December, 11th 2009

CLIFv2 Quick Start manual

http://clif.ow2.org/

Table of contents

December, 11th 2009

1. Introduction

In this Quick Start, you will learn how to create your first CLIF project.

1.1. Requirements

We will use the Eclipse-RCP based standalone CLIF in this Quick Start. You will need to install standalone CLIF as explained in the Installation manual.

1.2. Summary

We will makes three different tests. In the first test, you will learn how to create a CLIF project, a Test Plan, add probes to the Test Plan and deploy it.

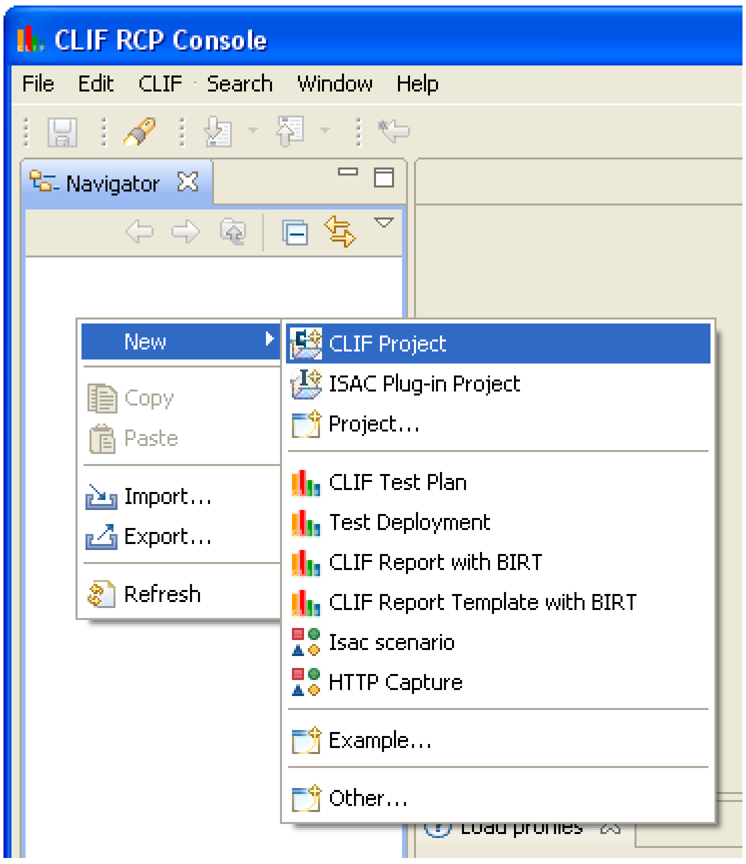
In the second test, we will create an Isac Scenario using several plug-ins to test an HTTP server.

In the third test, we will use HttpCapture to create a scenario and replay it.

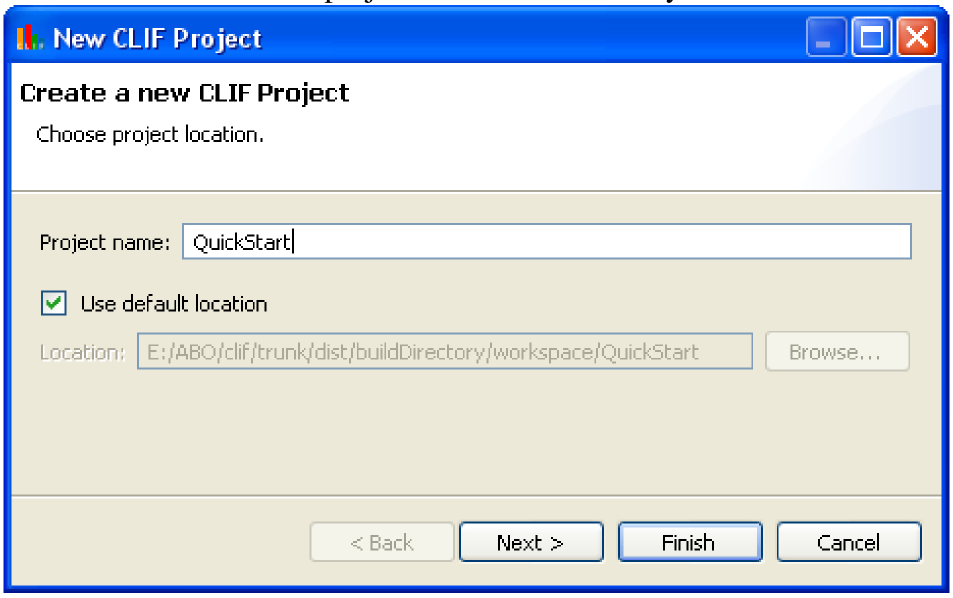
2. Basic test

2.1. Project creation

Create a new CLIF project with New → CLIF Project



Enter "QuickStart" as the name of the project and validate directly with Finish.



Create a new CLIF Project

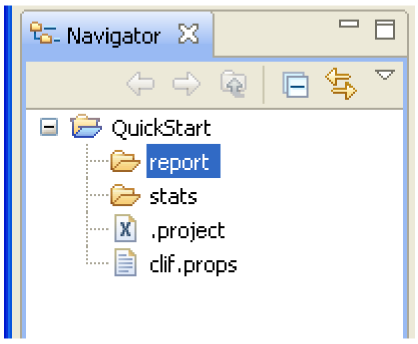
Choose project location.

Project name: QuickStart

✓ Use default location

Location: E:/ABO/clif/trunk/dist/buildDirectory/workspace/QuickStart

You will have a CLIF skeleton project created with base files.



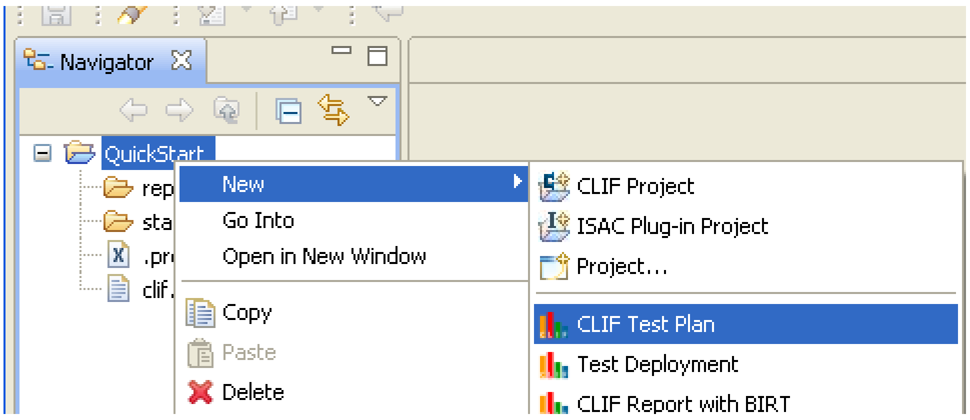
clif.props contains configuration properties for the project. You might change your registry configuration or tune CLIF engine here. See the user manual for more details.

The report directory will contain collected results.

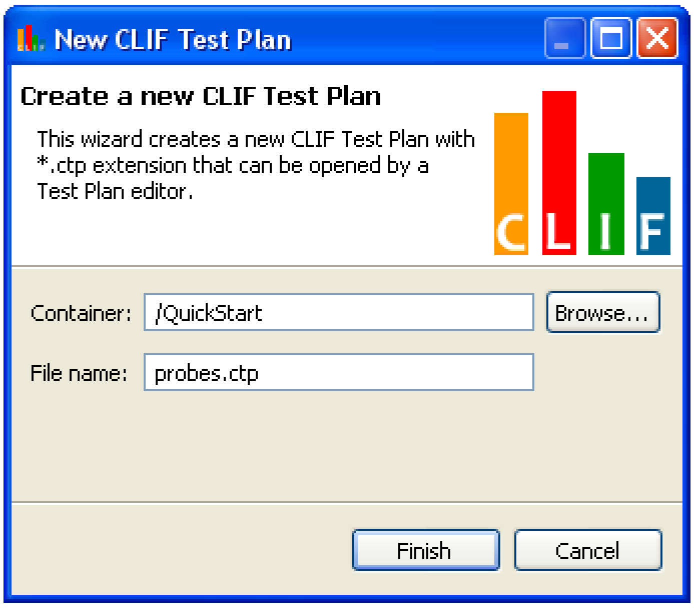
This project will act as your container for your tests and scenarios. Now, you need to create a Test Plan to configure which probes you will deploy. A probe will permit you to monitor the system and record data.

2.2. Test Plan Creation

Create a Test Plan with New → CLIF Test Plan



Enter "probes.ctp" as the file name and Validate


New CLIF Test Plan

Create a new CLIF Test Plan

This wizard creates a new CLIF Test Plan with *.ctp extension that can be opened by a Test Plan editor.

Container: /QuickStart

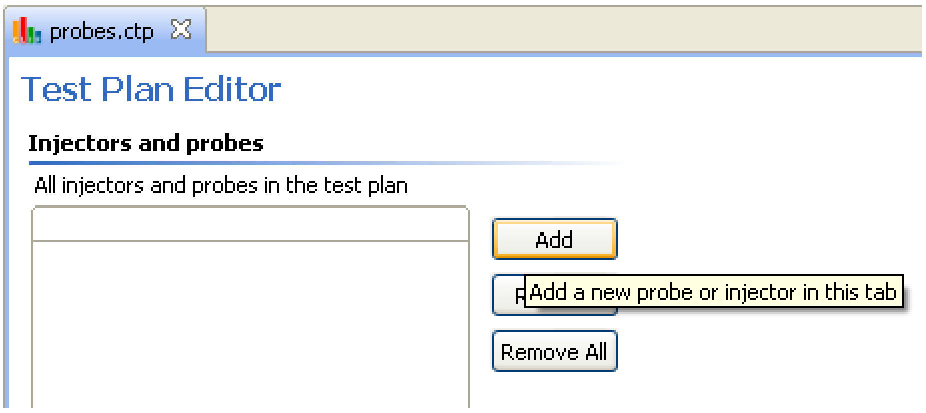
File name: probes.ctp

Your test plan will permit you to configure probes and injectors, deploy them and control their execution. First, we need to add our first probes to our test.

2.3. Add probes

2.3.1. Add cpu probe

We will create a first probe to monitor the CPU. Add a probe with the add button.


probes.ctp

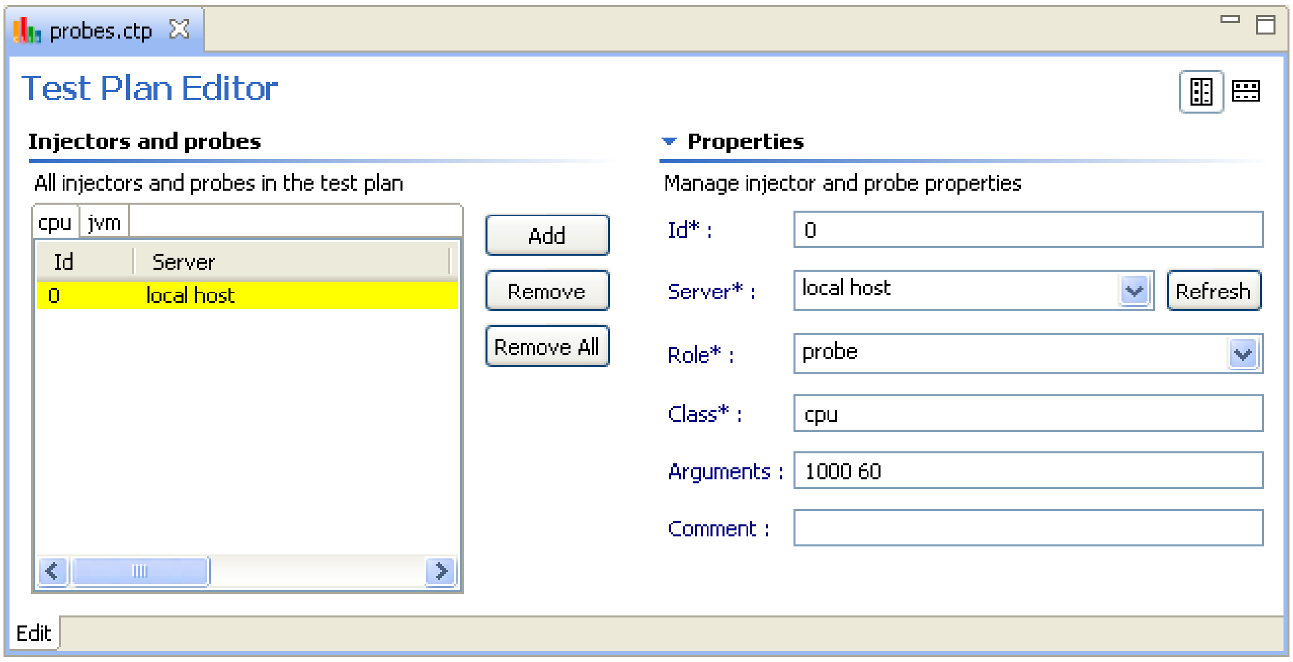
Test Plan Editor

Injectors and probes

All injectors and probes in the test plan

Add a new probe or injector in this tab

Put "cpu" in Class and 1000 60 in arguments. The first argument is the polling period (ms), the second argument is the execution duration (s).



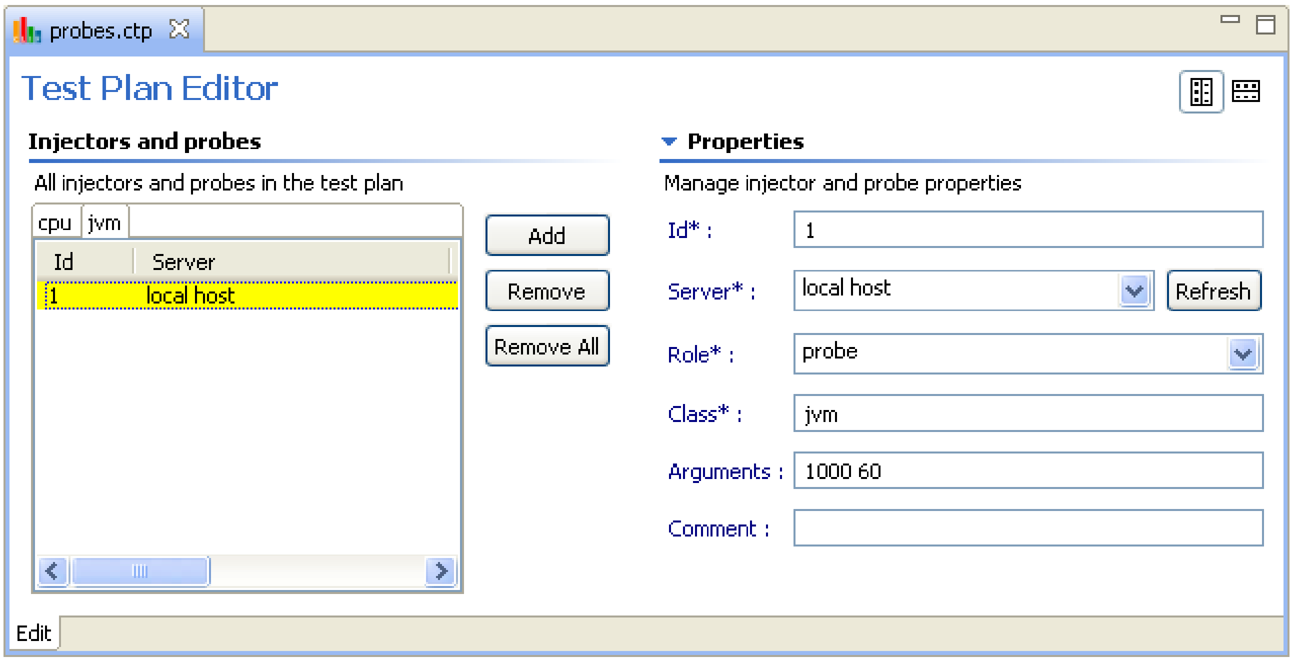
Id	Server
0	local host

Id* :	0
Server* :	local host
Role* :	probe
Class* :	cpu
Arguments :	1000 60
Comment :	

2.3.2. Add jvm probe

Let's create a second probe to monitor the jvm memory.

Add a second probe of class "jvm" and use 1000 60 as arguments. They have the same meaning as the cpu probe.



Id	Server
1	local host

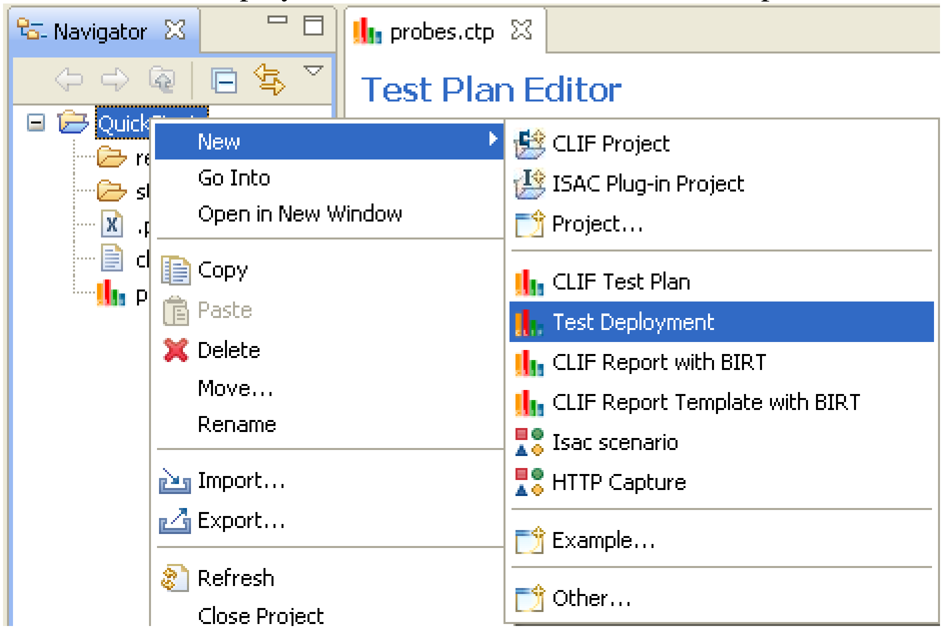
Id* :	1
Server* :	local host
Role* :	probe
Class* :	jvm
Arguments :	1000 60
Comment :	

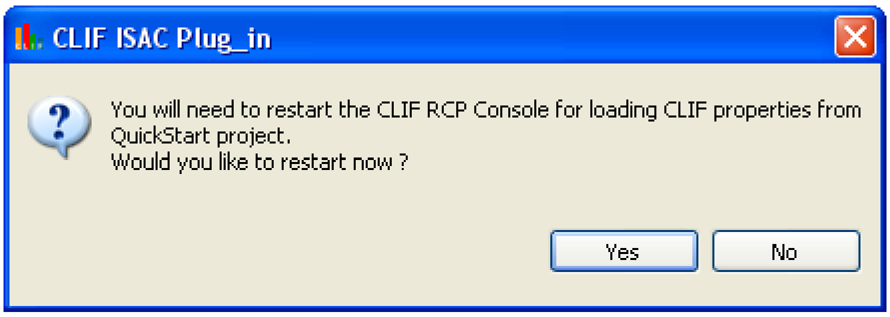
To see other available probes, refer to the User Manual.

You have now an operational Test Plan which will monitor the cpu consumption and the jvm memory. You need to deploy it in order to execute it.

2.4. Test Deployment

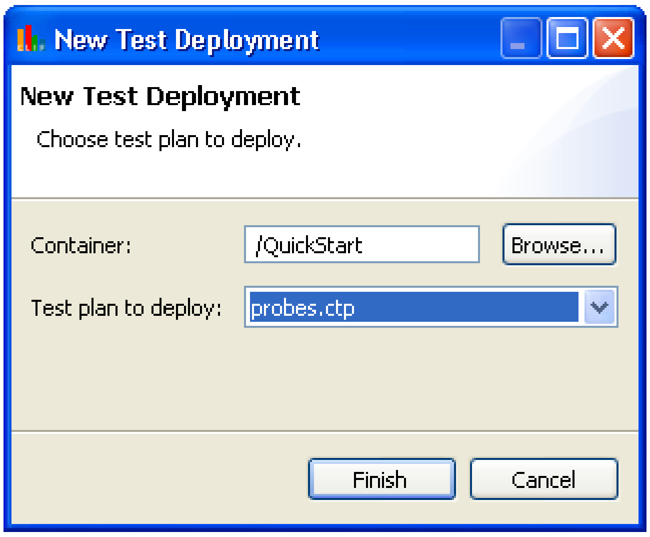
Deploy it with "New -> Test Deployment" and Validate with the default parameters.




CLIF ISAC Plug_in

You will need to restart the CLIF RCP Console for loading CLIF properties from QuickStart project.
Would you like to restart now ?

CLIF needs to restart in order to reload properties from clif.props. Accept it and repeat the test plan deployment after restart.


New Test Deployment

New Test Deployment

Choose test plan to deploy.

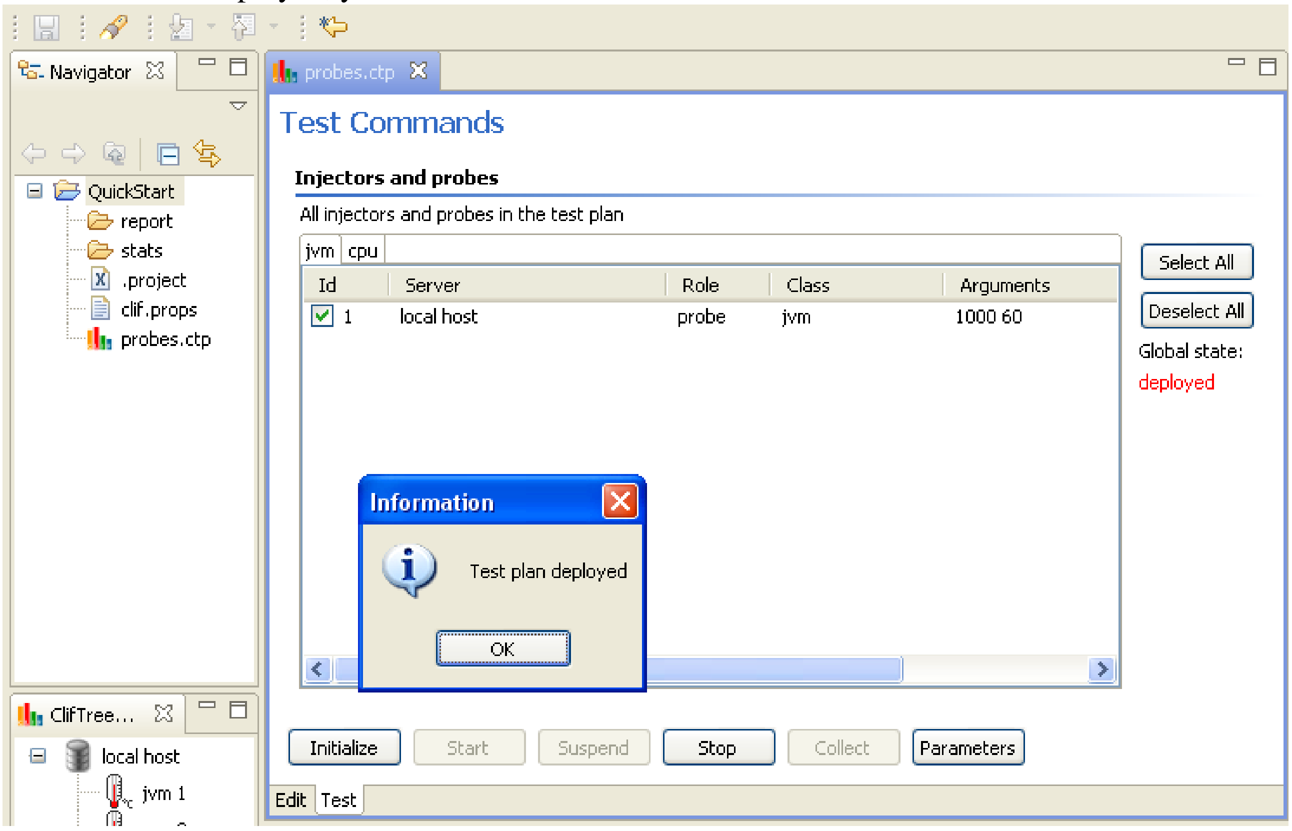
Container: /QuickStart

Test plan to deploy: probes.ctp

Just validate Finish and the test plan should be deployed.

2.5. Test execution

Once the test is deployed, you will be able to initialize the Test.



Id	Server	Role	Class	Arguments
✓ 1	local host	probe	jvm	1000 60

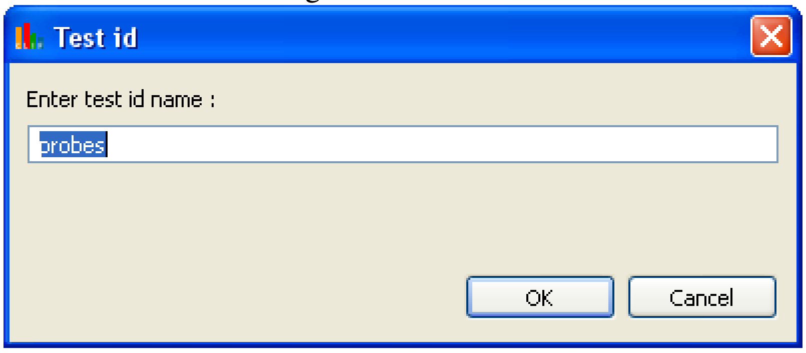
Global state:
deployed

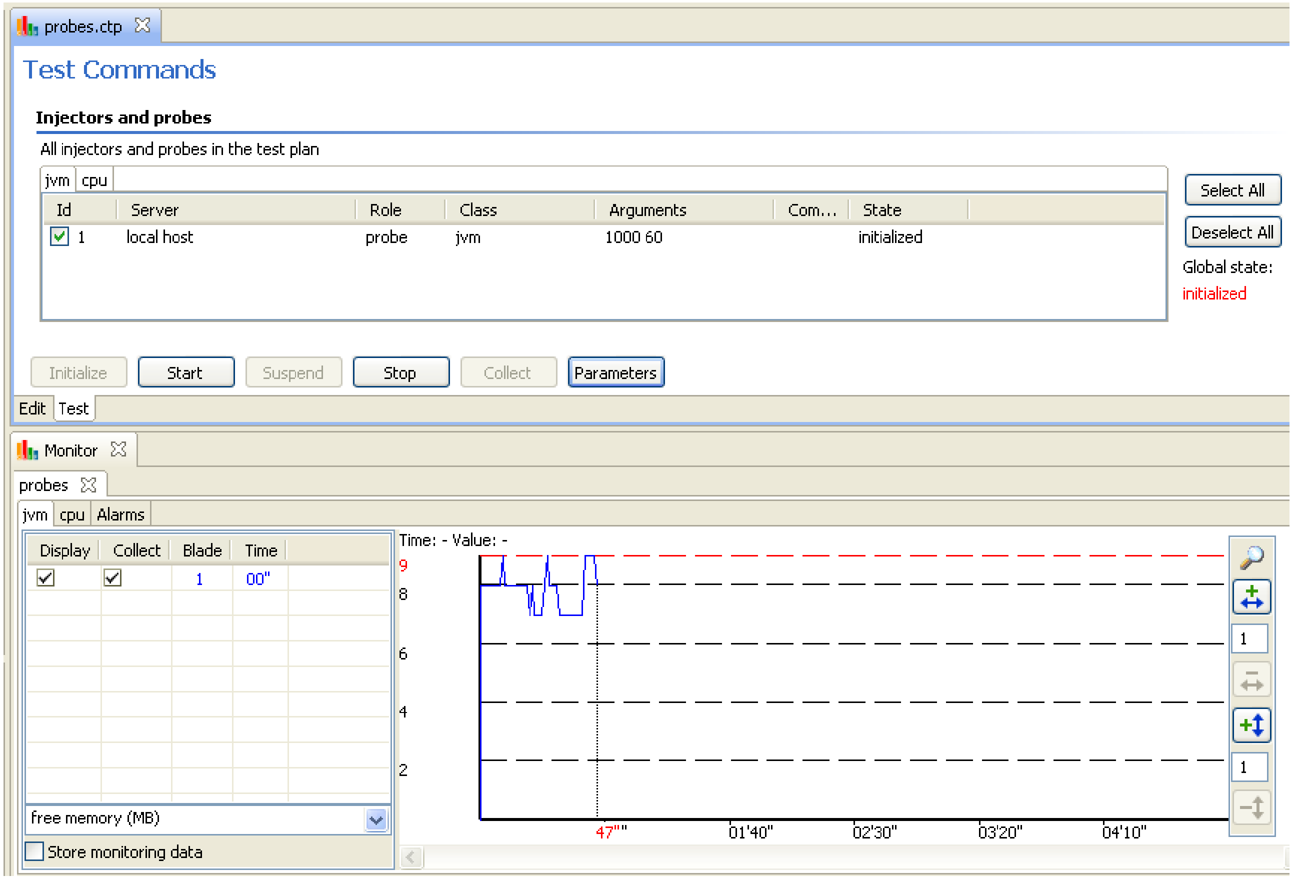
Information
Test plan deployed
OK

2.5.1. Initialize

The initialization will bootstrap probes and injectors but won't start the test.

Initialize the Test Plan and use the default given id.


Test id
Enter test id name :
probes



Id	Server	Role	Class	Arguments	Com...	State
✓ 1	local host	probe	jvm	1000 60		initialized

Global state: initialized

Display	Collect	Blade	Time
✓	✓	1	00"

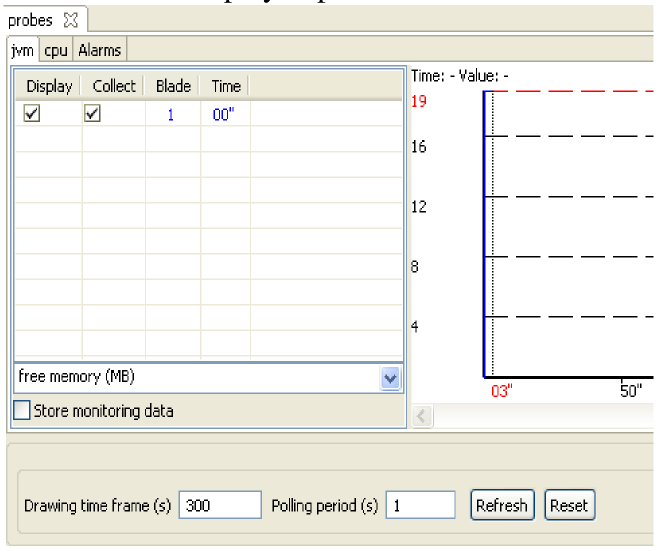
free memory (MB)

Probes monitoring will start and you will be able to review every probes but data are not collected unless the test start.

2.5.2. Monitor tabs

The monitor view allows you to see the details of each type of probes/injectors. An extra tab shows all alarm events sent by probes.

In each tab, you will be able to control displayed probes and measurements.

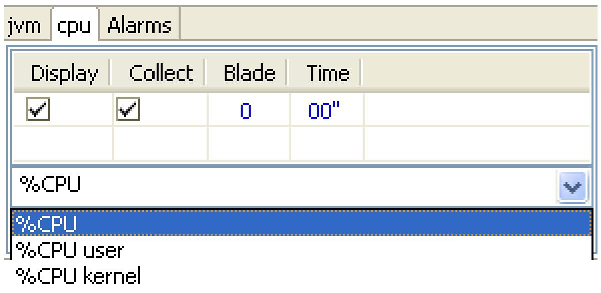


Display	Collect	Blade	Time
✓	✓	1	00"

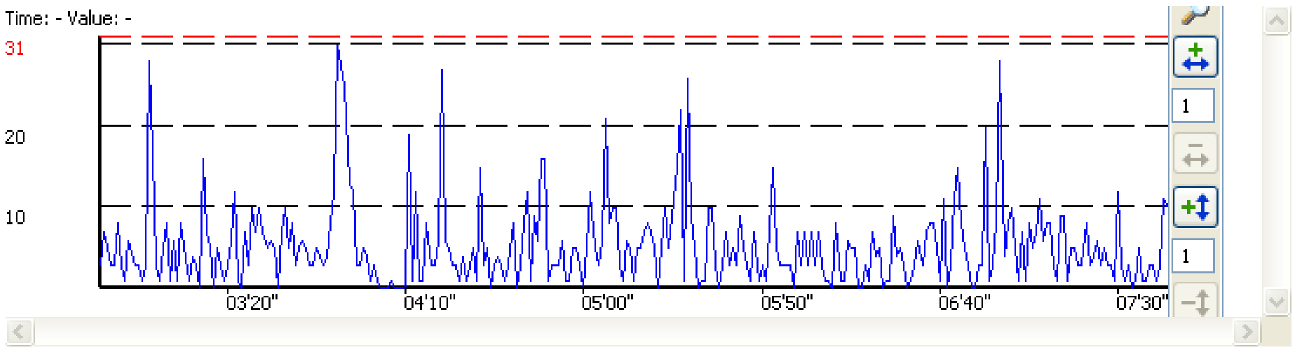
free memory (MB)

Drawing time frame (s) 300 Polling period (s) 1

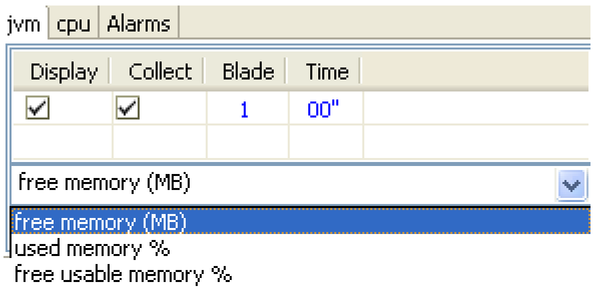
Cpu



The cpu probe measures the global CPU usage, kernel CPU usage and the user CPU usage.

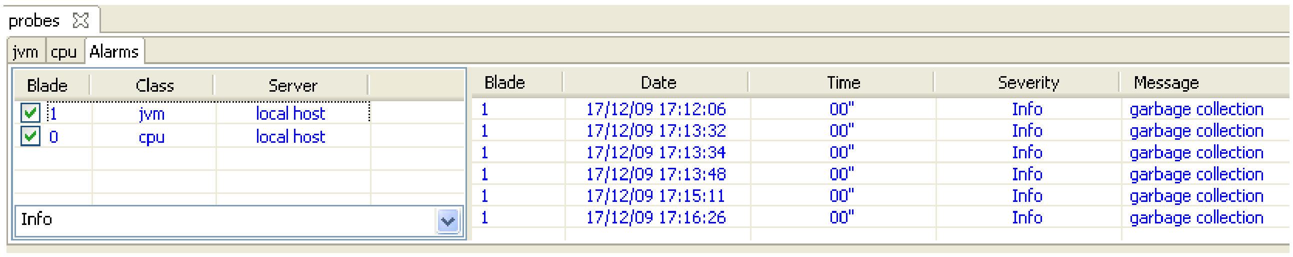


Jvm



The jvm probe monitors free memory in currently allocated heap (MB), used memory % with regard to currently allocated heap, free % of maximum allocatable memory heap.

Alarms



Blade	Class	Server
✓ 1	jvm	local host
✓ 0	cpu	local host

Blade	Date	Time	Severity	Message
1	17/12/09 17:12:06	00"	Info	garbage collection
1	17/12/09 17:13:32	00"	Info	garbage collection
1	17/12/09 17:13:34	00"	Info	garbage collection
1	17/12/09 17:13:48	00"	Info	garbage collection
1	17/12/09 17:15:11	00"	Info	garbage collection
1	17/12/09 17:16:26	00"	Info	garbage collection

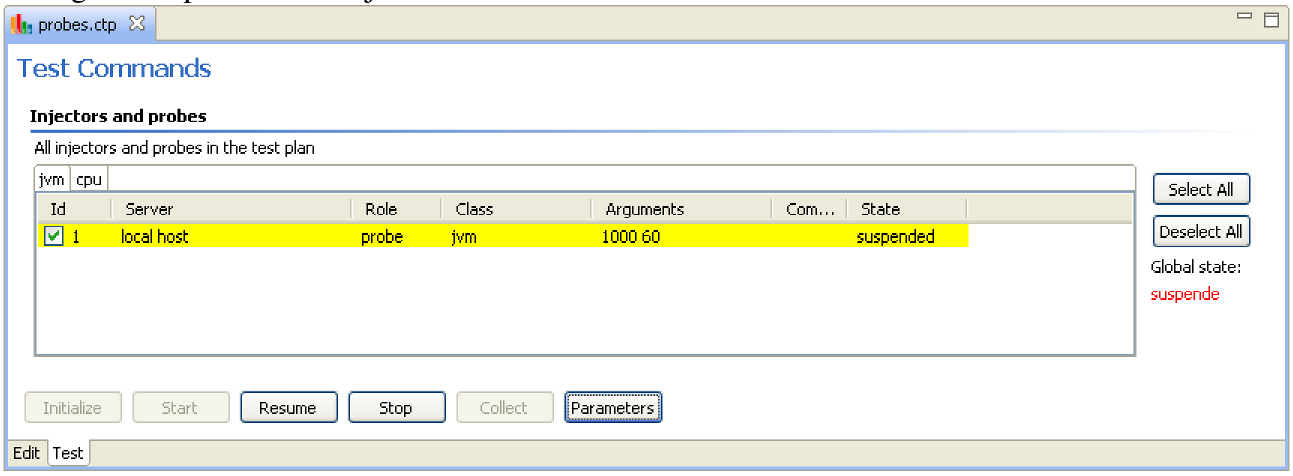
An alarm message will appear depending on certain events. For example, the jvm probe will send an alarm when a garbage collection occurs.

2.5.3. Start

Starting the test will begin the test and store data. From the start, the test will last 60 seconds as defined in probes.

2.5.4. Suspend/Resume

You can suspend the test during execution and resume it when desired. No data will be stored during the suspension nor injection will be realized.

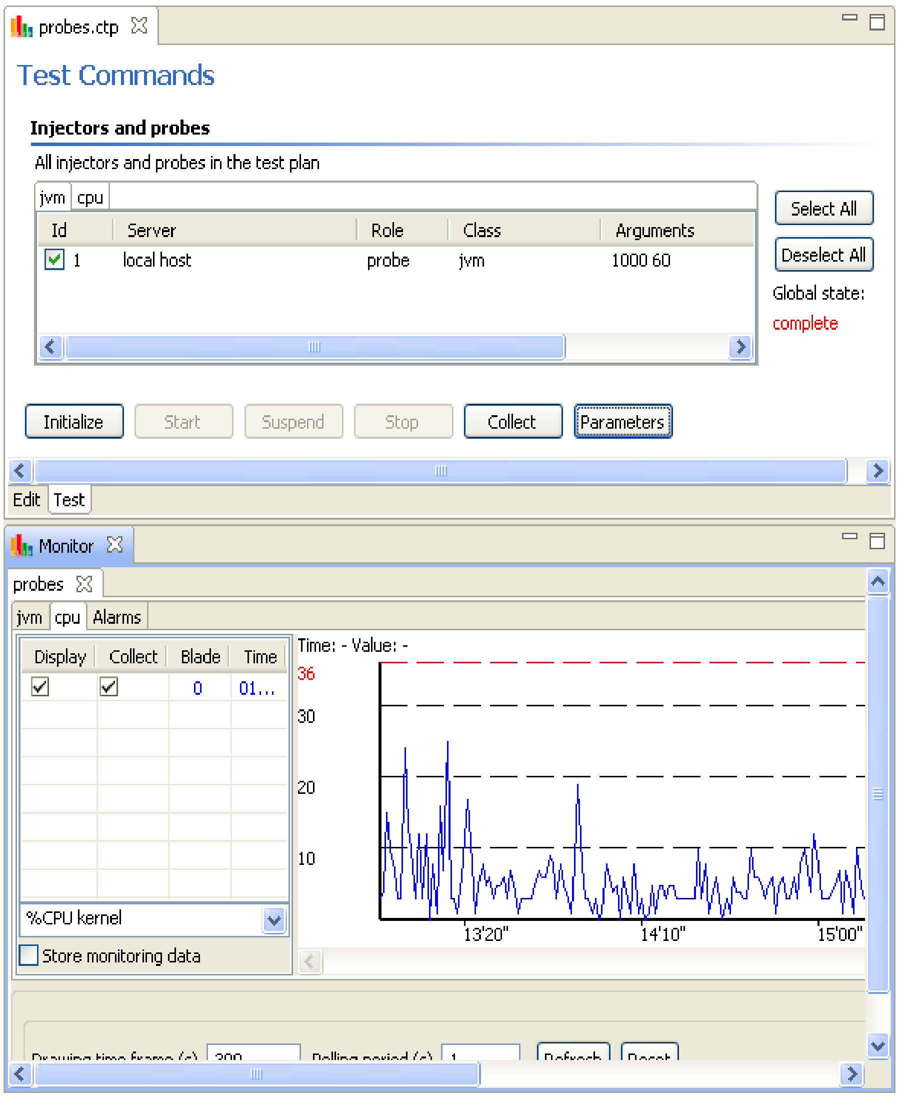


Id	Server	Role	Class	Arguments	Com...	State
✓ 1	local host	probe	jvm	1000 60		suspended

Once the execution time is finished, the test will automatically stop.

2.5.5. Collect

Once the test is completed, you will be able to collect data in the report directory with the collect button.



All injectors and probes in the test plan

Id	Server	Role	Class	Arguments
✓ 1	local host	probe	jvm	1000 60

Global state: complete

Display	Collect	Blade	Time
✓	✓	0	01...

%CPU kernel

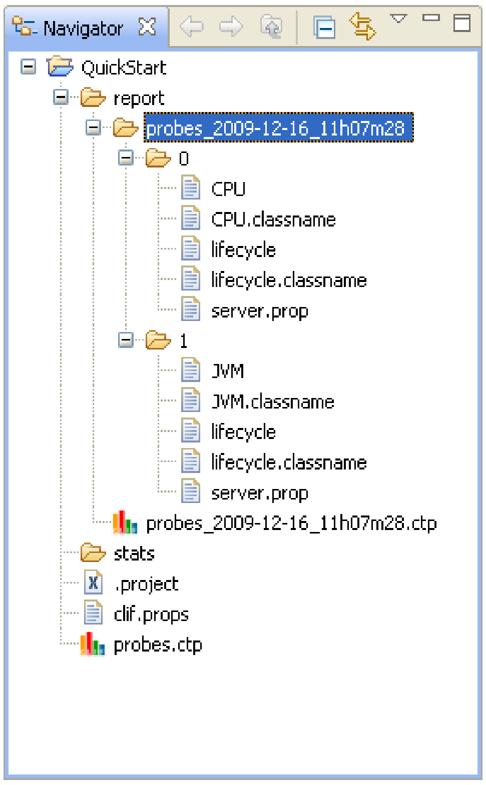
☐ Store monitoring data

Drawing time frame (s) 300 Polling period (s) 1

This should be immediate since all probes are locals.

2.6. Results

Once collected, test data will be available in report directory.



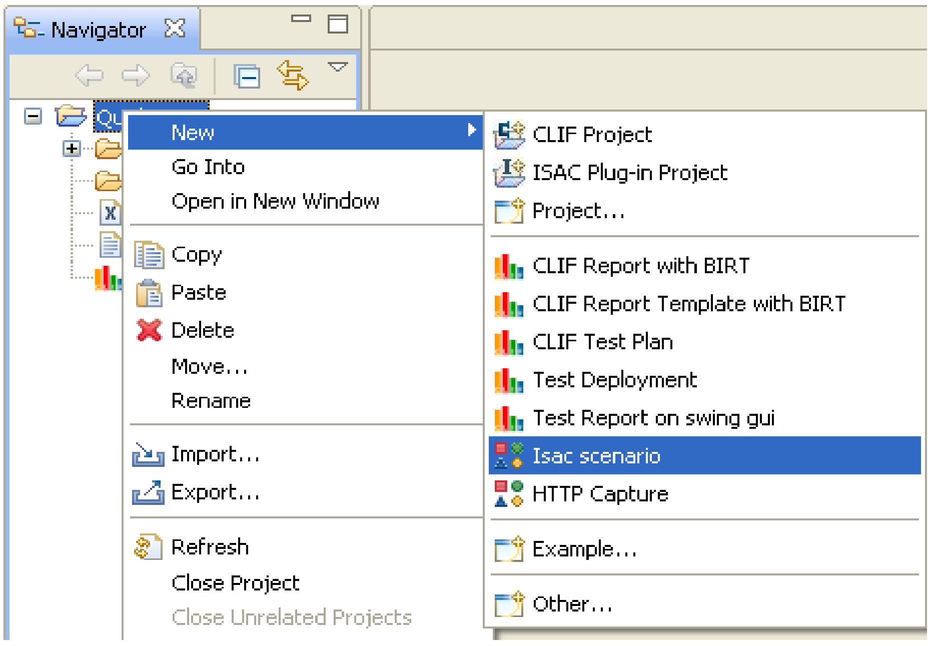
```
QuickStart
  report
    probes_2009-12-16_11h07m28
      0
        CPU
        CPU.classname
        lifecycle
        lifecycle.classname
        server.prop
      1
        JVM
        JVM.classname
        lifecycle
        lifecycle.classname
        server.prop
    probes_2009-12-16_11h07m28.ctp
  stats
  .project
  clif.props
  probes.ctp
```

3. Http scenario

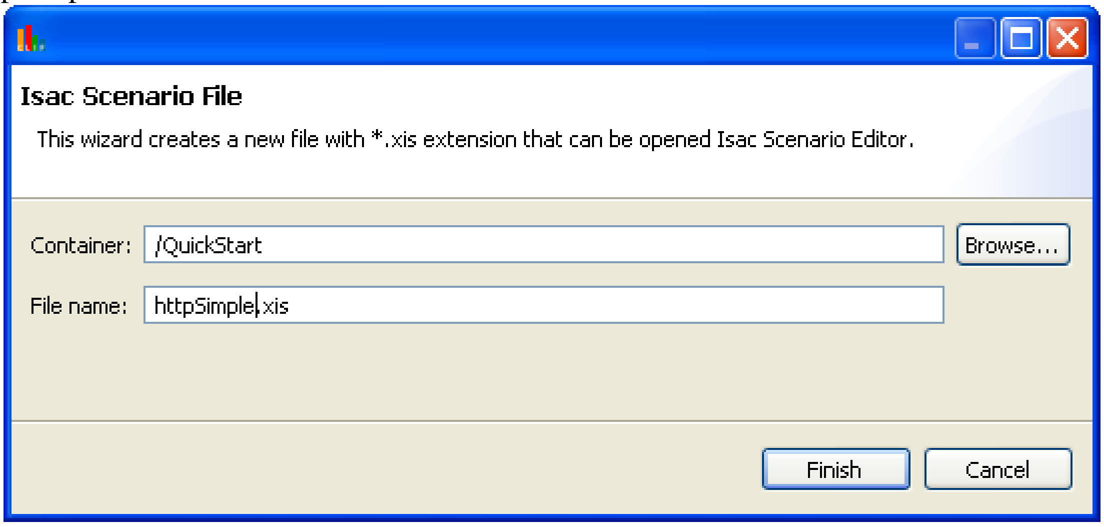
We will create a scenario to produce a HTTP get request. This scenario will define actions to execute during the test. The scenario comes with a load profile defining the number of worker who will execute the actions and how those workers are balanced in time.

3.1. Create an Isac Scenario

First, we will create an Isac Scenario with New → Isac Scenario



Use httpSimple.xis as the file name.



Isac Scenario File

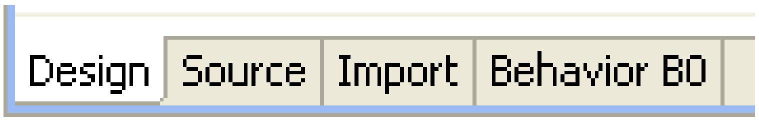
This wizard creates a new file with *.xis extension that can be opened Isac Scenario Editor.

Container: /QuickStart

File name: httpSimple.xis

3.2. Edit Isac Scenario

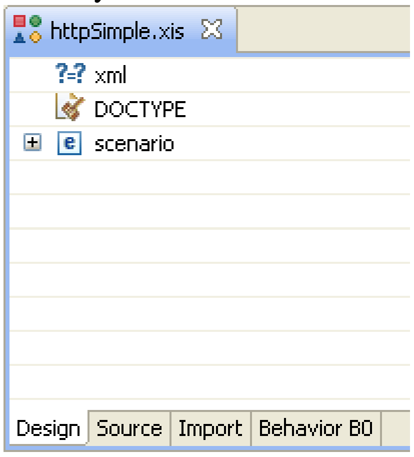
CLIF will open the editor for the Isac scenario. It is stored in a XML format and you have different ways of editing the scenario files accessible by different tabs.



3.2.1. Isac Editors tabs

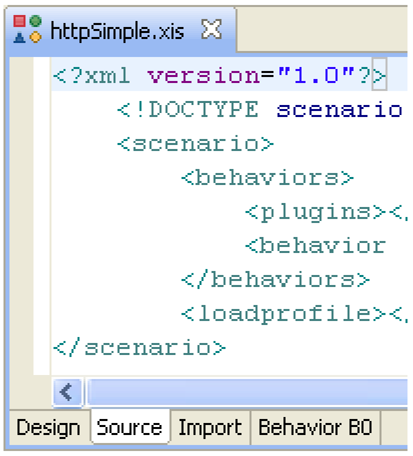
Design

First is the design tab which lets you analyze the XML structure.



Source

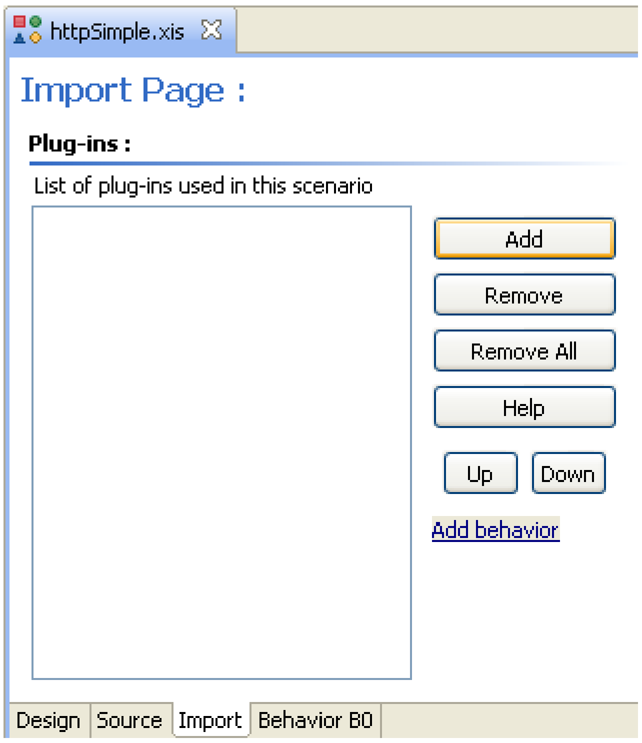
The Source tab is the plain XML text editor.



```
<?xml version="1.0"?>
    <!DOCTYPE scenario
    <scenario>
        <behaviors>
            <plugins><
            <behavior
        </behaviors>
        <loadprofile><
</scenario>
```

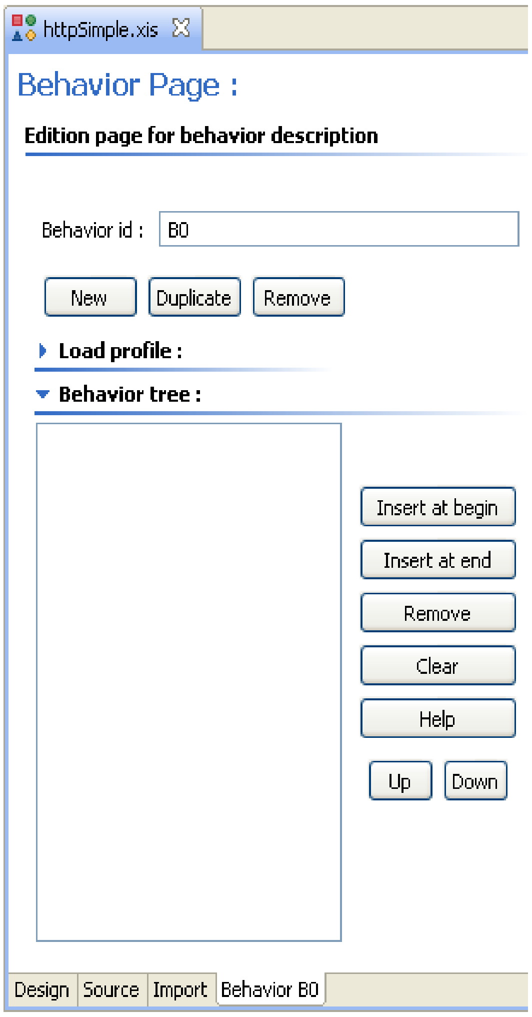
Import

Import tab will let you add plugins in your scenario. Plugins are plugable modules which are used to execute actions during test. There is already a numbers of plugins available which can be used to cover most case for your injection scenario.



Behavior

You will have a tab for each existing behavior. This tab lets you edit your scenario in a user friendly way. You will be able to insert nodes, drag&drop, copy/paste, edit the load profiles...



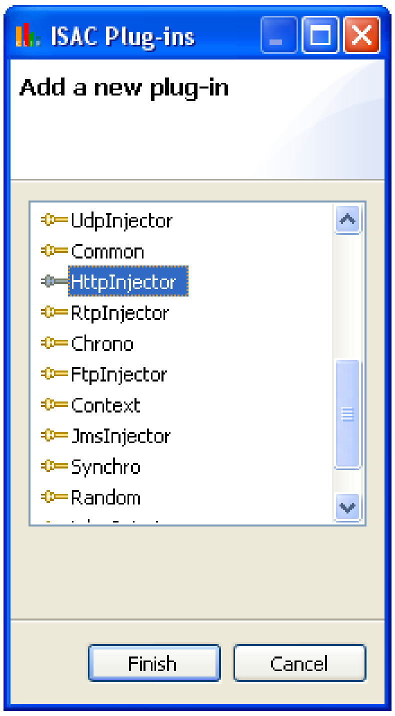
Behavior id : B0

3.3. Our first scenario

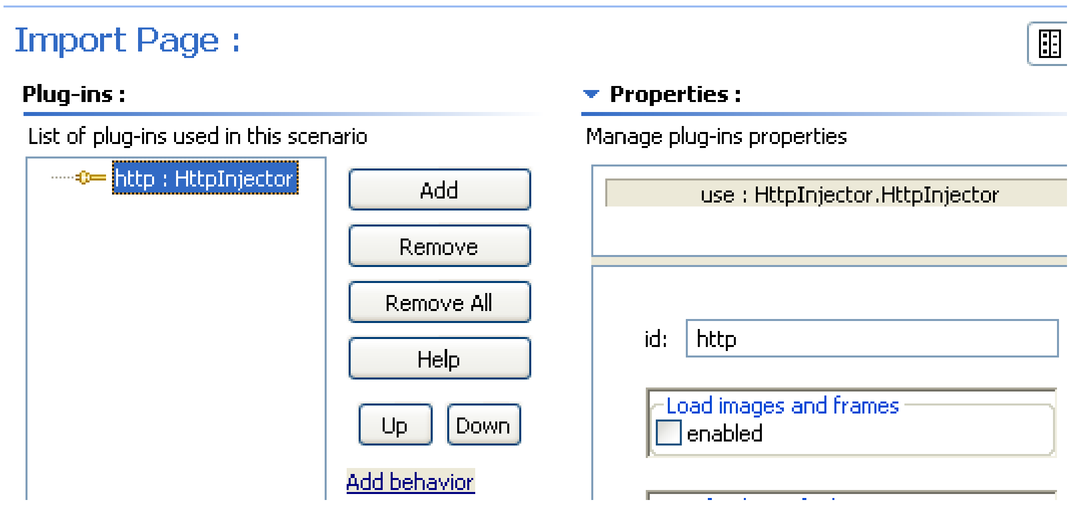
Let's write our first scenario. We will import all necessary plug-ins to realize HTTP injection.

3.3.1. Import plug-ins

First, import the HttpInject plug-in.



Change the HttpInjector's id to http

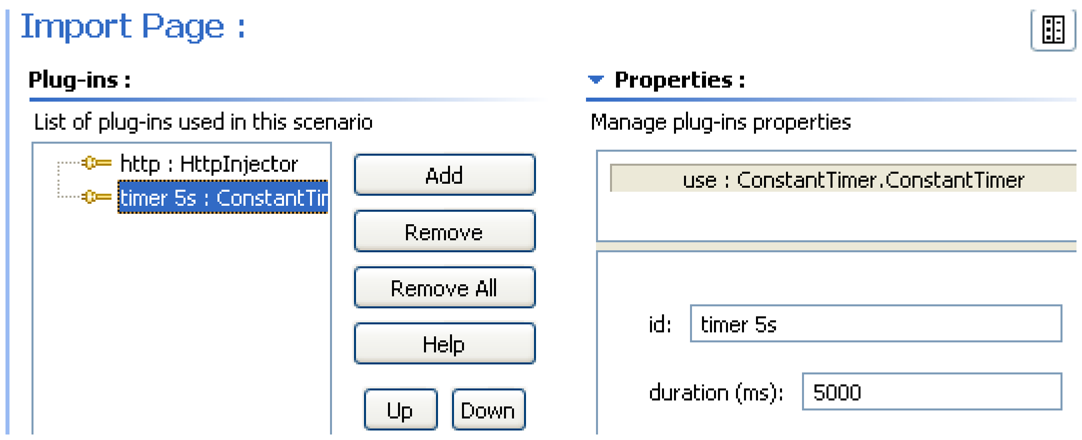


use : HttpInjector.HttpInjector

id: http

enabled [ ]

Import a ConstantTimer, rename it "timer 5s" and set the duration to 5000ms.


Import Page :

Plug-ins :

List of plug-ins used in this scenario

http : HttpInjector

timer 5s : ConstantTi

Properties :

Manage plug-ins properties

use : ConstantTimer.ConstantTimer

id: timer 5s

duration (ms): 5000

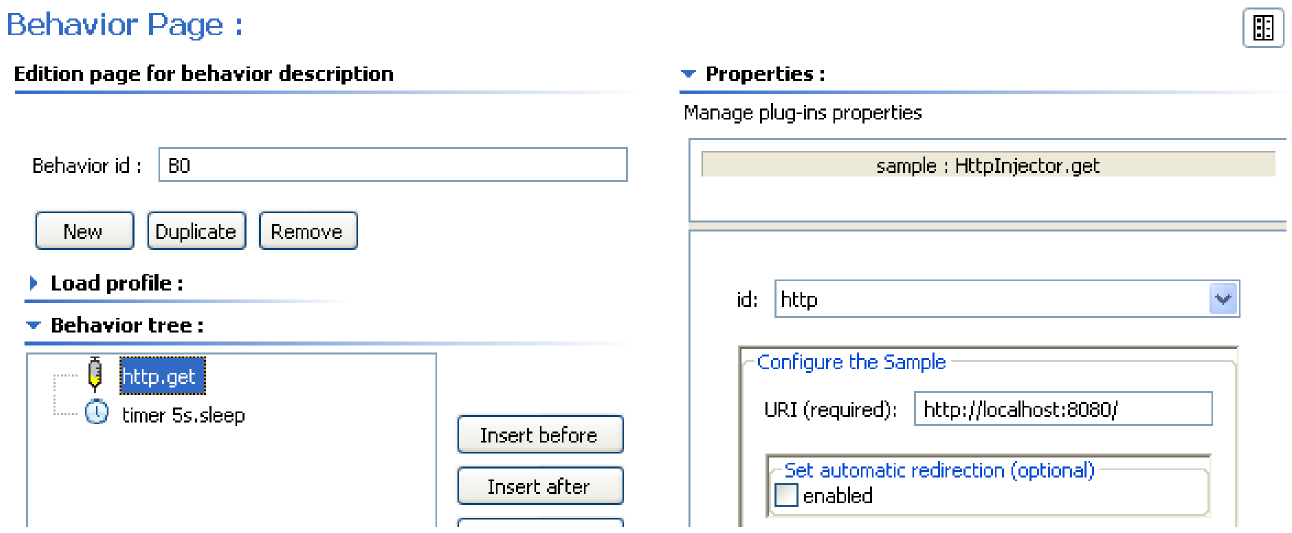
That's all the plug-ins we need.

3.3.2. Write scenario

Switch to Behavior tab to edit the scenario.

Actions

Insert a http.get action and a sleep action of the “timer 5s”.


Behavior Page :

Edition page for behavior description

Behavior id : B0

http.get

timer 5s.sleep

Properties :

Manage plug-ins properties

sample : HttpInjector.get

id: http

Configure the Sample

URI (required): http://localhost:8080/

Set automatic redirection (optional)

☐ enabled

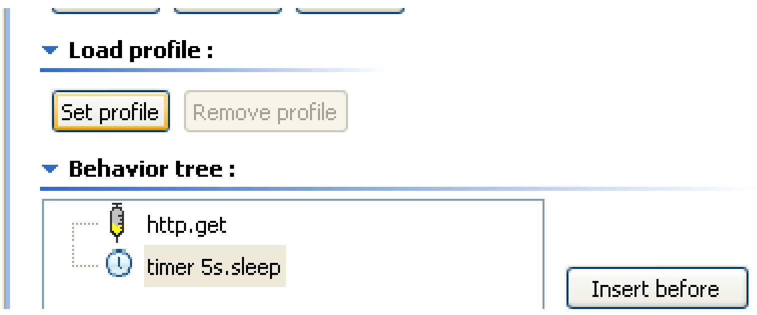
This scenario will realize an Http get action on the given URL and sleep 5 seconds.

For the URL, don't use public site to run your load testing, use your own site. Alternatively, you may launch a light servlet container like Jetty (http://www.eclipse.org/jetty/downloads.php). You just need to get the distribution, unzip it and launch it with “`java -jar start.jar DEBUG=true`”, the container will be accessible at the URL http://localhost:8080/ and you will have direct feedback of the requests with debug logs.

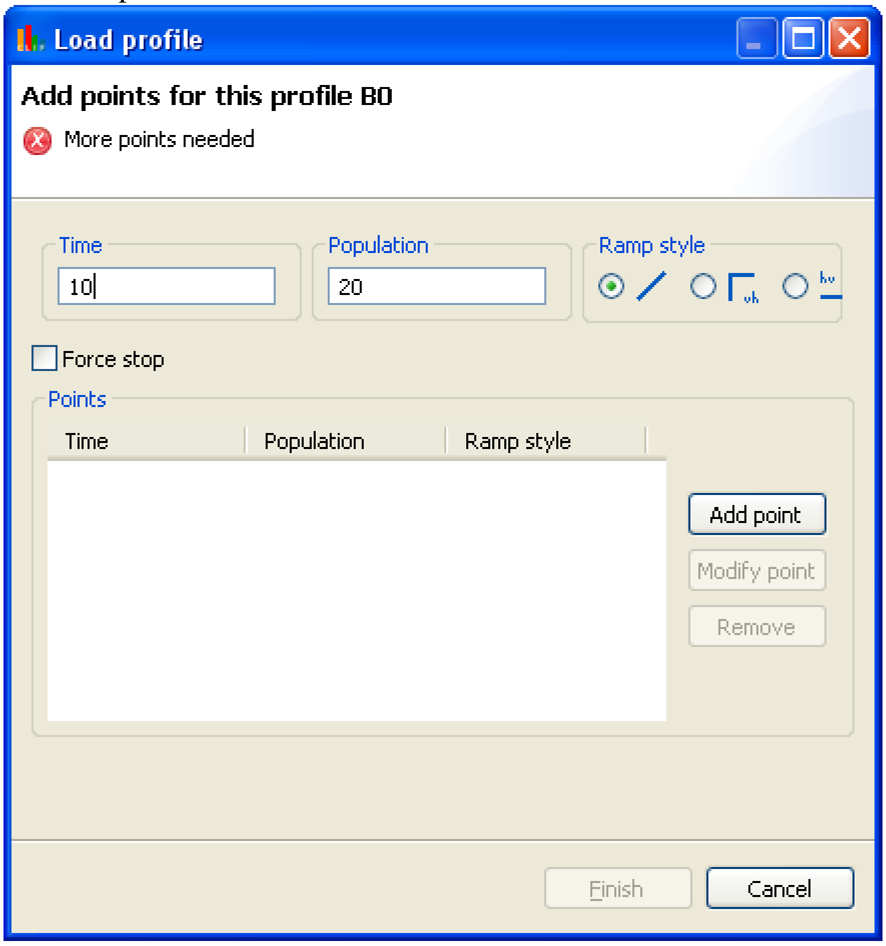
Load profile

You need to define how many actions will be realized and how it will be balanced. This is the role of the load profile.

Open Load Profile and click on “Set Profile”



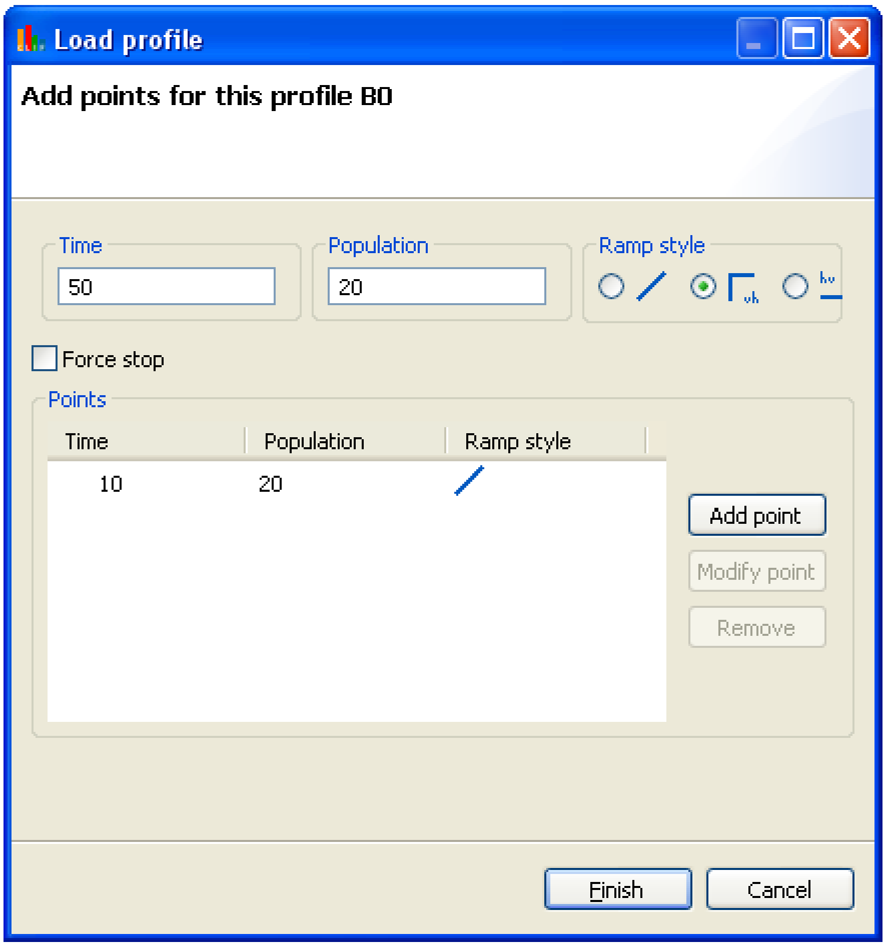
Create a straight line charge with 20 users for the first 10 seconds. Fill time and population box with these values and add the point.



Time: 10
Population: 20
Ramp style: [✓ line]

Time	Population	Ramp style

Then, stay at 20 users from 10 to 50 seconds with a plateau style.



Load profile

Add points for this profile B0

Time: 50

Population: 20

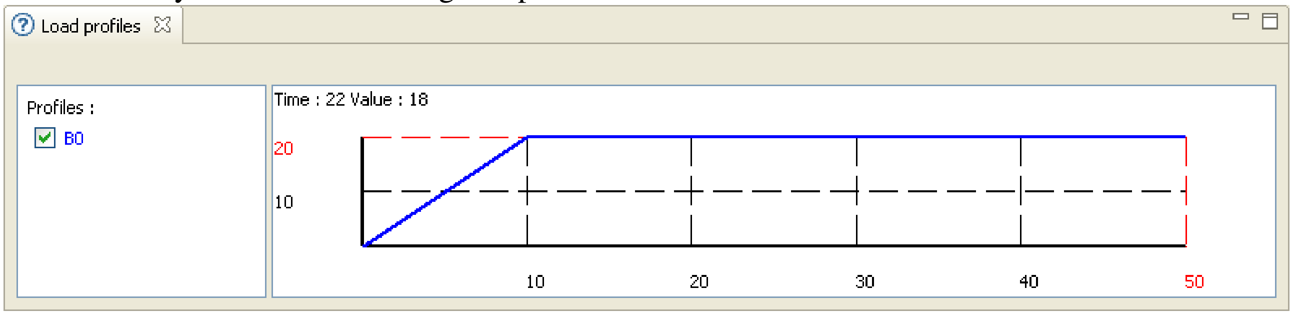
Ramp style: ○ ╱ [✓ ● ┌] ○

☐ Force stop

Points

Time	Population	Ramp style
10	20	╱

Validate and you should have the given profile.



Profiles :

✓ B0

Time : 22 Value : 18

Your scenario is now ready to use.

3.4. Deploy test plan with injector

We will create a new Test plan with our new scenario.

Create a Test Plan named httpSimple.ctp (see Test Plan Creation).

3.4.1. Add injector to Test Plan

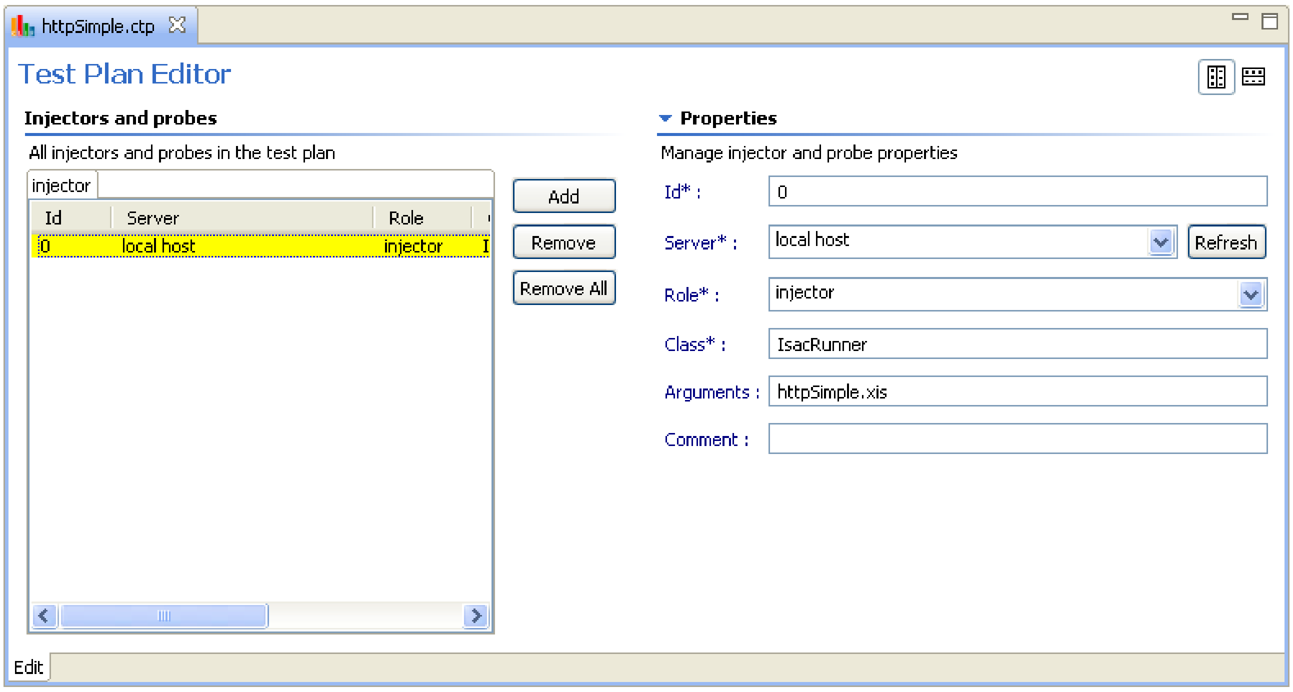
Add an injector with following parameters:

-Server: local host

-Role: Injector

-class: IsacRunner

-arguments: httpSimple.xis



Id	Server	Role
0	local host	injector

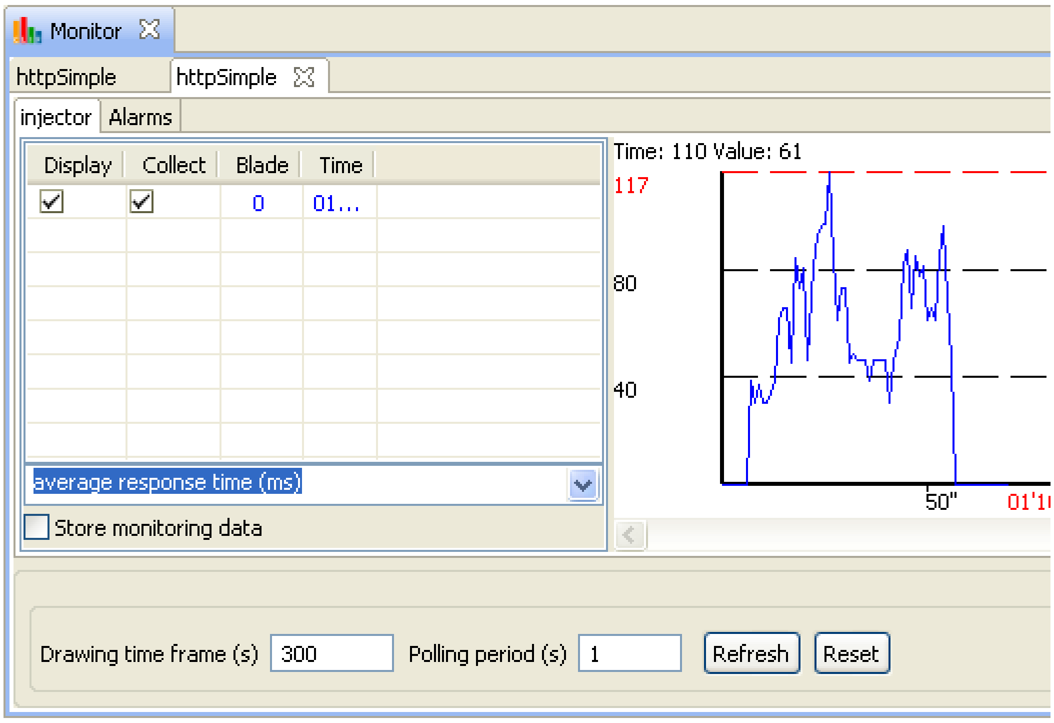
Id* : 0
Server* : local host
Role* : injector
Class* : IsacRunner
Arguments : httpSimple.xis
Comment :

3.4.2. Deploy and execution

Deploy the Test Plan (see Test Deployment).

Initialize and start the test (see Test execution).

Starting the test will trigger the injection. Once the test is complete, you may review different data like the average response time.



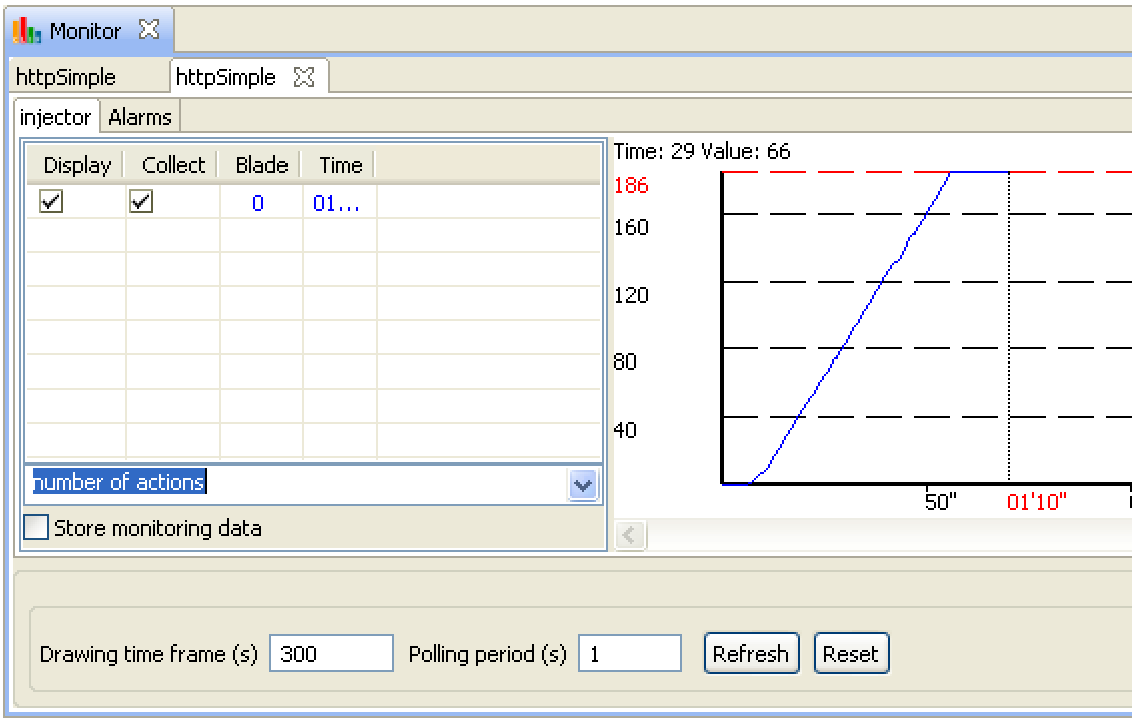
Display	Collect	Blade	Time
✓	✓	0	01...

average response time (ms)

Store monitoring data

Drawing time frame (s) 300 Polling period (s) 1

Or the total numbers of action of the test.



Display	Collect	Blade	Time
✓	✓	0	01...

number of actions

Store monitoring data

Drawing time frame (s) 300 Polling period (s) 1

You can now collect results in order to analyze them later (see Collect).

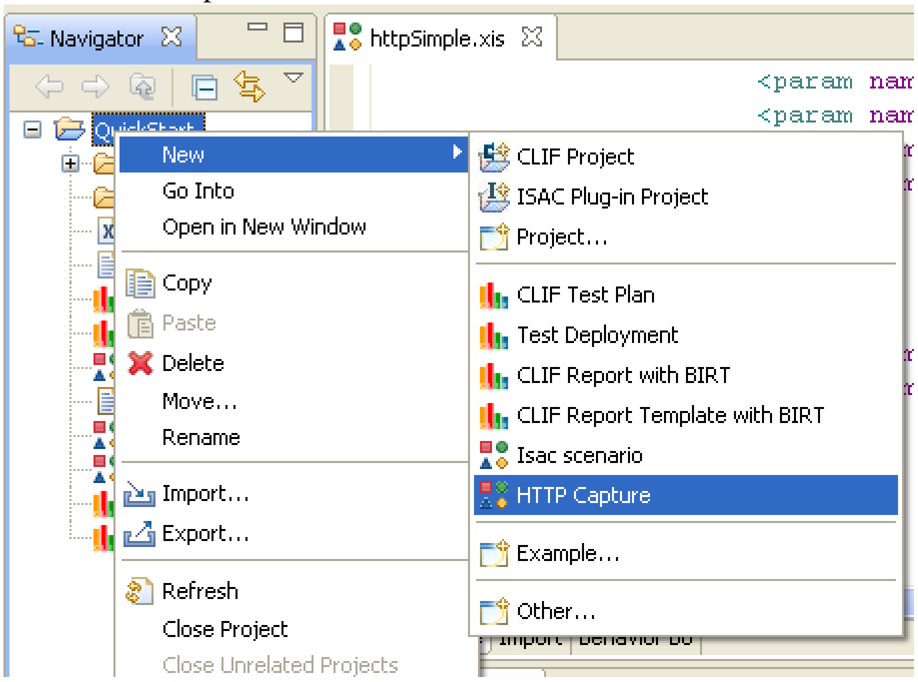
4. Http capture and replay

We will create a Test based on a scenario created by HttpCapture.

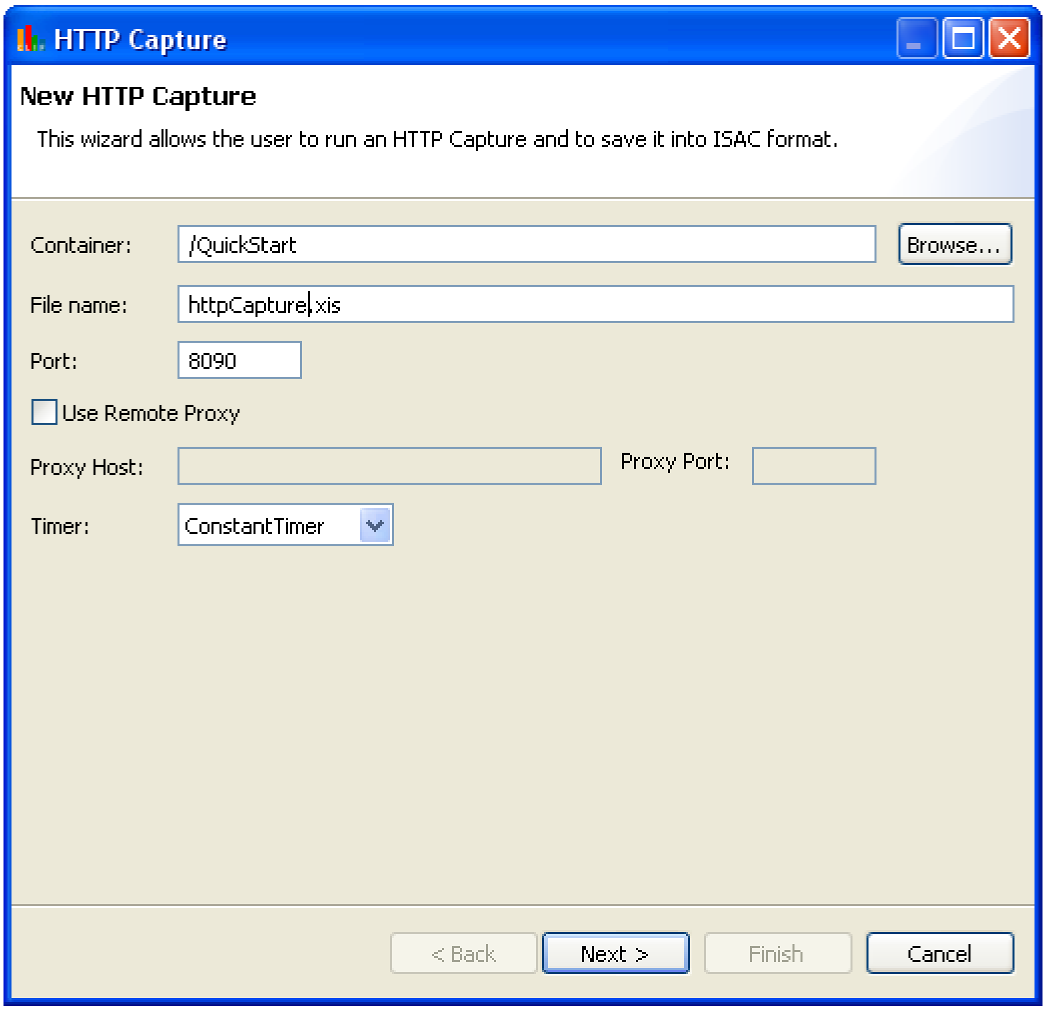
4.1. Use Http capture

4.1.1. Launch Http capture

First, select New → HTTP Capture



Name the capture file “httpCapture.xis” and validate


HTTP Capture

New HTTP Capture

This wizard allows the user to run an HTTP Capture and to save it into ISAC format.

Container: /QuickStart

File name: httpCapture.xis

Port: 8090

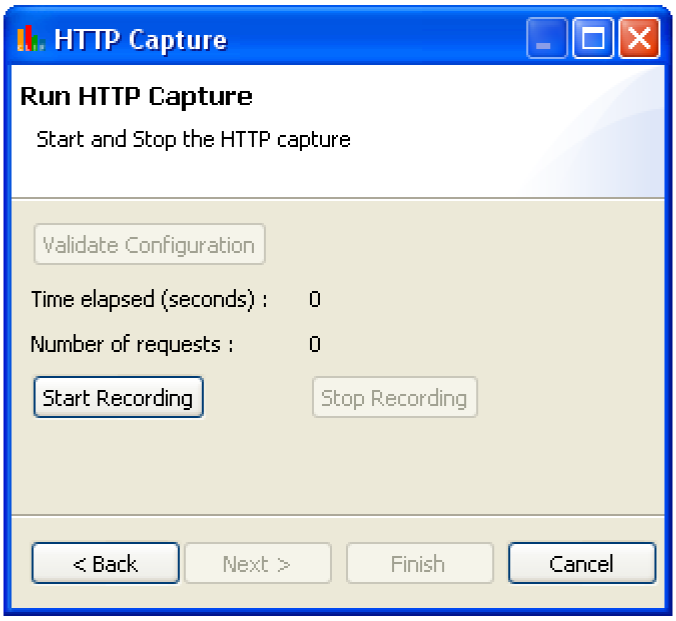
Use Remote Proxy

Proxy Host:

Proxy Port:

Timer: ConstantTimer

Validate the configuration, it will start the proxy needed for capture.


HTTP Capture

Run HTTP Capture

Start and Stop the HTTP capture

Time elapsed (seconds) : 0

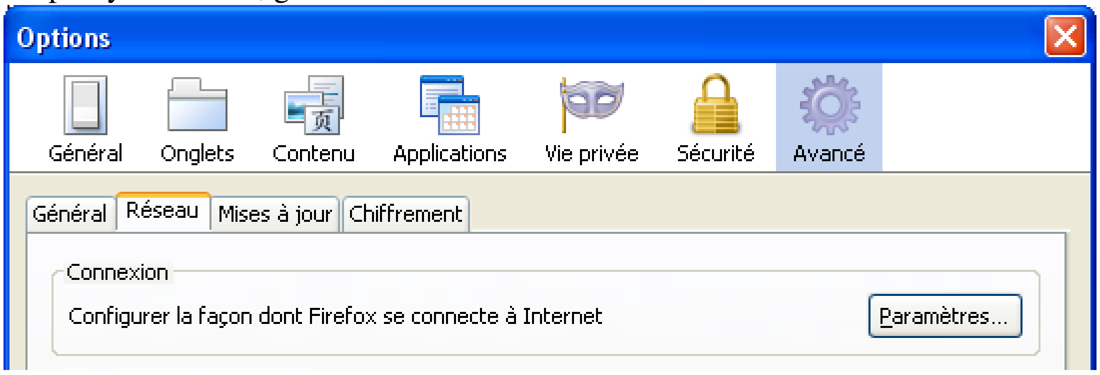
Number of requests : 0

Now, you can make requests through the proxy and it will be captured and converted in a scenario once you start the recording.

4.1.2. Configure your navigator with the created proxy

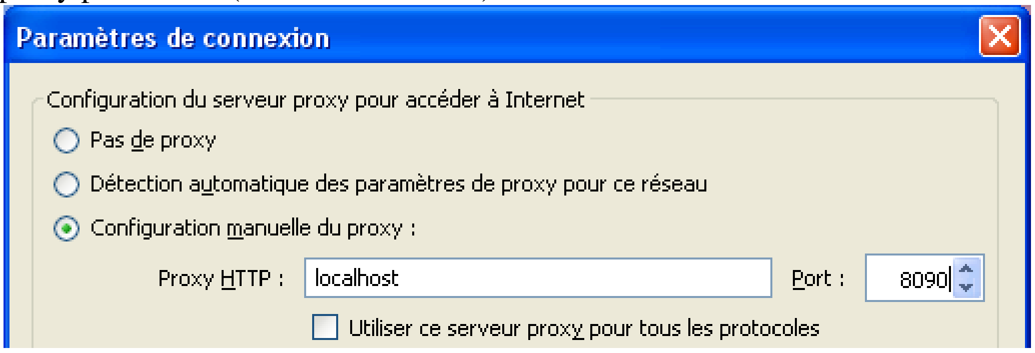
Firefox

To use the proxy in Firefox, go to Tools → Advanced → Network tab → Parameters



Configurer la façon dont Firefox se connecte à Internet

Enter the proxy parameters (localhost and 8090) and validate.

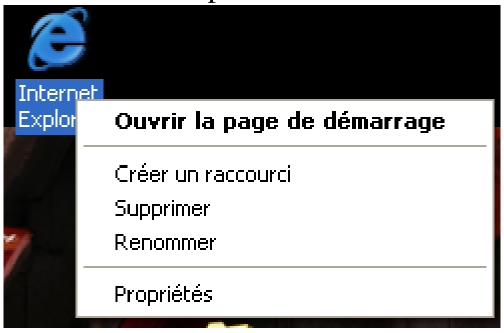


( ) Pas de proxy
( ) Détection automatique des paramètres de proxy pour ce réseau
[✓ Configuration manuelle du proxy :]
Proxy HTTP : localhost
Port : 8090
[ ] Utiliser ce serveur proxy pour tous les protocoles

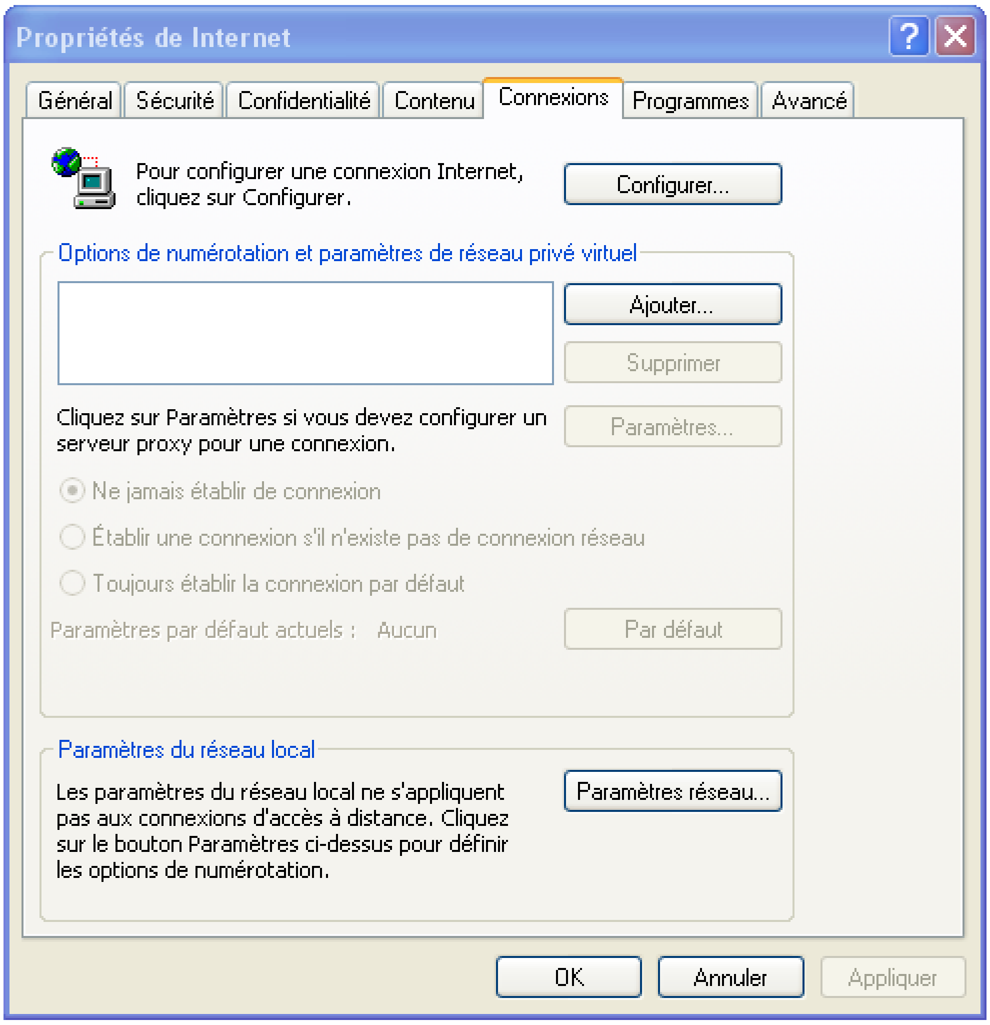
Just use Firefox normally after this.

Internet explorer

Right-click on Internet Explorer and chose Properties



Go to the connection tab and chose network parameters



Pour configurer une connexion Internet, cliquez sur Configurer.

Options de numérotation et paramètres de réseau privé virtuel

Cliquez sur Paramètres si vous devez configurer un serveur proxy pour une connexion.

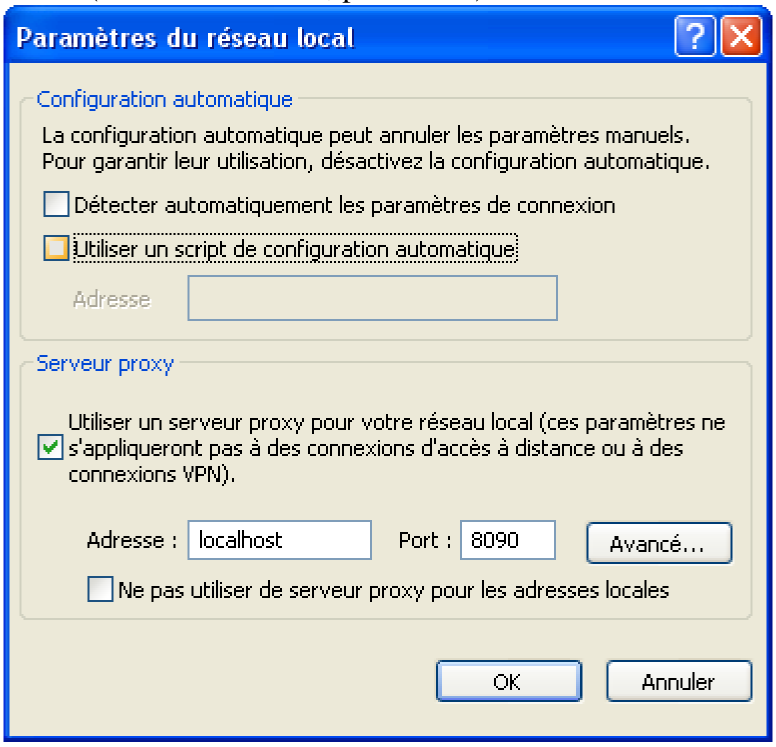
- [✓ Ne jamais établir de connexion]
- Établir une connexion s'il n'existe pas de connexion réseau
- Toujours établir la connexion par défaut

Paramètres par défaut actuels : Aucun

Paramètres du réseau local

Les paramètres du réseau local ne s'appliquent pas aux connexions d'accès à distance. Cliquez sur le bouton Paramètres ci-dessus pour définir les options de numérotation.

Enter the proxy parameter (address: localhost, port: 8090) and validate



Configuration automatique

La configuration automatique peut annuler les paramètres manuels. Pour garantir leur utilisation, désactivez la configuration automatique.

- ☐ Détecter automatiquement les paramètres de connexion
- ☐ Utiliser un script de configuration automatique

Adresse

Serveur proxy

- ✓ Utiliser un serveur proxy pour votre réseau local (ces paramètres ne s'appliqueront pas à des connexions d'accès à distance ou à des connexions VPN).

Adresse : localhost Port : 8090

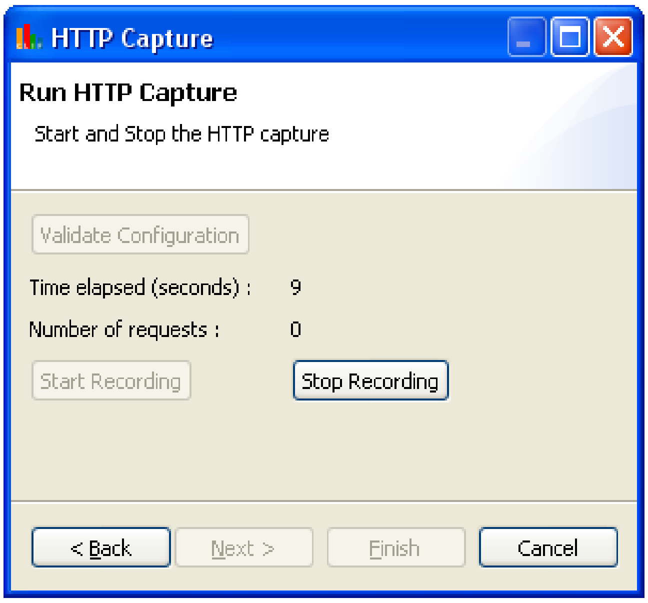
- ☐ Ne pas utiliser de serveur proxy pour les adresses locales

You can now use Internet Explorer to make HTTP request which will be captured.

Don't forget to remove proxy once you finished your capture.

4.1.3. Record the scenario

Start the recording.



HTTP Capture

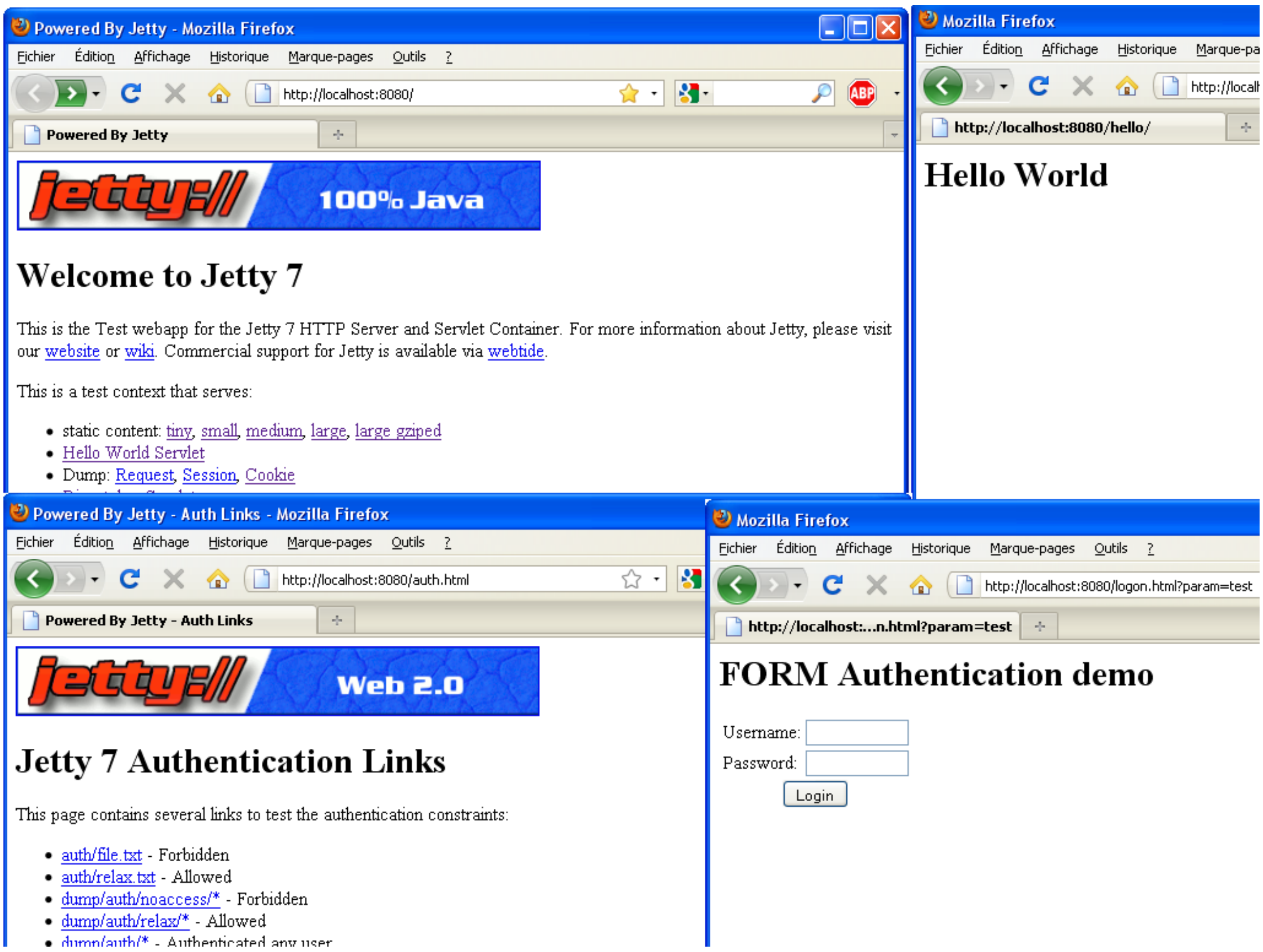
Run HTTP Capture

Start and Stop the HTTP capture

Time elapsed (seconds) : 9

Number of requests : 0

Use your navigator. The number of requests should increase accordingly.


Powered By Jetty - Mozilla Firefox

http://localhost:8080/

Welcome to Jetty 7

This is the Test webapp for the Jetty 7 HTTP Server and Servlet Container. For more information about Jetty, please visit our website or wiki. Commercial support for Jetty is available via webtide.

This is a test context that serves:

- static content: tiny, small, medium, large, large gziped
- Hello World Servlet
- Dump: Request, Session, Cookie

Mozilla Firefox

http://localhost:8080/hello/

Hello World

Powered By Jetty - Auth Links - Mozilla Firefox

http://localhost:8080/auth.html

Jetty 7 Authentication Links

This page contains several links to test the authentication constraints:

- auth/file.txt - Forbidden
- auth/relax.txt - Allowed
- dump/auth/noaccess/* - Forbidden
- dump/auth/relax/* - Allowed
- dump/auth/* - Authenticated any user

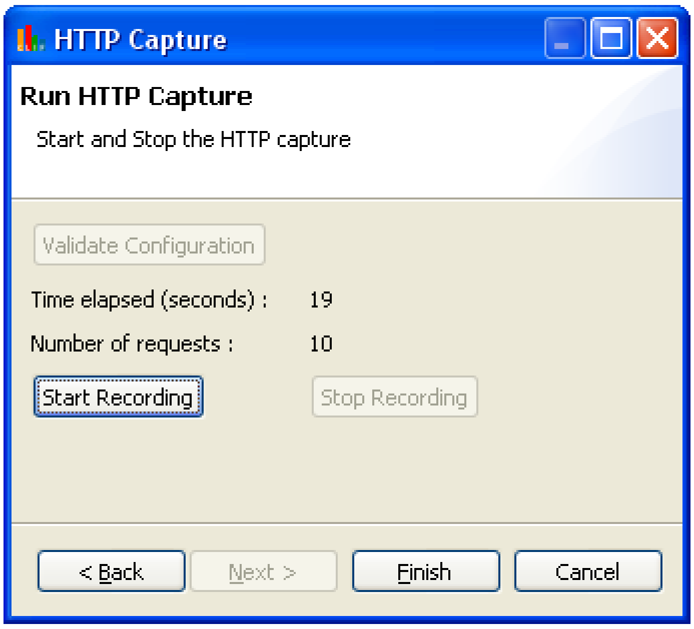
Mozilla Firefox

http://localhost:8080/logon.html?param=test

FORM Authentication demo

Username:
Password:
Login

Stop recording when have finished.


HTTP Capture

Run HTTP Capture

Start and Stop the HTTP capture

Validate Configuration

Time elapsed (seconds) : 19

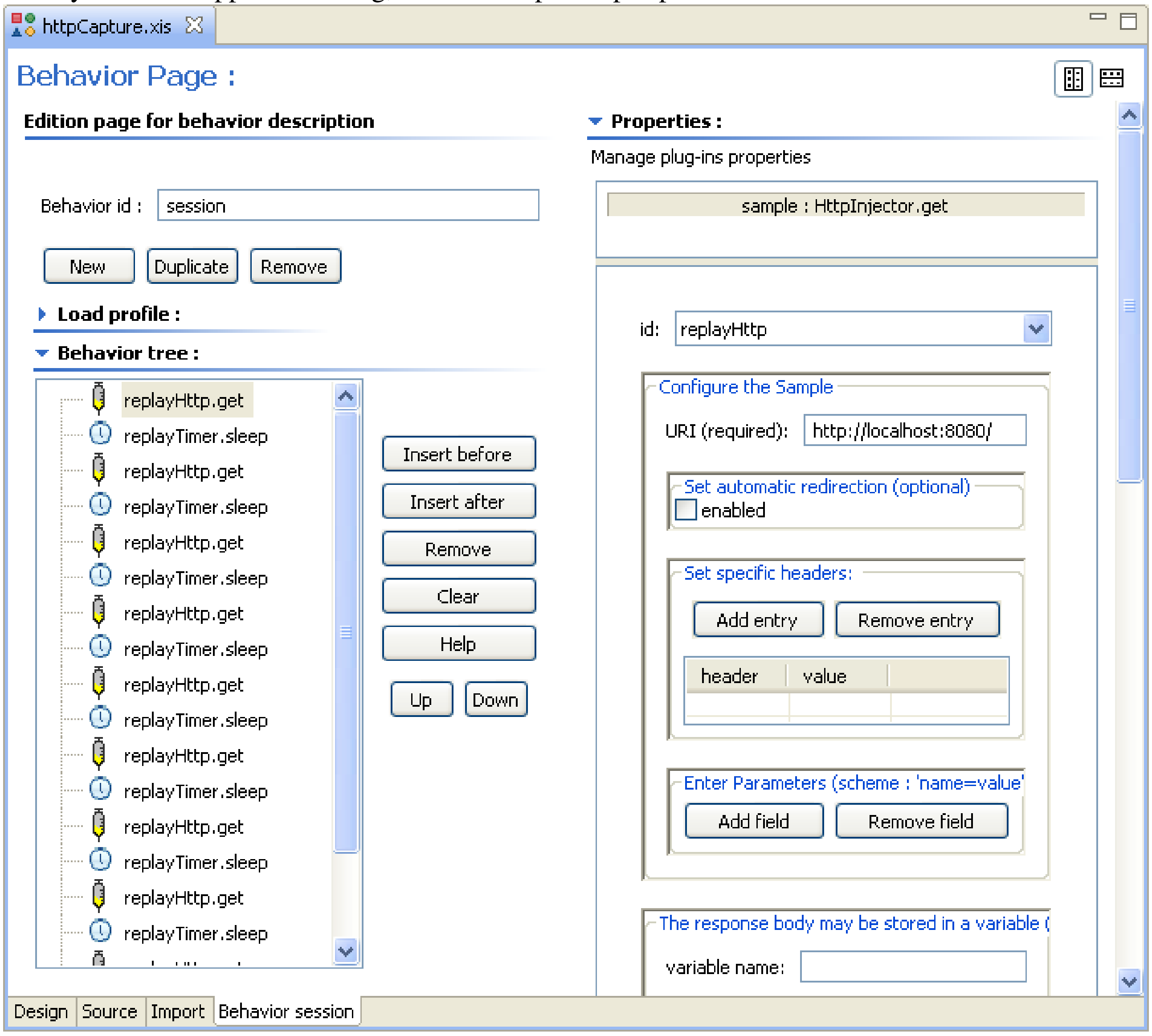
Number of requests : 10

Start Recording   Stop Recording

< Back   Next >   Finish   Cancel

4.2. Edit HttpCapture scenario

Once you have stopped recording, finish and open httpCapture.xis.



Behavior id : session

sample : HttpInjector.get

id: replayHttp

URI (required): http://localhost:8080/

☐ enabled

variable name:

```
replayHttp.get
replayTimer.sleep
replayHttp.get
replayTimer.sleep
replayHttp.get
replayTimer.sleep
replayHttp.get
replayTimer.sleep
replayHttp.get
replayTimer.sleep
replayHttp.get
replayTimer.sleep
replayHttp.get
replayTimer.sleep
replayHttp.get
replayTimer.sleep
```

You need to define a load profile before using this scenario in an injector (see Load profile). You might as well edit it to match your needs.

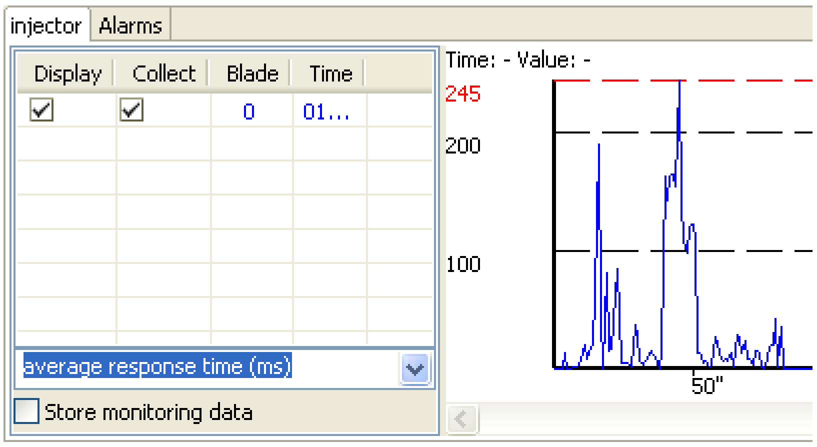
4.3. Deployment and execution of httpCapture

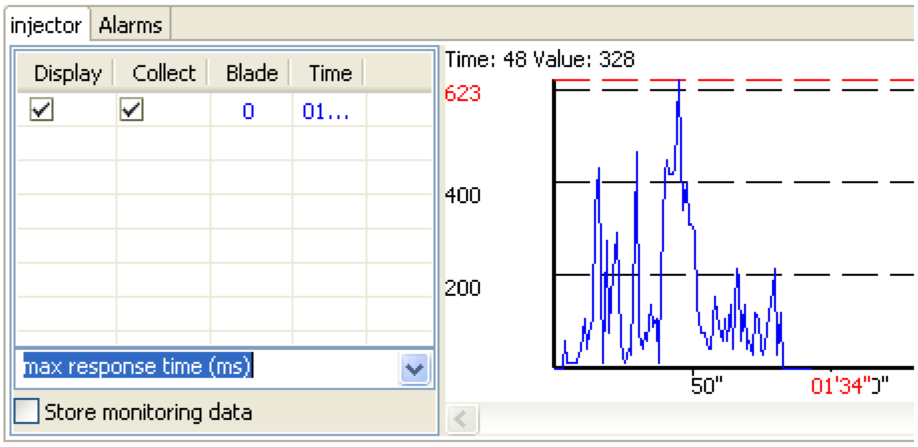
Create a test plan httpCapture.ctp (see Test Plan Creation)

Add an injector with parameters: (Role: Injector; class: IsacRunner; Arguments: httpCapture.xis) (see Add injector to Test Plan)

Deploy the test plan (see Deploy and execution)

Initialize, start and collect results.





You have completed the QuickStart. You know may discover other CLIF features by reading the User manual.

You may also extend CLIF yourself by defining new probes and injectors with the Developer manual.