

HOW TO

Use CLIF v2 and ISAC PLUGINS



with the ECLIPSE RCP console

<http://clif.ow2.org/>

Copyright © 2006-2009 France Telecom

License information: <http://creativecommons.org/licenses/by-nc-sa/3.0>

Table of contents

1. Introduction to ISAC Plug-ins.....	3
1.1. What is an ISAC Plug-ins.....	3
1.2. Installation.....	3
2. ISAC is a Scenario Architecture for CLIF.....	4
2.1. Defining an ISAC scenario for LDAP.....	4
2.2. Record your ISAC scenario.....	5
2.2.1. <i>Create your project and your ISAC scenario.....</i>	<i>5</i>
2.2.2. <i>Import ISAC plugins.....</i>	<i>8</i>
2.2.3. <i>Define your behavior.....</i>	<i>12</i>
2.2.4. <i>Load Profiles.....</i>	<i>21</i>
3. 4. Define your test plan.....	25
3.1. Description of the context.....	25
3.2. Creating your test plan.....	25
3.3. Add Probes and Injectors to your test plan.....	26
3.4. Deploying and executing your test plan.....	29
4. CLIF server.....	31
4.1. Installation.....	31
4.2. Rationale.....	31
4.3. Running a registry.....	31
4.4. Configuring a CLIF server.....	32
4.5. Running a CLIF server.....	32

1. Introduction to ISAC Plug-ins

1.1. What is an ISAC Plug-ins

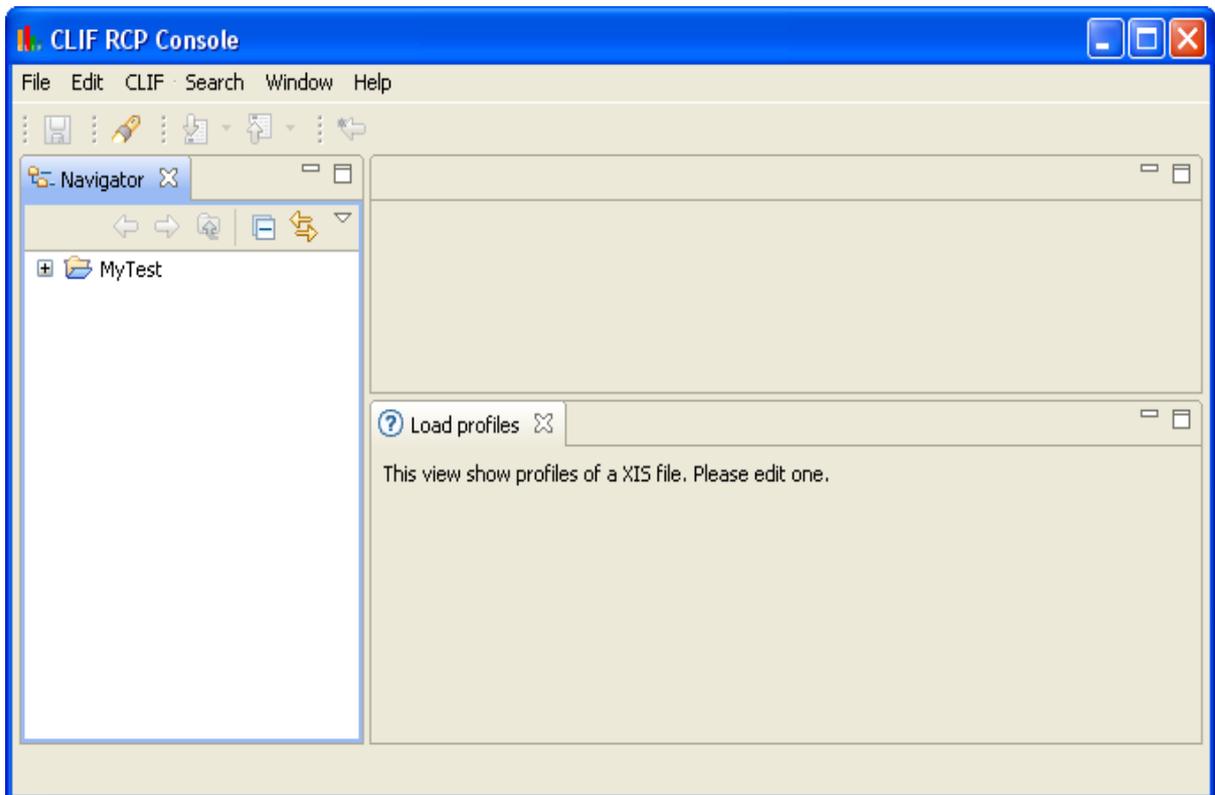
In order to actually generate traffic on a System Under Test (SUT), we need to define a behavior. A behavior can be understood as a logical definition, a kind of a skeleton. This skeleton must be associated to one or more ISAC plug-ins. Plug-ins are external Java libraries, that are responsible for:

- performing actions (i.e. generating requests) on the SUT, using and managing specific protocols whose response times will be measured (e.g. HTTP, DNS, JDBC, TCP/IP, DHCP, SIP, LDAP);
- providing conditions used by the behaviors' conditional statements (if-then-else, while, preemptive);
- providing timers to implement delays (think time), for example with specific random distributions or computed in some arbitrary way;
- providing ad hoc controls for the plug-in itself (e.g. to change some settings);
- providing support for external data provisioning (e.g. a database of product references or a file containing identifier-password pairs for some user accounts), used as parameters by the behaviors.

1.2. Installation

See Installation Manual for Eclipse-based console GUI installation.

2. ISAC is a Scenario Architecture for CLIF



With ISAC, testers are given a way to define load scenarios by combining:

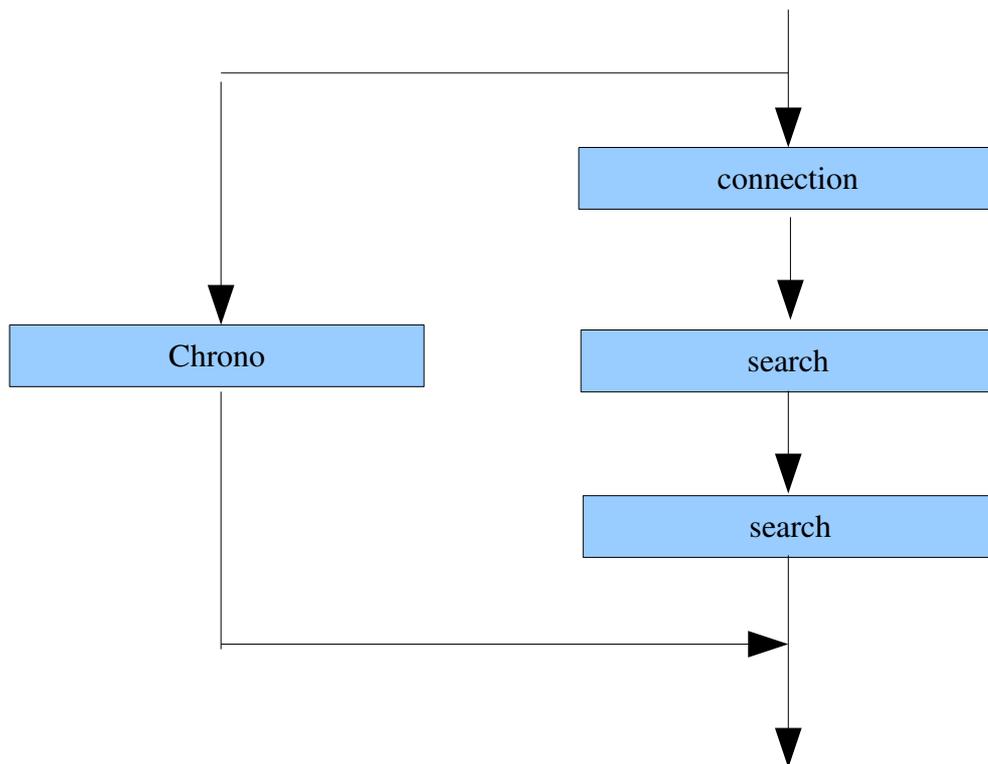
- definitions of elementary behaviors, typically representing users;
- optional definitions of load profiles setting the population (i.e. the number of active instances) of each behavior as a function of time.

2.1. Defining an ISAC scenario for LDAP

In order to make realistic scenarios corresponding to real users behaviors, actions on LDAP can be recorded as an ISAC scenario.

In our tutorial we will define an ISAC scenario which will do some actions on the LDAP directory.

We can see here the actions that our scenario will do:



In this scenario we will connect to the LDAP directory, make a search and disconnect it. We will import the LdapInjector plug-in to be able to do this action.

To externalize the data used in the scenario (mandatory parameters of the LdapInjector plug-in) we will create CSV file that will contain all the needed data. Each line of the CSV file represents the parameters needed to execute the scenario. Then we will loop on each line to make different calls to the LDAP directory. To do this we will use the CSVProvider plug-in.

In this ISAC scenario the scenario duration will be set to 1 second. We need to import the ConstantTimer plug-in.

In our load test we will use two or more injectors and we don't want to launch the test at the same moment on each injector. So we will import the Random plug-in.

Finally we want to know the response time of group of operation, that's why we will import the Chrono plug-in to insert chrono action's duration in the report.

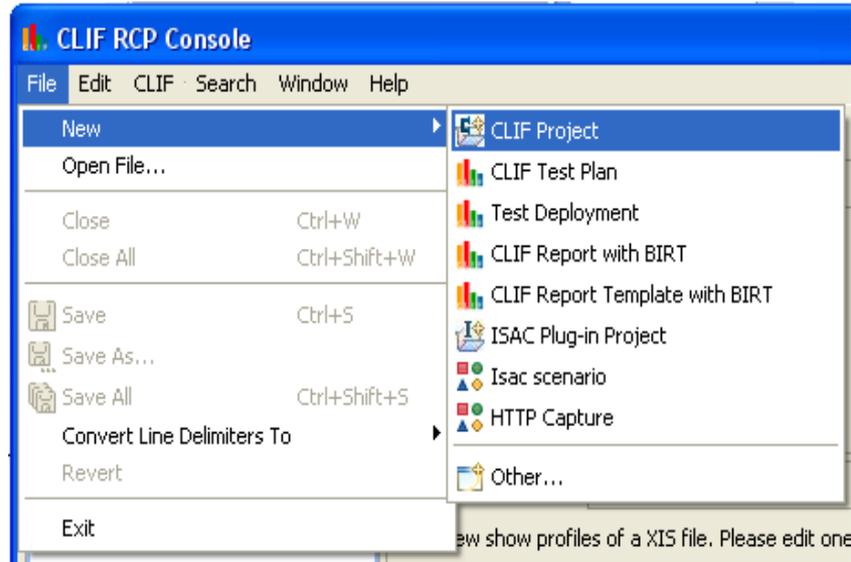
2.2. Record your ISAC scenario

2.2.1. Create your project and your ISAC scenario

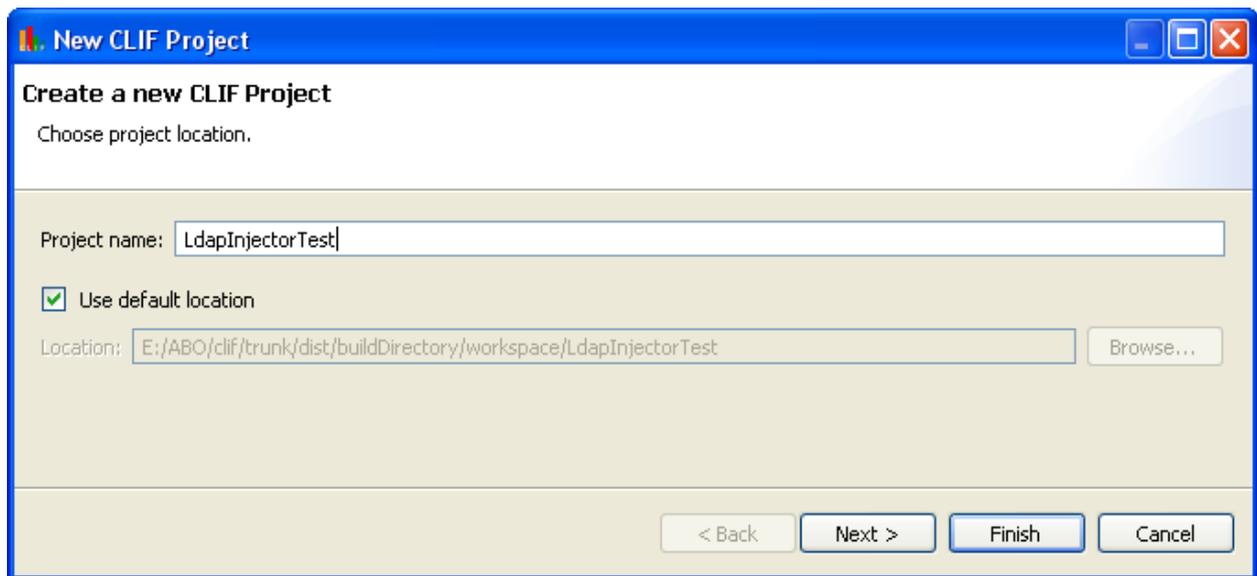
First of all you have to create a new general project:

Click on File -> New -> CLIF project

How To Use CLIF v2 and ISAC plug-ins

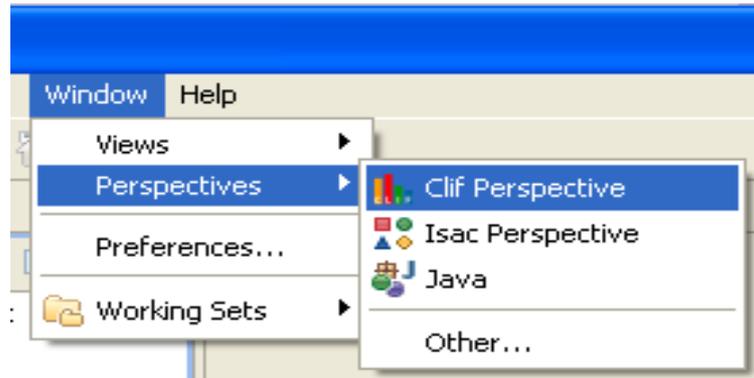


Enter your project name (you can use default location or choose an other location)

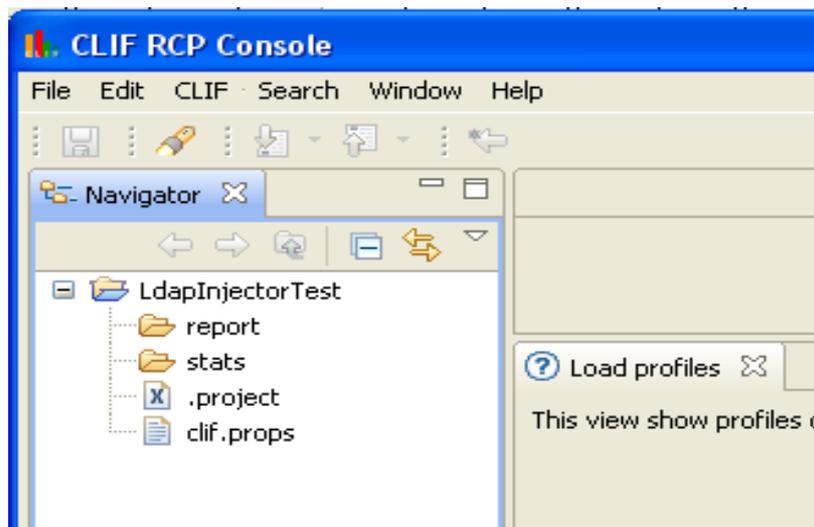


Click on Finish.

If you use your own Eclipse IDE instead of the Clif console you have to open the Clif perspective:

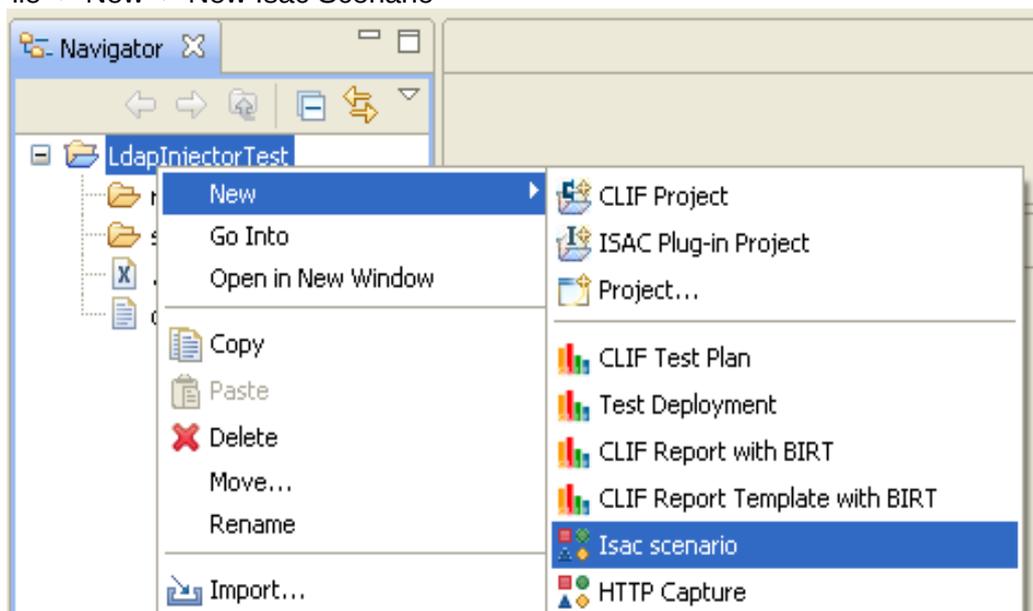


You can now see this perspective:



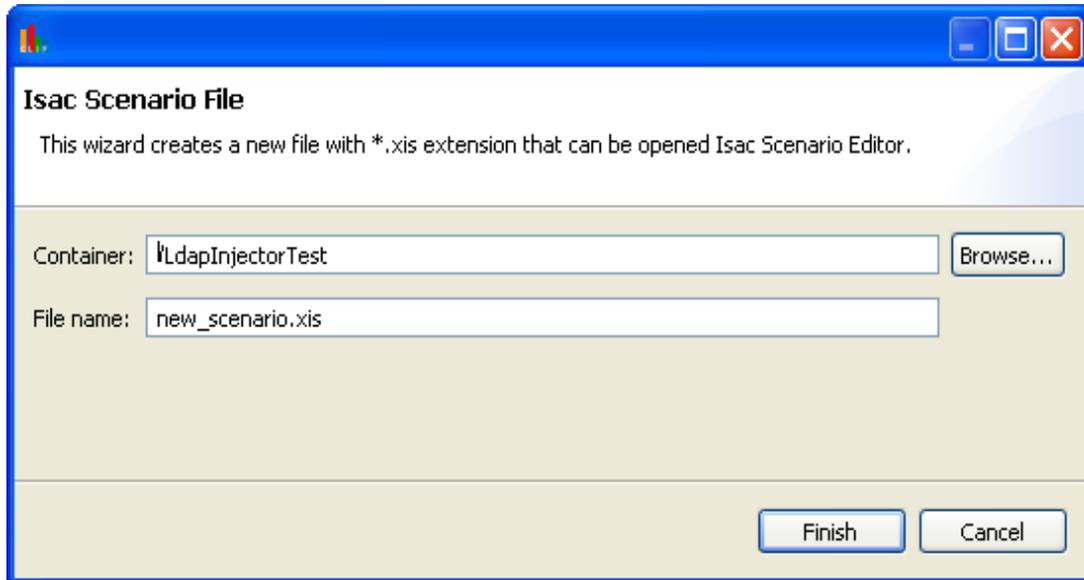
Create your new Isac scenario.

Click on File -> New -> New Isac Scenario

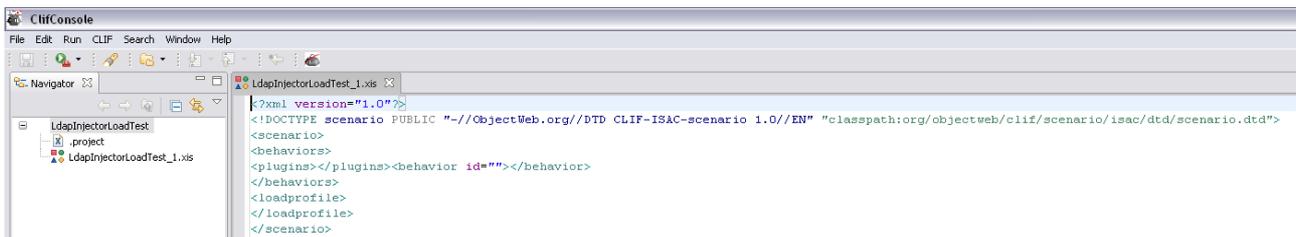


How To Use CLIF v2 and ISAC plug-ins

Choose the container (if you have several project into your workspace) and give a name to your ISAC scenario,



Click on Finish.



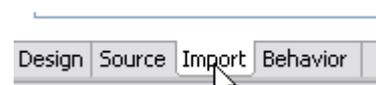
Now you can see the content of your ISAC scenario file.
At the bottom of the file content there are four tabulations



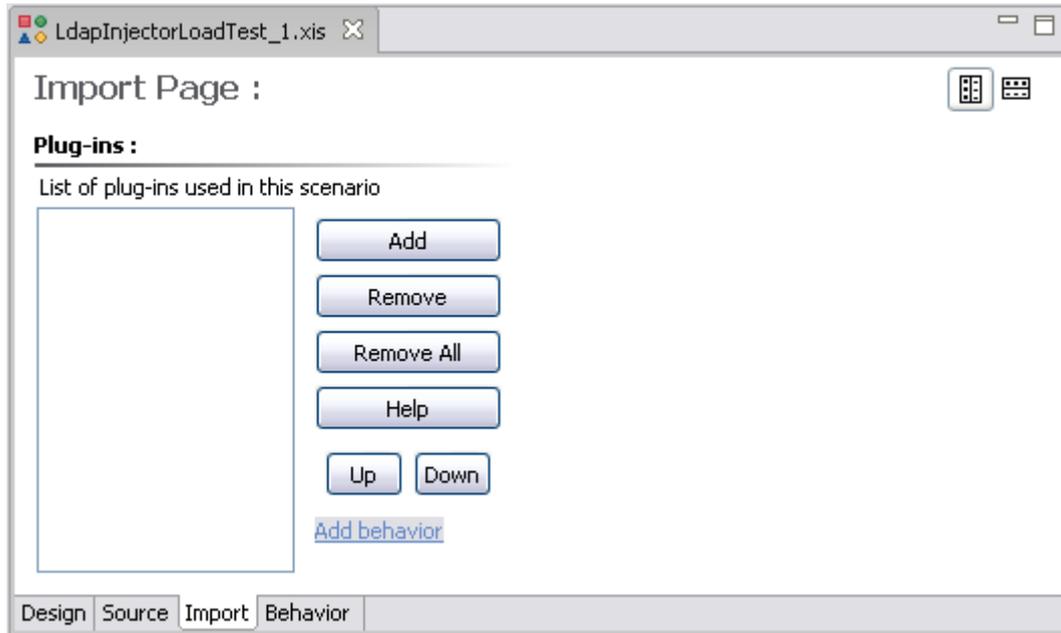
The first one designs the xis file (which is an xml file),
The second one shows the source code of the xis file.
The third one allows you to import the plugins that you will need in your ISAC scenario,
The last one allows you to define your scenario and your load profile.

2.2.2. Import ISAC plugins

Click on the Import tabulation to load the Import perspective.



Import the plugins:



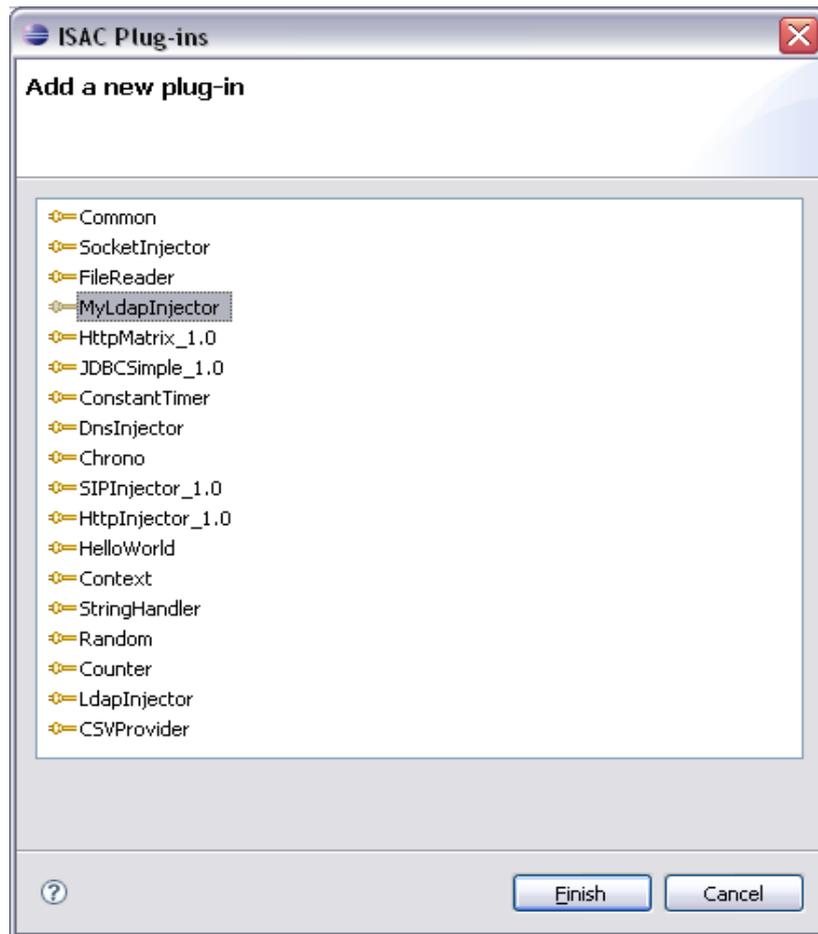
- LdapInjector
- CSVProvider
- ConstantTimer
- Random
- Chrono

Click on the Add button: 

Choose the plug-ins you need one by one:

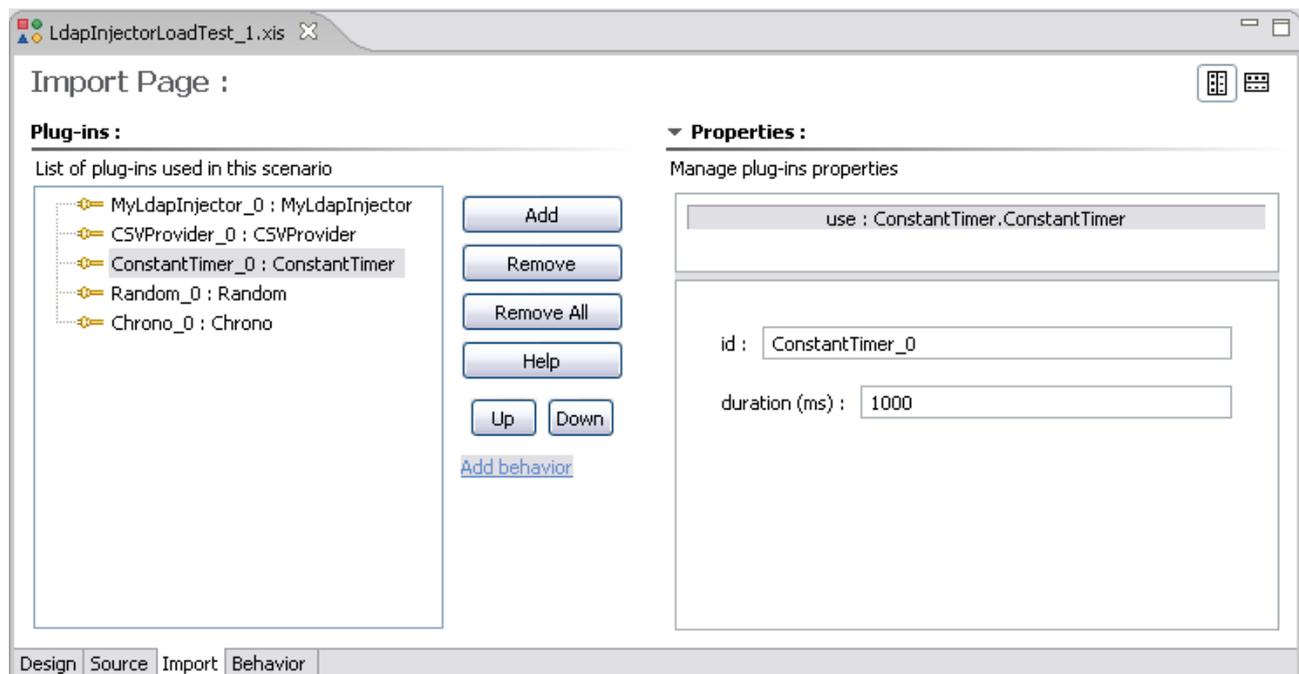
Now all the imported plug-ins are present in the Import perspective.

How To Use CLIF v2 and ISAC plug-ins



We just have to set plug-in default/initialization parameters.

Select the MyLdapInjector_0:MyLdapInjector and sets the parameters:



Do the same thing with the ConstantTimer plug-in.

The Random and Chrono plug-in have no general parameter.

Select the CSVProvider plugin and sets its parameters:

▼ **Properties :**
Manage plug-ins properties

use : CSVProvider.CSVProvider

id :

File name :

Field separator :

fields names (separated by the given separator) :

MacOS9 line separator
 enable

shared
 enable

loop
 enable

The file name is the name of the CSV file that will contain the externals data of the LdapInjector behavior.

A line in the CSV File will have this format:

```
login#password#searchBase#searchFilter#searchScope
uid=0,ou=appli,dc=annuaire,dc=com#secret_0#ou=appli,dc=annuaire,dc=com#(&(uid=0)(autorisation=all))#2
uid=1,ou=appli,dc=annuaire,dc=com#secret_1#ou=appli,dc=annuaire,dc=com#(&(uid=1)(autorisation=all))#2
uid=2,ou=appli,dc=annuaire,dc=com#secret_2#ou=appli,dc=annuaire,dc=com#(&(uid=2)(autorisation=all))#2
```

In the Field separator you have to set the field separator used in the CSV file. (# in our case),

In the Fields names input you have to set the list of the parameters name present in the CSV file separated with the given separator (# here),

Description of the check-box

- MacOS9 line separator: use CR instead of LF as line separator.
- shared: when set, progression in the lines is shared by all session objects. In other words, each session object will get a different line instead of all getting the same sequence of lines.
- loop: when set, the line sequence wraps up to the first line when the end of file has been reached. Otherwise, an alarm is thrown when trying to get a field value while the end of file

How To Use CLIF v2 and ISAC plug-ins

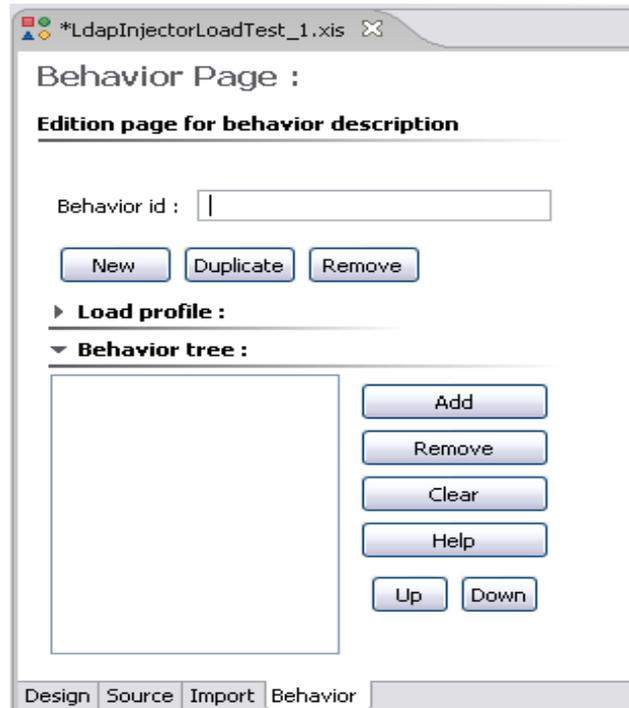
has been reached, and the empty string is used as value.

2.2.3. Define your behavior

To define your behavior you have to select the behavior perspective of your ISAC scenario xis file.



Once this tabulation selected you can see the following perspective:



You can now give a name to your behavior setting the Behavior id text field:



In our case we want to test an LDAP directory so we will describe a scenario that executes sequential actions.

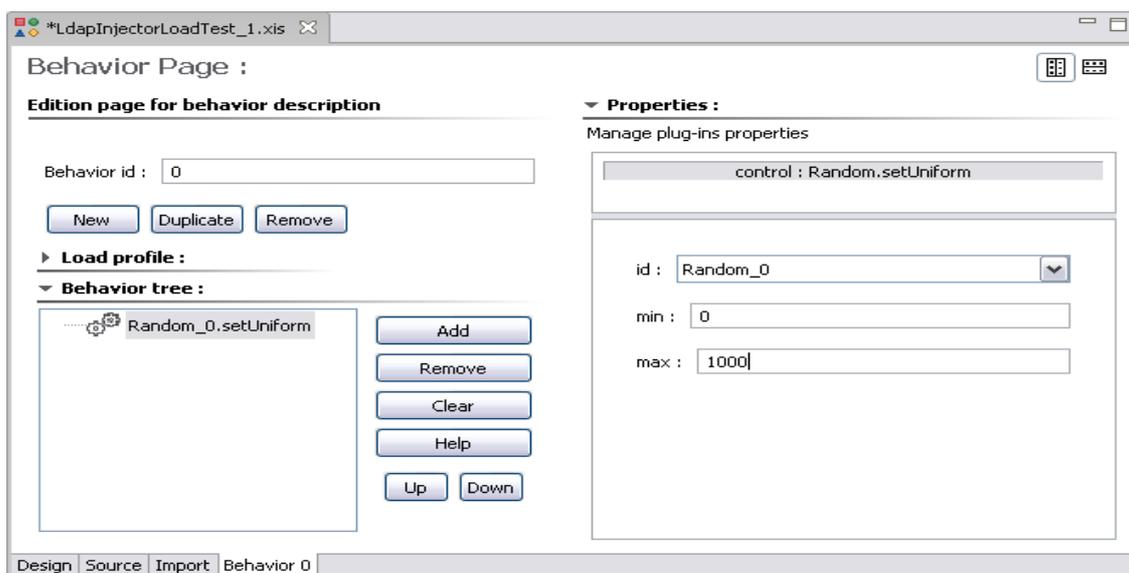
At the beginning of the behavior we add a sleep action. The sleep time will be variable and its variation will depend on a specific random distribution.

Click on the add button  in the behavior perspective.

First we choose the random distribution that we need to use.



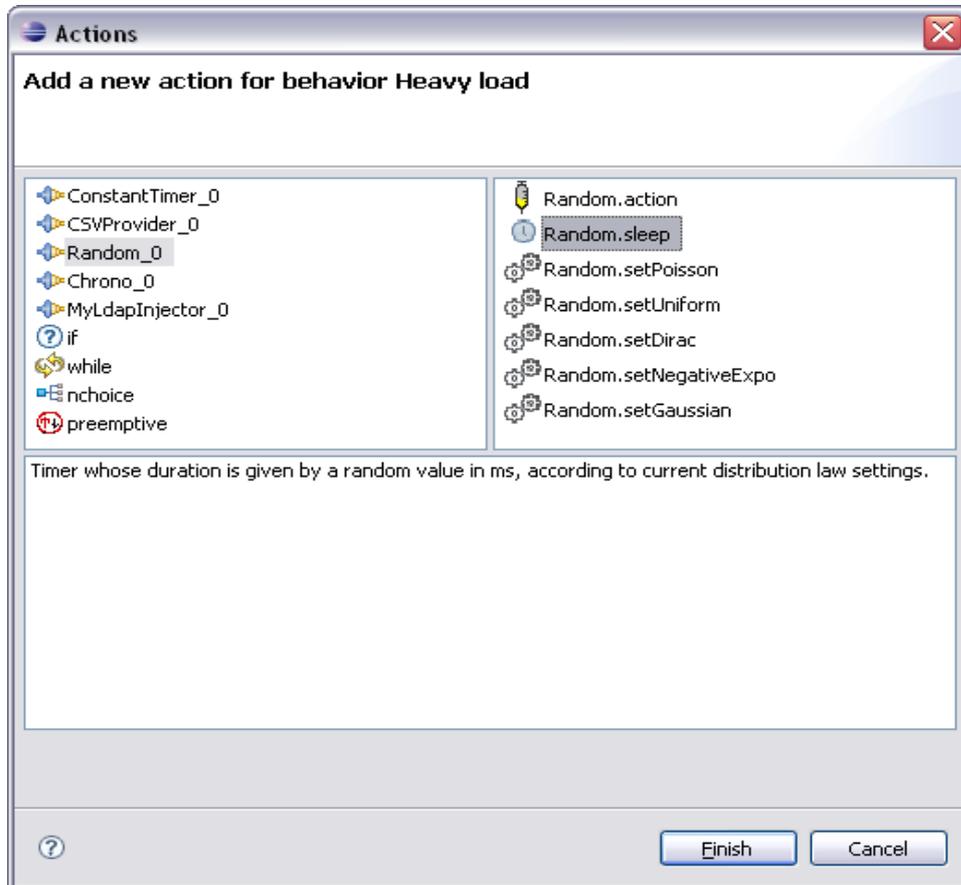
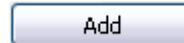
We sets the required parameter's value:



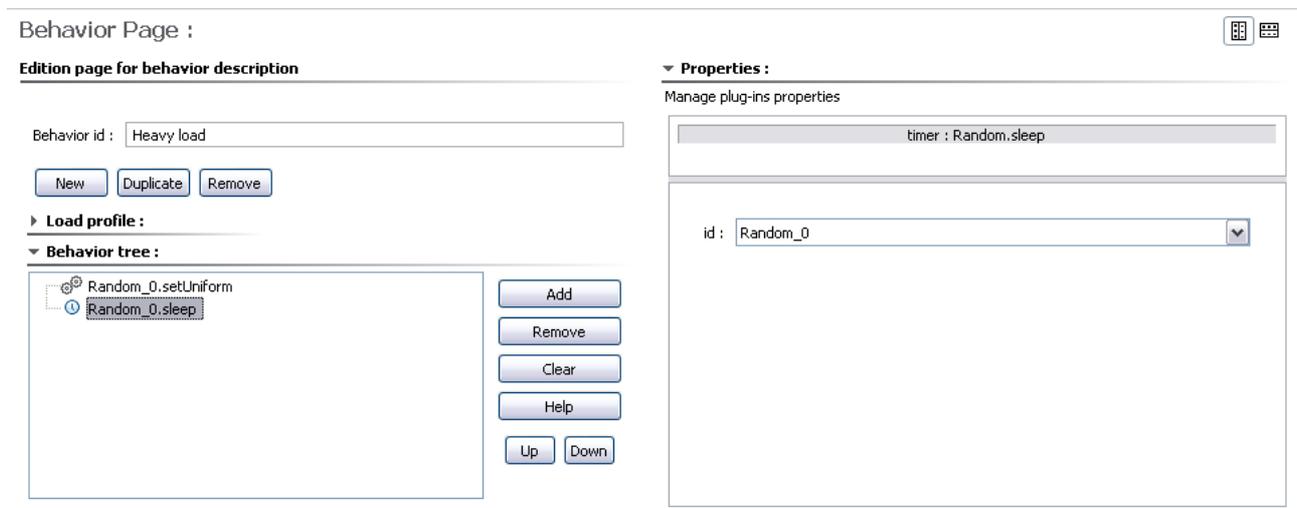
How To Use CLIF v2 and ISAC plug-ins

And now we add a sleep action that will sleep the behavior for a duration included between the min and the max value define in the random distribution.

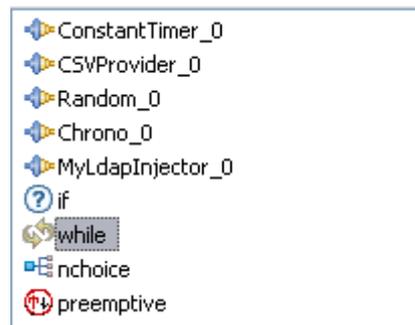
Click again on the add button in the behavior perspective.



And set the parameter's value of this action:

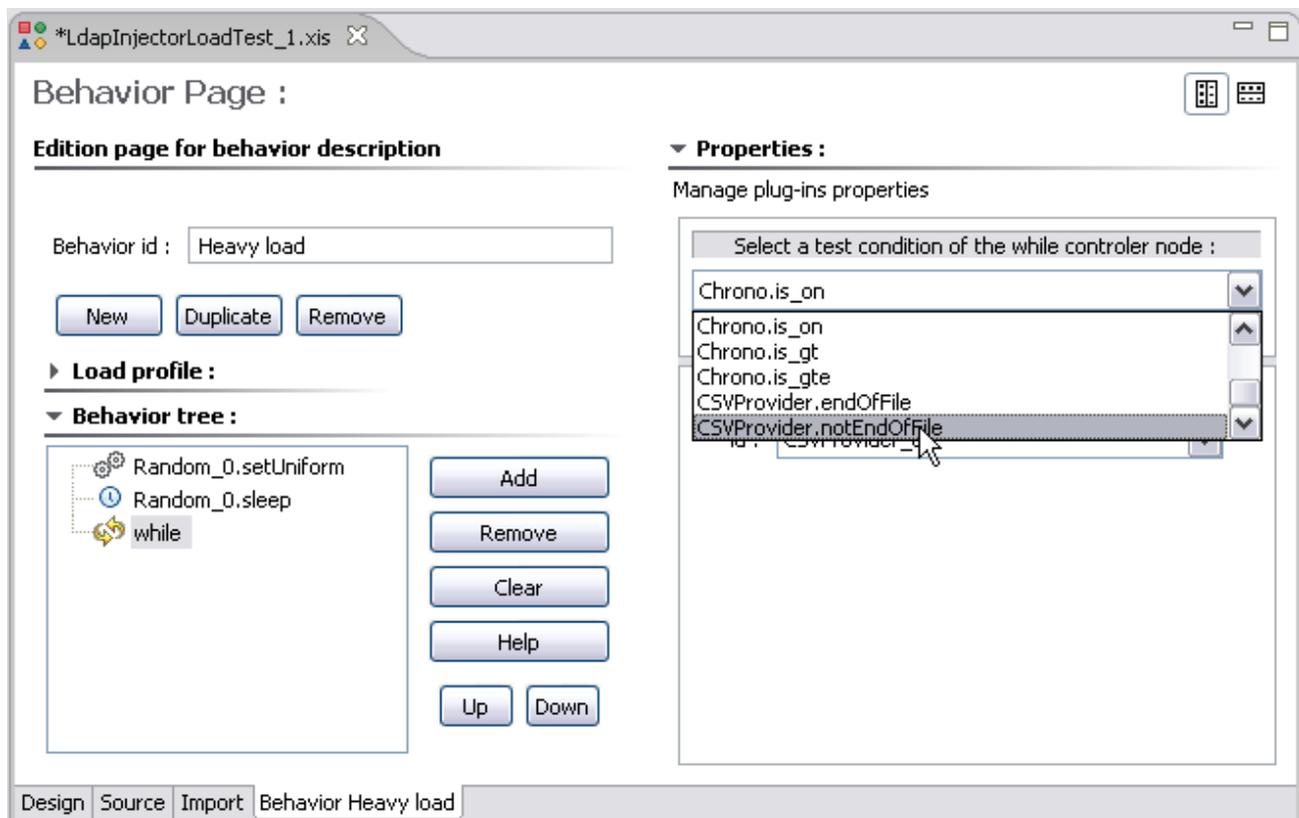


Now we add a while loop.



The while loop condition is: while we are not at the end of the CSV file we iterate.

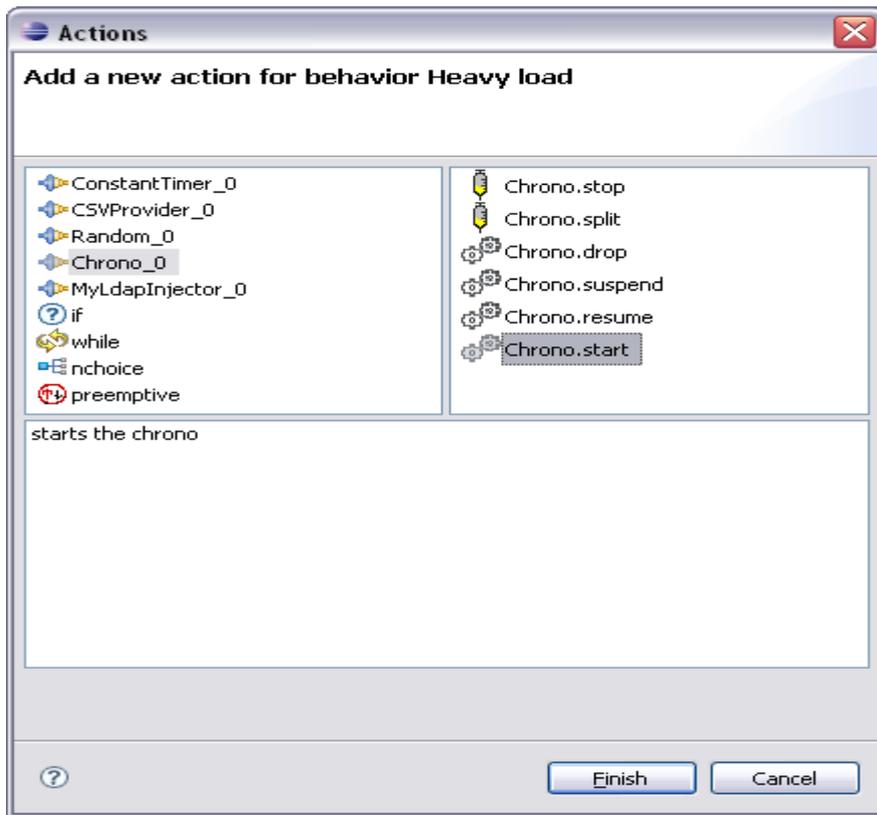
So we will iterate on each line of the CSV file and will exit from the loop when the end of the file will be reached.



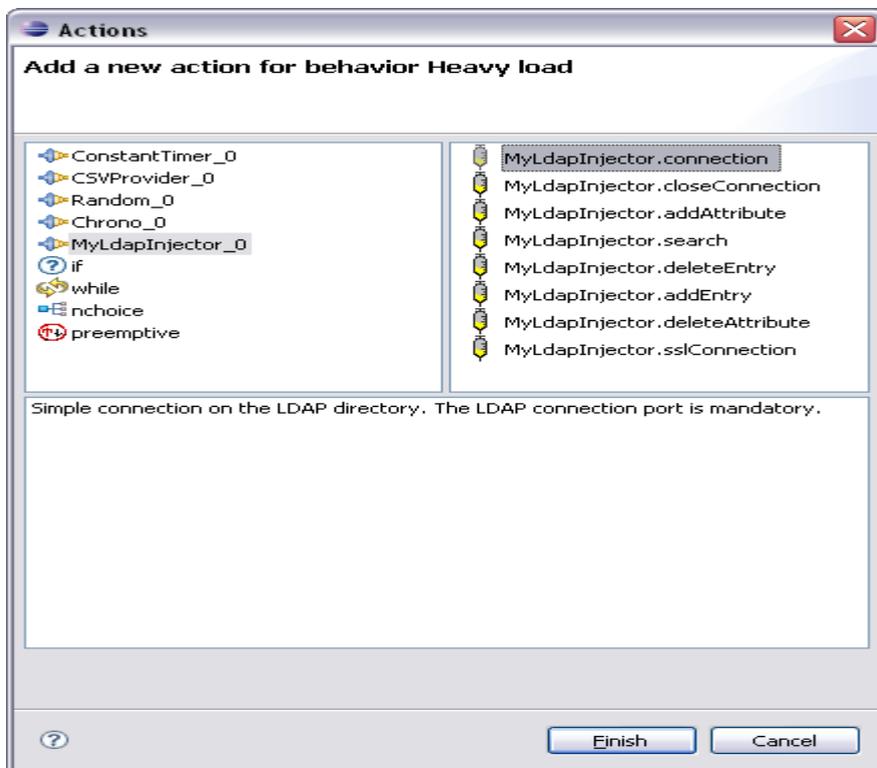
To include actions in your while loop you should select the while action in the Behavior tree before to click on the Add button .

The first action that we will add is starting the chronometer:

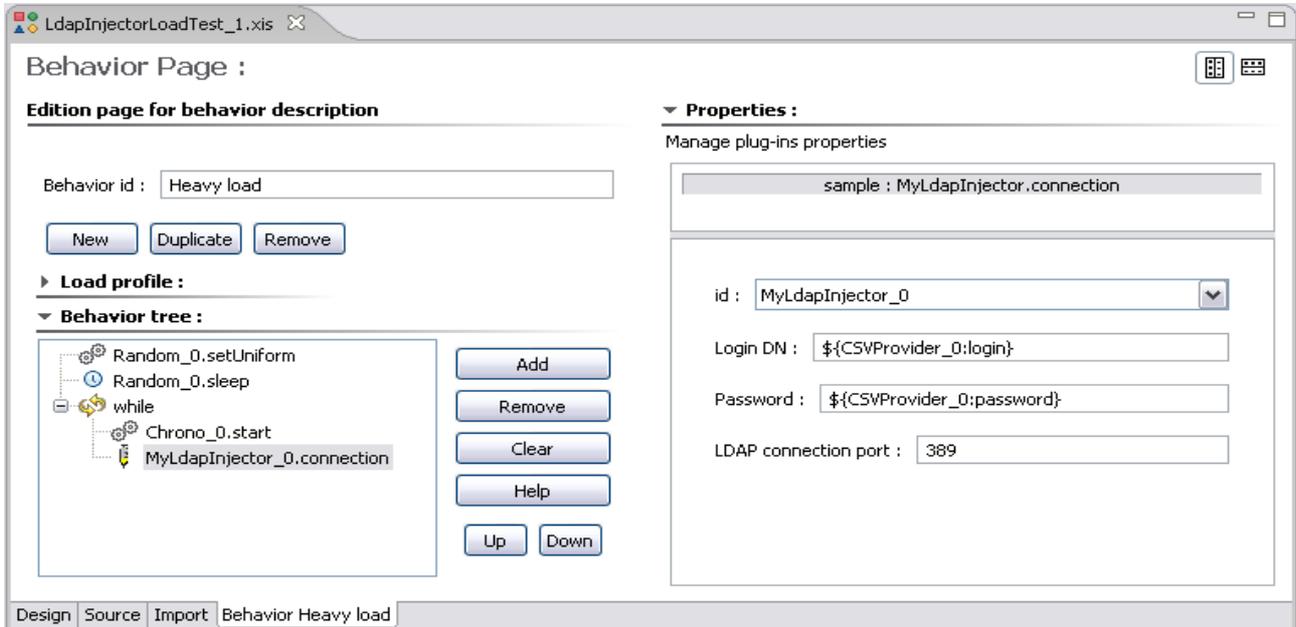
How To Use CLIF v2 and ISAC plug-ins



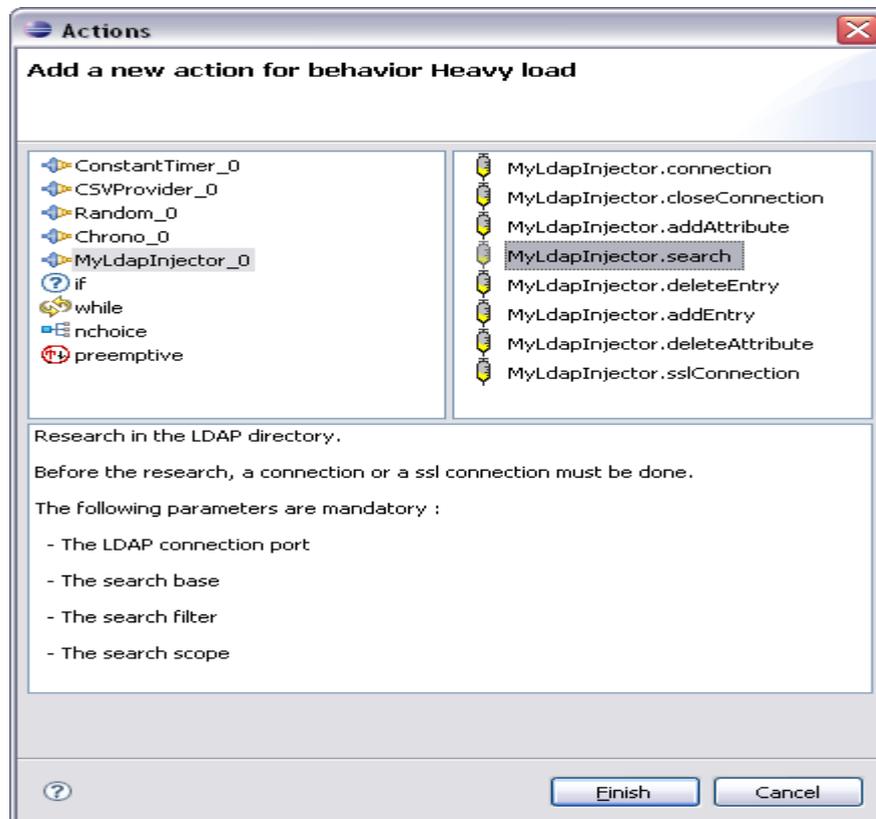
Then we will do a connection and disconnection on the LDAP directory.



At this time we will have to set the LDAP connection parameter's value. These values are provided by the CSV file. To get this value from the reading line of the CSV file we need to indicate the reference of the plug-in which provides the value and the name of the attribute needed. In our case the CSVProvider plug-in provides the login and the password. Use this specific String format: `${pluginIdentifier:key}`

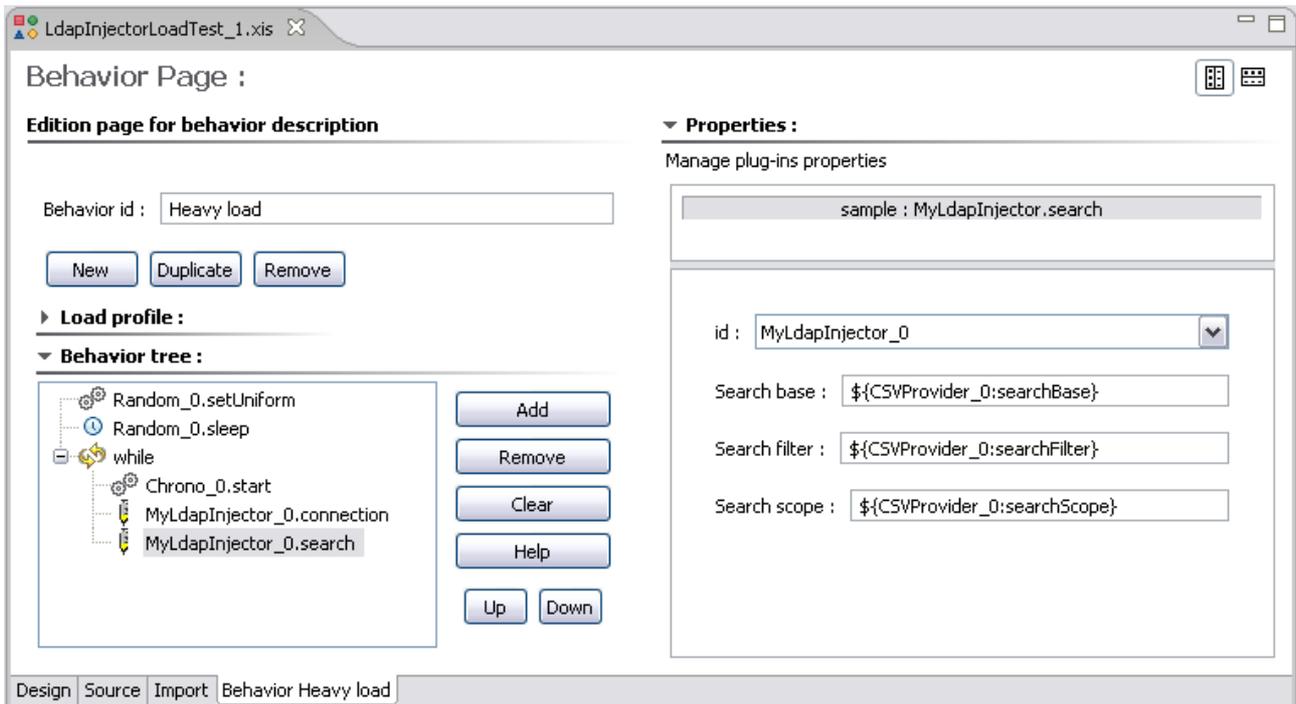


Then we add actions to make a search on the LDAP directory: we need to connect to it, make a search and disconnect.



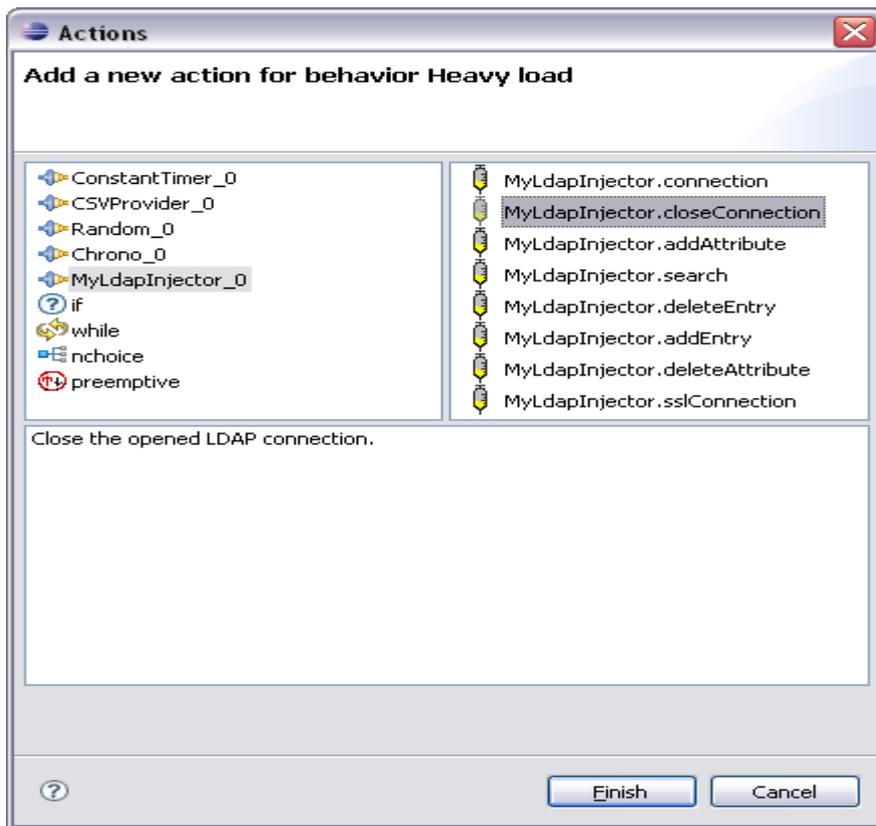
How To Use CLIF v2 and ISAC plug-ins

To set the connection and search action parameter's we use the same method as the first connection action.

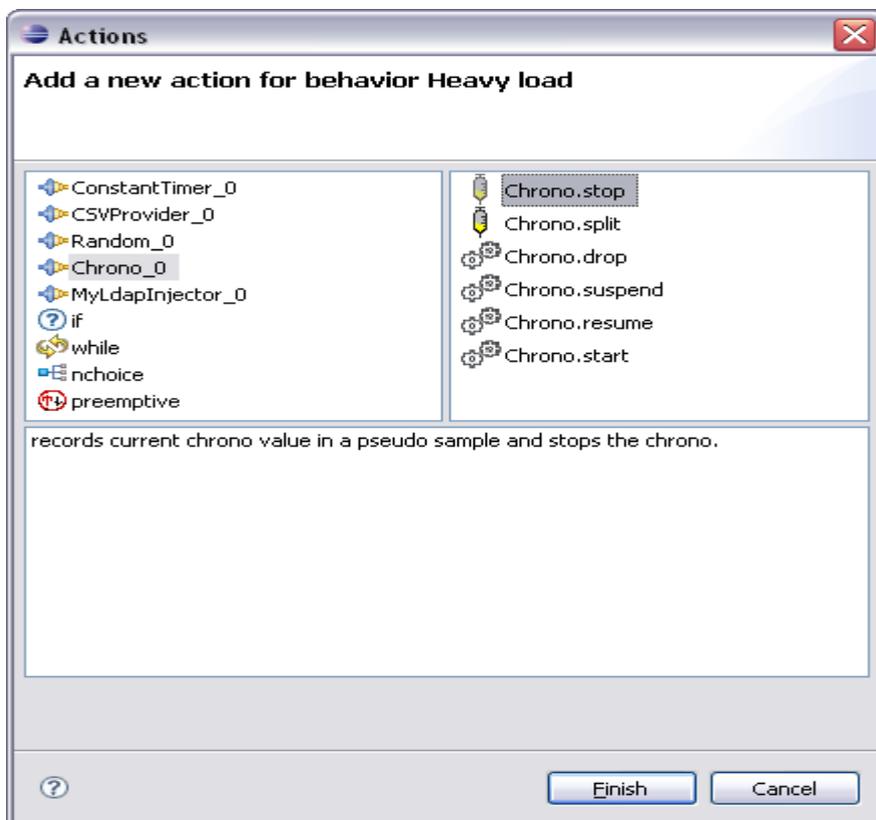


Once the research done we have to close the connection.

Click on the add button and choose the MyLdapInjector.closeConnection action.

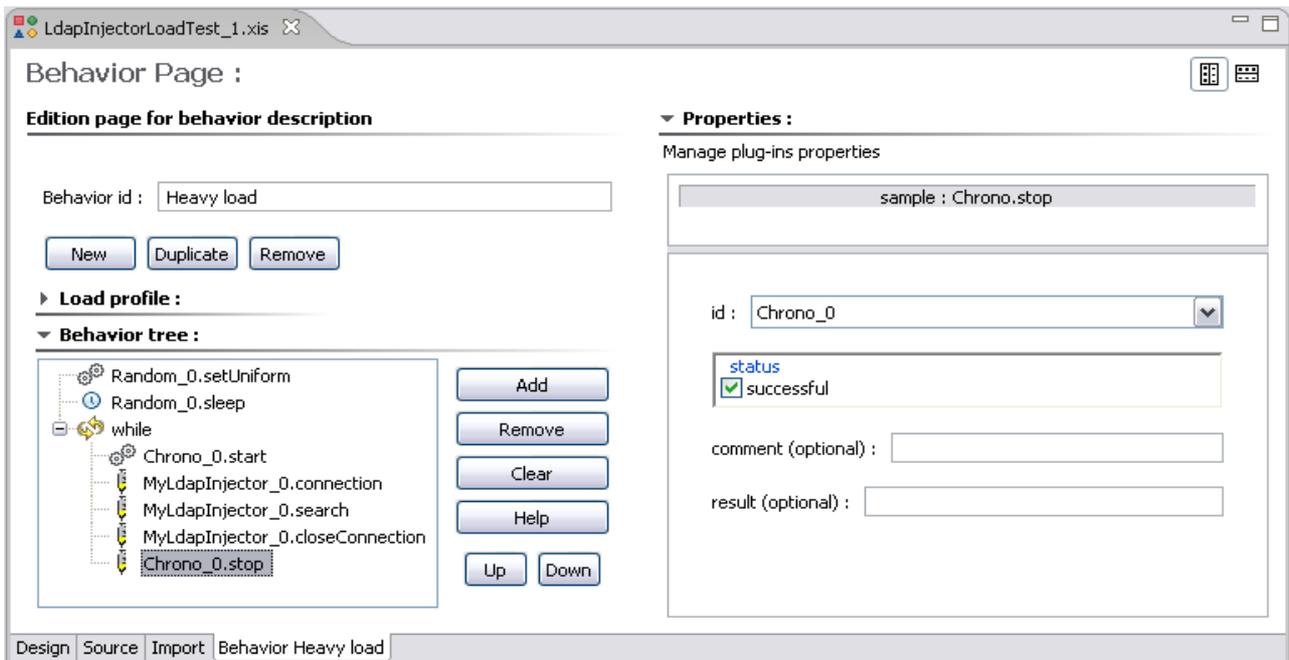


Now we have to stop the chrono to be able to get the total duration time of one loop with the

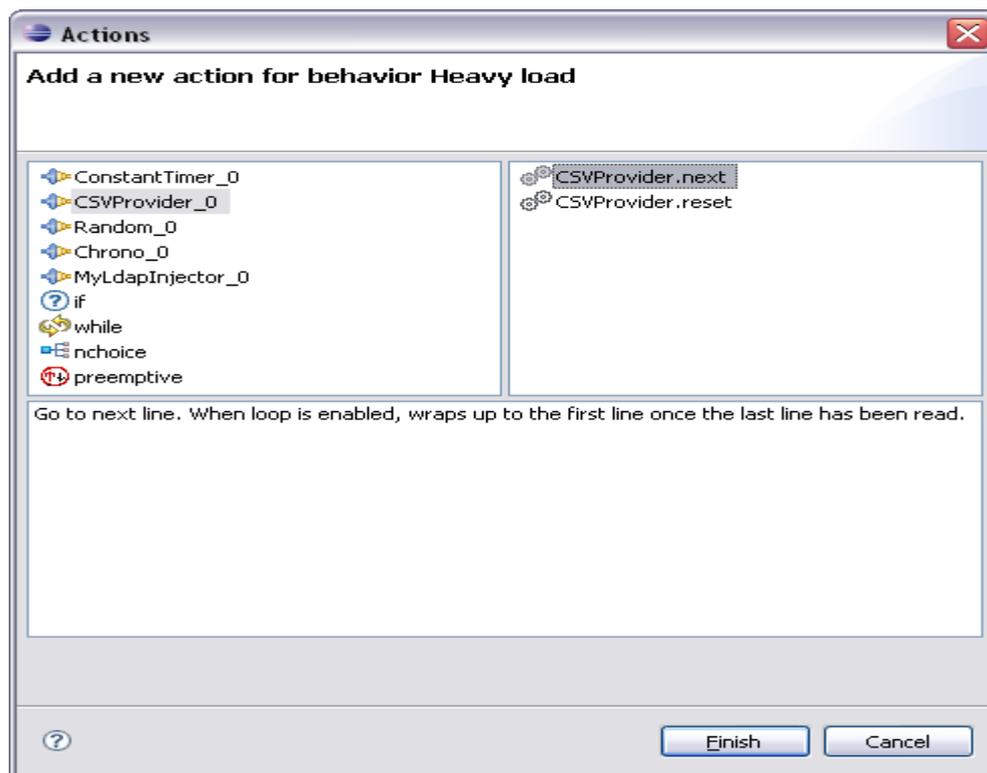


Chrono.stop action. Click on the add button and choose the Chrono_0.stop action.

How To Use CLIF v2 and ISAC plug-ins



Click on the add button to add a CSVProvider_0.next action.



It will get the following line of the CSV file. If there is no more line it will exit the loop otherwise it will restart at the beginning of the loop with new value provided by the new line got by the CSVProvider.

Behavior Page :

Edition page for behavior description

Behavior id :

▶ **Load profile :**

▼ **Behavior tree :**

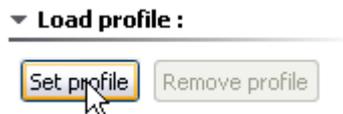
- ⊖ Random_0.setUniform
- ⊖ Random_0.sleep
- ⊖ while
 - ⊖ Chrono_0.start
 - ⊖ MyLdapInjector_0.connection
 - ⊖ MyLdapInjector_0.search
 - ⊖ MyLdapInjector_0.closeConnection
 - ⊖ Chrono_0.stop
 - ⊖ CSVProvider_0.next

Design Source Import Behavior Heavy load

2.2.4. Load Profiles

Load profiles enable predefining how the population of each behavior will evolve, by setting the number of active instances according to time. A load profile is a sequence of lines. For each load profile, a flag states if active instances shall be stopped to enforce a decrease of the population, or if the extra behaviors shall complete in a kind of a "lazy" approach.

To create a load profile you should be on the Behavior Page and click on the Load profile link:



and after it on the Set profile button:

Be careful, the time box contains the time in seconds since the start of the test and not the time in seconds of the ramp we are defining.

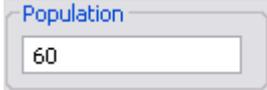
Now we want a ramp that increases the number of simultaneously active behavior instances. It begins at time 0 during 60 seconds and increases the number of active behavior instances from 0 to 60.

Enter 60 in the Time text field:



How To Use CLIF v2 and ISAC plug-ins

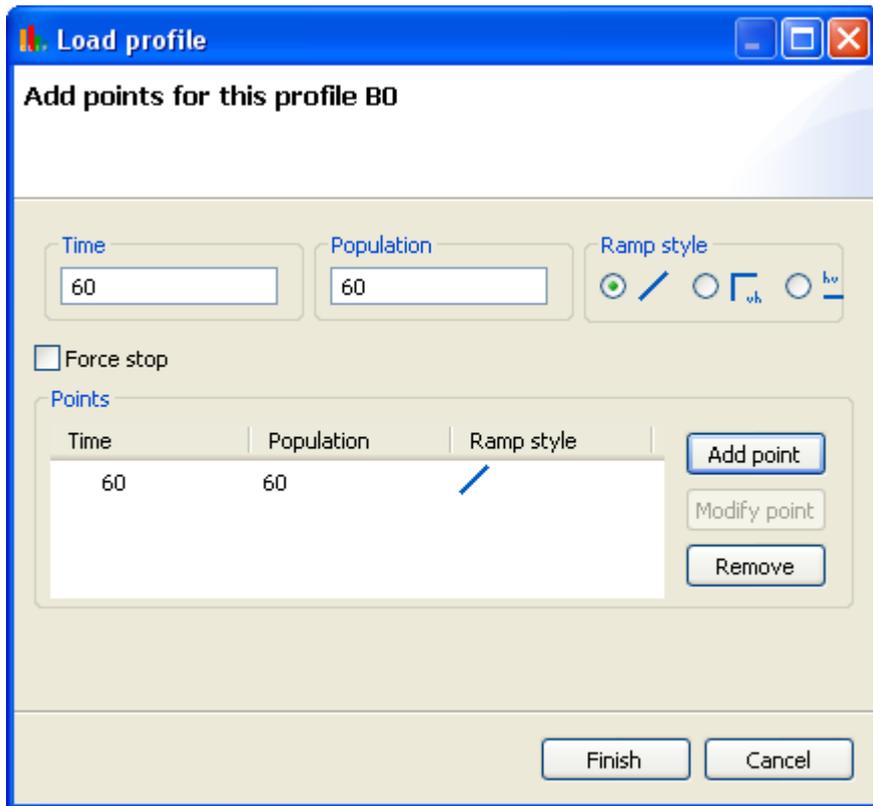
Enter 60 in the Population text field:



Choose the ramp style:



To save these settings click on the Add point button:



The screenshot shows a dialog box titled "Load profile" with a subtitle "Add points for this profile B0". It contains several input fields and a table. The "Time" field is set to 60, and the "Population" field is set to 60. The "Ramp style" is set to a diagonal line. There is a checkbox for "Force stop" which is unchecked. Below these is a table with columns "Time", "Population", and "Ramp style". The table contains one row with values 60, 60, and a diagonal line. To the right of the table are buttons for "Add point", "Modify point", and "Remove". At the bottom of the dialog are "Finish" and "Cancel" buttons.

Time	Population	Ramp style
60	60	/

After we want a stable number of simultaneously active behavior instances (aka virtual users) during 2 minutes (120 seconds); so the profile duration will be about 180 seconds. And at the end of the 2 minutes, the number of virtual users for this behavior will rise to 100.

Now we want a ramp that increases the number of virtual users. It begins at time 0 during 60 seconds and increase the number of virtual users from 0 to 60.

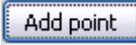
Enter 60 in the Time text field:

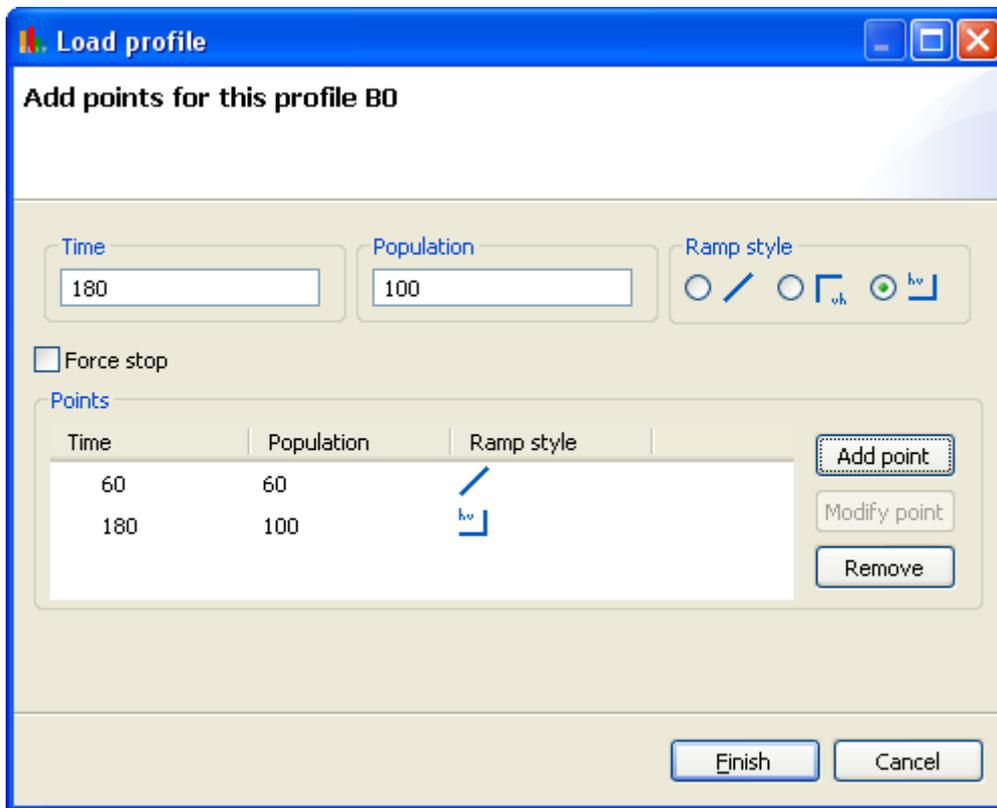


Enter 60 in the Population text field:

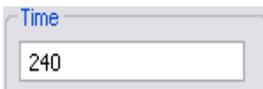


Choose the ramp style: 

To save these settings click on the Add point button: 

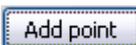


And finally we want to decrease the number of virtual users from 100 to 0 during 60 seconds.

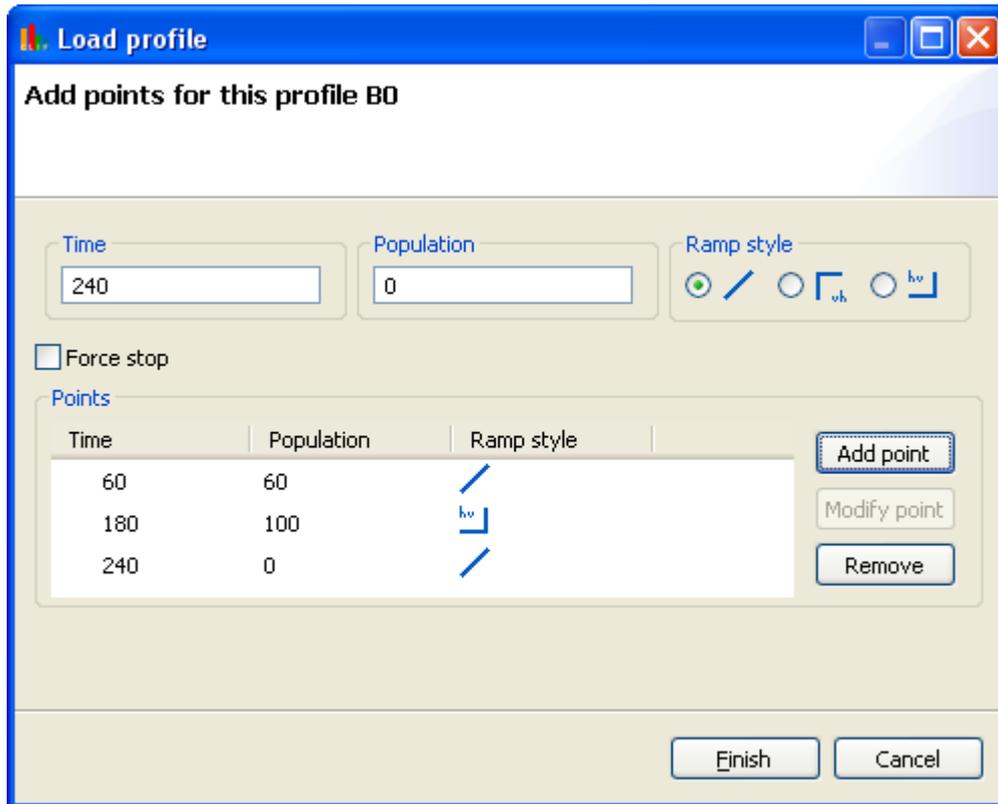
Enter 60 in the Time text field: 

Enter 60 in the Population text field: 

Choose the ramp style: 

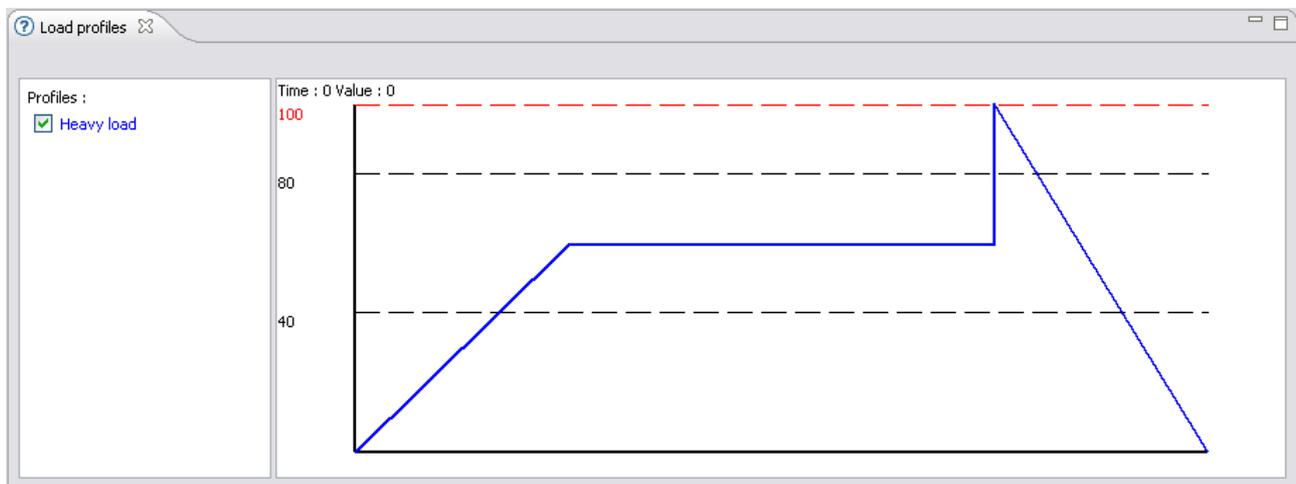
To save these settings click on the Add point button: 

How To Use CLIF v2 and ISAC plug-ins



Click on the Finish button: 

You can see the load profile form watching the “Load profiles” view:



3. 4. Define your test plan

3.1. Description of the context

In our case we want to test an LDAP directory. To test it and to get the results during the loading test we will deploy injectors and probes.

We want to test the LDAP directory sending requests from two injectors deployed on two different servers. To be able to see the server's CPU usage rate we will deploy on each of them a CPU probe.

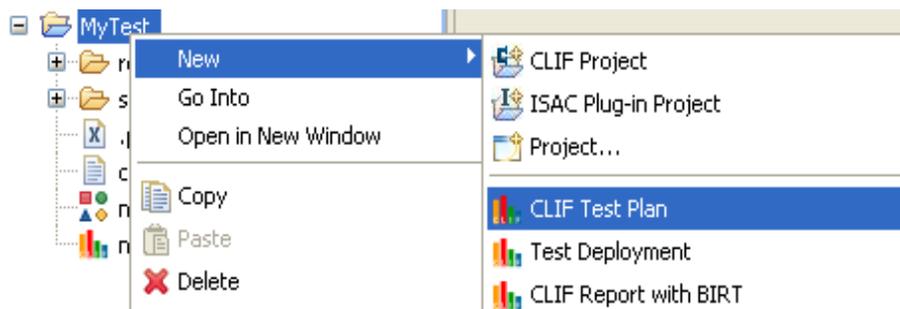
To be able to get necessary information to generate a load test report for our LDAP directory we will deploy a CPU probe and a JVM probe on the LDAP server.

Before to start we have to introduce the term of « blade ». A blade is an active component that can be deployed within a CLIF application, under control of the supervisor component, that provides statistical information about its execution (for monitoring purpose), and produce results stored by the storage component. Blades exist either as load injectors or probes.

3.2. Creating your test plan

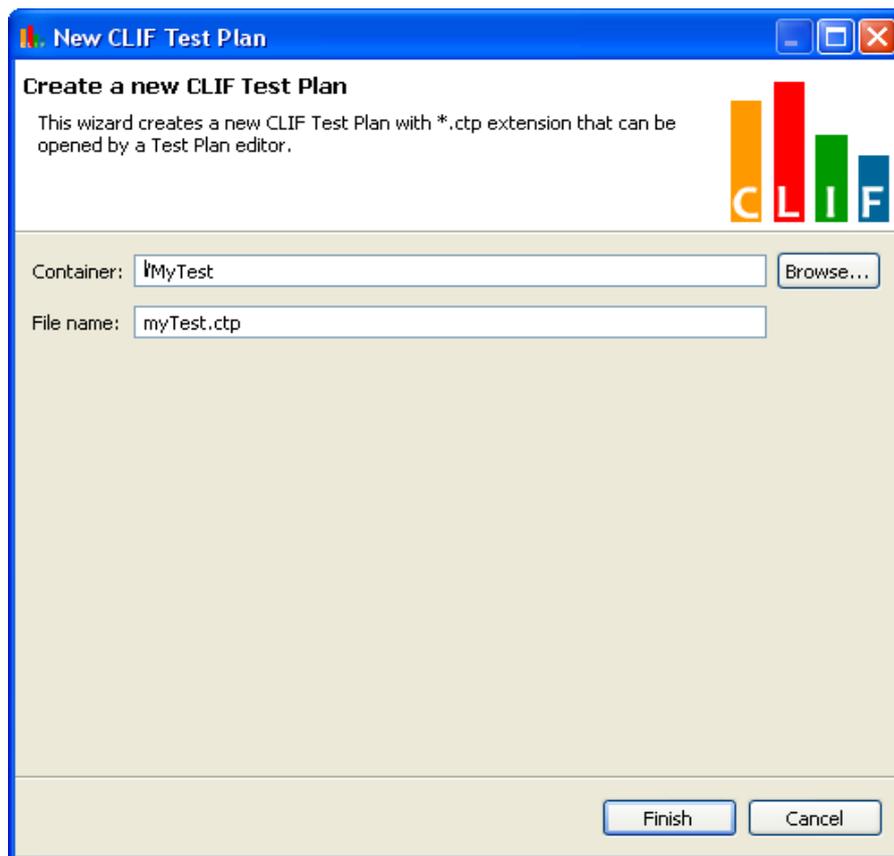
This test plan will be created at the same location of the ISAC scenario file.

Select your LdapInjectorLoadTest project. Right click on it and choose Clif testPlan



Then enter your test plan name and click on the Finish button:

How To Use CLIF v2 and ISAC plug-ins



3.3. Add Probes and Injectors to your test plan

In our case we will add:

- Two injectors, one on each server used to inject requests on the LDAP directory.
- Three CPU probes, one on each injector server and one on the server where the LDAP directory is installed.
- One memory probe, on the server where the LDAP directory is installed.

Click on the add button

Then you have to set the properties of the component you want to add:

▼ Properties

Manage injector and probe properties

Id* :	<input type="text" value="0"/>
Server* :	<input type="text" value="local host"/> <input type="button" value="Refresh"/>
Role* :	<input type="text" value="probe"/> <input type="button" value="Refresh"/>
Class* :	<input type="text"/>
Arguments :	<input type="text"/>
Comment :	<input type="text"/>

As we want to add three probes we have to set the Server name or the IP address of the server where the probes will be deployed.

To be able to set the Server name or the IP address you have to start your CLIF registry and launch your CLIF server (refer to Installation Manual).

Otherwise you will not be able to change the server name which is local host by default.

For example:

```
Id: 0
Server: 10.0.0.1
Role: probe
Class: cpu
Arguments: 1000 600 (a measure will be taken each 1000 ms and during 600 s)
Cpu probe deployed on 10.0.0.1
```

Click on the add button:

```
Id: 1
Server: 10.0.0.2
Role: probe
Class: cpu
Arguments: 1000 600 (a measure will be taken each 1000 ms and during 600 s)
Cpu probe deployed on 10.0.0.2
```

Click on the add button:

```
Id: 2
Server: 10.0.0.3
Role: probe
Class: cpu
Arguments: 1000 600 (a measure will be taken each 1000 ms and during 600 s)
Cpu probe deployed on 10.0.0.3
```

Don't forget to save your test plan clicking on File -> Save (or Ctrl+s)

Now we have this view:

Test Plan Editor

Injectors and probes

All injectors and probes in the test plan

Id	Server	Role	Class	Arguments	Comment
2	10.0.0.3	probe	cpu	1000 600	Cpu probe deployed on 10.0.0.3
1	10.0.0.2	probe	cpu	1000 600	Cpu probe deployed on 10.0.0.2
0	10.0.0.1	probe	cpu	1000 600	Cpu probe deployed on 10.0.0.1

Add

Remove

Remove All

We do the same thing with the jvm probe

```
Id: 3
Server: 10.0.0.2
Role: probe
```

How To Use CLIF v2 and ISAC plug-ins

Class: jvm
Arguments: 1000 600 (a measure will be taken each 1000 ms and during 600 s)
Memory probe deployed on 10.0.0.2

Now we have this view:

The screenshot shows the 'Test Plan Editor' window with the 'Injectors and probes' section. It displays a table of all injectors and probes in the test plan. The table has columns for Id, Server, Role, Class, Arguments, and Comment. A single row is visible with Id 3, Server 10.0.0.2, Role probe, Class jvm, Arguments 1000 600, and Comment Jvm probe deployed on 10.0.0.2. To the right of the table are three buttons: Add, Remove, and Remove All.

Id	Server	Role	Class	Arguments	Comment
3	10.0.0.2	probe	jvm	1000 600	Jvm probe deployed on 10.0.0.2

And finally with the injectors:

Id: 4
Server: 10.0.0.1
Role: injector
Class: IsacRunner
Arguments: LdapInjectorLoadTest_1.xis
Injector deployed on 10.0.0.1

Id: 5
Server: 10.0.0.3
Role: injector
Class: IsacRunner
Arguments: LdapInjectorLoadTest_1.xis
Injector deployed on 10.0.0.3

Now we have this view:

The screenshot shows the 'Test Plan Editor' window with the 'Injectors and probes' section. It displays a table of all injectors and probes in the test plan. The table has columns for Id, Server, Role, Class, Arguments, and Comment. Two rows are visible: Id 5, Server 10.0.0.3, Role injector, Class IsacRunner, Arguments LdapInjectorLoadTest_1.xis, Comment Injector deployed on 10.0.0.3; and Id 4, Server 10.0.0.1, Role injector, Class IsacRunner, Arguments LdapInjectorLoadTest_1.xis, Comment Injector deployed on 10.0.0.1. To the right of the table are three buttons: Add, Remove, and Remove All.

Id	Server	Role	Class	Arguments	Comment
5	10.0.0.3	injector	IsacRunner	LdapInjectorLoadTest_1.xis	Injector deployed on 10.0.0.3
4	10.0.0.1	injector	IsacRunner	LdapInjectorLoadTest_1.xis	Injector deployed on 10.0.0.1

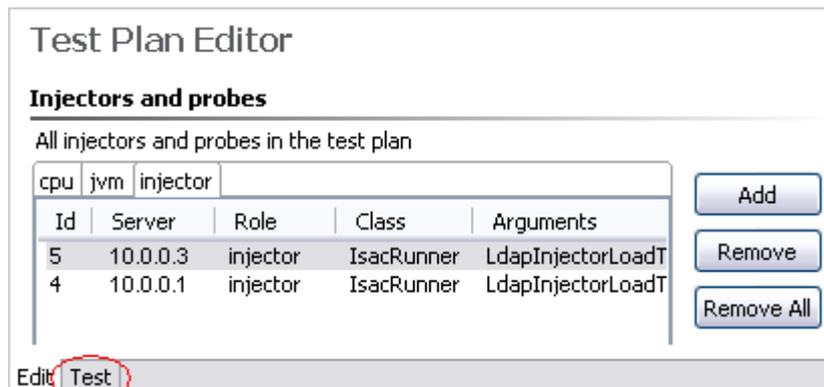
3.4. Deploying and executing your test plan

Your code server path should include the directory where your scenario file is, in order to benefit from the automatic remote loading of the scenario file by every remote ISAC execution engine you may have defined in your test plan.

In our case, to be able to test the Ldap directory we also need to include the directory where your csv file is in order to access to the connection and research parameters. (refer to chapter 5.3 Running a registry).

Now you can open your test plan. A new tabulation named "test" appears.

Click on it.

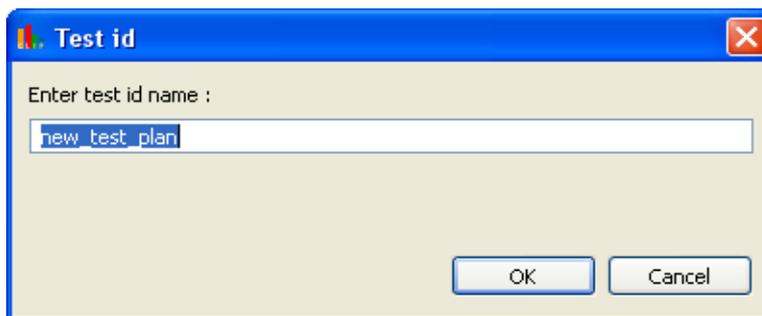


On this tabulation you can see the status of all your blades:

And the global status of your test plan: Global state: **undeployed**

Click on the deploy button

Once the deployment of all your blades done (global state = deployed) you can initialize it (button:) and choose the test plan you want to initialize:



Once the initialization done you can see it on the monitor tabulation:

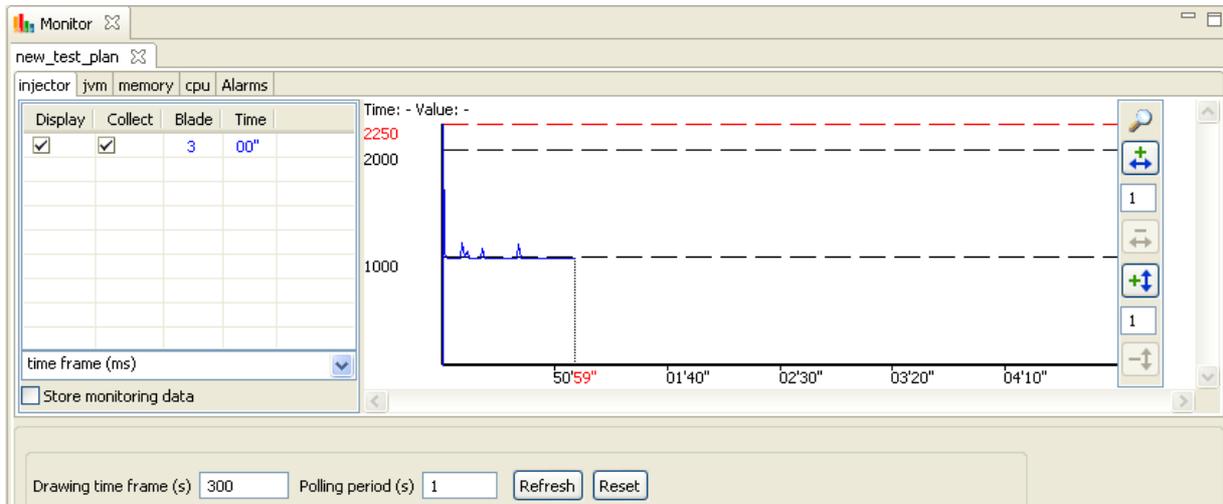


As soon as the test plan is deployed and initialized, the monitoring area pops up in the test plan window's bottom part. This area holds a set of tabbed panels:

- one for all injectors

How To Use CLIF v2 and ISAC plug-ins

- one for each probe family



For each panel, the user may set the monitoring time-frame, the polling period, and start or stop the monitoring process. Moreover, a check-box table at the left side of each panel makes it possible to selectively disable or enable the collect and display of monitoring data, for each blade.

Now you can start the injection by clicking on the start button:

And then you can suspend , stop or change dynamically some parameters of the load test.

Several parameters of the execution engine may be modified, including at runtime, using the Parameters button:

- about the engine itself (size of the thread pool, polling period for load profile management, tolerance on deadlines);
- about the active scenario, in particular the number of active instances (population) of each behavior.

Once the load test completed you have access to the collect action. Collecting CLIF data will locally gather CSV-formatted files containing all the measures, in a tree-organized breakdown of directories and files. The root directory is determined by a CLIF property. The default is a directory named "report" in the corresponding CLIF project.

4. CLIF server

4.1. Installation

See Install Manual

4.2. Rationale

CLIF servers are necessary to deploy any test plan, since they host load injectors and probes. CLIF servers are designated by a name, which is registered in a Registry. In order to run, CLIF servers must be able to find this Registry, which implies that:

1. the Registry must be running before a CLIF server can be launched;
2. parameters must be given to tell the CLIF servers where to find the Registry and register themselves.

4.3. Running a registry

The CLIF registry is launched when necessary by the Java Swing GUI or the eclipse-based GUI. If you want to launch it from command-line, see the User Manual.

There are three ways of starting a registry:

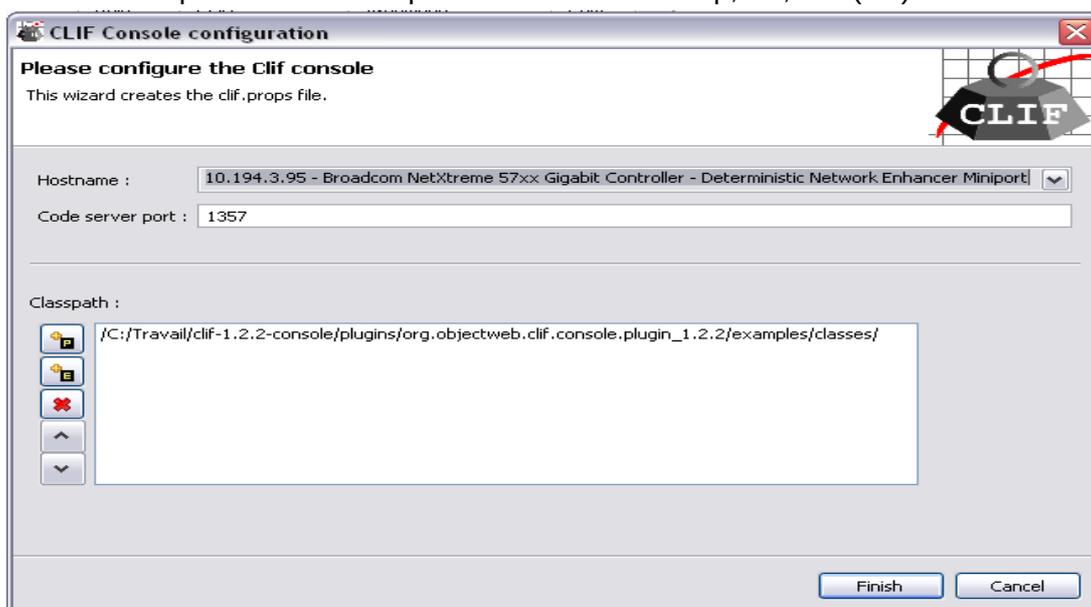
1. running the Java Swing console GUI
2. using the Eclipse-based console GUI
3. using the appropriate command

In our case (test an LDAP directory) we will use the Eclipse-based console GUI:

Click on the start registry button:  in the task button bar

Then you can set:

- the host name which identifies the server where the registry will be launched,
- the code server port
- and the classpath which is the path to access to the ctp, xis, csv (etc) files.



How To Use CLIF v2 and ISAC plug-ins

You can add or delete paths using these buttons:

-  to add projects path
-  to add externals path
-  to delete the selected path

In our case we will delete the default path and add our project path.

Select the default Classpath and click on this button: 

Then on the add projects path button  and choose your project path:

Finally click on Finish to start the registry.

4.4. Configuring a CLIF server

You may configure CLIF either by editing file `clif.props` in the `etc/` subdirectory, or by using command "ant config". In the latter case, the following questions will be asked:

- *please enter the host where the console will be run:*
enter the IP address or name of the computer where you will run the Registry, either embedded in the Swing or Eclipse GUI, or launched by command line.
- *please enter the port number for the console embedded code server:*
enter the port number used by the code server, for example 1357.

This configuration operation must be done everywhere you want to run a CLIF server or a console. You may also make this configuration step only once, and copy the resulting file `etc/clif.props` wherever needed.

In our case we will deploy a CLIF server on:

- Server 1: 10.0.0.1
- Server 2: 10.0.0.2
- Server 3: 10.0.0.3

The console will be run on 10.0.0.10 and the code server will be the default one 1357.

4.5. Running a CLIF server

CLIF must be configured on each host you plan to run a CLIF server, accordingly to where your Registry is running. Your Registry must be running to be able to launch Clif server. (cf chapter 4.3) Then, run a CLIF server with command:

- **ant server** to create a CLIF server that registers with the local host name as CLIF server name
- **ant -Dserver.name=myFirstServer server** to create a CLIF server that registers with the provided name