Monitoring Web Services - Webinject

Introduction: The problem

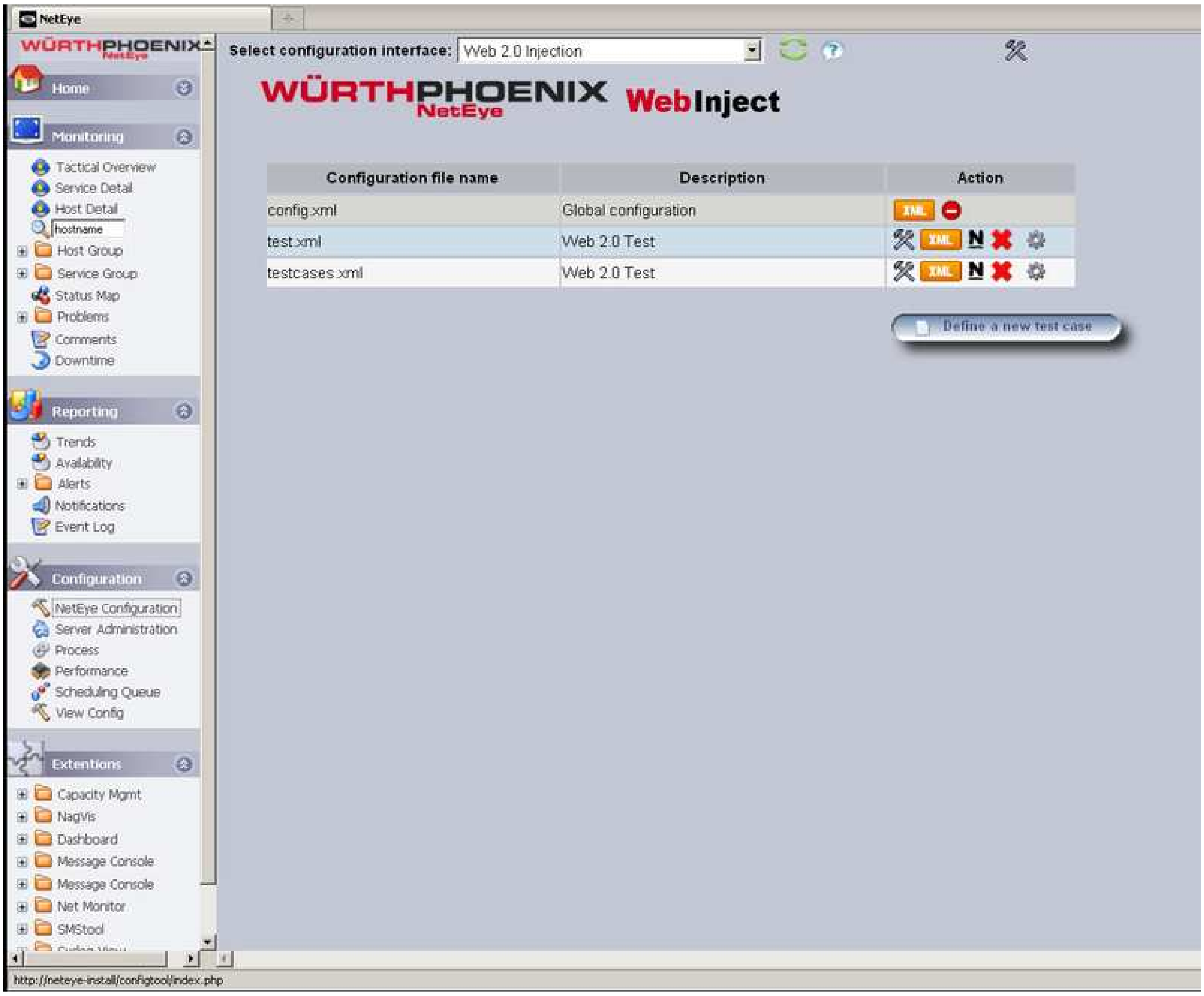
Todays web sites provide advanced web services with user authentication and interaction. Therefore it is no more enough for monitoring to write a test checking if a web server responds on a port, i.e. 80.

For this purpose comes WebInject, a tool for automated testing of web applications and web services. With this testing framework it is possible to configure a complete login procedure into a web service and check the returned answers from the web server. It can be used to test individual system components that have HTTP interfaces (JSP, ASP, CGI, PHP, AJAX, Servlets, HTML Forms, XML/SOAP Web Services, REST, etc), and can be used as a test harness to create a suite of [HTTP level] automated functional, acceptance, and regression tests. A test harness allows you to run many test cases and collect/report your results. WebInject offers real-time results display and may also be used for monitoring system response times.

Web 2.0 Monitoring with NetEye

The fist step consists in the definition of test cases used to query for the availability of a web service and the evaluation of the responce.

Open from the NetEye main menu “Configuraion” → ” NetEye Configuration” to open the configuration selector. From there select “Web 2.0 Monitoring” to access the main view with a list of all configured Webinject test cases and the global configuration file **config.xml**.



Select configuration interface: Web 2.0 Injection

Configuration file name	Description	Action
config.xml	Global configuration	XML
test.xml	Web 2.0 Test	XML N
testcases.xml	Web 2.0 Test	XML N

Available action commands:

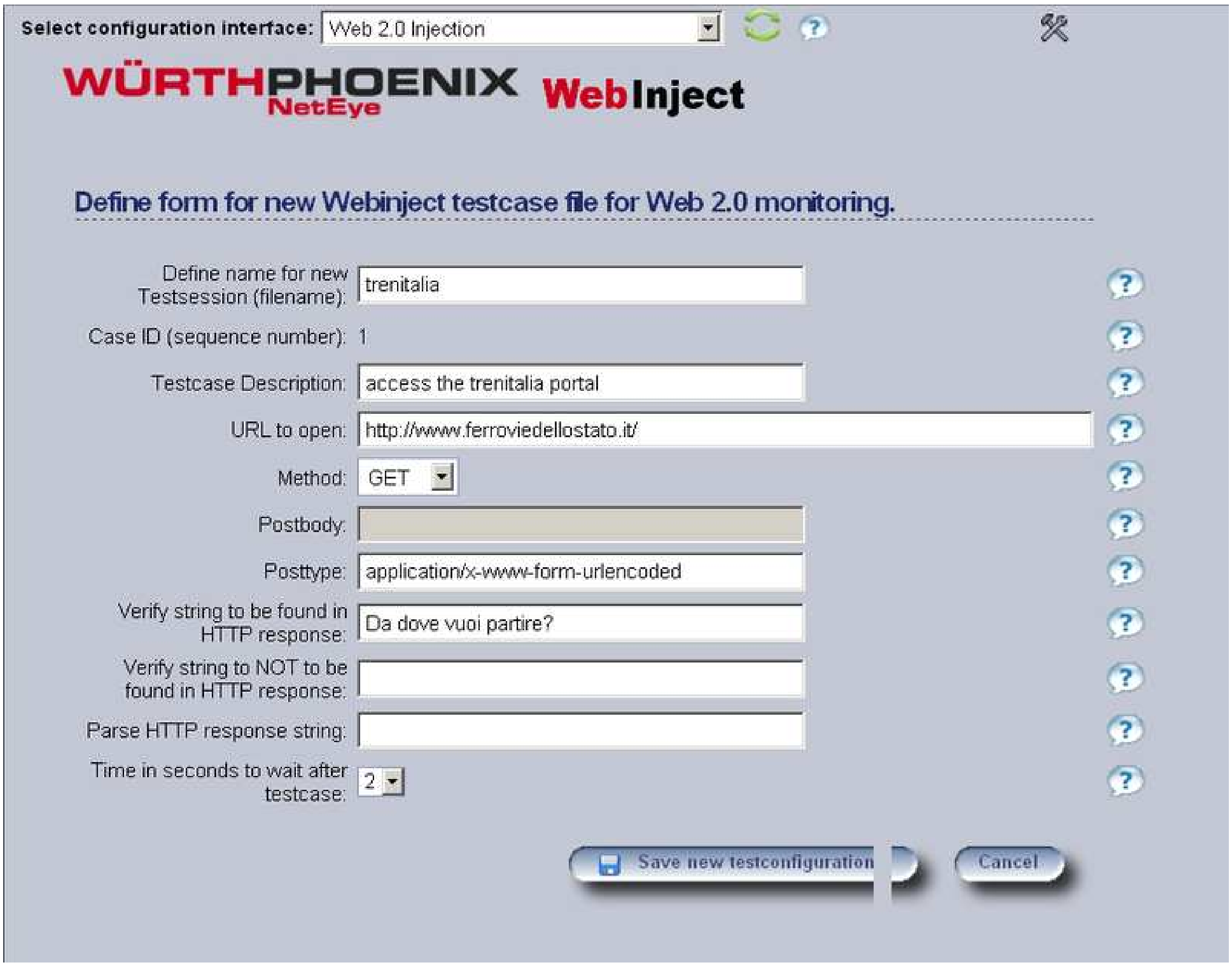
- NEW testcase definition form button
- EDIT: Modify an existing webinject testcases configuration through a user web form

- XML VIEW: View and modify directly the xml code of the webinject testcase configuration
- NAGIOS CONFIGURAGTION: Instruction on how to integrate an xml testcase configuration into the automated monitoring of Nagios with NetEye
- DELETE the testcase configuration (the global configuration config.xml can never be deleted)
- RUN and DEBUG a defined testcase configuration with webinject and view the detailled results

NEW TESTCASE

Click on the button "Define a new testcase" and fill into the form:

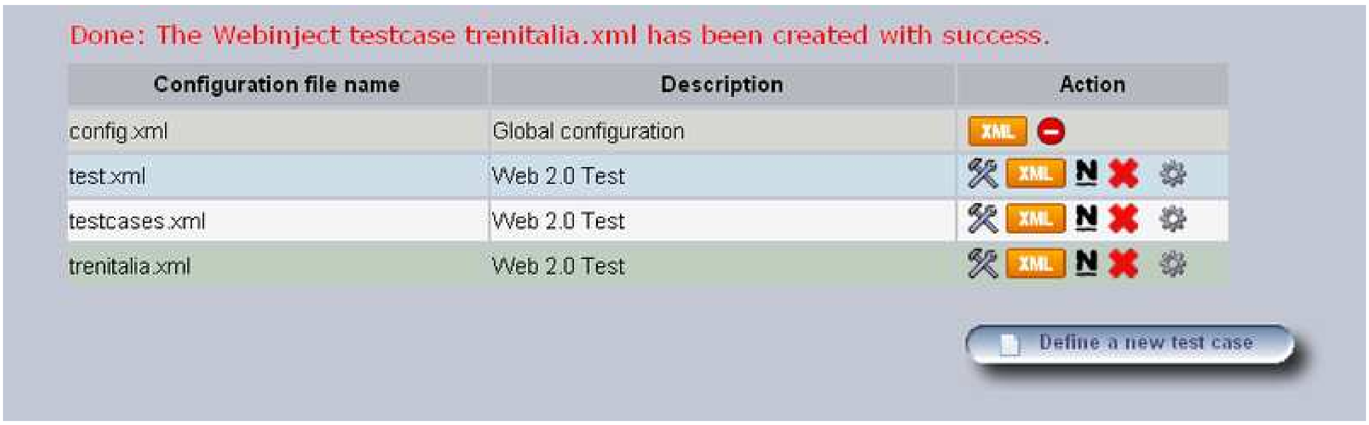
- The name for the new testcase collection: this will be the name used for the file (therefore the use of special characters if not suggested)
- The case ID is generated automatically and will be at the moment of the definition "1". (Afterwards there can be defined additional cases through the edit view)
- A user fiendly description of the purpose of this check
- The URL to upen. The URL can be indicated with variables to post.
- In alternative the send menthod can be changed from "GET" method to "POST" and a POSTBODY can be defined. (A postbody may have this syntax to send a username and password variable: postbody="username=corey& password=welcome")
- With Posttype can be defined the kind of encoding to use. By default "application/x-www-form-urlencoded" is suggested.
- String to be found in the response HTML code
- String to NOT to be found in the response HTML code
- Parseresponse:


Select configuration interface: Web 2.0 Injection
WÜRTHPHOENIX NetEye
WebInject
Save new testconfiguration
Cancel

Define form for new Webinject testcase file for Web 2.0 monitoring.

Define name for new Testsession (filename):	trenitalia
Case ID (sequence number):	1
Testcase Description:	access the trenitalia portal
URL to open:	http://www.ferroviedellostato.it/
Method:	GET
Postbody:	
Posttype:	application/x-www-form-urlencoded
Verify string to be found in HTTP response:	Da dove vuoi partire?
Verify string to NOT to be found in HTTP response:	
Parse HTTP response string:	
Time in seconds to wait after testcase:	2

The main view after confirm:



Done: The Webinject testcase trenitalia.xml has been created with success.

Configuration file name	Description	Action
config.xml	Global configuration	XML
test.xml	Web 2.0 Test	XML N
testcases.xml	Web 2.0 Test	XML N
trenitalia.xml	Web 2.0 Test	XML N

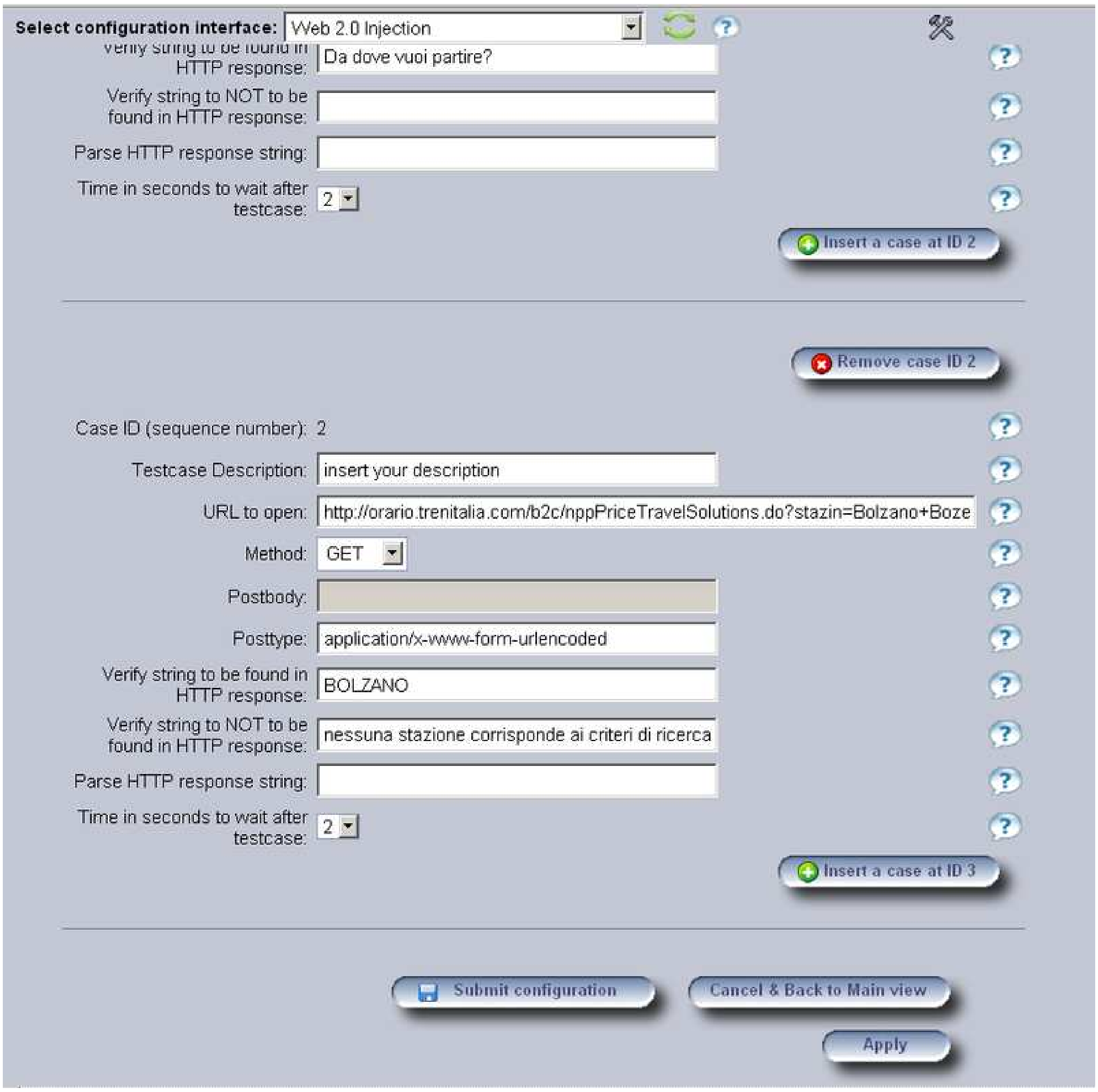
EDIT / EXTEND the testcase

Click on the "tools" icon in the row of the testcase to edit, to open the editor for adding additional testcases.

Click on the button "Insert a case at ID 2" to add another case after the first one. - Later it will be possible to insert cases at every ID, but never substituting the first one.

Here we define now a query for a train from Bozen/Bolzano to Trento passing all search parameters as GET arguments with the URL.

Click on "Apply" to save or on "Submit configuration" to save all changes and to return to the main view.

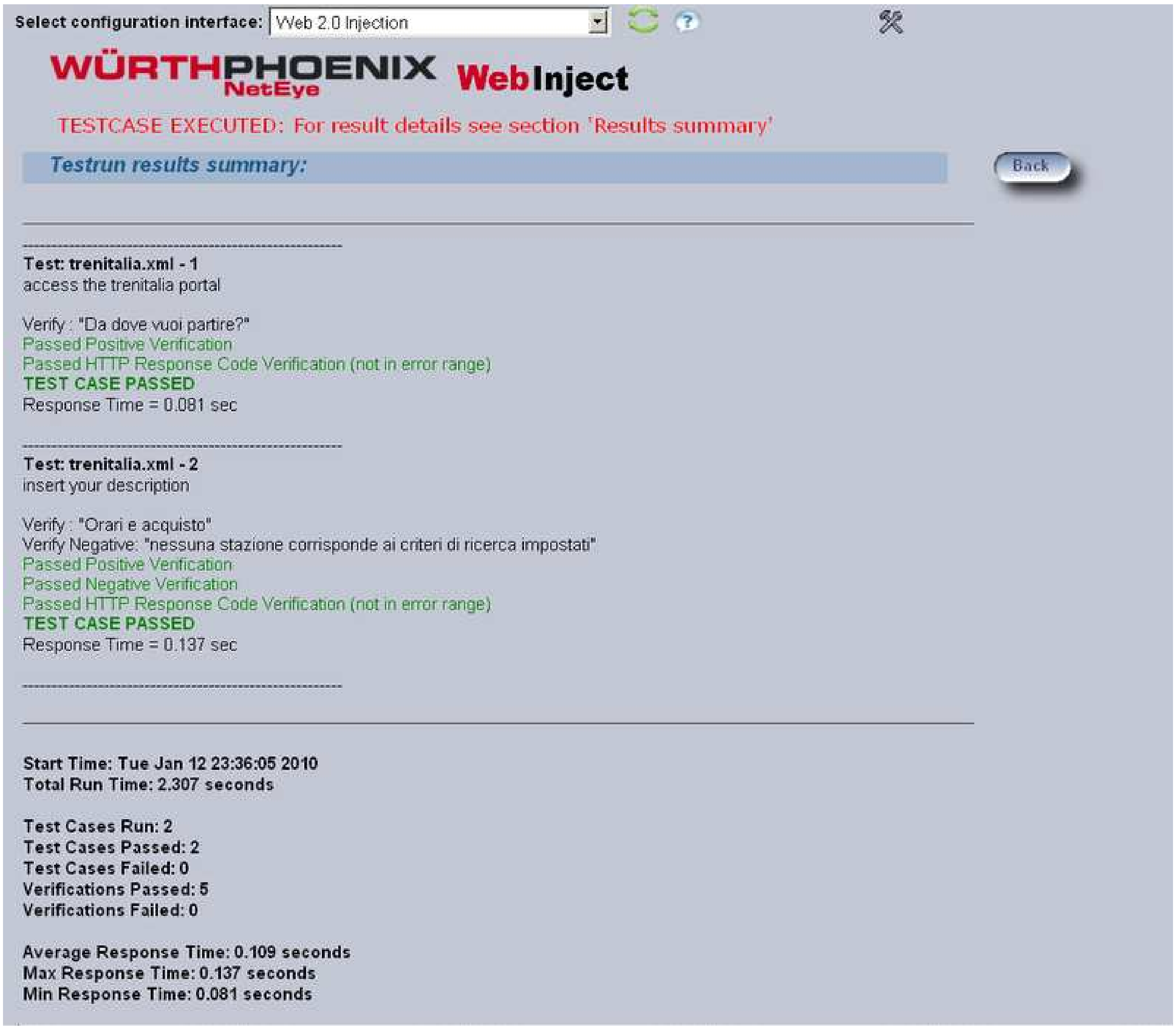


Select configuration interface: Web 2.0 Injection

Verify string to be found in HTTP response:	Da dove vuoi partire?
Verify string to NOT to be found in HTTP response:	
Parse HTTP response string:	
Time in seconds to wait after testcase:	2

Case ID (sequence number):	2
Testcase Description:	insert your description
URL to open:	http://orario.trenitalia.com/b2c/nppPriceTravelSolutions.do?stazin=Bolzano+Boze
Method:	GET
Postbody:	
Posttype:	application/x-www-form-urlencoded
Verify string to be found in HTTP response:	BOLZANO
Verify string to NOT to be found in HTTP response:	nessuna stazione corrisponde ai criteri di ricerca
Parse HTTP response string:	
Time in seconds to wait after testcase:	2

Now click from the main view on the “Run” icon to perform a test exectuion of the just created testcase configuration:



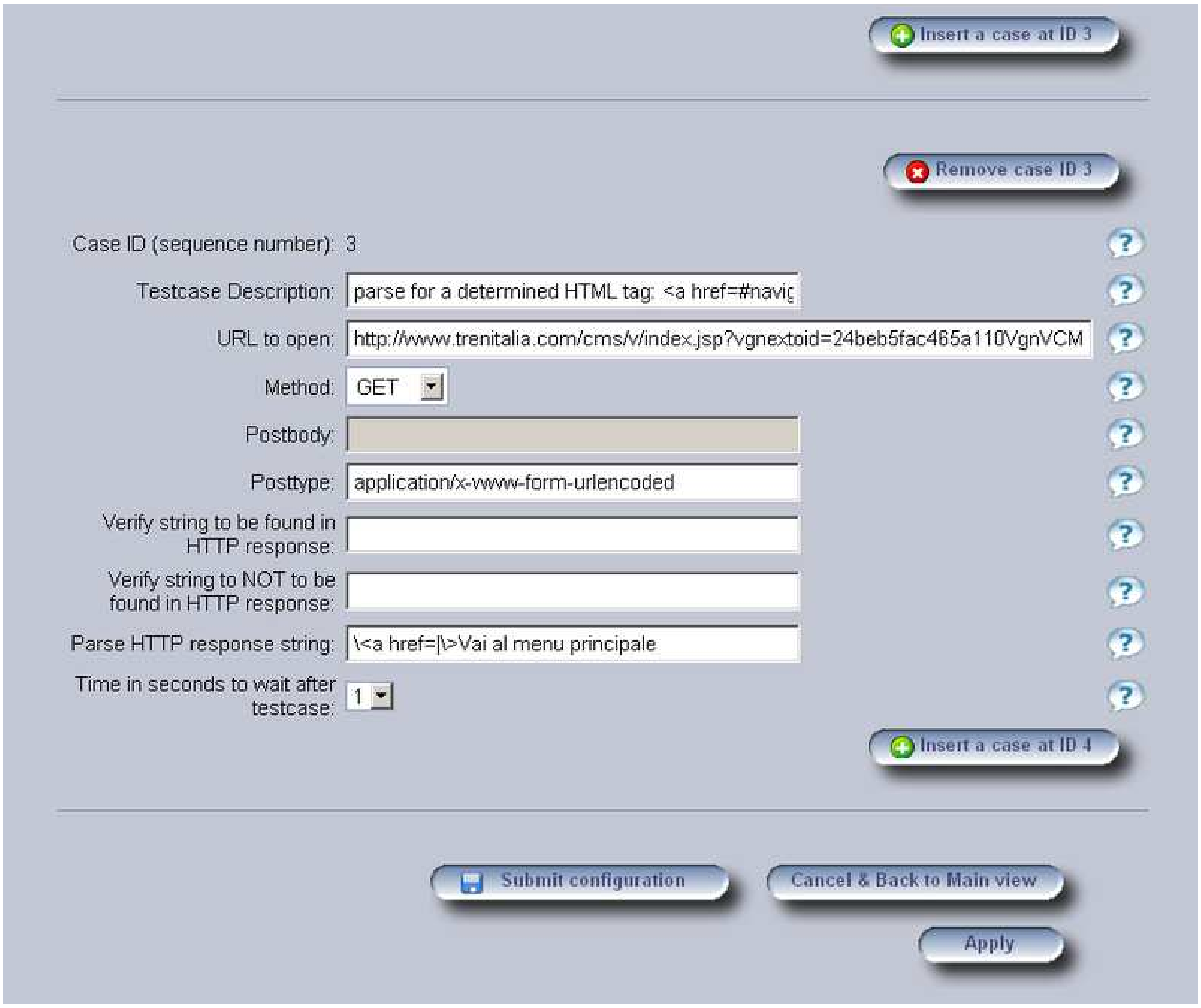
TESTCASE EXECUTED: For result details see section 'Results summary'

Testrun results summary:

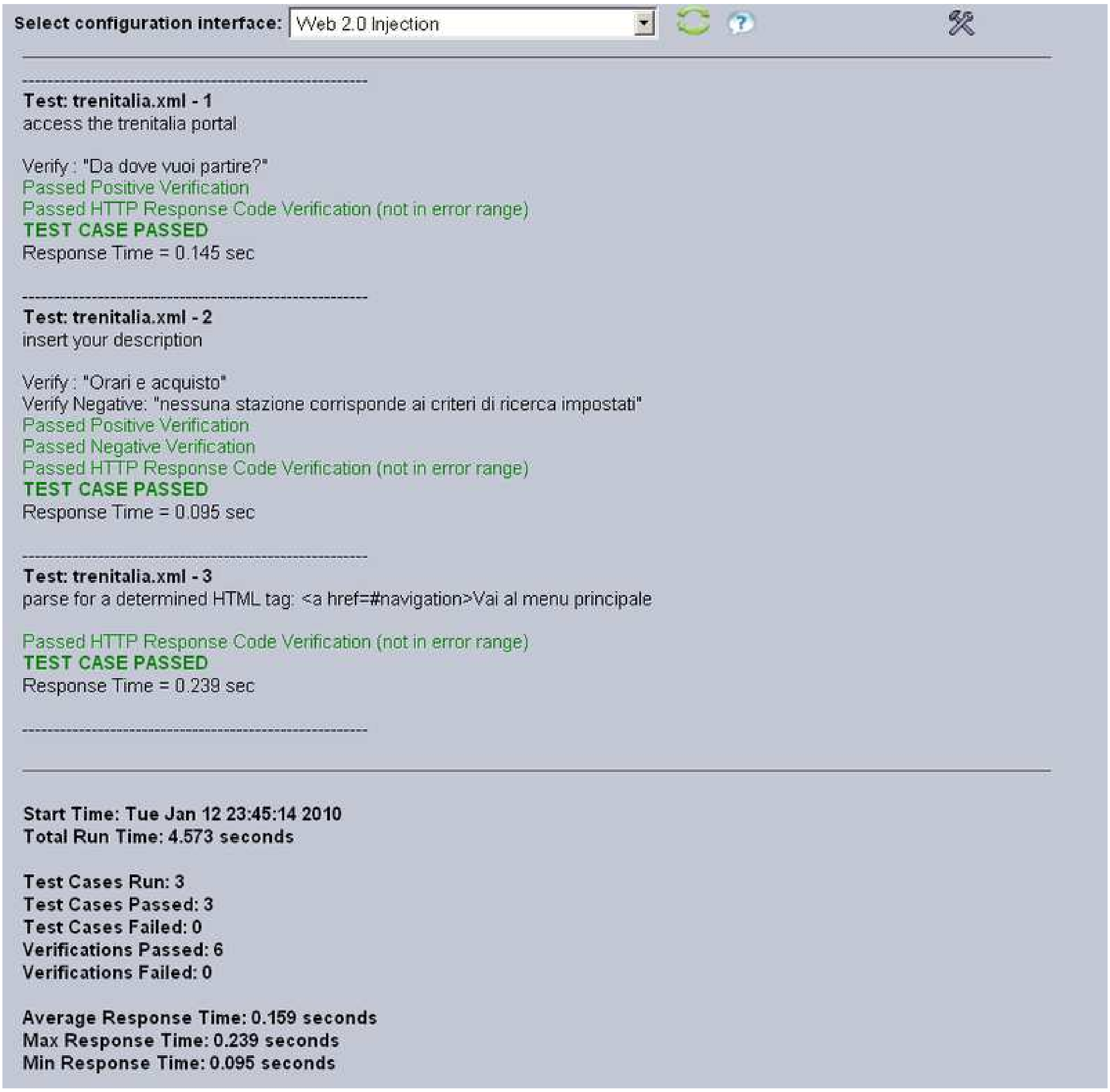
```
------------------------------------------------------------
Test: trenitalia.xml - 1
access the trenitalia portal

Verify : "Da dove vuoi partire?"
Passed Positive Verification
Passed HTTP Response Code Verification (not in error range)
TEST CASE PASSED
Response Time = 0.081 sec

------------------------------------------------------------
Test: trenitalia.xml - 2
insert your description

Verify : "Orari e acquisto"
Verify Negative: "nessuna stazione corrisponde ai criteri di ricerca impostati"
Passed Positive Verification
Passed Negative Verification
Passed HTTP Response Code Verification (not in error range)
TEST CASE PASSED
Response Time = 0.137 sec

------------------------------------------------------------

Start Time: Tue Jan 12 23:36:05 2010
Total Run Time: 2.307 seconds

Test Cases Run: 2
Test Cases Passed: 2
Test Cases Failed: 0
Verifications Passed: 5
Verifications Failed: 0

Average Response Time: 0.109 seconds
Max Response Time: 0.137 seconds
Min Response Time: 0.081 seconds
```

Parsing with regular expressions

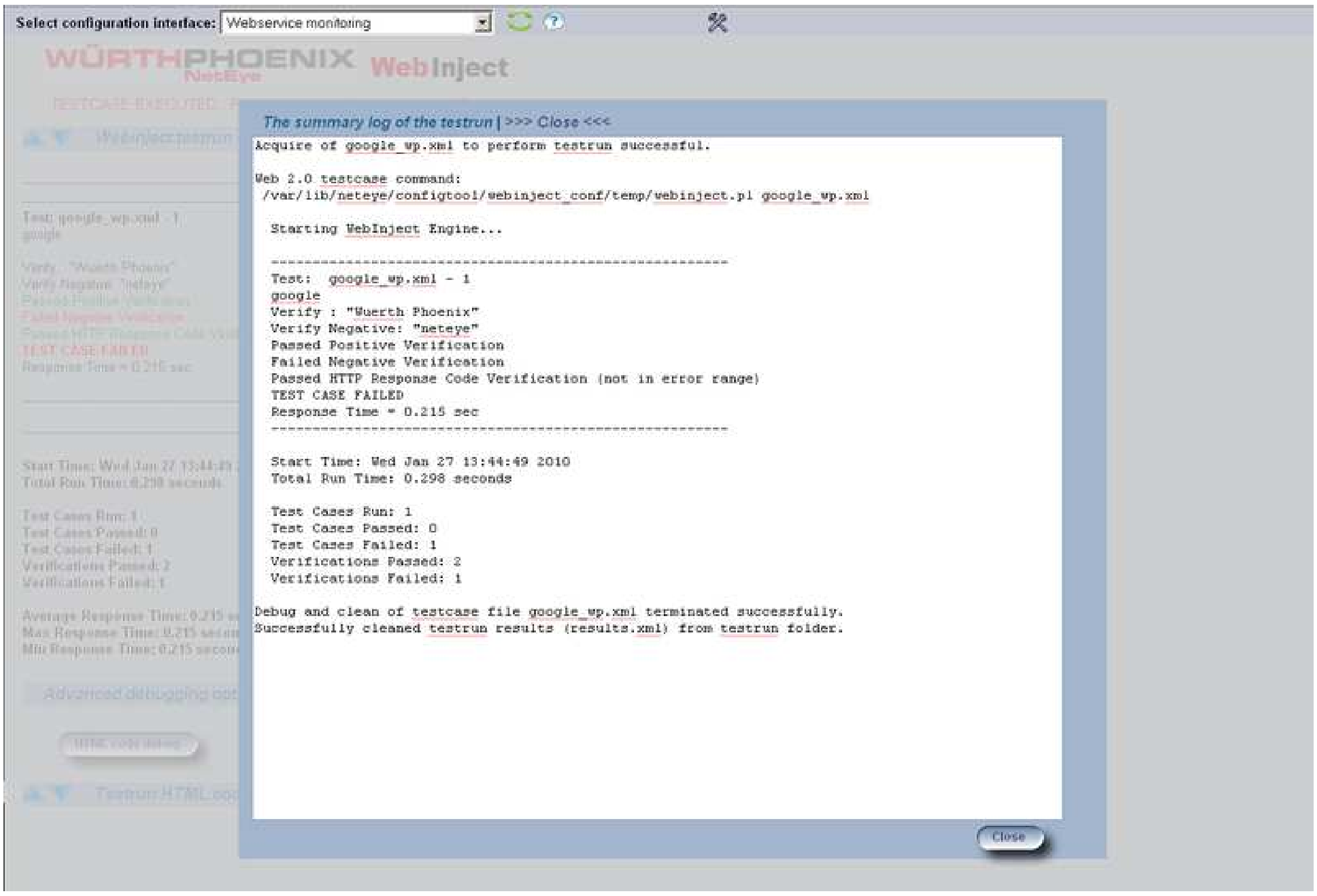
After retruning again to the edit forms and adding a third case, we fill in the following information to parse for a determined tag - content combination:



Case ID (sequence number):	3
Testcase Description:	parse for a determined HTML tag: <a href=#navig
URL to open:	http://www.trenitalia.com/cms/v/index.jsp?vgnextoid=24beb5fac465a110VgnVCM
Method:	GET
Postbody:	
Posttype:	application/x-www-form-urlencoded
Verify string to be found in HTTP response:	
Verify string to NOT to be found in HTTP response:	
Parse HTTP response string:	\<a href=|\>Vai al menu principale
Time in seconds to wait after testcase:	1

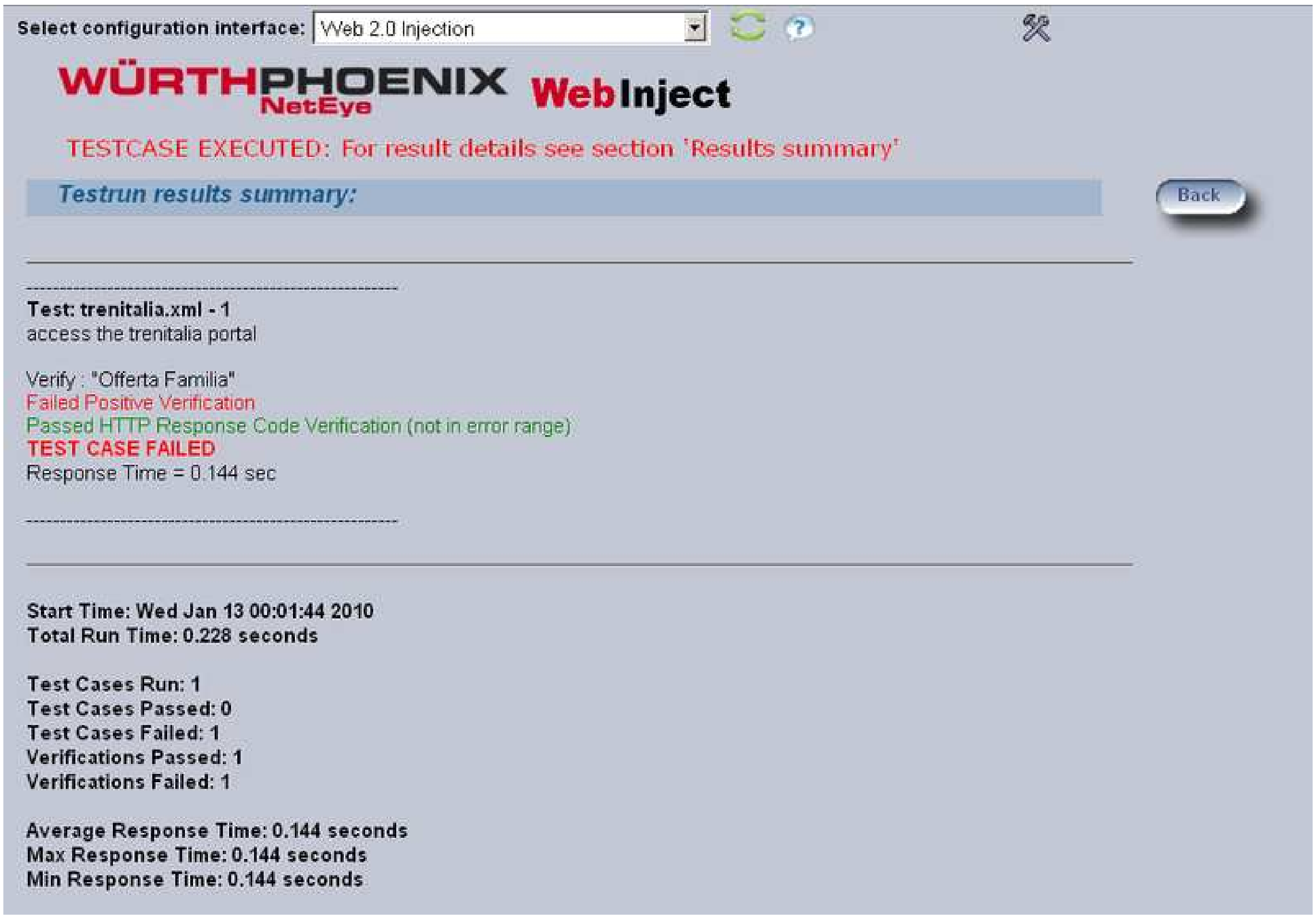


```
------------------------------------------------------
Test: trenitalia.xml - 1
access the trenitalia portal

Verify : "Da dove vuoi partire?"
Passed Positive Verification
Passed HTTP Response Code Verification (not in error range)
TEST CASE PASSED
Response Time = 0.145 sec

------------------------------------------------------
Test: trenitalia.xml - 2
insert your description

Verify : "Orari e acquisto"
Verify Negative: "nessuna stazione corrisponde ai criteri di ricerca impostati"
Passed Positive Verification
Passed Negative Verification
Passed HTTP Response Code Verification (not in error range)
TEST CASE PASSED
Response Time = 0.095 sec

------------------------------------------------------
Test: trenitalia.xml - 3
parse for a determined HTML tag: <a href=#navigation>Vai al menu principale

Passed HTTP Response Code Verification (not in error range)
TEST CASE PASSED
Response Time = 0.239 sec

------------------------------------------------------

Start Time: Tue Jan 12 23:45:14 2010
Total Run Time: 4.573 seconds

Test Cases Run: 3
Test Cases Passed: 3
Test Cases Failed: 0
Verifications Passed: 6
Verifications Failed: 0

Average Response Time: 0.159 seconds
Max Response Time: 0.239 seconds
Min Response Time: 0.095 seconds
```


Select configuration interface: Webservice monitoring
WÜRTHPHOENIX NetEye WebInject
The summary log of the testrun | >>> Close <<<
Close

```
Acquire of google_wp.xml to perform testrun successful.

Web 2.0 testcase command:
 /var/lib/neteye/configtool/webinject_conf/temp/webinject.pl google_wp.xml

  Starting WebInject Engine...

  -------------------------------------------------------
  Test:  google_wp.xml - 1
  google
  Verify : "Wuerth Phoenix"
  Verify Negative: "neteye"
  Passed Positive Verification
  Failed Negative Verification
  Passed HTTP Response Code Verification (not in error range)
  TEST CASE FAILED
  Response Time = 0.215 sec
  -------------------------------------------------------

  Start Time: Wed Jan 27 13:44:49 2010
  Total Run Time: 0.298 seconds

  Test Cases Run: 1
  Test Cases Passed: 0
  Test Cases Failed: 1
  Verifications Passed: 2
  Verifications Failed: 1

Debug and clean of testcase file google_wp.xml terminated successfully.
Successfully cleaned testrun results (results.xml) from testrun folder.
```

Failing configuration example

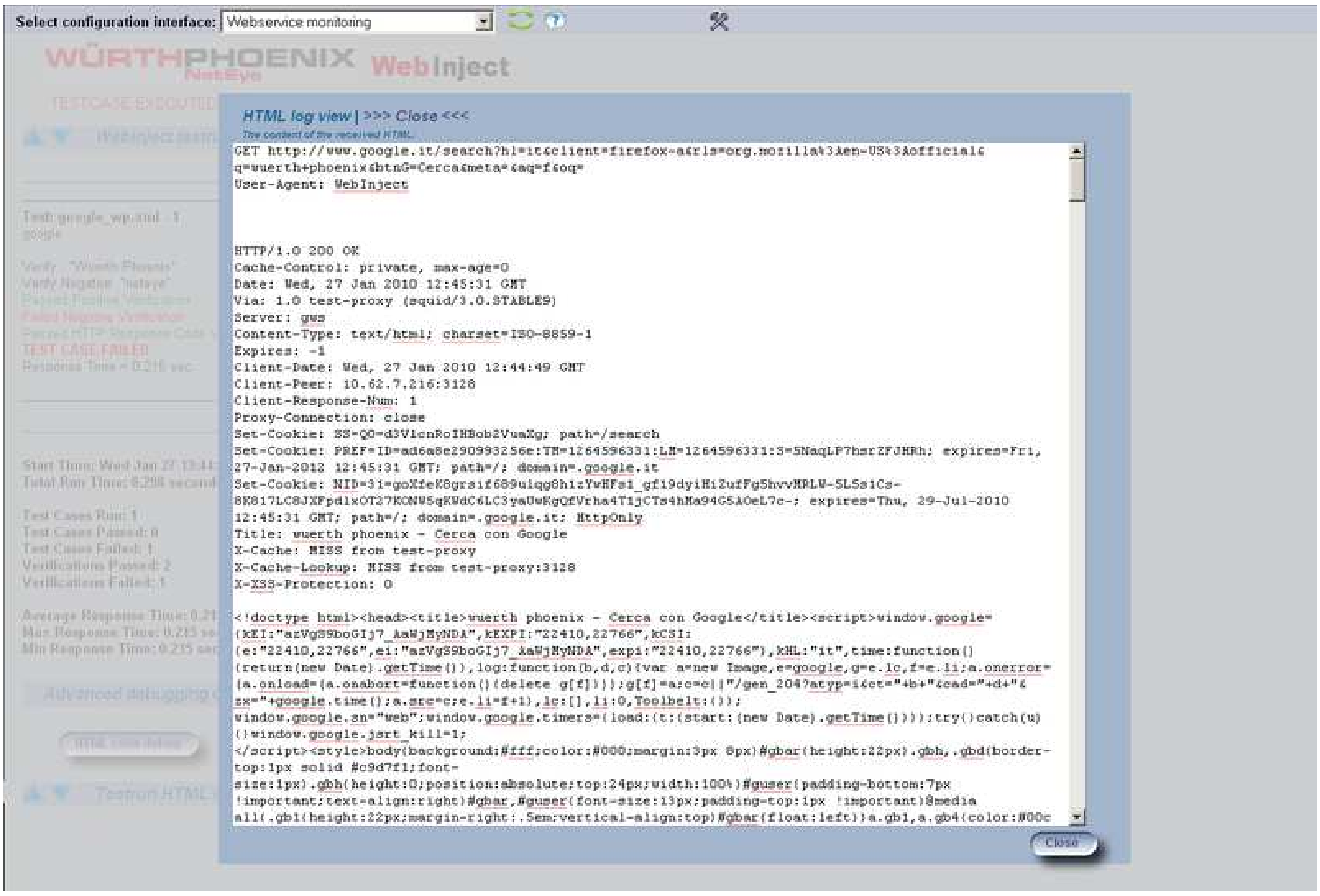

Select configuration interface: Web 2.0 Injection
WÜRTHPHOENIX NetEye WebInject
Back

TESTCASE EXECUTED: For result details see section 'Results summary'

Testrun results summary:

```
-------------------------------------------------------
Test: trenitalia.xml - 1
access the trenitalia portal

Verify : "Offerta Familia"
Failed Positive Verification
Passed HTTP Response Code Verification (not in error range)
TEST CASE FAILED
Response Time = 0.144 sec

-------------------------------------------------------

Start Time: Wed Jan 13 00:01:44 2010
Total Run Time: 0.228 seconds

Test Cases Run: 1
Test Cases Passed: 0
Test Cases Failed: 1
Verifications Passed: 1
Verifications Failed: 1

Average Response Time: 0.144 seconds
Max Response Time: 0.144 seconds
Min Response Time: 0.144 seconds
```

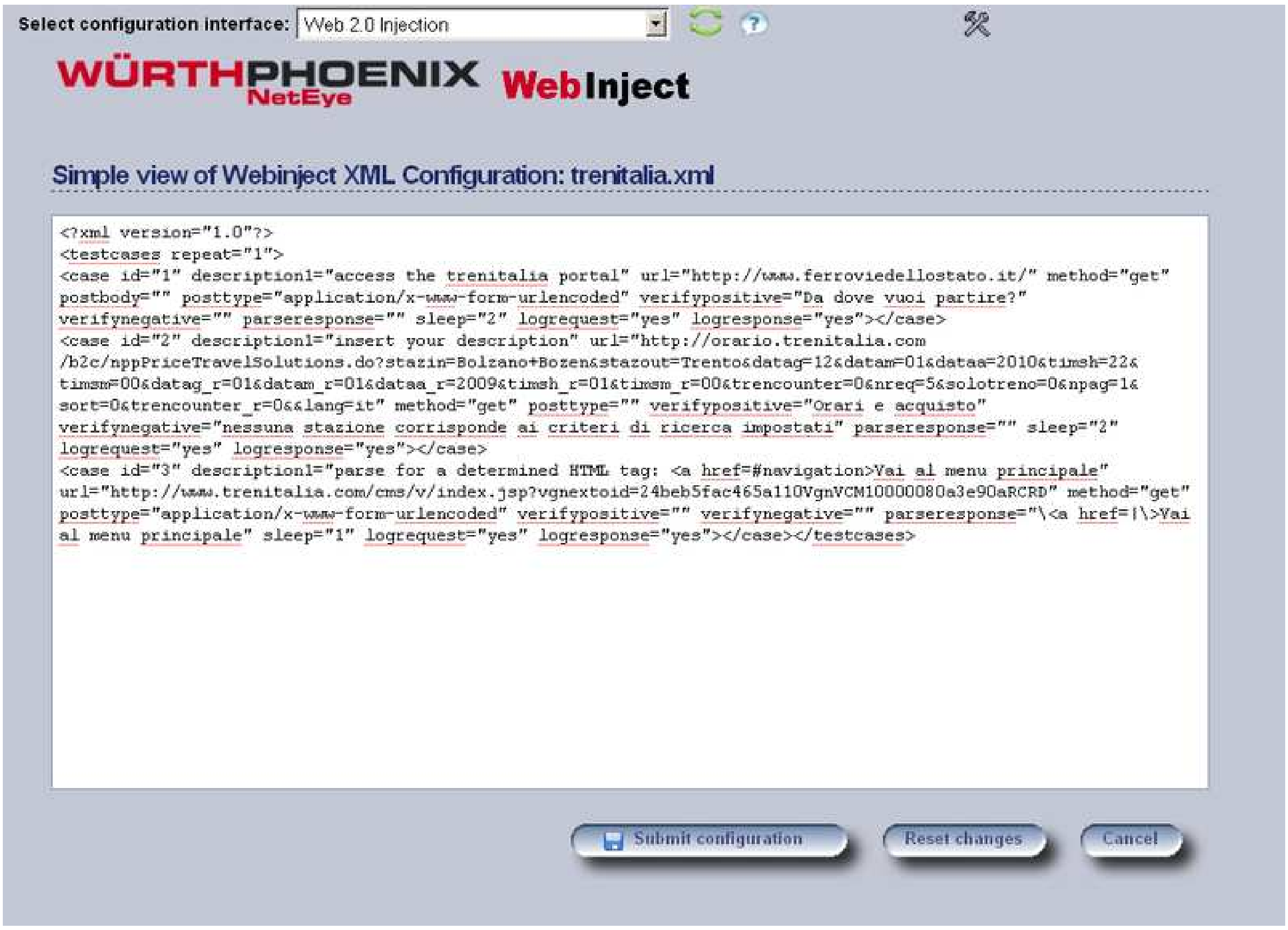
The debug view of the HTML code:


Select configuration interface: Webservice monitoring
WÜRTHPHOENIX NetEye WebInject
HTML log view | >>> Close <<<
The content of the received HTML
Close

```
GET http://www.google.it/search?hl=it&client=firefox-a&rls=org.mozilla%3Aen-US%3Aofficial&
q=wuerth+phoenix&btnG=Cerca&meta=&aq=f&oq=
User-Agent: WebInject



HTTP/1.0 200 OK
Cache-Control: private, max-age=0
Date: Wed, 27 Jan 2010 12:45:31 GMT
Via: 1.0 test-proxy (squid/3.0.STABLE9)
Server: gws
Content-Type: text/html; charset=ISO-8859-1
Expires: -1
Client-Date: Wed, 27 Jan 2010 12:44:49 GMT
Client-Peer: 10.62.7.216:3128
Client-Response-Num: 1
Proxy-Connection: close
Set-Cookie: SS=Q0=d3VicnRoIHBob2VuaXg; path=/search
Set-Cookie: PREF=ID=ad6a8e290993256e:TM=1264596331:LM=1264596331:S=5NaqLP7hsrZFJHRh; expires=Fri,
27-Jan-2012 12:45:31 GMT; path=/; domain=.google.it
Set-Cookie: NID=31=goXfeK8grsif689uiqg8h1zYwHFs1_gf19dyiHi2ufFg5hvvHRLW-SL5s1Cs-
8K817LC8JXFpdlxOT27KONW5qKWdC6LC3yaUwKgQfVrha4T1jCTs4hMa94G5AOeL7c-; expires=Thu, 29-Jul-2010
12:45:31 GMT; path=/; domain=.google.it; HttpOnly
Title: wuerth phoenix - Cerca con Google
X-Cache: MISS from test-proxy
X-Cache-Lookup: MISS from test-proxy:3128
X-XSS-Protection: 0

<!doctype html><head><title>wuerth phoenix - Cerca con Google</title><script>window.google=
{kEI:"azVgS9boGIj7_AaWjMyNDA",kEXPI:"22410,22766",kCSI:
{e:"22410,22766",ei:"azVgS9boGIj7_AaWjMyNDA",expi:"22410,22766"},kHL:"it",time:function()
{return(new Date).getTime()},log:function(b,d,c){var a=new Image,e=google,g=e.lc,f=e.li;a.onerror=
(a.onload=(a.onabort=function(){delete g[f]}));g[f]=a;c=c||"/gen_204?atyp=i&ct="+b+"&cad="+d+"&
zx="+google.time();a.src=c;e.li=f+1},lc:[],li:0,Toolbelt:{}};
window.google.sn="web";window.google.timers={load:{t:{start:(new Date).getTime()}}};try{}catch(u)
{}window.google.jsrt_kill=1;
</script><style>body{background:#fff;color:#000;margin:3px 8px}#gbar{height:22px}.gbh,.gbd{border-
top:1px solid #c9d7f1;font-
size:1px}.gbh{height:0;position:absolute;top:24px;width:100%}#guser{padding-bottom:7px
!important;text-align:right}#gbar,#guser{font-size:13px;padding-top:1px !important}@media
all{.gb1{height:22px;margin-right:.5em;vertical-align:top}#gbar{float:left}}a.gb1,a.gb4{color:#00c
```

XML direct editing

Simply click on the “XML” icon in the main menu and view / edit the xml code

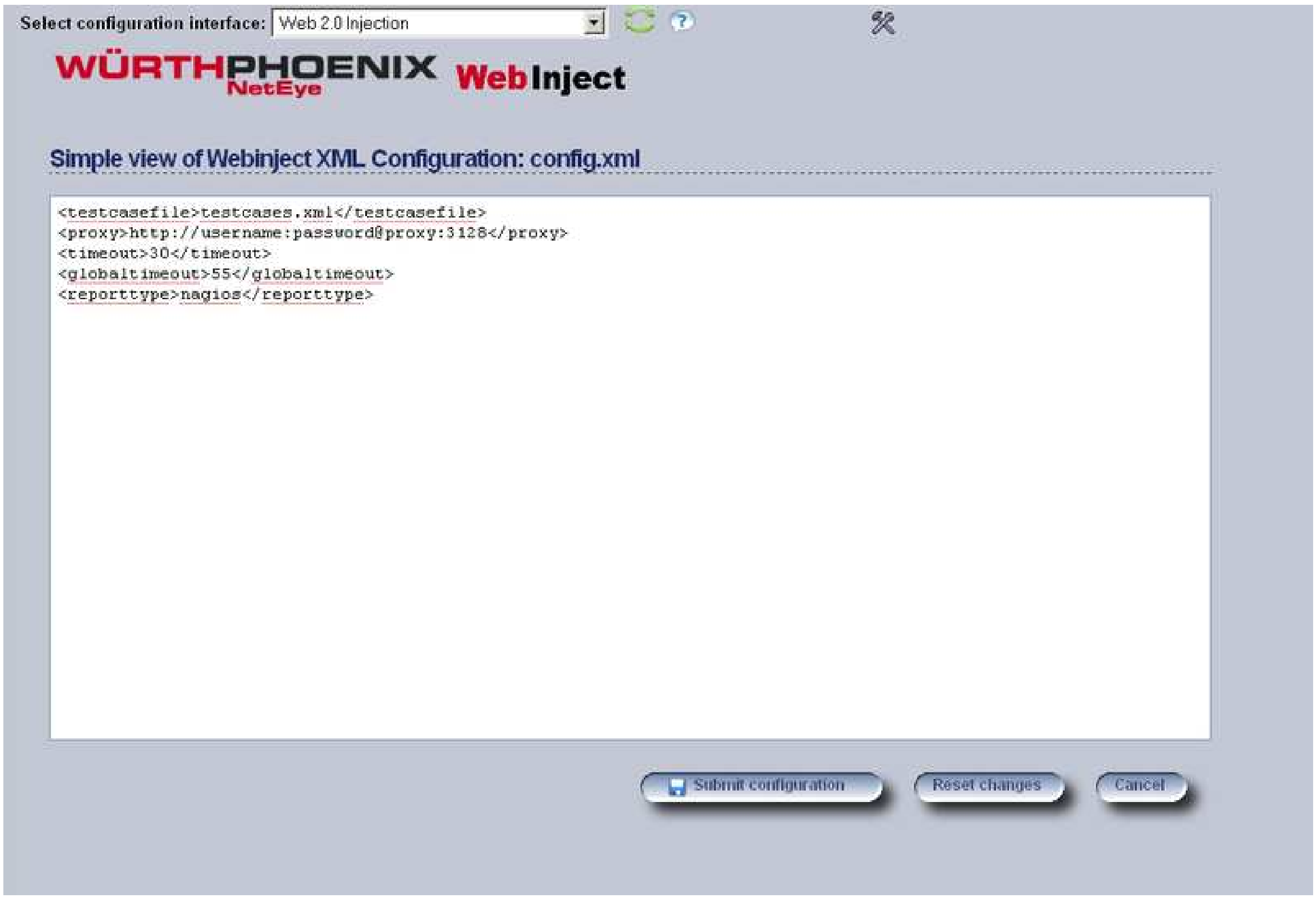

Select configuration interface: Web 2.0 Injection
WÜRTHPHOENIX NetEye WebInject
Submit configuration
Reset changes
Cancel

Simple view of Webinject XML Configuration: trenitalia.xml

```
<?xml version="1.0"?>
<testcases repeat="1">
<case id="1" description1="access the trenitalia portal" url="http://www.ferroviedellostato.it/" method="get"
postbody="" posttype="application/x-www-form-urlencoded" verifypositive="Da dove vuoi partire?"
verifynegative="" parseresponse="" sleep="2" logrequest="yes" logresponse="yes"></case>
<case id="2" description1="insert your description" url="http://orario.trenitalia.com
/b2c/nppPriceTravelSolutions.do?stazin=Bolzano+Bozen&stazout=Trento&datag=12&datam=01&dataa=2010&timsh=22&
timsm=00&datag_r=01&datam_r=01&dataa_r=2009&timsh_r=01&timsm_r=00&trencounter=0&nreq=5&solotreno=0&npag=1&
sort=0&trencounter_r=0&&lang=it" method="get" posttype="" verifypositive="Orari e acquisto"
verifynegative="nessuna stazione corrisponde ai criteri di ricerca impostati" parseresponse="" sleep="2"
logrequest="yes" logresponse="yes"></case>
<case id="3" description1="parse for a determined HTML tag: <a href=#navigation>Vai al menu principale"
url="http://www.trenitalia.com/cms/v/index.jsp?vgnextoid=24beb5fac465a110VgnVCM10000080a3e90aRCRD" method="get"
posttype="application/x-www-form-urlencoded" verifypositive="" verifynegative="" parseresponse="\<a href=|\>Vai
al menu principale" sleep="1" logrequest="yes" logresponse="yes"></case></testcases>
```

The global configuration: config.xml

The main configuration holds values regarding the overall behaviour. Important variables are the setup of the proxy and the timeouts. Within this configuration there are stored automatically also all testcase-configurations created with the webinterface form.

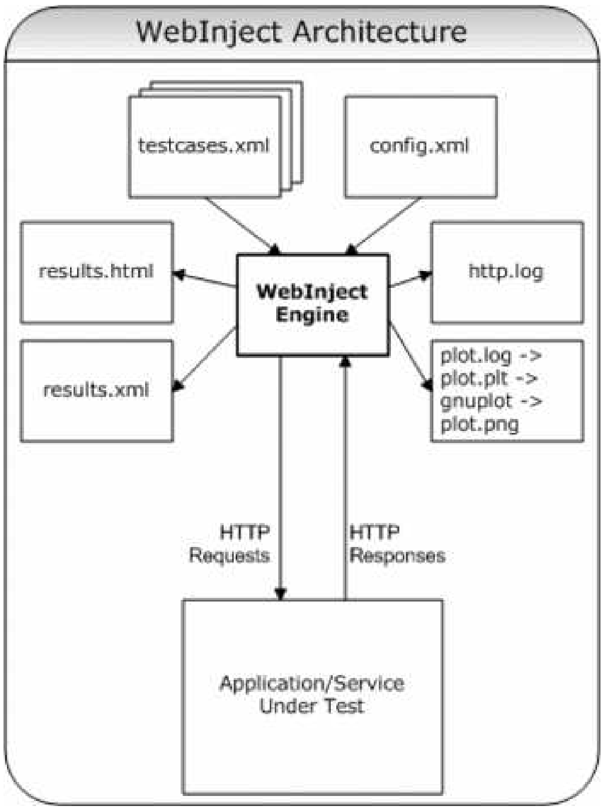

Select configuration interface: Web 2.0 Injection
WÜRTHPHOENIX NetEye WebInject
Submit configuration
Reset changes
Cancel

Simple view of Webinject XML Configuration: config.xml

```
<testcasefile>testcases.xml</testcasefile>
<proxy>http://username:password@proxy:3128</proxy>
<timeout>30</timeout>
<globaltimeout>55</globaltimeout>
<reporttype>nagios</reporttype>
```

Overview of Webinject core

Webinject architecure:



Global configuration

The global configuration of webinject is stored in the configuration file /opt/neteye/webinject/config.xml

Patameters:

- Set use of proxy:

```
http://username:password@127.0.0.1:8080
```

- If desired to pass the request by indicating the web user agent

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
```

- Log test cases: yes | onfail:

```
yes
```

- Timeout:

```
10
```

- The report output for Nagios structure:

```
nagios
```

- Timeout for use with Nagios: Also if tests succeed, but overall execution of all tests takes longer than globaltimeout a warning is returned

```
10
```

Writing a test case

Webinject provides a dedicated xml structure for the definition of an effective test case.

Define inside the /opt/neteye/webinject/config.xml the file of the testcase to use:

```
test_google.xml
```

Create the file test_google.xml in /opt/neteye/webinject

Define the number of times the testcase to repeat: Write the opening and closing tag

IN BETWEEN THE TAGS “testcases” define now the first case to perform:

- id: Set id for numbering of testcases
- description: Description of the testcase
- method: get | post
  - postbody: used to set values for post method

```
postbody="username=corey&password=welcome"
```

- url: url to web service
- posttype: specify content encoding. (default is: “application/x-www-form-urlencoded”)
- verifypositive=“PHOENIX NetEye”: Check if the string can be found in the http response
- verifynegative: same as above just for negative determination
- verifynextpositive/verifynextnegative: Response string for verification in next test case
- logrequest=“yes” Activates HTTP request log to http.log
- logresponse=“yes” Activates HTTP response log to http.log
- parseresponse='href=“leftboundary|rightboundary”
- errormessage: In case of test failure returned message

Implementing the scheduled check in NetEye

Now the check can be implemented in NetEye. For this open the NetEye web-interface on Monarch and define a new Command:

- Name: check_webinject
- Command: (without using a macro) /opt/neteye/webinject/webinject.sh $ARG1$

Now define a new Service by cloning an existing one (i.e. Ping):

- Rename the clone to “WEBINJECT”
- Select the Service check tab: Select check command “check_webinject”
- Complete the Command line to get: check_webinject!test_google.xml

Save

Apply this check on the desired host and Commit to start scheduled checking.

Webinject Copyright