Tntnet configuration-reference

Author: Tommi Mäkitalo

Date: 2010-06-21

For Tntnet version 2.0

Table of contents

AccessLog

Syntax

`AccessLog` *`filename`*

Description

If a filename is given, a logentry for each request is written to the file. The format of the file is the same as in other common web servers.

Example

```
AccessLog /var/log/tntnet/access.log
```

BufferSize

Syntax

`BufferSize` *`bytes`*

Description

Specifies the number of bytes sent in a single system-call. This does not limit anything in application-level. It does not affect e.g. savepoints or exception-handling. Component-output is collected completely and then passed in chunks of *BufferSize* bytes to the operating system.

The default value for this is 16384.

Example

```
BufferSize 32768
```

CompLifetime

Syntax

`CompLifetime` *`seconds`*

Description

Components are instatiated when needed and reused. After the time specified here the instances are deleted. This defaults to 600 seconds.

Example

```
CompLifetime 120
```

CompPath

Syntax

`CompPath` *`directory`*

Description

CompPath specifies, where tntnet should search for webapplications. Tntnet

searches first in the current directory and then in each directory, you specify here, until a library is found. You can repeat the directive as many times as desired to add more entries. If it is not found, the next *MapUrl*-entry is tried.

Example

```
CompPath /usr/local/lib/tntnet
CompPath /usr/local/share/tntnet
```

Chroot

Syntax

Chroot *directory*

Description

Does a `chroot(2)`-system call on startup, which locks the process into the directory at system-level.

Example

```
Chroot /var/tntnet
```

Daemon

Syntax

Daemon 0|1

Description

If this flag is set to 1, Tntnet forks at startup and terminates the parent-process on successful initialization.

Example

```
Daemon 1
```

DefaultContentType

Syntax

DefaultContentType *contenttype*

Description

Sets the default content type. This content type is set in the http header, if the component does not set it. The default is “text/html; charset=UTF-8”

Example

```
DefaultContentType "text/html; charset=ISO8859-1"
```

Dir

Syntax

`Dir` *directory*

Description

Changes the current working directory of the process on startup.

Example

```
Dir /var/tntnet
```

EnableCompression

Syntax

`EnableCompression` *yes*|*no*

Description

Specifies, if Tntnet should use gzip-compression at http-level. By default Tntnet use compression.

A http-client like a web browser can send a header “Accept-Encoding”, to tell Tntnet, that it would accept compressed data. Tntnet then can decide, if it use compression. When the body is complete, Tntnet tries to compress the body. If the data can be compressed by more than 10%, Tntnet sends this compressed data. With this flag, this feature can be turned off.

Compression slows down processing but reduces the network-load. Normally the size of html-pages can be compressed by about 70%, while Tntnet slows down by up to 30%.

Example

```
EnableCompression no
```

Group

Syntax

`Group` *unix-group*

Description

Changes the group under which tntnet answers requests.

Example

```
Group tntnet-group
```

include

Syntax

`include` *filemask*

Description

Reads additional settings from the specified files. *filemask* might contain glob-characters, so that multiple files might be read. The order is not specified.

Example

```
include /etc/tntnet.d/*.conf
```

KeepAliveTimeout

Syntax

`KeepAliveTimeout` *`milliseconds`*

Description

Sets the timeout for keep-alive requests. Tntnet tries to do keep-alive-requests wherever possible. This has the effect, that tntnet can receive multiple requests within a single tcp-connection. The connection times out after *KeepAliveTimeout* milliseconds. The timeout defaults to 15000ms.

Example

```
KeepAliveTimeout 300000
```
sets the keep-alive-timeout to 5 minutes.

KeepAliveMax

Syntax

`KeepAliveMax` *`number`*

Description

Sets the maximum number of request per tcp-connection. This defaults to 100.

Example

```
KeepAliveMax 10
```

Listen

Syntax

`Listen` *`ip [port]`*

Description

Specifies, on which address tntnet waits for connections. There can be more than one `Listen`-directives, in which case tntnet waits on every address. If there is no `Listen`-directive tntnet listens on 0.0.0.0 port 80.

ip might also be a hostname.

Example

```
Listen 127.0.0.1 8000
```
Listens on localhost on port 8000.

ListenRetry

Syntax

```
ListenRetry number
```

Description

On startup Tntnet calls listen on the specified port. When the systemcall returns with an error, Tntnet tries again and fails after the specified number of attempts. The default number is 5.

Example

```
ListenRetry 10
```
Retries 10 times

ListenBacklog

Syntax

```
ListenBacklog number
```

Description

The system-call listen needs a parameter backlog, which specifies, how many pending connections the operating-system should queue before it starts to ignore new request. The value is configurable here.

The default value is 16.

Example

```
ListenBacklog 64
```
Changes the backlog-queue to 64.

Load

Syntax

```
Load webapplication
```

Description

Load specifies, which webapplications are preloaded on startup. Normally webapplications are loaded as needed. The disadvantage is, that there is no way to check on startup, if a application is loadable at all. With this directive startup fails, if the application is not loadable.

Example

```
Load myapp
```

MapUrl

Syntax

```
MapUrl url component-identifier [ path-info { additional-arguments } ]
```

Description

Tells tntnet, which component should be called, when it receives a http-request. *url* is a regular expression, which is tried against the request-url. If it matches, the *component-identifier* is evaluated. *component-idetifier* may contain backreferences to the url. By default the url is passed as path-info to the component, but this can be changed with a third parameter. Additional parameters can be passed to the component and accessed through tnt::httpRequest::getArgs().

This variable can occur more than once and they are tried in the order they are found in the configurationfile, until the regular expression matches and the component does not return tnt::DECLINED. If no MapUrl-directive is found, http-error 404 (not found) is sent.

Example

```
# maps html-pages to components in myapp.so; e.g. /foo.html calls foo@myapp
MapUrl /([^/.]+)\.html  $1@myapp
# maps jpeg-urls to myapp; e.g. /foo.jpeg calls foo_jpg@myapp
MapUrl /([^/.]+)\.jpeg  $1_jpg@myapp
# maps /foo/bar.html to bar@foo
MapUrl /([^/.]+)/([^/.]\.html $2@$1
```

MaxUrlMapCache

Syntax

```
MaxUrlMapCache size
```

Description

As described in MapUrl urls are mapped to components with regular expressions. This is a quite expensive operation, while the number of different urls used in a typical web application is small. Therefore Tntnet has a simple cache, which stores mappings to prevent the need to process the same regular expression multiple times. The size of this cache is limited. After the size is exceeded the cache is simply cleared. This clearing is logged with the message “clear url-map-cache”. If you have a application whit many different urls and you often see this warning-message, you might want to increase the cache.

The default value is 8192.

Example

```
MaxUrlMapCache 32768
```

MaxRequestSize

Syntax

`MaxRequestSize` *number*

Description

This directive limits the size of the request. After *number* Bytes the connection is just closed.

This prevents deniel-of-service-attacks through long requests. Every request is read into memory, so it must fit into it.

Bear in mind, that if you use file-upload-fields a request might be larger than just a few bytes.

The value defaults to 0, which means, that there is no limit at all.

Example

```
MaxRequestSize 65536
```

MaxUrlMapCache

Syntax

`MaxUrlMapCache` *number*

Description

Tntnet has a cache, which stores results from the MapUrl setting. Since processing regular expressions as used in MapUrl does take time and often only a few distinct urls are processed, the result is cached, so that the regular expression is executed only once. The cache is very simple. When the cache is full, is it just discarded and the caching starts from the beginning.

Setting the cache size may affect the performance of tntnet a little. It is settable here. The default size is 8192 entries.

Example

```
MaxUrlMapCache 256
```

MimeDb

Syntax

`MimeDb` *file*

Description

The static@tntnet component, which sends files from the file system, uses the specified file to look up the content type depending on the file extension. This is by default */etc/mime.types*.

Example

```
MimeDb /usr/local/etc/mime.types
```

MinCompressSize

Syntax

```
MinCompressSize number
```

Description

Http-compression for replies smaller than this are not compressed at all. The default value for this is 1024.

Example

```
MinCompressSize 1024
```

MinThreads

Syntax

```
MinThreads number
```

Description

Tntnet uses a dynamic pool of worker-threads, which wait for incoming requests. *MinThreads* specifies, how many worker threads there have to be. This defaults to 5.

Example

```
MinThreads 10
```

MaxThreads

Syntax

```
MaxThreads number
```

Description

Tntnet uses a dynamic pool of worker-threads, which wait for incoming requests. *MaxThreads* limits the number of threads. The default is 10.

Example

```
MaxThreads 20
```

NoCloseStdout

Syntax

```
NoCloseStdout 0|1
```

Description

If *Daemon* is set to 1, tntnet closes standard-in-, standard-out- and standard-error-channels, unless *NoCloseStdout* is set to 1.

Example

```
NoCloseStdout 1
```

NoMonitor

Syntax

```
NoMonitor 0|1
```

Description

On startup, if *Daemon* is set to 1, tntnet does a second fork. The parent keeps waiting for the child and the child does the work. If tntnet crashes, what might happen, when a webapplication is buggy, the worker-process is restarted. This can be switched off by setting *NoMonitor* to 0.

Example

```
NoMonitor 0
```

PidFile

Syntax

PidFile *filename*

Description

When run in daemon-mode, tntnet writes the process-id of the monitor-process to *filename*. When the monitor-process is deactivated, the pid of the worker-process is written. This ensures, that sending a sigkill to the the stored process-id stops tntnet.

Example

```
PidFile /var/run/tntnet.pid
```

PropertyFile

Syntax

PropertyFile *filename*

Description

This directive specifies the property-file, where logging is configured. The content of the property-file is dependend of the underlying logging-library.

Example

```
PropertyFile /etc/tntnet/tntnet.property
```

QueueSize

Syntax

`QueueSize` *`number`*

Description

Tntnet has a request-queue, where new requests wait for service. This sets a maximum size of this queue, after wich new requests are not accepted. The default value is 100.

Example

```
QueueSize 50
```

SessionTimeouot

Syntax

`SessionTimeout` *`seconds`*

Description

This sets the number of seconds without requests after which a sesssion is erased. The default value is 300 seconds.

Example

```
SessionTimeout 600
```

SocketReadTimeout

Syntax

`SocketReadTimeout` *`milliseconds`*

Description

A worker-thread waits for some milliseconds on incoming data. If there is no data, the job is put into a queue and another thread waits with poll(2) on incoming data on multiple sockets. The workerthreads are freed and they can respond to other requests quickly. The default value is 200 milliseconds, which is good for normal operation. A value of 0 results in non-blocking read.

If timeout is reached, this does not mean, that the socket is closed.

A small timeout reduces contextswitches on slow connections.

Example

```
SocketReadTimeout 0
```

SocketWriteTimeout

Syntax

`SocketWriteTimeout` *`milliseconds`*

Description

This defines the time, how long the workerthreads wait on write. If the timeout is exceeded, the socket is closed and the browser might not get all data. The default value is 10000 milliseconds.

Example

```
SocketWriteTimeout 20000
```

SslListen

Syntax

`SslListen` *`ip [port[ssl-certificate-file [ssl-key-file]]]`*

Description

Specifies, on which ip and port tntnet waits for incoming ssl-connections. Optionally a certificate- and key-file can be passed.

Example

```
SslListen 192.168.0.1 8443
```

Waits on the specified address and port on incoming ssl-connections. The server is reachable with https://192.168.0.1:8443/

ThreadStartDelay

Syntax

`ThreadStartDelay` *`ms`*

Description

Example

```
ThreadStartDelay 1000
```

User

Syntax

`User` *`username`*

Description

Example

```
User www-data
```

VMapUrl

Syntax

```
VMapUrl host url component-identifier [ path-info { additional-arguments } ]
```

Description

This is like MapUrl, but is specific for the virtual host.

This rule matches only if the host and the url matches against the specified values. Both are regular-expressions, so one rule can also match multiple hosts.

Example

```
# maps request for the host www1.tntnet.org to application1
VMapUrl www1.tntnet.org /([^/.]+)  $1@application1
# maps request for the host www2.tntnet.org to application2
VMapUrl www2.tntnet.org /([^/.]+)  $1@application2
# maps all calls to port 8000 to myapp
VMapUrl .*:8000 /([^/.]+)    $1@myapp
```