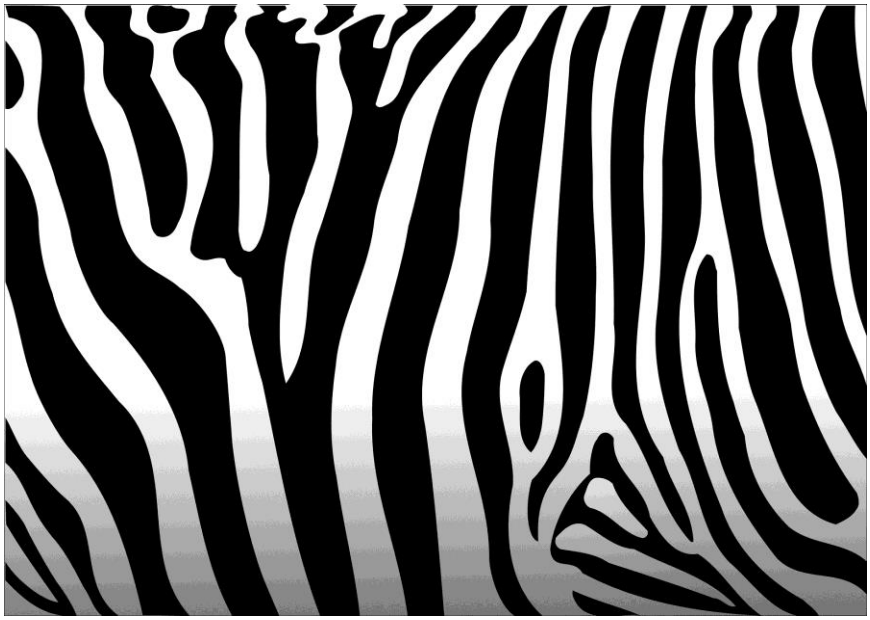
Ingenieurbüro David Fischer AG | A Company of the Apica Group
http://www.apicasystem.com

ZebraTester

Application Reference Manual

Version 5.4



 March 8, 2016

Table of Contents

1 Introduction

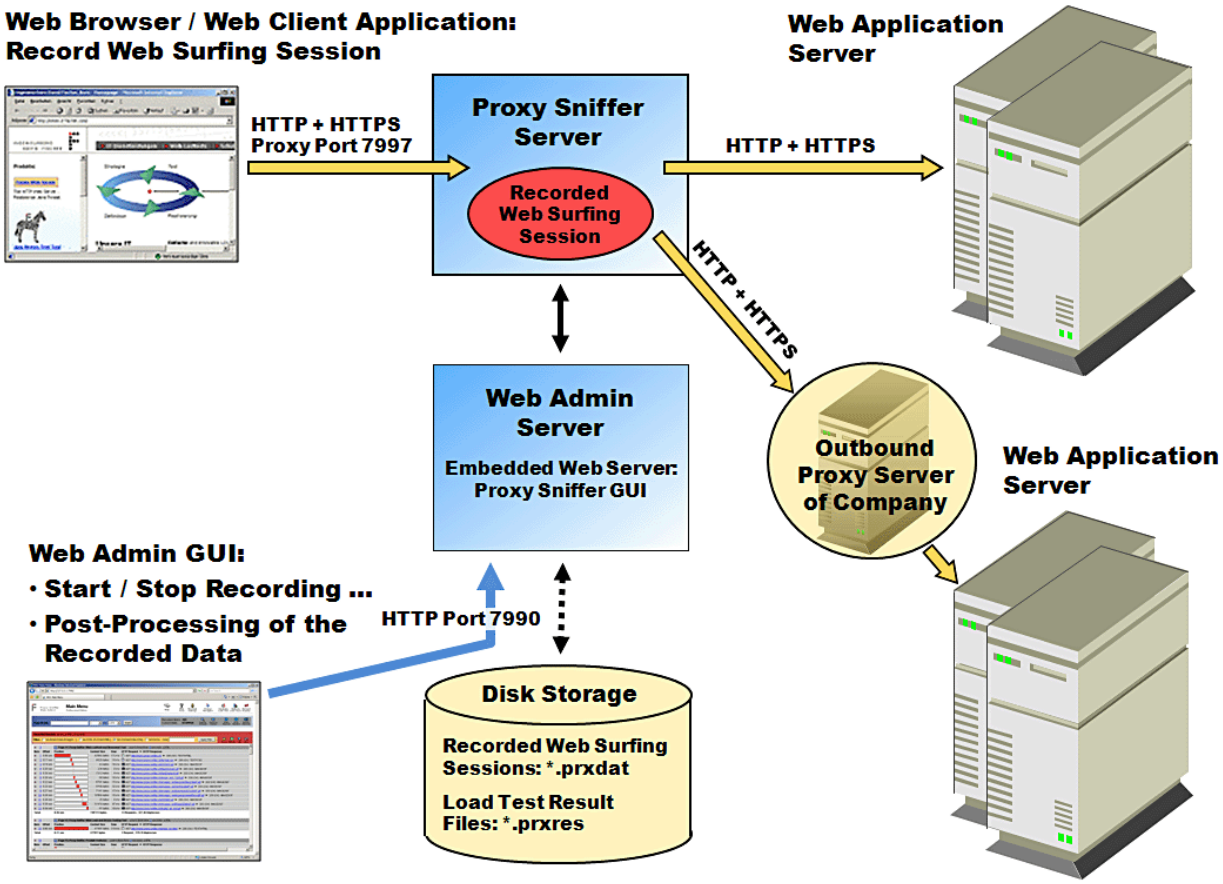
This document contains a short description of the ZebraTester architecture (chapter 2), followed by detailed descriptions of all programs delivered with the ZebraTester product (chapter 3). Chapter 3.6 also contains information about how to install ZebraTester on Unix-like systems.

In addition, this document contains information about scripting load test programs (chapter 4).

At the end of the document, you will find a list of all supported time zones (chapter 5), as well as a list of the ZebraTester configuration files (chapter 6).

2 The ZebraTester Architecture

Image 1: Recording of Web Surfing Sessions



The product contains four integrated server types:

1. **The Proxy Sniffer Server**. Acts as special proxy server and is able to record entire Web surfing sessions, which are stored inside its transient memory. This special proxy server decrypts HTTPS connections automatically, on-the-fly, by using built-in SSL tunnels. Cascading the Proxy Sniffer Server with an (already existing) outbound proxy server is supported. In case NTLM authentication or HTTPS client certificates are required for (target) Web server authentication, these authentication elements can be loaded into the Proxy Sniffer Server, enabling the Proxy Sniffer Server to perform authentication against the (target) Web server – instead of the Web browser. Recording a Web surfing session over several target Web servers is also supported.

2. **The Web Admin Server**. GUI. This is a thin HTTP Web server which retrieves its static contents directly from prxsniff.jar. It is used to control the Proxy Sniffer Server, and to manage its recorded data by offering a graphical user interface, or GUI. The Web Admin Server displays the (transient) data of a recorded Web surfing session on its GUI, and supports storing/loading Web surfing sessions to/from disk **(*.prxdat files**). In addition, it supports the generation and compilation of load test programs (*.java), the starting and controlling of load test jobs, and the reading and display of load test result files **(*.prxres files**).

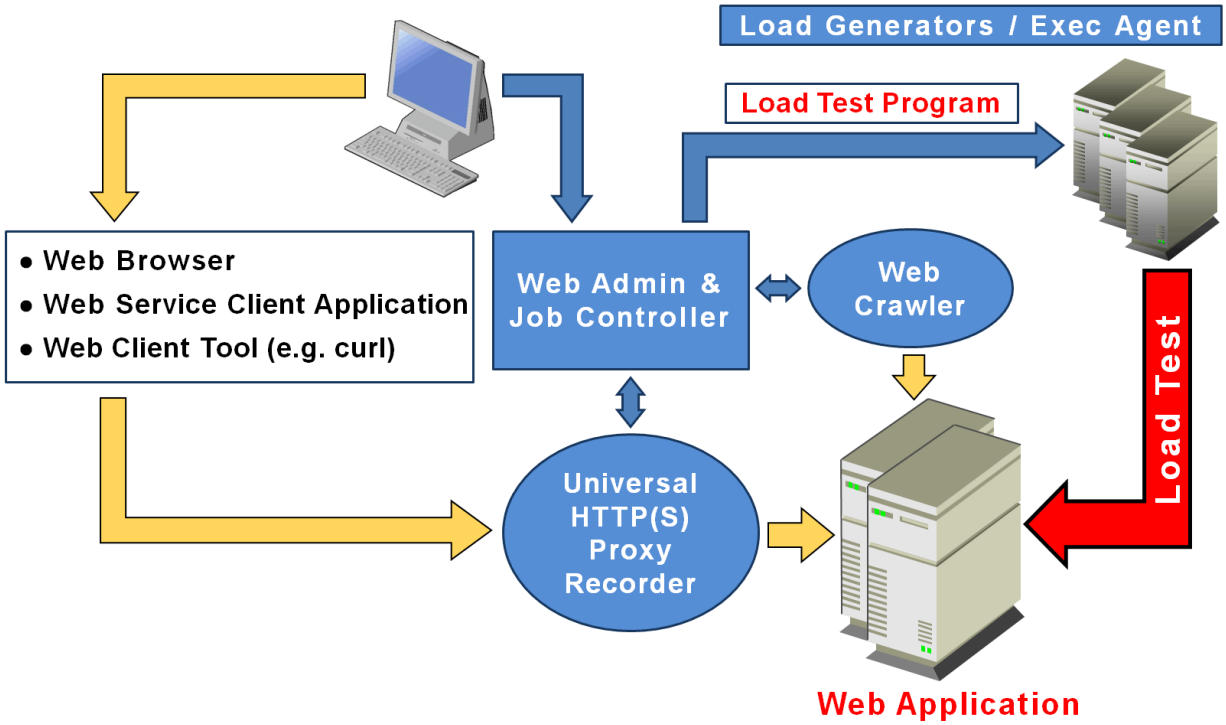
3. **The Exec Agent Server**. Executes load test programs as “Jobs”, and works closely with the Web Admin Server. One Exec Agent Server is able to run several load test jobs at the same time in parallel by using an own virtual Java CPU for each load test job. The communication between the Web Admin Server and the Exec Agent Server usually runs over plain TCP/IP connections (port 7993). Alternatively, this communication can be performed via HTTP or HTTPS tunneling over outbound proxy servers - a feature which allows to communicate with external Exec Agent Servers which are located in (external) data centers.

4. **The Cluster Job Controller**. Executes load test programs as “Cluster Jobs”. The Job Controller is able to distribute (and split) a single load test program over an unlimited number of Exec Agent Servers, and is able to merge the load test results of all of the Exec Agents into a single unified result. The Cluster Job Controller works closely with the Web Admin Server and the Exec Agent Server(s), and must be started on the host running the Web Admin Server. Several Cluster Jobs can run at the same time by using the same, or a different collection of Exec Agent Servers.

All these four types of servers can be started individually on Windows as well as on Unix-like systems.

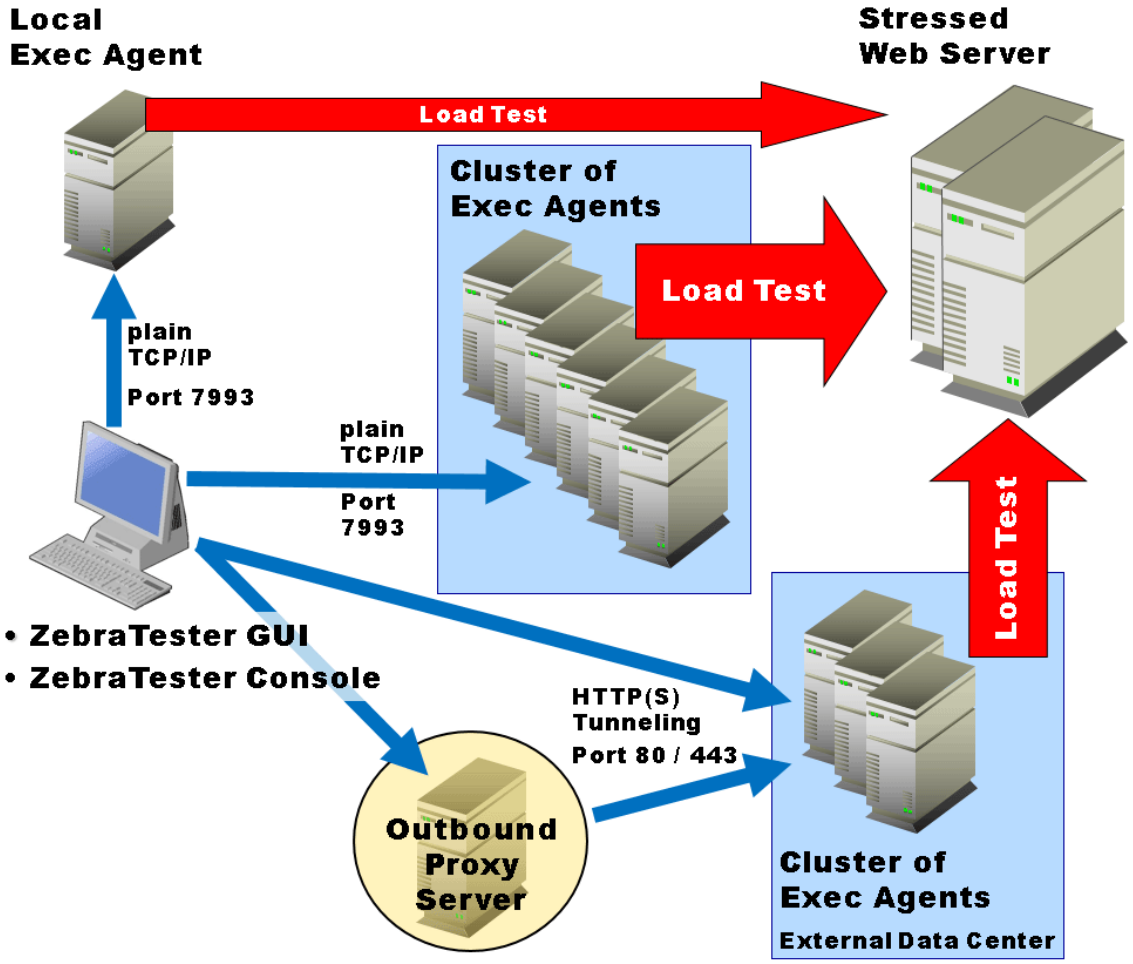
Note: on Windows, Mac OS X and Ubuntu systems, all four servers can be locally started all at once by clicking on the installed desktop-icon “**ZebraTester Console**”. The “ZebraTester Console” can also be started on other Unix-like systems (Solaris, Linux, BSD …) – as described in chapter 3.5 .

Image 2: Architecture Overview



To run heavy load tests with **hundreds or thousands of virtual users,** we recommend that you combine several Exec Agent Servers into a virtual **Exec Agent Cluster**. The ZebraTester product is able to handle “distributed” load tests completely transparent, in the same manner as load tests are started from a single host.

Image 3: Load Test Execution



3 Application Reference Manual

This chapter describes all command-line options of the built-in servers and utilities, as well as all command-line options associated with the automatically-generated load test programs.

All product-related functionality is packed inside a single file: the Java archive **prxsniff.jar**. The only prerequisites for starting a program or utility are that the Java SDK 1.7 (Java 7) must be installed on the system, and that the files **prxsniff.jar**, **iaik_jce_full.jar**, **iaik_ssl.jar**, **iaik_eccelerate.jar**, **iaikPkcs11Provider.jar** and the path to the "ZebraTester installation directory" must be contained in the Java **CLASSPATH**. Note that on Windows systems, the Java SDK is included as part of the ZebraTester installation kit, and does not need to be installed separately. On Mac OS X systems, the Java SDK is always preinstalled.

ZebraTester uses its own encryption library (IAIK) which contains additional classes; therefore **all ZebraTester programs must be started** such that the libraries **prxsniff.jar**, **iaik_jce_full.jar**, **iaik_ssl.jar**, **iaik_eccelerate.jar** and **iaikPkcs11Provider.jar** are found inside the CLASSPATH.

The GUI of ZebraTester (the Proxy Recorder component) requires in addition that the two files **root.cer** and **privkey.der** exists in the ZebraTester installation directory. These two files are already delivered by the installation kits, but it is strongly recommended that you replace them by your own versions (see ZebraTester "Installation and Configuration Guide").

Special Notes for Mac OS X Servers:

On Mac OS X Servers which do not have an X11 display (no graphics card installed) you must additionally use the Java option `-Djava.awt.headless=true` when starting ZebraTester or when starting an Exec Agent.

3.1 Proxy Sniffer Server

Invocation:

```
java ProxySniffer [-tz <timezone>]
                  [-dgs a | c]
                  [-WebAdmin]
                  [-ExecAgent]
                  [-JobController]
                  [-runtimedatadir <path>]
                  [-jobdir <directory>]
                  [-clusterjobdir <directory>]
                  [-enableRemoteOsCommands]
                  [-guitimeout <seconds>]
                  [-dh]
                  [-dcontent]
                  [-ssl <version>]
                  [-ecc]
                  [-ssltunneltimeout <seconds>]
                  [-execAgentConfigDir <path>]
                  [-execAgentClusterConfigDir <path>]
                  [-outboundipfile <file-name>]¨
                  [-dnshosts <file-name>]
                  [-dnstranslation <file-name>]
                  [-dnssrv <IP-name-server-1>[,<IP-name-server-N>]]
                  [-dnsfixttl <seconds>]
                  [-dnsstatistic]
                  [-redirectLog <log-file>]
                  [-debugDNS]
                  [-debugRecorderSocketPools]
                  [-debugRecorderPlugins]
                  [-periodicTimestamp <seconds>]
                  [-RESTAPIServer [<IP-port>]]
                  [-log4j]
                  [-h] [-help]
```

Created TCP/IP server ports:

Port Number	Description
7990	**Web Admin** HTTP port. This port will only be created if the option -WebAdmin has been set.
7993	**Exec Agent** port. This port will only be created if the option -ExecAgent has been set.
7995	(Cluster) **Job Controller** port. This port will only be created if the option -JobController has been set.
7996	**Proxy Recorder REST API** port. This port will only be created if the option - RESTAPIServer has been set.
7997	**HTTPS and HTTP proxy client port**, used by the recording Web browser
7998	Data port, used for internal communication with the Web Admin server
~~7999~~	**Deprecated since Version 5.0: use port 7997 for HTTP and HTTPS.** Old HTTP proxy client port.
15000…15000 + n	Dynamic port range of SSL tunnels. Used only for internal communication on the localhost. Every browser request for an additional target HTTPS Web server will cause the Proxy Sniffer Server to create a new SSL tunnel with a faked server certificate. If the recorded surf-session calls only one HTTPS server, only one tunnel will be created, and only one port will be used.

Example:

```
SET CLASSPATH=.;prxsniff.jar;
iaik_jce_full.jar;iaik_ssl.jar;iaik_eccelerate.jar;iaikPkcs11Provider.jar
java -Xmx1024m ProxySniffer -WebAdmin -JobController -ExecAgent
-tz PNT -dgs c

or on Unix-like systems:

export CLASSPATH=.:prxsniff.jar:
iaik_jce_full.jar:iaik_ssl.jar:iaik_eccelerate.jar:iaikPkcs11Provider.jar
nohup java -Xmx1024m ProxySniffer -WebAdmin -JobController -ExecAgent
-tz PNT -dgs c 2>&1 > ProxySniffer.log &
```

List of all supported options:

[-tz <timezone>]
Sets a time zone for the current date and time of the server. The default time zone is ECT. See also chapter 6: Supported Time Zones. Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6).

[-dgs a | c]
Sets the number format. a = apply apostrophe as decimal grouping separator (example: 123'456.00). c = apply comma as decimal grouping separator (example: 123,456.00). The default value is "a". Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6).

[-WebAdmin]
Starts the Proxy Sniffer Server together with the Web Admin HTTP server – inside the same program. Please note that the Web Admin server can also started as a stand-alone program with the command "java WebAdmin".

[-ExecAgent]
Starts the Proxy Sniffer Server together with a local Exec Agent server – inside the same program. Please note that the Exec Agent server can also started as a stand-alone program with the command "java ExecAgent" – on a Windows system as well as on a Unix system.

[-JobController]
Starts the Proxy Sniffer Server together with the (Cluster) Job Controller server – inside the same program. Please note that the Job Controller server can also started as a stand-alone program with the command "java JobController".

[-runtimedatadir <path>]
Set the directory in which the runtime configuration data of Web Admin and ZebraTester are written. The default directory is the working directory of ZebraTester. Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6).

[-jobdir <directory>]
Sets the working directory of the Exec Agent. The default temporary directory of the operating system will be used as working directory if this option is not set (subdirectory: 'PrxExecAgentJobs'). Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6).

[-clusterjobdir <directory>]
Sets the working directory of the Job Controller. The default temporary directory of the operating system will be used as working directory if this option is not set (subdirectory: 'PrxClusterJobs'). Alternatively, this option can also be configured on Windows and Mac OS X

systems in the **prxsniff.dat** file (see chapter 6). **Note**: you cannot use the same working directory for the Exec Agent and for the Job Controller – you have to use two different (sub-) directories.

[-enableRemoteOsCommands]
Allows to execute operating system commands which are remotely released by using the PrxJob utility (see chapter 0). Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6). Note: this option **should not be enabled** in insecure operating environments.

[-guitimeout <seconds>]
Sets the request timeout in seconds of the Web Admin GUI. The default value is 420 seconds. Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6).

[-dh]
Debug HTTP headers: the proxy server displays the exchanged HTTP header (-fields) on standard output.

[-dcontent]
Debug HTTP content: the proxy server displays the exchanged HTTP content on standard output.

[-ssl <version>]
Allows the setting of a fixed SSL protocol version for the recording of encrypted Web surfing sessions. Possible values are “all” (automatic detection of v3/tls/tls11/tls12 – recommended default value), “v3”, “tls”, “tls11”, or “tls12” (fixed protocol version).

[-ecc]
Enable support of Elliptic Curve Cryptography (ECC) for the proxy recorder.

[-ssltunneltimeout <seconds>]
The maximum time in seconds an SSL tunnel will wait for a response from the HTTPS Web server. The default is 60 seconds. This option is not normally needed.

[-execAgentConfigDir <path>]
Set the disk path (w/o file name) at which the file **execAgent.dat** is located

[-execAgentClusterConfigDir <path>]
Set the disk path (w/o file name) at which the file **execAgentCluster.dat** is located

[-outboundipfile <file-name>]
Use multiple outbound TCP/IP addresses that are read from a configuration file to communicate with the Exec Agents, and perform TCP/IP load balancing for outbound TCP/IP connections that are established from the WebAdmin and the JobController component of ZebraTester to the Exec Agents. The configuration file can contain multiple IP addresses on the same line or on several lines. Multiple IP addresses on the same line must be separated by space characters, or tab characters, or by commas, or by semicolons. The hash character can be used as a marker for comments within any position in a line.

[-dnshosts <file-name>]
Effects that the Proxy Recorder uses an own DNS hosts file to resolve host names - rather than using the hosts file of the underlying operating system. Note that the DNS settings can also be dynamically configured at runtime by using the REST API.

[-dnstranslation <file-name>]
Effects that the Proxy Recorder uses a DNS translation file which is a text file that contains on each line a translation between two DNS names. If the first DNS name in the file match to the DNS name that is passed to the resolver then the second DNS name is used to resolve the IP

address.
The first DNS name can also contain one or more wildcard characters ('*' = wildcard for multiple characters, '?' = wildcard for single character). Lines or a part of a line can be commented out by using the hash char '#'.
Example of a DNS translation file:

```
www.proxy*sniffer.com www.proxy-sniffer.com # comment
# comment
www.mutong.com www.d-fischer.com
mail?.google.com mail.google.com
```

Note 1: It could be needed that TLS SNI (Server Name Indication) must be disabled if a DNS translation table is used.
Note 2: The HTTP request header field "Host" is not updated, so it might happen that you call the Web server with the wrong host name.

[-dnssrv <IP-name-server-1>[,<IP-name-server-N>]]
Effects that the Proxy Recorder uses specific (own) DNS server(s) to resolve host names - rather than using the DNS library of the underlying operating system. Multiple DNS servers can be configured separated by commas. Note that the DNS settings can also be dynamically configured at runtime by using the REST API.

[-dnsfixttl <seconds>]
Configure the Proxy Recorder to apply a fixed TTL value of seconds for all DNS resolves – instead of using the TTL values received DNS from resolves. Note that the DNS settings can also be dynamically configured at runtime by using the REST API.

[-dnsstatistic]
Effects that statistical data about DNS resolves are measured when recording web surfing sessions, by using an own DNS stack.
Note: there is no need to use this option if any other, more specific DNS option is enabled because the (other) DNS options **-dnshosts, -dnssrv** and **-dnsfixttl** also effect implicitly that statistical data about DNS resolutions are measured.

[-redirectLog <log-file>]
Redirects stdout and stderr to a log file. The output will be appended if the file already exists.

[-debugDNS]
Debug DNS resolves and the DNS cache of the proxy recorder.

[-debugRecorderSocketPools]
Debug the socket pools of the proxy recorder.

[-debugRecorderPlugins]
Debug recorder plug-ins (do not confuse – “recorder plug-ins” are not “load test plug-ins”).

[-periodicTimestamp <seconds>]
Direct the Job Controller to write a time stamp in periodical intervals of seconds to stdout. Note that this option is considered only if also the argument **-JobController** was set.

[-RESTAPIServer [<TCP/IP port>]]
Start additionally the REST API server of the Proxy Recorder. The default REST API server port is 7996.

[-log4j]
Enable log4j (see chapter 3.3.1).

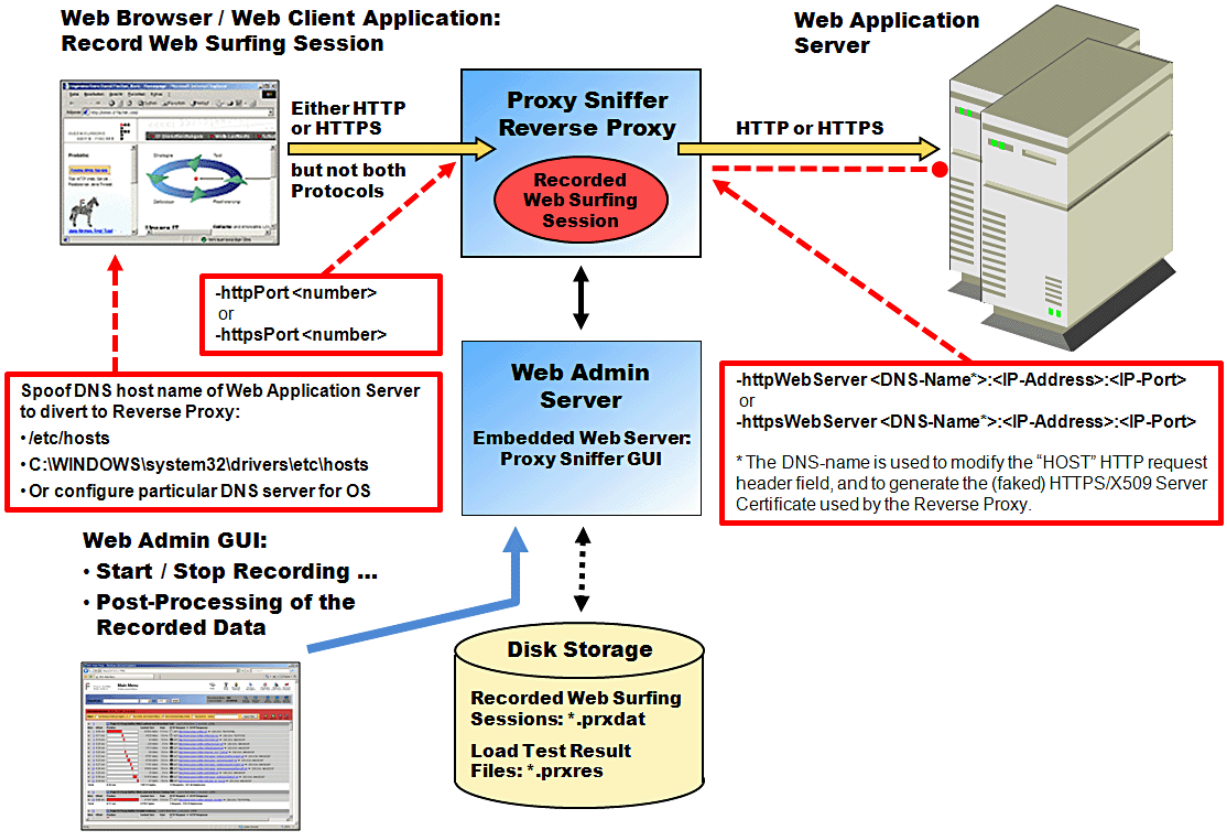
[-h] or [-help]
Displays a short help text about all Proxy Sniffer server options.

3.1.1 Starting Proxy Sniffer as a Reverse Proxy / Special Option

Normally the recording Proxy Server program is started as an outbound HTTP(S) proxy, as described before in this manual. Unlike to that, as a special option, it's also supported to operate the recording Proxy Server program as a "reverse proxy" – meaning as an upstream server in front of a Web application server.

However, when using this special option, you can record HTTP or HTTPS traffic for one (target) Web server only, and you have to spoof the DNS host name of the target Web server on that machine where the Web client program (i.e. the Web browser used for recording) is running - pointing the spoofed DNS name to the "reverse proxy".

Proxy Sniffer Reverse Proxy Architecture (Special Option)



Additional Program Arguments for starting a HTTP Reverse Proxy (unencrypted traffic)

`-httpPort <IP-port-number>`
Set the HTTP IP port number of the reverse proxy. For example: 80.

`-httpWebServer <DNS-name>:<IP-address>:<IP-port-number>`
Set the DNS name of the Web application server that is used by the Web client program (Web browser), and set the IP address and the IP port number of the Web application server

Example / HTTP Reverse Proxy:

```
SET CLASSPATH=.;prxsniff.jar;
iaik_jce_full.jar;iaik_ssl.jar;iaik_eccelerate.jar;iaikPkcs11Provider.jar
java -Xmx1024m ProxySniffer –httpPort 80
-httpWebServer www.myserver.com:192.16.4.44:80
-WebAdmin -JobController -ExecAgent -tz PNT -dgs c

or on Unix-like systems:

export CLASSPATH=.:prxsniff.jar:
iaik_jce_full.jar:iaik_ssl.jar:iaik_eccelerate.jar:iaikPkcs11Provider.jar
nohup java -Xmx1024m ProxySniffer –httpPort 80
-httpWebServer www.myserver.com:192.16.4.44:80 -WebAdmin
-JobController –ExecAgent -tz PNT -dgs c 2>&1 > ProxySniffer.log &
```

Additional Program Arguments for stating a HTTPS Reverse Proxy (encrypted traffic)

-httpsPort <IP-port-number>
Set the HTTPS IP port number of the reverse proxy. For example: 443.

-httpsWebServer <DNS-name>:<IP-address>:<IP-port-number>
Set the DNS name of the Web application server that is used by the Web client program (Web browser), and set the IP address and the IP port number of the Web application server

Example / HTTPS reverse proxy:

```
SET CLASSPATH=.;prxsniff.jar;
iaik_jce_full.jar;iaik_ssl.jar;iaik_eccelerate.jar;iaikPkcs11Provider.jar
java -Xmx1024m ProxySniffer –httpsPort 443
-httpsWebServer www.myserver.com:192.16.4.44:443
-WebAdmin -JobController -ExecAgent -tz PNT -dgs c

or on Unix-like systems:

export CLASSPATH=.:prxsniff.jar:
iaik_jce_full.jar:iaik_ssl.jar:iaik_eccelerate.jar:iaikPkcs11Provider.jar
nohup java -Xmx1024m ProxySniffer –httpsPort 443
-httpsWebServer www.myserver.com:192.16.4.44:443 -WebAdmin
-JobController –ExecAgent -tz PNT -dgs c 2>&1 > ProxySniffer.log &
```

Note for HTTPS: ZebraTester generates automatically on the fly a faked X509 server certificate used to decrypt the SSL traffic. This certificate contains that DNS name which you have passed by the program arguments **-httpsWebServer <DNS-name>:<IP-address>:<IP-port-number>**. The server certificate is derived from a root certificated named **root.cer** that is located in the ZebraTester installation directory. You have to import the root certificate into **root.cer** your Web browser (or machine), or into the trust store of your technical Web client program in order that you can establish an encrypted network connection to the reverse proxy.

3.1.2 Proxy Recorder REST API

Since version 5.2-D, an integrated REST API server for the Proxy Recorder is available if Proxy Sniffer is started as an independent process or as a Windows Service/Unix Daemon and if the startup argument **`-RESTAPIServer`** is set.

The REST API server runs normally on TCP/IP port 7996 and uses HTTP as communication protocol. API input parameters are passed as CGI values. The API output is always in JSON data format.

The following API functions are available:

- Get the product version from Proxy Recorder process
- Get the DNS/IP blacklist configuration
- Set the DNS/IP blacklist configuration
- Get the DNS/IP whitelist configuration
- Set the DNS/IP whitelist configuration
- Get the URL blacklist configuration
- Set the URL blacklist configuration
- Get the URL whitelist configuration
- Set the URL whitelist configuration
- Get the SSL/TLS version
- Set the SSL/TLS version
- Get the DNS configuration
- Set the DNS configuration
- Get the DNS translation map
- Set the DNS translation map
- Clear the DNS resolve cache
- Get outline information about all “recorder plug-ins”
- Change the state of a “recorder plug-in” to active or to inactive
- Manually call a “recorder plug-in”
- Trigger a re-scan for “recorder plug-ins”
- Get the current recorder state (recoding started or stopped)
- Trigger start of recording
- Trigger stop of recording
- Clear all recorded data
- Insert (append) a “page break”
- Set the “user’s think time” for all “page breaks”
- Enable/disable parallel execution of URL calls when running the session as load test
- Get the total number of recoded “page breaks” and URL calls.
- Get a short dump about the recorded session
- Get the session filter
- Set the session filter
- Get the recoded session in ProxySniffer data format (*.prxdat file content)
- Get the recorded session in HAR data format (HTTP Archive version 1.2)
- Load a recorded session (ProxySniffer data format) into the Proxy Recorder

- Load a recorded session (HAR data format) into the Proxy Recorder

The corresponding API function is selected by the CGI Parameter **cmd**. The API server responses with a 200 (ok) HTTP status code if the command was successful performed. If an unknown command was sent, or a mandatory CGI parameter was missing, then the API server responses with a 400 (bad request) HTTP status code. If parsing of uploaded data fails then a 422 (Unprocessable Entity) HTTP status code is returned.

Example:

API URL call: **GET** `http://127.0.0.1:7996/?cmd=getRecorderVersion`

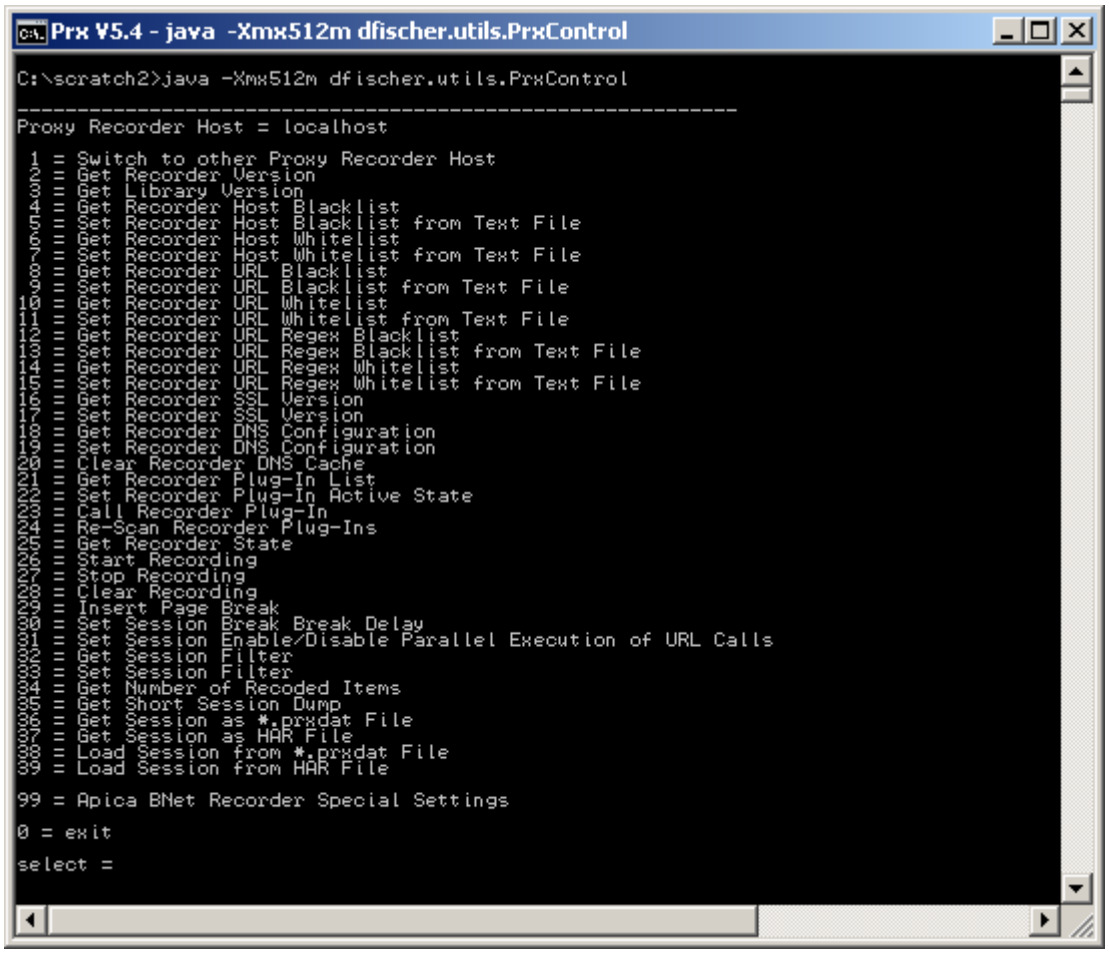
JSON response: `{"RecorderVersion":"5.2-D"}`

The access to the REST API can optionally be protected by Basic Access Authentication. If you want to enable that you have to create a file with the name **prxControlBasicAuth.dat** in the ProxySniffer installation directory (respectively in the working directory of the Proxy Recorder process). Inside this text-file each line can contain an arbitrary <username>:<password> string in Base 64 format. Example: “miller:secret” is equal to “bWlsbGVyOnNlY3JldA==” in Base 64 format. Lines that are starting with a hash sign ‘#’ are seen as comments.

Note 1: If you wish to call the REST API from a Terminal Window, or from a Shell Script you can use **curl**, see http://curl.haxx.se/.

Note 2: If you wish to call these functions directly from an own written Java program there is no need to use the REST API, and there is also no need to start the REST API server. Instead you can used the methods in the class
`dfischer.utils.PrxControl`
which provide the same functionality. Further information is available in the **ZebraTester Java API Documentation** which is delivered by all of the ZebraTester installation kits.

Note 3: There is also a ready to use Terminal Application available that allows you to try out the API w/o programming. Invoke: **`java -Xmx512m dfischer.utils.PrxControl`**



3.1.2.1 getRecorderVersion

Description: Get the product version of the proxy recorder (meaning the version that the running proxy recorder process is currently using).

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object
  - Key: “RecorderVersion”
  - Value: Product version (String), w/o ‘V’ at start of string.

Example:

API URL call: **GET** `http://127.0.0.1:7996/?cmd=getRecorderVersion`

JSON response: `{"RecorderVersion":"5.2-D"}`

3.1.2.2 getRecorderBlacklist

Description: Get a list of DNS names and IP addresses for which recording is suppressed by the proxy recorder. This means that URL requests for such list elements are not forwarded to the web server and also not recorded. Instead of this the web client (web browser) receive a faked response directly from the proxy recorder. A DNS name or an IP address can also contain multiple times the wildcard char ‘*’.

The response sent to the web client depends on the “blacklist action” which can contain one of the following values per DNS name or per IP address:

- **abort** : Abort the network connection to the web client w/o sending any response.
- **webbugimage** : Send a transparent image of 1x1 pixel to the web client.
- **<number / HTTP status code>** : Send a HTTP response containing the configured status code to the web client (w/o any content data).

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object
  - Key: “RecorderBlacklist”
  - Array
    - Object
      - Key: DNS name or IP address (can also contain wildcard chars ‘*’)
      - Value: blacklist action

Example:

API URL call: **GET** `http://127.0.0.1:7996/?cmd=getRecorderBlacklist`

JSON response:

```
{"RecorderBlacklist":[{"*google*":"webbugimage"},{"*.nothingforme.ch"
:"abort"},{"www.tracking.com":"404"}]}
```

or

`{"RecorderBlacklist":[]}` (no blacklist configured)

3.1.2.3 setRecorderBlacklist

Description: Set a list of DNS names and IP addresses for which recording is suppressed by the proxy recorder. This means that URL requests for such list elements are not forwarded to the web server and also not recorded. Instead of this the web client (web browser) receive a faked response directly from the proxy recorder. A DNS name or an IP address can also contain multiple times the wildcard char '*'.

The response sent to the web client depends on the configured "blacklist action" which can contain one of the following values per DNS name or per IP address:

- **abort** : Abort the network connection to the web client w/o sending any response.
- **webbugimage** : Send a transparent image of 1x1 pixel to the web client.
- **<number / HTTP status code>** : Send a HTTP response containing the configured status code to the web client (w/o any content data).

CGI Input Parameter	Description of Parameter
format	Optional, String in capital letters. The value can be either "JSON" or "TEXT", depending on the data format of the request content. Default = "JSON"

POST Request Content:
Either JSON or TEXT data.
Example of JSON data:

```
{"RecorderBlacklist":[{"*google*":"webbugimage"},{"*.nothingforme.ch":
"abort"},{"www.tracking.com":"404"}]}
```

Example of TEXT data (lines separated by <CR><LF> or <CR> only or <LF> only):

```
# my blacklist
*google* webbugimage
www.tracking.com 404
*.notthis.ch abort
```

Lines which are starting with a hash char '#' are commented out.

POST Request Content-Type:
femto/binary (mandatory)

JSON Response:

- Object (empty)

Example:

API URL call: **curl --request POST -H "Content-Type:femto/binary" --data-binary "@C:\scratch2\recorderBlacklist.json" "http://127.0.0.1:7996/?cmd=setRecorderBlacklist&format=JSON"**

JSON response: **{}**

Note 1: To disable the blacklist you can post the following JSON data `{"RecorderBlacklist":[]}` or TEXT data that contain at least one space character.

Note 2: Any modifications are only temporary applied and are revoked when the Proxy Recorder process is restarted. For a permanent configuration you have to create a text file named **recorderBlacklist.dat** in the installation directory.

3.1.2.4 getRecorderWhitelist

Description: Get a list of DNS names and IP addresses that are exceptions of the recorder blacklist. A DNS name or an IP address can also contain multiple times the wildcard char '*'.

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object
  - Key: "RecorderWhitelist"
  - Array
    - Value: DNS name or IP address (can also contain wildcard chars '*')

Example:

API URL call: **GET `http://127.0.0.1:7996/?cmd=getRecorderWhitelist`**

JSON response: **`{"RecorderWhitelist":["*ae*","www.x.com"]}`**
or
`{"RecorderWhitelist":[]}` (no whitelist configured)

3.1.2.5 setRecorderWhitelist

Description: Set a list of DNS names and IP addresses that are exceptions of the recorder blacklist. A DNS name or an IP address can also contain multiple times the wildcard char '*'.

CGI Input Parameter	Description of Parameter
format	Optional, String in capital letters. The value can be either "JSON" or "TEXT", depending on the data format of the request content. Default = "JSON"

POST Request Content:
Either JSON or TEXT data.
Example of JSON data:

```
{"RecorderWhitelist":["*ae*","www.x.com"]}
```

Example of TEXT data (lines separated by <CR><LF> or <CR> only or <LF> only):

```
# my whitelist
*ae*
www.x.com
```

Lines which are starting with a hash char '#' are commented out.

POST Request Content-Type:
`femto/binary` (mandatory)

JSON Response:

- Object (empty)

Example:

API URL call: **`curl --request POST -H "Content-Type:femto/binary" --data-binary "@C:\scratch2\recorderWhitelist.json" "http://127.0.0.1:7996/?cmd=setRecorderWhitelist&format=JSON"`**

JSON response: **{}**

Note 1: To disable the whitelist you can post the following JSON data `{" RecorderWhitelist ":[]}` or TEXT data that contain at least one space character.

Note 2: Any modifications are only temporary applied and are revoked when the Proxy Recorder process is restarted. For a permanent configuration you have to create a text file named **recorderWhitelist.dat** in the installation directory.

3.1.2.6 getRecorderURLBlacklist

Description: Get a list of URLs for which recording is suppressed by the proxy recorder. This means that URL requests for such list elements are not forwarded to the web server and also not recorded. Instead of this the web client (web browser) receive a faked response directly from the proxy recorder. An URL can also contain multiple times the wildcard char '*'.

The response sent to the web client depends on the "blacklist action" which can contain one of the following values per defined URL:

- **abort** : Abort the network connection to the web client w/o sending any response.
- **webbugimage** : Send a transparent image of 1x1 pixel to the web client.
- **<number / HTTP status code>** : Send a HTTP response containing the configured status code to the web client (w/o any content data).

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object
  - Key: "RecorderURLBlacklist"
  - Array
    - Object
      - Key: URL (can also contain wildcard chars '*')
      - Value: blacklist action

Example:

API URL call: **GET `http://127.0.0.1:7996/?cmd=getRecorderURLBlacklist`**

JSON response:

```
{"RecorderURLBlacklist":[{"*.google.*":"webbugimage"},{"*www.tracking
.com/images/*":"404"},{"http://*.nothingforme.ch/*":"abort"}]}
```

or

`{"RecorderURLBlacklist":[]}` (no blacklist configured)

3.1.2.7 setRecorderURLBlacklist

Description: Set a list of URLs for which recording is suppressed by the proxy recorder. This means that URL requests for such list elements are not forwarded to the web server and also not recorded. Instead of this the web client (web browser) receive a faked response directly from the proxy recorder. An URL can also contain multiple times the wildcard char '*'.

The response sent to the web client depends on the configured "blacklist action" which can contain one of the following values per defined URL:

- **abort** : Abort the network connection to the web client w/o sending any response.
- **webbugimage** : Send a transparent image of 1x1 pixel to the web client.
- **<number / HTTP status code>** : Send a HTTP response containing the configured status code to the web client (w/o any content data).

CGI Input Parameter	Description of Parameter
format	Optional, String in capital letters. The value can be either "JSON" or "TEXT", depending on the data format of the request content. Default = "JSON"

POST Request Content:
Either JSON or TEXT data.
Example of JSON data:

```
{"RecorderURLBlacklist":[{"*.google.*":"webbugimage"},{"http://*.nothi
ngforme.ch/*":"abort"},{"*www.tracking.com/images/*":"404"}]}
```

Example of TEXT data (lines separated by <CR><LF> or <CR> only or <LF> only):

```
# my blacklist
*.google.* webbugimage
http://*.nothingforme.ch/* abort
*www.tracking.com/images/* 404
```

Lines which are starting with a hash char '#' are commented out.

POST Request Content-Type:
femto/binary (mandatory)

JSON Response:

- Object (empty)

Example:

API URL call: **curl --request POST -H "Content-Type:femto/binary" --data-binary "@C:\scratch2\recorderURLBlacklist.json" "http://127.0.0.1:7996/?cmd=setRecorderURLBlacklist&format=JSON"**

JSON response: **{}**

Note 1: To disable the blacklist you can post the following JSON data `{"RecorderURLBlacklist":[]}` or TEXT data that contain at least one space character.

Note 2: Any modifications are only temporary applied and are revoked when the Proxy Recorder process is restarted. For a permanent configuration you have to create a text file named **recorderURLBlacklist.dat** in the installation directory.

3.1.2.8 getRecorderURLWhitelist

Description: Get a list of URLs that are exceptions of the recorder URL blacklist. An URL can also contain multiple times the wildcard char '*'.

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object
  - Key: “RecorderURLWhitelist”
  - Array
    - Value: URL (can also contain wildcard chars ‘*’)

Example:

API URL call: **GET** **`http://127.0.0.1:7996/?cmd=getRecorderURLWhitelist`**

JSON response: **`{"RecorderURLWhitelist":["*www.x.com/*","https://*ae*"]}`**
or
`{"RecorderURLWhitelist":[]}` (no whitelist configured)

3.1.2.9 setRecorderURLWhitelist

Description: Set a list of URLs that are exceptions of the recorder URL blacklist. An URL can also contain multiple times the wildcard char ‘*’.

CGI Input Parameter	Description of Parameter
format	Optional, String in capital letters. The value can be either “JSON” or “TEXT”, depending on the data format of the request content. Default = “JSON”

POST Request Content:
Either JSON or TEXT data.
Example of JSON data:

```
{"RecorderURLWhitelist":["https://*ae*","*www.x.com/*"]}
```

Example of TEXT data (lines separated by <CR><LF> or <CR> only or <LF> only):

```
# my whitelist
https://*ae*
*www.x.com/*
```

Lines which are starting with a hash char ‘#’ are commented out.

POST Request Content-Type:
`femto/binary` (mandatory)

JSON Response:

- Object (empty)

Example:

API URL call: **`curl --request POST -H "Content-Type:femto/binary" --data-binary "@C:\scratch2\recorderURLWhitelist.json" "http://127.0.0.1:7996/?cmd=setRecorderURLWhitelist&format=JSON"`**

JSON response: **`{}`**

Note 1: To disable the whitelist you can post the following JSON data `{"RecorderURLWhitelist":[]}` or TEXT data that contain at least one space character.

Note 2: Any modifications are only temporary applied and are revoked when the Proxy Recorder process is restarted. For a permanent configuration you have to create a text file named **recorderURLWhitelist.dat** in the installation directory.

3.1.2.10 getRecorderURLRegexBlacklist

Description: Get a list of regex matching to URLs for which recording is suppressed by the proxy recorder. This means that URL requests for such list elements are not forwarded to the web server and also not recorded. Instead of this the web client (web browser) receive a faked response directly from the proxy recorder.

The response sent to the web client depends on the "blacklist action" which can contain one of the following values per defined regex:

- **abort** : Abort the network connection to the web client w/o sending any response.
- **webbugimage** : Send a transparent image of 1x1 pixel to the web client.
- **<number / HTTP status code>** : Send a HTTP response containing the configured status code to the web client (w/o any content data).

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object
  - Key: "RecorderURLRegexBlacklist"
  - Array
    - Object
      - Key: Regex matching to URLs
      - Value: Blacklist action

Example:

API URL call: **GET** `http://127.0.0.1:7996/?cmd=getRecorderURLRegexBlacklist`

JSON response:

```
{"RecorderURLRegexBlacklist":[{".*google.*":"webbugimage"},{".*://(www\\.)?zebratester.com.*":"404"},{".*://(www\\.)?pmonframe.org.*":"webbugimage"}]}
```

or

`{"RecorderURLRegexBlacklist":[]}` (no blacklist configured)

3.1.2.11 setRecorderURLRegexBlacklist

Description: Set a list of regex matching to URLs for which recording is suppressed by the proxy recorder. This means that URL requests for such list elements are not forwarded to the web server and also not recorded. Instead of this the web client (web browser) receive a faked response directly from the proxy recorder.

The response sent to the web client depends on the configured "blacklist action" which can contain one of the following values per defined regex:

- **abort** : Abort the network connection to the web client w/o sending any response.
- **webbugimage** : Send a transparent image of 1x1 pixel to the web client.
- **<number / HTTP status code>** : Send a HTTP response containing the configured status code to the web client (w/o any content data).

CGI Input Parameter	Description of Parameter
format	Optional, String in capital letters. The value can be either "JSON" or "TEXT", depending on the data format of the request content. Default = "JSON"

POST Request Content:
Either JSON or TEXT data.
Example of JSON data:

```
{"RecorderURLRegexBlacklist":[{".*google.*":"webbugimage"},{".*://(www
\\.)?zebratester.com.*":"404"}]}
```

Example of TEXT data (lines separated by <CR><LF> or <CR> only or <LF> only):

```
# my regex blacklist
.*google.* webbugimage
.*://(www\\.)?zebratester.com.* 404
```

Lines which are starting with a hash char '#' are commented out.

Note that some chars of the regex like '\' should be escaped as show above (for JSON and for TEXT data).

POST Request Content-Type:
femto/binary (mandatory)

JSON Response:

- Object (empty)

Example:

API URL call: **curl --request POST -H "Content-Type:femto/binary" --data-binary "@C:\scratch2\recorderURLRegexBlacklist.json" "http://127.0.0.1:7996/?cmd=setRecorderURLRegexBlacklist&format=JSON"**

JSON response: **{}**

Note 1: To disable the blacklist you can post the following JSON data `{" recorderURLRegexBlacklist ":[]}` or TEXT data that contain at least one space character.

Note 2: Any modifications are only temporary applied and are revoked when the Proxy Recorder process is restarted. For a permanent configuration you have to create a text file named **recorderURLRegexBlacklist.dat** in the installation directory.

3.1.2.12 getRecorderURLRegexWhitelist

Description: Get a list of regex matching to URLs that are exceptions of the recorder URL regex blacklist.

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object
  - Key: "RecorderURLRegexWhitelist"
  - Array
    - Value: Regex matching to URLs

Example:

API URL call: **GET http://127.0.0.1:7996/?cmd=getRecorderURLRegexWhitelist**

JSON response:
{"RecorderURLRegexWhitelist":[".*www.x.com.*","https://.*ae.*"]}
or
{"RecorderURLRegexWhitelist":[]} (no whitelist configured)

3.1.2.13 setRecorderURLRegexWhitelist

Description: Set a list of regex matching to URLs that are exceptions of the recorder URL regex blacklist.

CGI Input Parameter	Description of Parameter
format	Optional, String in capital letters. The value can be either "JSON" or "TEXT", depending on the data format of the request content. Default = "JSON"

POST Request Content:
Either JSON or TEXT data.
Example of JSON data:

```
{"RecorderURLRegexWhitelist":[".*www.x.com.*","https://.*ae.*"]}
```

Example of TEXT data (lines separated by <CR><LF> or <CR> only or <LF> only):

```
# my regex whitelist
.*www.x.com.*
https://.*ae.*
```

Lines which are starting with a hash char '#' are commented out.

Note that some chars of the regex like '\' should be escaped.

POST Request Content-Type:
femto/binary (mandatory)

JSON Response:

- Object (empty)

Example:

API URL call: **curl --request POST -H "Content-Type:femto/binary" --data-binary "@C:\scratch2\recorderURLRegexWhitelist.json" "http://127.0.0.1:7996/?cmd=setRecorderURLRegexWhitelist&format=JSON"**

JSON response: **{}**

Note 1: To disable the whitelist you can post the following JSON data `{"RecorderURLRegexWhitelist":[]}` or TEXT data that contain at least one space character.

Note 2: Any modifications are only temporary applied and are revoked when the Proxy Recorder process is restarted. For a permanent configuration you have to create a text file named **recorderURLRegexWhitelist.dat** in the installation directory.

3.1.2.14 getRecorderSSLVersion

Description: Get the configured SSL/TLS version of the proxy recorder.

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object
  - Key: "SSLVersion"
  - Value: The SSL/TLS version ("all", "tls12", "tls11", "tls" or "v3")

Example:

API URL call: **GET** `http://127.0.0.1:7996/?cmd=getRecorderSSLVersion`

JSON response: `{"SSLVersion":"tls11"}`

3.1.2.15 setRecorderSSLVersion

Description: Set the SSL/TLS version of the proxy recorder. Note that this setting is permanently stored and reapplied when the product is restarted.

CGI Input Parameter	Description of Parameter
version	The SSL/TLS version ("all", "tls12", "tls11", "tls" or "v3")

JSON Response:

- Object (empty)

Example:

API URL call:
GET `http://127.0.0.1:7996/?cmd=setRecorderSSLVersion&version=all`

JSON response: `{}`

3.1.2.16 getRecorderDNSConfig

Description: Get the special DNS configuration of the proxy recorder.

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object
  - Key: “DNSServers”
  - Array
    - Value: IP address of DNS server
  - Key: “Hosts”
  - Array
    - Object
      - Key: IP Address
      - Value: local configured DNS name (instead of using DNS resolves)
  - Key: “EnableTTL” Value: Boolean
  - Key: “FixedTTL” Value: Integer, the fixed TTL applied for all DNS resolves in seconds, or -1 if no fixed TTL is configured (= use DNS TTL)

Example:

API URL call: **GET** `http://127.0.0.1:7996/?cmd=getRecorderDNSConfig`

JSON response:

```
{"DNSServers":["194.246.118.118","212.25.28.55"],"Hosts":[{"3.3.3.7":"aber.com"},{"127.0.0.1":"localhost"},{"1.8.8.9":"mama"}],"EnableTTL":true,"FixedTTL":-1}
```

or

`{}` (no special DNS configuration / use OS default)

3.1.2.17 setRecorderDNSConfig

Description: Set a special DNS configuration for the proxy recorder.

CGI Input Parameter	Description of Parameter
[none]	

POST Request Content:
JSON data. Keys:

- **DNSServers**: an array of IP addresses of DNS servers.
- **Hosts**: an array of objects which contain as key an IP address and as value a local configured DNS name.
- **EnableTTL**: (boolean) if set to true considering of TTL is enabled for DNS resolves (as received from the DNS server(s)). If set to false resolved IP addresses are cached forever.
- **FixedTTL**: (integer) if the value is not equal to -1 (minus one) then a fixed number of seconds is used as TTL for all DNS resolves, instead of using the received TTL value from the DNS server(s).

Note that all Keys are optional. If one or multiple keys are not sent then default values are applied by the recorder itself. If an empty JSON object is sent then the special DNS configuration is cleared.

Example of JSON data:

```
{"DNSServers":["1.2.3.4","10.2.28.5"],"Hosts":[{"3.3.3.7":"aber.co
m"},{"127.0.0.1":"localhost"},{"1.8.8.9":"mama"}],"EnableTTL":true
,"FixedTTL":-1}
```

Example of JSON data for clearing the special DNS configuration:

```
{}
```

POST Request Content-Type:
femto/binary (mandatory)

JSON Response:

- Object (empty)

Example:

API URL call: **curl --request POST -H "Content-Type:femto/binary" --data-binary "@C:\scratch2\DNSconfig.json" http://127.0.0.1:7996/?cmd=setRecorderDNSConfig**

JSON response: **{}**

Note: Any modifications are only temporary applied and are revoked when the ProxySniffer process is restarted. For a permanent configuration you have either to edit or create a file named **prxsniff.dat** in the installation directory which contains optional startup settings (all optional settings must be placed on one, single line) or to start the ProxySniffer process by passing directly the optional parameters. Example of **prxsniff.dat**:
`-dnssrv 194.246.118.118,212.25.28.55 -dnshosts dnshosts.txt -tz PST`

3.1.2.18 clearRecorderDNSResolveCache

Description: Clear the DNS resolve cache of the special DNS configuration of the recorder.

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object (empty)

Example:

API URL call: **GET** `http://127.0.0.1:7996/?cmd=clearRecorderDNSResolveCache`

JSON response: `{}`

3.1.2.19 getRecorderDNSTranslationTable

Description: Get the DNS translation table of the proxy recorder. This information is only available if the proxy recorder contains also a special DNS configuration

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object
  - Key: “DNSTranslationTable”
  - Array
    - Object
      - Key: DNS name - match key for resolver (String)
      - Value: DNS name - used to resolve the IP address (String)

Example:

API URL call: **GET** `http://127.0.0.1:7996/?cmd=getRecorderDNSTranslationTable`

JSON response:

`{"DNSTranslationTable":[{"www.aber.com":"www.agoda.com"},{"mail?.google.com":"mail.google.com"},{"www.proxy*sniffer.com":"www.d-fischer.com"},{"www.mutong.com":"www.d-fischer.com"}]}`

or

`{}` (no DNS translation table configured)

3.1.2.20 setRecorderDNSTranslationTable

Description: Set or clear the DNS translation table of the proxy recorder. You can only set a DNS translation table if the proxy recorder contains already a special DNS configuration.

A DNS translation table is a hash map that contains as **keys** DNS names which are translated to the **value** = a second DNS name: that is used to resolve the IP address. The keys can also contain one or more wildcard characters ('*' = wildcard for multiple characters, '?' = wildcard for single character).

Note 1: It could be needed that TLS SNI (Server Name Indication) must be disabled if a DNS translation table is used.

Note 2: The HTTP request header field "Host" is not updated, so it might happen that you call the Web server with the wrong host name.

CGI Input Parameter	Description of Parameter
[none]	

POST Request Content:
JSON data:

- **DNSTranslationTable**: an array objects that contain for each object a DNS key and a DNS value.

Example of JSON data:

```
{"DNSTranslationTable":[{"www.aber.com":"www.agoda.com"},{"mail?.google.com":"mail.google.com"}]}
```

Example of JSON data for clearing the DNS translation table:

```
{}
```

POST Request Content-Type:
femto/binary (mandatory)

JSON Response:

- Object (empty)

Example:

API URL call: **curl --request POST -H "Content-Type:femto/binary" --data-binary "@C:\scratch2\DNSTranslationTable.json" http://127.0.0.1:7996/?cmd=setRecorderDNSTranslationTable**

JSON response: **{}**

3.1.2.21 recorderPluginsGetList

Description: Get outline information about all loaded proxy “recorder plug-ins”.

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object
  - Key “**RecorderPlugins**”
  - Array
    - Object
      - Key: “**className**”
      - Value: the Java class name of the recorder plug-in (String)
      - Key: “**classMD5**”
      - Value: A MD5 checksum about the Java class of the recorder plug-in (String)
      - Key: “**isActive**”
      - Value: true if the state of the recorder plug-in is currently active (Boolean)
      - Key: “**isDefaultActive**”
      - Value: true if the default state of the recorder plug-in is active (Boolean)
      - Key: “**executionPosition**”
      - Value: the execution position of the recorder plug-in (Integer)
      - Key: “**allowManualCall**”
      - Value: true if the plug-in allows to be called manually (Boolean)
      - Key: “**allowManualCallOnly**”
      - Value: true if the plug-in can be called manually only (Boolean)
      - Key: **"manualCallArgumentDescriptors"**
      - Array of Objects:
        - Key: "**label**"
        - Value: the label (-text) of the argument
        - Key: "**defaultValue**"
        - Value: the default value of the argument
      - Key: “**description**”
      - Value: a short description about the recorder plug-in in plain ASCII format (String)
      - Key: “**extendedHTMLDescription**”
      - Value: an extended description about the recorder plug-in in HTML formatted text (String)

Example:

API URL call: **GET** **`http://127.0.0.1:7996/?cmd=recorderPluginsGetList`**

JSON response:

{"RecorderPlugins":[{"className":"RecorderPluginDefineUserAccountInputFile","classMD5":"90c65fbdc63e8365d4d25346de945607","isActive":false,"isDefaultActive":false,"executionPosition":1,"allowManualCall":true,"allowManualCallOnly":true,"manualCallArgumentDescriptors":[{"label":"File Name","defaultValue":"UserAccounts.txt"}],"description":"Create Input File Definition for User Accounts","extendedHTMLDescription":"Create an <B>Input File<\/B> definition for <B>User Accounts<\/B> that contains on each line an username and a password, separated by a ';' character. Use the '#' character to comment out lines.<P>This plug-in creates also two variables (<B>vUsername<\/B> and <B>vPassword<\/B>) and the corresponding var extractors. The scope of the file is <B>new line per user<\/B> and the load test is <B>aborted when the end of the file is reached<\/B>.<P>Please note that the name of the input file must match the following rules:<UL style=\"margin-top:0px\"><LI style=\"padding-top:0px\">The name of the input file can contain only the following characters: '0'..'9', 'A'..'Z', 'a'..'z', '_' and '.'<\/LI>\n<LI style=\"padding-top:5px\">The name of the input file must contain a point character ('.'), which is not at the first and not at the last position<\/LI>\n<LI style=\"padding-top:5px\">The first character of the input file must not be a digit ('0'..'9' is not allowed for the first character)<\/LI>\n<\/UL>The recommended file name extension is '.txt'"},{"className":"RecorderPluginDisableContentTestForImagesCSSJS","classMD5":"c59e7436e06fdbf4b4f97f72643edc3c","isActive":false,"isDefaultActive":false,"executionPosition":8,"allowManualCall":true,"allowManualCallOnly":false,"manualCallArgumentDescriptors":[],"description":"Disable Content Test for Images, CCS and JavaScript","extendedHTMLDescription":"Modify the recorded session in such a way that Content Tests for Images, CCS Files and JavaScript Files are disabled.<P>Note that the HTTP response codes and the Content-Types are still verified."},{"className":"RecorderPluginTagRequestResponseHeaderField","classMD5":"e1e4e2c27ca21db6ea96177fb20edf7c","isActive":false,"isDefaultActive":false,"executionPosition":100,"allowManualCall":false,"allowManualCallOnly":false,"manualCallArgumentDescriptors":[],"description":"Tag HTTP Requests and Responses with Random Number","extendedHTMLDescription":"This plug-in adds during recording the HTTP request header field <B>RandomNumberTag<\/B> to each HTTP request and to each HTTP response. Per single pair of request/response the same random number is used.<P>Note that the additional header field is also sent to the Web server."}]}

3.1.2.22 recorderPluginSetActive

Description: Change the state of a loaded “recorder plug-in” to active or to inactive and notify the plugin about the change the state. Note: the plug-in is not notified if its state does not change.
In case if the plug-in allows manual calls only the it remains always in inactive state and trying to set it in active state will result with a 422 "Unprocessable Entity" HTTP status error code.

CGI Input Parameter	Description of Parameter
className	the Java class name of the “recorder plug-in”
isActive	“true” = set the plug-in to active, “false” = set the plug-in to inactive

JSON Response:

- Object (empty)

Example:

API URL call:
GET `http://127.0.0.1:7996/?cmd=recorderPluginSetActive&className=SecondRecorderPlugin&isActive=true`

JSON response: **`{}`**

3.1.2.23 recorderPluginManualCall

Description: Execute a manual call of a “recorder plug-in”. In case if the plug-in does not allow manual calls then trying to call it will result with a 422 "Unprocessable Entity" HTTP status error code.

CGI Input Parameter	Description of Parameter
[none]	

POST Request Content:
JSON data. Keys:

- **ClassName**: the Java class name of the “recorder plug-in”.
- **Arguments**: an array of Strings which are the arguments of the manual call.

Example of JSON data:

```
{"ClassName":"AAARecorderPlugin","Arguments":["true","10.2.28.5",
"hello"]}
```

POST Request Content-Type:
`femto/binary` (mandatory)

JSON Response:

- Object (empty)

Example:

API URL call: **`curl --request POST -H "Content-Type:femto/binary" --data-binary "@C:\scratch2\manualCallArguments.json" http://127.0.0.1:7996/?cmd=recorderPluginManualCall`**

JSON response: **`{}`**

3.1.2.24 recorderPluginsRescan

Description: Trigger the proxy recorder to re-scan the product directory for “recorder plug-ins”. Note: when a re-scan is performed then first all previously plug-in instances that are in active state are notified to be now in inactive state - before the references to the old plug-in instances are lost! After that then the re-scan is performed.

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object (empty)

Example:

API URL call:
GET **`http://127.0.0.1:7996/?cmd=recorderPluginsRescan`**

JSON response: **`{}`**

3.1.2.25 getRecordingState

Description: Get the current recording state of the proxy recorder.

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object
  - Key: “RecordingState”
  - Value: 0 = recoding stopped, 1 = recording started (int number)

Example:

API URL call: **GET** **`http://127.0.0.1:7996/?cmd=getRecordingState`**

JSON response: **`{"RecordingState":0}`**

3.1.2.26 startRecording

Description: Set the proxy recorder in recording state. Note: Any previously recorded data are NOT deleted.

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object (empty)

Example:

API URL call: **GET** `http://127.0.0.1:7996/?cmd=startRecording`

JSON response: `{}`

3.1.2.27 stopRecording

Description: Set the proxy recorder in non-recording state. Note: Any previously recorded data are NOT deleted.

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object (empty)

Example:

API URL call: **GET** `http://127.0.0.1:7996/?cmd=stopRecording`

JSON response: `{}`

3.1.2.28 clearRecording

Description: Delete all recorded data in the proxy recorder. Note: The recording state remains unchanged.

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object (empty)

Example:

API URL call: **GET** `http://127.0.0.1:7996/?cmd=clearRecording`

JSON response: `{}`

3.1.2.29 insertPageBreak

Description: Insert (append) a new “page break” to the recorded session. Note: this can be done regardless of the recording state.

CGI Input Parameter	Description of Parameter
comment	Optional, String. A comment about the page break.
delaySeconds	Optional, Integer. The user's think time for the page break, in seconds. If this parameter is absent the think time is set to 3 seconds.
randomization	Optional, Integer. The percentage of the randomization for the user's think time. Pass zero or a positive value not greater than 100, that can be divided by 5 (0, 5, 10, 15 ..). Example: 50 means +/- 50 percent. If this parameter is absent the randomization is set to 35 percent.

JSON Response:

- Object (empty)

Example:

API URL call: **GET**
`http://127.0.0.1:7996/?cmd=insertPageBreak&comment=this+is+my+comment&delaySeconds=4&randomization=50`

JSON response: **`{}`**

3.1.2.30 setSessionBreakDelay

Description: Set inside the Proxy Recorder for the current session a fixed value for the “user's think time” and for the randomization of the user's think time for all “page breaks”.
Note: Only already existing “page breaks” are updated - no default values are set for further added “page breaks”.

CGI Input Parameter	Description of Parameter
delaySeconds	Optional, Integer. The user's think time for all page breaks, in seconds. If this parameter is absent the think time is set to 3 seconds.
randomization	Optional, Integer. The percentage of the randomization for the user's think time. Pass zero or a positive value not greater than 100, that can be divided by 5 (0, 5, 10, 15 ..). Example: 50 means +/- 50 percent. If this parameter is absent the randomization is set to 35 percent.

JSON Response:

- Object (empty)

Example:

API URL call: **GET**
`http://127.0.0.1:7996/?cmd=setSessionBreakDelay&delaySeconds=7&randomization=60`

JSON response: **`{}`**

3.1.2.31 setSessionEnableParallelURLCalls

Description: Set inside the Proxy Recorder for the current session whether the URL calls are executed during a load test in parallel order (such as a Web browser) or in serial order (such as a Web client application).
Note: Only already existing URL calls are updated - no default values are set for further added URL calls.

CGI Input Parameter	Description of Parameter
enable	Optional, Boolean. True = execute the URL calls in parallel order. If this parameter is absent then the default value is = false.

JSON Response:

- Object (empty)

Example:

API URL call: **GET**
`http://127.0.0.1:7996/?cmd=setSessionEnableParallelURLCalls&enable=true`

JSON response: **`{}`**

3.1.2.32 getNumRecordedItems

Description: Get the total number of recorded “page breaks” + URL calls.

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object
  - Key: “NumRecordedItems”
  - Value: the number of recorded “page breaks” + URL calls (int number)

Example:

API URL call: **GET** **`http://127.0.0.1:7996/?cmd=getNumRecordedItems`**
JSON response: **`{"NumRecordedItems":41}`**

3.1.2.33 getShortSessionDump

Description: Get a short dump about the recorded session with applying the session filter.

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object
  - Key: “ShortSessionDump”
  - Value: Array of Strings. Each string contains either information about a “page break” or information about an URL call.

Example:

API URL call: **GET** `http://127.0.0.1:7996/?cmd=getShortSessionDump`
JSON response:
`{"ShortSessionDump":["[0] Page 1: Apica | Website Testing, Optimization and Monit... | User's think time: 3 sec. +/- 35%","[1] URL: https://www.apicasystem.com/ | 200 \"OK\" TEXT/HTML;charset=UTF-8 19151","[2] URL: https://www.apicasystem.com/wp-content/themes/apica-1.2/img/logo.png | 200 \"OK\" IMAGE/PNG 13412"]}`

3.1.2.34 getSessionFilter

Description: Get the filter to suppress some URL calls of the recorded session.

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object
  - Key: “SessionFilter”
  - Value: Object
    - Key: “filterBinary”
    - Value: Boolean. True = Filter out all URL calls whose HTTP response content contains binary data AND have a response status code of 200 (ok).
    - Key: “filterCache”
    - Value: Boolean. True = Filter out all URL calls whose HTTP response status code is 304 (not modified).
    - Key: “filterError”
    - Value: Boolean. True = Filter out all URL calls whose HTTP response status code is less than 100 or greater than or equal to 400.
    - Key: “filterNoHtmlText”
    - Value: Boolean. True = Filter out all URL calls whose HTTP response content is in ASCII format (=non binary) but has another content type than text/html.
    - Key: “filterHost”
    - Value: String. Filter out all URL calls whose request do not match to one or several host names. Additionally, an exclamation mark in front of a host name is also supported which means that only URL calls for this host are filtered. Several host names (or IP addresses) can be specified, separated by commas (,) - with or without an exclamation mark.

Example:

API URL call: **GET** `http://127.0.0.1:7996/?cmd=getSessionFilter`
JSON response:
`{"SessionFilter":{"filterBinary":false,"filterCache":true,"filterError":true,"filterNoHtmlText":false,"filterHost":"!adv.host.ch"}}`

3.1.2.35 setSessionFilter

Description: Set a filter to suppress some URL calls of the recorded session. The filter is applied when a recorded session is readout from the proxy recorder.
Note that the recording of URL calls by them self is not affected by the filter, meaning that the proxy recorder captures always all URL calls.

CGI Input Parameter	Description of Parameter
filterBinary	Mandatory, Boolean. True = Filter out all URL calls whose HTTP response content contains binary data AND have a response status code of 200 (ok).
filterCache	Mandatory, Boolean. True = Filter out all URL calls whose HTTP response status code is 304 (not modified).
filterError	Mandatory, Boolean. True = Filter out all URL calls whose HTTP response status code is less than 100 or greater than or equal to 400.
filterNoHtmlText	Mandatory, Boolean. True = Filter out all URL calls whose HTTP response content is in ASCII format (=non binary) but has another content type than text/html.
filterHost	Mandatory, String. Filter out all URL calls whose request do not match to one or several host names. Additionally, an exclamation mark in front of a host name is also supported which means that only URL calls for this host are filtered. Several host names (or IP addresses) can be specified, separated by commas (,) - with or without an exclamation mark.

JSON Response:

- Object (empty)

Example:

API URL call: **GET**
`http://127.0.0.1:7996/?cmd=setSessionFilter&filterBinary=true&filterCache=true&filterError=true&filterNoHtmlText=false&filterHost=www.d-fischer.com,www.pmonframe.org`
JSON response: **`{}`**

3.1.2.36 getSession

Description: Get the recorded session from the Proxy Recorder (data format: content of *.prxdat file). Note: Any settings of the session filter are not applied, but the filter settings by themself are stored in the result.

CGI Input Parameter	Description of Parameter
projectName	Optional, String. Project name for this session. Will be stored in the result data.
author	Optional, String. Author of this session. Will be stored in the result data.
comment	Optional, String. Comment for this session. Will be stored in the result data.
asBinary	Optional, Boolean. Value “true” = Get the recorded session in binary data format instead of in JSON data format. So that the received data can be directly stored as a content in a (new created) *.prxdat file.

JSON Response:

- Object
  - Key: “Session”
  - Value: Base 64 encoded String. Contains the recorded session in ProxySniffer data format.

Example 1 (JSON data):

API URL call: **GET**

```
http://127.0.0.1:7996/?cmd=getSession&projectName=test+project&author=max+miller&comment=
```

JSON response:

```
{"Session":"AQAGVjUuMi1uMi1EAAAAAAAAAAABAAZWNS4yLUQAAQAAAQAAAAAABg=="}
```

Example 2 (Binary data):

```
curl -o test-project.prxdat "http://127.0.0.1:7996/?cmd=getSession&projectName=test+project&asBinary=true"
```

Response: recorded session in ProxySniffer data format, saved in file “test-project.prxdat”

3.1.2.37 getSessionAsHAR

Description: Get the recorded session from the Proxy Recorder (data format: HTTP Archive version 1.2).

CGI Input Parameter	Description of Parameter
applySessionFilter	Mandatory, Boolean. True = Any settings of the session filter are directly applied before generating the result.

JSON Response:

- HAR/JSON Object, as defined at http://www.softwareishard.com/blog/har-12-spec/

Example 1:

API URL call: **GET**
`http://127.0.0.1:7996/?cmd=getSessionAsHAR&applySessionFilter=true`
JSON response: HAR/JSON Object (ASCII text)

Example 2:

```
curl -o TestSession.har
"http://127.0.0.1:7996/?cmd=getSessionAsHAR&applySessionFilter=true"
```

Response: recorded session in HAR data format, saved in file “TestSession.har”

3.1.2.38 loadSession

Description: Load a recorded session into the proxy recorder (upload data format: content of *.prxdat file). Note: The Session Filter will also be updated.

CGI Input Parameter	Description of Parameter
[none]	

POST Request Content:
The binary content-data of a *.prxdat file

POST Request Content-Type:
`femto/binary` (mandatory)

JSON Response:

- Object (empty)

Example 1:

API URL call: **POST** **`http://127.0.0.1:7996/?cmd=loadSession`**
JSON response: **`{}`**

Example 2:

```
curl --request POST -H "Content-Type:femto/binary" --data-binary
"@C:\scratch2\MyTests\Std_BOK.prxdat"
http://127.0.0.1:7996/?cmd=loadSession
```

3.1.2.39 loadSessionFromHAR

Description: Load a recorded session into the proxy recorder (upload data format: HTTP Archive version 1.2). Note: The Session Filter will be reset to the default value (filterCache = true, all other filter settings = false).

CGI Input Parameter	Description of Parameter
[none]	

POST Request Content:
The content-data of a *.har file

POST Request Content-Type:
`femto/binary` (mandatory)

JSON Response:

- Object (empty)

Example 1:

API URL call: **POST** **`http://127.0.0.1:7996/?cmd=loadSessionFromHAR`**
JSON response: **`{}`**

Example 2:

```
curl --request POST -H "Content-Type:femto/binary" --data-binary
"@C:\scratch2\MyTests\sbb_ch.har"
http://127.0.0.1:7996/?cmd=loadSessionFromHAR
```

3.1.2.40 getRecorderOutboundProxyTimeout (Apica BNet only)

Description: Get the recorder timeout in seconds for outbound connections to the web servers. **Internal Command – Reserved to be used by Apica BNet only.** Normally, no timeout is configured for all standard purposes.

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object
  - Key: "OutboundProxyTimeout"
  - Value: The timeout in seconds (integer value), or -1 if no timeout is configured

Example:

API URL call:
GET `http://127.0.0.1:7996/?cmd=getRecorderOutboundProxyTimeout`

JSON response: `{"OutboundProxyTimeout":-1}`

3.1.2.41 setRecorderOutboundProxyTimeout (Apica BNet only)

Description: Set the recorder timeout in seconds for outbound connections to the web servers. **Internal Command – Reserved to be used by Apica BNet only.** Normally, no timeout is configured for all standard purposes. Note that this setting is permanently stored and reapplied when the product is restarted.

CGI Input Parameter	Description of Parameter
seconds	The timeout in seconds, or pass a value of -1 (minus one) to disable the timeout (standard setting).

JSON Response:

- Object (empty)

Example:

API URL call:
GET
`http://127.0.0.1:7996/?cmd=setRecorderOutboundProxyTimeout&seconds=60`

JSON response: `{}`

3.1.2.42 getRecorderRecordFailedTransmitRequests (Apica BNet only)

Description: Get if requests that cannot transmitted to the web servers are recorded. **Internal Command – Reserved to be used by Apica BNet only.** Normally, this setting is disabled for all standard purposes.

CGI Input Parameter	Description of Parameter
[none]	

JSON Response:

- Object
  - Key: "RecordFailedTransmitRequests"
  - Value: true or false (Boolean value)

Example:

API URL call:
GET
`http://127.0.0.1:7996/?cmd= getRecorderRecordFailedTransmitRequests`

JSON response: **`{"RecordFailedTransmitRequests":false}`**

3.1.2.43 setRecorderRecordFailedTransmitRequests (Apica BNet only)

Description: Set if requests that cannot transmitted to the web servers are recorded. **Internal Command – Reserved to be used by Apica BNet only.** Normally, this setting is disabled for all standard purposes. Note that this setting is permanently stored and reapplied when the product is restarted.

CGI Input Parameter	Description of Parameter
enable	Pass a value of true or false. True = requests that cannot transmitted to the web servers are recorded.

JSON Response:

- Object (empty)

Example:

API URL call:
GET
`http://127.0.0.1:7996/?cmd=setRecorderRecordFailedTransmitRequests&enable=true`

JSON response: **`{}`**

3.2 Web Admin Server

Invocation:

```
java WebAdmin  [-tz <timezone>]
               [-dgs a | c]
               [-guitimeout <seconds>]
               [-port <number>]
               [-remote <hostname>]
               [-largeCodeUrlPerSubpage <number>]
               [-ecc]
               [-nocache]
               [-runtimedatadir <path>]
               [-execAgentConfigDir <path>]
               [-execAgentClusterConfigDir <path>]
               [-outboundipfile <file-name>]
               [-log4j]
               [-debug]
               [-h] [-help]
```

Created TCP/IP server port:

Port Number	Description
7990	Web Admin HTTP port. Displays the recorded web surfing session, and allows the management of the Proxy Sniffer Server by using a Web browser. Allows the generation of load test programs, and starting them as remote jobs on Exec Agent servers.

The Web Admin server always requires a “live” network connection to the Proxy Sniffer Server data port 7998. If the option -remote is not used, the Web Admin server will try to connect to a Proxy Sniffer Server on the localhost (127.0.0.1).

Example:

```
java -Xmx768m WebAdmin -tz PNT -dgs c
```

List of all supported options:

`-tz <timezone>`
Sets a time zone for the current date and time of the server. The default time zone is ECT. See also chapter 6: Supported Time Zones.

`[-dgs a | c]`
Sets the number format. a = apply apostrophe as decimal grouping separator (example: 123’456.00). c = apply comma as decimal grouping separator (example: 123,456.00). The default value is “a”.

`[-guitimeout <seconds>]`
Sets the request timeout in seconds of the Web Admin GUI. The default value is 420 seconds.

`[-port <number>]`
Starts the HTTP Web Admin server on a port other than 7990.

`[-remote <host>]`
Specifies a host other than localhost (127.0.0.1) to connect to the Proxy Sniffer Server.

[-largeCodeUrlPerSubpage <number>]
Set the number of max URL calls per sub-page when generating Java code.

[-ecc]
Enable support of Elliptic Curve Cryptography (ECC) for the Web Tools and Page Scanner menus.

[-nocache]
Disable caching of static content of the integrated GUI Web server.

[-runtimedatadir <path>]
Set the directory in which the runtime configuration data of Web Admin and Proxy Sniffer are written. The default directory is the working directory of Web Admin. Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6).

[-execAgentConfigDir <path>]
Set the disk path (w/o file name) at which the file **execAgent.dat** is located

[-execAgentClusterConfigDir <path>]
Set the disk path (w/o file name) at which the file **execAgentCluster.dat** is located

[-outboundipfile <file-name>]
Use multiple outbound TCP/IP addresses that are read from a configuration file to communicate with the Exec Agents, and perform TCP/IP load balancing for outbound TCP/IP connections that are established from the WebAdmin and the JobController component of ProxySniffer to the Exec Agents. The configuration file can contain multiple IP addresses on the same line or on several lines. Multiple IP addresses on the same line must be separated by space characters, or tab characters, or by commas, or by semicolons. The hash character can be used as a marker for comments within any position in a line.

[-debug]
Enables debugging of the integrated Femto Web server.

[-log4j]
Enable log4j (see chapter 3.3.1).

[-h] or **[-help]**
Displays a short help text about all Web Admin options.

3.3 Exec Agent Server

Invocation:

```
java ExecAgent [-tz <timezone>]
               [-dgs a | c]
               [-jobdir <directory>]
               [-ecc]
               [-enableJobSecurityManager]
               [-enableJobOverrideJavaMemory]
               [-enableRemoteOsCommands]
               [-minPageBreakMillis <milliseconds>]
               [-plain <port-number>]
               [-http <port-number>]
               [-https <port-number>]
               [-bind <ip-address>]
               [-restrict <ip-address>[,ip-address]]
               [-webserver]
               [-debugHousekeeping]
               [-debugSchedule]
               [-debug]
               [-periodicTimestamp <seconds>]
               [-logClientCommands]
               [-auth <base64autorisation>]
               [-internalmonitorports <port-range-start>:<port-range-end>]
               [-redirectLog <log-file>]
               [-log4j]
               [-h] or [-help]
```

TCP/IP server port:

Port Number	Description
7993	ExecAgent server port. Allows to execute several load test programs concurrently at the same time. Note that the server port can be modified by using one of the following options: -plain <port-number>, -http <port-number>, or –https <port-number>.

Example:

```
SET CLASSPATH=.;prxsniff.jar;
iaik_jce_full.jar;iaik_ssl.jar;iaik_eccelerate.jar;iaikPkcs11Provider.jar
java -Xmx768m ExecAgent -tz PNT -dgs c

or on Unix-like systems:

export CLASSPATH=.:prxsniff.jar:
iaik_jce_full.jar:iaik_ssl.jar:iaik_eccelerate.jar:iaikPkcs11Provider.jar
nohup java -Xmx768m ExecAgent -tz PNT -dgs c 2>&1 > ExecAgent.log &
```

Job Working Directory:
Note: If the **`-jobdir`** parameter is not set, the Exec Agent server uses the sub-directory "PrxExecAgentJobs", inside the system-wide temporary directory "java.io.tmpdir", as its “scratch area” (working directory).

Windows example of default Job Working Directory:
C:\DOCUME~1\miller\LOCAL~1\Temp\PrxExecAgentJobs.

The sub-directory "PrxExecAgentJobs" is automatically created if it does not exist.

List of all supported options:

[-tz <timezone>]
Set a time zone for the current date and time of the server. The default time zone is ECT. See also chapter 6: Supported Time Zones. Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6).

[-dgs a|c]
Set the number format. a = apply apostrophe as decimal grouping separator (example: 123'456.00). c = apply comma as decimal grouping separator (example: 123,456.00). The default value is "a". Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6).

[-jobdir <directory>]
Set the working directory of the Exec Agent server. Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6).

[-enableJobOverrideJavaMemory]
Allow load tests jobs to override the max. size of the Java memory used when executing the job (to override the javaXmx value configured in the file javaSetup.dat).

[-enableJobSecurityManager]
Enable a security sandbox which is applied for all load test jobs (all executed load test programs) which are started by the Exec Agent. The security sandbox protects the operating system on which the Exec Agent is running to be damaged or hacked by using malicious java code which is manually programmed inside a load test program or inside a load test plug-in. Enabling of the security sandbox has the following effect:

- The load tests programs cannot create new operating system processes (for example, shell scripts cannot be called from the load test programs).
- The load tests programs cannot write and delete files which are located outside the job directory of the current load test job.
- The load test programs cannot open new network connections to the (own) local host on which the Exec Agent is running.

Hint: it is additionally recommended that you start the Exec Agent without root or administrator privileges.

[-ecc]
Enable support of Elliptic Curve Cryptography (ECC) for the Exec Agent process. Note: using this option is **non-recommended** and does not enable ECC for the executed load test programs.

[-enableRemoteOsCommands]
Allow to execute operating system commands which are remotely released by using the **PrxJob** utility (see chapter 0). Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6). Note: this option **should not be enabled** in insecure operating environments. Hint: if you enable this option you may additionally use the option **-restrict <ip-address>[,ip-address]]** to avoid calling the Exec Agent from unwanted remote systems.

[-minPageBreakMillis <milliseconds>]
Set a minimum value for the page break delay, applied to all load test programs that are started by the Exec Agent (default: 0 milliseconds).

[-plain <port-number>]
Start the Exec Agent by applying the plain/raw protocol to communicate with the Web Admin on a specified port, other than the default port 7993.

[-http <port-number>]
Start the Exec Agent by applying the HTTP protocol to communicate with the Web Admin.

[-https <port-number>]
Start the Exec Agent by applying the encrypted HTTPS protocol to communicate with the Web Admin.

[-bind <ip-address>]
Binds the TCP/IP server port to a particular IP address of a local network interface (default: bind to all local IP addresses – and to all local network interfaces).

[-restrict <ip-address>[,ip-address]]
Restricts the usage of the Exec Agent to a list of remote TCP/IP addresses (IP address filter used to deny calls of the Exec Agent from unwanted remote systems). By default – if this option is not used – the Exec Agent can be called from any remote system.

[-webserver]
Enable the integrated Femto Web Server in order that the Exec Agent serve also static files such as Web pages, or dynamic content generated by "weblets". This option is only supported if the Exec Agent was started with the -http or -https option.

[-debugHousekeeping]
Enable debugging of the internal housekeeping that deletes corrupt job data on disk. The debug information is written to stdout.

[-debugSchedule]
Enable debugging of the job scheduler. The debug information is written to stdout.

[-debug]
Enable debugging of the integrated Femto Web Server. This option is only supported if the Exec Agent was started with -http or -https. The debug information is written to stdout.

[-periodicTimestamp <seconds>]
Direct the Exec Agent to write a time stamp in periodical intervals of seconds to stdout.

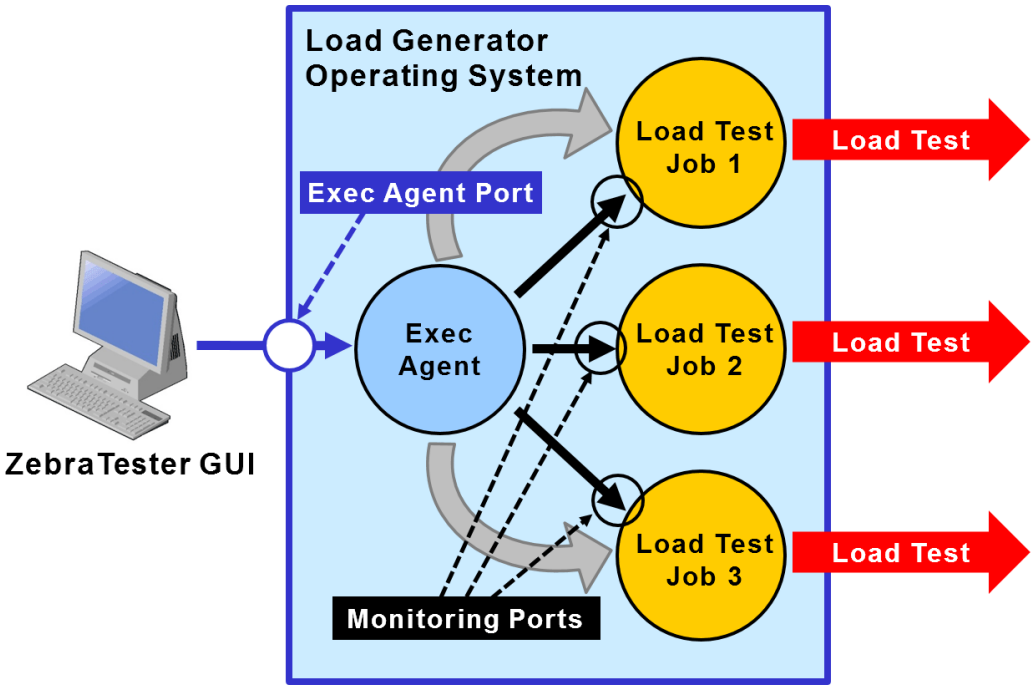
[-logClientCommands]
Direct the Exec Agent to write basic information about each valid client command to stdout (current time, remote IP address, and number of command type).

[-auth <base64autorisation>]
Protect the usage of the Exec Agent with an username and password, which must set as a base64-encoded argument; for example: test:access ➔ dGVzdDphY2Nlc3M=. You can use the **Base64 Text Transformation** utility located inside the **Web Tools** menu of the Web Admin GUI to convert a plain username:password to the base64 format.

[-internalmonitorports <port-range-start>:<port-range-end>]
Allows you to configure the TCP/IP port range which is used for real-time monitoring of the load test jobs from the GUI (trough the Exec Agent / via the Exec Agent port – the GUI do not require access to such ports, but each running load test job is bound to an own, unique monitoring port). The default range of the **internal monitoring ports** starts from port 40001 and ends at port 43999. Note that you have to use this option only in cases if you start several Exec Agents on the same computer system – to avoid double occupancy of the internal monitoring ports. This means that each Exec Agent which is started on the same machine must use an own port range, which differs from the port range of the other Exec Agents. Each port range must contain at least 100 ports. The lowest used port should not be less than port number 1025 (we recommend that all port ranges are starting form a port greater than 40000).

Hint: Please consider that starting several Exec Agents on the same machine is in principle not recommended. Each Exec Agent process is already able to handle several running load

test jobs simultaneously.



Example: java -Xmx768m ExecAgent -internalmonitorports 19300:19999

`[-redirectLog <log-file>]`
Redirect stdout and stderr to a log file, new output will be appended if the file already exists.

`[-log4j]`
Enable log4j (see next subchapter, below on this page).

`[-h]` or **`[-help]`**
Display a short help text about all Exec Agent options.

3.3.1 Log4j

As an option, stand-alone started ZebraTester components (such as ProxySniffer, WebAdmin, ExecAgent and JobController) which run as a Windows Service or as a Linux daemon can configured in such a way that log4j is used for logging. Note that log4j is only used by the ZebraTester component(s) itself, but not by the generated load test programs.

To enable log4j proceed as follows:

1. If you have manually installed ZebraTester (w/o ZebraTester installation kit) then copy the two files **log4j-core-2.3.jar** and **log4j2.xml** to your ZebraTester installation directory. You can take these files from any ZebraTester installation.
2. If you run the ZebraTester component as a Windows Service you have to stop and un-install the Windows Service.
3. Then modify the Windows Service *.bat file or the Linux startup file of the ZebraTester component in such a way that **log4j-core-2.3.jar** is part of the **CLASSPATH** and **add the startup option -log4j** to the ZebraTester component. Command line example (Windows):

   ```
   set CLASSPATH=%CLASSPATH%;log4j-core-2.3.jar
   Java -Xmx1024m ExecAgent -tz PST -log4j
   ```

4. Restart the Linux daemon, respectively re-install and re-start the Windows Service.

The following log levels are implemented:

- **INFO**: Most commonly used.
- **WARN**: Rarely used. An error did happen but it is possible to recover from this error by repeating the task, or by executing an alternative failover algorithm, or by repairing a corrupted data file on the fly.
- **ERROR**: A function was aborted because of an error. This includes also a Java stack trace. The process of the ZebraTester component is still alive after an error occurred.
- **FATAL**: A severe error occurred and the processes of the ZebraTester component performs a self-termination by calling System.exit(). After that the processes of the ZebraTester component is dead.

There is only one log4j logger implemented, named "**ZBA**".

XML configuration example (log4j2.xml):

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="warn" monitorInterval="30">
<Appenders>
    <Console name="Console" target="SYSTEM_OUT">
        <PatternLayout pattern="%d{dd MMM yyyy HH:mm:ss} %logger{36} [%t] %-5level %msg%n"/>
    </Console>
</Appenders>
<Loggers>
    <Logger name="ZBA" level="info" additivity="false">
        <AppenderRef ref="Console"/>
    </Logger>
    <Root level="error">
        <AppenderRef ref="Console"/>
    </Root>
</Loggers>
</Configuration>
```

Note: It's strongly recommended that the name of the thread (**[%t]**) is always part of the "PatternLayout".

Special Notes for Exec Agents: If log4j is configured in such a way that no output is written to stdout then the WebAdmin GUI function of remotely getting the last 2000 lines of the Exec Agent log will no longer work. In such a case it can be a workaround to configure a second logger to stdout, and to redirect stdout of the Exec Agent process to the null device. This will re-enable the function of getting the last 2000 lines, even when the null device is used.

3.3.2 Dynamic Memory Configuration for Load Test Jobs

If all of the Exec Agent you are using support the **`-enableJobOverrideJavaMemory`** option you can additionally, as a further option, place a configuration file named **DynamicExecAgentMemory.conf** in the Working Directory of the WebAdmin process and of the JobController process (in both of them if they use different working directories).

This effects that the Java Memory setting of the load test jobs is dynamically configured at runtime.

The most common reason for a dynamic configuration is to reduce the OS memory for such load tests jobs which are simulating only a few number of users. In order that many of such “small” jobs can be run at the Exec Agents, in parallel with other jobs that simulate many users.

Example of **DynamicExecAgentMemory.conf**

```
numUsersUpperLimit = 200
offsetMB = 50
memPerUserMB = 10
maxLimitMB = 1536
```

Configuration Parameter:

- **numUsersUpperLimit**: The maximum number of simulated users of a load test job for which a dynamic memory size is calculated. If the load test job simulates a larger number of users than this value then no dynamic memory size is applied for that job. This parameter can be used to decrease the memory for load test jobs which simulate a small number of users. Default value = -1 (disabled limit).
- **offsetMB**: The basic offset of memory assigned to all load test jobs, in megabytes. Default value = 100.
- **memPerUserMB**: The additional memory assigned per simulated user of a load test job, in megabytes. Default value = 10.
- **maxLimitMB**: The maximum size of memory assigned to a load test job, in megabytes. Note that this value is only applied when the memory is dynamically configured. Default value = 2048.

Examples:

numUsersUpperLimit = 200
offsetMB = 50
memPerUserMB = 10
maxLimitMB = 1536

Number of Users of Load Test Job	Java Memory in Megabytes
5	100
10	150
20	250
50	550
100	1050
200	1536 [1]
500 [2]	No Dynamic Configuration → Use Settings of Exec Agent(s)

[1] = maxLimitMB reached

[2] = greater than numUsersUpperLimit

3.4 Job Controller Server

Invocation:

```
java JobController   [-tz <timezone>]
                     [-dgs a | c]
                     [-jobdir <directory>]
                     [-ecc]
                     [-outboundipfile <file-name>]
                     [-debugHousekeeping]
                     [-debugSchedule]
                     [-periodicTimestamp <seconds>]
                     [-log4j]
                     [-h] or [-help]
```

TCP/IP server port:

Port Number	Description
7995	Job Controller server port. Able to distribute (and split) a single load test program over an unlimited number of Exec Agents and able to merge the load test results of all of the Exec Agents to a single united result.

Example:

```
java -Xmx768m JobController -tz PNT -dgs c
```

If no **-jobdir** parameter is set, the Job Controller server uses the sub-directory "PrxClusterJobs" inside the system wide temporary directory "java.io.tmpdir" as its “scratch area” (working directory).

Example: C:\DOCUME~1\miller\LOCAL~1\Temp\ PrxClusterJobs.

The sub-directory is automatically created if it does not exist.

List of all supported options:

[-tz <timezone>]
Set a time zone for the current date and time of the server. The default time zone is ECT. See also chapter 6: Supported Time Zones.

[-dgs a | c]
Set the number format. a = apply apostrophe as decimal grouping separator (example: 123’456.00). c = apply comma as decimal grouping separator (example: 123,456.00). The default value is “a”.

[-jobdir <directory>]
Set the working directory of the Job Controller server.

[-ecc]
Enable support of Elliptic Curve Cryptography (ECC) for the Job Controller process. Note: using this option is **non-recommended** and does not enable ECC for the executed load test programs.

[-outboundipfile <file-name>]
Use multiple outbound TCP/IP addresses that are read from a configuration file to communicate with the Exec Agents, and perform TCP/IP load balancing for outbound TCP/IP connections that are established from the WebAdmin and the JobController component of ProxySniffer to the Exec Agents. The configuration file can contain multiple IP addresses on the same line or on several lines. Multiple IP addresses on the same line must be separated by space characters, or tab characters, or by commas, or by semicolons. The hash character can be used as a marker for comments within any position in a line.

[-debugHousekeeping]
Enable debugging of the internal housekeeping that deletes corrupt cluster job data on disk. The debug information is written to stdout.

[-debugSchedule]
Enable debugging of the cluster job scheduler. The debug information is written to stdout.

[-periodicTimestamp <seconds>]
Direct the Job Controller to write a time stamp in periodical intervals of seconds to stdout.

[-log4j]
Enable log4j (see chapter 3.3.1).

[-h] or **[-help]**
Display a short help text about all Job Controller options.

3.4.1 Additional Settings of the Job Controller / JobControllerProperties.dat

The Web Admin GUI, as well as the Job Controller, consider at runtime a large number parameters used to detect timeouts inside the internal ZebraTester communication. Normally there is no need to configure these parameters. However, if a timeout or an error occurs at the GUI level when running large cluster jobs these parameters can be configured by creating manually a file named **JobControllerProperties.dat** . The following example shows the content of such a file. The listed values in this example are the default values which are applied when the file JobControllerProperties.dat does not exists:

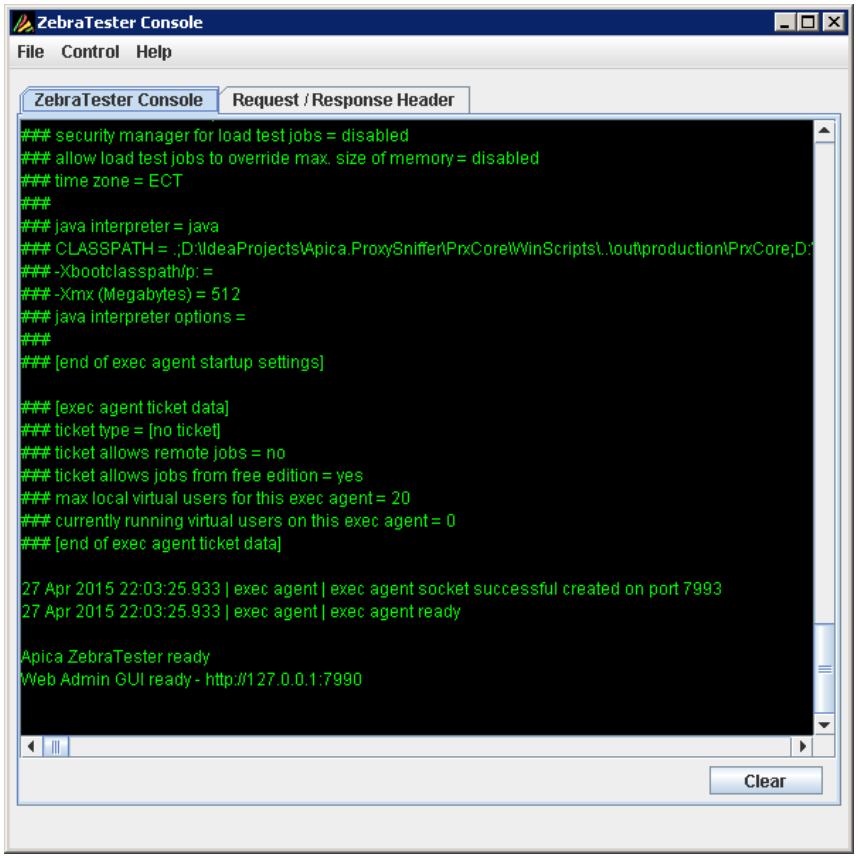
```
# Internal thread pools and delays inside the Job Controller used to avoid that too many
# new network connections are opened in a short time from the Job Controller to the Exec
# Agents (cluster members). All *Delay values are in milliseconds.

measureTimeDiffExecAgentJobsThreadPoolSize = 32
measureTimeDiffExecAgentJobsThreadStartDelay = 20
transmitExecAgentJobsThreadPoolSizeMin = 2
transmitExecAgentJobsThreadPoolSizeMax = 64
transmitExecAgentJobsThreadStartDelay = 20
startExecAgentJobsThreadPoolSize = 64
startExecAgentJobsThreadStartDelay = 20
startExecAgentJobsStatusThreadStartDelay = 10
simultaneousThreadStartDelay = 10

# Communication timeouts between the GUI and the Job Controller, all values are in seconds.
transmitJobRequestTimeout = 180
setScheduleTimeTimeout = 60
getScheduleSplitFileDefinitionTimeout = 60
getJobInputFileTimeout = 60
getJobFileZiplistTimeout = 60
splitInputFileVirtualTimeout = 180
getClusterDataTimeout = 60
setClusterDataTimeout = 90
getExecAgentListTimeout = 60
createExecAgentSubdirectoriesTimeout = 90
createExecAgentJobInputTimeout = 120
createExecAgentJobFileTimeout = 480
startExecAgentJobsTimeout = 540
getExecAgentJobsStartStatusTimeout = 240
getExecAgentJobIdTimeout = 60
getClusterJobStatusTimeout = 240
getClusterJobsListTimeout = 240
checkExecAgentKeepAliveTimeout = 60
getClusterJobExecutorsAnnotationTimeout = 90
setClusterJobExecutorsAnnotationTimeout = 90
getRunningClusterJobJavaSystemPropertiesTimeout = 90
setRunningClusterJobJavaSystemPropertiesTimeout = 90
checkClusterExecAgentsKeepAliveTimeout = 180
abortClusterJobTimeout = 90
abortClusterJobImmediatelyTimeout = 90
deleteClusterJobTimeout = 90
cleanupOldCompletedJobsTimeout = 300
cleanupAllNonRunningJobsTimeout = 300
doJobPerformanceDataAggregateTimeout = 480
getCombinedClusterFileTimeout = 360
getClusterJobFileTimeout = 180
getJobDirectoryLookupTimeout = 60
getJobClusterSubdirectoryLookupTimeout = 60
getJobExecAgentSubdirectoryLookupTimeout = 60
getJobDirectoryNameTimeout = 60
acquireJobFileTimeout = 300
```

Please note that you have to configure only such parameters which should differ from the default configuration. For example the file **JobControllerProperties.dat** may contain only two lines:

```
startExecAgentJobsThreadPoolSize = 32
startExecAgentJobsThreadStartDelay = 50
```

3.5 ZebraTester Console

Description: Starts all 4 components of the product (Proxy Sniffer, Exec Agent, Web Admin and Job Controller) in one process and displays additionally a console window. If the console window is closed all 4 components are terminated immediately.



Invocation:

```
java ProxySnifferConsole [-tz <timezone>]
                         [-dgs a | c]
                         [-runtimedatadir <path>]
                         [-jobdir <directory>]
                         [-clusterjobdir <directory>]
                         [-enableRemoteOsCommands]
                         [-guitimeout <seconds>]
                         [-outboundipfile <file-name>]
                         [-execAgentConfigDir <path>]
                         [-execAgentClusterConfigDir <path>]
                         [-log4j]
```

Example:

```
Windows:
SET CLASSPATH=.;prxsniff.jar;
iaik_jce_full.jar;iaik_ssl.jar;iaik_eccelerate.jar;iaikPkcs11Provider.jar

Unix-like systems:
export CLASSPATH=.:prxsniff.jar:
iaik_jce_full.jar:iaik_ssl.jar:iaik_eccelerate.jar:iaikPkcs11Provider.jar

java -Xmx1024m ProxySnifferConsole -tz PNT -dgs c
```

List of all supported options:

[-tz <timezone>]
Sets a time. The default time zone is ECT. See also chapter 6: Supported Time Zones. Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6).

[-dgs a | c]
Sets the number format. a = apply apostrophe as decimal grouping separator (example: 123'456.00). c = apply comma as decimal grouping separator (example: 123,456.00). The default value is "a". Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6).

[-runtimedatadir <path>]
Set the directory in which the runtime configuration data of Web Admin and Proxy Sniffer are written. The default directory is the working directory. Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6).

[-jobdir <directory>]
Sets the working directory of the Exec Agent. The default temporary directory of the operating system will be used as working directory if this option is not set (subdirectory: 'PrxExecAgentJobs'). Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6).

[-clusterjobdir <directory>]
Sets the working directory of the Job Controller. The default temporary directory of the operating system will be used as working directory if this option is not set (subdirectory: 'PrxClusterJobs'). Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6). **Note**: you cannot use the same working directory for the Exec Agent and for the Job Controller – you have to use two different (sub-) directories.

[-enableRemoteOsCommands]
Allows to execute operating system commands which are remotely released by using the PrxJob utility (see chapter 0). Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6). Note: this option **should not be enabled** in insecure operating environments.

[-guitimeout <seconds>]
Sets the request timeout in seconds of the Web Admin GUI. The default value is 420 seconds. Alternatively, this option can also be configured on Windows and Mac OS X systems in the **prxsniff.dat** file (see chapter 6).

[-outboundipfile <file-name>]
Use multiple outbound TCP/IP addresses that are read from a configuration file to communicate with the Exec Agents, and perform TCP/IP load balancing for outbound TCP/IP connections that are established from the WebAdmin and the JobController component of ProxySniffer to the Exec Agents. The configuration file can contain multiple IP addresses on the same line or on several lines. Multiple IP addresses on the same line must be separated by space characters, or tab characters, or by commas, or by semicolons. The hash character can be used as a marker for comments within any position in a line.

[-execAgentConfigDir <path>]
Set the disk path (w/o file name) at which the file **execAgent.dat** is located

[-execAgentClusterConfigDir <path>]
Set the disk path (w/o file name) at which the file **execAgentCluster.dat** is located

[-log4j]
Enable log4j (see chapter 3.3.1).

3.6 Unix Examples

3.6.1 Starting the Exec Agent Server on a SUN/Solaris System manually

```
[~/zebratester] # ls
ExecAgentTicket.dat iaik_jce_full.jar prxsniff.jar
[~/zebratester] # export CLASSPATH=.:prxsniff.jar:
iaik_jce_full.jar:iaik_ssl.jar:iaik_eccelerate.jar:iaikPkcs11Provider.jar
[~/zebratester] # java -Xmx768m ExecAgent -tz ECT -dgs c

+-----------------------------------------------------------------+
| ZebraTester V5.4-A                                              |
| Welcome to the ZebraTester Exec Agent                           |
| Copyright by Ingenieurbuero David Fischer GmbH, Switzerland     |
| http://www.proxy-sniffer.com   http://www.apicasystem.com       |
| All Rights Reserved                                             |
+-----------------------------------------------------------------+

### [startup settings]
###
### exec agent port = 7993
### exec agent bind address = [all local tcp/ip addresses]
### exec agent protocol = plain
### working directory = /tmp/PrxExecAgentJobs
### execution of remotely feed in os commands = disabled
### security manager for load test jobs = disabled
### time zone = ECT
###
### java interpreter = java
### CLASSPATH =
.:/home/miller/ZebraTester/prxsniff.jar:/home/miller/ZebraTester:/home/miller/ZebraTester/iaik_
jce_full.jar:/home/miller/ZebraTester/ iaik_ssl.jar:/home/miller/ZebraTester/
iaik_eccelerate.jar:/home/miller/ZebraTester/iaikPkcs11Provider.jar
### -Xbootclasspath/p: =
### -Xmx (Megabytes) = 256
### java interpreter options =
###
### [end of startup settings]

### [exec agent ticket data]
### ticket type = standard
### ticket allows remote jobs = yes
### max local virtual users for this exec agent = 1000
### currently running virtual users on this exec agent = 0
### [end of exec agent ticket data]

6 Jul 2015 00:56:39.191 | exec agent | exec agent socket successful created on port 7993
6 Jul 2015 00:56:39.191 | exec agent | exec agent ready
```

The configuration file **javaSetup.dat** is created automatically if it does not exist. You may modify this file; for example, you may alter “javaXmx=256” to “**javaXmx=768**” to specify 768 megabytes memory for each load test job (note - do not enter “768m”, enter only “768”).

3.6.2 Installing the Exec Agent Server on a SUN/Solaris System as a Daemon

You need root privileges to install the Exec Agent server as a daemon. A suggested procedure is as follows:

1. Create the account **zebratester** and set its home directory to **/export/home/zebratester**
2. Create the Directory **/export/home/zebratester** and make sure that the account zebratester has write access to this directory
3. Copy the files **prxsniff.jar**, **iaik_jce_full.jar**, **iaik_ssl.jar**, **iaik_eccelerate.jar**, and **iaikPkcs11Provider.jar** into this directory (you may copy these files from a Windows, Linux or Mac OS X installation)
4. Using vi, create the file **ExecAgentTicket.dat** inside this directory and add the text containing your Exec Agent license ticket
5. Using vi, create the file **/etc/rc3.d/S98zebratester** and add the lines shown below
6. Change the file protection of **/etc/rc3.d/S98zebratester** such that it can be executed as script; that is, add execute permissions
7. Run the command **/etc/rc3.d/S99zebratester start**

```
#!/bin/sh
# Startup for ZebraTester Exec Agent
#
case "$1" in
'start')
        echo "Starting ZebraTester Exec Agent ..."
        cd /export/home/zebratester
        CLASSPATH=.:prxsniff.jar:
iaik_jce_full.jar:iaik_ssl.jar:iaik_eccelerate.jar:iaikPkcs11Provider.jar
        export CLASSPATH
        su - zebratester -c "nohup java -Xmx768m ExecAgent -jobdir /export/home/zebratester -tz
ECT -dgs a > zebratester.log 2>&1 &"
        ;;
'stop')
        for proc in ExecAgent
        do
                PID=`ps -o pid,args -e | grep ${proc} | egrep -v grep | awk '{print $1}'`
                if [ ! -z "$PID" ] ; then
                        echo "ZebraTester Exec Agent with pid : $PID stopped"
                        kill -9 ${PID} 1> /dev/null 2>&1
                fi
        done
        ;;
*)
        echo "Usage: $0 { start | stop }"
        ;;
esac
exit 0
```

3.6.3 Starting the Exec Agent Server on a Linux System as a Daemon

A suggested procedure is as follows:

1. Install ZebraTester as usual by using the Linux installation kit
2. Create the startup script **ExecAgent** as shown below in **/etc/init.d** (replace in the script the path of the installation directory) and install the startup procedure with **update-rc.d ExecAgent defaults**. Note that this script example starts two ExecAgents, one on TCP/IP port 7993 and one on TCP/IP port 80.
3. After successful installation of ZebraTester install also the Linux **entropy daemon haveged** (see http://www.issihosts.com/haveged)

```
#!/bin/sh
# /etc/init.d/ExecAgent
# install with: update-rc.d ExecAgent defaults

### BEGIN INIT INFO
# Provides:          ExecAgent
# Required-Start:    $local_fs $network $time $syslog
# Required-Stop:     $local_fs $network
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start ExecAgent daemon(s) at boot time
# Description:       Apica ZebraTester ExecAgent daemon(s)
### END INIT INFO

PSDIR=/home/mutong/ZebraTester
CLASSPATH=.:prxsniff.jar:iaik_jce_full.jar:iaik_eccelerate.jar:iaik_ssl.jar:iaik
Pkcs11Provider.jar
export CLASSPATH

case "$1" in
  start)
    sleep 10
    cd $PSDIR
    nohup java -Xmx256m ExecAgent -jobdir $PSDIR/jobs -tz ECT -periodicTimestamp
60 >$PSDIR/ExecAgent7993.log 2>&1 &
    nohup java -Xmx256m ExecAgent -jobdir $PSDIR/jobs -tz ECT -periodicTimestamp
60 -http 80 -webserver >$PSDIR/ExecAgent80.log 2>&1 &
    ;;
  stop)
       PID=`ps -o pid,args -e | grep "ExecAgent -jobdir" | egrep -v grep | awk '
{print $1}'`
       if [ ! -z "$PID" ] ; then
          echo "ExecAgent stopped with pid(s) : $PID"
          kill -9 ${PID} 1> /dev/null 2>&1
       fi
    ;;
  status)
       PID=`ps -o pid,args -e | grep "ExecAgent -jobdir" | egrep -v grep | awk '
{print $1}'`
       if [ ! -z "$PID" ] ; then
          echo "ExecAgent running with pid(s) : $PID"
       else
          echo "No ExecAgent running"
       fi
    ;;
  *)
    echo "Usage: /etc/init.d/ExecAgent {start|stop|status}"
    exit 1
    ;;
esac

exit 0
```

3.7 Starting ZebraTester as a Windows Service

Special Note for Windows 7/2008/2012 Systems:

On Windows 7/2008/2012 systems you have to disable “User Account Control” (UAC) in order that the Windows Service starts successful.

Note that the 64-bit Windows installation kit of ZebraTester is optimized for high-duty load generators and require machines with at least 16 GB of RAM.

3.7.1 Starting the Exec Agent as a Windows Service

For Windows systems which are used only as a member of an Exec Agent Cluster, or which are used only as a "remote" load generator, you can start the Exec Agent as a Windows Service.

32-bit Windows system: Follow the instructions contained in the file **InstallExecAgentService.bat**.

64-bit Windows system: Follow the instructions contained in the file **InstallExecAgentServiceWin64.bat**.

Both files are located in the ZebraTester installation directory.

3.7.2 Starting the ZebraTester GUI as a Windows Service

32-bit Windows installation kit of ZebraTester: Follow the instructions contained in the file **InstallZebraTesterService.bat**.

64-bit Windows installation kit of ZebraTester: Follow the instructions contained in the file **InstallZebraTesterServiceWin64.bat**.

Both files are located in the ZebraTester installation directory.

This service will start the components "ProxySniffer", "WebAdmin" and "JobController". We recommend that you install also the Exec Agent as a service.

3.7.3 Upgrades from older ZebraTester Versions

If you upgrade to ZebraTester from an older version – and if you have already started the Exec Agent as a service – you have also first to stop and uninstall the service. This must be done before the older ZebraTester version is uninstalled.

3.8 Load Test Program Arguments

All automatically-generated load test programs have some common options available:

```
java <class-name>   [-u <concurrent users> *]
                    [-d <duration-seconds> *]
                    [-t <timeout-seconds> *]
                    [-maxloops <loops-per-user>]
                    [-sdelay <milliseconds>]
                    [-nosdelayCluster]
                    [-mtpu <number-of-threads>]
                    [-setuseragent "<text>"]
                    [-downlink <Kbps>]
                    [-uplink <Kbps>]
                    [-sampling <seconds>]
                    [-percpage <percent>]
                    [-percurl <percent>]
                    [-percurlopt <numeric-value>]
                    [-maxerrsnap <number>]
                    [-maxerrmem <megabytes>]
                    [-multihomed <input file>]
                    [-ipperloop]
                    [-ssl <version>]
                    [-sslcache <timeout-seconds>]
                    [-sslrandom <type>]
                    [-sslcmode]
                    [-ecc]
                    [-nosni]
                    [-snicritical]
                    [-dnshosts <file-name>]
                    [-dnstranslation <file-name>]
                    [-dnssrv <IP-name-server-1>[,<IP-name-server-N>]]
                    [-dnsenattl]
                    [-dnsfixttl <seconds>]
                    [-dnsperloop]
                    [-dnsstatistic]
                    [-dnsdebug]
                    [-res <filename>]
                    [-nostdoutlog]
                    [-dfl][-dl][-dh][-dc][-dC][-dK][-dssl]
                    [-h] [-help]
                    [-tz <time-zone>]
                    [-dgs a | c]
                    [-annotation "<text>"]
                    [-h]  [-help]
```

* mandatory program argument

List of all supported options:

-u <concurrent users>
Number of emulated, concurrent users.

-d <duration-seconds>
Planned test duration in seconds.

-t <timeout-seconds>
Timeout in seconds per single URL call. If this timeout expires, the URL call will be reported as an error (no response from Web server), and the current loop (Web surfing session) of the emulated user is aborted. The emulated user will then continue with the next loop.

-maxloops <loops-per-user>
Limits the maximum number of loops per user (default: 0 = unlimited).

-sdelay <milliseconds>
Alters the startup delay of the concurrent users (startup ramp of load). The default value is 200 milliseconds.

-nosdelayCluster
Effects for Cluster Jobs that the startup delay per user is applied per Exec Agent Job instead of applying it overall simulated users of the Cluster Job. Thus a faster ramp up of load can be achieved.

-mtpu <number-of-threads>
Allows to configure how many threads per simulated user are used to process URLs in parallel (simultaneously). Note: this value applies only for URLs which have been configured to be executed in parallel. The default value is 6 threads.

-setuseragent "<text>"
Replaces the recorded value of the HTTP request header field User-Agent with a new value. The new value is applied for all executed URL calls.

-downlink <Kbps>
Limits the network bandwidth per concurrent user (in kilobits per second) for the downlink (Web server to Web browser/load test)

-uplink <Kbps>
Limits the network bandwidth per concurrent user (in kilobits per second) for the uplink (Web browser/load test to Web server)

-sampling <seconds>
Alters the statistics sampling interval applied during the load test. The default value is 15 seconds. If you run a load test over several hours, it is recommended that you increase the statistics sampling interval. If the load test lasts only a few seconds, you may decrease this sampling interval.

-percpage <percent>
Alters the sampling rate of percentile statistics data for Web pages. If > 0%, it enables the display of percentile response time diagrams for each Web page in the Analyse Load Test Details menu. (default value: 100%).

-percurl <percent>
Alters the sampling rate of percentile statistics data for URL calls. If > 0%, it enable the display of percentile response time diagrams for each URL call in the Analyse Load Test Details menu. (default value: 20%).

-percurlopt <numeric-value>
Additional measuring option per sampled URL call. (default value: 0). Is only considered if the value of **-percurl** is greater than 0.

percurlopt value	Meaning
0	no options (recommended)
1	all URL performance details (TCP socket open time, request transmit time, ...)
2	request header (+ all URL performance details)
3	request content (+ all URL performance details
4	request header & request content (+ all URL performance details)
5	response header
6	response header & response content
7	all - but without response content
8	all - full URL snapshot
9	TCP/IP client data (+ all URL performance details)
11	in-depth throughput data of the received HTTP response content (+ all URL performance details)

-maxerrsnap <number>
Limits the maximum number of error snapshots per URL – counted per Exec Agent (default value: 0 = unlimited). Note: using this option is not recommended.

-maxerrmem <megabytes>
Limits the maximum memory in megabytes used to store all error snapshots – counted overall Exec Agents (default value: 20, unlimited: -1). Note: using greater values than 100 or a value of -1 (unlimited) is not recommended.

-multihomed <input file>
The effect of this option is that each concurrent user will use, during the load test, its own client IP address. If there are more concurrent users than IP addresses, the IP addresses are averaged over the concurrent users. To use this option, the underlying Windows or Unix operating system must be configured such that multiple (virtual) IP addresses are assigned to the same physical network interface(s). The input file contains only one line, on which all client IP addresses are listed, separated by comma characters.

-ipperloop
Using this option in combination with the option -multihomed effects that a own local IP address is used for each executed loop rather than for each simulated user.
This option is considered only if also the option -multihomed is used.

-ssl <version>
Allows the setting of a fixed SSL protocol version. Possible values are "**all**" (automatic detection of v3/tls/tls11 – recommended default value), "**v3**", "**tls**", or "**tls11**" (fixed protocol version).

-sslcache <timeout-seconds>
Alters the timeout of the user-related SSL session cache. The default value is 300 seconds. A value of 0 (zero) disables the SSL cache.

-sslrandom <type>
Set the type of the random generator used for SSL handshakes. Possible options for the <type> are "java", "iaik" or "fast".

- **java**: Use the standard Java secure random generator for SSL handshakes. This random generator generate secure random numbers, but its usage is non-recommended for load test execution because it may be very slow and may block on Linux-like systems.
- **iaik**: Use the IAIK random generator for SSL handshakes (SHA256PRNG-FIPS). This random generator is used by default.
- **fast**: Use a fast but very weak (non-)secure random generator for SSL handshakes. This random generator will never block on any operating system.

-sslcmode
Apply SSL (https) compatibility workarounds for defective SSL servers. You may try this option if you constantly see the error type "Network Connection aborted by Server" for all http/s calls.

-ecc
Enable for SSL/TLS connections rarely used, additional encryption algorithms like ECC.

-nosni
Disable support for TLS server name indication (SNI).

-snicritical
Set the TLS SNI extension as critical (default: non-critical).

-dnshosts <file-name>
Effects that the load test job uses an own DNS hosts file to resolve host names - rather than using the hosts file of the underlying operating system.
Note that you have to ZIP the hosts file together with the compiled class of the load test program. To automate the ZIP it's recommended to declare the hosts file as an external resource (w/o adding it to the CLASSPATH).

-dnstranslation <file-name>
Effects that the load test job uses a DNS translation file which is a text file that contains on each line a translation between two DNS names. If the first DNS name in the file match to the DNS name that is passed to the resolver then the second DNS name is used to resolve the IP address.
The first DNS name can also contain one or more wildcard characters ('*' = wildcard for multiple characters, '?' = wildcard for single character). Lines or a part of a line can be commented out by using the hash char '#'.

Example of a DNS translation file:

```
www.proxy*sniffer.com www.proxy-sniffer.com # comment
# comment
www.mutong.com www.d-fischer.com
mail?.google.com mail.google.com
```

Note 1: It could be needed that TLS SNI (Server Name Indication) must be disabled if a DNS translation table is used.
Note 2: The HTTP request header field "Host" is not updated, so it might happen that you call the Web server with the wrong host name.

You have to ZIP the DNS translation file together with the compiled class of the load test program. To automate the ZIP it's recommended to declare the DNS translation file as an external resource (w/o adding it to the CLASSPATH).

-dnssrv <IP-name-server-1>[,<IP-name-server-N>]
Effects that the load test job uses specific (own) DNS server(s) to resolve host names - rather than using the DNS library of the underlying operating system.

When using this option, at least one IP address of a DNS server must be specified. Multiple DNS servers can be configured separated by commas. If a resolved DNS host name contains multiple IP addresses the stressed Web servers are called in a round-robin order (user 1 uses resolved IP Address no. 1, user 2 uses resolved IP Address no. 2, etc.).

-dnsenattl
Enable consideration of DNS TTL by using the received TTL-values from the DNS server(s). This option cannot be used in combination with the option **-dnsperloop**.
Note: when using this option the resolved IP addresses (and therefore the stressed Web servers) may alter inside the executed loop of a simulated user at any time - suddenly from one URL call to the next one.

-dnsfixttl <seconds>
Enable DNS TTL by using a fixed TTL-value of seconds for all DNS resolves. This option cannot be used in combination with the option **-dnsperloop**.
Note: when using this option the resolved IP addresses (and therefore the stressed Web servers) may alter inside the executed loop of a simulated user at any time - suddenly from one URL call to the next one.

-dnsperloop
Perform new DNS resolves for each executed loop. All resolves are stable within the same loop (no consideration of DNS TTL within a loop).
This option cannot be used in combination with the options **-dnsenattl** or **-dnsfixttl**.
Note: consider when using this option that the default or the configured DNS servers are stressed more than usual because each executed loop of each simulated user will trigger one or more DNS queries.

-dnsstatistic
Effects that statistical data about DNS resolutions are measured and displayed in the load test result, by using an own DNS stack on the load generators.
Note: there is no need to use this option if any other, more specific DNS option is enabled because all (other) DNS options also effect implicitly that statistical data about DNS resolutions are measured. If you use this option without any other DNS option, the (own) DNS stack on the load generators will communicate with the default configured DNS servers of the operating system - but without considering the "hosts" file.

-dnsdebug
Effects that debug information about the DNS cache and about DNS resolves is written to the stdout file (*.out) of the load test job.

-res <filename>
Overrides the default name for the binary output file (*.prxres).

-nostdoutlog
Disables writing any date to the *.out file of the load test job. Note that the *.out file is nevertheless created but contains zero bytes.

-dfl
Debug Failed Loops: Write log data about the execution steps of all failed loops to standard output.

-dl
Debug Loops: Write log data about the execution steps of all loops to standard output.

-dh
Debug Headers: Write all HTTP headers exchanged during the load test execution to standard output. This implicitly enables the option -dl.

-dc
Debug Content: Write the transmitted and received content to standard output. This implicitly enables the option -dl.

-dC
Debug Cookies: Write all cookies exchanged during the load test execution to standard output (debug cookies). This implicitly enables the option -dl.

-dK
Debug Keep-Alive: Write the behavior of re-used https/s network connections to standard output. This implicitly enables the option -dl.

-dssl
Debug SSL: Write additional debug information about the SSL protocol and the SSL handshake to standard output. This implicitly enables the options -dl and -dK.

-tz <time zone>
Sets a time zone for the current date and time. The default time zone depends on the time zone set inside the Web Admin menu “Personal Settings” – before the load test program has been generated. See also chapter 6: Supported Time Zones.

-dgs a | c
Sets the number format. a = apply apostrophe as decimal grouping separator (example: 123’456.00). c = apply comma as decimal grouping separator (example: 123,456.00). The default value depends the settings set inside the Web Admin menu “Personal Settings” – before the load test program has been generated.

-annotation "<text>"
Allows the setting of an annotation, or comment, about the test-run. The annotation will be stored in the binary result file (*.prxres), and is displayed together with the measured data.

[-h] or **[-help]**
Displays a short help text about all program options.

Note: if a load test program contains **User Input Fields**, additional parameters are required, and each additional parameter must have the same name as the variable name of the corresponding User Input Field.

3.9 License Information Utilities

There are two license information utilities available which allows you to get detailed information about a GUI License Key and about an Exec Agent License Ticket.

3.9.1 PrxGuiLicenseKeyInfo

Write information about a GUI License Key to stdout.

```
java PrxGuiLicenseKeyInfo <GUI License Key>
                          [<host name or IP address>]
```

- The first argument is required and contains the GUI License Key.
- The second argument is optional and contains any TCP/IP host name or IP address and can be used to verify if the GUI License Key match to a specific machine (if the key is node locked / Universal Key = false).

Exit Code: 0 = success: key data format valid, 1 = failure: key data format invalid

Example 1:

```
java PrxGuiLicenseKeyInfo QRsA-DAIe-AAAU-UjmV-s*E@
Key Id = 3521
License Type = Professional Edition
Max Version = 5.2
Valid Until = 30 Mar 2015
Universal Key = true
```

Example 2:

```
java PrxGuiLicenseKeyInfo PhEA-BwcB-j8oV-Q05S-UA0@ win72
Key Id = 2238
License Type = Professional Edition
Max Version = 4.3
Valid Until = 1 Aug 2010
Universal Key = false
Valid for Host = true
```

Notes:

- The output field “Valid Until” can also have the value “forever” if the GUI License Key is infinitely valid.
- The output field “Valid for Host” is only shown if the GUI License Key is bound to a host name or IP address AND if the second (optional) argument is passed to the utility.

3.9.2 PrxExecAgentLicenseTicketInfo

Write information about an Exec Agent License Ticket to stdout.

```
java PrxExecAgentLicenseTicketInfo <License Ticket File>
```

- The argument <License Ticket File> contains the relative or absolute path of the file containing the license ticket (usually **ExecAgentTicket.dat**).

Exit Code: 0 = success: license ticket data format valid, 1 = failure: license ticket data format invalid – or unable to read license ticket file

Example 1:

```
java PrxExecAgentLicenseTicketInfo ExecAgentTicket_old.dat
Ticket Number = 1520
Ticket Type = standard
Max Version = 4.6
Allow Remote Jobs = true
Universal Host = false
Bound to Host = esload-fhu
Universal Port = false
Bound to Port = 7993
Not Valid Before = 23 Jul 2012
Valid Until = 31 Jul 2013
Max Users = 200
Issued For = Scalaris AG
Comment = Load Test ZebraTester new Version
```

Example 2:

```
java PrxExecAgentLicenseTicketInfo ExecAgentTicket.dat
Ticket Number = 1815
Ticket Type = standard
Max Version = 5.2
Allow Remote Jobs = true
Universal Host = true
Universal Port = true
Not Valid Before = 20 May 2014
Valid Until = 10 Jan 2015
Max Users = 250
Issued For = PRX: Product Evaluation
Comment = Develop / fischer@d-fischer.com
```

Notes:

- The output field “Bound to Host” is only shown if “Universal Host” = false.
- The output field “Bound to Port” is only shown if “Universal Port” = false.
- The output field “Not Valid Before” can be absent.
- The output field “Valid Until” can also have the value “forever” if the Exec Agent License Ticket is infinitely valid.
- The output field “Max Users” can also have the value “unlimited”.
- The output fields “Issued For” and “Comment” can be absent.

3.10 Speed Test Tool

The Speed Test tool can be used to determine the maximum possible network speed between a Web client and a Web Server. This tool contains an ultra-lightweight HTTP client, as well as an ultra-lightweight HTTP server. . This measured network speed can be used as a baseline for the fastest possible response time for a real Web application server.

The Speed Test client can also be run against a real Web server (without using the Speed Test Server program); however, using this additional feature may have show a lower measured network speed than is really possible.

3.10.1 Speed Test Server

First you must copy the file **prxsniff.jar** to the target (Web server) system. After this, you can start the Speed Test Server program manually from the command line. A ZebraTester license key is not required; therefore, you can run the server on any system.

The implementation of the Speed Test Server program is multi-threaded in order that several client requests can be processed at the same time. For performance reasons, no log data are written during execution.

Invocation:

```
java SpeedTestServer [-port <TCP/IP port number>]
                     [-bind <IP address>]
                     [-size <bytes>]
                     [-buffer <bytes>]
                     [-h] [-help]
```

Windows Example:

```
set CLASSPATH=.;prxsniff.jar
java SpeedTestServer -port 8090
```

Unix Example:

```
bash
export CLASSPATH=.:prxsniff.jar
java SpeedTestServer -port 8090
```

List of all supported options:

-port <TCP/IP port number>
Set the TCP/IP server port. The default port is 80. Note that you need Administrator or root privileges to start the Speed Test Server on a TCP/IP port which is lower than 1024.

-bind <IP address>
Allows the binding of the Speed Test Server program to a particular physical network interface. Using this option only makes sense if several physical network interfaces are available. The default is for the Speed Test Server program to be bound to all physical network interfaces.

-size <bytes>
Set the size of the HTTP response content (server response size). The default value is 1048576 bytes (1 MB). Note: setting a response size lower than 65536 bytes (64 kbytes) is not recommended.

-buffer <bytes>
Set the size of the HTTP response buffer. The default value is 65536 bytes (64 kbytes). Note: setting a buffer size lower than 8192 bytes (8 kbytes) is not recommended.

[-h] or **[-help]**
Display a short help text about all program options.

3.10.2 Speed Test Client

You must specify an absolute URL as the first argument to the Speed Test Client. If you run the client against a real Web server, instead of against the Speed Test Server, the URL must be valid and the server must respond with a 200 ok HTTP status code. Please note that the requested content should be large - less than 65536 bytes is not recommended.
The Speed Test Client does not support encrypted HTTPS connections.

Invocation:

```
java SpeedTestClient <absolute URL>
                     [-threads <number>]
                     [-iterations <number>]
                     [-buffer <bytes>]
                     [-h] [-help]
```

Example:

java SpeedTestClient http://192.16.4.55/

List of all supported options:

-threads <number>
Set the number of requests (or threads) which are executed in parallel. The default value is 4. We recommend that you experiment a little, using different numbers of threads, to get the maximum network speed.

-iterations <number>
Set the number of request repetitions executed by each thread. The default value is 10.

-buffer <bytes>
Set the size of the HTTP receive buffer. The default value is 65536 bytes (64 kbytes). Note: it is not recommended to set a buffer size lower than 8192 bytes (8 kbytes).

[-h] or **[-help]**
Display a short help text about all program options.

Measurement Result:

After measurement has completed, the Speed Test Client writes the result to stdout. Example:

```
======================== speed test done ========================

input data:
- url = http://192.16.4.55/
- nr of threads = 4
- nr of iterations per thread = 10
- receive buffer size = 65536
[end of list]

output data:
- av socket open time = 51 ms
- av request delay = 1 ms
- av response delay = 12 ms
- total throughput = 68.26563 mbit/sec
- total received bytes = 40.003662 mbytes
- nr of executed calls = 40
- av response throughput per call = 55.515472 mbit/sec
- av response content size = 1048576 bytes
[end of list]
```

Description of output data:

- **av socket open time:** time to open a new network connection (average, measured per request in milliseconds)
- **av request delay:** time to transmit the request (average, measured per request in milliseconds)
- **av response delay:** time to receive the first byte of the response, after transmitting the request (average, measured per request/response in milliseconds)
- **total throughput:** total network throughput (megabits per seconds, measured overall)
- **total received bytes:** total number of received bytes (megabytes, measured overall)
- **nr of executed calls:** total number of successfully-executed URL calls
- **av response throughput per call:** network throughput while receiving the response (average, measured per response in milliseconds). Note: this value is usually lower than the “total throughput” because several requests (or threads) run in parallel.
- **av response content size:** size of the received response content (average, measured per response in bytes)

3.11 Page Scanner Tool

The Page Scanner tool can also be started from the command line. In addition to this chapter, please read the corresponding chapter in the User's Guide to learn more about this tool. The command line version does only support basic authentication against Web servers, in contrast to the GUI version which supports additionally NTLM authentication and X509 client certificate-based authentication.

Note: by using the **Session Converter** Tool (see next chapter 3.11.1) and the **Session To Java Tool** (chapter 3.11.2) the Page Scanner result can be converted into a ready-to-run load test program which can be started from the command line by using the **PrxJob** utility (see chapter 4).

Invocation:

```
java PageScanner <absolute URL>
                 [-tz <time-zone>]
                 [-dgs a | c]
                 [-nowelcome]
                 [-language <abbreviation>]
                 [-charset <name>]
                 [-addserver <server-url[, ...]>]
                 [-add <file-type | type-list[, ...]>]
                 [-remove <file-type | type-list[, ...]>]
                 [-ignore <file-path-pattern[, ...]>]
                 [-timeout <seconds>]
                 [-maxtree <tree-depth>]
                 [-maxpages <number>]
                 [-maxurls <number>]
                 [-maxbytes <number>[k | m]]
                 [-maxtime <seconds>]
                 [-followredirections <number>]
                 [-followpathrepetitions <number>]
                 [-followcgi]
                 [-nokeepurldata]
                 [-ssl <version>]
                 [-authbasic <username>:<password>]
                 [-nolog]
                 [-title]
                 [-checksum]
                 [-save [<file-name>]]
                 [-comment <text>]
                 [-load <file-name>]
                 [-input]
                 [-noscandone]
                 [-nostatistic]
                 [-sort]
                 [-nopages]
                 [-largepages]
                 [-slowpages]
                 [-details]
                 [-doublets]
                 [-failures]
                 [-nononprocessed]
                 [-h] [-help]
```

Example:

```
java -Xmx256m PageScanner http://www.proxy-sniffer.com -save -maxtime
600 -largepages -slowpages -comment "my first scan" -tz PNT -dgs c
```

List of all supported options:

-tz <timezone>
Set a time zone for the current date and time of the server. The default time zone is ECT. See also chapter 6: Supported Time Zones.

[-dgs a | c]
Set the number format. a = apply apostrophe as decimal grouping separator (example: 123'456.00). c = apply comma as decimal grouping separator (example: 123,456.00). The default value is "a".

-nowelcome
Suppress the Page Scanner welcome message.

-language <abbreviation>
Set the preferred Web browser language; for example, 'en' or 'de' (default: no preference)

-charset <name>
Set a fixed character set for parsing Web pages (default: auto detect). Example of a character set name: "ISO-8859-1"

-addserver <server-url[, ...]>
Allows the inclusion of content and Web pages from other Web servers within the scan. Example: -addserver http://imageserver.proxy-sniffer.com

-add <file-type | type-list[, ...]>
Add file types to the list of processed URL calls (default: only HTML files are processed)

Type List Name	Corresponding File Extensions
css	.css, .stylesheet
js	.js, .javascript
image	.img, .bmp, .gif, .pct, .pict, .png, .jpg, .jpeg, .tif, .tiff, .tga, .ico
animation	.swf, .stream
pdf	.pdf
text	.txt, .text, .log, .asc, .ascii, .csv
office	.doc, .docx, .ppt, .pps, .xls, .mdb, .wmf, .rtf, .wri, .vsd, .rtf, .rtx
exe	.exe, .msi, .dll, .bat, .com, .pif
binary	.dat, .bin, .vcd, .sav
archive	.zip, .gzip, .cab, .jar, .tar, .gtar , .rar, .gz, .bz2, .z
movie	.avi, .mov, .wm, .rm, .mpeg
music	.mp2, .mp3, .mpg, .avi, .wav

Example to add a single file type: -add .js
Example to add a type list: -add js

-remove <file-type | type-list[, ...]>
Remove file types from the list of processed URL calls

-ignore <file-path-pattern[, ...]>

Add fragments of URL file paths which must not be processed

-timeout <seconds>
Set the timeout for URL calls, in seconds (default: 20)

-maxtree <tree-depth>
Restrict the level of scanning to a maximum file path depth (default: unrestricted)

-maxpages <number>
Set the maximum number of scanned Web pages (default: unlimited)

-maxurls <number>
Set the maximum number of executed URL calls (default: unlimited)

-maxbytes <number>[k|m]
Set the maximum number of received bytes, k = kilobytes, m = megabytes
(default: 40MB, -1 = unlimited). Example: -maxbytes 100m

-maxtime <seconds>
Set the maximum time for the scan (default: 3600 seconds, -1: unlimited)

-followredirections
Limit the number of followed redirections (default: 10, -1: unlimited)

-followpathrepetitions <number>
Limit the number of followed file path repetitions (default: 1, -1: unlimited)

-followcgi
Follow URLs with same file path, but with different CGI parameters (default: disabled)

-nokeepurldata
Do not keep the transmitted and received data of the URL calls (default: keep URL data).
Note: do not use this option if you plan to convert the scan result into a Web surfing session.

-ssl <version>
Set the SSL protocol version. Possible values are v2, v3, or tls (default: automatic selection)

-authbasic <username>:<password>
Enable basic authentication

-nolog
Disable the logging of executed URL calls during the scan

-title
Display the title of the Web pages

-checksum
Display a checksum of the received content for each processed URL call

-save [<file-name>]
Save the scan result file (*.prxscn). The file name is optional. The default file name is derived from the <absolute URL> input parameter. Note: you must use this option if you plan to convert the scan result into a Web surfing session

-comment <text>
Add a comment to the scan result file (default: no comment)

-load <file-name>

Load a saved scan result file, and display the scan result

-input
Display the input arguments of the scan upon completion

-noscandone
Suppress displaying the "scan done" message upon completion

-nostatistic
Suppress the scan statistic upon completion

-sort
Display a sorted list of all executed URL calls upon completion

-nopages
Suppress a statistic of all Web pages upon completion

-largepages
Display a statistic of all Web pages upon completion, sorted by page size

-slowpages
Display a statistic of all Web pages upon completion, sorted by page response time

-details
Display a list of all Web pages, and their containing URL calls, upon completion

-doublets
Display, upon completion, a list of URLs which contain duplicated content data

-failures
Display a list failed URL calls upon completion

-nononprocessed
Suppress the display of the list of non-processed Web servers upon completion

-h or **-help**
Display a short help text about all program options.

Exit status codes:

-2 = internal error
-1 = scan not started
0 = scan terminated normally
11 = scan terminated: max number of Web pages exceeded
12 = scan terminated: max number of URL calls exceeded
13 = scan terminated: max number of bytes exceeded
20 = scan terminated: max scan time exceeded

3.11.1 Page Scanner “Session Converter” Tool

The Page Scanner “Session Converter” tool allows you to convert a scan result file (*.prxscn) into a Web surfing session file (*.prxdat), from which a ready-to-run load test program can be generated (see next chapter 3.11.2).

Invocation:

```
java PageScannerSessionConverter <input-file>
                                 [-save <output-file>]
                                 [-projectname <text>]
                                 [-author <text>]
                                 [-comment <text>]
                                 [-thinktime <seconds>]
                                 [-thinkrandom <percent>]
                                 [-mtpu <number-of-threads>]
                                 [-sortpages]
                                 [-nocache]
                                 [-maxpages <number>]
                                 [-maxurls <number>]
                                 [-maxbytes <number>[k | m]]
                                 [-nostatistic]
                                 [-h] [-help]
```

Example:

```
java -Xmx256m PageScannerSessionConverter
proxysniffer_com_80_08Jun07_220645.prxscn
```

List of all supported options:

-save <output-file>
Set the output-file name (default: same name as the input file, but with file extension ".prxdat")

-projectname <text>
Set the project name of the Web session (default: no project name)

-author <text>
Set the author of the Web session (default: no author)

-comment <text>
Add a comment about the Web session (default: adopt comment from the input file)

-thinktime <seconds>
Set the user's think time per Web page (default: 3 seconds)

-thinkrandom <percent>
Set the randomization of the user's think time (default: 35 percent)

-mtpu <number-of-threads>
Configure how many threads per simulated user are used to process URLs in parallel (default: all URLs are processed in serial order). Note: if you use this option you should set the amount of threads to an even number between 2 and 24. The recommended value is 6.

-sortpages
Sort the Web pages by host, protocol, IP port, and file path (default: no sort = scan order)

-nocache
Disable the simulation of the Web browser cache (default: simulation is enabled)

-maxpages <number>
Set the maximum number of converted Web pages (default: unlimited)

-maxurls <number>
Set the maximum number of converted URL calls (default: unlimited)

-maxbytes <number>[k | m]
Set the maximum number of converted bytes (default: unlimited). k = kilobytes, m = megabytes. Example: -maxbytes 100m

-nostatistic
Suppress the display of the conversion statistic upon completion

-h or **-help**
Display a short help text about all program options.

Exit status codes:

-2 = internal error
-1 = convert not started
0 = convert terminated normally
11 = convert terminated: max number of Web pages reached
12 = convert terminated: max number of URL calls exceeded
13 = convert terminated: max number of bytes exceeded

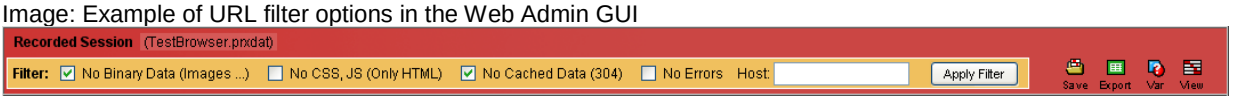
3.11.2 "Session To Java" Tool

This tool allows you to convert a Web surfing session file (*.prxdat) into a ready-to-run Java load test program (*.java). After that the load test program can be compiled by calling the "javac" compiler and then executed by using the **PrxJob** utility (see chapter 4).

This tool is commonly used to convert Web surfing sessions into load test programs which were created by the Page Scanner Session Converter tool (see previous chapter 3.11.1).

Note: if the Web surfing session has been manually post-processed in the Web Admin GUI all modifications inclusive the URL filter options and the Var Handler definitions are considered and applied.

Image: Example of URL filter options in the Web Admin GUI



Invocation:

```
java PageScannerSessionToJava <input-file>
[-tz <timezone>]
[-dgs a | c]
[-description <text>]
[-test heuristic | size | none]
[-noFilter]
[-setErrorfilter]
[-noFailureAction]
[-noStripReferer]
[-noStripAccept]
[-userThinkTime <seconds>[:<percent of random>]]
[-setBasicAuth <username>:<password>]
[-setDigestAuth <username>:<password>]
[-setNtlmAuth v1|v2|lm:<domain>:<username>:<password>]
[-setSslCertAuth <pkcs#12 file>:<password>]
[-proxyConfig <XML file>]
[-h] [-help]
```

Example:

```
java -Xmx256m PageScannerSessionToJava
proxysniffer_com_80_08Jun09_220645.prxdat -test none -noFailureAction
```

List of all supported options:

-tz <timezone>
Set a time zone for the current date and time of the server. The default time zone is ECT. See also chapter 6: Supported Time Zones.

-dgs a|c
Set the number format. a = apply apostrophe as decimal grouping separator (example: 123'456.00). c = apply comma as decimal grouping separator (example: 123,456.00). The default value is "a".

-description <text>
Set a description for the load test program.

-test heuristic | size | none

Set the algorithm which is applied to check the received content of the URL calls. “**heuristic**” means that whenever possible, the received content of the URL calls is checked by an automatically-determined text fragment. “**size**” means that the received content is checked by its size, which must not differ more than +/- 5 percent from the size received when the scan was originally executed. “**none**” means that the received content is not checked. The default value is “**heuristic**”.

-noFilter
Disable all URL filter options except that the filter for "no cached data (304)" is enabled.
Note: The filter options in the *.prxdat file are not modified.

-setErrorfilter
Enable the URL filter options "no errors" and "no cached data (304)". All other URL filter options are considered if they have been configured in the Web Admin GUI.
Note: the filter options in the *.prxdat file are not modified.

-noFailureAction
Set the failure action to "none - continue loop" for all URL calls.
Note: The failure actions in the *.prxdat file are not modified.

-noStripReferer
Don't strip the referer HTTP request header field.

-noStripAccept
Don't strip the accept HTTP request header field to */*.

-userThinkTime <seconds>[:<percent of random>]
Set a new value for the user's think time, applied for all web pages.
Note: The user’s think times in the *.prxdat file are not modified.

-setBasicAuth <username>:<password>
Modify username and password for basic authentication - or enable basic authentication.

-setDigestAuth <username>:<password>
Enable digest authentication.

-setNtlmAuth v1|v2|lm:<domain>:<username>:<password>
Enable NTLM authentication.

-setSslCertAuth <pkcs#12 file>:<password>
Enable SSL authentication by using a client certificate which is stored in a pkcs#12 file.

-proxyConfig <XML file>
Execute the load test via an outbound proxy server. The XML file contains the data of the outbound proxy. Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<NextProxyConfig>
  <isActive>true</isActive>
  <disableHttpProxyCache>true</disableHttpProxyCache>
  <httpHost>192.16.4.11</httpHost>
  <httpPort>807</httpPort>
  <httpsHost>192.16.4.11</httpsHost>
  <httpsPort>807</httpsPort>
  <authUsername>miller</authUsername>
  <authPassword>topsecret</authPassword>
  <noNextProxy>localhost</noNextProxy>
</NextProxyConfig>
```

Notes: The field “isActive” should always set to a value of true. If no proxy authentication is needed then you can let the fields “authUsername” and “authPassword” empty. The field

“noNextProxy” contains an exclusion list of IP addresses and DNS names of Web servers which are directly addressed by the load test program (separated by commas or semicolons).

`-h` or **`-help`**
Display a short help text about all program options.

Exit status codes:

-2 = internal error
-1 = convert not started
0 = convert terminated normally

3.12 Converting Recorded Sessions To and From HAR-Files

Starting from Version 5.2-D there are two Command Line Tools available that can be used to convert a ZebraTester session (*.prxdat file) to a HTTP Archive (*.har file) and vice versa.

3.12.1 ProxyDataDumpToHAR

Convert a ZebraTester session (*.prxdat file) to a HTTP Archive (*.har file).

Invocation:

```
java ProxyDataDumpToHAR <input-file> <output-file>
```

Exit Code: 0 = success, 1 = error

Example:

```
java –Xmx128m ProxyDataDumpToHAR FirstTest.prxdat FirstTest.har
```

3.12.2 HARToProxyDataDump

Convert a HTTP Archive (*.har file) to a ZebraTester session (*.prxdat file).

Invocation:

```
java HARToProxyDataDump <input-file> <output-file>
```

Exit Code: 0 = success, 1 = error

Example:

```
java –Xmx128m HARToProxyDataDump FFRecording.har FFRecording.prxdat
```


3.13 JUnit Load Test Automation Tools

Starting from Version 5.4-D (patch No. 3) there are four Command Line Tools available that support the automated generation of ZebraTester load test programs which execute Junit test cases.

Note: Basic information about JUnit load tests is available in the manual "**Executing JUnit Test Cases as ZebraTester Load Test**".

3.13.1 GenerateJUnitPluginCode

This command line tool generates Java code for a JUnit load test plug-in from a JUnit plug-in template (XML).

If you don't have yet a plug-in template you can generate it first manually following the instructions in the manual "Executing JUnit Test Cases as ZebraTester Load Test". After that you can copy and modify the template. Alternatively you can generate such templates also by an own written tool.

Invocation:

From Command Line:

```
java GenerateJUnitPluginCode <template-file> <plug-in-class-name>
<output-file>
```

From a Java Program:

```
GenerateJUnitPluginCode.execute(String templateFile, String
pluginClassName, String outputFile)
```

Exit Code: 0 = success, 1 = error | Or when call from a Java program: throws IOException

Example:

```
java GenerateJUnitPluginCode
JUnit_ClassInPackageTest_PluginTemplate.xml MyPlugin MyPlugin.java
```

This example reads the JUnit plug-in template "JUnit_ClassInPackageTest_PluginTemplate.xml" and generates the Java source code (written to "MyPlugin.java") by using the Java class name "MyPlugin".

After that you can compile the plug-in. Note that the Java CLASSPATH must contain all jar files of ZebraTester and also all jar files required to execute the Junit test. CLASSPATH Example (made in the ZebraTester Command Prompt terminal which contains already the CLASSPATH to all ZebraTester jar files):

```
set CLASSPATH=%CLASSPATH%;C:\ProgrammingLibraries\JUnit-4.11\junit-
4.11.jar;C:\ProgrammingLibraries\JUnit-4.11\hamcrest-core-
1.3.jar;C:\MyLibraries\PackageTest.jar
```


Example of a JUnit Plug-In Template (XML):

```
<?xml version="1.0" encoding="UTF-8"?>
<loadTestJUnitPluginTemplate>
  <proxySnifferVersion>V5.4-F</proxySnifferVersion>
  <jUnitClassName>MyPackage.ClassInPackageTest</jUnitClassName>
  <jUnitTestMethodVector>
    <string>secondCheck</string>
    <string>sortedSet</string>
  </jUnitTestMethodVector>
  <jUnitExecutionOrder>1</jUnitExecutionOrder>
  <combineErrorTypes>2</combineErrorTypes>
  <jUnitTestPacingSeconds>0</jUnitTestPacingSeconds>
  <systemPropertiesFileName></systemPropertiesFileName>
  <className>JUnit_ClassInPackageTest</className>
  <allowMultipleUsage>false</allowMultipleUsage>
  <pluginLabel>JUnit ClassInPackageTest</pluginLabel>
  <description></description>
  <constructScope>3</constructScope>
  <execScope>5</execScope>
  <execOrder>0</execOrder>
  <inputOptional>-1</inputOptional>
  <outputOptional>-1</outputOptional>
  <inputData>
    <loadTestPluginTemplateInputData>
      <label>Pacing Seconds (0 = no pacing)</label>
      <localVar>vPacingTime</localVar>
      <convertTo>1</convertTo>
      <isUserInputField>true</isUserInputField>
      <defaultValue>0</defaultValue>
    </loadTestPluginTemplateInputData>
    <loadTestPluginTemplateInputData>
      <label></label>
      <localVar></localVar>
      <convertTo>1</convertTo>
      <isUserInputField>false</isUserInputField>
      <defaultValue></defaultValue>
    </loadTestPluginTemplateInputData>
    <loadTestPluginTemplateInputData>
      <label></label>
      <localVar></localVar>
      <convertTo>1</convertTo>
      <isUserInputField>false</isUserInputField>
      <defaultValue></defaultValue>
    </loadTestPluginTemplateInputData>
  </inputData>
  <outputData>
    <loadTestPluginTemplateOutputData>
      <label></label>
      <localVar></localVar>
      <convertFrom>1</convertFrom>
      <defaultValue></defaultValue>
    </loadTestPluginTemplateOutputData>
  </outputData>
</loadTestJUnitPluginTemplate>
```

Do not modify

Value of XML field **jUnitExecutionOrder**:

1 = Execute the JUnit test methods in natural sequence, as read by using Java reflection.

2 = Execute the JUnit tests methods in alphabetical order (case insensitive).

3 = Execute the JUnit tests methods in random order.

4 = Execute the JUnit tests methods in parallel.

Value of XML field **combineErrorTypes**:

1 = Combine the error types by class name + method name.

2 = Combine the error types by failure text.

3 = Combine the error types by Java exception text.

4 = Combine the error types by Java stack trace.

3.13.2 CreateJUnitSession

This command line tool creates a session (*prxdat file) which contain one as non-executable marked URL and contain all external resources required to execute the JUnit test case(s).

Invocation:

From Command Line:

```
java CreateJUnitSession <project-name>
                        <author>
                        <comment>
                        <*.prxdat-output-file>
                        <external-resource-file-1>
                        ...
                        <external-resource-file-n>
```

From a Java Program:

```
CreateJUnitSession.execute(String projectName, String author, String
comment, String outputFile, ArrayList<String> externalResourceFiles)
```

Note: The project name, the author and the comment can be empty strings.

Exit Code: 0 = success, 1 = error | Or when call from a Java program: throws IOException

Example:

```
java CreateJUnitSession "My Project" "Max Meier" ""
junit_test_01.prxdat
C:\ProgrammingLibraries\JUnit-4.11\junit-4.11.jar
C:\ProgrammingLibraries\JUnit-4.11\hamcrest-core-1.3.jar
C:\MyLibraries\PackageTest.jar
```


3.13.3 CreateJUnitExternalResourcesXml

This command line tool creates from a session (*prxdat file) an XML file containing the references for all external resources.

Invocation:

From Command Line:

```
java CreateJUnitExternalResourcesXml <*.prxdat-input-file>
                                     <XML-output-file>
```

From a Java Program:

```
CreateJUnitExternalResourcesXml.execute(String sessionFile, String
xmlFile)
```

Exit Code: 0 = success, 1 = error | Or when call from a Java program: throws Exception

Note: The name of the XML file must be the same as the name of the load test program (respectively the same as the session file name), but ending with "_ExternalResources.xml".

Example:

```
java CreateJUnitExternalResourcesXml junit_test_01.prxdat
junit_test_01_ExternalResources.xml
```

3.13.4 AddJUnitPlugin

This command line tool adds a compiled "JUnit load test plug-in" to a ZebraTester session.

Invocation:

From Command Line:

```
java AddJUnitPlugin <*.prxdat-file>
                    <plugin-class-file>
```

From a Java Program:

```
AddJUnitPlugin.execute(String sessionFile, String pluginFile)
```

Exit Code: 0 = success, 1 = error | Or when call from a Java program: throws Exception

Example:

```
java AddJUnitPlugin junit_test_01.prxdat MyPlugin.class
```

3.13.5 Windows Script Example

This script generates from a “JUnit Load Test Plug-In Template” (an XML file) a ready-to-run load test program, and starts this program on the local load generator.

```
ECHO OFF
ECHO Execute JUnitAutomation.bat

REM Define input parameter
SET TEST_CASES_JAR_FILE=C:\MyLibraries\PackageTest.jar
SET PLUGIN_TEMPLATE_FILE=JUnit_ClassInPackageTest_PluginTemplate.xml

REM Define output parameter
SET PLUGIN_CLASS_NAME=MyPlugin
SET LOAD_TEST_PROGRAM_NAME=junit_test_01

REM Set the load generator used to verify the load test
SET EXECAGENT_NAME=Local Exec Agent

REM Cleanup previously generated files
del %PLUGIN_CLASS_NAME%.java
del %PLUGIN_CLASS_NAME%.class
del %LOAD_TEST_PROGRAM_NAME%.prxdat
del %LOAD_TEST_PROGRAM_NAME%_ExternalResources.xml
del %LOAD_TEST_PROGRAM_NAME%.java
del %LOAD_TEST_PROGRAM_NAME%.class
del %LOAD_TEST_PROGRAM_NAME%.zip

REM Set the ZebraTester Java CLASSPATH
SET
CLASSPATH=.;C:\Users\mutong\ZebraTester\prxsniff.jar;C:\Users\mutong\ZebraTester;C:\Users\muton
g\ZebraTester\iaik_jce_full.jar;C:\Users\mutong\ZebraTester\iaik_ssl.jar;C:\Users\mutong\ZebraT
ester\iaik_eccelerate.jar;C:\Users\mutong\ZebraTester\iaikPkcs11Provider.jar

REM Add all required external resources to the Java CLASSPATH
set CLASSPATH=%CLASSPATH%;C:\ProgrammingLibraries\JUnit-4.11\junit-
4.11.jar;C:\ProgrammingLibraries\JUnit-4.11\hamcrest-core-1.3.jar;%TEST_CASES_JAR_FILE%

REM Generate the JUnit load test plug-in from the XML template file
java GenerateJUnitPluginCode %PLUGIN_TEMPLATE_FILE% %PLUGIN_CLASS_NAME%
%PLUGIN_CLASS_NAME%.java
IF %ERRORLEVEL% NEQ 0 (
 ECHO GenerateJUnitPluginCode failed
 GOTO ABORT_END
)

REM Compile the JUnit load test plug-in
javac %PLUGIN_CLASS_NAME%.java

REM Create a ZebraTester session containing a page break and a non-executed URL, and add all
required external resources to that session
java CreateJUnitSession "My Project" "Max Meier" "" %LOAD_TEST_PROGRAM_NAME%.prxdat
C:\ProgrammingLibraries\JUnit-4.11\junit-4.11.jar C:\ProgrammingLibraries\JUnit-4.11\hamcrest-
core-1.3.jar %TEST_CASES_JAR_FILE%
IF %ERRORLEVEL% NEQ 0 (
 ECHO CreateJUnitSession failed
 GOTO ABORT_END
)

REM Create an XML file containing the references for all external resources
java CreateJUnitExternalResourcesXml %LOAD_TEST_PROGRAM_NAME%.prxdat
%LOAD_TEST_PROGRAM_NAME%_ExternalResources.xml
IF %ERRORLEVEL% NEQ 0 (
 ECHO CreateJUnitExternalResourcesXml failed
 GOTO ABORT_END
)

REM Add the JUnit load test plug-in to the ZebraTester session
java AddJUnitPlugin %LOAD_TEST_PROGRAM_NAME%.prxdat %PLUGIN_CLASS_NAME%.class
IF %ERRORLEVEL% NEQ 0 (
 ECHO AddJUnitPlugin failed
 GOTO ABORT_END
)

REM Generate the load test program
java PageScannerSessionToJava %LOAD_TEST_PROGRAM_NAME%.prxdat -tz PST
IF %ERRORLEVEL% NEQ 0 (
 ECHO PageScannerSessionToJava failed
 GOTO ABORT_END
```

```
)

REM Compile the load test program
javac %LOAD_TEST_PROGRAM_NAME%.java

REM Zip all together to have a ready-to-run load test program
java -Xmx1024m PrxJob zip %LOAD_TEST_PROGRAM_NAME%.class
%LOAD_TEST_PROGRAM_NAME%_ExternalResources.xml %PLUGIN_CLASS_NAME%.class
C:\ProgrammingLibraries\JUnit-4.11\junit-4.11.jar C:\ProgrammingLibraries\JUnit-4.11\hamcrest-
core-1.3.jar %TEST_CASES_JAR_FILE% -out %LOAD_TEST_PROGRAM_NAME%.zip
IF %ERRORLEVEL% NEQ 0 (
 ECHO zip failed
 GOTO ABORT_END
)

REM Transmit the load test program to the load generator
java PrxJob transmitJob "%EXECAGENT_NAME%" %LOAD_TEST_PROGRAM_NAME%.zip -vPacingTime "0" -u 1 -
d 60 -t 60 -sdelay 200 -maxloops 0 -sampling 15 -percpage 100 -percurl 100 -maxerrmem 20 -nolog
SET JOB_ID=%ERRORLEVEL%
IF %JOB_ID% LSS 0 (
 ECHO Error %JOB_ID%: unable to define job
 GOTO ABORT_END
)
ECHO job %JOB_ID% transmitted

REM Start the load test program on the load generator
java PrxJob startJob "%EXECAGENT_NAME%" %JOB_ID%
SET STATUS=%ERRORLEVEL%
IF %STATUS% NEQ 0 (
 ECHO Error %STATUS%: unable to start job %JOB_ID%
 GOTO ABORT_END
)
ECHO job %JOB_ID% started on "%EXECAGENT_NAME%"

EXIT /B 0

:ABORT_END
ECHO JUnitAutomation.bat failed
EXIT /B -1
```

3.14 SMTP Tool

The SMTP Tool allows to send a bulk of user-specific emails to a SMTP server. This tool is normally used to prepare tests for Web portals which support Web Mail functionality.

Invocation:

```
java SMTPTool <definition file>
              [-tz <timezone>]
              [-parseOnly]
              [-verbose]
              [<var name>=<value> .. <var name>=<value>]
```

Example:

```
java SMTPTool mailDefinition1.txt "varDate=January 15, 2010"
```

List of all supported options:

[-tz <timezone>]
Sets a time zone for the date and time values. The default time zone is ECT. See also chapter 6: Supported Time Zones. Note: the time zone can alternatively be set inside the definition file by using the command -timeZone.

[-parseOnly]
Allows to verify the data of the definition file without sending emails.

[-verbose]
Enable verbose logging.

[<var name>=<value> .. <var name>=<value>]
Allows to set one or several variables which can be used in the definition file.

Exit Value:

0: success
1: failure

3.14.1 Commands and Predefined Variables inside the Definition File

The core of the SMTP Tool are **commands** which are made in the definition file. A simple example is as follows:

```
# this line is a comment
-timeZone PST
-smtpServerName 192.16.4.31
-setvar $message1 "this is an email message"
-setvar $message2 "this is another email message"
newmail from@company.com  to@company.com  "Mail No. {$mailCounter}" {$message1}
send
newmail from@company.com  to@company.com  "Mail No. {$mailCounter}" {$message2}
send
```

Please note that some (but not all) of the commands start with a minus sign (-).

Almost all commands support to use **variables** in command arguments. The notation of variables is different when they are defined as when they are used for substitution.

$<variable name> is used for definitions
{$<variable name>} is used for substitutions

The only exceptions of this notation are variables which are passed from command line arguments of the SMTP Tool: the $ sign is not used (or rather said automatically added) in such a case to avoid conflicts with shell variables on Unix-like systems.

Some variables are already predefined and can be used directly for substitutions inside a definition file:

Predefined Variable	Description and Value
$mailCounter	This variable has an initial value of 1 (one) and is incremented by one each time after a email was transmitted to the SMTP server, independently if the transmission was successful or did fail.
$loopCounter	This variable is set to 1 (one) when the execution of the command **-sendloop** starts and is incremented by one each time after a email was transmitted by **-sendloop** to the SMTP server, independently if the transmission was successful or did fail.
$totalMails	This variable has an initial value of 0 (zero) and is incremented by one each time after a email was transmitted to the SMTP server, independently if the transmission was successful or did fail.
$passedMails	This variable has an initial value of 0 (zero) and is incremented by one each time after a email was successfully transmitted to the SMTP server. Note: this variable is not incremented when the program option **-parseOnly** is used.
$failedMails	This variable has an initial value of 0 (zero) and is incremented by one each time after a email transmission to the SMTP server did fail. Note: this variable is not incremented when the program option **-parseOnly** is used.

All of the commands can be used several times inside the same definition file except the command **-inputFile** – because currently only one input file is supported per definition file.

The number sign (#) is used as end-of-line comment delimiter inside the definition file. All command arguments which contain white spaces or contain the number sign must be enfolded by double quotes (").

The following subchapters contain a compilation of all supported commands. Command arguments which are listed in squared brackets [] are optional arguments.

3.14.1.1 -smtpServerName

Description: Defines the SMTP server to which the emails are send.

Argument	Description	Var Substitution
<server name or IP>	DNS name or TCP/IP address or the SMTP server	supported

Example:
-smtpServerName 192.16.4.30

3.14.1.2 -smtpAuthUsername

Description: Set the default user name which is used for authentication against the DNS server

Argument	Description	Var Substitution
<user name>	User name for DNS authentication	supported

Example:
-smtpAuthUsername miller

3.14.1.3 -smtpAuthPassword

Description: Set the default password which is used for authentication against the DNS server

Argument	Description	Var Substitution
<password>	Password for DNS authentication	supported

Example:
-smtpAuthPassword “topsecret 123”

3.14.1.4 -inputFile

Description: Defines a input files from which a new line is read each time before a new email is send. The values of the lines are extracted into variables which can be used as arguments for other commands.

Argument	Description	Var Substitution
<file name>	Absolute or relative path of the input file	supported
eofReopen | eofAbort	Defines what happens if no more new lines are available: • **eofReopen** means that the file is closed and opened again so that reading new lines continues from the first line. • **eofAbort** means that sending further emails is aborted when the end of the input file is reached.	not supported
trim | notrim	Defines if the values of the extracted variables are trimmed at the start and at the end.	not supported
<line comment character>	Defines the line comment character used in the input file. Note: the character # cannot be used because this is already the line comment character of the definition file.	not supported
<variable delimiter character>	Character which is used within one line to separate the values of the variables from each other.	not supported
$<variable name 1> [$<variable name 2>] … [$<variable name n>]	Allows to define a unlimited number of variables. Each variable must start with a $ (dollar sign)	not supported

Example:
-inputFile /users/miller/inputFile.txt eofReopen trim ! ; $from $A $B $C # input file with 4 variables

Note that the SMTP tool supports currently only one input file. This means that this command cannot be used several times inside the same definition file.

3.14.1.5 -setvar

Description: Allows to define a new variable or to alter the value of a already defined variable.

Argument	Description	Var Substitution
$<variable name>	Name of the variable. The name must start with a $ (dollar sign)	not supported
<variable value>	Value of the variable.	not supported

Example:
-setvar $F “hello world”

3.14.1.6 -dumpvars

Description: Dumps list of all variables. Can be used for debugging purposes

Example:
-dumpvars

3.14.1.7 -timeZone

Description: Sets a time zone for the date and time values.

Argument	Description	Var Substitution
<time zone>	Value of the time zone abbreviation. See also chapter 5	supported

Example:
-timeZone PST

3.14.1.8 -log

Description: Writes a message to the log.

Argument	Description	Var Substitution
<message>	Text of log message	supported

Example:
-log “sending message {$F}”

3.14.1.9 -prompt

Description: interrupts temporarily the execution and waits for user input.

Argument	Description	Var Substitution
<message>	Message which is written to stdout.	supported
[$<variable name>]	If this optional argument is defined, the entered text is stored inside the specified variable.	not supported

Example:
-prompt “Enter the name of the SMTP server:” $serverName

3.14.1.10 -sleep

Description: Interrupts the execution for a value of milliseconds.

Argument	Description	Var Substitution
<milliseconds>	Time in milliseconds	supported

Example:
-sleep 5000

3.14.1.11 -eof

Description: Aborts further reading of the definition file and terminates the SMTP tool.

Example:
-eof

3.14.1.12 newmail

Description: Defines a new **template** for sending emails.

Argument	Description	Var Substitution
<from>	from email address	supported
<to>	to email address	supported
<subject>	Subject of email	supported
[<ASCII text message]	Optional. Message text of email in ASCII format. Alternatively the command **textMessageFile** or **htmlMessageFile** can be used to define the message text.	supported

Example:
newmail fromme@company.com {$recipient} “Message no. {$mailCounter}” “Hello {$salutation}”

3.14.1.13 authUsername

Description: Sets for a template the user name which is used for authentication against the DNS server.

Argument	Description	Var Substitution
<user name>	User name for DNS authentication	supported

Example:
authUsername {$authUser}

3.14.1.14 authPassword

Description: Sets for a template the password which is used for authentication against the DNS server.

Argument	Description	Var Substitution
<password>	Password for DNS authentication	supported

Example:
authUsername {$authPwd}

3.14.1.15 textMessageFile

Description: Sets for a template the message text (in ASCII format) which is read from a file.

Argument	Description	Var Substitution
<file name>	Absolute or relative path of the file which contains the message text in ASCII format.	supported * * The file name cannot be replaced by a variable, but variables can be used inside the message text (inside the content of the file).

Example:
textMessageFile asciiMessage.txt

3.14.1.16 htmlMessageFile

Description: Sets for a template the message text (in HTML format) which is read from a file.

Argument	Description	Var Substitution
<file name>	Absolute or relative path of the file which contains the message text in HTML format.	supported * * The file name cannot be replaced by a variable, but variables can be used inside the message text (inside the content of the file).

Example:
htmlMessageFile htmlMessage.html

3.14.1.17 addHtmlImage

Description: Adds an image to a template which contains a HTML message. Note: the image is embedded inside the email.

Argument	Description	Var Substitution
<image file>	Absolute or relative path of the file which contains the image.	supported
<cid>	Content id of the image which must be referenced in the HTML message. Example (HTML fragment): <img src="cid:{$mailCounter}">	supported

Example:
addHtmlImage image_{$mailCounter}.gif {$mailCounter} # each e-mail contains a different image

3.14.1.18 attachFile

Description: Adds an attachment to a template.

Argument	Description	Var Substitution
<file name>	Absolute or relative path of the file which is attached to the email.	supported
[<mime type>]	Optional but recommended. Defines the data type of the attached file. Example: application/pdf Note: The MIME type is automatically determined if this argument is not passed to the command. However, if the MIME type cannot be determined, the further execution of the SMTP Tool is aborted.	supported

Example:
attachFile report.pdf

3.14.1.19 send

Description: transmits the data of a template to the SMTP server (or in general parlance: transmits the email).

Argument	Description	Var Substitution
<pacing time>	Optional. Pacing time in milliseconds. Example: the pacing time is set to 5000 ms and the email was transmitted in 450 milliseconds. In such a case the execution of the SMTP Tool is interrupted for 4500 milliseconds before the next command of the definition file will be executed.	supported

Example:
send

3.14.1.20 sendloop

Description: transmits the data of a template repeatedly to the SMTP server (or in general parlance: transmits several emails).

Argument	Description	Var Substitution
<number of iterations>	Number of emails which are transmitted to the SMTP server by using the same template.	supported
<pacing time>	Optional. Pacing time in milliseconds, applied after transmission of every email. Example: the pacing time is set to 5000 ms and a email was transmitted in 450 milliseconds. In such a case the transmission of the next email is delayed for 4500 milliseconds – or if the last email in sendloop was send – the execution of the SMTP Tool is interrupted for 4500 milliseconds before the next command of the definition file will be executed.	supported

Example:
sendloop 100 5000

3.14.2 Complex Example of Definition File – Personalized Emails

Definition File (definitionFile.txt):

```
-timeZone PST
-log "starting up"
-prompt "Enter SMTP server name or IP address: " $smtpServer
-smtpServerName {$smtpServer}
-inputFile inputFile.txt eofReopen trim ! ; $from $username $password $to $givenname
                $attachFile

# send 3 personalized emails
newmail {$from} {$to} "Mail No. {$mailCounter}"
htmlMessageFile message.html
addHtmlImage image{$loopCounter}.gif {$loopCounter}
authUsername {$username}
authPassword {$password}
attachFile {$attachFile} "application/pdf"
sendloop 3 2000

-log ""
-log "All Done"
-log "--------"
-dumpvars
-log "total {$totalMails} mails"
-log "{$passedMails} mails passed"
-log "{$failedMails} mails failed"
```

Input File (inputFile.txt):

```
! from; auth username; auth password; to; given name; attach file name
! ------------------------------------------------------------------

direct@d-fischer.com; direct; topsecret1; fischer@d-fischer.com; Sarah; doc1.pdf
fischer@d-fischer.com; fischer; topsecret2; direct@d-fischer.com; Karl; doc2.pdf
sales@d-fischer.com; sales; topsecret3; fischer@d-fischer.com; Andy; doc3.pdf
```

HTML Message File (message.html):

```
<html>
<body>
Hello {$givenname},<br>
What do you think about this image?<br>
<img src="cid:{$loopCounter}"><br>
 <br>
Please take also a look at the attached PDF document.
</body>
</html>
```

Other Files:

- image1.gif
- image2.gif
- image3.gif
- doc1.pdf
- doc2.pdf
- doc3.pdf

Verification of the Definition File (without sending emails):

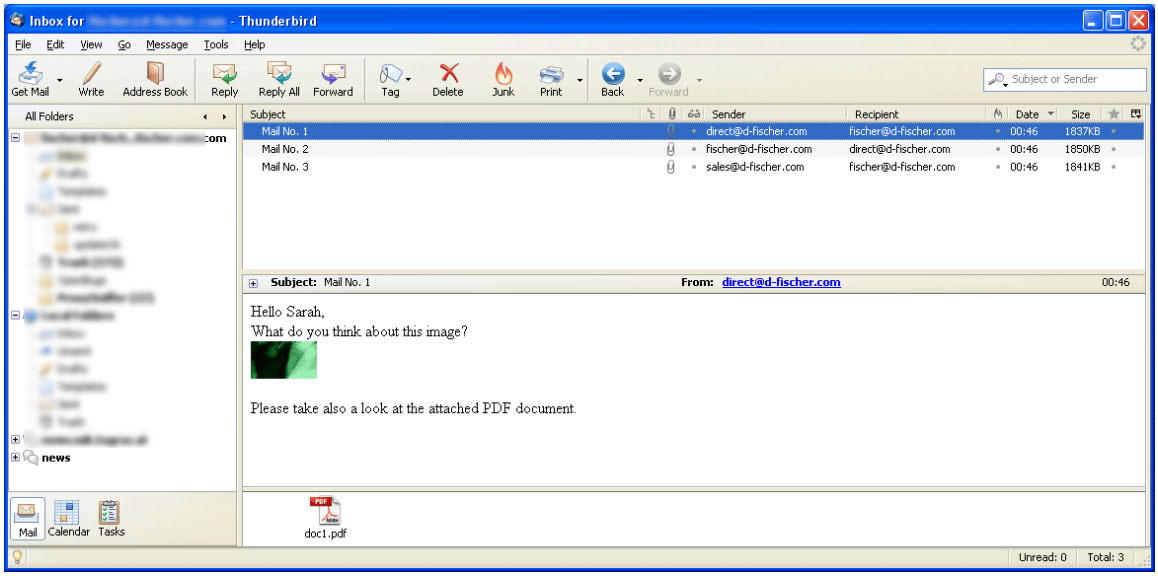
```
D:\ZebraTester>java SMTPTool definitionFile.txt -parseOnly -verbose
18 Dec 2009 16:15:46.203 main: starting up
Enter SMTP server name or IP address: 192.16.4.31
18 Dec 2009 16:15:48.875 main: --- vvv --- email data #1 --- vvv ---
18 Dec 2009 16:15:48.875 main: smtpServerName = 192.16.4.31
18 Dec 2009 16:15:48.875 main: smtpServerPort = 25
18 Dec 2009 16:15:48.875 main: smtpAuthUsername = direct
18 Dec 2009 16:15:48.875 main: smtpAuthPassword = topsecret1
18 Dec 2009 16:15:48.875 main: emailFrom = direct@d-fischer.com
18 Dec 2009 16:15:48.875 main: emailTo = fischer@d-fischer.com
18 Dec 2009 16:15:48.890 main: emailSubject = Mail No. 1
18 Dec 2009 16:15:48.890 main: emailMessage = <html>
<body>
Hello Sarah,<br>
What do you think about this image?<br>
<img src="cid:1"><br>
 <br>
Please take also a look at the attached PDF document.
</body>
</html>
18 Dec 2009 16:15:48.890 main: isHtmlMessage = true
18 Dec 2009 16:15:48.890 main: htmlImage[1] = image1.gif, cid = 1
18 Dec 2009 16:15:48.890 main: attachFile[1] = doc1.pdf, mimeType = application/pdf
18 Dec 2009 16:15:48.890 main: --- ^^^ --- email data #1 --- ^^^ ---
18 Dec 2009 16:15:48.890 main: --- vvv --- email data #2 --- vvv ---
18 Dec 2009 16:15:48.890 main: smtpServerName = 192.16.4.31
18 Dec 2009 16:15:48.890 main: smtpServerPort = 25
18 Dec 2009 16:15:48.890 main: smtpAuthUsername = fischer
18 Dec 2009 16:15:48.890 main: smtpAuthPassword = topsecret2
18 Dec 2009 16:15:48.890 main: emailFrom = fischer@d-fischer.com
18 Dec 2009 16:15:48.890 main: emailTo = direct@d-fischer.com
18 Dec 2009 16:15:48.890 main: emailSubject = Mail No. 2
18 Dec 2009 16:15:48.890 main: emailMessage = <html>
<body>
Hello Karl,<br>
What do you think about this image?<br>
<img src="cid:2"><br>
 <br>
Please take also a look at the attached PDF document.
</body>
</html>
18 Dec 2009 16:15:48.890 main: isHtmlMessage = true
18 Dec 2009 16:15:48.890 main: htmlImage[1] = image2.gif, cid = 2
18 Dec 2009 16:15:48.890 main: attachFile[1] = doc2.pdf, mimeType = application/pdf
18 Dec 2009 16:15:48.890 main: --- ^^^ --- email data #2 --- ^^^ ---
18 Dec 2009 16:15:48.890 main: --- vvv --- email data #3 --- vvv ---
18 Dec 2009 16:15:48.890 main: smtpServerName = 192.16.4.31
18 Dec 2009 16:15:48.890 main: smtpServerPort = 25
18 Dec 2009 16:15:48.890 main: smtpAuthUsername = sales
18 Dec 2009 16:15:48.890 main: smtpAuthPassword = topsecret3
18 Dec 2009 16:15:48.890 main: emailFrom = sales@d-fischer.com
18 Dec 2009 16:15:48.890 main: emailTo = fischer@d-fischer.com
18 Dec 2009 16:15:48.890 main: emailSubject = Mail No. 3
18 Dec 2009 16:15:48.890 main: emailMessage = <html>
<body>
Hello Andy,<br>
What do you think about this image?<br>
<img src="cid:3"><br>
 <br>
Please take also a look at the attached PDF document.
</body>
</html>
18 Dec 2009 16:15:48.890 main: isHtmlMessage = true
18 Dec 2009 16:15:48.890 main: htmlImage[1] = image3.gif, cid = 3
18 Dec 2009 16:15:48.890 main: attachFile[1] = doc3.pdf, mimeType = application/pdf
18 Dec 2009 16:15:48.890 main: --- ^^^ --- email data #3 --- ^^^ ---
18 Dec 2009 16:15:48.906 main:
18 Dec 2009 16:15:48.906 main: All Done
18 Dec 2009 16:15:48.906 main: --------
18 Dec 2009 16:15:48.906 main: --- vvv --- list of all variables --- vvv ---
```

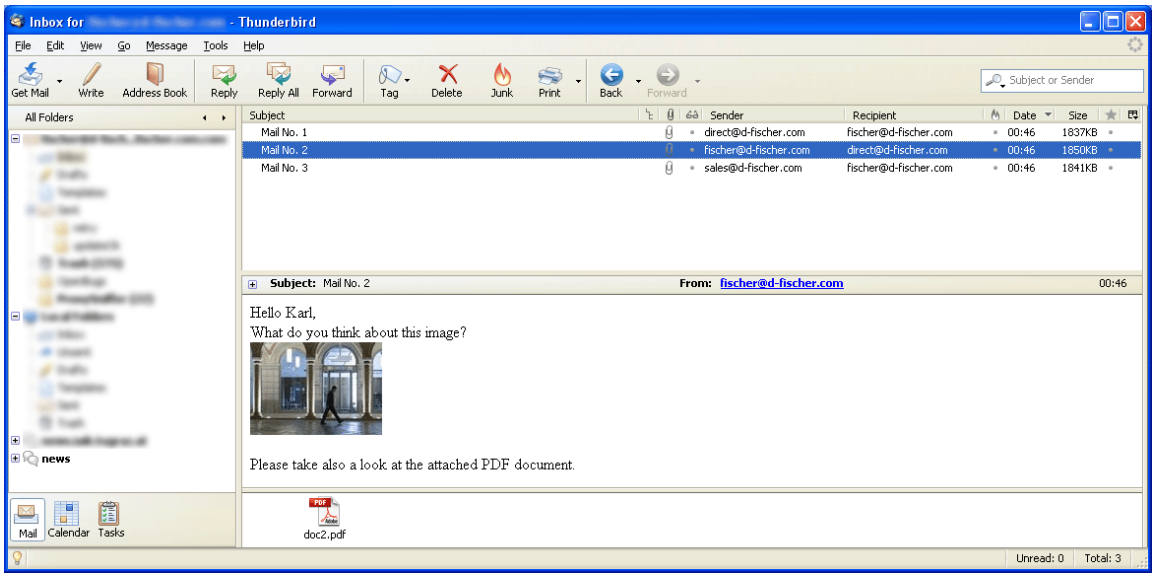
```
18 Dec 2009 16:15:48.906 main: $givenname = Sarah
18 Dec 2009 16:15:48.906 main: $username = direct
18 Dec 2009 16:15:48.906 main: $attachFile = doc1.pdf
18 Dec 2009 16:15:48.906 main: $passedMails = 0
18 Dec 2009 16:15:48.906 main: $totalMails = 3
18 Dec 2009 16:15:48.906 main: $smtpServer = 192.16.4.31
18 Dec 2009 16:15:48.906 main: $password = topsecret1
18 Dec 2009 16:15:48.906 main: $to = fischer@d-fischer.com
18 Dec 2009 16:15:48.906 main: $from = direct@d-fischer.com
18 Dec 2009 16:15:48.906 main: $loopCounter = 3
18 Dec 2009 16:15:48.906 main: $failedMails = 0
18 Dec 2009 16:15:48.906 main: $mailCounter = 4
18 Dec 2009 16:15:48.906 main: --- ^^^ --- list of all variables --- ^^^ ---
18 Dec 2009 16:15:48.906 main: total 3 mails
18 Dec 2009 16:15:48.906 main: 0 mails passed
18 Dec 2009 16:15:48.906 main: 0 mails failed
```

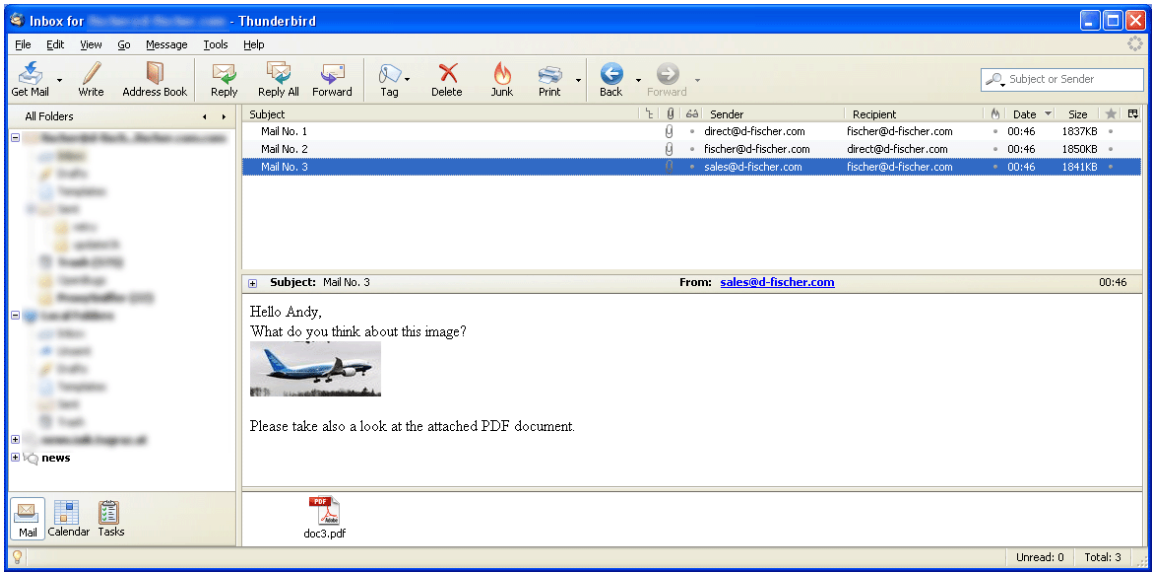
Sending the Emails:

```
D:\ ZebraTester>java SMTPTool definitionFile.txt
18 Dec 2009 15:46:22.312 main: starting up
Enter SMTP server name or IP address: 192.16.4.31
18 Dec 2009 15:46:26.609 main: Sending email to fischer@d-fischer.com ...
18 Dec 2009 15:46:26.609 main: ... waiting for send completion ...
18 Dec 2009 15:46:27.515 main: ... send completed, transmission time = 908 ms, status
                = succeeded
18 Dec 2009 15:46:28.609 main: Sending email to direct@d-fischer.com ...
18 Dec 2009 15:46:28.609 main: ... waiting for send completion ...
18 Dec 2009 15:46:29.312 main: ... send completed, transmission time = 702 ms, status
                = succeeded
18 Dec 2009 15:46:30.625 main: Sending email to fischer@d-fischer.com ...
18 Dec 2009 15:46:30.625 main: ... waiting for send completion ...
18 Dec 2009 15:46:31.312 main: ... send completed, transmission time = 689 ms, status
                = succeeded
18 Dec 2009 15:46:32.625 main:
18 Dec 2009 15:46:32.625 main: All Done
18 Dec 2009 15:46:32.625 main: --------
18 Dec 2009 15:46:32.625 main: --- vvv --- list of all variables --- vvv ---
18 Dec 2009 15:46:32.625 main: $givenname = Sarah
18 Dec 2009 15:46:32.625 main: $username = direct
18 Dec 2009 15:46:32.625 main: $attachFile = doc1.pdf
18 Dec 2009 15:46:32.625 main: $passedMails = 3
18 Dec 2009 15:46:32.625 main: $totalMails = 3
18 Dec 2009 15:46:32.625 main: $smtpServer = 192.16.4.31
18 Dec 2009 15:46:32.625 main: $password = topsecret1
18 Dec 2009 15:46:32.625 main: $to = fischer@d-fischer.com
18 Dec 2009 15:46:32.625 main: $from = direct@d-fischer.com
18 Dec 2009 15:46:32.625 main: $loopCounter = 3
18 Dec 2009 15:46:32.625 main: $failedMails = 0
18 Dec 2009 15:46:32.625 main: $mailCounter = 4
18 Dec 2009 15:46:32.625 main: --- ^^^ --- list of all variables --- ^^^ ---
18 Dec 2009 15:46:32.625 main: total 3 mails
18 Dec 2009 15:46:32.625 main: 3 mails passed
18 Dec 2009 15:46:32.625 main: 0 mails failed

D:\ ZebraTester>echo %ERRORLEVEL%
0
```


Result / Received Emails:







4 Scripting Load Test Program Execution

4.1 PdfReport Utility

In combination with scripting the execution of load test programs from the command line by using the **PrxJob** utility (see chapter 4.4 in this document), the generation of PDF reports is also supported from the command line. The PdfReport utility requires that a Proxy Sniffer and a Web Admin process are up and running on the machine calling the PdfReport utility. The PdfReport utility uses the cache and internal functionality of the Web Admin process.

Hint: when starting a load test program, you can override the name of the binary result file (*.prxres) with the program argument -res <filename> (see chapter 3.8). It may be useful to know in advance the result file name which is used as input for this utility.

Invocation:

```
java PdfReport clear
```

Clears the internal Web Admin cache. This means also that all entries in the Analyse Load Tests menu are cleared.

```
java PdfReport load <*.prxres file>
```

Loads a binary result file into the internal Web Admin cache.

```
java PdfReport loadWithDetailReport <*.prxres file>
[-reportTemplate <name>] [-annotation <string>] [-conductedBy <string>]
[-output <PDF output file name>] [-comment:<name> <value>]
```

Loads a binary result file into the internal Web Admin cache, and creates the PDF detail report inside the default directory. If no output file name is specified, the PDF detail report is stored using the same file name as the binary result file, but with the file extension: *.pdf (instead of *.prxres).

```
java PdfReport generateComparisonReport <PDF output file name>
[-reportTemplate <name>] [-conductedBy <string>]
[-comment:<name> <value>]
```

Creates a PDF comparison report by using all loaded binary result files.

```
java PdfReport generateLoadCurvesReport <PDF output file name>
[-reportTemplate <name>] [-conductedBy <string>]
[-comment:<name> <value>]
```

Creates a PDF load curves report by using all loaded binary result files. Note that all loaded binary result files must be results from the same load test program, executed several times, each time with a different number of concurrent users.

```
java PdfReport listtemplates
```

Displays a list of all available report templates.

Hint: it is also supported do define customized report templates and/or to fill in comments in PDF reports by using one or several times the PdfReport option **`-comment:<name> <value>`** .

We recommend that you take a look on the **PDF Report Templates Handbook**.

4.2 PrxTestResultToXml Utility

The PrxTestResultToXml utility extracts the most important measurement result data from a binary result file (*.prxres file) and converts them into XML data format.

Invocation:

```
java PrxTestResultToXml <*.prxres file>  [-tz <timezone>]
                                         [-annotation <text>]
                                         [-outfile [<XML file name>]]
```

Example:

```
java PrxTestResultToXml Test01_c1_25Sep09_120555_4000u.prxres -tz PNT
-outfile
```

List of all supported options:

[-tz <timezone>]
Sets a time zone for the date and time values of the XML result data. The default time zone is ECT. See also chapter 6: Supported Time Zones.

[-annotation <text>]
Allows to set or override the annotation which is transferred to the XML data. By default the annotation stored inside the binary result data (*.prxres file) is used.

[-outfile [<XML file name>]]
If this option is not set the XML data are written to **stdout**. If no <XML file name> is specified – but if the option **-outfile** is used – the XML data are written to the default directory using the same file name as the binary result file (*.prxres), but with the file extension: *.xml.

Example of XML Output:

```
<?xml version="1.0" encoding="UTF-8"?>
<prxTestResult>
  <proxySnifferVersion>V5.0-E</proxySnifferVersion>
  <loadTest>TEST_02</loadTest>
  <startDate>18 Jun 2012 10:42:09 (ECT)</startDate>
  <testDuration>2:41 min</testDuration>
  <annotation></annotation>
  <testStartDateTimeStamp>1340008929640</testStartDateTimeStamp>
  <testEndDateTimeStamp>1340009090859</testEndDateTimeStamp>
  <testInputData>
    <concurrentUser>6</concurrentUser>
    <startupDelayPerUserMs>200</startupDelayPerUserMs>
  </testInputData>
  <testResult>
    <urlCallsPerSecond>12.75</urlCallsPerSecond>
    <averageSessionTimePerLoopSeconds>16.16</averageSessionTimePerLoopSeconds>
    <averageResponseTimePerPageSeconds>3.39</averageResponseTimePerPageSeconds>
    <sessionFailureRatePercent>1.89</sessionFailureRatePercent>
    <urlErrorRatePercent>0.19</urlErrorRatePercent>
    <passedLoops>52</passedLoops>
    <failedLoops>1</failedLoops>
    <passedUrlCalls>2055</passedUrlCalls>
    <failedUrlCalls>4</failedUrlCalls>
    <httpKeepAliveEfficiencyPercent>0.00</httpKeepAliveEfficiencyPercent>
    <sslSessionCacheEfficiencyPercent></sslSessionCacheEfficiencyPercent>
    <averageTcpSocketConnectTimeMs>4</averageTcpSocketConnectTimeMs>
    <averageNetworkThroughputMbit>1.54</averageNetworkThroughputMbit>
    <totalTransmittedBytesMb>29.63</totalTransmittedBytesMb>
    <detailData>
      <dataRecord>
        <index>0</index>
        <dataRecordType>0</dataRecordType>
        <infoText>Page #1: Start Page</infoText>
```

```
        <maxAcceptableResponseTime>-1</maxAcceptableResponseTime>
        <averageResponseTime>4251</averageResponseTime>
        <passedCount>53</passedCount>
        <failedCount>0</failedCount>
        <pageBreakAverageDelay>0</pageBreakAverageDelay>
        <urlAverageNetworkEstablishTime>-1</urlAverageNetworkEstablishTime>
        <urlAverageRequestTransmitTime>-1</urlAverageRequestTransmitTime>
        <urlAverageResponseContentReceiveTime>-1</urlAverageResponseContentReceiveTime>
        <urlAverageResponseHeaderReceiveTime>-1</urlAverageResponseHeaderReceiveTime>
        <urlAverageResponseHeaderWaitTime>-1</urlAverageResponseHeaderWaitTime>
        <urlAverageSize>0</urlAverageSize>
        <urlPercentile90Time>15731</urlPercentile90Time>
        <urlExecutionMode>-1</urlExecutionMode>
        <urlFailureActionType>2</urlFailureActionType>
      </dataRecord>
      <dataRecord>
        <index>1</index>
        <dataRecordType>1</dataRecordType>
        <infoText>GET http://www.proxy-sniffer.com:80/</infoText>
        <maxAcceptableResponseTime>-1</maxAcceptableResponseTime>
        <averageResponseTime>161</averageResponseTime>
        <passedCount>53</passedCount>
        <failedCount>0</failedCount>
        <pageBreakAverageDelay>-1</pageBreakAverageDelay>
        <urlAverageNetworkEstablishTime>1</urlAverageNetworkEstablishTime>
        <urlAverageRequestTransmitTime>0</urlAverageRequestTransmitTime>
        <urlAverageResponseContentReceiveTime>158</urlAverageResponseContentReceiveTime>
        <urlAverageResponseHeaderReceiveTime>0</urlAverageResponseHeaderReceiveTime>
        <urlAverageResponseHeaderWaitTime>2</urlAverageResponseHeaderWaitTime>
        <urlAverageSize>12920</urlAverageSize>
        <urlPercentile90Time>221</urlPercentile90Time>
        <urlExecutionMode>1</urlExecutionMode>
        <urlFirstReceivedContentType>TEXT/HTML</urlFirstReceivedContentType>
        <urlFailureActionType>2</urlFailureActionType>
      </dataRecord>
      <dataRecord>
        <index>2</index>
        <dataRecordType>1</dataRecordType>
        <infoText>GET http://www.proxy-sniffer.com:80/ProxySnifferLogo.gif</infoText>
        <maxAcceptableResponseTime>-1</maxAcceptableResponseTime>
        <averageResponseTime>1010</averageResponseTime>
        <passedCount>53</passedCount>
        <failedCount>0</failedCount>
        <pageBreakAverageDelay>-1</pageBreakAverageDelay>
        <urlAverageNetworkEstablishTime>2</urlAverageNetworkEstablishTime>
        <urlAverageRequestTransmitTime>1</urlAverageRequestTransmitTime>
        <urlAverageResponseContentReceiveTime>1007</urlAverageResponseContentReceiveTime>
        <urlAverageResponseHeaderReceiveTime>0</urlAverageResponseHeaderReceiveTime>
        <urlAverageResponseHeaderWaitTime>0</urlAverageResponseHeaderWaitTime>
        <urlAverageSize>6235</urlAverageSize>
        <urlPercentile90Time>5113</urlPercentile90Time>
        <urlExecutionMode>2</urlExecutionMode>
        <urlFirstReceivedContentType>IMAGE/GIF</urlFirstReceivedContentType>
        <urlFailureActionType>1</urlFailureActionType>
      </dataRecord>
      ...
      ...
    </detailData>
  </testResult>
</prxTestResult>
```

Special Data Fields:

- dataRecordType: 0 = page break, 1 = URL call
- urlExecutionMode: 1 = serial executed, 2 = parallel executed
- urlFailureActionType: 1 = continue loop (yellow error), 2 = abort loop (red error)

Note for all numeric data fields: a negative value of -1 (minus one) means "no data available".

4.3 PrxCombineTestResults Utility

The PrxCombineTestResults utility allows you to combine several load test results (*.prxres files) of cluster jobs to one load test result. Hoverer this make only sense if the cluster jobs did run in parallel at the same time.

Invocation:

```
java PrxCombineTestResults  <*.prxres file 1> .. <*.prxres file n>
                            [-addDiscreteResults]
                            -result <*.prxres result file>
```

Example:

```
java –Xmx1024m PrxCombineTestResults
TRANSACTION_04_C-Strawberry_08Mar16_004007_10u.prxres
TRANSACTION_04_C-Strawberry_08Mar16_004016_10u.prxres
-result TRANSACTION_04_C-Strawberry_08Mar16_20u.prxres
```

List of all supported options:

[-addDiscreteResults]
Append the discrete results of the cluster members to the combined data. Using this option is normally not needed. Note that the result file may be become very large if this option is enabled.

4.4 PrxJob Utility

The **PrxJob** utility allows you to use the ZebraTester infrastructure from Windows scripts (.bat) as well as from Unix scripts (.sh, .ksh, .csh …). Several load test programs can be started as jobs at the same time or in sequence, on local or remote Exec Agents, as well as on Exec Agent clusters. Synchronizing jobs and acquiring their result files is also supported.

The usage of the **PrxJob** utility requires, on the local host where the utility is called only, that a Proxy Sniffer Process and a Web Admin process be running. If cluster jobs are executed, a local Job Controller process must also have been started.

Exec Agent Job related commands

transmitJob (1)	**<exec agent name> [-pmaTemplate <template name>] <load test> <load test arguments>**
setJobScheduleTime	**<exec agent name> <job id> <number of seconds in future>**
startJob	**<exec agent name> <job id>**
addRealTimeComment	**<exec agent name> <job id> <comment>**
getJobExecutorsAnnotation	**<exec agent name> <job id> [-outfile <file name>]**
setJobExecutorsAnnotation	**<exec agent name> <job id> <text>**
getJobJavaSystemProperties	**<exec agent name> <job id> <key select pattern> [-outfile <file name>]**
setJobJavaSystemProperties	**<exec agent name> <job id> <key=value> [.. <key=value>]**
getJobState (2)	**<exec agent name> <job id>**
getJobRealTimeData	**<exec agent name> <job id>**
changeJobNumSimulatedUser (7)	**<exec agent name> <job id> <delta number> [<startup delay ms>]**
setJobSuspend	**<exec agent name> <job id> <"true" | "false"> [<startup delay ms>]**
isJobSuspend (8)	**<exec agent name> <job id>**
changeJobTestDuration	**<exec agent name> <job id> <delta seconds>**
getJobRealTimeUserInputFields	**<exec agent name> <job id>**
abortJob	**<exec agent name> <job id>**
waitForJobCompletion	**<exec agent name> <job id 1> [.. <job id n>]**
acquireJobResultFile	**<exec agent name> <job id> <result file name>**
acquireJobOutputFile	**<exec agent name> <job id> <output file name>**
acquireJobErrorFile	**<exec agent name> <job id> <error file name>**
downloadJobFile	**<exec agent name> <job id> <remote file name> [<local file name>]**
deleteJob	**<exec agent name> <job id>**
deleteAllJobs	**<exec agent name>**
getJobList (6)	**<exec agent name> <job state> [-verbose] [-outfile <file name>]**

Cluster Job related commands

```
transmitClusterJob (1)           <cluster name> [-pmaTemplate <template name>]
                                 <load test> <load test arguments>

setClusterJobScheduleTime        <cluster name> <cluster job id>
                                 <number of seconds in future>
                                 [-split <input file> <line comment tag>]

startClusterJob                  <cluster name> <cluster job id>
                                 [-split <input file> <line comment tag>]

getClusterJobStartStatistics     <cluster name> <cluster job id> [-outfile <file name>]

addClusterRealTimeComment        <cluster name> <cluster job id> <comment>

getClusterJobExecutorsAnnotation <cluster name> <cluster job id> [-outfile <file name>]

setClusterJobExecutorsAnnotation <cluster name> <cluster job id> <text>

getClusterJobJavaSystemProperties <cluster name> <cluster job id> <key select pattern>
                                  [-outfile <file name>]

setClusterJobJavaSystemProperties <cluster name> <cluster job id> <key=value>
                                  [.. <key=value>]

getClusterJobState (2)           <cluster name> <cluster job id>

getClusterJobRealTimeData        <cluster name> <cluster job id> [-detailed]

changeClusterJobNumSimulatedUser (7) <cluster name> <cluster job id> <delta number>
                                     [<startup delay ms>]

setClusterJobSuspend (9)         <cluster name> <cluster job id> <"false"|"true">
                                 [<startup delay ms>]

isClusterJobSuspend (10)         <cluster name> <cluster job id>

changeClusterJobTestDuration (11)   <cluster name> <cluster job id> <delta seconds>

getClusterJobRealTimeUserInputFields <cluster name> <cluster job id>

abortClusterJob                  <cluster name> <cluster job id>

waitForClusterJobCompletion      <cluster name> <cluster job id 1>
                                 [.. <cluster job id n>]

acquireClusterJobResultFile      <cluster name> <cluster job id> <result file name>

combineClusterFile               <cluster name> <cluster job id> <file pattern>
                                 <result file name>

downloadClusterJobFile           <cluster name> <cluster job id> <remote file name>
                                 [<local file name>]

getClusterJobExecAgentCount (3)  <cluster name> <cluster job id>

getClusterJobExecAgentName       <cluster name> <cluster job id> <cluster member number>
                                 [-outfile <file name>]

getClusterJobExecAgentJobId (1)  <cluster name> <cluster job id> <cluster member number>

deleteClusterJob                 <cluster name> <cluster job id>

deleteAllClusterJobs             <cluster name>

getClusterJobList (6)            (<cluster name> | \"-all\") <job state> [-verbose]
                                 [-outfile <file name>]
```

Exec Agent Utility commands

getExecAgentList	**[-triggerReload] [-pingVersion] [-outfile <file name>]**
pingExecAgent	**<exec agent name>**
getExecAgentLog	**<exec agent name> [-outfile <file name>]**
checkTcpPort	**<exec agent name> <host name | ip address> <port number> <timeout seconds> [-outfile <file name>]**
checkSslPort	**<exec agent name> <host name | ip address> <port number> <timeout seconds> [-outfile <file name>]**
translateDnsName	**<exec agent name> <dns name> <timeout seconds> [-outfile <file name>]**
checkUrl	**<exec agent name> <url> [<args 1> … <args n>]**
getExecAgentDefaultDirectory	**<exec agent name> [-outfile <file name>]**
getExecAgentJobsDirectory	**<exec agent name> [-outfile <file name>]**
getExecAgentJobDirectory	**<exec agent name> <job id> [-outfile <file name>]**
getDirectoryFileList	**<exec agent name> <remote directory> [-includeDirectories] [-outfile <file name>]**
fileExists	**<exec agent name> <remote file | remote directory>**
createDirectory (4)	**<exec agent name> remote directory>**
deleteFile (4)	**<exec agent name> <remote file | empty remote directory>**
getOsFilePathSeparator	**<exec agent name> [-outfile <file name>]**
uploadFile (4)(5)	**<exec agent name> <local file> <remote file>**
downloadFile (4)(5)	**<exec agent name> <local file> <remote file>**
uploadFileToJobDirectory (4)(5)	**<exec agent name> <job id> <local file> <remote file name>**
downloadFileFromJobDirectory (4)(5)	**<exec agent name> <job id> <local file> <remote file name>**
getOsType	**<exec agent name> [-outfile <file name>]**
execOsCommand (4)	**<exec agent name> <timeout seconds> <os command> [<arg 1>..<arg n>] [-remdir <remote directory>] [-setosenv <variable name>=<value>] [-infile <file name>] [-outfile <file name>] [-debug]**

Cluster Utility commands

```
getClusterList                      [-outfile <file name>]

getClusterExecAgentList             <cluster name> [-pingVersion] [-outfile <file name>]

pingCluster                         <cluster name>

getClusterExecAgentNames            <cluster name> [-outfile <file name>]

execOsCommandCluster  (4)           <cluster name> <timeout seconds>
                                    <os command> [<arg 1>..<arg n>]
                                    [-remdir <remote directory>]
                                    [-setosenv <variable name>=<value>]
                                    [-infile <file name>] [-outfile <file name>]
```

Other commands

```
sleep <seconds>

version [<exec agent name>]

zip <file 1> [, <file 2> … , <file n>] -out <zip-archive-file>

unzip <zip-file> [-select <plain-file-name>] [-outdir <target-directory-path>] [-listonly]

importNetworkConfiguration <import file>

getApicaPMAMonitoringTemplateList [-outfile <file name>]

getGuiLicenseKey [-outfile <file name>]

help
```

Special Exit Codes:

```
(1) exit code = job id / cluster job id (>0)
(2) exit code: 10 = job defined, 11 = running, 12 = completed, 13 = unknown,
               14 = scheduled
(3) exit code = number of active cluster members (max. exec agent number)
(4) the exec agent must be started with the option "-enableRemoteOsCommands"
(5) max file size limited to 128 MB
(6) supported input values for job state are: "defined", "scheduled", "running",
                                              "completed", "unknown" - or "all"
(7) exit code = the number of accepted users that are pending to remove or to add
(8) exit code: 1 = suspended, 0 = not suspended
(9) exit code: the number of cluster members for which a change of the state has
               failed. 0 = success
(10) exit code: the number of cluster members whose corresponding load test jobs are
                NOT suspended
(11) exit code: the number of cluster members for which a change of the test duration
                has failed. 0 = success
```

Note 1: adding **-ps** before any command effects that the value of the exit code is returned as a positive number instead of a negative number (useful on UNIX-like systems)

Note 2: adding **-s** before any command effects to write the exit code additionally to a file named 'PRXSTAT'

Note 3: adding **-sf <file-name>** before any command effects to write the exit code additionally to an arbitrary file

Note 4: adding **-tz <time-zone>** before any command effects to set a specific time zone

Note 5: adding **-outboundipfile <file-name>** to any command effects that multiple outbound TCP/IP addresses are read from a configuration file and used to communicate with the Exec Agents in such a way that TCP/IP load balancing is applied for outbound TCP/IP connections that are established from the WebAdmin and the JobController component of ProxySniffer to the Exec Agents. The configuration file can contain

multiple IP addresses on the same line or on several lines. Multiple IP addresses on the same line must be separated by space characters, or tab characters, or by commas, or by semicolons. The hash character can be used as a marker for comments within any position in a line.

PrxJob: Exit Codes:

```
<0 = failure
 0 = success

Error Codes:
-84: procedure partially failed
-85: procedure failed
-86: timeout expired
-87: remote execution failed
-88: insufficient license on exec agent
-89: no job output file
-90: write to file failed
-91: wrong job state
-92: invalid job id
-93: invalid extension of file name
-94: file not found
-95: exec agent not reachable
-96: invalid exec agent name
-97: invalid cluster name
-98: required argument missing
-99: no local job controller process running
-100: no local web admin process running
-101: internal error
-102: invalid command
-103: invalid argument
-104: invalid argument value
-105: argument value missing
-110: url call failed
-111: wrong http status code
-112: wrong http response header
-113: wrong response content
```

Important note for Unix-like systems:

Some shells of Unix-like systems are not able deal with negative values of exit codes – or with exit code values which are greater than 128. As a workaround you can use the **-s** or **-sf <file name>** option as additional (first) parameter for the PrxJob command. In such a case the exit code is stored in a file (see note 2 and note 3 on previous page).

Mac OS X example (bash shell):

```
# transmit load test job to exec agent
java PrxJob -s transmitJob "$execAgentName" $loadTestProgram1 $loadTestProgram1Args
prxstat=`cat PRXSTAT`
if [ $prxstat -lt "0" ]; then
  echo "unable to transmit job, status = $prxstat"
  exit 1;
fi
jobId=$prxstat
```

Alternatively you can use the **-ps** option (as a first argument, before -s or -sf <file name> is optionally used) which will turn negative values of exit codes to positive values (see note 1 on previous page).

Example of getting the Exit Code:

```
java PrxJob deleteJob "Local Exec Agent" 551
```

With Windows scripts (.bat), you can retrieve the exit code with "ECHO **%ERRORLEVEL%**"
With Unix scripts (.sh, .ksh, .csh …), you can retrieve the exit code with "echo **$?**"

4.4.1 Exec Agent Job Related Commands

4.4.1.1 transmitJob

Transmits a load test program to a local or remote Exec Agent without starting it. The corresponding job on the Exec Agent is created and is then in the state “defined”.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the “Exec Agent Network Configuration” menu (shown as “description”)
[-pmaTemplate <template name>]	Optional. If this argument is used external measured performance data are added to the load test result, collected by an Apica Performance Monitoring Agent Controller (PMA). The <template name> must referrer a valid PMA template name as defined in the ZebraTester GUI (WebAdmin). See also: command getApicaPMAMonitoringTemplateList.
<load test>	Local or absolute path of the load test program (*.class or *.zip).
<load test arguments>	All arguments of the load test program (see chapter 3.8) Note: if the (optional) argument -annotation is used, it must be the last argument

Success exit code: job id (> 0)

Example 1:
java PrxJob transmitJob "Local Exec Agent" Test01.zip -u 3 -d 30 -t 60 -nolog -annotation "first test run"

Example 2:
java PrxJob transmitJob "Local Exec Agent" -pmaTemplate TEMPLATE_02.xml Test01.zip -u 3 -d 30 -t 60 -nolog -annotation "first test run"

4.4.1.2 setJobScheduleTime

Schedules a job to be automatically started in future. Requirement: the corresponding job must be already created by using **transmitJob**. The job changes the state from “defined” to “scheduled”.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the “Exec Agent Network Configuration” menu (shown as “description”)
<job id>	Exec Agent job id Note: the job id is only unique for each Exec Agent, not across all Exec Agents
<number of seconds in future>	Number of seconds in future, when the job will be automatically started. A value of -1 (minus one) effects that an already defined schedule will be cancelled.

Success exit code: 0

Example:
java PrxJob setJobScheduleTime "Local Exec Agent" 445 600

(start the job no. 445 in 10 minutes = 600 seconds)

4.4.1.3 startJob

Starts a job on a local or remote Exec Agent. The job must be in the state “defined”, and will then change state to “running”. This command exits immediately after the job has been started – you can use the command **getJobState** or **waitForJobCompletion** to detect when a job has been completed.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the “Exec Agent Network Configuration” menu (shown as “description”)
<job id>	Exec Agent job id Note: the job id is only unique for each Exec Agent, not across all Exec Agents

Success exit code: 0

Example:
java PrxJob startJob "Local Exec Agent" 445

4.4.1.4 addRealTimeComment

Add a real time comment to a running job. The real time comment is only added if the job is in the state “job running”.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the “Exec Agent Network Configuration” menu (shown as “description”)
<job id>	Exec Agent job id Note: the job id is only unique for each Exec Agent, not across all Exec Agents
<comment>	The real time comment

Success exit code: 0

Example:
java PrxJob addRealTimeComment "Local Exec Agent" 445 “failover event triggered”

4.4.1.5 getJobExecutorsAnnotation

Get the "executor's annotation" of a running job. The "executor's annotation" is only returned if the job is in the state “job running”.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the “Exec Agent Network Configuration” menu (shown as “description”)
<job id>	Exec Agent job id Note: the job id is only unique for each Exec Agent, not across all Exec Agents
[-outfile <file name>]	Optional. If this argument is used the output is written to an arbitrary file instead to stdout.

Success exit code: 0

Example:
java PrxJob getJobExecutorsAnnotation "Local Exec Agent" 445

4.4.1.6 setJobExecutorsAnnotation

Set a new/updated "executor's annotation" for a running job. The "executor's annotation" is only updated if the job is in the state “job running”.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the “Exec Agent Network Configuration” menu (shown as “description”)
<job id>	Exec Agent job id Note: the job id is only unique for each Exec Agent, not across all Exec Agents
<text>	The executor's annotation

Success exit code: 0

Example:
java PrxJob setJobExecutorsAnnotation "Local Exec Agent" 445 “verification test after tuning web app”

4.4.1.7 getJobJavaSystemProperties

Get the Java system properties of a running job. The system properties are only returned if the job is in the state “job running”.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the “Exec Agent Network Configuration” menu (shown as “description”)
<job id>	Exec Agent job id Note: the job id is only unique for each Exec Agent, not across all Exec Agents
<key select pattern>	A text-pattern that is used to select/filter Java system properties keys. The text-pattern can contain the wildcards '*' (multiple chars) and '?' (single char). If this parameter is an empty string then all available Java system properties are returned.
[-outfile <file name>]	Optional. If this argument is used the output is written to an arbitrary file instead to stdout.

Success exit code: 0

Example:
java PrxJob getJobJavaSystemProperties "Local Exec Agent" 4 "java.awt.*"

Output:
java.awt.graphicsenv=sun.awt.Win32GraphicsEnvironment
java.awt.printerjob=sun.awt.windows.WPrinterJob

4.4.1.8 setJobJavaSystemProperties

Update, add or remove Java system properties for or a running job. The system properties are only updated, added or removed if the job is in the state “job running”.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the “Exec Agent Network Configuration” menu (shown as “description”)
<job id>	Exec Agent job id Note: the job id is only unique for each Exec Agent, not across all Exec Agents
<key=value> [<key=value> .. <key=value>]	A pair of key and value, separated by an equal sign. Multiple arguments of key=value pairs are supported to set multiple system properties at once. If an argument contains no equal sign then the corresponding system propertie is removed.

Success exit code: 0

Example:
java PrxJob setJobJavaSystemProperties "Local Exec Agent" 3 "xx4=5" "xx5=6"

4.4.1.9 getJobState

Returns the current state of an Exec Agent job.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the “Exec Agent Network Configuration” menu (shown as “description”)
<job id>	Exec Agent job id

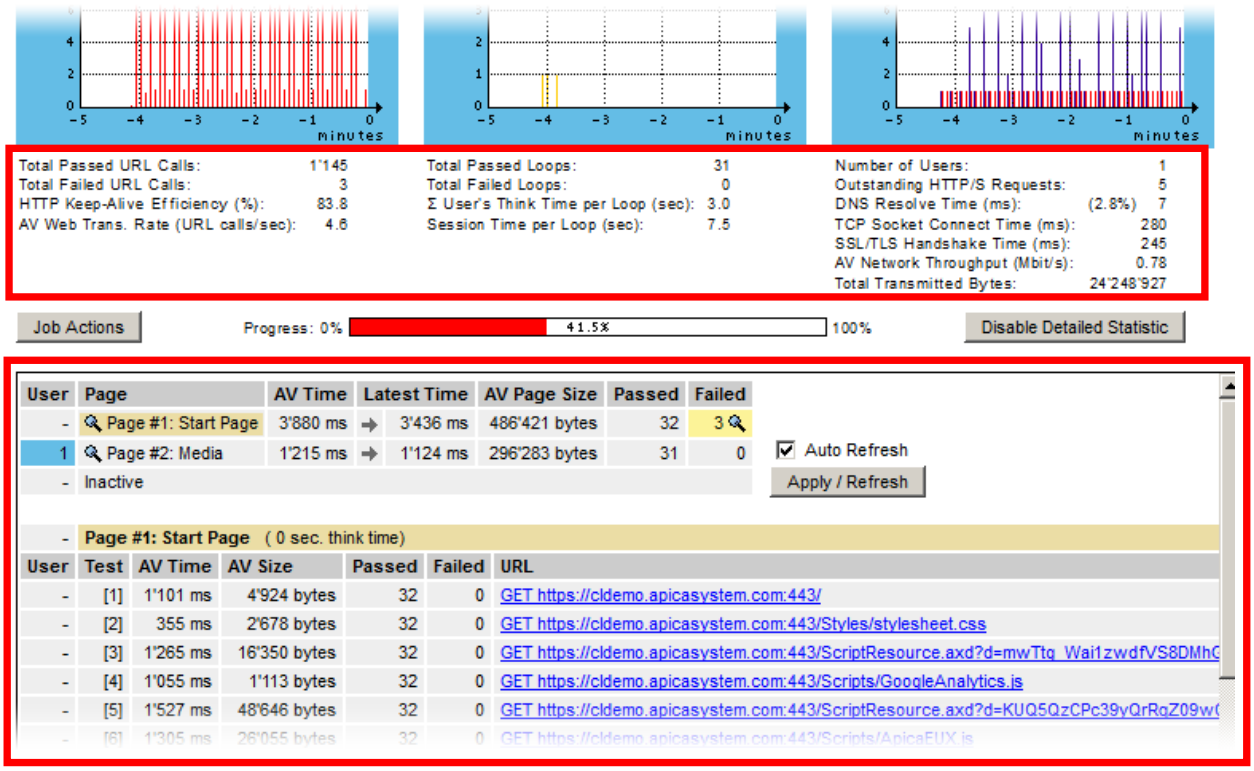
Success exit codes:
10 = job defined, 11 = job running, 12 = job completed, 13 = unknown state, 14 = scheduled

Note: The exit code 13, unknown state, is also returned if an invalid job id was passed to the command

Example:
java PrxJob getJobState “Local Exec Agent” 445

4.4.1.10 getJobRealTimeData

Provides measured performance data of a running job at real-time and writes them in XML format to stdout. To get a valid result the current state of the Exec Agent job must be “job running”. The XML data contain performance information as displayed in the Web Admin GUI:



Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the “Exec Agent Network Configuration” menu (shown as “description”)
<job id>	Exec Agent job id
[-detailed]	Optional. Include detailed data for all URLs and pages in the XML result

Success exit code: 0

Example:
java PrxJob getJobRealTimeData "Local Exec Agent" 1158 -detailed

Example of XML output:

```
<?xml version="1.0" encoding="UTF-8"?>
<jobRealTimeData>
  <proxySnifferVersion>V5.2-T</proxySnifferVersion>
  <localTimeStamp>1430168021374</localTimeStamp>
  <execAgentTimeStamp>1430168021363</execAgentTimeStamp>
  <execAgentName>Local Exec Agent</execAgentName>
  <execAgentLocalDate>27 Apr 2015 22:53:41</execAgentLocalDate>
  <execAgentLocalTimeZone>ECT</execAgentLocalTimeZone>
  <jobId>441</jobId>
  <nearEndOfTest>false</nearEndOfTest>
  <suspended>false</suspended>
  <jobStartExecAgentTimeStamp>1430168005314</jobStartExecAgentTimeStamp>
  <loadTestProgramName>CLDEMO_SSL</loadTestProgramName>
  <plannedNumberOfUsers>1</plannedNumberOfUsers>
  <plannedTestDurationSeconds>600</plannedTestDurationSeconds>
  <plannedLoopsPerUser>0</plannedLoopsPerUser>
  <activeUsers>1</activeUsers>
  <outstandingRequests>0</outstandingRequests>
  <totalPassedLoops>1</totalPassedLoops>
  <totalFailedLoops>0</totalFailedLoops>
  <totalPassedUrlCalls>67</totalPassedUrlCalls>
  <totalFailedUrlCallsHandleAsError>0</totalFailedUrlCallsHandleAsError>
  <totalFailedUrlCallsIgnoreError>2</totalFailedUrlCallsIgnoreError>
  <averagePassedUrlCallsPerSecond>4.3925786</averagePassedUrlCallsPerSecond>
```

```
  <averageSumUsersThinkTimePerLoopMilliseconds>3000</averageSumUsersThinkTimePerLoopMilliseconds>
  <averageSessionTimePerLoopMilliseconds>9108</averageSessionTimePerLoopMilliseconds>
  <actualSessionTimePerLoopMilliseconds>8841</actualSessionTimePerLoopMilliseconds>
  <averageHttpKeepAliveEfficiencyPercent>83.78378</averageHttpKeepAliveEfficiencyPercent>
  <averageDNSResolveTime>37</averageDNSResolveTime>
  <actualDNSResolveTime>26</actualDNSResolveTime>
  <dnsResolveRate>2.9850745</dnsResolveRate>
  <averageSocketConnectTimeTimeMilliseconds>350</averageSocketConnectTimeTimeMilliseconds>
  <actualSocketConnectTimeTimeMilliseconds>293</actualSocketConnectTimeTimeMilliseconds>
  <averageSSLHandshakeTime>456</averageSSLHandshakeTime>
  <actualSSLHandshakeTime>599</actualSSLHandshakeTime>
  <averageNetworkThroughputMegabitsPerSecond>0.6650652</averageNetworkThroughputMegabitsPerSecond>
  <totalTransmittedBytes>1268030</totalTransmittedBytes>
  <detailData>
    <dataRecord>
      <index>0</index>
      <dataRecordType>0</dataRecordType>
      <infoText>Page #1: Start Page</infoText>
      <maxAcceptableResponseTime>-1</maxAcceptableResponseTime>
      <averageResponseTime>5109</averageResponseTime>
      <lastValidResponseTime>4842</lastValidResponseTime>
      <passedCount>2</passedCount>
      <failedCount>0</failedCount>
      <pageBreakAverageDelay>0</pageBreakAverageDelay>
      <urlAverageDNSResolveTime>-1</urlAverageDNSResolveTime>
      <urlDNSResolveRate>-1.0</urlDNSResolveRate>
      <urlAverageNetworkEstablishTime>-1</urlAverageNetworkEstablishTime>
      <urlAverageSSLHandshakeTime>-1</urlAverageSSLHandshakeTime>
      <urlAverageRequestTransmitTime>-1</urlAverageRequestTransmitTime>
      <urlAverageResponseContentReceiveTime>-1</urlAverageResponseContentReceiveTime>
      <urlAverageResponseHeaderReceiveTime>-1</urlAverageResponseHeaderReceiveTime>
      <urlAverageResponseHeaderWaitTime>-1</urlAverageResponseHeaderWaitTime>
      <urlAverageSize>0</urlAverageSize>
      <urlExecutionMode>-1</urlExecutionMode>
      <urlFailureActionType>2</urlFailureActionType>
    </dataRecord>
    <dataRecord>
      <index>1</index>
      <dataRecordType>1</dataRecordType>
      <infoText>GET https://cldemo.apicasystem.com:443/</infoText>
      <maxAcceptableResponseTime>-1</maxAcceptableResponseTime>
      <averageResponseTime>1460</averageResponseTime>
      <lastValidResponseTime>1222</lastValidResponseTime>
      <passedCount>2</passedCount>
      <failedCount>0</failedCount>
      <pageBreakAverageDelay>-1</pageBreakAverageDelay>
      <urlAverageDNSResolveTime>37</urlAverageDNSResolveTime>
      <urlDNSResolveRate>100.0</urlDNSResolveRate>
      <urlAverageNetworkEstablishTime>319</urlAverageNetworkEstablishTime>
      <urlAverageSSLHandshakeTime>695</urlAverageSSLHandshakeTime>
      <urlAverageRequestTransmitTime>1</urlAverageRequestTransmitTime>
      <urlAverageResponseContentReceiveTime>297</urlAverageResponseContentReceiveTime>
      <urlAverageResponseHeaderReceiveTime>0</urlAverageResponseHeaderReceiveTime>
      <urlAverageResponseHeaderWaitTime>113</urlAverageResponseHeaderWaitTime>
      <urlAverageSize>8916</urlAverageSize>
      <urlExecutionMode>1</urlExecutionMode>
      <urlFirstReceivedContentType>TEXT/HTML</urlFirstReceivedContentType>
      <urlFailureActionType>2</urlFailureActionType>
    </dataRecord>
    <dataRecord>
      <index>2</index>
      <dataRecordType>1</dataRecordType>
      <infoText>GET https://cldemo.apicasystem.com:443/Styles/stylesheet.css</infoText>
      <maxAcceptableResponseTime>-1</maxAcceptableResponseTime>
      <averageResponseTime>786</averageResponseTime>
      <lastValidResponseTime>328</lastValidResponseTime>
      <passedCount>2</passedCount>
      <failedCount>0</failedCount>
      <pageBreakAverageDelay>-1</pageBreakAverageDelay>
      <urlAverageDNSResolveTime>-1</urlAverageDNSResolveTime>
      <urlDNSResolveRate>0.0</urlDNSResolveRate>
      <urlAverageNetworkEstablishTime>389</urlAverageNetworkEstablishTime>
      <urlAverageSSLHandshakeTime>249</urlAverageSSLHandshakeTime>
      <urlAverageRequestTransmitTime>1</urlAverageRequestTransmitTime>
      <urlAverageResponseContentReceiveTime>465</urlAverageResponseContentReceiveTime>
      <urlAverageResponseHeaderReceiveTime>0</urlAverageResponseHeaderReceiveTime>
      <urlAverageResponseHeaderWaitTime>1</urlAverageResponseHeaderWaitTime>
      <urlAverageSize>2678</urlAverageSize>
      <urlExecutionMode>2</urlExecutionMode>
      <urlFirstReceivedContentType>TEXT/CSS</urlFirstReceivedContentType>
      <urlFailureActionType>1</urlFailureActionType>
    </dataRecord>
    <dataRecord>    ...
    ...
  </detailData>
</jobRealTimeData>
```

Notes:

- Values which names are starting with “average…” are calculated over the whole load test duration.
- Time stamp values are the number of milliseconds, elapsed since January 1, 1970 UTC.
- The elapsed time of the load test can be calculated as execAgentTimeStamp minus jobStartExecAgentTimeStamp
- If the load test job has already been completed, the **exit code is -91** (wrong job state).

The following XML tags may have a value of **-1 (minus one)** which means that **no data** are available. This may happen near the start or the near the end of the load test – or also if the load test completely fails.

- averageSumUsersThinkTimePerLoopMilliseconds
- averageSessionTimePerLoopMilliseconds (returns -1 if totalPassedLoops < 1)
- actualSessionTimePerLoopMilliseconds (returns -1 if totalPassedLoops < 1)
- averageSocketConnectTimeTimeMilliseconds
- actualSocketConnectTimeTimeMilliseconds
- totalTransmittedBytes

Special Data Fields:

- dataRecordType: 0 = page break, 1 = URL call
- urlExecutionMode: 1 = serial executed, 2 = parallel executed
- urlFailureActionType: 1 = continue loop (yellow error), 2 = abort loop (red error)

Note: starting from ZebraTester version 5.0 the measuring value "usersWaitingForResponse" is no longer available and has been replaced by "outstandingRequests".

4.4.1.11 changeJobNumSimulatedUser

Increase or decrease the number of simulated users of a load test job at runtime.

Argument	Description
<exec agent name>	The name of the Exec Agent, as defined in the “Exec Agent Network Configuration” menu (shown as “description”)
<job id>	The Exec Agent job id
<± delta number of users>	The number of simulated users to remove or to add at runtime (negative or positive value)
[<startup delay per user>]	Optional, the startup delay in milliseconds for new added user, -1 or absent = use load test job default value. Not considered if <delta number of users> is equal or less than zero.

Success exit code: greater than zero: The accepted number of simulated users that are pending to decrease or to increase (always a positive value).

4.4.1.12 setJobSuspend

Suspend or resume the execution of a (running) load test job.

Argument	Description
<exec agent name>	The name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<job id>	The Exec Agent job id
<"true" or "false">	true = suspend, false = resume
[<startup delay per user>]	Optional, the startup delay per simulated user in milliseconds applied when the load test is resumed, -1 or absent = use load test default value. Only considered if the load test job is resumed.

Success exit code: 0

4.4.1.13 isJobSuspend

Get if a currently running load test job is suspended.

Argument	Description
<exec agent name>	The name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<job id>	The Exec Agent job id

Success exit code: 1 = job currently suspended, 0 = job currently not suspended, or job not running.

4.4.1.14 changeJobTestDuration

Increase or decrease the test duration of a currently running load test job.

Argument	Description
<exec agent name>	The name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<job id>	The Exec Agent job id
<± seconds>	The number of seconds to remove or to add from the planned test duration (negative or positive value)

Success exit code: 0.

4.4.1.15 getJobRealTimeUserInputFields

Get a list of "User Input Fields" whose values can be changed in real-time. Note: The value of such a "real-time" User Input Field is wired to a Java System Property of the load test job. To alter the value you can call **setJobJavaSystemProperties**.

Argument	Description
<exec agent name>	The name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<job id>	The Exec Agent job id

Success exit code: 0.

Example:
java PrxJob getJobRealTimeUserInputFields "Local Exec Agent" 1158

Example of XML output:

```
<?xml version="1.0" encoding="UTF-8"?>
<list>
  <realTimeUserInputField>
    <userInputField>
      <varName>vHost</varName>
      <varLabelText>Hostname</varLabelText>
      <defaultValue>cldemo.apicasystem.com</defaultValue>
    </userInputField>
    <realTimeJavaSystemPropertyKey>vHost</realTimeJavaSystemPropertyKey>
    <realTimeValue>cldemo.apicasystem.com</realTimeValue>
  </realTimeUserInputField>
  <realTimeUserInputField>
    <userInputField>
      <varName>vPort</varName>
      <varLabelText>TCP Port</varLabelText>
      <defaultValue>443</defaultValue>
    </userInputField>
    <realTimeJavaSystemPropertyKey>vPort</realTimeJavaSystemPropertyKey>
    <realTimeValue>443</realTimeValue>
  </realTimeUserInputField>
</list>
```

4.4.1.16 abortJob

Causes the Exec Agent to abort the job. Note that the job will terminate itself in an orderly fashion, which may take some time. This means that the job is not immediately completed after this command has been called. You can use the command **getJobState** or **waitForJobCompletion** to detect when a job has been completed.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<job id>	Exec Agent job id

Success exit code: 0

4.4.1.17 waitForJobCompletion

Waits until one or several jobs for the same Exec Agent have been completed. The command exits when the last specified job has been completed

Warning: if one of the jobs is in the state "defined", but has not yet been started, this command will never return.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<job id 1> .. [<job id n>]	Exec Agent job id(s)

Success exit code: 0

Example:
java PrxJob waitForJobCompletion "Local Exec Agent" 445 446 447

4.4.1.18 acquireJobResultFile

Acquires the result file (*.prxres), containing all measured data, of a completed job. The result file is transferred back to the local machine, and stored on disk.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<job id>	Exec Agent job id
<result file name>	Absolute or relative local path of the result file. The freely-selectable result file name must have the file extension ".prxres"

Success exit code: 0

4.4.1.19 acquireJobOutputFile

Acquires the output file (log file) of a completed job. The output file is transferred back to the local machine, and stored on disk.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<job id>	Exec Agent job id
<output file name>	Absolute or relative local path of the output file. The freely-selectable output file name must have the file extension ".out"

Success exit code: 0

4.4.1.20 acquireJobErrorFile

Acquires the error file of a completed job. The error file is transferred back to the local machine, and stored on disk.

Note: this file will only contain data if the job itself failed due to a run-time exception, and was not able to complete. This file does not contain information about load test errors.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<job id>	Exec Agent job id
<error file name>	Absolute or relative local path of the error file. The freely-selectable error file name must have the file extension ".err"

Success exit code: 0

4.4.1.21 downloadJobFile

Download the data on any file located in the (remote) directory of the specified Exec Agent job. The file is transferred back to the local machine, and stored on disk. If no <local file name> is passed the locally created file will have the same file name as the <remote file name>.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<job id>	Exec Agent job id
<remote file name>	Remote file name – without any path. The following characters are not allowed to be part of a remote file name: .., /, \, :, ~, ^, *, $, @, ", „, %, `, ?, &, !
[<local file name>]	Optional. Absolute or relative path of the local file.

Success exit code: 0

4.4.1.22 deleteJob

Deletes an Exec Agent job.

Note: currently running jobs cannot be deleted; in this case, you may use the abortJob command.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<job id>	Exec Agent job id

Success exit code: 0

4.4.1.23 deleteAllJobs

Deletes all Exec Agent jobs.

Note: currently running and/or scheduled jobs are not deleted.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")

Success exit code: 0

4.4.1.24 getJobList

Displays a list of all Exec Agent job IDs, or – if the option -verbose is used – a list of all Exec Agent job data similar to the output of the Web Admin “Exec Agent Jobs” Menu.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the “Exec Agent Network Configuration” menu (shown as “description”)
<job state>	Selection criteria of displayed jobs. Possible values are "defined", "scheduled", "running", "completed", "unknown", or "all"
[-verbose]	Optional argument. Display a list of all Exec Agent job data similar to the output of the Web Admin “Exec Agent Jobs” Menu. The line fields (columns) of the job output data are separated by semicolons (;) • field 1: job ID • field 2: current job state (defined, scheduled, running completed or unknown) • field 3: date and time of current job state (milliseconds since January 1, 1970) • field 4: load test program & arguments • field 5: TCP/IP address (remote computer) from which the job has been initiated
[-outfile <file name>]	Optional. If this argument is used the output is written to an arbitrary file instead to stdout.

Success exit code: 0

Example:
java PrxJob getJobList "Local Exec Agent" all

Output:
1368
1367
1366
1365

4.4.2 Cluster Job Related Commands

4.4.2.1 transmitClusterJob

Transmits a load test program to an Exec Agent cluster, but does not start it. The corresponding cluster job is created, and will be in the state “defined”.

Note: the number of “concurrent users” which are specified as program argument -u are automatically distributed over the Exec Agent cluster members, depending on the load factors of the cluster members.

Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<load test>	Local or absolute path of the load test program (*.class or *.zip).
[-pmaTemplate <template name>]	Optional. If this argument is used external measured performance data are added to the load test result, collected by an Apica Performance Monitoring Agent Controller (PMA). The <template name> must referrer a valid PMA template name as defined in the ZebraTester GUI (WebAdmin). See also: command getApicaPMAMonitoringTemplateList.
<load test arguments>	All arguments for the load test program (see chapter 3.8) Note: if the (optional) argument -annotation is used, it must be the last argument

Success exit code: cluster job id (>0)

Note: cluster jobs have their own ids, and they are not the same as Exec Agent job ids.

Example:
java PrxJob transmitClusterJob "Cluster 1" Test01.zip -u 20 -d 30 -t 60 -nolog -annotation "first test run"

4.4.2.2 setClusterJobScheduleTime

Schedules a cluster job to be automatically started in future. Requirement: the corresponding cluster job must be already created by using **transmitClusterJob**. The cluster job changes the state from “defined” to “scheduled”.

Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<cluster job id>	Cluster job id
<number of seconds in future>	Number of seconds in future, when the cluster job will be automatically started. A value of -1 (minus one) effects that an already defined schedule will be cancelled.
[-split <input file 1> <line comment tag>] … [-split <input file n> <line comment tag>]	Optional: splits a load test program input file such that each cluster member (Exec Agent) receives only a part of the input file. You can use this functionality, for example, to avoid duplicated user account logins when executing a cluster job. If the -split option is not set, each cluster member will receive a full copy of the input file. The "line comment tag" is required, and specifies the starting pattern for commented-out lines in the input file.

Success exit code: 0

Example:
java PrxJob startClusterJob "Cluster 1" 224 600 -split userAccounts.txt “#”

4.4.2.3 startClusterJob

Starts a cluster job. The cluster job must be in the state “defined”, and will then change state to “running”. This command exits immediately after the cluster job has been started. You can use the commands **getClusterJobState** or **waitForClusterJobCompletion** to detect when a cluster job has completed.

Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<cluster job id>	Cluster job id
[-split <input file 1> <line comment tag>] … [-split <input file n> <line comment tag>]	Optional: splits a load test program input file such that each cluster member (Exec Agent) receives only a part of the input file. You can use this functionality, for example, to avoid duplicated user account logins when executing a cluster job. If the -split option is not set, each cluster member will receive a full copy of the input file. The "line comment tag" is required, and specifies the starting pattern for commented-out lines in the input file.

Success exit code: 0

Example:
java PrxJob startClusterJob "Cluster 1" 224 -split userAccounts.txt “#”

4.4.2.4 getClusterJobStartStatistics

Get internal statistical data collected during starting a cluster job.

Argument	Description
<cluster name>	Name of the cluster, as defined in the "Exec Agent Network Configuration" menu
<cluster job id>	Cluster job id
[-outfile <file name>]	Optional. If this argument is used the output is written to an arbitrary file instead to stdout.

Success exit code: 0

Example:
java PrxJob -tz PST getClusterJobStartStatistics "Cluster1" 27

Output:
jobId = 27
validInstanceCreateTimestamp = 22 Sep 2015 23:19:17.597
startState = START_STATE_START_DONE
localDataReplicationStartTimestamp = 22 Sep 2015 23:19:19.952
localDataReplicationProgressPercent = 100.00
localDataReplicationEndTimestamp = 22 Sep 2015 23:19:22.390
measuringOSTimeDifferenceStartTimestamp = 22 Sep 2015 23:19:22.396
measuringOSTimeDifferenceProgressPercent = 100.00
measuringOSTimeDifferenceEndTimestamp = 22 Sep 2015 23:19:22.647
dataTransmissionStartTimestamp = 22 Sep 2015 23:19:22.648
dataTransmissionProgressPercent = 100.00
dataTransmissionEndTimestamp = 22 Sep 2015 23:19:23.521
startTriggeringStartTimestamp = 22 Sep 2015 23:19:23.522
startTriggeringProgressPercent = 100.00
startTriggeringEndTimestamp = 22 Sep 2015 23:19:50.886
startDoneTimestamp = 22 Sep 2015 23:19:50.988

Note 1: The argument **-tz PST** used in the example above is optional and allows to set a specific time zone as documented in chapter 5 Supported Time Zones.

Note 2: If this command is called asynchronously – during the start of the cluster job – then the field **startState** can have one of the following values:

0. START_STATE_START_NOT_STARTED
   The start of the cluster job was not yet triggered (initial value).

1. START_STATE_LOCAL_DATA_REPLICATION
   The local data replication has begun.

2. START_STATE_MEASURING_OS_TIME_DIFFERENCE
   Measuring the OS time differences to the Exec Agents (cluster members) has begun.

3. START_STATE_DATA_TRANSMISSION
   The data transmission of the load test to the Exec Agents has begun.

4. START_STATE_START_TRIGGERING
   Triggering of the start of the Exec Agent jobs has begun.

5. START_STATE_START_DONE
   The start of the cluster job is completed. Note that this does not means that the cluster job was successfully started – it means only that the startup sequence is completed at Job Controller side.

Depending on the current startState the progress percent values may be less than 100%.

4.4.2.5 addClusterRealTimeComment

Adds a real time comment to a running cluster job. The real time comment is only added if the job is in the state “job running”.

Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<cluster job id>	Cluster job id
<comment>	The real time comment

Success exit code: 0

Example:
java PrxJob addClusterRealTimeComment "Cluster 1" 224 “failover event triggered”

4.4.2.6 getClusterJobExecutorsAnnotation

Get the "executor's annotation" of a running cluster job. The "executor's annotation" is only returned if the job is in the state “job running”.

Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<cluster job id>	Cluster job id
[-outfile <file name>]	Optional. If this argument is used the output is written to an arbitrary file instead to stdout.

Success exit code: 0

Example:
java PrxJob getClusterJobExecutorsAnnotation "Cluster 1" 224

4.4.2.7 setClusterJobExecutorsAnnotation

Set a new/updated "executor's annotation" for a running cluster job. The "executor's annotation" is only updated if the cluster job is in the state “job running”.

Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<cluster job id>	Cluster job id
<text>	The executor's annotation

Success exit code: 0

Example:
java PrxJob setClusterJobExecutorsAnnotation "Cluster 1" 224 “verification test after tuning web app”

4.4.2.8 getClusterJobJavaSystemProperties

Get cluster-wide Java system properties from a running cluster job. The system properties are only returned if the cluster job is in the state “job running”.

Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<cluster job id>	Cluster job id
<key select pattern>	A text-pattern that is used to select/filter Java system properties keys. The text-pattern can contain the wildcards '*' (multiple chars) and '?' (single char). If this parameter is an empty string then all available Java system properties are returned.
[-outfile <file name>]	Optional. If this argument is used the output is written to an arbitrary file instead to stdout.

Success exit code: 0

Example:
java PrxJob getClusterJobJavaSystemProperties "Cluster 1" 224 "os.*"

Output:
A : os.arch=x86
1 : os.name=Windows 7
2 : os.name=Windows XP
1 : os.version=6.1
2 : os.version=5.1

Note: the first column, followed by a colon char ':', is the reference count about the number of Exec Agents (cluster members) that contain exactly the same key and value of a Java system proprietie. The character 'A' means that the Java system proprietie for that key has the same value on all cluster members. Any numeric value means that only some of the cluster members have that value for this key.

4.4.2.9 setClusterJobJavaSystemProperties

Set cluster-wide Java system properties for a running cluster job. The system properties are only updated if the cluster job is in the state “job running”.

Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<cluster job id>	Cluster job id
<key=value> [<key=value> .. <key=value>]	A pair of key and value, separated by an equal sign. Multiple arguments of key=value pairs are supported to set multiple system properties at once. If an argument contains no equal sign then the corresponding system propertie is removed.

Success exit code: 0

Example:
java PrxJob setClusterJobJavaSystemProperties "Cluster 1" 224 "xx2=D" "xx3=E"

4.4.2.10 getClusterJobState

Returns the current state of a cluster job.

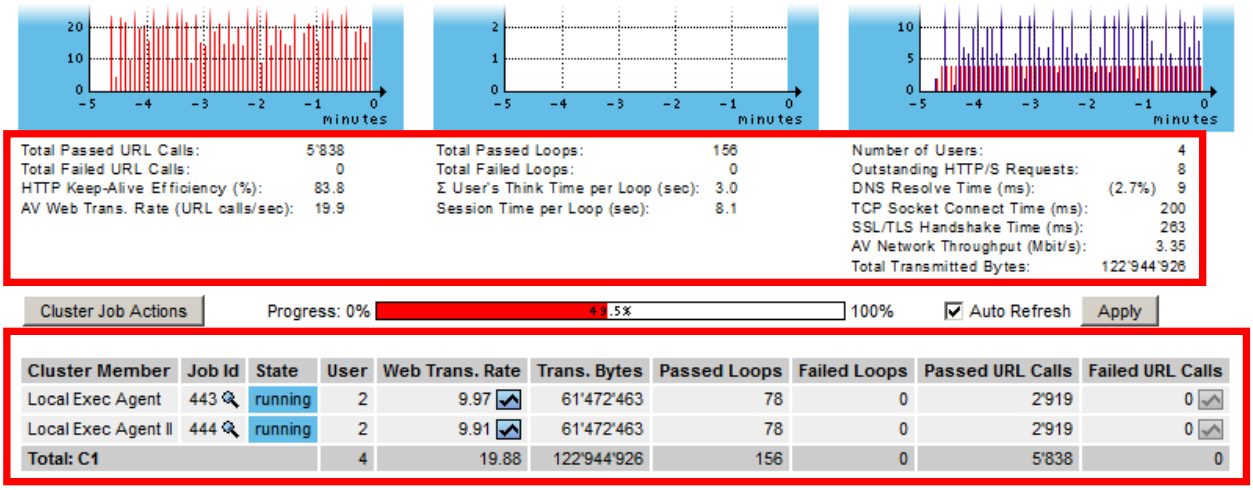
Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<cluster job id>	Cluster job id

Success exit codes: 10 = job defined, 11 = job running, 12 = job completed, 13 = unknown state, 14 = scheduled

Note: exit code 13, unknown state, is also returned if an invalid cluster job id was passed to the command.

4.4.2.11 getClusterJobRealTimeData

Provides measured performance data of a running cluster job at real-time and writes them in XML format to stdout. To get a valid result the current state of the cluster job must be “job running”. The XML data contain performance information as displayed in the Web Admin GUI:



Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<cluster job id>	Cluster job id
[-detailed]	Optional. Include detailed data of all cluster members in the XML result

Success exit code: 0

Example:
java PrxJob getClusterJobRealTimeData C1 27 -detailed

Example of XML output:

```
<?xml version="1.0" encoding="UTF-8"?>
<clusterJobRealTimeData>
  <proxySnifferVersion>V5.2-T</proxySnifferVersion>
  <timeStamp>1430172547351</timeStamp>
  <clusterJobId>27</clusterJobId>
  <clusterJobStartDate>1430172531923</clusterJobStartDate>
  <validExecAgentCount>2</validExecAgentCount>
  <totalActiveUsers>4</totalActiveUsers>
  <totalOutstandingRequests>1</totalOutstandingRequests>
  <totalPassedUrlCalls>281</totalPassedUrlCalls>
  <totalFailedUrlCallsHandleAsError>0</totalFailedUrlCallsHandleAsError>
  <totalFailedUrlCallsIgnoreError>0</totalFailedUrlCallsIgnoreError>
  <totalPassedLoops>5</totalPassedLoops>
  <totalFailedLoops>0</totalFailedLoops>
  <totalWebTransRate>24.618977</totalWebTransRate>
  <totalTransBytes>5367264</totalTransBytes>
  <totalNetworkThroughput>3.7300053</totalNetworkThroughput>
  <recycledNetworkConnectionRate>83.78378</recycledNetworkConnectionRate>
  <userThinkTimeSeconds>3.0</userThinkTimeSeconds>
  <sessionTimeSeconds>7.4175</sessionTimeSeconds>
  <dnsResolveTime>11</dnsResolveTime>
  <networkEstablishTime>190</networkEstablishTime>
  <sslHandshakeTime>256</sslHandshakeTime>
  <realPlannedTestDuration>600</realPlannedTestDuration>
  <hasJavaMemoryWarning>false</hasJavaMemoryWarning>
  <maxGarbageCollectionsTotalCount>2</maxGarbageCollectionsTotalCount>
  <maxGarbageCollectionsTotalTime>40</maxGarbageCollectionsTotalTime>
  <nearEndOfTest>false</nearEndOfTest>
  <suspended>false</suspended>
  <memberPerformanceDataList>
    <execAgentJobRealTimeData>
      <proxySnifferVersion></proxySnifferVersion>
      <localTimeStamp>1430172547353</localTimeStamp>
      <execAgentTimeStamp>1430172547236</execAgentTimeStamp>
      <execAgentName>Local Exec Agent</execAgentName>
      <execAgentLocalDate>28 Apr 2015 00:09:07</execAgentLocalDate>
```

```
      <execAgentLocalTimeZone></execAgentLocalTimeZone>
      <jobId>445</jobId>
      <nearEndOfTest>false</nearEndOfTest>
      <suspended>false</suspended>
      <jobStartExecAgentTimeStamp>1430172532332</jobStartExecAgentTimeStamp>
      <loadTestProgramName>CLDEMO_SSL</loadTestProgramName>
      <plannedNumberOfUsers>2</plannedNumberOfUsers>
      <plannedTestDurationSeconds>600</plannedTestDurationSeconds>
      <plannedLoopsPerUser>0</plannedLoopsPerUser>
      <activeUsers>2</activeUsers>
      <outstandingRequests>1</outstandingRequests>
      <totalPassedLoops>3</totalPassedLoops>
      <totalFailedLoops>0</totalFailedLoops>
      <totalPassedUrlCalls>143</totalPassedUrlCalls>
      <totalFailedUrlCallsHandleAsError>0</totalFailedUrlCallsHandleAsError>
      <totalFailedUrlCallsIgnoreError>0</totalFailedUrlCallsIgnoreError>
      <averagePassedUrlCallsPerSecond>9.7338505</averagePassedUrlCallsPerSecond>
      <averageSumUsersThinkTimePerLoopMilliseconds>3000</averageSumUsersThinkTimePerLoopMilliseconds>
      <averageSessionTimePerLoopMilliseconds>8085</averageSessionTimePerLoopMilliseconds>
      <actualSessionTimePerLoopMilliseconds>7350</actualSessionTimePerLoopMilliseconds>
      <averageHttpKeepAliveEfficiencyPercent>83.78378</averageHttpKeepAliveEfficiencyPercent>
      <averageDNSResolveTime>33</averageDNSResolveTime>
      <actualDNSResolveTime>13</actualDNSResolveTime>
      <dnsResolveRate>0.0</dnsResolveRate>
      <averageSocketConnectTimeTimeMilliseconds>268</averageSocketConnectTimeTimeMilliseconds>
      <actualSocketConnectTimeTimeMilliseconds>186</actualSocketConnectTimeTimeMilliseconds>
      <averageSSLHandshakeTime>306</averageSSLHandshakeTime>
      <actualSSLHandshakeTime>254</actualSSLHandshakeTime>
      <averageNetworkThroughputMegabitsPerSecond>1.5419247</averageNetworkThroughputMegabitsPerSecond>
      <totalTransmittedBytes>2831552</totalTransmittedBytes>
      <detailData/>
    </execAgentJobRealTimeData>
    <execAgentJobRealTimeData>
      <proxySnifferVersion></proxySnifferVersion>
      <localTimeStamp>1430172547353</localTimeStamp>
      <execAgentTimeStamp>1430172547246</execAgentTimeStamp>
      <execAgentName>Local Exec Agent II</execAgentName>
      <execAgentLocalDate>28 Apr 2015 00:09:07</execAgentLocalDate>
      <execAgentLocalTimeZone></execAgentLocalTimeZone>
      <jobId>446</jobId>
      <nearEndOfTest>false</nearEndOfTest>
      <suspended>false</suspended>
      <jobStartExecAgentTimeStamp>1430172532595</jobStartExecAgentTimeStamp>
      <loadTestProgramName>CLDEMO_SSL</loadTestProgramName>
      <plannedNumberOfUsers>2</plannedNumberOfUsers>
      <plannedTestDurationSeconds>600</plannedTestDurationSeconds>
      <plannedLoopsPerUser>0</plannedLoopsPerUser>
      <activeUsers>2</activeUsers>
      <outstandingRequests>0</outstandingRequests>
      <totalPassedLoops>2</totalPassedLoops>
      <totalFailedLoops>0</totalFailedLoops>
      <totalPassedUrlCalls>138</totalPassedUrlCalls>
      <totalFailedUrlCallsHandleAsError>0</totalFailedUrlCallsHandleAsError>
      <totalFailedUrlCallsIgnoreError>0</totalFailedUrlCallsIgnoreError>
      <averagePassedUrlCallsPerSecond>14.885126</averagePassedUrlCallsPerSecond>
      <averageSumUsersThinkTimePerLoopMilliseconds>3000</averageSumUsersThinkTimePerLoopMilliseconds>
      <averageSessionTimePerLoopMilliseconds>7947</averageSessionTimePerLoopMilliseconds>
      <actualSessionTimePerLoopMilliseconds>7485</actualSessionTimePerLoopMilliseconds>
      <averageHttpKeepAliveEfficiencyPercent>83.78378</averageHttpKeepAliveEfficiencyPercent>
      <averageDNSResolveTime>19</averageDNSResolveTime>
      <actualDNSResolveTime>8</actualDNSResolveTime>
      <dnsResolveRate>0.0</dnsResolveRate>
      <averageSocketConnectTimeTimeMilliseconds>237</averageSocketConnectTimeTimeMilliseconds>
      <actualSocketConnectTimeTimeMilliseconds>194</actualSocketConnectTimeTimeMilliseconds>
      <averageSSLHandshakeTime>264</averageSSLHandshakeTime>
      <actualSSLHandshakeTime>258</actualSSLHandshakeTime>
      <averageNetworkThroughputMegabitsPerSecond>2.1880805</averageNetworkThroughputMegabitsPerSecond>
      <totalTransmittedBytes>2535712</totalTransmittedBytes>
      <detailData/>
    </execAgentJobRealTimeData>
  </memberPerformanceDataList>
</clusterJobRealTimeData>
```


4.4.2.12 changeClusterJobNumSimulatedUser

Increase or decrease the number of simulated users of a cluster job at runtime.

Argument	Description
<cluster name>	Name of the cluster, as defined in the "Exec Agent Network Configuration" menu
<cluster job id>	Cluster job id
<± delta number of users>	The number of simulated users to remove or to add at runtime (negative or positive value)
[<startup delay per user>]	Optional, the startup delay in milliseconds for new added user, -1 or absent = use load test job default value. Not considered if <delta number of users> is equal or less than zero.

Success exit code: greater than zero: The accepted number of simulated users that are pending to decrease or to increase (always a positive value).

4.4.2.13 setClusterJobSuspend

Suspend or resume the execution of a (running) cluster job.

Argument	Description
<cluster name>	Name of the cluster, as defined in the "Exec Agent Network Configuration" menu
<cluster job id>	Cluster job id
<"true" or "false">	true = suspend, false = resume
[<startup delay per user>]	Optional, the startup delay per simulated user in milliseconds applied when the load test is resumed, -1 or absent = use load test default value. Only considered if the load test job is resumed.

Exit code: the number of cluster members for which a change of the state has failed. 0 = success.

4.4.2.14 isClusterJobSuspend

Get if a currently running cluster job is suspended.

Argument	Description
<cluster name>	Name of the cluster, as defined in the "Exec Agent Network Configuration" menu
<cluster job id>	Cluster job id

Exit code: the number of cluster members whose corresponding load test jobs are NOT suspended.

4.4.2.15 changeClusterJobTestDuration

Increase or decrease the test duration of a currently running cluster job.

Argument	Description
<cluster name>	Name of the cluster, as defined in the "Exec Agent Network Configuration" menu
<cluster job id>	Cluster job id
<± seconds>	The number of seconds to remove or to add from the planned test duration (negative or positive value)

Success exit code: the number of cluster members for which a change of the test duration has failed. 0 = success

4.4.2.16 getClusterJobRealTimeUserInputFields

Get a list of “User Input Fields” whose values can be changed in real-time. Note: The value of such a “real-time” User Input Field is wired to a Java System Property of the load test job. To alter the value you can call **setClusterJobJavaSystemProperties**.

Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<cluster job id>	Cluster job id

Success exit code: 0.

Example:
java PrxJob getClusterJobRealTimeUserInputFields "C1" 44

Example of XML output:

```
<?xml version="1.0" encoding="UTF-8"?>
<list>
  <realTimeUserInputField>
    <userInputField>
      <varName>vHost</varName>
      <varLabelText>Hostname</varLabelText>
      <defaultValue>cldemo.apicasystem.com</defaultValue>
    </userInputField>
    <realTimeJavaSystemPropertyKey>vHost</realTimeJavaSystemPropertyKey>
    <realTimeValue>cldemo.apicasystem.com</realTimeValue>
  </realTimeUserInputField>
  <realTimeUserInputField>
    <userInputField>
      <varName>vPort</varName>
      <varLabelText>TCP Port</varLabelText>
      <defaultValue>443</defaultValue>
    </userInputField>
    <realTimeJavaSystemPropertyKey>vPort</realTimeJavaSystemPropertyKey>
    <realTimeValue>443</realTimeValue>
  </realTimeUserInputField>
</list>
```

4.4.2.17 abortClusterJob

Causes the Job Controller to abort the cluster job. Note that the cluster job will terminate in an orderly fashion, and this may take some time. This also means that the cluster job will not necessarily end immediately after this command is called. You can use the commands **getClusterJobState** or **waitForClusterJobCompletion** to detect when a cluster job has completed.

Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<cluster job id>	Cluster job id

Success exit code: 0

4.4.2.18 waitForClusterJobCompletion

Waits until one or more cluster jobs have completed. The command exits when the last specified cluster job has completed

Warning: if one of the cluster jobs is in the state “defined”, but has not yet been started, this command will never exit.

Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<cluster job id 1> .. [<cluster job id n>]	Cluster job id(s)

Success exit code: 0

Example:
java PrxJob waitForClusterJobCompletion “Cluster 1” 224 225

4.4.2.19 acquireClusterJobResultFile

Acquires the result file (*.prxres), containing all measurement data, for a completed cluster job. The result file is transferred to the local machine, and stored on disk.

The result file contains a consolidated overall result of the measured data for all cluster members; therefore, creating the result file (merging the cluster member data) may take some time.

Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<cluster job id>	Cluster job id
<result file name>	Absolute or relative local path of the result file. The freely-selectable result file name must have the file extension “.prxres”

Success exit code: 0

4.4.2.20 combineClusterFile

Search inside all Exec Agent job directories of the cluster job for files with match with a specific file name pattern and combine these files to one new file on cluster job level. Note: the new file on cluster job level is created by merging the binary data of the Exec Agent files.

Argument	Description
<cluster name>	Name of the cluster, as defined in the "Exec Agent Network Configuration" menu
<cluster job id>	Cluster job id
<file pattern>	File name pattern on Exec Agent level, for example "***.log**"
<result file name>	Combined/merged (remote) file name on cluster level. The following characters are not allowed to be part of a result file name: .., /, \, :, ~, ^, *, $, @, ", „, %, `, ?, &, !

Success exit code: 0

Example:
java PrxJob combineClusterFile MyCluster 4 "*.log" combined_cluster_job_4.log

4.4.2.21 downloadClusterJobFile

Download the data on any file located in the (remote) directory of the specified cluster job. The file is transferred back to the local machine, and stored on disk. If no <local file name> is passed the locally created file will have the same file name as the <remote file name>.

Argument	Description
<cluster name>	Name of the cluster, as defined in the "Exec Agent Network Configuration" menu
<cluster job id>	Cluster job id
<remote file name>	Remote file name – without any path. The following characters are not allowed to be part of a remote file name: .., /, \, :, ~, ^, *, $, @, ", „, %, `, ?, &, !
[<local file name>]	Optional. Absolute or relative path of the local file.

Success exit code: 0

Example:
java PrxJob downloadClusterJobFile MyCluster 4 combined_cluster_job_4.log

4.4.2.22 getClusterJobExecAgentCount

Returns the number of active cluster members (number of Exec Agents) in a cluster job. “Active” means that only cluster members which are currently executing, or have executed the load test **with at least one concurrent user**, are counted. This command can only be called if the cluster job is in the state “running” or “completed”.

Note / Example: if a cluster job has been started on 3 cluster members with a total of one concurrent user, one member is active and two members are inactive, because a single user cannot be split.

Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<cluster job id>	Cluster job id

Success exit code: number of active cluster members (number of Exec Agents)

4.4.2.23 getClusterJobExecAgentName

Writes the name of an active cluster member (Exec Agent) in a cluster job **to stdout** (standard output). This command can only be called if the cluster job is in the state “running” or “completed”.

Hint: you can use the output of this command, in combination with the command **getClusterJobExecAgentJobId**, as input parameters for the commands **acquireJobOutputFile** and **acquireJobErrorFile**, to get access to the output (log) files and the error files of the cluster members. Calling **acquireJobResultFile** in the same way is **not recommended**, and unnecessary, because a cluster job result file already contains all results for the cluster members.

Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<cluster job id>	Cluster job id
<cluster member number>	Number (1, 2, 3 ..) of active cluster member. Hint: you can use the command getClusterJobExecAgentCount to get the highest allowable number. The numbers start at 1 (one), not at 0 (zero).
[-outfile <file name>]	Optional. If this argument is used the name of the cluster member is written to an arbitrary file instead to stdout.

Success exit code: 0

Example
Java PrxJob getClusterJobExecAgentName “Cluster 1” 224 1 -outfile x.txt

4.4.2.24 getClusterJobExecAgentJobId

Returns the Exec Agent job ID of a cluster member. This command can only be called if the cluster job is in the state “running” or “completed”.

See also the command **getClusterJobExecAgentName** mentioned above.

Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<cluster job id>	Cluster job ID
<cluster member number>	Number (1, 2, 3 ..) of active cluster member. Hint: you can use the command getClusterJobExecAgentCount to get the highest allowable number. The numbers start at 1 (one), not at 0 (zero).

Success exit code: Exec Agent job ID

4.4.2.25 deleteClusterJob

Deletes a cluster job.

Note: currently running cluster jobs cannot be deleted; in this case, you may use the **abortClusterJob** command.

Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu
<cluster job id>	Cluster job id

Success exit code: 0

4.4.2.26 deleteAllClusterJobs

Deletes all jobs in a cluster.

Note: currently running and/or scheduled jobs are not deleted.

Argument	Description
<cluster name>	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu

Success exit code: 0

4.4.2.27 getClusterJobList

Displays a list of cluster job IDs, or – if the option -verbose is used – a list of all cluster job data similar to the output of the Web Admin “Cluster Jobs” Menu.

Argument	Description
<cluster name> | "-all”	Name of the cluster, as defined in the “Exec Agent Network Configuration” menu – or “-all” can be used to select cluster jobs of all defined clusters.
<job state>	Selection criteria of displayed jobs. Possible values are "defined", "scheduled", "running", "completed", "unknown", or "all"
[-verbose]	Optional argument. Display a list of all cluster job data similar to the output of the Web Admin “Cluster Jobs” Menu. The line fields (columns) of the job output data are separated by semicolons (;) • field 1: job ID • field 2: current job state (defined, scheduled, running completed or unknown) • field 3: date and time of current job state (milliseconds since January 1, 1970) • field 4: load test program & arguments • field 5: TCP/IP address (remote computer) from which the job has been initiated
[-outfile <file name>]	Optional. If this argument is used the output is written to an arbitrary file instead to stdout.

Success exit code: 0

Example:
java PrxJob getClusterJobList -all all

Output:
214
213
212
211

4.4.3 Exec Agent Utility Commands

4.4.3.1 getExecAgentList

Displays a list of all Exec Agents, as defined in the “Exec Agent Network Configuration” menu (shown as “description”).

Argument	Description
[-triggerReload]	Optional. Triggers the Web Admin GUI to re-read the Exec Agent data file (execAgent.dat). This option should be used when the data file was modified manually without using the Web Admin GUI.
[-pingVersion]	Optional. Verifies additionally if all Exec Agent processes are alive (reachable) and appends the ZebraTester product version of the corresponding Exec Agent to the output (separated by a semicolon) Note: if an Exec Agent is unreachable the value “unreachable” is shown instead of the ZebraTester product version.
[-outfile <file name>]	Optional. If this argument is used the output is written to an arbitrary file instead to stdout.

Success exit code: 0

Example:
java PrxJob getExecAgentList -triggerReload -pingVersion

Output:
Local Exec Agent;V4.3-F
Local Exec Agent II;V4.3-F
Remote Exec Agent;unreachable

4.4.3.2 pingExecAgent

Checks if an Exec Agent process is alive (reachable) and writes some data about the Exec Agent to **stdout**.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the “Exec Agent Network Configuration” menu (shown as “description”)

Success exit code: 0 (if the Exec Agent process is alive)

Example 1:
java PrxJob pingExecAgent "Local Exec Agent"

Output 1:
Exec Agent OS Type = Windows XP 5.1 / Java 1.5.0_15 / ZebraTester V4.3-F
Exec Agent OS Time Offset = 0 Seconds
Exec Agent Load Factor = 2.610966

Example 2:
java PrxJob pingExecAgent "Remote Exec Agent"

Output 2:
(no output)

D:\ZebraTester>echo %ERRORLEVEL%
-95

4.4.3.3 getExecAgentLog

Get the last 1000 lines of the Exec Agent log output and write it to **stdout**.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the “Exec Agent Network Configuration” menu (shown as “description”)
[-outfile <file name>]	Optional. If this argument is used the output is written to an arbitrary file instead to stdout.

Success exit code: 0

Example 1:
java PrxJob getExecAgentLog "Local Exec Agent"

4.4.3.4 checkTcpPort

Checks if a TCP/IP network connection to an arbitrary host and port can be established from an Exec Agent. The time to open the network connection (socket open time) – measured in milliseconds – is written to **stdout**.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the “Exec Agent Network Configuration” menu (shown as “description”)
<host name | ip address>	The host name or the IP address of the network connection
<port number>	The TCP/IP server-port number (0..65535) of the network connection
<timeout seconds>	Max. time to wait to establish the network connection. If this time is exceeded the command returns with an error exit code.
[-outfile <file name>]	Optional. If this argument is used the measured time is written to an arbitrary file instead to stdout.

Success exit code: 0 (if the TCP/IP network connection can be established)

Example
java PrxJob checkTcpPort "Local Exec Agent" 192.16.4.5 80 10

Output:
2

4.4.3.5 checkSslPort

Checks if an encrypted SSL network connection to an arbitrary host and port can be established from an Exec Agent. The time to open the SSL network connection (socket open time + SSL handshake time) – measured in milliseconds – is written to **stdout**.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<host name | ip address>	The host name or the IP address of the SSL network connection
<port number>	The TCP/IP server-port number (0..65535) of SSL the network connection
<timeout seconds>	Max. time to wait to establish the SSL network connection. If this time is exceeded the command returns with an error exit code.
[-outfile <file name>]	Optional. If this argument is used the measured time is written to an arbitrary file instead to stdout.

Success exit code: 0 (if the SSL network connection can be established)

Example
java PrxJob checkSslPort "Local Exec Agent" 192.16.4.5 443 10

Output:
69

4.4.3.6 translateDnsName

Translates a DNS name to an IP address on a Exec Agent. The IP address is written to **stdout**.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<dns name>	The DNS name
<timeout seconds>	Max. time to wait for the translation. If this time is exceeded the command returns with an error exit code.
[-outfile <file name>]	Optional. If this argument is used the IP address is written to an arbitrary file instead to stdout.

Success exit code: 0 (if the DNS name can be translated to an IP address)

Example:
java PrxJob translateDnsName "Local Exec Agent" www.proxy-sniffer.com 10

Output:
62.2.218.203

4.4.3.7 checkUrl

Executes an URL call on an Exec Agent and writes the result in XML format to stdout. Follows also HTTP redirections – which means that multiple URL calls may be executed by checkUrl. Received response content data are automatically decompressed if needed.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<url>	URL, like for example http://www.proxy-sniffer.com. The URL may also contain CGI parameters. Note: this argument must always be placed at the second position, before any optional argument is specified
[-httpMethod <method>]	Optional. Allows to set the HTTP request method (for example "GET", "POST", "HEAD" ..). The default value is "GET". Additionally all methods of the WebDAV protocol extension are also supported.
[-maxRedirections <number>]	Optional. Limits the number of followed HTTP redirections. The default value is 10. Hint: you may set a value of 0 (zero) to disable the following of HTTP redirections.
[-urlTimeout <seconds>]	Optional. Aborts the URL call if the specified number of seconds is exceeded. The default value is 60 seconds. Note: the value is applied per each "integrated" URL call. This means that also calls for followed HTTP redirections use this value. Example: 2 redirections and 1 normal URL call may take up to (3 x 60) seconds until a result – or a timeout – is returned.
[-ssl <version>]	Optional. Set the SSL or TLS version. Possible values are "all", "v3", "tls", "tls11" or "tls12"
[-dnsstatistic]	Optional. Measure also the DNS revolve time
[-dnssrv <IP-name-server-1>[,<IP-name-	Optional. Set an alternative list of DNS servers. If this

server-N>]]	option is used then the option –dnsstatistic is also implicitly enabled.
[-addHeader <headerLine>] … [-addHeader <headerLine>]	Optional. Adds an additional header line to the HTTP request. This argument can be passed multiple times which allows to add multiple header lines. Note 1: the following HTTP request header lines are already automatically set by checkUrl: • "Host: <host name>[:<TCP/IP port number>]" • "Accept: */*" • "Accept-Encoding: gzip,deflate" • "Connection: Keep-Alive" • “Content-Length: <value>” (only in case if the option -requestContent <file> is used) Note 2: all additional request header lines are set per each “integrated” URL call. Note 3: if you set the HTTP method “POST” for submitting “normal” form data you should also add the following line to the request header: “Content-Type: application/x-www-form-urlencoded“ and you should use the option -requestContent <file>.
[-requestContent <file>]	Optional. Adds request content data to the URL call. The content data are read from the specified file in binary format (as a stream of bytes). Note 1: the request header line “Content-Length: <value>” is automatically set when this option is used. Note 2: the request content data are applied per each “integrated” URL call.
[-expectHttpCode <number>]	Optional. Allows to configure which HTTP status code is expected for the last executed URL call. The default value is 200 (ok). In case of a mismatch checkURL will return with the exit code -111 (wrong http status code).
[-expectHeaderPattern <text>] … [-expectHeaderPattern <text>]	Optional. Allows to check the received HTTP response header of the last executed URL call by searching a text fragment overall received header lines. This argument can be passed multiple times to check the occurrence – or the absence – of multiple text fragments. The result of multiple checks is logically combined by **AND** conditions. Passing a “normal” text means that the text fragment must be found in the response header. Passing a text enfolded by **![**<text>**]** means the response header must **NOT** contain such a text. In case of a mismatch checkURL will return with the exit code -112 (wrong http response header).
[-expectContentPattern <text>] … [-expectContentPattern <text>]	Optional. Allows to check the received response content (the received data) of the last executed URL call by searching a text fragment inside the content. This argument can be passed multiple times to check the occurrence – or the absence – of multiple text fragments. The result of multiple checks is logically combined by **AND** conditions. Passing a “normal” text means that the text fragment must be found in the response content. Passing a text enfolded by **![**<text>**]** means the response content must

	NOT contain such a text. In case of a mismatch checkURL will return with the exit code -113 (wrong response content).
[-responseContentCharset <charset>]	Optional. Sets a specific character set for the received response content, like for example “UTF-8” or “ISO-8859-1”. This option is only expedient in combination with the option - expectContentPattern. The default value for the character set depends on the used operating system.
[-responseContent <file>]	Optional. Saves the received response content data of the last URL call in raw data format into a file (as a stream of bytes). Note 1: if the file already exits it will be first deleted before the URL call is executed. This means that such a file does not exist if the URL call completely fails (for example when a timeout occurs). Note 2: the size of the file respectively the size of the received content data must not exceed 128 megabytes. For receiving larger data, up to 2 gigabytes, you must additionally use the option -storeMaxResponseContent
[-limitMaxResponseContent <number of bytes>]	Optional. Aborts further receiving of the response content data during the execution of the URL call(s) if a maximum size of bytes is exceeded. The further processing of the URL call will be continued normally which means that no error is reported if the limit is exceeded.
[-storeMaxResponseContent <number of bytes>]	Optional. In contrast to the option -limitMaxResponseContent using this option effects that all response content data are fully received, but that only a part of them are stored. This option can be used to verify the download of large response content data up to 2 gigabytes (for example ISO disk-images or movies). Note: the real size of the stored response content data may be up to 32 kilobytes smaller or lager than specified. Therefore using a value smaller than 32768 bytes may have the result that 0 (zero) bytes of data are stored.
[-basicAuth <username>:<password>]	Optional. Allows to apply HTTP basic authentication for the URL call(s). Both, the user name and the password must be set in the same argument value, separated by a colon.
[-digestAuth <username>:<password>]	Optional. Allows to apply HTTP digest authentication for the URL call(s). Both, the user name and the password must be set in the same argument value, separated by a colon.
[-pkcs12File <file>]	Optional. Allows to apply a client certificate, stored in PKCS#12 format, for the URL call(s).
[-pkcs12Password <password>]	Optional. Allows to specify the password for the PKCS#12 file (see option - pkcs12File). The default value is an empty string.
[-proxy <host>:<port>]	Optional. Allows to set an outgoing HTTP/S proxy server. Both, the host name and the TCP/IP port must be set in the same argument value, separated by a colon.
[-proxyAuth <username>:<password>]	Optional. Allows to apply HTTP basic authentication for the outgoing proxy server. Both, the user name and the password must be set in the same argument value, separated by a colon.

Success exit code: 0

Error exit codes are usually -110 (url call failed), -111 (wrong http status code), -112 (wrong http response header) or -113 (wrong response content). However, any other error exit code may also be returned, like for example -95 (exec agent not reachable).

Example:
java -Xmx256m PrxJob checkUrl "Remote Exec Agent" http://www.apica.se

Note 1: If the web server cannot be reached because of establishing the network connection fails – or also if a timeout has occurred – the exit code is set as usual to -110, but the value of the returned XML tag **<finalUrlStatusCode>** contains additional information about why the URL call has failed. In addition to "normal" HTTP status codes, which range from 100..599 (see RFC 2616), the **<finalUrlStatusCode>** tag may contain some additional HTTP status codes in error situations that are not directly related to the HTTP protocol. These additional status codes have all negative values:

Additional HTTP Status Code	Meaning
-98	An internal network error occurred on the client side (on the Exec Agent). Commonly there are not enough free TCP/IP client sockets available. The TCP/IP configuration of the operating system on which the Exec Agent runs must be tuned to solve this problem.
-97	A Java out of memory error occurred.
-11	The network connection to an outgoing SSL proxy server has failed.
-10	Unknown host. DNS problem or wrong hostname.
-9	Unable to open the network connection to the web server (connection refused, the web server is offline).
-8	The web server has first accepted, but later closed/aborted the network connection, before all response data have been transmitted (connection aborted by peer). Usually the web server is overloaded in such a case.
-7	The web server response violates the HTTP protocol - invalid protocol data have been received.
-2	Request timeout expired - no response from the web server.

Note 2: If a timeout has occurred (see note 1, status code = -2) – but also in case of many other error types – the XML tag **<finalUrlExecutionStep>** contains an additional hint about the internal execution step of the URL call (state of the URL call) measured at the time when the error was detected. Possible values are:

URL Execution Step	Meaning
10	**Resolve DNS Name**: The URL call failed during resolving the DNS name of the web server.
0	**Open Network Connection to Proxy**: The URL call failed during the opening of a network connection to an outgoing proxy server.
1	**Open Network Connection**: The URL call failed during the opening of a network connection to the web server.
11	**SSL/TLS Handshake:** The URL call failed during executing a SSL/TLS handshake with the web server.
2	**Transmit HTTP Request**: The URL call failed during the transmission of the HTTP request data.
3	**Wait for Server Response**: The URL call failed while waiting for the first byte of the HTTP response data from the web server.
4	**Receive HTTP Header**: The URL call failed while receiving the HTTP response header from the web server.
5	**Receive Content**: The URL call failed while receiving the HTTP response content from the web server (HTML data, images, etc...)
6	**Close Network Connection**: The URL call failed while closing the network connection to

	the web server.
7	**All Done**: The URL call itself completed successfully (all data have been transmitted and received), but the received HTTP status code was incorrect, or an error was detected inside the received response header or inside the received content data.

Note 3: All measured response times are in milliseconds. However, if an URL call fails with an additional HTTP status code (negative value) the measured response times are not valid.

Example:
java -Xmx256m PrxJob checkUrl "Local Exec Agent" http://www.apica.se -ssl tls11 -dnsstatistic

Output:

```
<?xml version="1.0" encoding="UTF-8"?>
<checkUrlResult>
  <proxySnifferVersion>V5.2-T</proxySnifferVersion>
  <exitCode>0</exitCode>
  <exitCodeText>SUCCESS</exitCodeText>
  <finalUrl>https://www.apicasystem.com:443/se/</finalUrl>
  <finalUrlStatusCode>200</finalUrlStatusCode>
  <finalUrlStatusCodeText>OK</finalUrlStatusCodeText>
  <finalUrlExecutionStep>7</finalUrlExecutionStep>
  <finalUrlExecutionStepText>All Done</finalUrlExecutionStepText>
  <finalUrlFailureException></finalUrlFailureException>
  <overallTime>22183</overallTime>
  <overallSize>21414</overallSize>
  <totalRedirections>3</totalRedirections>
  <remoteDate>28 Apr 2015 06:56:02</remoteDate>
  <remoteTimeZone>ECT</remoteTimeZone>
  <remoteTimeStamp>1430196962462</remoteTimeStamp>
  <localTimeStamp>1430196962446</localTimeStamp>
  <urlCallList>
    <urlCall>
      <urlCallNumber>1</urlCallNumber>
      <url>http://www.apica.se:80/</url>
      <urlStatusCode>301</urlStatusCode>
      <urlStatusCodeText>Moved Permanently</urlStatusCodeText>
      <urlExecutionStep>7</urlExecutionStep>
      <urlExecutionStepText>All Done</urlExecutionStepText>
      <urlFailureException></urlFailureException>
      <totalTime>8791</totalTime>
      <totalSize>493</totalSize>
      <performanceDetails>
        <dnsResolveTime>5319</dnsResolveTime>
        <socketConnectTime>3079</socketConnectTime>
        <sslHandshakeTime>-1</sslHandshakeTime>
        <requestTransmitTime>0</requestTransmitTime>
        <responseHeaderWaitTime>393</responseHeaderWaitTime>
        <responseHeaderReceiveTime>0</responseHeaderReceiveTime>
        <responseContentReceiveTime>0</responseContentReceiveTime>
        <requestSize>106</requestSize>
        <responseHeaderSize>203</responseHeaderSize>
        <responseContentSize>184</responseContentSize>
      </performanceDetails>
      <requestHeader>
        <string>GET / HTTP/1.1</string>
        <string>Host: www.apica.se</string>
        <string>Accept-Encoding: gzip,deflate</string>
        <string>Connection: Keep-Alive</string>
        <string>Accept: */*</string>
      </requestHeader>
      <responseHeader>
        <string>HTTP/1.1 301 Moved Permanently</string>
        <string>Server: nginx/1.2.6</string>
        <string>Date: Tue, 28 Apr 2015 04:56:10 GMT</string>
        <string>Content-Type: text/html</string>
        <string>Content-Length: 184</string>
        <string>Connection: keep-alive</string>
        <string>Location: http://www.apicasystem.com/se</string>
      </responseHeader>
    </urlCall>
    <urlCall>
      <urlCallNumber>2</urlCallNumber>
      <url>http://www.apicasystem.com:80/se</url>
      <urlStatusCode>301</urlStatusCode>
      <urlStatusCodeText>Moved Permanently</urlStatusCodeText>
      <urlExecutionStep>7</urlExecutionStep>
      <urlExecutionStepText>All Done</urlExecutionStepText>
```

```
      <urlFailureException></urlFailureException>
      <totalTime>1391</totalTime>
      <totalSize>296</totalSize>
      <performanceDetails>
        <dnsResolveTime>953</dnsResolveTime>
        <socketConnectTime>65</socketConnectTime>
        <sslHandshakeTime>-1</sslHandshakeTime>
        <requestTransmitTime>0</requestTransmitTime>
        <responseHeaderWaitTime>373</responseHeaderWaitTime>
        <responseHeaderReceiveTime>0</responseHeaderReceiveTime>
        <responseContentReceiveTime>-1</responseContentReceiveTime>
        <requestSize>115</requestSize>
        <responseHeaderSize>181</responseHeaderSize>
        <responseContentSize>0</responseContentSize>
      </performanceDetails>
      <requestHeader>
        <string>GET /se HTTP/1.1</string>
        <string>Host: www.apicasystem.com</string>
        <string>Accept: */*</string>
        <string>Accept-Encoding: gzip,deflate</string>
        <string>Connection: Keep-Alive</string>
      </requestHeader>
      <responseHeader>
        <string>HTTP/1.1 301 Moved Permanently</string>
        <string>Content-Type: text/html</string>
        <string>Location: https://www.apicasystem.com/se</string>
        <string>Content-Length: 0</string>
        <string>Date: Tue, 28 Apr 2015 04:56:12 GMT</string>
        <string>Connection: keep-alive</string>
      </responseHeader>
    </urlCall>
    <urlCall>
      <urlCallNumber>3</urlCallNumber>
      <url>https://www.apicasystem.com:443/se</url>
      <urlStatusCode>301</urlStatusCode>
      <urlStatusCodeText>Moved Permanently</urlStatusCodeText>
      <urlExecutionStep>7</urlExecutionStep>
      <urlExecutionStepText>All Done</urlExecutionStepText>
      <urlFailureException></urlFailureException>
      <totalTime>11927</totalTime>
      <totalSize>438</totalSize>
      <performanceDetails>
        <dnsResolveTime>-1</dnsResolveTime>
        <socketConnectTime>53</socketConnectTime>
        <sslHandshakeTime>11836</sslHandshakeTime>
        <requestTransmitTime>0</requestTransmitTime>
        <responseHeaderWaitTime>38</responseHeaderWaitTime>
        <responseHeaderReceiveTime>0</responseHeaderReceiveTime>
        <responseContentReceiveTime>-1</responseContentReceiveTime>
        <requestSize>115</requestSize>
        <responseHeaderSize>323</responseHeaderSize>
        <responseContentSize>0</responseContentSize>
      </performanceDetails>
      <requestHeader>
        <string>GET /se HTTP/1.1</string>
        <string>Host: www.apicasystem.com</string>
        <string>Accept: */*</string>
        <string>Accept-Encoding: gzip,deflate</string>
        <string>Connection: Keep-Alive</string>
      </requestHeader>
      <responseHeader>
        <string>HTTP/1.1 301 Moved Permanently</string>
        <string>Server: nginx</string>
        <string>Content-Type: text/html; charset=UTF-8</string>
        <string>X-Pingback: https://www.apicasystem.com/se/xmlrpc.php</string>
        <string>Location: https://www.apicasystem.com/se/</string>
        <string>X-Powered-By: EasyEngine 3.0.4</string>
        <string>rt-Fastcgi-Cache: MISS</string>
        <string>Content-Length: 0</string>
        <string>Date: Tue, 28 Apr 2015 04:56:24 GMT</string>
        <string>Connection: keep-alive</string>
      </responseHeader>
    </urlCall>
    <urlCall>
      <urlCallNumber>4</urlCallNumber>
      <url>https://www.apicasystem.com:443/se/</url>
      <urlStatusCode>200</urlStatusCode>
      <urlStatusCodeText>OK</urlStatusCodeText>
      <urlExecutionStep>7</urlExecutionStep>
      <urlExecutionStepText>All Done</urlExecutionStepText>
      <urlFailureException></urlFailureException>
      <totalTime>74</totalTime>
```

```
      <totalSize>20187</totalSize>
      <performanceDetails>
        <dnsResolveTime>-1</dnsResolveTime>
        <socketConnectTime>-1</socketConnectTime>
        <sslHandshakeTime>-1</sslHandshakeTime>
        <requestTransmitTime>0</requestTransmitTime>
        <responseHeaderWaitTime>73</responseHeaderWaitTime>
        <responseHeaderReceiveTime>0</responseHeaderReceiveTime>
        <responseContentReceiveTime>1</responseContentReceiveTime>
        <requestSize>116</requestSize>
        <responseHeaderSize>372</responseHeaderSize>
        <responseContentSize>19699</responseContentSize>
      </performanceDetails>
      <requestHeader>
        <string>GET /se/ HTTP/1.1</string>
        <string>Host: www.apicasystem.com</string>
        <string>Accept: */*</string>
        <string>Accept-Encoding: gzip,deflate</string>
        <string>Connection: Keep-Alive</string>
      </requestHeader>
      <responseHeader>
        <string>HTTP/1.1 200 OK</string>
        <string>Server: nginx</string>
        <string>Content-Type: text/html; charset=UTF-8</string>
        <string>X-Pingback: https://www.apicasystem.com/se/xmlrpc.php</string>
        <string>Link: &lt;https://www.apicasystem.com/se/&gt;; rel=shortlink</string>
        <string>X-Powered-By: EasyEngine 3.0.4</string>
        <string>rt-Fastcgi-Cache: MISS</string>
        <string>Content-Encoding: gzip</string>
        <string>Content-Length: 19699</string>
        <string>Date: Tue, 28 Apr 2015 04:56:24 GMT</string>
        <string>Connection: keep-alive</string>
        <string>Vary: Accept-Encoding</string>
      </responseHeader>
    </urlCall>
  </urlCallList>
</checkUrlResult> >
```

Important Hint: The exit code should not be extracted from the XML data because no XML data are written to stdout when it was not possible to execute the URL call – for example when the Exec Agent was not reachable. Therefore the exit status should always be extracted from the corresponding shell or operating system environment variable (for example %ERRORLEVEL% on Windows systems).

XML data are only available if the exit code contains one of the following values: 0 (success), -110 (url call failed), -111 (wrong http status code), -112 (wrong http response header) or -113 (wrong response content).

4.4.3.8 getExecAgentDefaultDirectory

Writes the name of the default directory of an Exec Agent to **stdout**.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
[-outfile <file name>]	Optional. If this argument is used the name of the directory is written to an arbitrary file instead to stdout.

Success exit code: 0

Example:
java PrxJob getExecAgentDefaultDirectory "blade2"

Output:
/usr/local/prxsniff

4.4.3.9 getExecAgentJobsDirectory

Writes the name of the (main-) directory of an Exec Agent in which all Exec Agent Jobs are stored to **stdout**.

Note: this directory can be configured when starting an Exec Agent by using the startup option "-jobdir" (see chapter 3.3). If this directory is not configured the Exec Agent uses the default temporary directory of the operating system.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
[-outfile <file name>]	Optional. If this argument is used the name of the directory is written to an arbitrary file instead to stdout.

Success exit code: 0

Example:
java PrxJob getExecAgentJobsDirectory "blade2"

Output:
/var/tmp/PrxExecAgentJobs

4.4.3.10 getExecAgentJobDirectory

Writes the name of the working directory of an Exec Agen Job to **stdout**.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<job id>	Exec Agent job id
[-outfile <file name>]	Optional. If this argument is used the name of the directory is written to an arbitrary file instead to stdout.

Success exit code: 0

Example:
java PrxJob getExecAgentJobDirectory "blade2" 122

Output:
/var/tmp/PrxExecAgentJobs/job_122

4.4.3.11 getDirectoryFileList

Writes all file names of an arbitrary directory of an Exec Agent to **stdout**.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<remote directory>	Directory name on Exec Agent (relative or absolute path)
[-includeDirectories]	Optional. If this argument is used the name of all subdirectories is also included in the result. If this argument is not set the result contains only names of "normal" files.
[-outfile <file name>]	Optional. If this argument is used the name of the directory is written to an arbitrary file instead to stdout.

Success exit code: 0 (if the directory exists and can be read on the Exec Agent)

Example:
java PrxJob getDirectoryFileList "blade2" "/usr/local/prxsniff"

Output:
.bash_history
prxsniff.jar
prxsniff.key

4.4.3.12 fileExists

Checks if a file or a directory exists on the Exec Agent.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<remote file | remote directory>	File name or directory name on the Exec Agent

Success exit code: 0 (if the file or the directory exists on the Exec Agent)

4.4.3.13 createDirectory

Creates a directory on the Exec Agent.

Note: this command requires that the Exec Agent was started with the startup option **-enableRemoteOsCommands**. If this startup option is not set the command returns always with an error exit code.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<remote directory>	Directory name on the Exec Agent (relative or absolute path)

Success exit code: 0 (if the directory is successfully created on the Exec Agent)

4.4.3.14 deleteFile

Deletes a file or an empty directory on the Exec Agent.

Note: this command requires that the Exec Agent was started with the startup option **-enableRemoteOsCommands**. If this startup option is not set the command returns always with an error exit code.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<remote file | empty remote directory>	File name or directory name on the Exec Agent (relative or absolute path)

Success exit code: 0 (if the file or directory is successfully deleted on the Exec Agent)

4.4.3.15 getOsFilePathSeparator

Writes the file path separator character of the operating system on which the Exec Agent runs to **stdout**. The file path separator character is "\" for Windows systems and "/" for Unix-Like systems.

Argument	Description
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
[-outfile <file name>]	Optional. If this argument is used the file path separator character is written to an arbitrary file instead to stdout.

Success exit code: 0

4.4.3.16 uploadFile

Uploads a file to an Exec Agent. Restriction: the maximum size of a file which can be uploaded is limited to 128 MB.

Note: this command requires that the Exec Agent was started with the startup option **-enableRemoteOsCommands**. If this startup option is not set the command returns always with an error exit code.

Argument	**Description**
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<local file>	Local file (relative or absolute path)
<remote file>	Arbitrary remote file name (relative or absolute path). The remote file will be created or replaced.

Success exit code: 0 (if the file is successfully uploaded to the Exec Agent)

4.4.3.17 downloadFile

Downloads a file from an Exec Agent. Restriction: the maximum size of a file which can be downloaded is limited to 128 MB.

Note: this command requires that the Exec Agent was started with the startup option **-enableRemoteOsCommands**. If this startup option is not set the command returns always with an error exit code.

Argument	**Description**
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<local file>	Arbitrary local file name (relative or absolute path). The local file will be created or replaced.
<remote file>	Remote file (relative or absolute path)

Success exit code: 0 (if the file is successfully downloaded from the Exec Agent)

4.4.3.18 uploadFileToJobDirectory

Uploads a file to an Exec Agent into the working directory of a load test job. Restriction: the maximum size of a file which can be uploaded is limited to 128 MB.

Note: this command requires that the Exec Agent was started with the startup option **-enableRemoteOsCommands**. If this startup option is not set the command returns always with an error exit code.

Argument	**Description**
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<job id>	Exec Agent job id
<local file>	Local file (relative or absolute path)
<remote file name>	Arbitrary remote file name – without any path. The remote file will be created or replaced. The following characters are not allowed to be part of a remote file name: .., /, \, :, ~, ^, *, $, @, ", ‘, %, `, ?, &, !

Success exit code: 0 (if the file is successfully uploaded to the Exec Agent)

Example:
java PrxJob uploadFileToJobDirectory "Remote Exec Agent" 456 "C:\data\t.txt" x.txt

4.4.3.19 downloadFileFromJobDirectory

Downloads a file from a working directory of an Exec Agent job. Restriction: the maximum size of a file which can be downloaded is limited to 128 MB.

Note: this command requires that the Exec Agent was started with the startup option **-enableRemoteOsCommands**. If this startup option is not set the command returns always with an error exit code.

Argument	**Description**
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<job id>	Exec Agent job id
<local file>	Arbitrary local file name (relative or absolute path). The local file will be created or replaced.
<remote file name>	Remote file – without any path. The following characters are not allowed to be part of a remote file name: .., /, \, :, ~, ^, *, $, @, ", ‘, %, `, ?, &, !

Success exit code: 0 (if the file is successfully downloaded from the Exec Agent)

Example:
java PrxJob downloadFileFromJobDirectory"Remote Exec Agent" 456 "C:\data\result_1.txt" result.txt

4.4.3.20 getOsType

Writes the name of the operating system on which the Exec Agent runs to **stdout**.

Argument	**Description**
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
[-outfile <file name>]	Optional. If this argument is used the name of the operating system is written to an arbitrary file instead to stdout.

Success exit code: 0

Example:
java PrxJob getOsType "blade2"

Output:
SunOS

4.4.3.21 execOsCommand

Executes an operating system command remotely on an Exec Agent and writes the output of the remotely executed command to **stdout**.

Note: this command requires that the Exec Agent was started with the startup option **-enableRemoteOsCommands**. If this startup option is not set the command returns always with an error exit code.

Argument	**Description**
<exec agent name>	Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description")
<timeout seconds>	Max. time to wait for the completion of the operating system command. If this time is exceeded the command returns with an error exit code.
<os command> [<arg 1>..<arg n>]	Operating system command, inclusive all arguments
[-remdir <remote directory>]	Optional. Working directory of the executed operating system command on the Exec Agent. If this option is not set then the default directory of the Exec Agent will be used as working directory. Note: this optional argument has to be placed subsequently to the operating system command.
[-setosenv <variable name>=<value>]	Optional. Adds an additional variable to the environment of the remotely executed command. This argument can be used several times which allows to add more than one environment variable. If a variable already exists, the value is overwritten. If no equal sign is passed after the variable name, the variable is removed from the environment. Note: this optional argument has to be placed subsequently to the operating system command.
[-infile <file name>]	Optional. The content of the file will be used as input stream for the operating system command. This option is normally not used. Note: this optional argument has to be placed subsequently to the operating system command.
[-outfile <file name>]	Optional. If this argument is used the output of the operating system command is written to an arbitrary file instead to stdout. Note: this optional argument has to be placed subsequently to the operating system command.
[-debug]	Optional. Shows additional data which can be used for debugging purposes. Note: this optional argument has to be placed subsequently to the operating system command.

Exit code: if the operating system command can be started successfully on the Exec agent then the exit code will be as same as the exit code of the operating system command.

For example if the command
"`java PrxJob execOsCommand blade2 30 /usr/sbin/ping 127.0.0.1`" completes successfully on blade2 – which means that the host 127.0.0.1 can be pinged from blade2 – then the exit code is 0. On the other hand the command
"`java PrxJob execOsCommand blade2 30 /usr/sbin/ping 122.22.1.2`" will return with an exit value of 1 which means that the host 122.22.1.2 cannot be pinged from blade2.

The value of the exit code depends on the remotely executed operating system command, or alternatively a PrxJob error code is returned if the operating system command cannot be started remotely on the Exec Agent (PrxJob error code have always a value which is less than zero – negative numbers).

Example 1 (Windows System):
java PrxJob execOsCommand "Windows Exec Agent" 60 cmd.exe /c tracert www.proxy-sniffer-com

Example 2 (Unix-like System):
java PrxJob execOsCommand "Solaris Exec Agent" 60 /usr/sbin/traceroute www.proxy-sniffer-com

Example 3 (Unix-like System):
java PrxJob execOsCommand "blade2" 60 ls -al /usr -outfile x.txt

4.4.4 Cluster Utility Commands

4.4.4.1 getClusterList

Writes the names of all load releasing clusters to **stdout**.

Argument	Description
[-outfile <file name>]	Optional. If this argument is used the names of all clusters are written to an arbitrary file instead to stdout.

Success exit code: 0

Example:
java PrxJob getClusterList

Output:
c1
c2

4.4.4.2 getClusterExecAgentList

Writes the names of all Exec Agents of a load releasing cluster to **stdout**.

Argument	Description
<cluster name>	Name of the cluster, as defined in the "Exec Agent Network Configuration" menu
[-pingVersion]	Optional. Verifies additionally if the Exec Agent processes are alive (reachable) and appends the ZebraTester product version of the corresponding Exec Agent to the output (separated by a semicolon) Note: if an Exec Agent is unreachable the value "unreachable" is shown instead of the ZebraTester product version.
[-outfile <file name>]	Optional. If this argument is used the output is written to an arbitrary file instead to stdout.

Success exit code: 0

Example:
java PrxJob getClusterExecAgentList c2 -pingVersion

Output:
Local Exec Agent;V4.3-F
Remote Exec Agent;unreachable

4.4.4.3 pingCluster

Checks if all Exec Agent processes of a load releasing cluster are alive and writes some data about the Exec Agents (cluster members) to **stdout**.

Argument	Description
<cluster name>	Name of the cluster, as defined in the "Exec Agent Network Configuration" menu

Success exit code: 0 (if all Exec Agent processes of a load releasing cluster are alive)

Example 1:
java PrxJob pingCluster c3

Output 1:
cluster member 1: "Local Exec Agent" (Windows XP 5.1 / Java 1.7.0_72 / ZebraTester V4.2-N)
cluster member 2: "Exec Agent II" (Windows XP 5.1 / Java 1.7.0_72 / ZebraTester V4.2-N)
cluster member 3: "Exec Agent III" (Windows XP 5.1 / Java 1.7.0_72 / ZebraTester V4.2-N)

Example 2:
java PrxJob pingCluster c2

Output 2:
cluster member 1: "Local Exec Agent" (Windows XP 5.1 / Java 1.7.0_72 / ZebraTester V4.3-F)
cluster member 2: "Remote Exec Agent" *** error: not reachable ***

D:\ZebraTester>echo %ERRORLEVEL%
-85

4.4.4.4 getClusterExecAgentNames

Writes the names of all cluster members (Exec Agents) to **stdout**.

Note: this command is deprecated. We recommend that you use the command **getClusterExecAgentList**.

Argument	Description
<cluster name>	Name of the cluster, as defined in the "Exec Agent Network Configuration" menu
[-outfile <file name>]	Optional. If this argument is used the names of all cluster members are written to an arbitrary file instead to stdout.

Success exit code: 0

Example:
java PrxJob getClusterExecAgentNames c2

Output:
Local Exec Agent
Remote Exec Agent

Note: The output of this command can be used by a script as input to do some tasks for all Exec Agents of a cluster. Windows script example (*.bat file):

```
java PrxJob getClusterExecAgentNames c2 -outfile members.txt
for /F "delims=" %%i in (members.txt) do java PrxJob pingExecAgent "%%i"
```

4.4.4.5 execOsCommandCluster

Executes an operating system command remotely on all Exec Agents of a load releasing cluster and writes the output of the remotely executed commands to **stdout**. This command can only be used if all of the cluster members (Exec Agents) are operated under the same operating system.

Note 1: this command requires that the Exec Agent was started with the startup option **-enableRemoteOsCommands**. If this startup option is not set the command returns always with an error exit code.

Note 2: in contrast to the similar command **execOsCommand**, this command does not return the exit code of the remotely executed operating system command(s). Instead of this the exit code contains only a hint if the operating system command have been successfully started on all cluster members.

Argument	**Description**
<cluster name>	Name of the cluster, as defined in the "Exec Agent Network Configuration" menu
<timeout seconds>	Max. time to wait for the completion of the operating system command. Note: the timeout is applied per cluster member.
<os command> [<arg 1>..<arg n>]	Operating system command, inclusive all arguments
[-remdir <remote directory>]	Optional. Working directory of the executed operating system command on the Exec Agents. If this option is not set then the default directory of the Exec Agents will be used as working directory. Note: this optional argument has to be placed subsequently to the operating system command.
[-setosenv <variable name>=<value>]	Optional. Adds an additional variable to the environment of the remotely executed command. This argument can be used several times which allows to add more than one environment variable. If a variable already exists, the value is overwritten. If no equal sign is passed after the variable name, the variable is removed from the environment. Note: this optional argument has to be placed subsequently to the operating system command.
[-infile <file name>]	Optional. The content of the file will be used as input stream for the operating system command. This option is normally not used. Note: this optional argument has to be placed subsequently to the operating system command.
[-outfile <file name>]	Optional. If this argument is used the output of the operating system command are written to an arbitrary file instead to stdout. Note: this optional argument has to be placed subsequently to the operating system command.

Success exit code: 0
Note: 0 means only that all operating system commands have been successfully started, but it does not mean that they have also been successfully completed! The exit codes of all executed operating system commands are written to **stdout**, or alternatively to the "outfile".

Example:
java PrxJob execOsCommandCluster c2 30 cmd.exe /c netstat

Output:

```
cluster member 1: "Local Exec Agent"  command exit value = 0

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    fieldtest:1809         localhost:1810         ESTABLISHED
  TCP    fieldtest:1810         localhost:1809         ESTABLISHED
  TCP    fieldtest:1812         localhost:1814         ESTABLISHED
  TCP    fieldtest:1814         localhost:1812         ESTABLISHED
  TCP    fieldtest:2773         localhost:2774         ESTABLISHED
  TCP    fieldtest:2774         localhost:2773         ESTABLISHED
  TCP    fieldtest:2775         localhost:2776         ESTABLISHED
  TCP    fieldtest:2776         localhost:2775         ESTABLISHED


cluster member 2: "Exec Agent II"  command exit value = 0

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    dynatest:3389          192.16.4.32:3107       ESTABLISHED
  TCP    dynatest:7993          192.16.4.32:3183       ESTABLISHED
```

4.4.5 Other Commands

4.4.5.1 sleep

Suspends the execution of the script for a number of seconds. This command can be used on Windows systems where no sleep utility is available. On Unix-like systems, we recommend that you use the native sleep command.

Argument	Description
<seconds>	Number of seconds

Success exit code: 0

4.4.5.2 version

Writes the local installed ZebraTester version or the version installed on an Exec Agent to stdout.

Argument	Description
[<exec agent name>]	Optional. Name of the Exec Agent, as defined in the "Exec Agent Network Configuration" menu (shown as "description") If no Exec Agent name is specified the local installed product version is displayed.

Success exit code: 0

Example:
java PrxJob version

```
V4.3-B
```

4.4.5.3 zip

Create a new zip archive that can be used to execute a load test. Note that this command does not support directories within zip archives, meaning that all zipped files are stored at the top level inside the zip archive without an absolute file path (in accordance with the ZebraTester convention for zipped load test programs). However, the zip archive can contain other zipped files or jar files.
Hint: the generic Java memory parameter (-Xmx) must be configured large enough to hold the uncompressed data of all files added to the zip archive.

Argument	Description
<file 1> [,<file 2> ..,<file n>]	Relative or absolute path of the files that have to be added to the zip archive, separated by commas. The files are stored always as plain file names inside the zip archive (without the absolute file path).
-out < zip-archive-file>	File name of the zip archive (relative or absolute path).

Success exit code: 0

Example:
java -Xmx256m PrxJob zip Test_01.class, My_Plugin.class, "C:\libs\My_library.jar" -out Test_01.zip

4.4.5.4 unzip

Extract all files or selected files from a zip archive – or list the files of a zip archive. Note that this command does not support directories within zip archives. However, the zip archive can contain other zipped files or jar files.
Hint: the generic Java memory parameter (-Xmx) must be configured large enough to hold the uncompressed data of all files of the zip archive.

Argument	Description
<zip-archive-file>	File name of the zip archive (relative or absolute path).
[-select <plain-file-name>]	Optional. Allows to extract a particular file from the zip archive, specified by its plain file name. Note: if this option is not used all files are extracted from the zip archive. This option can be used several times to extract several files by only one call of the unzip command.
[-outdir <target-directory-path>]	Optional. Extract all files – or the selected files (see -select) – to a target directory. Note: if this option is not used the files are extracted to the default working directory.
[-listonly]	Optional. Write a list of files of the zip archive (plain file names) to stdout, but does not extract the files.

Success exit code: 0

Example:
java -Xmx256m PrxJob unzip Test_01.zip -outdir "C:\Scratch\a"

4.4.5.5 importNetworkConfiguration

Import definitions of Exec Agents and "Exec Agent Clusters" from a data file.

Data File Syntax - List of Supported Import Commands:

- **delete all**
- **delete ExecAgent** "<description>" *note: wildcards such as * and ? are supported*
- **delete Cluster** "<name>" *note: wildcards such as * and ? are supported*
- **add ExecAgent** "<description>" -host <IP address or DNS name> [-port <number>] [-protocol <plain | http | https>] [-loadFactor <floating-point number> [1]] [-auth <base64(username:password)>] [-proxyHost <IP address or DNS name>] [-proxyPort <number>] [-proxyAuth <base64(username:password)>]
- **add Cluster** contains "<exec agent description 1>" "<exec agent description 2>" .. "<exec agent description N>"

[1] = non recommended option
Note: When importing data, delete commands are always executed first – independently of the line position in the data file.

Argument	Description
<import file>	Relative or absolute path of the data file that contains definitions of Exec Agents and "Exec Agent Clusters".

Success exit code: 0

Note: the content of the files **execAgent.dat** and **execAgentCluster.dat** is updated. See also chapter 6: "Installed Files".

Example:
java PrxJob importNetworkConfiguration PrxNetworkConfiguration_2.txt

Example of Data File:

```
add ExecAgent "pt-loadagent001-01" -host pt-loadagent001.d-fischer.com -port 7993 -protocol plain
add ExecAgent "pt-loadagent002-01" -host pt-loadagent002.d-fischer.com -port 7993 -protocol plain
add ExecAgent "pt-loadagent003-01" -host pt-loadagent003.d-fischer.com -port 7993 -protocol plain
add ExecAgent "pt-loadagent004-01" -host pt-loadagent004.d-fischer.com -port 7993 -protocol plain

add Cluster "pt-loadagents-cluster01" contains "pt-loadagent001-01" "pt-loadagent002-01" "pt-loadagent003-01"
"pt-loadagent004-01"
```

4.4.5.6 getApicaPMAMonitoringTemplateList

Get a list of all defined templates for Apica Performance Monitoring Agent Controller(s).

Argument	**Description**
[-outfile <file name>]	Optional. If this argument is used the output of the command is written to an arbitrary file instead to stdout.

Success exit code: 0

Example:
java PrxJob getApicaPMAMonitoringTemplateList

Output:
TEMPLATE_01.xml;All OS performance data of Web server no. 1
TEMPLATE_02.xml;MySQL database performance data

Note: each line contains the template name, followed by the description of the template, separated by a (first-occurred) semicolon character.

4.4.5.7 getGuiLicenseKey

Get the ZebraTester GUI license key from the running WebAdmin process (or from the running ZebraTester console process).

Argument	**Description**
[-outfile <file name>]	Optional. If this argument is used the output of the command is written to an arbitrary file instead of to stdout.

Success exit code: 0

Example:
java PrxJob getGuiLicenseKey

Output:
IRkA-CwMQ-AAAU-UT2G-Er0@

4.4.5.8 getWebAdminVersion

Get the product version from the running WebAdmin process (or from the running ZebraTester console process).

Argument	**Description**
[-outfile <file name>]	Optional. If this argument is used the output of the command is written to an arbitrary file instead of to stdout.

Success exit code: 0

Example:
java PrxJob getWebAdminVersion

Output:
V5.1-D

4.4.5.9 getPrxJobVersion

Get the product version of the PrxJob utility.

Argument	**Description**
[-outfile <file name>]	Optional. If this argument is used the output of the command is written to an arbitrary file instead of to stdout.

Success exit code: 0

Example:
java PrxJob getPrxJobVersion

Output:
V5.1-D

4.5 Windows Script Examples

```
@ECHO OFF

REM ------------------------------------------------------------------------------------------------
REM Description: start a load test job and wait for job completion.
REM Generate a PDF detail report
REM ------------------------------------------------------------------------------------------------

PATH=C:\Program Files\ProxySniffer\jre\bin;%PATH%
SET CLASSPATH=.;C:\Program Files\ProxySniffer\prxsniff.jar;
C:\Program Files\ProxySniffer\iaik_jce_full.jar;
C:\Program Files\ProxySniffer\iaikPkcs11Provider.jar;C:\Program Files\ProxySniffer

SET EXECAGENT_NAME=Local Exec Agent
SET LOAD_TEST_PROGRAM=C:\Program Files\ProxySniffer\MyTests\Scripts\Test01.zip
SET LOAD_TEST_ARGUMENTS=-u 3 -d 30 -t 60 -nolog -annotation "first test run"
SET RESULT_FILE=Test01.prxres


REM -- transmit job --
java PrxJob transmitJob "%EXECAGENT_NAME%" "%LOAD_TEST_PROGRAM%" %LOAD_TEST_ARGUMENTS%
SET JOB_ID=%ERRORLEVEL%
IF %JOB_ID% LSS 0 (
  ECHO Error %JOB_ID%: unable to define job
  GOTO ABORT_END
)
ECHO job %JOB_ID% transmitted


REM -- start job --
java PrxJob startJob "%EXECAGENT_NAME%" %JOB_ID%
SET STATUS=%ERRORLEVEL%
IF %STATUS% NEQ 0 (
  ECHO Error %STATUS%: unable to start job %JOB_ID%
  GOTO ABORT_END
)
ECHO job %JOB_ID% started on %EXECAGENT_NAME%


REM -- wait for job completion --
java PrxJob waitForJobCompletion "%EXECAGENT_NAME%" %JOB_ID%
SET STATUS=%ERRORLEVEL%
IF %STATUS% NEQ 0 (
  ECHO Error %STATUS%: unable to wait for job %JOB_ID%
  GOTO ABORT_END
)
ECHO job %JOB_ID% completed


REM -- acquire job result file --
java PrxJob acquireJobResultFile "%EXECAGENT_NAME%" %JOB_ID% "%RESULT_FILE%"
SET STATUS=%ERRORLEVEL%
IF %STATUS% NEQ 0 (
  ECHO Error %STATUS%: unable to acquire job %JOB_ID% result file
  GOTO ABORT_END
)
ECHO job %JOB_ID% result file acquired


REM -- generate PDF detail report --
java PdfReport clear
java PdfReport loadWithDetailReport %RESULT_FILE%
ECHO job %JOB_ID% PDF detail report created


ECHO job %JOB_ID% successful done
EXIT /B 0


:ABORT_END
ECHO job failed
EXIT /B -1
```

Alternatively, waiting for the job to complete can also written as a loop:

```
REM -- wait for job completion in a loop --
java PrxJob getJobState "%EXECAGENT_NAME%" %JOB_ID%
SET STATUS=%ERRORLEVEL%
:WAIT_JOB
  IF %STATUS% EQU 12 (      REM 12 = JOB COMPLETED
    GOTO END_WAIT_JOB
  )

  java PrxJob sleep 5
  ECHO wait for job %JOB_ID% end ..

  REM java PrxJob abortJob "%EXECAGENT_NAME%" %JOB_ID%      REM optional, break the loop earlier

  java PrxJob getJobStatus "%EXECAGENT_NAME%" %JOB_ID%
  SET STATUS=%ERRORLEVEL%
  GOTO :WAIT_JOB

:END_WAIT_JOB
ECHO job %JOB_ID% completed
```

The following example starts two jobs in parallel, and waits for their completion:

```
@ECHO OFF

REM ------------------------------------------------------------------------------------------
REM Description: start two jobs simultaneous and wait until both jobs are completed.
REM Generate two PDF detail reports
REM ------------------------------------------------------------------------------------------

PATH=C:\Program Files\ProxySniffer\jre\bin;%PATH%
SET CLASSPATH=.;C:\Program Files\ProxySniffer\prxsniff.jar;
C:\Program Files\ProxySniffer\iaik_jce_full.jar;
C:\Program Files\ProxySniffer\iaikPkcs11Provider.jar;C:\Program Files\ProxySniffer

SET EXECAGENT_NAME=Local Exec Agent

SET TEST_PROGRAM_1=C:\Program Files\ProxySniffer\MyTests\Test01.zip
SET TEST_ARGUMENTS_1=-u 1 -d 30 -t 60 -nolog -annotation "concurrent test 1"

SET TEST_PROGRAM_2=C:\Program Files\ProxySniffer\MyTests\Test02.class
SET TEST_ARGUMENTS_2=-u 2 -d 30 -t 60 -nolog -annotation "concurrent test 2"

SET RESULT_FILE_1=C:\Program Files\ProxySniffer\MyTests\Test01.prxres
SET RESULT_FILE_2=C:\Program Files\ProxySniffer\MyTests\Test02.prxres
SET REPORT_FILE_1=C:\Program Files\ProxySniffer\MyTests\Test01.pdf
SET REPORT_FILE_2=C:\Program Files\ProxySniffer\MyTests\Test02.pdf


REM -- transmit job 1 --
java PrxJob transmitJob "%EXECAGENT_NAME%" "%TEST_PROGRAM_1%" %TEST_ARGUMENTS_1%
SET JOB_ID_1=%ERRORLEVEL%
IF %JOB_ID_1% LSS 0 (
  ECHO Error %JOB_ID_1%: unable to define job 1
  GOTO ABORT_END
)
ECHO job %JOB_ID_1% transmitted


REM -- transmit job 2 --
java PrxJob transmitJob "%EXECAGENT_NAME%" "%TEST_PROGRAM_2%" %TEST_ARGUMENTS_2%
SET JOB_ID_2=%ERRORLEVEL%
IF %JOB_ID_2% LSS 0 (
  ECHO Error %JOB_ID_2%: unable to define job 2
  GOTO ABORT_END
)
ECHO job %JOB_ID_2% transmitted


REM -- start job 1 --
java PrxJob startJob "%EXECAGENT_NAME%" %JOB_ID_1%
SET STATUS=%ERRORLEVEL%
IF %STATUS% NEQ 0 (
  ECHO Error %STATUS%: unable to start job %JOB_ID_1%
  GOTO ABORT_END
```



```
)
ECHO job %JOB_ID_1% started on %EXECAGENT_NAME%


REM -- start job 2 --
java PrxJob startJob "%EXECAGENT_NAME%" %JOB_ID_2%
SET STATUS=%ERRORLEVEL%
IF %STATUS% NEQ 0 (
  ECHO Error %STATUS%: unable to start job %JOB_ID_2%
  GOTO ABORT_END
)
ECHO job %JOB_ID_2% started on %EXECAGENT_NAME%


REM -- wait until job 1 and 2 are completed --
java PrxJob waitForJobCompletion "%EXECAGENT_NAME%" %JOB_ID_1% %JOB_ID_2%
SET STATUS=%ERRORLEVEL%
IF %STATUS% NEQ 0 (
  ECHO Error %STATUS%: unable to wait for job 1 and 2 completion
  GOTO ABORT_END
)
ECHO job %JOB_ID_1% and %JOB_ID_2% completed


REM -- acquire job 1 result file and generate PDF detail report --
java PrxJob acquireJobResultFile "%EXECAGENT_NAME%" %JOB_ID_1% "%RESULT_FILE_1%"
SET STATUS=%ERRORLEVEL%
IF %STATUS% NEQ 0 (
  ECHO Error %STATUS%: unable to acquire job %JOB_ID_1% result file
  GOTO ABORT_END
)
java PdfReport clear
java PdfReport loadWithDetailReport "%RESULT_FILE_1%" -output "%REPORT_FILE_1%"


REM -- acquire job 2 result file and generate PDF detail report --
java PrxJob acquireJobResultFile "%EXECAGENT_NAME%" %JOB_ID_2% "%RESULT_FILE_2%"
SET STATUS=%ERRORLEVEL%
IF %STATUS% NEQ 0 (
  ECHO Error %STATUS%: unable to acquire job %JOB_ID_2% result file
  GOTO ABORT_END
)
java PdfReport loadWithDetailReport "%RESULT_FILE_2%" -output "%REPORT_FILE_2%"


ECHO PDF detail reports created
ECHO jobs successful done
EXIT /B 0


:ABORT_END
ECHO jobs failed
EXIT /B -1
```

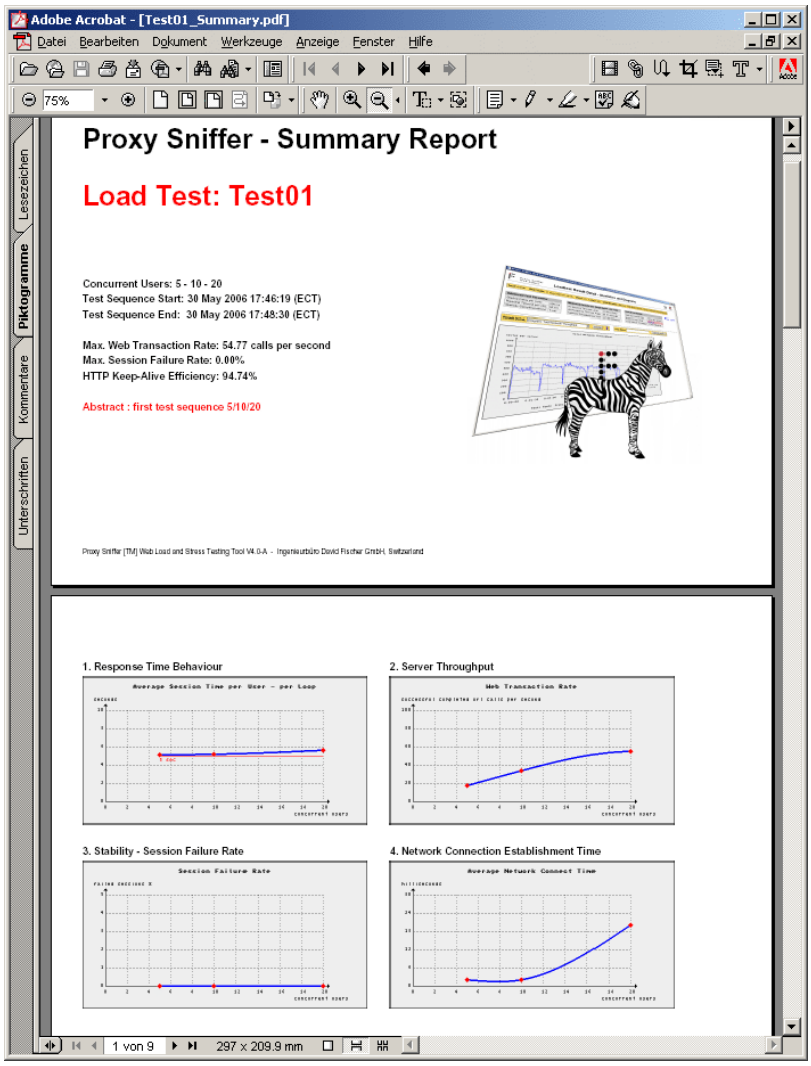
```
C:\Program Files\ProxySniffer\MyTests\ScriptExamples> TestConcurrentJobs.bat
job 16 transmitted
job 17 transmitted
job 16 started on Local Exec Agent
job 17 started on Local Exec Agent
job 16 and 17 completed
PDF detail reports created
jobs successful done
```

This example starts the same load test program several times – with a different number of concurrent users – to measure the load curves of the Web application. At the end, a PDF load curves report is generated:

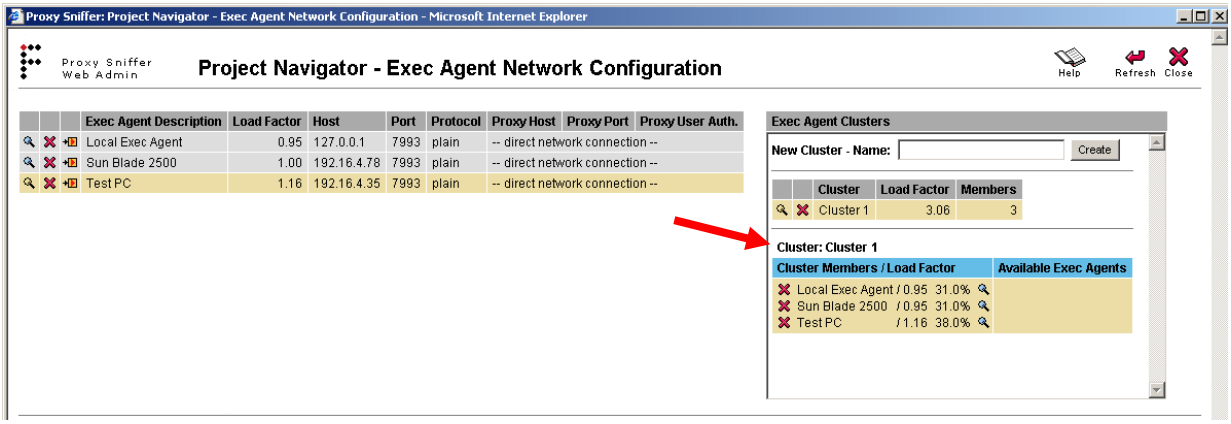
```
@ECHO OFF

REM ------------------------------------------------------------------------------------------
REM Description: execute the same test three times with 5, 10 and 20 users.
REM Generate a PDF summary report
REM ------------------------------------------------------------------------------------------

PATH=C:\Program Files\ProxySniffer\jre\bin;%PATH%
SET CLASSPATH=.;C:\Program Files\ProxySniffer\prxsniff.jar;
C:\Program Files\ProxySniffer\iaik_jce_full.jar;
C:\Program Files\ProxySniffer\iaikPkcs11Provider.jar;C:\Program Files\ProxySniffer

SET EXECAGENT_NAME=Local Exec Agent
SET LOAD_TEST_PROGRAM=C:\Program Files\ProxySniffer\MyTests\Test01.zip

REM -- clear first Web Admin result cache --
java PdfReport clear

SET LOOP_COUNTER=0
SET USERS=1
REM -- loop over serveral load tests --
:TEST_LOOP

     SET/A LOOP_COUNTER=%LOOP_COUNTER% + 1
     IF %LOOP_COUNTER% EQU 1 SET USERS=5
     IF %LOOP_COUNTER% EQU 2 SET USERS=10
     IF %LOOP_COUNTER% EQU 3 SET USERS=20
     IF %LOOP_COUNTER% EQU 4 GOTO TEST_END

     REM -- transmit job --
     java PrxJob transmitJob "%EXECAGENT_NAME%" "%LOAD_TEST_PROGRAM%" -u %USERS% -d 30 -t 60 -
nolog -annotation "test run with %USERS% user"
     SET JOB_ID=%ERRORLEVEL%
     IF %JOB_ID% LSS 0 (
       ECHO Error %JOB_ID%: unable to define job
       GOTO ABORT_END
     )
     ECHO job %JOB_ID% transmitted


     REM -- start job --
     java PrxJob startJob "%EXECAGENT_NAME%" %JOB_ID%
     SET STATUS=%ERRORLEVEL%
     IF %STATUS% NEQ 0 (
       ECHO Error %STATUS%: unable to start job %JOB_ID%
       GOTO ABORT_END
     )
     ECHO job %JOB_ID% started on %EXECAGENT_NAME% - %USERS% user


     REM -- wait for job completion --
     java PrxJob waitForJobCompletion "%EXECAGENT_NAME%" %JOB_ID%
     SET STATUS=%ERRORLEVEL%
     IF %STATUS% NEQ 0 (
       ECHO Error %STATUS%: unable to wait for job %JOB_ID%
       GOTO ABORT_END
     )
     ECHO job %JOB_ID% completed


     REM -- acquire job result file --
     java PrxJob acquireJobResultFile "%EXECAGENT_NAME%" %JOB_ID% "C:\Program
Files\ProxySniffer\MyTests\test01_%USERS%u.prxres"
     SET STATUS=%ERRORLEVEL%
     IF %STATUS% NEQ 0 (
       ECHO Error %STATUS%: unable to acquire job %JOB_ID% result file
       GOTO ABORT_END
     )
     ECHO job %JOB_ID% result file acquired


     REM -- add result to PDF report ---
     java PdfReport load "C:\Program Files\ProxySniffer\MyTests\Test01_%USERS%u.prxres"


     GOTO TEST_LOOP
```

```
:TEST_END

REM -- generate PDF load curves report --
java PdfReport generateLoadCurvesReport "C:\Program Files\ProxySniffer\MyTests\Test01_Summary.pdf"
ECHO PDF load curves report created


ECHO test successful done
EXIT /B 0


:ABORT_END
ECHO test failed
EXIT /B -1
```

```
C:\Program Files\ProxySniffer\MyTests\ScriptExamples> TestJobSequence.bat
job 12 transmitted
job 12 started on Local Exec Agent - 5 user
job 12 completed
job 12 result file acquired
job 13 transmitted
job 13 started on Local Exec Agent - 10 user
job 13 completed
job 13 result file acquired
job 14 transmitted
job 14 started on Local Exec Agent - 20 user
job 14 completed
job 14 result file acquired
PDF summary report created
test successful done
```



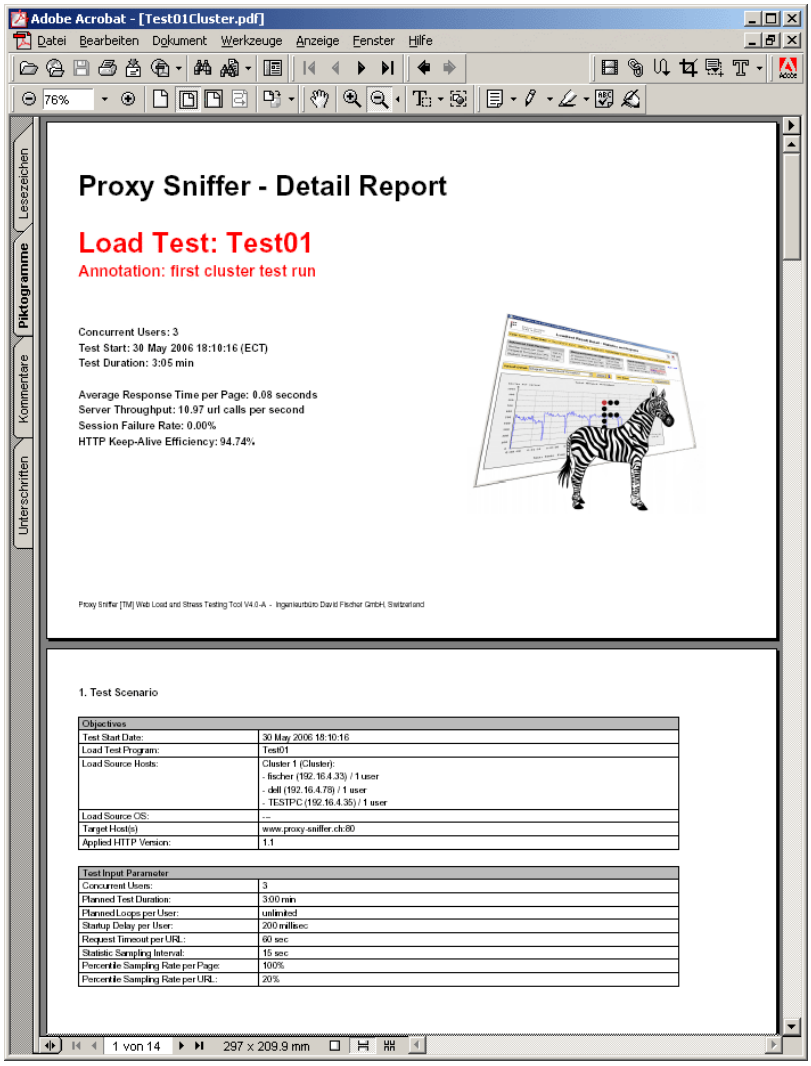
This example executes a cluster job and generates a PDF detail report




```
@ECHO OFF


REM -------------------------------------------------------------------------------------------
REM Description: start a cluster job and wait for job completion. Generate a PDF detail report
REM -------------------------------------------------------------------------------------------

PATH=C:\Program Files\ProxySniffer\jre\bin;%PATH%
SET CLASSPATH=.;C:\Program Files\ProxySniffer\prxsniff.jar;
C:\Program Files\ProxySniffer\iaik_jce_full.jar;
C:\Program Files\ProxySniffer\iaikPkcs11Provider.jar;C:\Program Files\ProxySniffer

SET CLUSTER_NAME=Cluster 1

SET LOAD_TEST_PROGRAM=C:\Program Files\ProxySniffer\MyTests\Test01.zip
SET LOAD_TEST_ARGUMENTS=-u 3 -d 180 -t 60 -nolog -annotation "first cluster test run"
SET RESULT_FILE=C:\Program Files\ProxySniffer\MyTests\Test01Cluster.prxres
SET REPORT_FILE=C:\Program Files\ProxySniffer\MyTests\Test01Cluster.pdf


REM -- transmit cluster job --
java PrxJob transmitClusterJob "%CLUSTER_NAME%" "%LOAD_TEST_PROGRAM%" %LOAD_TEST_ARGUMENTS%
SET CLUSTER_JOB_ID=%ERRORLEVEL%
IF %CLUSTER_JOB_ID% LSS 0 (
  ECHO Error %CLUSTER_JOB_ID%: unable to define cluster job
  GOTO ABORT_END
)
ECHO cluster job %CLUSTER_JOB_ID% transmitted


REM -- start cluster job and split input file "userAccounts.txt" --
java PrxJob startClusterJob "%CLUSTER_NAME%"
   → %CLUSTER_JOB_ID% -split userAccounts.txt "#"
SET STATUS=%ERRORLEVEL%
IF %STATUS% NEQ 0 (
  ECHO Error %STATUS%: unable to start cluster job %CLUSTER_JOB_ID%
  GOTO ABORT_END
)
ECHO cluster job %CLUSTER_JOB_ID% started on %CLUSTER_NAME%


REM -- wait for cluster job completion --
java PrxJob waitForClusterJobCompletion "%EXECAGENT_NAME%" %CLUSTER_JOB_ID%
SET STATUS=%ERRORLEVEL%
IF %STATUS% NEQ 0 (
  ECHO Error %STATUS%: unable to wait for cluster job %CLUSTER_JOB_ID%
  GOTO ABORT_END
)
ECHO cluster job %CLUSTER_JOB_ID% completed


REM -- acquire cluster job result file --
java PrxJob acquireClusterJobResultFile "%CLUSTER_NAME%" %CLUSTER_JOB_ID% "%RESULT_FILE%"
SET STATUS=%ERRORLEVEL%
IF %STATUS% NEQ 0 (
  ECHO Error %STATUS%: unable to acquire cluster job %CLUSTER_JOB_ID% result file
  GOTO ABORT_END
)
ECHO job %CLUSTER_JOB_ID% result file acquired
```

```
REM -- generate PDF detail report --
java PdfReport clear
java PdfReport loadWithDetailReport "%RESULT_FILE%" -output "%REPORT_FILE%"
ECHO cluster job %CLUSTER_JOB_ID% PDF detail report created


ECHO cluster job %CLUSTER_JOB_ID% successful done
EXIT /B 0


:ABORT_END
ECHO cluster job failed
EXIT /B -1
```

```
C:\Program Files\ProxySniffer\MyTests\ScriptExamples> TestClusterJob.bat
cluster job 2 transmitted
cluster job 2 started on Cluster 1
cluster job 2 completed
job 2 result file acquired
cluster job 2 PDF detail report created
cluster job 2 successful done
```



4.6 Mac OS X Script Example (Bash Shell)

```
#!/bin/bash

# set Java CLASSPATH
export
CLASSPATH=.:/Applications/ZebraTester/prxsniff.jar:/Applications/ZebraTester/iaik_jce_full.jar:/Applications/ZebraTester/iaik_ssl.jar:/Applications/ZebraT
ester/iaik_eccelerate.jar:/Applications/ZebraTester/iaikPkcs11Provider.jar:/Applications/ZebraTester

# add Java binaries to PATH
export PATH=/Applications/ZebraTester/jre/bin:$PATH

# change to directory of load test program
cd /Applications/ZebraTester/MyTests

# clear all data in analyse load test menu
java PdfReport clear

# loop over simulated users
# -------------------------
for users in 1 2 5 10
do

# define load test program. Note: if the program is zipped you have to add ".zip" to the program name
loadTestProgram="Test01"
loadTestProgramArgs="-u $users -d 30 -t 60 -sdelay 200 -maxloops 0 -sampling 15 -percpage 100 -percurl 20 -maxerrmem 20 -nolog"

# define exec agent name
execAgentName="Local Exec Agent"

# define load test result file
currentDate=`date "+%d%h%y_%H%M%S"`
loadTestResultFile="`echo $loadTestProgram`_`echo $currentDate`_`echo $users`u.prxres"

# transmit load test job to exec agent
java PrxJob -s transmitJob "$execAgentName" $loadTestProgram $loadTestProgramArgs
prxstat=`cat PRXSTAT`
if [ $prxstat -lt "0" ]; then
  echo "unable to transmit job, status = $prxstat"
  exit 1;
fi
jobId=$prxstat

# start load test job on exec agent
java PrxJob -s startJob "$execAgentName" $jobId
prxstat=`cat PRXSTAT`
if [ $prxstat -ne "0" ]; then
  echo "unable to start job, status = $prxstat"
```

```
  exit 1;
fi
echo "$loadTestProgram started with $users users on $execAgentName, job ID = $jobId"

# wait until job is completed
java PrxJob -s waitForJobCompletion "$execAgentName" $jobId
prxstat=`cat PRXSTAT`
if [ $prxstat -ne "0" ]; then
  echo "unable to wait for job $jobId, status = $prxstat"
  exit 1;
fi
echo "job ID = $jobId completed on $execAgentName"

# acquire load test result file
java PrxJob -s acquireJobResultFile "$execAgentName" $jobId "$loadTestResultFile"
prxstat=`cat PRXSTAT`
if [ $prxstat -ne "0" ]; then
  echo "acquire load test result file, status = $prxstat"
  exit 1;
fi
echo "load test result $loadTestResultFile acquired"

# load result into analyse load test menu
java PdfReport load $loadTestResultFile

# end loop over simulated users
# -----------------------------
done
```

```
# acquire load test result file
java PrxJob -s acquireJobResultFile "$execAgentName" $jobId "$loadTestResultFile"
prxstat=`cat PRXSTAT`
if [ $prxstat -ne "0" ]; then
  echo "unable to acquire load test result file, job id = $jobId, status = $prxstat"
  exit 1;
fi
echo "load test result acquired: $loadTestResultFile"

# generate PDF report
java PdfReport clear
java PdfReport loadWithDetailReport "$loadTestResultFile" -output "$pdfReportFile"
echo "PDF report created: $pdfReportFile
```


5 Supported Time Zones

```
MIT = GMT -11:00  Midway-Islands, Samoa
HST = GMT -10:00  Hawaii
AST = GMT -9:00   Alaska
PST = GMT -8:00   Pacific Time (US and Canada, Tijuana)
MST = GMT -7:00   Mountain Time (US and Canada)
PNT = GMT -7:00   Arizona
CST = GMT -6:00   Central Time (US and Canada)
EST = GMT -5:00   Eastern Time
IET = GMT -5:00   Bogota, Lima, Quito, Indiana
PRT = GMT -4:00   Atlantic
CNT = GMT -3:30   Newfoundland
BET = GMT -3:00   Brasil
CAT = GMT -1:00   Azores, Cape Verde Islands
GMT = GMT         GMT, Casablanca, Monrovia
WET = GMT         Dublin, Edinburg, Lissabon, London
ECT = GMT +1:00   Berlin, Bern, Paris, Madrid, Rom, Wien
EET = GMT +2:00   Athen, Istambul, Helsinki
EAT = GMT +3:00   Bagdad, Kuwait, Riad
NET = GMT +4:00   Abu Dhabi, Tiflis
PLT = GMT +5:00   Islamabad, Karachi
IST = GMT +5:30   New-Dehlhi, Bombay, Calcutta
BST = GMT +6:00   Dakka, Colombo
VST = GMT +7:00   Bangkok, Hanoi, Jakarta
CTT = GMT +8:00   Peking, Hongkong
JST = GMT +9:00   Osaka, Sapporor, Tokio
ACT = GMT +9:30   Darwin
AET = GMT +10:00  Canberra, Melbourne, Sydney
SST = GMT +11:00  Salomon-Islands
NST = GMT +12:00  Auckland, Wellington
```

6 Installed Files / Configuration Files

This chapter contains a list of the most important files installed by the ZebraTester installation kit, and also files which are automatically created at runtime in the installation directory.

File Name	File Type	Windows	Unix	Description
AlertConfig.xml	ASCII	yes	yes	Optional: contains the alert configuration for sending emails and SMS alert notifications.
dataCollector.dat	ASCII	yes	yes	Currently unused (reserved for future use). Created at runtime by the Web Admin GUI.
execAgent.dat	ASCII	yes	yes	Contains the list of configured Exec Agents. This file is modified or created at runtime by the Web Admin GUI. Hint: do not modify this file manually.
execAgentCluster.dat	ASCII	yes	yes	Contains the list of configured Exec Agent Clusters. This file is modified or created at runtime by the Web Admin GUI. Hint: do not modify this file manually.
ExecAgentTicket.dat	ASCII	yes	yes	Contains the Exec Agent license ticket which controls how many concurrent virtual users can be executed by the Exec Agent.
installdir.dat	ASCII	yes	no, OS X: yes	Contains the name of the installation directory itself. Hint: do not modify this file.
InstallExecAgentService.bat	ASCII	yes	no	Utility which allows the installation of the Exec Agent as a Windows Service. You may modify this file before execution.
InstallZebraTesterService.bat	ASCII	yes	no	Utility which allows the installation of the ZebraTester GUI as a Windows Service. You may modify this file before execution.
JavaService.exe	binary	yes	no	Utility which allows you to run a Java program as a Windows service
javaSetup.dat	ASCII	yes	yes	Contains configuration data for the Web Admin GUI and for the Exec Agent. This file is automatically created by the Exec Agent if it does not exist. Manual modifications are only supported on Unix systems on which no Web Admin GUI has been started. Do not modify this file manually on Windows systems. Note: the values in this file can be modified by calling the “Setup Icon” within the Project Navigator menu.
JobControllerProperties.dat	ASCII	yes	yes	Optional. Can be created manually if required. See chapter

File Name	File Type	Windows	Unix	Description
				3.4.1 for further information.
mytests.dat	ASCII	yes	yes	Unix: optional. Contains the directory (or share) used as the top-level directory by the Project Navigator. You may modify (or create) this file manually to keep the MyTests folder in another place. Important Note: after modifying this file, you must clear all cookies in your Web browser because the MyTests directory is also stored in a cookie.
privkey.der	binary	yes	yes	The private key, used by the common CA root certificate "root.cer". Note: it's strongly recommended that you replace this file by your own version (see ZebraTester Installation Guide).
prxsniff.dat	ASCII	yes	yes	Contains various ZebraTester startup options on Windows and Mac OS X systems. The following arguments are supported: -tz <time zone> -dgs a | c -ecc -enableJobOverrideJavaMemory -runtimedatadir <path> -jobdir <directory> -clusterjobdir <directory> -enableRemoteOsCommands -guitimeout <seconds> -execAgentConfigDir <path> -execAgentClusterConfigDir <path> -outboundipfile <file-name> -dnshosts <file-name> -dnssrv <IP-nsrv-1>[,IP-nsrv-N] -dnsfixttl <seconds> -dnsstatistic -debugRecorderSocketPools -debugRecorderPlugins -RESTAPIServer -log4j The -jobdir and the -clusterjob option allows you to set the working directories for the Exec Agent and the Job Controller. However, only in this configuration file, the directory names must not contain space characters. Please note that you have to choose two different directories. Using the same directory for the Exec Agent and for the Job Controller is not

File Name	File Type	Windows	Unix	Description
				supported and my result in failures.
prxsniff.jar	binary	yes	yes	ZebraTester product library
prxsniff.key	ASCII	yes	yes	Contains the ZebraTester GUI license key. You should modify this using a text editor when you receive a new license key. Note: This file is not deleted when uninstalling ZebraTester
recorderBlacklist.dat	ASCII	yes	yes	Optional file. Contains a list of DNS names and/or IP addresses for which recording is suppressed by the proxy recorder. This means that URL requests for such list elements are not forwarded to the web server and also not recorded. Instead of this the web client (web browser) receive a faked response directly from the proxy recorder. A DNS name or an IP address can also contain multiple times the wildcard char '*'. The response sent to the web client depends on the configured "blacklist action" which can contain one of the following values per DNS name or per IP address: • **abort** : abort the network connection to the web client w/o sending any response. • **webbugimage** : send a transparent image of 1x1 pixel to the web client. • **<number / HTTP status code>** : send a HTTP response containing the configured status code to the web client (w/o any content data). Example: # my hosts blacklist *google* webbugimage www.tracking.com 404 *.notthis.ch abort Lines which are starting with a hash char '#' are commented out. Note that the blacklist can also be dynamically configured by using the ProxySniffer REST API.
recorderWhitelist.dat	ASCII	yes	yes	Optional file. Contains a list of DNS names and IP addresses that are exceptions of the

File Name	File Type	Windows	Unix	Description
				recorder blacklist. A DNS name or an IP address can also contain multiple times the wildcard char '*'. Example: # my hosts whitelist www.google.com beta.notthis.ch Lines which are starting with a hash char '#' are commented out. Note that the whitelist can also be dynamically configured by using the ProxySniffer REST API.
recorderURLBlacklist.dat	ASCII	yes	yes	Optional file. Contains a list of URLs for which recording is suppressed by the proxy recorder. This means that URL requests for such list elements are not forwarded to the web server and also not recorded. Instead of this the web client (web browser) receive a faked response directly from the proxy recorder. An URL can also contain multiple times the wildcard char '*'. The response sent to the web client depends on the configured "blacklist action" which can contain one of the following values per DNS name or per IP address: • **abort** : abort the network connection to the web client w/o sending any response. • **webbugimage** : send a transparent image of 1x1 pixel to the web client. • **<number / HTTP status code>** : send a HTTP response containing the configured status code to the web client (w/o any content data). Example: # my urls blacklist *.google.* webbugimage http://*.nothingforme.ch/* abort *www.tracking.com/images/* 404 Lines which are starting with a hash char '#' are commented out. Note that the blacklist can also be dynamically configured by using

File Name	File Type	Windows	Unix	Description
				the ProxySniffer REST API.
recorderURLWhitelist.dat	ASCII	yes	yes	Optional file. Contains a list of URLs that are exceptions of the recorder URL blacklist. An URL can also contain multiple times the wildcard char '*'. Example: # my urls whitelist https://*ae* *www.x.com/* Lines which are starting with a hash char '#' are commented out. Note that the whitelist can also be dynamically configured by using the ProxySniffer REST API.
root.cer	ASCII	yes	yes	The common CA root certificate used to derive all automatically created SSL server certificates (*.crt and *.privkey files). Note: it's strongly recommended that you replace this file by your own version (see ZebraTester Installation Guide).
responseVerification.dat	ASCII	yes	yes	Contains the default settings about the automatically applied response verifications and the automatically applied failure actions for all recorded URLs of a Web surfing session.
UninstallExecAgentService.bat	ASCII	yes	no	Allows you to uninstall the Exec Agent Windows Service.
UninstallProxySnifferService.bat	ASCII	yes	no	Allows you to uninstall the ZebraTester GUI Windows Service.
***.keystore**	~~binary~~	~~yes~~	~~yes~~	Outdated. Starting from ZebraTester version 5.0 any old *.keystore files in the ZebraTester installation directory should be manually deleted in order that ZebraTester generates new Web server certificate derived from the new "root.cer" certificate.
***.crt**	ASCII	yes	yes	The *.crt files contain faked SSL server certificates (in X509 data format) which are derived from the CA root certificate "root.cer". The *.crt files are automatically created by the Web Admin GUI when recording an encrypted HTTPS Web surfing session. You can delete these files manually at any time – together with the *.privkey files. If such a file is needed and does not exist, it will be created automatically.

File Name	File Type	Windows	Unix	Description
***.privkey**	binary	yes	yes	The private keys used by the *.crt files.

The following sub-directories are automatically created by the installation kit:

Subdirectory	Windows	Unix	Description
Documentation	yes	yes	Contains the documentation (Application Reference Manual, User's Guide, Plug-In Developer's Manual, …).
Documentation /javadoc	yes	yes	Contains the ZebraTester Java API documentation (description of all product-specific classes and methods used by the load test programs).
jre	yes	no, OS X: yes	Contains the Java runtime environment and the Java compiler installed by the installation kit. Note: this JRE does not affect other Java JRE or SDK installations on the same computer system – it is only used by ZebraTester.
MyTests	yes	yes	Top level directory for the Project Navigator, in which all recorded Web surfing sessions, all load test programs, and all load test results are stored. On Unix systems, this directory is automatically created if it does not exist, when the Web Admin GUI is started. Note: this folder is not deleted when uninstalling ZebraTester. Hint: you may set another top level directory for the Project Navigator by configuring a new value inside the file **mytests.dat**

7 Manufacturer

Ingenieurbüro David Fischer AG, Switzerland | A company of the Apica Group

Manufacturer's Web Site: http://www.apicasystem.com
Support: support@apicasystem.com
Sales: sales@apicasystem.com