press the Copy from clipboard button.

## 3.10. Advanced administration

### 3.10.1. How to enable verbose logging

The amount of logs generated and stored can be changed by setting properties in a file named `settings.txt` located in the `agpolicy` installation directory. There should already be a suitable property in the file which is commented out. Just uncomment the line so that it reads:

```
com.appgate.agpolicy.level=ALL
```

The Appgate policy manager needs to be restarted when the `settings.txt` file has been changed.

### **3.10.2. How the server configuration is found on startup**

The Policy Manager configuration may contain multiple server entries. This means that each server must figure out which of these server entries is applicable to itself when starting.

The server first looks through the configuration file for server entry where the name is an exact match of the name of the local host. If that fails it looks up all IP-addresses associated with the local host name. It will then look up the IP-addresses of all the server entries in the configuration file. It will take the first server entry where any of the addresses matches.

The server will write detailed messages to the log if no matching entry is found. These messages shows which names and addresses it found for itself and for all of the defined server entries.

---

# Chapter 4. Ruleset Syntax

## 4.1. Summary of High-Level Rules

`block-bad [<log spec>]`

Block packets that should not appear in non-malicious network traffic but can be used by attackers. Recommended to always be the first rule of any rule set.

`allow-outgoing [<log spec>]`

Allow outgoing traffic and any incoming packets belonging to a session initiated by an outgoing packet.

`allow-util [<log spec>]`

Allow some ICMP messages that may be needed for IP to function optimally. These messages are normally passed by a `pass with-state`, but `allow-util` also permits packets belonging to sessions that were not permitted by a `keep-state-rule`.

`allow-out [to <host>] port <portnum> [udp | tcp] [<log spec>]`

Allow the specified outgoing traffic and any associated incoming packets.

`allow-out [to <host>] [ dns | dhcp | ping ] [<log spec>]`

Allow the specified outgoing traffic and any associated incoming packets.

`allow-out udp-broadcast to [<broadcastaddress>] port <port>[-<port>] [<log spec>]`

Allow outgoing UDP-broadcasts and any associated replies.

`allow-in [from <host>] port <portnum> [udp | tcp] [<log spec>]`

Allow the specified incoming traffic and any associated outgoing packets.

`allow-in [from <host>] [ dns | dhcp | ping ] [<log spec>]`

Allow the specified incoming traffic and any associated outgoing packets.

`allow-in udp-broadcast [from <network>/<mask>] to port <port>[-<port>] [<log spec>]`

Allow incoming UDP-broadcasts and any associated replies.

`allow-ipt [<log spec>]`

Allow traffic to and from the ip-tunnel interfaces. Even if the client allows all traffic on these interfaces, the AppGate server will filter the traffic.

`allow-appgate [<log spec>]`

Allow ssh traffic to the AppGate server.

`allow-policy-manager [<log spec>]`

Allow communication with the policy manager.

## 4.2. Macros

Two macros are defined: *myaddress* which expands to all the addresses owned by the client, and *mynet* which expands to all locally connected networks. When these macros are used, they are replaced with *hasflag to-myaddress*, *hasflag from-myaddress*, *hasflag to-mynet* or *hasflag from-mynet* depending on whether it is used as a from- or to-address in a rule. If a flag that is used in a rule hasn't been defined, a rule group that defines that flag is created automatically.

## 4.3. Low-Level Rule Syntax

```
<rule>:  
  <action> [<direction>] [<log spec>] [from <addr>]  
  [to <addr>] [<proto>] [<ipopts>] [short-frag]  
  [<flags>] [<related state>] [with-state] [keep-state] [count]
```

```
<action>:  
  pass | block [return (RST | HOSTUNREACH | PORTUNREACH | PROHIB)] |  
  setflag <flagname> | clearflag <flagname>
```

```
<direction>:  
  in | out | inout
```

```
<log spec>:  
  log | log-verbose | log-crit
```

```
<addr>:  
  [not] (<ip>[/<prefixlen>] | any | myaddress | mynet)  
  [port <portspec>]
```

```
<portspec>:  
  port [not] <portnum>[-<portnum>]
```

```
<proto>:  
  proto (<icmp-spec> | <tcp-spec> | <udp-spec> | <proto-num>)
```

```
<icmp-spec>:  
  ICMP [<icmptype>[/<icmpcode>]]
```

```
<tcp-spec>:  
  TCP [flags <tcpflags>[/<tcpflags>]]
```

```
<tcpflags>:  
  [F][S][R][P][A][U]
```

```
<udp-spec>:
```

UDP

```
<ipopts>:
  with-ipopts[/(any | (<ipoptlist>/<ipoptlist>))]
```

```
<ipoptlist>:
  [NOP][SEC][LSR][SSR][RR][SID][TS][?]
```

```
<flags>:
  <flag> [<flag> [<flag> [<flag>]]]
```

```
<flag>
  (hasflag | not-hasflag) <flagname>
```

```
<related state>:
  (hastcpstate | hasudpstate) local <statespec> remote <statespec>
```

```
<statespec>:
  (l_ip | r_ip | <ip>/<prefixlen>)]
  port (l_port | r_port | <portnum>[-<portnum>])
```

## 4.4. High-Level Rule Expansion

block-bad [<log spec>]

Expands to:

```
block <log spec> with-ipopts
block <log spec> short-frag
block <log spec> from 127.0.0.0/8
block <log spec> to 127.0.0.0/8
```

allow-outgoing [<log spec>]

Expands to:

```
pass out <log spec> proto TCP keep-state
block in proto TCP flags A/SAR
pass out <log spec> proto UDP keep-state
pass out <log spec> proto ICMP/ECHO keep-state
pass out <log spec> proto ICMP
```

allow-util [<log spec>]

Expands to:

```
pass <log spec> proto ICMP/UNREACH
pass <log spec> proto ICMP/SOURCEQUENCH
pass <log spec> proto ICMP/TIMXCEED
pass <log spec> proto ICMP/PARAMPROB
```

```
allow-out [to <host>] port <portnum> udp [<log spec>]
```

Expands to:

```
pass out <log spec> from myaddress to <host> port <portnum> \
    proto UDP keep-state
```

```
allow-out [to <host>] port <portnum> [tcp] [<log spec>]
```

Expands to:

```
pass out <log spec> from myaddress to <host> port <portnum> \
    proto TCP keep-state
block in from <host> port <portnum> to myaddress proto TCP flags A/SAR
```

```
allow-out [to <host>] dns [<log spec>]
```

Expands to:

```
pass out <log spec> from myaddress to <host> port 53 proto TCP keep-state
block in from <host> port 53 to myaddress proto TCP flags A/SAR
pass out <log spec> from myaddress to <host> port 53 proto UDP keep-state
```

```
allow-out [to <host>] dhcp [<log spec>]
```

Expands to:

```
pass out <log spec> to <host> port 67 proto UDP
pass in <log spec> from <host> to any port 68 proto UDP
```

```
allow-out [to <host>] ping [<log spec>]
```

Expands to:

```
pass out <log spec> from myaddress to <host> proto ICMP/ECHO keep-state
```

```
allow-out udp-broadcast to [<broadcastaddress>] port <port>[-<port>] [<log
spec>]
```

Expands to:

```
pass out <log spec> from myaddress to <broadcastaddress> port <portspec> \
    proto udp keep-state
```

```
allow-in [from <host>] port <portnum> udp [<log spec>]
```

Expands to:

```
pass in <log spec> from <host> to myaddress port <portnum> \  
    proto UDP keep-state
```

```
allow-in [from <host>] port <portnum> [tcp] [<log spec>]  
Expands to:
```

```
pass in <log spec> from <host> to myaddress port <portnum> \  
    proto TCP keep-state  
block out from myaddress port <portnum> to <host> proto TCP flags A/SAR
```

```
allow-in [from <host>] dns [<log spec>]  
Expands to:
```

```
pass in <log spec> from <host> to myaddress port 53 proto TCP keep-state  
block out from myaddress port 53 to <host> proto TCP flags A/SAR  
pass in <log spec> from <host> to myaddress port 53 proto UDP keep-state
```

```
allow-in [from <host>] dhcp [<log spec>]  
Expands to:
```

```
pass in <log spec> from <host> to myaddress port 67 proto UDP  
pass out <log spec> from myaddress to <host> port 68 proto UDP
```

```
allow-in [from <host>] ping [<log spec>]  
Expands to:
```

```
pass in <log spec> from <host> to myaddress proto ICMP/ECHO keep-state
```

```
allow-in udp-broadcast [from <network>/<mask>] to port <port>[-<port>] [<log  
spec>]  
Expands to:
```

```
pass in <log spec> from <address> to <broadcastaddress> port <portspec> \  
    proto udp keep-state
```

```
allow-ipt [<log spec>]  
Expands to:
```

```
pass out <log spec> from <tunnel interface address> to any  
pass in <log spec> from any to <tunnel interface address>
```

```
allow-appgate [<log spec>]  
Expands to:
```

```
pass out <log spec> from myaddress [port <localport>] to <AppGate> \  
    proto tcp keep-state
```

```

    port <ssh port> proto TCP keep-state
block in from &AppGate; port <ssh port> to myaddress [port <localport>] \
    proto TCP flags A/SAR

```

allow-policy-manager [<log spec>]

Expands to:

```

    pass out <log spec> from myaddress to <Policy Manager> port <port> \
    proto UDP keep-state

```

## 4.5. "opt" settings

opt settings must come first in a ruleset. Currently only one opt parameter is defined.

opt clear-states-on-load (true | false)

This option specifies whether the state table should be cleared when the rule set is loaded. If this is set to false, any states from the previous rule set are preserved. The default is to clear the state table.

## 4.6. ICMP types and codes

ECHOREPLY

UNREACH

NET

HOST

PROTOCOL

PORT

NEEDFRAG

SRCFAIL

NETUNKNOWN

HOSTUNKNOWN

ISOLATED

NETPROHIBITED

HOSTPROHIBITED

TOSNET

TOSHOST

PROHIBITED

HOSTPRECEDENCE

PRECEDENCECUTOFF

SOURCEQUENCH

REDIRECT

NET

HOST

TOSNET

TOSHOST

ALTHOSTADDR

ECHO

ROUTERADVERT

NORMAL

NOROUTECOMMON

ROUTERSOLICIT

TIMXCEED

TRANSIT

REASSEMBLY

PARAMPROB  
    ERRATPTR  
    OPTABSENT  
    LENGTH  
TIMESTAMP  
TIMESTAMPREPLY  
INFOREQUEST  
INFOREPLY  
MASKREQUEST  
MASKREPLY  
TRACEROUTE  
DATACONVERR  
MOBILEREDIRECT  
IPV6WHEREAREYOU  
IPV6IAMHERE  
MOBILEREQUIREMENT  
MOBILEREQUIREMENTREPLY  
SKIP  
PHOTURIS  
    UNKNOWNINDEX  
    AUTHFAILED  
    DECRYPTFAILED

---

# Glossary

|                          |  |
|--------------------------|--|
| Client type              | An arbitrary identifier which can be used when selecting policies. The client type is included in the client configuration and is never updated.   |
| Configuration identifier | A hexadecimal string which uniquely identifies a certain configuration. The configuration includes all servers, selectors, policies and rulesets defined on a policy server.   |
| Policy                   | A policy is the dynamic configuration of the client. It contains two rulesets and a list of policy servers to check for updates from. See Policies.  |
| Policy Manager           | This is the software responsible for managing and distributing policies.   |
| Poll interval            | How often the personal firewall will check with the policy server if there is a new policy available. Usually measured in minutes.   |
| Ruleset                  | A collection of firewall rules.  |
| Selector                 | A configuration element on the policy server which matches a certain set of client addresses and types. The selector indicates which policy should be applied to the matching clients. The server has an ordered list of selectors and the first matching one is used. |
| Policy Server            | One machine which runs the Policy Manager software.  |