DIET

USER'S FAQ

VERSION	2.7
DATE	September 2010
PROJECT MANAGER	Frédéric DESPREZ.
EDITORIAL STAFF	Yves CANIOU, Eddy CARON and David LOUREIRO.
AUTHORS STAFF	Eugène PAMBA CAPO-CHICHI.
Copyright	INRIA, ENS-Lyon, UCBL

Contents

The purpose of this document is to give answers to several simple questions in order to facilitate the familiarization with DIET for new users (newbies).

DIET: What does it mean?
DIET is an acronym of Distributed Interactive Engineering Toolbox.

What is the purpose of DIET?
The aim of the DIET project is to develop a set of tools to build computational servers. The Distributed Interactive Engineering Toolbox (DIET) project is focused on the development of scalable middleware with initial efforts focused on distributing the scheduling problem across multiple agents. It consists of a set of elements that can be used together to build applications using the Grid-RPC paradigm. This middleware is able to find an appropriate server according to the information given in the client's request.

How to get/download DIET?
DIET is available at: `http://graal.ens-lyon.fr/DIET`.

However, the graal members can directly download it from the Graal server using a CVS command. Before connecting to the Graal server, the new graal member has to generate a couple of private/public keys by executing the ssh-keygen command.
The generated public key (and localized on the .ssh directory) has to be sent to the graal administrator: eddy.caron@ens-lyon.fr.

How to install DIET?
Before installing DIET, the user needs to install other essential packages which are:

- OMNIORB4 available on `http://www.yolinux.com/TUTORIALS/CORBA.html` and,
- ccmake on `http://www.cmake.org/cmake/help/install.html`.

After that DIET can be installed. For more information about installation, the user manual is available on the directory: diet-2.7/doc/UsersManual (Chapter II). The installation is also well described on the README in diet-2.7/Cmake.

How to configure OMNIORB4?
Set environment variables:

```
export OMNIORB4_DIR="OMNIORB4_install_directory"
export OMNIORB4_INCLUDE_DIR=$OMNIORB4_DIR/include

export OMNINAMES_LOGDIR="/var/omninames" for root
export OMNINAMES_LOGDIR="/tmp" for non root user
(For OMNINAMES_LOGDIR, the previous directories can be changed according to the user permis

export PATH=$OMNIORB4_DIR/bin:$PATH
export OMNIORB_CONFIG=$OMNIORB4_DIR/etc/omniORB4.cfg
```

The OMNIORB4_install_directory is the path of the directory where OMNIORB4 is installed. It is also essential to export the omniORB4.cfg file:

```
#*****************************************************************************#
# (*) Reference to the CORBA Name server, to which all agents register and
#     connect to get references on other agent
#*****************************************************************************#
InitRef = NameService=corbaname::127.0.0.1:2809
supportBootstrapAgent = 1
```

What is a simple way to better understand how DIET works?
The scalars example (on diet-2.7/dietbin/src/examples/scalars) is a very simple and interesting client/server example to understand how DIET works.

How is the scalars example run?
The first step is to run the CORBA naming service:
If you are launching CORBA for the first time, type:
> omniNames -start
Otherwise you can directly launch
> omniNames

This error can occur if use use the start option whereas you have already used the naming service:

```
root@capo-chichi:/home/paco/Desktop/DIET/FAQ# omniNames -start
Sat Sep 11 20:27:23 2010:
Error: log file '/var/omninames/omninames-capo-chichi.log' exists.  Can't use -start option
```

There exist two solutions to solve it:

1. by executing : > omniNames,

   ```
   root@capo-chichi:/home/paco/Desktop/DIET/FAQ# omniNames
   Sat Sep 11 20:27:26 2010:
   Read log file successfully.
   Root context is IOR:010000002b00000049444c3a6f6d672e6f72672f436f734e616d696e672f4e616d
   ```

```
Checkpointing Phase 1: Prepare.
Checkpointing Phase 2: Commit.
Checkpointing completed.
```

2. by erasing the log files generated on OMNINAMES log directory and restarting the command omniNames.

The second step is to run an agent by using the dietAgent command (diet-2.7/dietbin/src/examples/scalars):
>dietAgent MA.cfg
The last step is the execution of server and client programs:
> ./server[1] Sed.cfg
>./client[2] client.cfg CADD where CADD is one of the service offer by the Sed.

```
/* This server offers all additions of two scalars:                  */
/*   - CADD = sum of two chars                                        */
/*   - BADD = sum of two bytes                                        */
/*   - IADD = sum of two ints                                         */
/*   - LADD = sum of two longs                                        */
/*   - FADD = sum of two floats                                       */
/*   - DADD = sum of two doubles                                      */
```

This is one example of the 3 configurations files MA.cfg, SeD.cfg and client.cfg:

```
MA.cfg

/***MA configuration file****/
traceLevel = 1
agentType = DIET_MASTER_AGENT
name = MA0

SeD.cfg

/***SeD configuration file****/
traceLevel = 10
parentName = MA0

client.cfg

/***Client configuration file****/
traceLevel = 10
MAName = MA0
```

For a better understanding of this example, the code sources of these programs (client.c and server.c) are available on diet-2.7/src/examples/scalars.

[1]Depending on the compilation, the programm may be called scalars_server
[2]Depending on the compilation, the programm may be called scalars_client

What is GoDiet?

GoDiet is a Java-based tool for automatic Diet deployment that manages configuration file creation, staging of files, launch of elements, monitoring and reporting on launch success, and process cleanup when the Diet deployment is no longer needed.

How to get/download GoDiet?

Graal members have to simply do the following:

- export CVSROOT=":ext:login@graal.ens-lyon.fr:/home/CVS/graal"
- cvs co -d GoDiet_CVS GRAAL/devel/diet/diet-contrib/GoDIET

How to install GoDiet?

The installation of GoDiet needs to check other essential pakages such as: ant, libgcj and opensdk. GoDiet uses ssh localhost connection for running that is why it is important to check it:

```
paco@capo-chichi:~$ ssh localhost
ssh: connect to host localhost port 22: Connection refused
```

To solve it, the user has to do:

```
paco@capo-chichi:~$ sudo apt-get install ssh
[sudo] password for paco:
Lecture des listes de paquets... Fait
Construction de l'arbre des dependances
Lecture des informations d'tat... Fait
Les paquets supplmentaires suivants seront installes:
  openssh-server
Paquets suggeres:
  rssh molly-guard openssh-blacklist openssh-blacklist-extra
Les NOUVEAUX paquets suivants seront installes:
  openssh-server ssh
```

And checking:

```
paco@capo-chichi:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
RSA key fingerprint is 4f:0f:e3:b3:55:d8:91:e4:81:1c:fc:2c:bf:34:da:72.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
paco@localhost's password:
Linux capo-chichi 2.6.32-24-generic #42-Ubuntu SMP Fri Aug 20 14:24:04 UTC 2010 i686 GNU/Li
Ubuntu 10.04.1 LTS
```

After checking all things described previously, GoDIET-2.3.0.jar is generated by the command ant:

```
paco@capo-chichi:~/Desktop/DIET/Godiet/GoDiet_CVS$ ant
Buildfile: build.xml
init:
idlj:
     [echo]  Generating stubs, helper classes and ties from IDL
     [echo]  Generating for tasks ...

package:

idlj_diet:
     [echo]  Generaxting stubs, helper classes and ties from
     [echo]     IDL
                 .
                 .
                 .
```

How do I run GoDiet?

The command for running GoDiet is: java -jar GoDIET-2.3.0.jar your_xml_file

```
paco@capo-chichi:~/DIET/Godiet/GoDiet_CVS$ java -jar GoDIET-2.3.0.jar examples/example1.xml
Parsing xml file: examples/example1.xml
```

On the example1.xml file, you have to ckeck local paths (with the hdail login), the PATH and LD_LIBRARY_PATH:

```
<env>
    <var name="PATH" value="/home/paco/DIET/diet-\dietversion/dietbin/src/examples/scalars:$
    <var name="LD_LIBRARY_PATH" value="$LD_LIBRARY_PATH"/>
</env>
```

After that, use the *launch* command to start the example. For more, you have to follow instructions on the DIET UsersManual (Chapter 9, p. 91).

What does mean the error message: "DIET ERROR: caught a CORBA exception (TRANSIENT) while submitting problem."?

The reason is probably that you are using a different version of DIET on the client side and on the server side (Master Agent or SeD). For example this happens if you use a DIET v2.6 client with a DIET v2.4 MA.

I run omniNames, the configuration file seems OK, but the MA tells me it cannot contact the naming service. What's wrong?

Check your omniORB configuration file. If the host is specified with the hostname, try to put the IP adress instead.

I get a TRANSIENT_ConnectionFailed error. What's wrong?

Check your omniORB configuration file. If necessary, when running omniNames add the endPointPublish option with the correct listening adress and port.

Compiling DIET statically on mac OS platforms.

If you plan to compile DIET statically on mac platforms. . . don't. You may face such an error:

```
/usr/libexec/gcc/i686-apple-darwin8/4.0.1/ld: can't locate file for: -lcrt0.o
collect2: ld returned 1 exit status
make[2]: *** [src/agent/dietAgent] Error 1
make[1]: *** [src/agent/CMakeFiles/dietAgent.dir/all] Error 2
make: *** [all] Error 2
```

or something like describe here: http://lists.apple.com/archives/xcode-users/2007/Apr/msg00327.html.

Here is the explanation of the problem: http://developer.apple.com/qa/qa2001/qa1118.html.

It seems that one can create static libraries but can't create static executables.

Please consider creating dynamic executable instead!

You should also be aware that since DIET 2.3 static compilation is disabled on mac platforms.