Overview

You will find here the recipe for installing the ETICS client.

Installation instructions

On Linux and Mac OS-X

Here are the instructions for installing the ETICS client (if the ETICS_HOME environment variable is set the installation will be executed in the directory pointed at by ETICS_HOME and not in the current directory. We recommend to unset ETICS_HOME for a first installation):

1. Prerequisites (these packages are not installed by the ETICS installation script and have to be present before running it): python-devel, openssl, openssl-devel, flex, gcc
3. python etics-client-setup

Executing this command will fetch and install the ETICS client in the current directory, as well as required dependencies. The client will be installed by default in the directory 'etics' in the current directory. Alternatively the client can be installed in a different directory by using the --prefix option of the script. Finally set the following environment variables:

1. export ETICS_HOME=<etics installation location>/etics (i.e. the etics directory from which you've run etics-client-setup)
2. export PATH=$ETICS_HOME/bin:$PATH

(Use the appropriate syntax for your shell, you may also want to set the variables in your login script). Once the client is installed and configured, you can create a workspace by running the etics-workspace-setup command in a directory of your choice prior to calling any other ETICS commands. This operation can be repeated to create as many workspaces as required.

Voila you're good to go! All ETICS commands start with etics-*, and you can use the --help or -h option to view the possible options.

Frequently Asked Questions

If you have questions, read the ClientFAQ first!

Quick tour

To get you started and also test your installation, here's a quick tour of the build and test commands available on the client. For the purpose of this quick tour, we'll assume that we're working under CVS, but don't worry, we can also use other types of Version Control Systems.

We'll use to start with a dummy project called myproject. This project is actually used for unit testing the client (remember, on ETICS we eat our own dog food ;-). This project is actually virtual (dummy) and doesn't correspond to any code in CVS, but it's good enough to start with.

Workspace setup

Make sure you setup your workspace using the etics-workspace-setup command. See the installation
instructions above for details.

**List project**

Once you've setup your workspace, you can use all the ETICS command. To start with, you might want to list the existing projects you can work with. This can be done using the `etics-list-project` command:

```javascript
> etics-list-project
```

You should get the following:

Get a list of projects

- org.etics
- externals
- org.glite
- myproject
- org.glite.test
- quattor
- org.diligentproject

Done!

These project can then be used with the `etics-get-project` command, as shown next.

**List platform**

Once you've setup your workspace, you can use all the ETICS command. To start with, you might want to list the existing platforms you can work with. This can be done using the `etics-list-platform` command:

```javascript
> etics-list-platform
```

You should get the following:

Get a list of platforms

- slc4_x86-64_gcc345
- default
- slc3_ia32_gcc323
- slc4_ia32_gcc345
- other
- windows

The `default` platform definition is a catch-all cross-platform entry representing all platforms. It is used when more specific definitions do not exist. These platforms can then be used with the `etics-remote-build` and `etics-remote-test` commands, as shown next.

**Get project**

Once you know which project you want to work with, you can set it up in your workspace using the `etics-get-project` command. In our example here, we'll work with `myproject`:

```javascript
> etics-get-project myproject
```

You should get the following:

Downloading the project 'myproject'...

Done!
Here the client will contact the ETICS server and get the metadata describing the layout of your project (e.g. project/subsystem/component relationships), and save it on your workspace.

Note: You can alternatively specify a project name to the `etics-workspace-setup` command, which would save you having to call `etics-get-project`.

**List configuration**

The next step, in order to be able to build something, is to specify the concrete instance you want to work with, which in turns corresponds to a CVS tag. By default, unless you specify it otherwise, the client will fetch the tag `HEAD` of the project configuration called `<project-name>.HEAD` (where `<project-name>` is the name of your project, as specified while calling `etics-get-project`). If you don't know the exact configuration name you want to checkout, you can list existing configurations. Here's an example, where we don't provide a module name, which by default will use the current project name:

>`etics-list-configuration`

You should get the following:

```
Loading workspace definition...Reparsing workspace definition... Done.
Done.
The following configurations are defined for module 'myproject':
  myconfig
  myproject.HEAD
```

**Checkout**

You can now use the checkout command, with the `--c <configuration-name> <module-name>` option, if you want to checkout a specific configuration, corresponding to a specific module. In the following example, we checkout the default configuration:

>`etics-checkout`

You should get the following:

```
Loading workspace definition...Reparsing workspace definition... Done.
Done.
Source preference is from source code (--fromsource this is the default)
Module not specified, using default project: 'myproject'
Configuration not specified, using default configuration: 'myproject.HEAD'
Downloading the configuration 'myproject.HEAD' of module 'myproject'
Merging project and configuration information... Done.
Saving data to workspace... Done.
Loading workspace definition...Reparsing workspace definition... Done.
Done.
Checking out configuration 'anothersubsys.acompconf'
  [checkout]: echo checkout
checkout
configuration 'anothersubsys.acompconf' is missing vcs commands for platform 'default'

Contacting ETICS Server.Done
configuration 'anothersubsysconf' is missing vcs commands for platform 'default'
configuration 'asubsys.acompconf' is missing vcs commands for platform 'default'
configuration 'asubsysconf' is missing vcs commands for platform 'default'
configuration 'anothercompconf' is missing vcs commands for platform 'default'
configuration 'anothersubsys.anothercompconf' is missing vcs commands for platform 'default'
```

Get project
configuration 'myproject.HEAD' is missing vcs commands for platform 'default'

Show configuration structure

Once your project is in place, you can visualise its structure using the following command:

> etics-show-configuration-structure

You should get the following:

> etics-build

You should get the following:

Building: anothersubsys.acompconf
Nothing to do for anothersubsys.acompconf
Reparsing workspace definition... Done.

Building: anothersubsys.anothercompconf
Nothing to do for anothersubsys.anothercompconf

Building: anothersubsysconf
Nothing to do for anothersubsysconf

Reparsing workspace definition... Done.

Building: asubsys.acompconf
  [init]: echo in init
  in init

  [checkstyle]: echo in checkstyle
  in checkstyle

  [compile]: echo in compile
  in compile

  [test]: echo in test
  in test

  [publish]: echo in publish
  in publish

Building: asubsysconf
Nothing to do for asubsysconf

Building: anothercompconf
  [init]: echo in init
  in init

  [checkstyle]: echo in checkstyle
  in checkstyle

  [compile]: echo in compile
  in compile

  [test]: echo in test
  in test

  [publish]: echo in publish
  in publish

Building: acompconf
  [init]: echo in init
  in init

  [checkstyle]: echo in checkstyle
  in checkstyle

  [compile]: echo in compile
  in compile

  [test]: echo in test
  in test

  [publish]: echo in publish
  in publish

Building: myproject.HEAD
Nothing to do for myproject.HEAD
Done!

**Test**

The `etics-build` allows you to execute a local build, while the `etics-test` command allows you to execute a test. The build command includes a `test` target which is intended for unit test. The `etics-test` is ment for
executing functional and/or system testsuites (e.g. stress tests, performance tests, functionality tests, deployment tests). Here is how to use the command:

```bash
> etics-test
```

You should get the following:

```
Loading workspace definition... Done.
Executing test: anothersubsys.acompconf
Nothing to do for anothersubsys.acompconf
Executing test: anothersubsys.anothercompconf
Nothing to do for anothersubsys.anothercompconf
Executing test: anothersubsysconf
Nothing to do for anothersubsysconf
Executing test: asubsys.acompconf
Nothing to do for asubsys.acompconf
Executing test: asubsysonfig
Nothing to do for asubsysonfig
Executing test: anothercompconf
Nothing to do for anothercompconf
Executing test: acompconf
Nothing to do for acompconf
Executing test: myproject.HEAD
Nothing to do for myproject.HEAD
Done!
```

-- MebSter - 12 Jun 2006

### Remote build/test

You can execute remote build and tests using these two commands:

```bash
> etics-remote-build > etics-remote-test
```

The remote commands permit you to build/test on remote machines with a lot of available platforms. With the option `--p < platforms-list >` you can specify the platforms where you want to build/test.

You should get the following:

```
Submission IDs:

    (Platform: slc3_ia32_gcc323, ID: tomcat4_lxb1119.cern.ch_1150103734_10792)
```

-- Main.mselmi - 12 Jun 2006

You now know the basics of building with ETICS!

-- MebSter - 12 Jun 2006
Comments

What other Version Control Systems are supported. You should specify, fully implemented and planned.

-- LaurenceField - 28 Mar 2006

I've now updated the instruction with a sub-section on etics-list-project.

-- MebSter - 02 Jun 2006

Temporarily removed the Windows installation instructions

-- MebSter - 08 May 2007