December, 5th 2007

HOW TO

Develop and use CLIF ISAC PLUGINS



with the ECLIPSE RCP console

http://clif.objectweb.org/

Copyright © 2006-2007 France Telecom License information : <u>http://creativecommons.org/licenses/by-nc-sa/3.0</u>

Table of contents

1. Introduction to ISAC Plug-ins	3
1.1. What is an ISAC Plug-ins	3
1.2. Technical requirements	3
1.3. Ready to use distributions	4
1.4. Installation	4
2. How to Define an LDAP injector writing an ISAC plug-ins	8
2.1. LDAP directory overview	8
2.2. How to Write a LDAP Injector ISAC Plug-in	9
2.2.1. How to Create an Isac Plug-in project	
2.2.2. Writing your ISAC plug-ins	15
3. ISAC is a Scenario Architecture for CLIF	
3.1. Defining an ISAC scenario for LDAP	
3.2. Record your ISAC scenario	
4. 4. Define your test plan	
4.1. Description of the context	58
4.2. Creating your test plan	58
4.3. Add Probes and Injectors to your test plan	59
4.4. Deploying and executing your test plan	62
5. CLIF server	64
5.1. Requirements	64
5.2. Rationale	64
5.3. Running a registry	64
5.4. Configuring a CLIF server	66
5.5. Running a CLIF server	67
6. Appendix A : Install mandatory requirements	68
6.1. Download requirements	68
6.1.1. Sun J2SDK TM 1.5 (1.5 version or greater is mandatory)	68
6.1.2. Apache ant utility version 1.5.4 or greater	69
6.2. Environement variables setting	69
6.2.1. Windows OS	
6.2.2. Linux OS	73
6.2.3. Mac OS X	

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. Technical requirements

The CLIF framework and provided load injectors are 100% Java[™]. The current version 1.2.2 is known to be working with:

- Sun J2SDK[™] 1.5 (1.5 version or greater is mandatory)
- Apache ant utility version 1.5.4 or greater
- · Linux 2.4 and 2.6 kernels
- MacOS X Tiger
- Microsoft Windows XP[™]

System probes for Linux are also 100% Java, while system probes for Windows and MacOS X are native (C code embedded in Java code via Java Native Interface).

Since CLIF is written in Java, the only constraint about the SUT is that it must be reachable from a Java Virtual Machine (JVM), either directly or indirectly through some wrapping, gateway or native library.

There are two ways of getting a CLIF runtime environment: either by getting the whole source from the CVS repository, or by getting a ready-to-use binary distribution.

1.3. Ready to use distributions

CLIF's site at OW2 Forge offers several binary distributions, available as zip files:

- console-Linux
 Eclipse-RCP based standalone console for Linux/Intel.
- console-Windows
 Eclipse-RCP based standalone console for Windows/Intel.
- console-Macosx
 Eclipse-RCP based standalone console for Windows/Intel.
- clif-plugin
 CLIF console as an Eclipse plug-in.
- isac-plugin
 ISAC editor as an Eclipse plug-in (requires CLIF console plug-in).

1.4. Installation

There are two ways to install the isac plug-ins development environment. The first one is to use your Eclipse environment and the second one is to use the Eclipse-RCP based standalone console.

To use your Eclipse environment you have to unzip the clif-plugin and isac-plugin archives into your Eclipse plug-ins directory.



🖙 plugins	
Fichier Edition Affichage Favoris Outils ?	
🔆 Précédente 🔹 🕥 🔹 🏂 🔎 Rechercher 🞼 Dossiers 🕼 🎲 ≽	(19
Adresse 🛅 C:\appft\eclipse\plugins	
Dossiers	×
Bureau Bureau Bureau G-DELL610-57 Jisque local (C:) appft apache-ant-1.5.2 apache-ant-1.7.0 ConTEXT ConTEXT configuration features plugins features org.apache.ant_1.6.5 org.apache.cactus_1.7.2.v200606181221 org.apache.cactus_1.7.2.v200606181221 org.apache.cactus_1.7.2.v200606131651 org.apache.jakarta_jog4j_1.2.8.v200606131651 org.apache.lucene_1.4.103.v20060601	

You also have to check that you Eclipse version have some requiered plugins. If not you can copy and paste it from the archive. Here are the requiered plugins :

- Directories:
 - org.apache.xerces_2.7.0
 - org.eclipse.jem.util_1.1.0
- Jar files:
 - org.eclipse.emf.common_2.1.0
 - org.eclipse.emf.ecore.edit_2.1.0
 - org.eclipse.emf.ecore.xml_2.1.0
 - org.eclipse.emf.ecore_2.1.0
 - org.eclipse.emf.edit_2.1.0
 - org.eclipse.wst.common.emf_1.0.0
 - org.eclipse.wst.common.emfworkbench.integration_1.0.0
 - org.eclipse.wst.common.environment_1.0.0

- -How To develop and use CLIF ISAC plug-ins
 - org.eclipse.wst.common.project.facet.core_1.0.0
 - org.eclipse.wst.common.ui_1.0.0
 - org.eclipse.wst.common.uriresolver_1.0.0
 - org.eclipse.wst.common.frameworks_1.0.0
 - org.eclipse.wst.dtd.core_1.0.0
 - org.eclipse.wst.sse.core_1.0.0
 - org.eclipse.wst.sse.ui_1.0.0
 - org.eclipse.wst.validation_1.0.0
 - org.eclipse.wst.xml.core_1.0.0
 - org.eclipse.wst.xml.ui_1.0.0
 - org.eclipse.xsd_2.1.0

When you will launch Eclipse you will see CLIF options. You will also be able to pass into an ISAC perspective and to access ISAC options for creating ISAC plug-ins.



🗦 Isac Perspecti	ve - LdapScenario.xis - Eclipse	SDK	
File Edit Navigate	Search Project Run CLIF Windo	ow Help	
New	Alt+Shift+N 🕨	📬 Project	:::::::::::::::::::::::::::::::::::::
Open File		19 ISAÇ Plug-in Project	
Close	Ctrl+W	a Clif testPlan	on="1.0"2>
Close All	Ctrl+Shift+W	Rew Isac scenario	PE scenario PUBLIC "-//Object
[] Save	Ctrl+S	T Example	io>
📓 Save As			_haviors>
Revert	Ctrl+Shift+S	Other	<pre></pre>
Kevelt		7	<params></params>
Move		8	<param log<="" name="lds</td></tr><tr><td>Rename</td><td></td><td>9</td><td><param name=" td=""/>
Refresh	ES	10	value="cn=ad
- ··· - h ·		11	

To use the Eclipse-RCP based standalone console you have to unzip the console-linux or console-Windows or console-Macosx archives (depends on your operating system).

To launch your Eclipse-RCP based standalone console launch the clif-console.exe / clif-console (depends on your OS) file in the clif-<version>-console directory.

Dossiers	×	Nom 🔺	Taille	Туре
🕀 🛅 Clif	~	Configuration		File Folder
😑 🚞 clif-1.2.1-console		adoc		File Folder
🗉 🧰 configuration		Colicense		File Folder
🗉 🧰 doc		🚞 plugins		File Folder
🗉 🧰 license		C workspace		File Folder
🕀 🛅 plugins		😂 clif-console.exe	108 Ko	Application
🕀 🚞 org.apache.ant_1.6.5		🔊 startup.jar	31 Ko	Executable Jar File
🕀 🧰 org.apache.lucene_1.4.3				
🕀 🧰 org.apache.xerces_2.7.0	=			
🕀 🧰 org.eclipse.help.webapp_3.1.0	_			
🕀 🧰 org.eclipse.jdt.debug_3.1.0				
🕀 🧰 org.eclipse.jem.util_1.1.0				
🕀 🧰 org.eclipse.tomcat_4.1.30.1				
🗉 🛅 org.objectweb.clif.console.plugin_1.2.2				
🗉 🧰 org.objectweb.clif.isac_1.2.2	~	<		>

You can also launch it using command line, a number of useful options may be set on the command line :

- · -consoleLog to see messages printed to your terminal
- -vm /path/to/the/jvm to set the right Java Virtual Machine to be used
- data my_workspace to use a different workspace from the default one

So to launch it with console log you have to tape :

/path/to/the/clif-<version>-console/clif-console(.exe) -consoleLog
Now you are able to write your ISAC plug-ins using the Eclipse-RCP standalone console :

🕷 ClifConsole		
File Edit CLIF Search Run Window	Help	
i 🔜 i 💁 • i 🖋 i 📾 • i 🖢	- 🔁 - 1 🏷 1 🍝	
🔁 Navigator 🛛 🗖 🗖		
(→ → @ 🖻 🕏 🏹		
	Load profiles 🚳 Monitor 🛛	

2. How to Define an LDAP injector writing an ISAC plug-ins

2.1. LDAP directory overview

LDAP is a lightweight directory access protocol described in RFC 2251-2256,2829-2830, It defines a lightweight access mechanism in which clients send requests to and receive responses from LDAP servers.

LDAP enabled directories use a data model that **assumes** or **represents** the data as a hierarchy of objects. This does not imply that LDAP is an object-oriented database. As pointed out above, LDAP itself is a protocol that allows access to an LDAP enabled service and does not define how the data is stored - but the operational primitives (read, delete, modify) operate on a model (description) of the data that has object-like characteristics (mostly).

Object Tree Structure :

Data are represented in an LDAP enabled directory as a hierarchy of objects, each of which is called an entry. The resulting tree structure is called a Data Information Tree (DIT). The top of the tree is commonly called the root (a.k.a **base** or **suffix**).

Each **entry** in the tree has one parent entry (object) and one or more child entries (objects). Each child entry (object) is a sibling of its parent's other child entries.

Each entry is composed of (is an instance of) one or more objectClass. **ObjectClasses** contain zero or more attributes. Attributes have names (and sometimes abbreviations or aliases) and typically contain data (at last!).

Summary:

- 1. Each Entry is composed of one or more objectClasses
- 2. Each objectClass has a name and contains Attributes.
- 3. Each Attribute has a name, usually contains data and is a member of an object class.

Attributes :

Each **attribute** has a name and normally contains data. **Attributes** are always associated with (are members of) one or more **ObjectClasses**. **Attributes** have a number of interesting characteristics:

- 1. All attributes are members of one or more objectclass(es)
- 2. Each **attribute** defines the data type that it may contain.
- 3. Attributes can be optional or mandatory.
- 4. Attributes can have single or multi values,
- 5. Attributes have names and sometimes aliases or abbreviations,

6. At each level in the hierarchy the data contained in one **attribute** should uniquely identify the **entry**. It can be any **attribute** in the <u>entry</u>. It can even be a combination of two or more attributes.

ObjectClass :

ObjectClasses are essentially packages of **attributes**. There are a confusing number of predefined **objectClasses**, each of which contains bucket-loads of **attributes** for almost all common or garden applications. But of course the one you NEED is never defined! **objectclasses** have two more characteristics:

- 1. The **objectclass** defines whether an attribute member **MUST** (<u>mandatory</u>) be present or **MAY** (<u>optional</u>) be present.
- 2. The **objectclass** may be part of a hierarchy in which case it **inherits** all the characteristics of its parent **objectclasses**.



2.2. How to Write a LDAP Injector ISAC Plug-in

Writing your own ISAC plug-in is a simple way to customize the injection capabilities of ISAC, still relying on the generic language for defining behaviors and load profiles. Writing an ISAC plug-in basically consists in defining a Java class that encapsulates (a part of) the state of each behavior instance, and provides specific methods for:

• instantiating new session objects for new behavior instances;

- implementing load injection primitives;
- implementing timer primitives (e.g. to implement think times);
- implementing external data provisioning;
- implementing condition primitives;
- session object control primitives.

The primitives offered by an ISAC plug-in, as well as a GUI-oriented description for its parameters, are declared through 3 descriptor files:

- plugin.properties specifies Java properties plugin.name, plugin.xmlFile and plugin.guiFile to respectively set the ISAC plug-in name, the name of the XML file describing the list of primitives and parameters, and the name of the XML file describing the GUI concerns. Usual values for these file names respectively are plugin.xml and gui.xml.
- plugin.xml (or any other name as specified in plugin.properties file)
- gui.xml (or any other name as specified in plugin.properties file)

2.2.1. How to Create an Isac Plug-in project

To add a new ISAC plug-in, you must create a directory in subdirectory isac/plugins of the CLIF execution environment. For our example we will create a MyLdapInjector directory.

🖃 🚞 clif-1.2.2-console
🗉 🧰 configuration
🗉 🧰 doc
🗉 🚞 license
🖃 🧰 plugins
🗉 🚞 org.apache.ant_1.6.5
🗉 🛅 org.apache.lucene_1.4.3
🗄 🛅 org.apache.xerces_2.7.0
🗉 🚞 org.eclipse.help.webapp_3.2.2.R322_v20061114
🗉 🧰 org.eclipse.jdt.debug_3.2.2.r322_v20070130
🗉 🧰 org.eclipse.jem.util_1.1.0
🗉 🧰 org.eclipse.tomcat_4.1.30.1
🖃 🚞 org.objectweb.clif.console.plugin_1.2.2
🕀 🧰 doc
🖽 🧰 examples
🗆 🛄 Isac
🗏 🛄 piugins
E ConstantTimer
E C DnsIniector
🗉 🛄 FileReader
🛅 HelloWorld
🕀 🫅 HttpInjector_1.0
🕀 🛅 HttpMatrix_1.0
🗉 🧰 Jdbc_1.0
🕀 🛅 JDBCSimple_1.0
🕀 🧰 LdapInjector
🕀 🧰 output 🦷 🔨
🕀 🚞 Random
🗉 🚞 SIPInjector_1.0
🕀 🛅 SocketInjector
🕀 🗀 StateTransitionMatrix
🖽 🧰 StringHandler

To be able to act on an LDAP directory we need some specific librairies. You have to get ldap.jar and utilities.jar. You can get these librairies into the LdapInjector/lib diectory, Once you have them, create repertory lib in MyLdapInjector and paste them into it.

Dossiers	×	Nom 🔺	Taille	Туре	Date de modification
🖃 🧰 LdapInjector	^	i≦ ldap.jar i≦ utilities.jar	432 Ko 42 Ko	Executable Jar File Executable Jar File	15/06/2007 11:32 15/06/2007 11:32

From the Eclipse-RCP standalone console you can use the ISAC plug-in creation Wizard :

Click on : File > New > ISAC Plug-in Project

🍯 ClifConsole	
File Edit CLIF Search	Run Window Help
New	🕨 🍱 ISAC Plug-in Project
Open File	💰 Clif testPlan
Close	Ctrl+W
Close All	Ctrl+Shift+W
📙 Save	Ctrl+s
📓 Save As	
😭 Save All	Ctrl+Shift+S
Convert Line Delimiters	;To 🕨
Revert	
1 LdapTest_plan.ctp [/	essai]

Enter your project name : in our case MyLdapInjector

Enter your project location (refer to the repertory that you created before). In our case the location is :

/path/to/the/eclipseStandAloneConsole/clif-<versions>-

console/plugins/org.objectweb.clif.console.plugin_1.2.2/isac/plugins/MyLdapInjector

🕷 New ISAC I	Plugin Project	×
New ISAC Plu	ıgin	
Create a new IS	AC Plugin project.	
Project name:	MyLdapInjector	
🗌 Use defaul	It location	
Location: C:\	Travail\clif-1.2.2-console\plugins\org.objectweb.clif.console.plugin_1.2.2\isac\plugins\MyLdapInjector	Browse
	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish	Cancel

Click on the Next button.

🍯 New ISAC Plugin Project	$\overline{\mathbf{X}}$
Java Settings	
Define the Java build settings.	
😕 Source 🛛 Projects 🛋 Libraries 🍫 Order and Export	
Source folders on build path:	
🗄 🇀 MyLdapInjector/src	Add Folder
Allow output folders for source folders	
Default output folder:	
MyLdapInjector/bin	Bro <u>w</u> se
< <u>Back</u>	Einish Cancel

In the java settings, you have to add LDAP librairies (Idap.jar and utilities.jar) to your classpath.

🕷 New ISAC Plugin Project 🛛 🔀
Java Settings Define the Java build settings.
Image: Source Projects Libraries Order and Export JARs and class folders on the build path: Image: Add JARs Add JARs Image: Add Jars Image: Add Jars Add JARs Image: Add Jars Image: Add Jars Add External JARs Image: Add Jars Image: Add Jars Add External JARs Image: Add Jars Add External JARs Add External JARs Image: Add Jars Add Mariable Add Mariable Image: Add Jars Add Jars Add Mariable
< <u>B</u> ack <u>N</u> ext > <u>Finish</u> Cancel

Click on the Next button.

Set the plugin properties. Most of them have default values.

You will need to tick the Implements DataProvider interface box if your plugin have to provide data.

🐻 New ISAC P	lugin Project	\mathbf{X}
Plugin proper	ties	
Complete propert	ties for your ISAC Plugin.	
		1
Plugin name:	MyLdapInjector	
Source:	/MyLdapInjector/src	Browse
Package:	org.objectweb.isac.plugin.myldapinjector]
Class name:	SessionObject]
	Implements DataProvider interface	
GUI file name:	qui.xml	1
Plugin file name:	plugin.xml	
		,
	< <u>B</u> ack <u>N</u> ext > <u>Einish</u>	Cancel

You can now click on the Finish button. The Eclipse Isac perspective will be loaded. It is recommended to click on Finish now to be sure that all settings will be saved in case of an Eclipse crash.

🕷 ClifConsole 📃 🗆 🔀					
Eile Edit Source Refactor CLIF Search Run Window Help					
i 🖫 i 🏇 • 🔘 • 🍇 • i 🖄 🕸 ♂ • i 🕭 🖋 i 🕞 • i ½ - i ½ - i ½ - i 🏍					
🛱 Package Explorer 🔀 Hierarchy 🗖 🗖		📴 Outline 🖾			
수 수 🗟 📄 😫 🏹		An outline is not avail	lable.		
😑 🍯 MyLdapInjector					
😟 🔬 clif-full.jar - C:\Travail\clif-1.2.2-console\plugi					
src					
⊕ ➡ JRE System Library [jre1.6.0_02]		<u> </u>	->1 <u>-</u>		
i Duild.xml	Corrers Querpipes Qipfos	Declaration			
			Resource	Path	
plugin.xml					
				_	
LdapLoadTest					

2.2.2. Writing your ISAC plug-ins

Descriptor files overview

The plugin.properties files :

```
plugin.name=MyLdapInjector
plugin.guiFile=gui.xml
plugin.xmlFile=plugin.xml
```

The plug-in descriptor file specifies (plugin.xml by default):

- the plug-in name, which must match the plug-in's directory name,
- · the associated session object class and the initial settings parameters, with some help
- the samples, controls, conditions and timers with their parameters and help.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plugin PUBLIC "-//objectweb.org//DTD CLIF Isac 1.0//EN"
"classpath:org/objectweb/clif/scenario/isac/dtd/plugin.dtd">
</plugin name="MyLdapInjector">
</object class="org.objectweb.isac.plugin.myldapinjector.SessionObject">
</plugin name="MyLdapInjector">
</object class="org.objectweb.isac.plugin.myldapinjector.SessionObject">
</plugin name="MyLdapInjector">
</plugin name="MyLdapInjector">
</plugin name="MyLdapInjector">
</plugin.myldapinjector.SessionObject">
</plugin name="MyLdapInjector">
</plugin.myldapinjector.SessionObject">
</plugin.myldapinjector.SessionObject</plugin.myldapinjector.SessionObject</plugin.myldapinjector.SessionObject</plugin.myldapinjector.SessionObject">
</plugin.myldapinjector.Se
```

The user interface descriptor file (gui.xml by default) adds explicit labels to primitives and parameters, and associates each parameter to GUI-related information. Possible graphical widgets are available through the following tags : radiobutton, field, checkbox, nfield (variable number of fields), combo. Parameters may also be visually grouped together with the group tag. The parameter value resulting from a nfield widget is the concatenation of the variable number of fields separated by one ';' character.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE gui PUBLIC "-//objectweb.org//DTD CLIF IsacGUI 1.0//EN"
"classpath:org/objectweb/clif/scenario/isac/dtd/gui.dtd">
<gui>
<gui>
<object name="SessionObject">
<params></params>
</object>
</gui>
```

The Eclipse wizard also creates a build.xml file which will be used when you will use the ant isacclean and ant isac-plugins tasks of the build.xml file present in the following directory :

clif-<version>-consoleplugins\org.objectweb.clif.console.plugin_<plugin_version>

This XML file contains several ant tasks :

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="ISAC-plugin_MyLdapInjector" default="compile">
<!-- This build.xml file must be called from CLIF main build.xml file. 4
properties are provided by CLIF main build.xml file:
              - libext.dir, all Jars used by the plugin must be copied to this directory
              - clif.classpath, classpath for full CLIF runtime library
              - build.dir, build directory (where classes must be generated and
necessary resource files,
              if any, must be copied)
      isac.dir, ISAC root directory -->
    <!-- General Configuration -->
     <property</pre>
              name="plugin.dir" value="${isac.dir}/plugins/MyLdapInjector">
    </property>
    <!-- classpath definition -->
     <path id="plugin.compile.classpath">
          <pathelement path="${clif.classpath}"></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement>
          <pathelement path="${build.dir}"></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement></pathelement>
    </path>
     <target name="compile">
          <javac
              srcdir="${plugin.dir}/src" destdir="${build.dir}"
             classpathref="plugin.compile.classpath">
          </javac>
          <copy todir="${build.dir}/MyLdapInjector">
               <fileset
                              dir="${plugin.dir}"
                              includes="plugin.properties,plugin.xml,gui.xml">
              </fileset>
          </copy>
    </target>
     <target name="clean"></target>
</project>
```

The SessionObject Class overview

The last file generated by the Eclipse wizard is the SessionObject.java file. This Java class implements the SessionObjectAction interface to handle replication of specimens for creation of session objects that will be actually associated to behavior instances (method createNewSessionObject()). This interface is also used for freeing resources used by session objects before they are discarded (method close()), and recycling old session objects into fresh ones (method reset()).

```
package org.objectweb.isac.plugin.myldapinjector;
import org.objectweb.clif.scenario.isac.plugin.SessionObjectAction;
import java.util.Hashtable;
import org.objectweb.clif.scenario.isac.plugin.DataProvider;
import org.objectweb.clif.scenario.isac.exception.IsacRuntimeException;
```

```
/**
 * Implementation of a session object for plugin ~MyLdapInjector~
 */
public class SessionObject implements SessionObjectAction, DataProvider {
 /**
   * Constructor for specimen object.
   * @param params key-value pairs for plugin parameters
  */
  public SessionObject(Hashtable params) {}
  /**
   * Copy constructor (clone specimen object to get session object).
  * @param so specimen object to clone
  */
  private SessionObject(SessionObject so) {}
  // SessionObjectAction implementation //
  /**
   * @see
org.objectweb.clif.scenario.isac.plugin.SessionObjectAction#createNewSessionObj
ect()
   */
  public Object createNewSessionObject() {
     return new SessionObject(this);
  }
  /**
   * @see org.objectweb.clif.scenario.isac.plugin.SessionObjectAction#close()
  */
  public void close() {}
  /**
   * @see org.objectweb.clif.scenario.isac.plugin.SessionObjectAction#reset()
  */
  public void reset() {}
  // DataProvider implementation //
  /**
   * @see org.objectweb.clif.scenario.isac.plugin.DataProvider#doGet()
   * /
  public String doGet(String var) {
```

```
throw new IsacRuntimeException("Unknown parameter value in
~MyLdapInjector~ ISAC plugin: " + var);
}
```

LDAP injector primitives

Now we can define all the primitives of our LDAP injector. In this tutorial only two primitives will be define :

- · Connection : connection to the LDAP directory (bind)
- CloseConnection : close the connection to the LDAP directory (unbind)

To define these primitives we will use the Eclipse wizard.

Right click on the MyLdapInjector project, then click on properties :



The following window appears :

a Properties for MyLda	plnjector 📃 🗖 🔀
type filter text	Info 🔶 - 🔿 -
… Info … Builders ⊕ ISAC Plugin … Java Build Path	Path: /MyLdapInjector Iype: Project Location: C:\Travail\clif-1.2.2-console\plugins\org.objectweb.clif.console.plugin_1.2.2 \isac\plugins\MyLdapInjector
Java Compiler Java Compiler Javadoc Location Project References Task Tags	Last modified: 6 décembre 2007 09:18:30 Text file encoding Inherited from container (Cp1252) Other: Cp1252 New text file line delimiter Inherited from container Other: Vertext Inherited from container Other: Vertext Inherited from container Other:
	OK Cancel

Now with the ISAC Plugin menu you will be able to define :



Session object

In the session object, global attributes of the LdapInjector will be defined.

First of all, you have to write the definition of your LDAP injector. This definition will be displayed in the help contextual menu when you will define your ISAC scenario.

🕷 Properties for Ldaplr	njector 📃 🗆 🔀
type filter text	~ SessionObject $\Leftrightarrow * \Rightarrow *$
Info Builders ■ ISAC Plugin Conditions Conditions Samples Timers Java Build Path ■ Java Code Style ■ Java Code Style ■ Javadoc Location Project References Task Tags	Help This plugin allows to test a Idap directory. It provides some methods which allow to connect, "SSL Connect", disconnect, search, add entries, delete entries, add attributes, delete attributes. Parameters Parameters
	OK Cancel

After adding contextual help you can define groups of global parameters. For example, you can define a group that contain severals parameters.

Click on

to add a group in the session object.

Give the group name :

°G |

add group	
Give the group name	
Ldap connection parameters	
	OK Cancel

Then you can add severals parameters into the group that you have created. First of all, select it, and then click on



Give the parameter name :

🛎 Add parameter	
Give the parameter name	
hostnþme	

Do it with all your parameters.

If you want to add parameters which are not connection parameters you have to click first on "root"

and then on



Finally you will have something like that :

Properties for MyLdap	pInjector	_ 🗆 🔀
Properties for MyLdag type filter text Info BeanInfo Path Builders ISAC Plugin ~ SessionObject Conditions Controls	A SessionObject Help This plugin allows to test a Idap directory. It provides some methods which allow to connect, "SSL connect", disconnect, search, add entries, delete entries, add attributes, delete attributes. Parameters	
	Image: second secon	
0	ОК	Cancel

Now you have to give each parameter a name, a type, a default text value if necessary, a list of values, depending of the parameters you are defining.

Click on the parameter you want to define :

SessionObject		⇔ - ⇔
Help This plugin allows to test a Idap directory. It provides so	ne methods which allow to connect, "SSL conr	nect",
	ouces, delete attributes.	
Parameters	abel: Host name Type: field Text:	Length: 8 💌

To choose the parameter's type there is a list of choice :

- field
- nfield
- table
- radio button
- checkbox
- combo

In your case (LDAP injector) you will choose field type for all parameters.

So you just have to define a Label (this will be the label displayed when you will write your ISAC scenario), the type, a text if our parameter has a default value and a length (this length represents the size of the input text field of the isac scenario wizard).

These actions with the wizard modify :

SessionObject.java class adding attributes :





• plugin.xml file adding parameters :

```
<params>
  <param name="hostname" type="String"></param>
  <param name="version" type="String"></param>
</params>
```

• gui.xml files adding parameters :

```
<params>
    <group name="Ldap connection parameters">
        <group name="Ldap connection parameters">
        <group name="Host name" name="hostname">
        <field text="" size="8"></field>
        </param>
        </group>
        <param label="LDAP version" name="version">
              <field text="" size="8"></field>
        </param>
        </param>
        </param>
        </param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param>
```

Adding Samples

To add samples is to define primitives of LDAP injector. These primitives are :

- connection
- SSL connection
- closeConnection
- search
- · add entry
- · delete entry
- · add attribute
- · delete attributes

Names	
	Parameters Parameters

Click on the solution and add all the primitives you want to define giving them a name and don't forget to write a description of all of them.

Image: Second connection search closeConnection addEntry deleteEntry addAttribute deleteAttribute Simple connection on the LDAP directory.		Simple connection on the LDAP directory. The Idap connection port is mandatory.	<	
--	--	--	---	--

After adding the primitive's name and description you will have to add their specific attributes. Each primitive has specific attributes :

- connection
 - login : the login DN to connect to the LDAP directory
 - password : the password to connect to the LDAP directory
 - port : the port to connect using non-secure connection.
- sslConnection
 - login : the login DN to connect to the LDAP directory
 - password : the password to connect to the LDAP directory
 - port : The port to connect using secure connection.
 - -keystorePath : The path to access the keystore file.
- closeConnection (no attributes)
- search
 - $-\,searchBase$: the place in the directory tree where to start to searching. (eg :
 - ou=people,dc=orange,dc=fr)
 - searchFilter : the search filter (e.g. : sn=fran*)
 - searchScope : the search depth. (e.g. : 2)
- addEntry
 - dn : the LDAP distinguished name
 - entryNodeType : type of the node to insert (cn or sn or ou ou or uid)
 - entryNodeName : name of the node to insert
 - attributesList : node's list of attributes
- deleteEntry

¥

۲

Length: 8

- dn : distinguished name representing the node to delete.

- addAttribute
 - dn : distinguished name of the node where we want to add attributes
 - attributesList : list of the couple attributes name / attributes values
- deleteAttribute
 - dn : distinguished name of the node where we want to delete attributes
 - attributestodelete : name of the attributes to delete.

Now we focus on the definition of the attributes of the addEntry primitive.

Parameters

÷.

÷

*

To an add attribute just click on the add parameter button (notice that you can define group of attributes as with session object).

- root

P

P

🖪 dn

entryNodeType

attributesList

entryNodeName

The dn attribute has :

- label : Distinguished name
- type : field
- · text : no default value
- · length : 8

The entryNodeType has :

- · label : Insert node type
- type : combo

To add values to combobox click on and enter the different values with a default state (enabled or disabled),

The entryNodeName has :

- label : Insert node name
- type : field
- text : no default text
- length : 8



Label: Distinguished name

Type: field

Text:



The attributesList has :

- label : List of attributes
 (sheme : name=value or name=value1,value2,value3)
- type : nfield

These actions with the wizard modify these files :

· SessionObject.java class :

This variable SAMPLE_ADDENTRY identifies the addEntry primitive in the LdapInjector plugin. Each primitive has unique identifier.

static final int SAMPLE_ADDENTRY = 4;

The variables below are added when you define addEntry attributes. Each attributes of each primitive is identified with a variable which value is the variable name defined in the wizard.

static final String SAMPLE_ADDENTRY_ATTRIBUTESLIST = "attributesList"; static final String SAMPLE_ADDENTRY_ENTRYNODENAME = "entryNodeName"; static final String SAMPLE_ADDENTRY_ENTRYNODETYPE = "entryNodeType"; static final String SAMPLE_ADDENTRY_DN = "dn";

Adding samples with the wizard modify this java class by adding a method doSample which return an ActionEvent. This method has three parameters :

- The first argument gives the primitive identifier.
- The second parameter gives the list of parameter values indexed by their names, as set in the plugin descriptor file using tag params;
- The third argument gives a report object whose fields will have to be filled before being returned.

Basically, the doSample() method is supposed to perform a load injection request, wait for some

kind of response, state if this request is a success or a failure, measure its response time and return a sample report. Returning null is also possible, to make CLIF ignore this sample.

	- Parameters	
s e or value3)	Image: Second	name=value or na

```
/**
  @see org.objectweb.clif.scenario.isac.plugin.SampleAction#doSample()
  */
  public ActionEvent doSample(int number, Map params, ActionEvent report) {
  switch (number) {
      case SAMPLE_DELETEATTRIBUTE:
       break;
     case SAMPLE_ADDATTRIBUTE:
       break;
     case SAMPLE_DELETEENTRY:
       break;
     case SAMPLE_ADDENTRY:
       break;
     case SAMPLE_SEARCH:
       break;
     case SAMPLE_SSLCONNECTION:
       break;
     case SAMPLE_CONNECTION:
       break;
     default:
       throw new Error ("Unable to find this sample in ~LdapInjector~ ISAC
plugin: " + number);
    }
    throw new IsacRuntimeException ("No action defined for this sample in
~LdapInjector~ ISAC plugin: " + number);
    }
```

• plugin.xml :

In the plugin.xml file the addEntry sample is added with all its parameters and its help.

```
<sample name="addEntry" number="4">
                         <params>
                                       <param name="dn" type="String"></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param>
                                      <param name="entryNodeType" type="String"></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param>
                                      <param name="entryNodeName" type="String"></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param>
                                      <param name="attributesList" type="String"></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param></param>
                          </params>
                           <help>Add entry into an ldap directory.&#xD;
To be able to add an entry into ldap directory the following parameters are
madatory :

                         - Distinguished name

                          - Node type (type of the node to insert) & #xD;
                            - Node name (name of the node to insert) & #xD;
                          - List of the attributes : couple name=value(s) of the attributes which
composed the node.</help>
           </sample>
```

• gui.xml :

The user interface descriptor file adds explicit labels to primitives and parameters, and associates each parameter to GUI related information. Graphical widgets appears here (combo, field, nfield).

```
<sample name="addEntry">
   <params>
      <param label="Distinguished name" name="dn">
        <field text="" size="8"></field>
      </param>
      <param label="Insert node type" name="entryNodeType">
       <combo>
          <choice value="cn" default="false"></choice>
          <choice value="sn" default="false"></choice>
          <choice value="ou" default="false"></choice>
          <choice value="uid" default="false"></choice>
       </combo>
      </param>
      <param label="Insert node name" name="entryNodeName">
        <field text="" size="8"></field>
      </param>
      <param label="List of attributes (sheme : name=value or</pre>
name=value1,value2,value3)" name="attributesList">
       <nfield></nfield>
     </param>
   </params>
  </sample>
```

Complete SessionObject class with Java code

At this point we have finished using the ISAC plug-in wizard. We are now going to write Java code to implement the LDAP connection (secure or non-secure), disconnection and add entry methods.

When you write an ISAC scenario you can import several times the LdapInjector plugin with differents settings. For each import the ISAC execution engine instantiates and initializes with specific settings a specimen session object. For that purpose, your plugin class must implement a public constructor taking a Map as a single argument. This Map will hold the specimen settings with the parameters names as keys, as specified in the plugin XML descriptor file. The specimen objects will be used just for replication, according to the load profiles, but will never be associated to behavior instances.

Here are global parameters which have to be set for every imported LdapInjector plugin.

```
private String hostname;
private int ldapVersion;
```

In the SessionObject constructor we checked the value of the parameters. These must not be null or empty, they must be set.

```
/**
   * Constructor for specimen object.
  * @param params key-value pairs for plugin parameters
  * @throws ClifException
  */
  public SessionObject(Hashtable params) throws ClifException {
  // local address setting
   String value = (String)params.get(PLUGIN_HOSTNAME);
    if (value != null && value.length() > 0) {
      try {
       hostname = value;
     } catch (Exception ex) {
       throw new ClifException ("ISAC can't get hostname because the specified
hostname is not valid: " + value, ex);
     }
    } else {
     throw new ClifException ("ISAC can't get hostname because the specified
hostname is not valid: " + value);
   }
 // ldap version setting
   value = (String)params.get(PLUGIN_VERSION);
    if (value != null && value.length() > 0) {
      try {
       ldapVersion = Integer.parseInt(value);
     } catch (Exception ex) {
       throw new ClifException ("ISAC can't get LDAP version because the
specified LDAP version is not valid: " + value, ex);
     }
   } else {
     throw new ClifException ("ISAC can't get LDAP version because the
specified LDAP version is not valid: " + value);
   }
 }
  /**
   * Copy constructor (clone specimen object to get session object).
  * @param so specimen object to clone
  */
   private SessionObject(SessionObject so) {
     this.hostname = so.hostname;
     this.ldapVersion = so.ldapVersion;
    }
```

Declare your LDAPConnection object to be able to access it from every implemented method which will manipulate your LDAP directory. This object will be instantiate in the doConnection() method.

private LDAPConnection ldapConnection;

All the methods implement for this LDAP injector have three parameters, like the doSample method :

- The first argument gives the primitive identifier;
- The second parameter gives the list of parameter values indexed by their names, as set in the plugin descriptor file using tag params;
- The third argument gives a report object whose fields will have to be filled before being returned.

And return an ActionEvent.

These following methods are called by the doSample() method depending of the primitive identifier and need to be implemented :

- doConnection(int number, Map params, ActionEvent report);
- doSslConnection(int number, Map params, ActionEvent report);
- doAddEntry(int number, Map params, ActionEvent report);
- doDeleteEntry(int number, Map params, ActionEvent report);
- doAddAttribute(int number, Map params, ActionEvent report);
- doDeleteAttribute(int number, Map params, ActionEvent report);
- doSearch(int number, Map params, ActionEvent report);
- doCloseConnection(int number, Map params, ActionEvent report);

```
/**
* @see org.objectweb.clif.scenario.isac.plugin.SampleAction#doSample()
*/
public ActionEvent doSample(int number, Map params, ActionEvent report) {
  switch (number) {
   case SAMPLE_DELETEATTRIBUTE:
     return doDeleteAttribute(number, params, report);
   case SAMPLE_ADDATTRIBUTE:
     return doAddAttribute(number, params, report);
   case SAMPLE_DELETEENTRY:
     return doDeleteEntry(number, params, report);
   case SAMPLE_ADDENTRY:
     return doAddEntry(number, params, report);
   case SAMPLE_CLOSECONNECTION:
     return doCloseConnection(number, params, report);
   case SAMPLE_SEARCH:
```

```
return doSearch(number, params, report);
case SAMPLE_SSLCONNECTION:
    return doSslConnection(number, params, report);
case SAMPLE_CONNECTION:
    return doConnection(number, params, report);
    default:
        throw new Error("Unable to find this sample in ~LdapInjector~ ISAC
plugin: " + number);
    }
}
```

Now we will focus more specifically on the addEntry method.

```
/**
 * Add an entry into a LDAP directory. The connection to the LDAP directory
 * must be established before doing the doAddEntry.
 *
 * @param number : The number which handles the connection to the ldap
 * directory.
 * @param report : The ActionReport to update.
 * @param params : A Map containing all the useful variable.
 * @return the report.
 */
 private ActionEvent doAddEntry(int number, Map params, ActionEvent report){
```

First of all we have to declare and set attributes of the addEntry primitive. This parameters are in the params Map passed in parameter of the doAddEntry method. The get method applied on a Map returns an Object so we have to cast it into String to be able to use it.

```
String dn = (String) params.get(SAMPLE_ADDENTRY_DN);
String insertNodeName = (String) params.get(SAMPLE_ADDENTRY_ENTRYNODETYPE);
String insertNodeValue = (String) params.get(SAMPLE_ADDENTRY_ENTRYNODENAME);
String attributesList = (String) params.get(SAMPLE_ADDENTRY_ATTRIBUTESLIST);
```

Then we create a LDAPAttributeSet which is a collection of LDAPAttribute objects. LDAPAttributeSet may be used to build an entry to be added to a directory.

LDAPAttributeSet attributeSet = **new** LDAPAttributeSet();

The attributesList is a string represented by a sequence of couple name=value or name=value1,value2,value3... Each couple is separated by ";".

}

We check if the entry node belongs to the attributesList. If it doesn't, we add it, at the end of the attributesList string :

```
if (attributesList.indexOf(insertNodeName) == -1) {
    attributesList = attributesList+insertNodeName+"="+insertNodeValue+";";
```

We check if the entry node belongs to the dn. If it doesn't, we add it, at the beginning of the dn string;

```
if (dn.indexOf(insertNodeName) == -1) {
    dn = insertNodeName+"="+insertNodeValue+","+dn;
}
```

Then we have to transform the String attributesList into a String[] using the split method.

String[] tabParam_NameValue = attributesList.split(";");

Now we can go into the String[] tabParam_NameValue and add LDAPAttribute to the LDAPAttributeSet :

Then we can create an LDAPEntry object with the dn and attributeSet as parameters.

LDAPEntry newEntry = **new** LDAPEntry(dn, attributeSet);

To be able to identified the add entry lines in the report file you can set the type paramter in the report object :

report.type = "ADDENTRY_TYPE";

And you also need to set the Date parameter of the report object to be able to set the add entry duration.

```
report.setDate(System.currentTimeMillis());
```

Now you just have to add the entry in your LDAP directory. To do it, you have to apply the add method to the IdapConnection object which is a global attribute of the SessionObject class and which is initialized in the connection() method.

Wee also have to catch exceptions and to set the result of the add, a comment, state of the add method (successful or not) :

```
try {
 ldapConnection.add( newEntry );
 report.duration = (int) (System.currentTimeMillis());
 report.successful = true;
 report.comment = "Added object: " + dn + " successfully.";
} catch (LDAPException ex) {
 report.successful = false;
 if (ex.getResultCode() == LDAPException.ATTRIBUTE_OR_VALUE_EXISTS) {
   report.result = ex.toString();
   report.comment = "Failed to add existing attribute.";
  } else {
   report.result = ex.toString();
   report.comment = "Failed to add attribute.";
  }
} catch (Exception e) {
 report.result = e.toString();
 report.comment = "Failed to add existing attribute.";
}
```

And finally you have to return the report.

return report;
}

To be able to test make a search in an LDAP directory you will need to implement doConnection(), doCloseConnection and search() methods.

Here is the Java code of these methods :

```
/**
 * Does a connection to a LDAP directory (bind).
 *
 * @param number : The number which handles the connection to the ldap
directory.
 * @param report : The ActionReport to update.
 * @param params : A Map containing all the useful variable.
 * @return the report.
 */
public ActionEvent doConnection(int number, Map params, ActionEvent report) {
    ldapConnection = new LDAPConnection();
    String login = (String) params.get(SAMPLE_CONNECTION_LOGIN);
    String password = (String) params.get(SAMPLE_CONNECTION_PASSWORD);
```

```
report.type = "CONNECT_TYPE";
    report.setDate(System.currentTimeMillis());
   try {
        // connect to the server
       ldapConnection.connect( hostname, Integer.parseInt((String)
params.get(SAMPLE CONNECTION PORT)));
        // bind to the server
        ldapConnection.bind(ldapVersion, login, password.getBytes("UTF8"));
       report.duration = (int) (System.currentTimeMillis() - report.getDate());
        report.successful = true;
        report.comment = "Successful bind with server.";
    } catch ( LDAPException e ) {
       report.successful = false;
        report.result = e.toString();
       report.comment = "ISAC LdapInjector error occured";
    } catch ( UnsupportedEncodingException e ) {
       report.successful = false;
       report.result = e.toString();
        report.comment = "ISAC LdapInjector can't UTF8 encode password
"+password;
   }
    return report;
}
/**
* Close an opened connection to a LDAP directory (bind).
* @param number : The number which handles the connection to the ldap
directory.
* @param report : The ActionReport to update.
* Oparam params : A Map containing all the useful variable.
 * @return the report.
*/
public ActionEvent doCloseConnection(int number, Map params, ActionEvent
report)
{
    if (ldapConnection != null) {
       report.type = "DISCONNECT_TYPE";
       report.setDate(System.currentTimeMillis());
       try {
           // disconnect with the server
           ldapConnection.disconnect();
           report.duration = (int) (System.currentTimeMillis() -
report.getDate());
           ldapConnection = null;
           report.successful = true;
           report.comment = "ISAC LdapConnection disconnection Successful";
```

```
return report;
       } catch (LDAPException e) {
           report.successful = false;
           report.comment = "ISAC LdapInjector can't disconnect from " +
hostname;
           report.result = e.toString();
           return report;
       }
    } else {
       report.successful = false;
       report.comment = "ISAC LdapInjector can't disconnect unopen ldap
connection";
       report.result = "ignored";
       return null;
      }
}
/**
* Does search into a LDAP directory.
* @param number : The number which handles the connection to the ldap
directory.
* @param report : The ActionReport to update.
* @param params : A Map containing all the useful variable.
* @return the report.
 */
public ActionEvent doSearch(int number, Map params, ActionEvent report) {
     try {
           report.type = "SEARCH_TYPE";
           report.setDate(System.currentTimeMillis());
            LDAPSearchResults searchResults = ldapConnection.search((String)
params.get(SAMPLE_SEARCH_SEARCHBASE),Integer.parseInt((String)params.get(SAMPLE
_SEARCH_SEARCHSCOPE)), (String)
params.get(SAMPLE_SEARCH_SEARCHFILTER),null,false);
            report.duration=(int) (System.currentTimeMillis()-report.getDate());
            report.successful = true;
           report.comment = "LDAP Search succeded";
      } catch ( LDAPException e ) {
           report.successful = false;
           report.result = e.toString();
           report.comment = "ISAC LdapInjector error occured";
     return report; }
```

To be able to compile your ISAC plug-in you have to complete the build.xml file. <?xml version="1.0" encoding="UTF-8"?> <project name="ISAC-plugin MyLdapInjector" default="compile"> <!-- This build.xml file must be called from CLIF main build.xml file. 4 properties are provided by CLIF main build.xml file: - libext.dir, all Jars used by the plugin must be copied to this directory - clif.classpath, classpath for full CLIF runtime library - build.dir, build directory (where classes must be generated and necessary resource files, if any, must be copied) - isac.dir, ISAC root directory -> <!-- General Configuration --> <property name="plugin.dir" value="\${isac.dir}/plugins/MyLdapInjector"></property> <!-- classpath definition --> <path id="plugin.compile.classpath"> <pathelement path="\${clif.classpath}"></pathelement> <pathelement path="\${build.dir}"></pathelement> <fileset dir="\${plugin.dir}/lib" includes="*.jar"></fileset> </path> <target name="compile"> <javac srcdir="\${plugin.dir}/src" destdir="\${build.dir}" classpathref="plugin.compile.classpath"></javac> <copy todir="\${libext.dir}" overwrite="yes" preservelastmodified="yes"> <fileset dir="\${plugin.dir}/lib" includes="*.jar"></fileset> </copy> <copy todir="\${build.dir}/MyLdapInjector"> <fileset dir="\${plugin.dir}" includes="plugin.properties,plugin.xml,gui.xml"></fileset> </copy> </target> <target name="clean"></target> </project>

Now you just have to compile and make a jar file with all your ISAC plug-ins. Go to the following repertory :

/path_to_your_clif_console/plugins/org.objectweb.clif.console.plugin_<version>

and launch the ant command : ant isac-plugins Once it is successful you can use your ISAC plug-ins.
3. 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.

3.1. 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.

Wee 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 plugin to be able to do this action.

To externalize the data used in the scenario (mandatory parameters of the LdapInjector plugin) 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 plugin.

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

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 plugin.

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

3.2. Record your ISAC scenario

Create your project and your ISAC scenario

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

Click on File -> New -> Other...

ile Edit Run CLIF Search Windo	ow Help
New	🕨 🎶 ISAC Plug-in Project
Open File	🚳 Clif testPlan
Close	Ctrl+W Rew Isac scenario
Close All	Ctrl+Shift+W
- J Save	Ctrl+s
Save As	
ੈ Save All	Ctrl+Shift+S
Convert Line Delimiters To	▶
Revert	
1 StringHandlerSessionObject.java [[Te]
2 Scenario_5000_50reqs_basic.xis [[Lda]
3 Scenario_5000_500reqs_basic.xis	[Ld]
4 Scenario_5000_2000reqs_basic.xi	s [L]
D .4	

After it, choose General and select Project.

🛎 New				
Select a wizard Create a new project resource				
Wizards:				
type filter text				
General File Folder Folder Project CLIF CLIF General General Project Stac General Gen				
	< Back	Next >	Finish	Cancel

Then click on Next.

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

🕷 New Project	
Project	
Create a new project resource.	
Project name: LdapInjectorLoadTest	
Use default location	
Location: C:/Travail/clif-1.2.2-console/workspace/LdapInjectorLoadTest	Browse
< Back Next > Finish	Cancel

Click on Finish.

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

CLIF	Window	Help		_
6 8 •	New New	Window Editor		* *\$ \$ + => *
	Open	Perspective	۲	Clif Perspective
	Show	View	•	💦 Isac Perspective 🛛 🗟
	Custo	omize Perspective		🐉 Java
	Save	Perspective As		Other
	Rese	t Perspective		

You can now see this perspective :

1	ClifC	onsol	le							
File	Edit	Run	CLIF	Search	Window	Help				
: 6	1 :	Q	1 🔗	16	• 🗄 🖢	- 6	* -	*	16	
8 5.	Navig	jator 8	×			$\Rightarrow \Rightarrow$	Q		\$₽ ~	
	L ![.dapInj 🗙 .pro	ectorLo)ject	adTest						

Create your new isac scenario.

```
Click on File -> New -> New Isac Scenario
```

6	ClifC	onsol	e				
File	Edit	Run	CLIF	Search	Window	Help	
I	New					►	🖄 ISAC Plug-in Project
	Open I	⁼ile…					🐔 Clif testPlan
	Close					Ctrl+W	Rew Isac scenario
	Close /	All				Ctrl+Shift+W	Dther
	Save					Ctrl+S	
H.	Save A	\s					
r)	Save A	di 👘				Ctrl+Shift+S	
	Conve	rt Line	Delimite	ers To		+	
I	Revert						

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

8		
Isac Scen	ario File	
This wizard o	creates a new file with *.xis extension that can be opened Isac Scenario Editor.	
<u>C</u> ontainer:	/LdapInjectorLoadTest	Browse
<u>F</u> ile name:	LdapInjectorLoadTest_1.xis	
	<u>Einish</u>	Cancel

Click on Finish.

🛎 ClifConsole	
File Edit Run CLIF Search Window Help	
i 🔜 i 💁 • i 🛷 i 📾 • i 🖄 • 🖗	
🞏 Navigator 🛛 🗌 🗖	🚼 ldapinjectorLoadTest_Lixis 🔀
C C C C C C C C C C C C C C C C C C C	<pre></pre>

Now you can see the content of your ISAC scenario file.

At the bottom of the file content there are four tabulations

<				
Design	Source	Import	Behavior	

.

Design Source Import Behavior

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.

Import ISAC plugins

Click on the Import tabulation to load the Import perspective.

Import Page :

Plug-ins :

List of plug-ins used in this scenario

Add

Remove

Remove All

Help

Up Down

Add behavior

Import the plugins :

- LdapInjector
- CSVProvider
- ConstantTimer
- Random
- Chrono

Click on the Add button : Add

Choose the plugins you need one by one :

ISAC Plug-ins	
Add a new plug-in	
©=Common	
- FileReader	
The action of the second secon	
© Constant miler	
*=Chrono	
<pre>SIPInjector_1.0</pre>	
€=HttpInjector_1.0	
≎=HelloWorld	
Context	
🗢 StringHandler	
*=Random	
≎=Counter	
*=LdapInjector	
0	Einish Cancel

Now all the imported plugins are present in the Import perspective.

LdapInjectorLoadTest_1.xis	
Import Page :	
Plug-ins :	
List of plug-ins used in this scenario	
 MyLdapInjector_0 : MyLdapInjector CSVProvider_0 : CSVProvider ConstantTimer_0 : ConstantTimer Random_0 : Random Chrono_0 : Chrono 	Add Remove Remove All Help Up Down
Design Source Import Behavior]

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

Select the MyLdapInjector_0:MyLdapInjector and sets the parameters :

Import Page :			
Plug-ins : List of plug-ins used in this scenario	Add	Properties : Manage plug-ins properties use : ConstantTimer.ConstantTimer	
ConstantTimer_0 : CovProvider ConstantTimer_0 : ConstantTimer Random_0 : Random C Chrono_0 : Chrono	Remove All Help Up Down Add behavior	id : ConstantTimer_0 duration (ms) : 1000	

Do the same thing with the ConstantTimer plugin.

The Random and Chrono plugin have no general parameter.

Select the CSVProvider plugin and sets its parameters :

	use : CSVProvider.CSVProvider		
d: CSVPro	vider 0		
	-		
ile name : 🛛	LdapLoadTest_1.csv		
-ield separat	эг: #		
ields names	(separated by the given separator) : [login#password#searchBase#searchFilter#searchScope]		
	······································		
_MacOS9 lin	e separator		
MacOS9 lin	e separator		
-MacOS9 lin	e separator		
-MacOS9 lin enable shared	e separator		
MacOS9 lin	e separator		
MacOS9 lin	e separator		

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
$uid=2, ou=appli, dc=annuaire, dc=com \# secret_2 \# ou=appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=com \# (\& (uid=2)(autorisation=all)) \# 2 = appli, dc=annuaire, dc=annua$

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 checkbox

- Mac0S9 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 has been reached, and the empty string is used as value.

Define your behavior

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

Design Source Import Behavior

Once this tabulation selected you can see the following perspective :

🚼 *Lda	apInjecto	orLoadTe	est_1.xis	×
Behavior Page :				
Editio	n page	for beh	avior des	cription
Beh.	avior id : New Dad prol Zhavior	Uuplica	te Rem	ove
			(Add
			(Remove
			(Clear
			(Help
				Up Down
Design	Source	Import	Behavior	

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

Behavior Page :

Edition page for behavior description

havior id : Heavy load
havior id : Heavy load

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 depends on a specific random distribution.

Click on the add button Add in the behavior perspective.

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

Actions	X
Add a new action for behavior I	Heavy load
 CSVProvider_0 Random_0 Chrono_0 MyLdapInjector_0 if while nchoice preemptive 	Random.sleep Random.setPoisson Random.setUniform Random.setUniform Random.setDirac Random.setNegativeExpo Random.setGaussian
Sets the uniform random distribution law. Parameters: - min (required): the minimal integer value - max (required): the maximal integer valu	, je,
?	<u>Finish</u> Cancel

We sets the required parameter's value :

Content of the second s	
Behavior Page :	
Edition page for behavior description	Properties :
	Manage plug-ins properties
Behavior id : 0	control : Random.setUniform
New Duplicate Remove	
Load profile :	id : Random_0
Behavior tree :	
	min : 0
Remove	max: 1000
Clear	
Help	
Up Down	
Design Source Import Behavior 0	

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 <u>Add</u> in the behavior perspective.

actions	×	
Add a new action for behavior Heavy load		
 ConstantTimer_0 CSVProvider_0 Random_0 Chrono_0 MyLdapInjector_0 if while nchoice preemptive Timer whose duration is given by a random variable is given by a random variable.	Random.action Random.sleep Standom.setPoisson Random.setUniform Random.setDirac Random.setNegativeExpo Random.setGaussian value in ms, according to current distribution law settings.	
0	<u>E</u> inish Cancel	

And set the parameter's value of this action :

Behavior Page :			
Edition page for behavior description		Properties :	
		Manage plug-ins properties	
Behavior id : Heavy load		timer : Random.sleep	
New Duplicate Remove			
Load profile :		id : Random_0	~
▼ Behavior tree :			
6 ⁹ Random_0.setUniform	Add		
	Remove		
	Clear		
	Help		
	Up Down		

Now we add a while loop.

ConstantTimer_0 Random_0 Chrono_0 MyLdapInjector_0 🥐 if 姠 while ■E nchoice 🚯 preemptive

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.

LdapInjectorLoadTest_1.xis	- 6
Behavior Page :	
Edition page for behavior description	Properties :
	Manage plug-ins properties
Behavior id : Heavy load	Select a test condition of the while controler node :
	Chrono.is_on
	Chrono.is_on Chrono.is_gt
Load profile : Rebayior tree :	Chrono.is_gte CSVProvider.endOfFile
	CSVBrovider, notEndoffile
Control Add Add Add	
while Remove	
Clear	
Help	
Up Down	
Design Source Import Behavior Heavy load	

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

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

actions 🗢	
Add a new action for behavior H	eavy load
ConstantTimer_0 CSVProvider_0 Random_0 Chrono_0 MyLdapInjector_0 if while nchoice preemptive starts the chrono	Chrono.stop Chrono.split Chrono.drop Chrono.suspend Chrono.resume
0	Einish Cancel

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}

LdapInjectorLoadTest_1.xis 🛛	
Behavior Page :	
Edition page for behavior description	Properties :
	Manage plug-ins properties
Behavior id : Heavy load	sample : MyLdapInjector.connection
New Duplicate Remove Load profile : Behavior tree :	id : MyLdapInjector_0
Image: Section of the section of th	Login DN : \${CSVProvider_0:login} Password : \${CSVProvider_0:password} LDAP connection port : 389

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

Add a new action for behavior Heavy load		
 ConstantTimer_0 CSVProvider_0 Random_0 Chrono_0 Chrono_10 MyLdapInjector_0 if while nchoice preemptive 	 MyLdapInjector.connection MyLdapInjector.closeConnection MyLdapInjector.addAttribute MyLdapInjector.search MyLdapInjector.deleteEntry MyLdapInjector.addEntry MyLdapInjector.deleteAttribute MyLdapInjector.sslConnection 	
 MyLdapInjector.sslConnection Research in the LDAP directory. Before the research, a connection or a ssl connection must be done. The following parameters are mandatory : The LDAP connection port The search base The search filter The search scope 		
(2)	Finish Cancel	

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

🚼 LdapInjectorLoadTest_1.xis 🛛				
Behavior Page :				
Edition page for behavior description		Properties :		
		Manage plug-ins prop	erties	
Behavior id : Heavy load			sample : MyLdapInjector.search	
New Duplicate Remove				
Load profile : Rehavior tree :		id : MyLdapIn	jector_0	~
@ Random_0.setUniform	Add	Search base :	\${CSVProvider_0:searchBase}	
 ⇒	Remove	Search filter :	\${CSVProvider_0:searchFilter}	
Orrono_0.start MyLdapInjector_0.connection	Clear	Search scope :	\${CSVProvider_0:searchScope}	
🦾 🖟 MyLdapInjector_0.search	Help			
	Up Down			
Design Source Import Behavior Heavy load				

Once the research done we have to close the connection.

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

Actions		
Add a new action for behavior Heavy load		
ConstantTimer_0 CSVProvider_0 Random_0 Chrono_0 MyLdapInjector_0 if while nchoice preemptive Close the opened LDAP connection.	MyLdapInjector.connection MyLdapInjector.doseConnection MyLdapInjector.addAttribute MyLdapInjector.search MyLdapInjector.deleteEntry MyLdapInjector.addEntry MyLdapInjector.deleteAttribute MyLdapInjector.sslConnection	
0	Einish Cancel	

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

Actions			
Add a new action for behavior Heavy load			
ConstantTimer_0			
<pre>Provider_U Provider_U Provid</pre>	Chrono.split		
Chrono_0	ල් ^{පා} Chrono.drop		
MyLdapInjector_0	ල් ^ස Chrono.suspend		
(♥) it it is a second	ල් ^ස Chrono.resume		
	ල් ^ස Chrono.start		
🔁 preemptive			
records current chrono value in a pseudo sample and stops the chrono.			
0	Finish Cancel		
<u> </u>			

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

LdapInjectorLoadTest_1.xis	
Behavior Page :	
Edition page for behavior description	Properties : Manage plug-ins properties
Behavior id : Heavy load New Duplicate Remove > Load profile :	Manage plug-ins properties sample : Chrono.stop id : Chrono_0 status v successful comment (optional) :
MyLdapInjector_0.connection MyLdapInjector_0.search MyLdapInjector_0.closeConnection Chrono_0.stop Up Down	result (optional) :

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

a Actions		
Add a new action for behavior Heavy load		
 ConstantTimer_0 CSVProvider_0 Random_0 Chrono_0 MyLdapInjector_0 if while nchoice preemptive Go to next line. When loop is enabled, wraps up	CSVProvider.next	
0	<u>E</u> inish Cancel	

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 : Heavy load New Duplicate Remove Load profile : Rebavior tree :	
▼ Behavior tree :	
Random_0.setUniform Andom_0.sleep	Add
🖻 🛷 while	Remove
∽⊚ [©] Chrono_0.start ↓ MyLdapInjector_0.connection	Clear
🦳 🦉 MyLdapInjector_0.search	Help
WyLdapInjector_0.closeConnection G Chrono_0.stop CSVProvider_0.next	Up Down
Design Source Import Behavior Heavy load	

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

🕑 Load profile :	
→ Behavior tree:	

and after it on the Set profile button :

Be carefull, 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.

Remove profile

Load profile :

Set profile

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

Enter 60 in the Populatior	i text field :	Population60	
Choose the ramp style :	Ramp style	when the second	
To save these settings cli	ck on the Ad	ld point button	Add point

Time 60

After we want a stable number of simultaneous scenarii 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 simultaneous scenarii will rise to 100 scenarii.

🕷 Load profi		
Add points fo	r this profile O	
Time	Population Ramp style	0 <u>ku</u>
Points		
Time 60	60 Add point	nt
	Modify po	e
	EinishCan	cel

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

Enter 60 in the Time text field :	80
Enter 60 in the Population text field	Population 100
Choose the ramp style :	
To save these settings click on the A	Add point button : Add point

😑 Load profil	e	$\overline{\mathbf{X}}$
Add points fo	r this profile Heavy load	
Time	Population 100	
Force stop		
Points		
Time	Population Ramp sty	/le Add point
60	60	
180	100	
		Remove
0		<u>F</u> inish Cancel

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

Enter 60 in the Time text field	d :
Enter 60 in the Population te	xt field :
Choose the ramp style :	$\odot \checkmark \bigcirc {}_{w_{k}} \bigcirc {}_{w_{k}} \bigcirc {}_{w_{k}} \bigcirc$

To save these settings click on the Add point button : Add point

😂 Load prof	ile		$\overline{\mathbf{X}}$
Add points t	for this profile He	avy load	
Time	Pop	ulation	Ramp style
240			
Force stop			
Points			
Time	Population	Ramp style	Add point
60	60		
180	100	ho	Modify point
240	0	1	Remove
?			Einish Cancel

Click on the Finish button :

Einish



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

4. 4. Define your test plan

4.1. 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.

4.2. 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

	-+		
,project	New	•	📸 Project
	Go Into		15AC Plug-in Project
🖶 📂 MyLdapInjector	Open in New Window		💰 Clif testPlan 🕟

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

🕷 New CL	.IF Test Plan	
Create a This wizard extension th	new CLIF Test Plan creates a new CLIF Test Plan with *.ctp nat can be opened by a Test Plan editor.	CLIF
<u>C</u> ontainer: <u>F</u> ile name:	/LdapLoadTest LdapLoadTest_plan ctp	Browse
	Einish	Cancel

4.3. 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

Add

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

 Properti 	rties	
Manage inje	njector and probe properties	
Id* :	0	
Server* :	local host	Refresh
Role* :	probe	×
Class* :		
Arguments	ts :	
Comment :		

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 chapter 5.3 Running a registry, 5.4 Configuring a CLIF server, 5.5 Running a CLIF server).

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

For example :

```
ld : 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 :

ld : 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

pu _						6 Add
Id	Server	Role	Class	Arguments	Comment	Had
2	10.0.0.3	probe	cpu	1000 600	Cpu probe deployed on 10.0.0.3	Remov
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	Remove

We do the same thing with the jvm probe

ld : 3 Server : 10.0.0.2 Role : probe 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 :

Test Plan Ed	ditor			
Injectors and pro	bes			
All injectors and pro	bes in the test plan			
cpu jym				
Id Server	Role Class	Arguments	Comment	
3 10.0.0.2	probe jvm	1000 600	Jvm probe deployed on 10.0.0.	I.2 Remove
				Remove All

And finally with the injectors :

ld : 4 Server : 10.0.0.1 Role : injector Class : ISacRunner Arguments : LdapInjectorLoadTest_1.xis Injector deployed on 10.0.0.1

ld : 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 :

Test Plan Editor

Injectors and probes

All inj	jectors and pr	obes in the	e test plan		
cpu	jvm injector	·]			
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

Add
Remove
Remove All

4.4. 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 testplan. A new tabulation named "test" appears.

Click on it.

Test	Plan E	ditor			
1030	TUTL	arcor			
Injecto	ors and pr	obes			
All injec	tors and pr	obes in the t	est plan		
cpu jv	m injector]			Add
Id	Server	Role	Class	Arguments	
5	10.0.0.3	injector	IsacRunner	LdapInjectorLoadT	Remove
4	10.0.0.1	injector	IsacRunner	LdapInjectorLoadT	Remove All
Edit Tes	t)				

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

Deploy

And the global status of your testplan :

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 testplan you want to initialize :

🖟 Test id	X
Enter test id name :	
LdapLoadTest_plan	
	, ,
	OK Cancel

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

undeployed undeployed

State

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
- · one for each probe family

🐔 Monitor	×										- 8
LdapLoadTe	st_plan -	10 octob	ore 2007	Ohim5is 🖂							
memory cp	u injecto	or Alarm	IS								
Display	Collect	Blade	Time	Time : 0	Value : 0						
	✓	3	00	85							
				60	<u></u>	_ <u> </u>					_
				- 40							_
				40							
				20							
% used ra	m			~	L	48)	100	150	200	250	_
			_								
Drawing ti	meFrame	: 300		ec.Polling Per	od : 1	sec. Ref	resh Reset				

For each panel, the user may set the monitoring timeframe, the polling period, and start or stop the monitoring process. Moreover, a checkbox 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 ca	an sta	rt the inject	tion by clicking o	n th	ie stari	t but	tton : 🕒	Start		
And then arameters	you of	Suspend Parameters	can suspend the load test.	,	stop	or	Stop	change	dynamically	some

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 generate reports files. These text files are stored in :

/path_to_clif_directory/clif-1.2.2-console/plugins/org.objectweb.clif.console.plugin_1.2.2/report

5. CLIF server

5.1. Requirements

To see the technical requirements refer to chapter 1.2 Technical requirements

You need to download and unzip the <u>clif-server-1.2.2.zip</u> archive on the platforms on which you'll deploy the CLIF servers.

5.2. 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.

5.3. Running a registry

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.

a CLIF Console o	configuration	
Please configure This wizard creates th	e the Clif console he clif.props file.	CLIF
Hostname : Code server port :	10,194.3.95 - Broadcom NetXtreme 57xx Gigabit Controller - Deterministic Network Enha 1357	ncer Miniport 💌
Classpath :	clif-1.2.2-console/plugins/org.objectweb.clif.console.plugin_1.2.2/examples/classes/	
	Finish	Cancel

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 in and choose your project path :

	100		Properties
plan	Eolder Selection	×	Manage injector and probe prop
CLIF Console configure Please configure the C This wizard creates the clif.pu	Select folder to add to classpath.		
Hostname : 10.194 Code server port : 1357 Classpath :			ork Enhancer Miniport
		OK Cancel	
	1		Einish Cancel

And finally click on Finish to start the registry.

5.4. 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 :

```
Cifig 10.0.010 /clif-1.2.2] $ ant config
Buildfile: build.xml
ask-console-config:
    [input] please enter the host where the console will be run:
10.0.0.10
ask-codeserver-config:
    [input] please enter the port number for the console embedded code server:
1357
config:
    [copy] Copying 1 file to /home/clif/clif-1.2.2/etc
    [echo] /home/clif/clif-1.2.2/etc/clif.props has been generated.
BUILD SUCCESSFUL
Total time: 1 minute 29 seconds
[clif@ 10.0.10 clif-1.2.2] $ ]
```

5.5. 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



6. Appendix A : Install mandatory requirements

6.1. Download requirements

6.1.1. Sun J2SDK[™] 1.5 (1.5 version or greater is mandatory)

Click on the following link : <u>http://java.sun.com/javase/downloads/?intcmp=1281</u>

Click on the download button of the current version of the JDK :

JDK 6 Update 3 The Java SE Development Kit (JDK) includes the Java Runtime Environment (JRE) and command-line development tools that are useful for developing applets and applications.

Accept the licence agreement :

Required: You must accept the license agreement to download the product.
Accept License Agreement | Review License Agreement
Secline License Agreement

Select the platform where CLIF will be used :

Window	s Platform - Java(TM) SE Development Kit 6 Update 3		
≶8			
Downlo	ad the full version as a single file .		
	⊻ <u>Windows Offline Installation, Multi-language</u>	jdk-6u3-windows-i586-p.exe	65.64 MB
		jdk-6u3-windows-i586-p-iftw.exe	373.39 KB

or

Linux P	latform - Java(TM) SE Development Kit 6 Update 3		
≶8			
	Linux RPM in self-extracting file	jdk-6u3-linux-i586-rpm.bin	61.64 MB
	 Yw ▲ Linux self-extracting file 	jdk-6u3-linux-i586.bin	65.40 MB

and save the downloading file on your computer.

Once the download done, you can launch the installation on the environment you want to install it.

6.1.2. Apache ant utility version 1.5.4 or greater

Click on the following link : <u>http://ant.apache.org/bindownload.cgi</u>

Click on the current release of apache ant utility depending of the platform where CLIF will be used (For windows select the .zip file and for linux the .tar.gz file) :

Current Release of Ant
Currently, Apache Ant 1.7.0 is the best available version, see the <u>release</u> <u>notes</u> .
Note
Ant 1.7.0 has been released on 19-Dec-2006 and may not be available on all mirrors for a few days.
Tar files may require gnu tar to extract
Tar files in the distribution contain long file names, and may require gnu tar to do the extraction.

- .sip archive: apache-ant-1.7.0-bin.zip [PGP] [SHA1] [MD5]
- .ter.gs archive: apathe-ant-1.7.0-bin.tar.qz [PGP] [SHA1] [MD5]
- .ter.bs2 archive: apache-ant-1.7.0-bin.tar.bz2 [PGP] [SHA1] [MD5]

Then save the downloading file on your computer.

Once the download done, you can unzip the archive file where you want on your platform.

6.2. Environement variables setting

6.2.1. Windows OS

Now you have to set the following environement variable :

- JAVA_HOME=C:\Program Files\Java\jdk1.6.0_03
- ANT_HOME=C:\Program Files\apache-ant-1.7.0

And to modify :

• PATH=<value already in path>;%JAVA_HOME%\bin;%ANT_HOME%\bin

Go to :

Start > Settings > Control Panel



Double click on the system icon :



Choose advanced tab and click on the environment Variables button :

System Proper	ties			? 🔀
System Re:	store	Automa	atic Updates	Remote
General	Compu	uter Name	Hardware	Advanced
You must be lo	ogged on as	s an Administra	tor to make mos	t of these changes.
Visual effects	, processor	scheduling, m	emory usage, ar	nd virtual memory
				<u>S</u> ettings
User Profiles				
Desktop setti	ings related	to your logon		
				S <u>e</u> ttings
Startup and F	Recovery-			
System startu	ip, system f	ailure, and deb	ugging informati	on
				Settings
	Er	iviro <u>n</u> ment Vari	iables E	ror Reporting
		ОК	Car	ncel Apply

nvironment Varia	bles 🤶
User variables for ta	we8231
Variable	Value
	<u>N</u> ew <u>E</u> dit <u>D</u> elete
System variables	
Variable	Value
NUMBER_OF_P	1 Windows NT
Path	c:\orawin\bin;C:\oracle\ora81\bin\;C:\P
PROCESSOR_A	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;
	New Edit Delete

Now click on the New button of the System variables parameters group :

Then enter the variable name and its value :

New System Var	iable 🔹 🔀
Variable <u>n</u> ame:	JAVA_HOME
Variable <u>v</u> alue:	C:\Program Files\Java\jdk1.6.0_03
	OK Cancel

Do the same thing for ANT_HOME variable :

Edit System Varia	ble 🔹 💽
Variable <u>n</u> ame:	ANT_HOME
Variable <u>v</u> alue:	C:\Program Files\apache-ant-1.7.0
	OK Cancel

Now modify the PATH variable. Select the PATH variable and click on the Edit button :

Variable	Value
OS	Windows_NT
PATH	c:\orawin\bin;C:\oracle\ora81\bin\;C:\P
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;
PROCESSOR_A	x86
PROCESSOR_ID	x86 Family 6 Model 13 Stepping 8, Genu 💌

Now add the reference to the ANT_HOME\bin and JAVA_HOME\bin repertory at the end of the PATH line :

Edit System Variable 🔹 👔 🔀	
Variable <u>n</u> ame:	РАТН
Variable <u>v</u> alue:	;;%ANT_HOME%\bin;%JAVA_HOME%\bir ¹
	OK Cancel

Now you just have to check if the good java version and ant version are used by your system.

Type the name of a program, folder, document, or internet resource, and Windows will open it for you.
OK Cancel <u>B</u> rowse

i i

C:\WINDOWS\system32\cmd.exe	- 🗆 🗙
Microsoft Windows XP [Version 5.1.2600] <c> Copyright 1985-2001 Microsoft Corp.</c>	
H:\>ant -version Apache Ant version 1.7.0 compiled on December 13 2006 H:\>java -version java version "1.6.0_03" Java(TM> SE Runtime Environment (build 1.6.0_03-b05) Java HotSpot(TM> Client VM (build 1.6.0_03-b05, mixed mode)	-
6.2.2. Linux OS

Now you have to set the following environement variable :

- JAVA_HOME=/usr/lib/jdk1.6.0_03
- ANT_HOME=/usr/lib/apache-ant-1.7.0

And to modify :

• PATH=\$PATH:\$HOME/bin:\$ANT_HOME/bin:\$JAVA_HOME/bin

Go to the root directory of your user account.

Modify the .bash_profile file with the following command line : vi .bash_profile

If this file doesn't exist you can create it.

Put in it the following line :

```
# .bash_profile
```

Get the aliases and functions

if [-f ~/.bashrc]; then

. ~/.bashrc

fi

User specific environment and startup programs

ANT_HOME=/usr/lib/apache-ant-1.7.0

JAVA_HOME=/usr/lib/jdk1.6.0_03

PATH=\$PATH:\$HOME/bin:\$ANT_HOME/bin:\$JAVA_HOME/bin

export ANT_HOME export JAVA_HOME export PATH

unset USERNAME

Now you have to logg off from your platform and then logg on to reload the .bash_profile file. Now you just have to check if the good java version and ant version are used by your system. -How To develop and use CLIF ISAC plug-ins

[clif@ ~]\$ ant -version Apache Ant version 1.7.0 compiled on December 13 2006 [clif@ ~]\$ java -version java version "1.6.0_02" Java (TM) SE Runtime Environment (build 1.6.0_02-b05) Java HotSpot(TM) Server VM (build 1.6.0_02-b05, mixed mode) [clif@ ~]\$

6.2.3. Mac OS X

[TODO]