Witango 5 Professional Server Configuration Guide

February 2003

With Enterprise Pty Ltd
44 Miller St
North Sydney NSW 2060
Australia
PH: +61 2 9460 0500
FX: +61 2 9460 0502

www.witango.com

Table of Contents

Introduction

For low to moderate traffic on your web site, one Standard Witango Server is generally sufficient to sustain acceptable performance. As your web site grows and traffic volume or user concurrency increases, processing of requests naturally becomes slower. At some point, you may consider adding one or more extra Witango Servers processes and/or machines running Witango Server processes to improve performance.

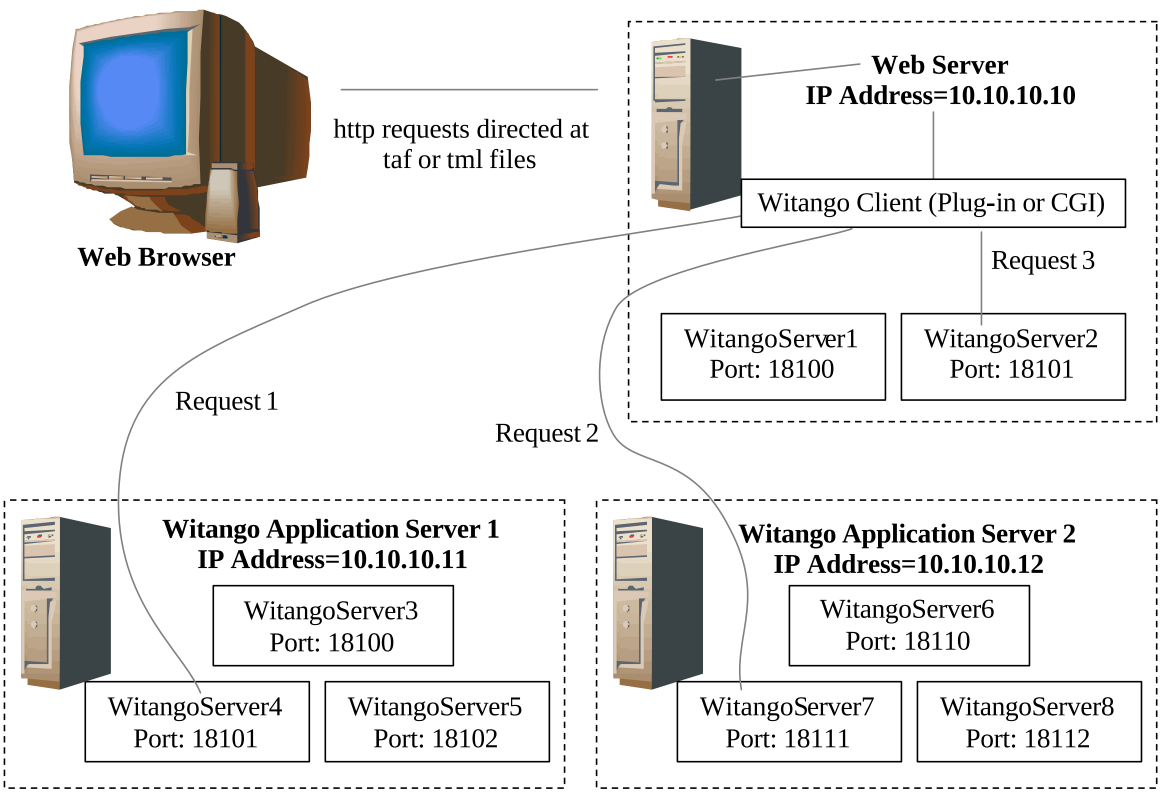
When running more than one Witango Server for your web site, you need to split the web site traffic to balance the load between the multiple server processes. Witango allows you to load split without having to alter your Witango application files . Thus you do not have to go through another cycle of development and testing of your applications as your web site grows. Adding a new Witango Server to your load group is quick and easy.

Understanding Load-Splitting

You can add Witango Servers to your system by installing the additional Witango Server processes on the same machine as an existing Witango Server (using a Professional license) or on a separate machine. The Witango Client configuration file (*clients.ini*) stores the information on how the load splitting is to be carried out. The Witango Client forwards requests to the various Witango Servers in the load group, based on settings in the *clients.ini* file. The Witango Client can be either a web server specific plug-in or a cgi executable.

How the Load-Splitting client works

To demonstrate how the load splitting client works, suppose you have eight Witango Servers distributed across three machines. Each of these machines has a unique IP address and each Witango Server on each of these machines is listening on a unique port (LISTENERPORT in the *witango.ini* file). This setup could be configured like this:



In this configuration, 2 machines are dedicated Witango Application Servers and the third runs the web server process, the Witango Load Splitting Client and 2 Witango Application Servers. Note how the port numbers have been allocated and that the IP address/port number pairs are unique across the solution.

When the web server starts it loads the modules and plug-ins that have been registered with it. When the web server loads the Witango Server Client it starts the initialisation process which begins by reading the *clients.ini* file, verifying its content and then configuring the load splitting configuration. In this example the *clients.ini* file would need to be configured to communicate with the follow server combinations.

Machine	IP address	Witango Server	Port number
Web Server, Witango Client and Witango Application Servers	10.10.10.10	1	18100
		2	18101
Witango Application Server 1	10.10.10.11	1	18100
		2	18101
		3	18102
Witango Application Server 2	10.10.10.12	1	18110
		2	18111
		3	18112

It is a good idea to map out the configuration before modifying the *clients.ini* file to avoid errors with IP/Port pairs.

The anatomy of a user reference

To associate a user variable with a particular user, Witango must have a piece of information that it can use to uniquely identify that user. Witango refers to this unique identifier used for tracking a user's variables as the user reference key. When users execute their first Witango application file Witango generates a unique user reference number and uses it as the user key. When results are sent back to the user, Witango includes the user reference number as an HTTP cookie. This cookie is utilised by the Web browser and is sent automatically with every subsequent request to your server. Witango checks for the existence of the cookie whenever it accesses a user variable. If it was sent, the cookie value is used as the user key. Thus if a user opens two windows in a web browser application, both windows share the same cookies and, therefore, the same user variables.

The user reference key number generated by the server is based on the configuration for USERKEY and ALTUSERKEY in the *witango.ini* file. The default setting for USERKEY is <@USERREFERENCE>. The server will produce a user reference key which is either a 24-digit or 36-digit hexadecimal string. If your configuration does not use the load splitting functionality (i.e. you only run one Witango Application Server) your user reference keys will be 24-digits long. When load splitting functionality is utilisied, the user reference key has an additional 12 digits prefixed to the key. These 12 extra digits represent the IP address and port that the server that generated the user reference key number is running on.

e.g. Decoding a user reference keys 7F00000146B4488D0C5B847CA5853794E3 and 54A497684AD2A5853794E38C5940014FDD

7F00000146B4488D0C5B847CA5853794E3		54A497684AD2A5853794E38C5940014FDD	
7F00000146B4	488D0C5B847CA5853794E3	54A497684AD2	A5853794E38C5940014FDD
127.0.0.1:1810	<@USERREFERENCE>	84.164.151.104:19154	<@USERREFERENCE>
IP Addr & Port	USERKEY set ting	IP Addr & Port	USERKEY set ting

How Witango Client Finds the Right User

If no user key is found when the web server receives a request for a taf or tml file it is automatically redirected to the Witango Server Client (wapache.dll, wiis.dll or cgi) located on the same machine as the web server. Upon receiving the request from the web server, the Witango Client examines the http header information that has been received to determine if a user reference key is either a cookie or an argument.

If no user key is found

If no user key is received, the Witango Client randomly chooses a Witango Server from the WITANGO_SERVER list of the *clients.ini* configuration and forwards the request to the chosen server. Again using the example above, the Witango Client upon receiving 3 new requests may forward the requests to the Witango Servers as follows:

Request Number	**Forward to Witango Server at...** **IP address**	**Port number**
1	10.10.10.11	18101
2	10.10.10.12	18111
3	10.10.10.10	18101

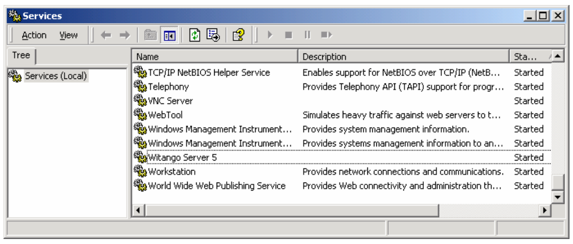
If user key is found

When the Witango Client finds a user reference key in the web server request (as either a cookie or an argument), it strips off the first 12 bytes of the user reference and forwards the request to the Witango Server that corresponds to the address and port encoded in the 12 bytes. This ensures that the user is always sent back to the same server the previously served that user. If a user key is received but server parameters are invalid for connection (it fails to connect), then Witango Client randomly chooses a Witango Server – this would have the effect of creating a new session for the user.

Creating Witango Servers in a Load Group

When you first install Witango Server you have one Witango Server process running on the server and only one server in your load group. With the appropriate license you can add additional Witango Server processes to the server and add them into the load group. All the

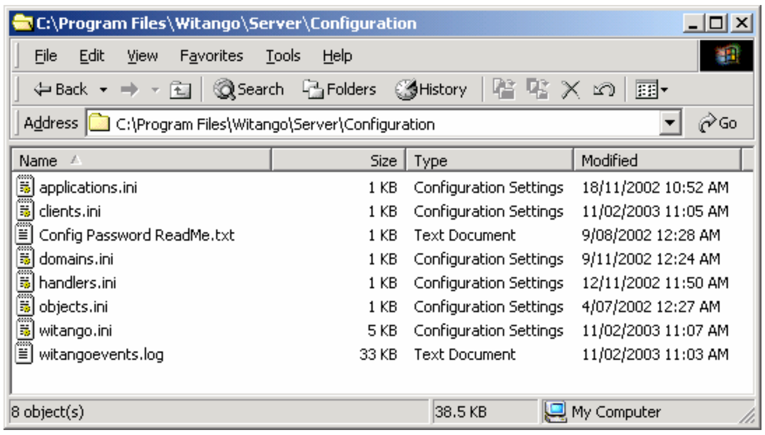
Witango Servers in the load group will then work together with the Witango Client to split the load between the server processes.



Name	Description	Sta...
TCP/IP NetBIOS Helper Service	Enables support for NetBIOS over TCP/IP (NetB...	Started
Telephony	Provides Telephony API (TAPI) support for progr...	Started
VNC Server		Started
WebTool	Simulates heavy traffic against web servers to t...	Started
Windows Management Instrument...	Provides system management information.	Started
Windows Management Instrument...	Provides systems management information to an...	Started
Witango Server 5		Started
Workstation	Provides network connections and communications.	Started
World Wide Web Publishing Service	Provides Web connectivity and administration th...	Started

Adding multiple Witango Servers to a Machine

1 Install an initial copy of the Witango Application Server on the machine (see the Witango Server Installation Guide).

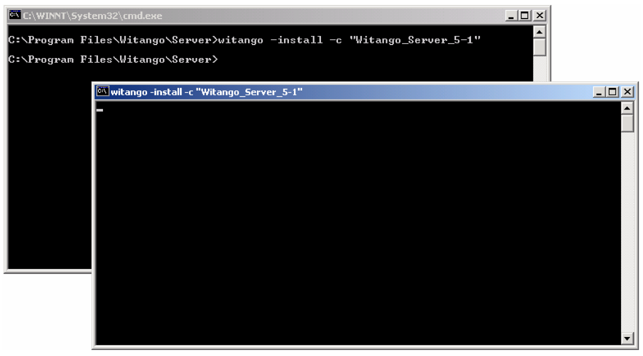

C:\Program Files\Witango\Server\Configuration
File Edit View Favorites Tools Help
Back Search Folders History
Address C:\Program Files\Witango\Server\Configuration
Go

Name	Size	Type	Modified
applications.ini	1 KB	Configuration Settings	18/11/2002 10:52 AM
clients.ini	1 KB	Configuration Settings	11/02/2003 11:05 AM
Config Password ReadMe.txt	1 KB	Text Document	9/08/2002 12:28 AM
domains.ini	1 KB	Configuration Settings	9/11/2002 12:24 AM
handlers.ini	1 KB	Configuration Settings	12/11/2002 11:50 AM
objects.ini	1 KB	Configuration Settings	4/07/2002 12:27 AM
witango.ini	5 KB	Configuration Settings	11/02/2003 11:07 AM
witangoevents.log	33 KB	Text Document	11/02/2003 11:03 AM

8 object(s) 38.5 KB My Computer

2 Install additional server processes by running the following command in a shell/command window. During the execution of this command, a second shell/command window will open momentarily as the service is installed.

```
witango.exe –install –c “<new service name>”
```

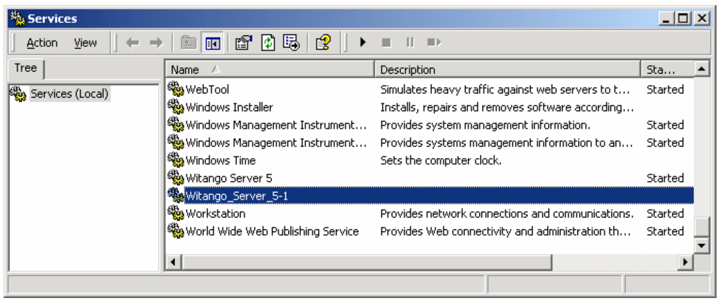

C:\WINNT\System32\cmd.exe
witango -install -c "Witango_Server_5-1"

```
C:\Program Files\Witango\Server>witango -install -c "Witango_Server_5-1"

C:\Program Files\Witango\Server>
```

You will need to have a unique service name for each Witango Server processes that you have running on your server. It is recommended that you have a naming convention for the additional processes that is both predictable and extensible. E.g. Witango Server 5-1, Witango Server 5-2, etc.

After you have run the -install command the new Witango Server processes will appear in the services control panel and will be named <new service name>.

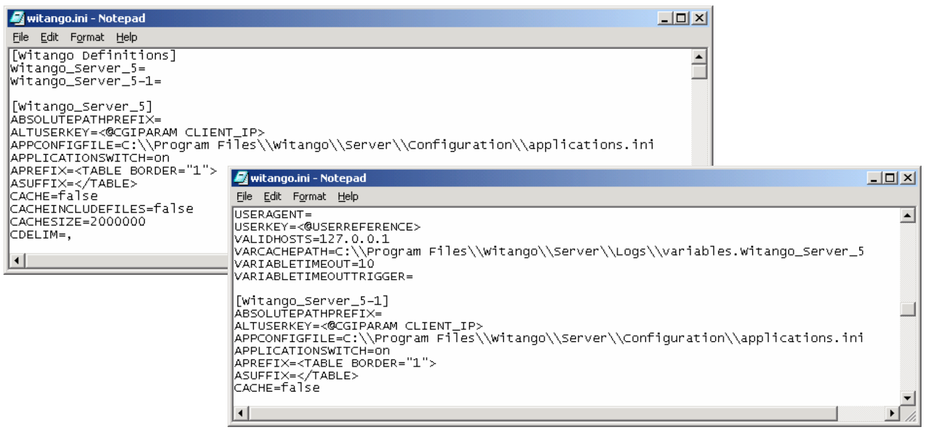


Name	Description	Sta...
WebTool	Simulates heavy traffic against web servers to t...	Started
Windows Installer	Installs, repairs and removes software according...	
Windows Management Instrument...	Provides system management information.	Started
Windows Management Instrument...	Provides systems management information to an...	Started
Windows Time	Sets the computer clock.	
Witango Server 5		Started
Witango_Server_5-1		
Workstation	Provides network connections and communications.	Started
World Wide Web Publishing Service	Provides Web connectivity and administration th...	Started

3 You will then need to create the server configuration stanzas for each Witango service installed. These stanzas are all kept in the *witango.ini* file. There are two ways to do generate the new stanzas:

a. Edit the *witango.ini* file and duplicate one of the existing stanzas then modify the parameters of the duplicate to suit the new server.

b. Open the Services control panel and attempt to start each process you have installed. The new service will not start as the default LISTENERPORT will default to 18100 which will already be in use by the original Witango Server service. Although the service does not start, a stanza with default settings for the service will be added to the *witango.ini* file. The name of the new stanza will be the same as the service name with the spaces replaces with underscore (_) characters. This stanza can then be modified to reflect the parameters of the new Server.



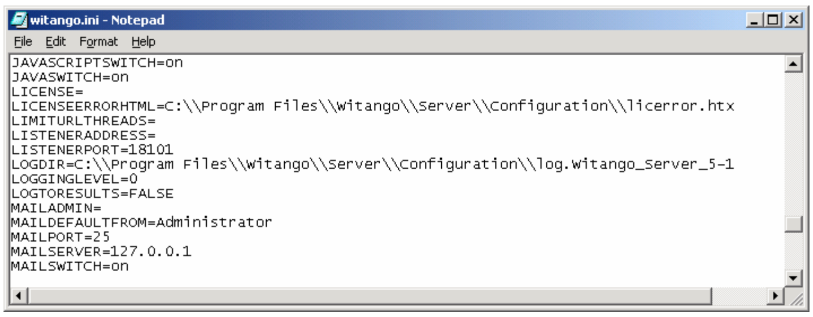
```
[Witango Definitions]
Witango_Server_5=
Witango_Server_5-1=

[Witango_Server_5]
ABSOLUTEPATHPREFIX=
ALTUSERKEY=<@CGIPARAM CLIENT_IP>
APPCONFIGFILE=C:\\Program Files\\Witango\\Server\\Configuration\\applications.ini
APPLICATIONSWITCH=on
APREFIX=<TABLE BORDER="1">
ASUFFIX=</TABLE>
CACHE=false
CACHEINCLUDEFILES=false
CACHESIZE=2000000
CDELIM=,
```

```
USERAGENT=
USERKEY=<@USERREFERENCE>
VALIDHOSTS=127.0.0.1
VARCACHEPATH=C:\\Program Files\\Witango\\Server\\Logs\\variables.Witango_Server_5
VARIABLETIMEOUT=10
VARIABLETIMEOUTTRIGGER=

[Witango_Server_5-1]
ABSOLUTEPATHPREFIX=
ALTUSERKEY=<@CGIPARAM CLIENT_IP>
APPCONFIGFILE=C:\\Program Files\\Witango\\Server\\Configuration\\applications.ini
APPLICATIONSWITCH=on
APREFIX=<TABLE BORDER="1">
ASUFFIX=</TABLE>
CACHE=false
```

4 Edit the LISTENERPORT for each server so that they do not conflict with each other.



```
JAVASCRIPTSWITCH=on
JAVASWITCH=on
LICENSE=
LICENSEERRORHTML=C:\\Program Files\\Witango\\Server\\Configuration\\licerror.htx
LIMITURLTHREADS=
LISTENERADDRESS=
LISTENERPORT=18101
LOGDIR=C:\\Program Files\\Witango\\Server\\Configuration\\log.Witango_Server_5-1
LOGGINGLEVEL=0
LOGTORESULTS=FALSE
MAILADMIN=
MAILDEFAULTFROM=Administrator
MAILPORT=25
MAILSERVER=127.0.0.1
MAILSWITCH=on
```

The simplest way to achieve this is to start with the first stanza and set the LISTENERPORT to 18100 (the default port). Move down the *witango.ini* file incrementing the LISTENERPORT by 1 each time. i.e 18100 then 18101 then 18102, …

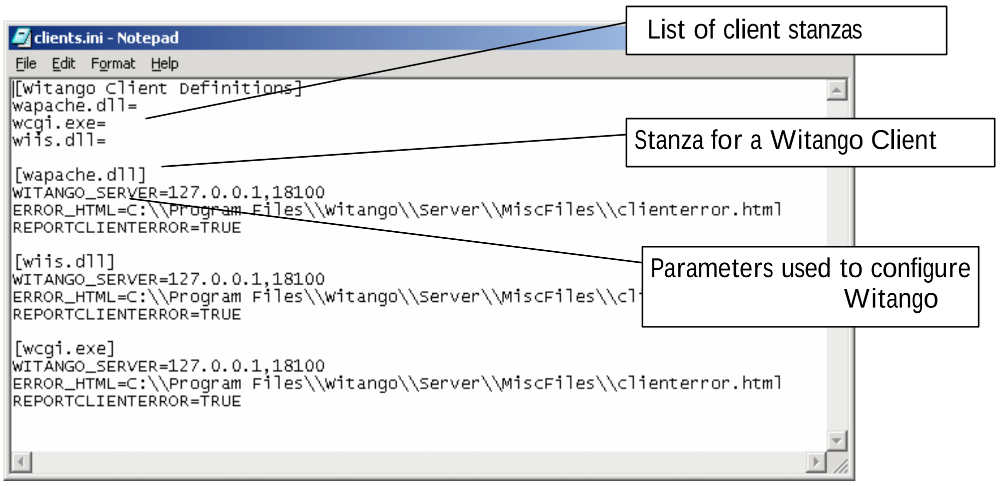
5 Start all the Witango Server processes. If one of the processes fails to start, check that the port number you have chosen does not conflict with any of the other processes.

6 Configure the Witango Client by editing the *clients.ini* file.

NOTE: If you are using DNS load balancing with you web servers you must ensure that the *clients.ini* files are identical on each server.

The *clients.ini* file is structured like other Witango ini files - a header that lists all the different client stanzas that can be initialised on the server and a stanza for each client setting the CONNECTION_TIMEOUT, WITANGO_SERVER, FORCE_SERVER_ARG_NAME and ERROR_HTML.

- **CONNECTION_TIMEOUT**: sets the number of seconds that the client waits before timing out a connection
- **WITANGO_SERVER**: is a comma separated, colon separated list of servers the client will connect to.
- **FORCE_SERVER_ARG_NAME**: sets the name of the search argument that can be passed to the client that will force the taf to be executed on a specified server. The value of the search arg corresponds to an alias that is configured in the WITANGO_SERVER parameter.
- **ERROR_HTML**: sets the path to a file that will be served if an error is encountered by the Witango Client.



```
[witango Client Definitions]
wapache.dll=
wcgi.exe=
wiis.dll=

[wapache.dll]
WITANGO_SERVER=127.0.0.1,18100
ERROR_HTML=C:\\Program Files\\witango\\Server\\MiscFiles\\clienterror.html
REPORTCLIENTERROR=TRUE

[wiis.dll]
WITANGO_SERVER=127.0.0.1,18100
ERROR_HTML=C:\\Program Files\\witango\\Server\\MiscFiles\\cl
REPORTCLIENTERROR=TRUE

[wcgi.exe]
WITANGO_SERVER=127.0.0.1,18100
ERROR_HTML=C:\\Program Files\\witango\\Server\\MiscFiles\\clienterror.html
REPORTCLIENTERROR=TRUE
```

A typical *clients.ini* file configured for a single Witango Server would like this:

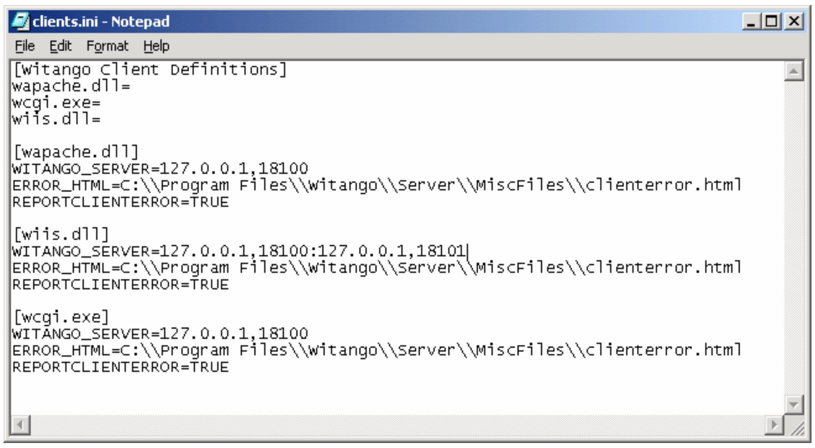
7 Add each of the new servers that the client will communicate with to the WITANGO_SERVER. The format of the entries in this parameter is:

```
{[server alias],server IP address,server listener port]
```

e.g

```
WITANGO_SERVER=127.0.0.1,18100:127.0.0.1,18101:127.0.0.1,18102:127.0.0.1,18103
OR
WITANGO_SERVER=WitangoServer1,127.0.0.1,18100:WitangoServer2,127.0.0.1,18101:
                          WitangoServer3,127.0.0.1,18102:127.0.0.1,18103
```

NOTE: The server alias is optional and is only necessary if you will need the functionality to execute a taf on a particular server.



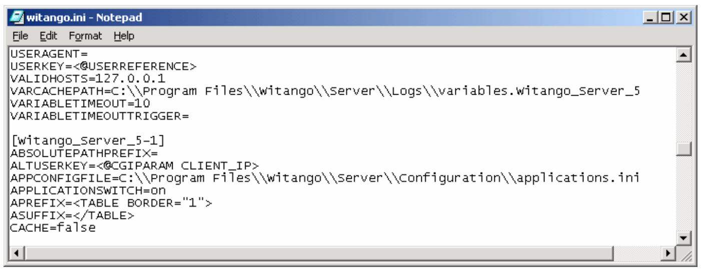
```
[Witango Client Definitions]
wapache.dll=
wcgi.exe=
wiis.dll=

[wapache.dll]
WITANGO_SERVER=127.0.0.1,18100
ERROR_HTML=C:\\Program Files\\Witango\\Server\\MiscFiles\\clienterror.html
REPORTCLIENTERROR=TRUE

[wiis.dll]
WITANGO_SERVER=127.0.0.1,18100:127.0.0.1,18101
ERROR_HTML=C:\\Program Files\\Witango\\Server\\MiscFiles\\clienterror.html
REPORTCLIENTERROR=TRUE

[wcgi.exe]
WITANGO_SERVER=127.0.0.1,18100
ERROR_HTML=C:\\Program Files\\Witango\\Server\\MiscFiles\\clienterror.html
REPORTCLIENTERROR=TRUE
```

8 Check the VALIDHOSTS entry in the *witango.ini* file of each server to ensure that it only contains the IP addresses of the Witango Clients that you want to allow to connect. This setting is a security measure that controls which hosts are allowed to send a request to the Witango Server.



```
USERAGENT=
USERKEY=<@USERREFERENCE>
VALIDHOSTS=127.0.0.1
VARCACHEPATH=C:\\Program Files\\Witango\\Server\\Logs\\variables.Witango_Server_5
VARIABLETIMEOUT=10
VARIABLETIMEOUTTRIGGER=

[Witango_Server_5-1]
ABSOLUTEPATHPREFIX=
ALTUSERKEY=<@CGIPARAM CLIENT_IP>
APPCONFIGFILE=C:\\Program Files\\Witango\\Server\\Configuration\\applications.ini
APPLICATIONSWITCH=on
APREFIX=<TABLE BORDER="1">
ASUFFIX=</TABLE>
CACHE=false
```

Directing Requests to a Specific Witango Server

In a load-splitting environment Witango requests are randomly distributed across Witango Servers in the load-splitting group. To guarantee execution of a certain task on a specific Witango Server, Witango allows you to direct new requests for application file execution to a specific Witango Server within the load-splitting environment. This may be useful when you want to:

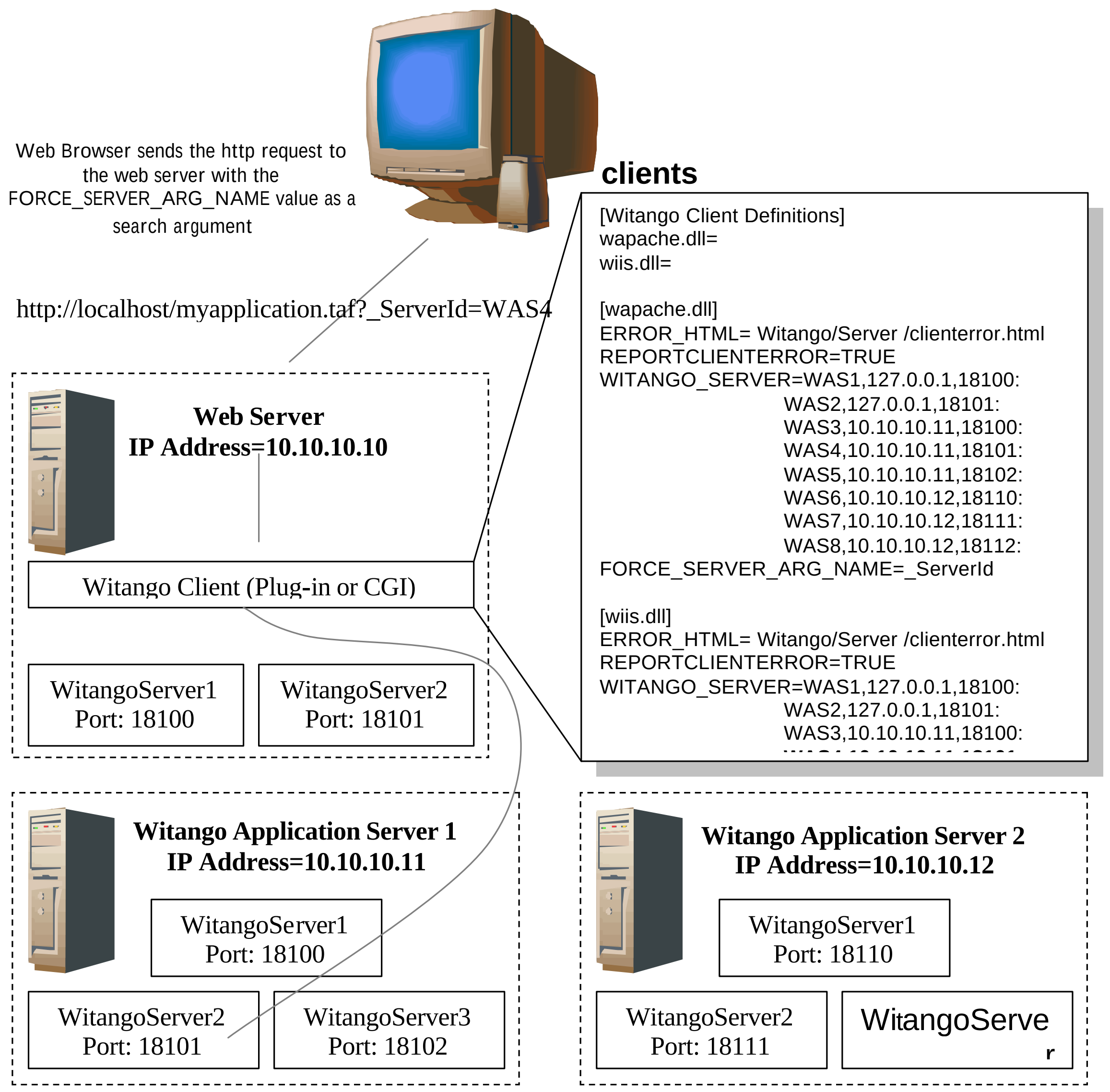
- purge the cache of a particular Witango Server
- modify system-scope variables by running a Witango application file (such as config.taf) for this purpose on a particular Witango Server
- generate status information on a particular Witango Server.
- run routine maintenance

Directing a request to a specific Witango Server is accomplished by executing a Witango application file that passes a Server ID as a search argument called _SeverID by default. The Server ID is defined by parameters in the Witango Client configuration file (*clients.ini*); for example, _ServerId could be defined as the name of the Server ID search argument (FORCE_SERVER_ARG_NAME=_ServerId), and WitangoServer1 could be defined as a server alias and passed as the value of the Server ID search argument.

NOTE: For security purposes, you can set the Server ID search argument name and value to whatever you want in the Witango Client configuration file.

When the Witango Client receives a HTTP request that contains the Server ID search argument from a Web browser, it reads the Witango Client configuration file (*clients.ini*) to determine where to forward the request. It then redirects it to the specified Witango Server.

The following diagram outlines how a HTTP request for a Witango application file running on IIS is directed to a specific Witango Server based on the configuration (*clients.ini*) of the Witango Server Client:



```
[Witango Client Definitions]
wapache.dll=
wiis.dll=

[wapache.dll]
ERROR_HTML= Witango/Server /clienterror.html
REPORTCLIENTERROR=TRUE
WITANGO_SERVER=WAS1,127.0.0.1,18100:
                WAS2,127.0.0.1,18101:
                WAS3,10.10.10.11,18100:
                WAS4,10.10.10.11,18101:
                WAS5,10.10.10.11,18102:
                WAS6,10.10.10.12,18110:
                WAS7,10.10.10.12,18111:
                WAS8,10.10.10.12,18112:
FORCE_SERVER_ARG_NAME=_ServerId

[wiis.dll]
ERROR_HTML= Witango/Server /clienterror.html
REPORTCLIENTERROR=TRUE
WITANGO_SERVER=WAS1,127.0.0.1,18100:
                WAS2,127.0.0.1,18101:
                WAS3,10.10.10.11,18100:
```

When the Witango Client receives a request that contains a search argument that matches the value of the FORCE_SERVER_ARG_NAME, the client then tries to match the value of the search argument with one of the server aliases listed in the WITANGO_SERVER parameter.

Setting the Server ID

The Witango Client configuration file controls configuration of the Witango Client (Witango CGI or Witango plug-in) and defines which Witango Server the Witango Client talks to. To configure Witango Client to direct requests to a specific Server, you must edit the Witango Client configuration file (*clients.ini*). The default location of the Witango Client configuration file is:

```
<PATH TO THE WITANGO SERVER>\Witango\Server\Configuration\
```

Note: It is not necessary to stop or restart the Witango Server before or after editing the Witango Client configuration file to direct requests to a specific Witango Server

To configure the Witango Client for direct requests to a specific Witango Server

1 Open the clients.ini file in a text editor.

2 Locate the stanza for the type of Witango Client you are running:

- If you are using the Witango CGI: [wcgi.exe]
- If you are using the Witango plug-in for Microsoft IIS: [wiis.dll]
- If you are using the Witango plug-in for Apache: [wapache.dll]

3 Add a new line, FORCE_SERVER_ARG_NAME=, below the WITANGO_SERVER line:

e.g.

```
WITANGO_SERVER=127.0.0.1,18100
FORCE_SERVER_ARG_NAME=
```

4 Assign the Server ID search argument name by typing a value in the FORCE_SERVER_ARG_NAME line, as shown:

e.g.

```
WITANGO_SERVER=127.0.0.1,18100
FORCE_SERVER_ARG_NAME=_ServerId
```

Tip: For security purposes, you can set the Server ID search argument name and value to whatever you want.

5 Assign a Server ID value in the WITANGO_SERVER line by typing a value (before the IP address and port number) in the appropriate stanza followed by a comma. Repeat for each Witango Server that you want to direct requests to.

e.g.

```
WITANGO_SERVER=WAS1,127.0.0.1,18100
FORCE_SERVER_ARG_NAME=_ServerId
```

Note: The Server ID value is an alphanumeric string consisting of a maximum of 16 characters. Server ID values should be unique. If the same Server ID is assigned to more than one Witango Server, the Witango Client uses the first IP address and port number to identify that Witango Server.

The following example shows Server ID value WAS1 assigned to a Witango Server with IP address 127.0.0.1 and port number 18100, receiving requests from the wiis.dll.

```
[wiis.dll]
ERROR_HTML=/Witango/Server/clienterror.html
REPORTCLIENTERROR=TRUE
WITANGO_SERVER=WAS1,127.0.0.1,18100:WAS2,127.0.0.1,18101:
               WAS3,10.10.10.11,18100:WAS4,10.10.10.11,18101:WA
               S5,10.10.10.11,18102:
               WAS6,10.10.10.12,18110:WAS7,10.10.10.12,18111:WA
               S8,10.10.10.12,18112
FORCE_SERVER_ARG_NAME=_ServerId
```

WAS1 through WAS8 are the Server ID values assigned to each Witango Application Server that the Witango Server Client can communicate with as part of the Load Group.

6. Repeat steps 1-5 for each Witango Client you will be using.

   NOTE: You do not need to assign a Server ID name or value when adding a Witango Server to the load group however, these Witango Servers are not available for direct access.. Witango Servers without an assigned Server ID continue to randomly process new requests that do not contain the Server ID search argument.

   When the targeted Witango Server fails to accept a direct request, the Witango Client does not redirect the request to another server in the load group. Instead, an error is returned.

Configuring Witango Client Connection Timeout

Witango allows you to configure the period of time before the connection to the Witango Client (Witango CGI or Witango plug-in) times out. Configuring Witango Client connection timeout is especially useful in a load-splitting environment. By default Witango uses the default system settings for connection timeout, which are generally longer.

To configure Witango Client connection timeout, you edit the Witango Client configuration file (*clients.ini*) directly to specify the CONNECT_TIMEOUT parameter.

To configure Witango Client connection timeout

1 Open the *clients.ini* file in a text editor.

2 Add the CONNECT_TIMEOUT parameter to one of the following stanzas:

- If you are using the Witango CGI: [wcgi.exe]
- If you are using the Witango plug-in for Microsoft IIS: [wiis.dll].
- If you are using the Witango plug-in for Apache: [wapache.dll]

3 Specify the number of seconds before connection to Witango Client times out by assigning a number to the CONNECT_TIMEOUT parameter, as shown:

eg:

```
 [wcgi.exe]
      WITANGO_SERVER=127.0.0.1,18100:127.0.0.1,18101:127.0.0.1,18102:
      CONNECT_TIMEOUT=2
```

There are three Witango Servers specified in this example so 6 seconds is the maximum delay before connection to Witango Server fails (3 Witango Servers x 2 second connection timeouts). In contrast, if connection to two Witango Servers is unsuccessful, the maximum connection delay is 4 seconds (2 Witango Servers x 2 second connection timeouts) before contact is successful with the third server.

When CONNECT_TIMEOUT is not specified (default) or when CONNECT_TIMEOUT=0, the Witango Client timeout connection period is based on the system settings for connection timeout.

Configuring Error HTML for Witango Client

Witango allows you to specify a custom error message for the Witango Client (Witango CGI or plug-in). To configure Witango Client Error HTML, you edit the Witango Client configuration file (*clients.ini*) directly to specify an ERROR_HTML parameter in the plug-in or CGI stanza.

To configure Witango Client Error HTML

1 Open the *clients.ini* file in a text editor.

2 Add the ERROR_HTML parameter to one of the following stanzas:

- If you are using the Witango CGI: [wcgi.exe]
- If you are using the Witango plug-in for Microsoft IIS: [wiis.dll].
- If you are using the Witango plug-in for Apache: [wapache.dll]

3 Assign an absolute path to your custom client error file to the ERROR_HTML parameter.

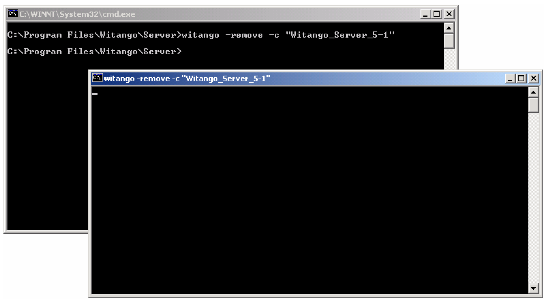
eg: `as shown: ERROR_HTML=C:\Program Files\Witango\Server\clienterror.html`

When this line is found in the stanza and it points to an existing file, this file is displayed instead of the standard Witango Client error message when a Witango Client error occurs.

Removing a Witango Server instance

You can remove server processes by running the following command in a shell/command window. During the execution of this command, a second shell/command window will open momentarily as the service is removed.

```
witango.exe –install –c “<service name>”
```


C:\WINNT\System32\cmd.exe
witango -remove -c "Witango_Server_5-1"

```
C:\Program Files\Witango\Server>witango -remove -c "Witango_Server_5-1"

C:\Program Files\Witango\Server>
```

The witangoevents.log

With multiple servers running on the same machine, you will see multiple entries in the *witango.ini* file as each server process starts. Server processes may be started simultaneously by the operating system so entries into the log file may be mixed together. To assist in identifying the lines that relate to a particular process, the process ID is added to the beginning of each line.

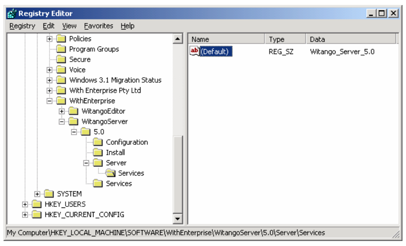
eg.

```
[ 1152] 2003-02-11 11:09:56 START      WARNING  Server Watcher executable file is missing
[ 1152] 2003-02-11 11:09:56 START      INFO     JavaScript-C 1.4 release 1 1998 10 31
[ 1152] 2003-02-11 11:09:56 RUNTIME    WARNING  Caching of application files has been disabled
[ 1152] 2003-02-11 11:09:56 RUNTIME    WARNING  Caching of include files has been disabled
[ 1152] 2003-02-11 11:09:56 START      WARNING  Cannot initialize Oracle Call Interface (OCI) environment
[ 1152] 2003-02-11 11:09:56 START      INFO     Witango daemon started with configuration stanza 'Witango_Server_5'
[ 1152] 2003-02-11 11:09:56 VERSION    INFO     Version 5.0.1.054 Chimera (Win32)
[ 1152] 2003-02-11 11:09:56 START      INFO     Allocated 20 worker threads
[ 1152] 2003-02-11 11:10:01 RUNTIME    INFO     Started accepting user requests
[  884] 2003-02-11 11:10:05 START      WARNING  Server Watcher executable file is missing
[  884] 2003-02-11 11:10:05 START      INFO     JavaScript-C 1.4 release 1 1998 10 31
[  884] 2003-02-11 11:10:05 RUNTIME    WARNING  Caching of application files has been disabled
[  884] 2003-02-11 11:10:05 RUNTIME    WARNING  Caching of include files has been disabled
[  884] 2003-02-11 11:10:05 START      WARNING  Cannot initialize Oracle Call Interface (OCI) environment
[  884] 2003-02-11 11:10:05 START      INFO     Witango daemon started with configuration stanza 'Witango_Server_5-1'
[  884] 2003-02-11 11:10:05 VERSION    INFO     Version 5.0.1.054 Chimera (Win32)
[  884] 2003-02-11 11:10:05 LICENSE    INFO     Time-limited license expires in 30 days
[  884] 2003-02-11 11:10:05 START      INFO     Allocated 20 worker threads
[  884] 2003-02-11 11:10:06 RUNTIME    INFO     Skipped empty startup URL
[  884] 2003-02-11 11:10:07 RUNTIME    INFO     Started accepting user requests
```

Troubleshooting the Install of a New Service

There is an issue with the way that the initial Witango Server 5 service is created by the installer. A registry item may not have been automatically created. If this is the case, the –install option will fail to create a new service. To fix this problem, ensure that the default value for the following registry item is set to Witango_Server_5.0

```
HKEY_LOCAL_MACHINE\SOFTWARE\WithEnterprise\WitangoServer\5.0\Server\Services
```



Name	Type	Data
(Default)	REG_SZ	Witango_Server_5.0

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WithEnterprise\WitangoServer\5.0\Server\Services