Open-PEOPLE: Development Platform

Kévin Roussel, Olivier Zendra

Open-PEOPLE

Open Power and Energy Optimization Platform and Estimator

Deliverable 2.9

Development Platform

	Document Manager	Contributors	Checked by
Name	Kévin ROUSSEL	INRIA Nancy (CRI-NGE)	Olivier ZENDRA
Contact	Kevin.Roussel@inria.fr		
Date	09-sep-2011 **(version 1.0)**		
Summary	This document describes the various, integrated development tools provided by the Open-PEOPLE project to its software developers.		

Table of contents

1. Preface

1.1. Versions

Version	*Date*	*Description & rationale of modifications*	*Sections mainly modified*
0.9	07 September 2011	First version	–
1.0	09 September 2011	Corrected some typos	4

1.2. Table of references and applicable documents

Reference	*Title & edition*	*Author(s)*	*Date*

1.3. Acronyms and glossary

Term	*Definition*
OP	**Open-PEOPLE:** the name of the project we're talking about, and by extension, the name of the platform(s) developed within this project.
OPSWP	**Open-PEOPLE's software platform:** the central piece of code, around which resolve all of the software developments undertaken within the OP project; it is one of the two main realisations of this project.
OPHWP	**Open-PEOPLE's hardware platform:** automated benchmark platform, allowing for measurement of power consumptions on various systems (e.g. evaluation boards); it is one of the two main realisations of this project.
TCP	**Transmission Control Protocol:** one of the core protocols used on the Internet, conceived to ensure reliable, ordered delivery of byte streams at the transport level (level-4 layer). TCP allows multiplexing of byte streams related to many applications, by using unsigned 16-bit port numbers. Many of these port numbers are reserved for specific applications and protocols, like 22 for SSH, 25 for SMTP, 80 for HTTP, etc.
HTTP	**HyperText Transfer Protocol:** networking applicative (level-5) protocol. Initially conceived for distributive hypermedia information systems (the “Web”), it is now used for a much wider range of applications, making it one of the most widely used applicative protocol for data exchange on the Internet.
HTTPS	**HyperText Transfer Protocol with SSL/TLS:** secured version of the HTTP protocol, where all communications are crypted using the SSL (Secure Sockets Layer) or TLS (Transport Layer Security) cryptographic protocol, which also ensures the authentication of HTTP servers

Term	***Definition***
DMZ	**De-Militarized Zone:** a (sub)network that contains and exposes an organization's external services to another untrusted network, that is the Internet. Defined in opposition the organization's internal, trusted Local Area Network (LAN).
RAID	**Redundant Array of Independent Disks:** technology providing increased storage functions and reliability through redundancy. The RAID level 1 focuses on increased reliability by mirrorring disks.
FCGI	**Fast Common Gateway Interface:** a protocol for interfacing external applications to web servers. FastCGI tries to improve on the earlier CGI protocol by reducing the overhead associated with the web–application interface, thus improving general performance.
CRI-NGE	**Centre de Recherche INRIA – Nancy Grand-Est:** the place where the development of OPSWP is managed and—for a major part—realized.
Lab-STICC	**Laboratoire en Sciences et Techniques de l'Information, de la Communication et de la Connaissance:** the place where the development of OP hardware platform is realized, and where many contributions to OPSWP come from.
RCP	**Rich Client Platform:** a framework (comprising widgets, desktop/workbench, extensible architecture, update management, and other paradigms) on which one can build coherent, robust, standardized, and feature-rich desktop stand-alone and client applications (thus named “rich client” applications). The OPSWP is to be built on Eclipse project's RCP.
IDE	**Integrated Development Environment:** a graphical application/framework gathering tools in order to help developers and programmers to perform more easily and efficiently their work; one of the most advanced, best known and most used IDE for the Java platform is the Eclipse JDT, built upon the RCP produced by the Eclipse project. A variant of this JDT is precisely specialized in development of applications based on Eclipse's RCP.

2. Executive summary

As it is stated in its name, the platform we wish to develop is **open**. This means that anyone with reasonable skills in computer science and systems modeling should be able to expand and tailor the features of Open-PEOPLE's software platform (OPSWP) to his needs.

To this end, we provide a complete set of development tools for the OP project's developers community: a source code repository, a bug tracker (ticket management system), a lightweight development wiki, a.nd a continuous integration server. All these tools are hosted on a dedicated development server.

The purpose of this document is to describe the installation and configuration of this development server, as well as the way our developers can make good use of it.

3. Scope of the document

This document is a ***technical*** and an ***administrative reference***: it shall describe with all needed precision what development services are provided to the OP developers community, and how they are installed and configured. This document shall be ***self-contained,*** so that no other complementary data is needed to make good use of it: it will thus contain detailed listing of configuration files, as well as administrative and “good-use” procedures.

On the other hand, it is supposed that the reader of the present document knows what OP is, and what goals it is supposed to achieve. This document won't talk about these subjects: these informations can be found in the D1.1 to D1.4 deliverables of the project.

Moreover, this document focuses itself exclusively on the services provided for ***software development***. All data concerning the project's hardware development (on the OP hardware platform) is located in other deliverables (deliverables from task 3; plus D2.3 for the remote control of hardware platform).

4. Configuration of the Development Server

Our development server is physically a rack-mounted computer, whose hardware reference (i.e. brand and model) is: Dell PowerEdge R610.

This server is powered by two x86-64-bit quadri-core processors (Intel Xeon L5520), and possesses 8 gigabytes of RAM.

It possesses 900 gigabytes of available disk space (the hard drives being mounted in a RAID 1 – mirrorring – array).

This development server is installed inside a dedicated DMZ. For evident security reasons, it cannot connect to the other computers that are installed in CRI-NGE internal network.

This machine, named “vulcan” works with the Linux Debian 6.0 (“Squeeze”) operating system, in its amd64 version. It runs an OpenSSH server, an Apache httpd server powering the Subversion (mod_svn) and Trac (Python FCGI) applications, and the Jenkins continuous integration server (embedding its own JEE server: Winstone).

These services represent the main software packages installed and running on the “vulcan” system. One can also note that the Debian Squeeze OS integrates a DNS multicast daemon (“avahi”).

4.1. Network interfaces and firewall

The machine is connected to the CRI-NGE DMZ – and thus to the Internet – through its primary Ethernet interface. Its IP address is statically configured through the standard **`/etc/network/interfaces`** file, whose content is shown hereafter:

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
  address 10.10.10.32
  netmask 255.255.0.0
  gateway 10.10.10.1
  pre-up /etc/rc.firewall-vulcan
  post-up route add default gw 10.10.10.1
  post-down route del default gw 10.10.10.1

# secondary interface, useful for troubleshooting...
iface eth1 inet dhcp
```

NOTE: for security reasons, the real IP addresses have been replaced by dummy, unexploitable addresses in all the listings presented in this document.

This configuration file calls, every time the primary Ethernet interface is activated, the **`/etc/rc.firewall-vulcan`** script file, whose role is to configure the OS integrated IP-filtering (firewall) features.

For security reasons, we won't reproduce the content of this file in the present document; we can however signal that our configuration forbids any incoming connection except on TCP ports 22 (SSH), 80 (HTTP), 443, and 8443 (HTTPS) on the primary Ethernet interface (“white list” mechanism).

The SSH connection is used for system administration and maintenance tasks; all the development services are provided through the HTTPS (SSL-secured) protocol, any call via the (non-secured) HTTP protocol being automatically redirected adequately (see the Apache configuration section below).

4.2. System startup

A compatibilty issue between the Linux Debian OS and the Dell hardware (more precisely the PERC6/i RAID controller integrated into our server) compelled us to impose a 9-second delay to the Linux kernel, during its boot sequence, in order to let the RAID controller detect and mount correctly our RAID-1 disk partitions.

Thus, the following part of the **`/boot/grub/menu.lst`** has been modified:

```
### BEGIN AUTOMAGIC KERNELS LIST
## lines between the AUTOMAGIC KERNELS LIST markers will be modified
## by the debian update-grub script except for the default options below

## DO NOT UNCOMMENT THEM, Just edit them to your needs

## ## Start Default Options ##
## default kernel options
## default kernel options for automagic boot options
## If you want special options for specific kernels use kopt_x_y_z
## where x.y.z is kernel version. Minor versions can be omitted.
## e.g. kopt=root=/dev/hda1 ro
##      kopt_2_6_8=root=/dev/hdc1 ro
##      kopt_2_6_8_2_686=root=/dev/hdc2 ro
# kopt=root=/dev/mapper/vg--vulcan-lv--vulcan--root ro rootdelay=9

## default grub root device
## e.g. groot=(hd0,0)
# groot=(hd0,0)
```

The `'rootdelay=9'` option has been added to the line beginning with `'# kopt='`.

Thanks to that approach, every time the `update-grub` command is called (more especially: every time the kernel and base system is updated via the `apt-get [dist-]upgrade` command), this option is systematically added to every entry of the system boot menu.

Apart from this specificity, the startup sequence of our system is completely standard.

4.3. Apache HTTP server

Its main purpose is to act as a “container” for Subversion and Trac (see the next two sections).

It has been installed “the Debian way”, simply by using the package offered by the Debian-stable distribution (`apache2 2.2.16-6+squeeze`).

We activated many of the standard modules bundled with this version of Apache, more specifically: `mod_dav`, `mod_dav_svn` (both necessary for Subversion), `mod_fcgid` (needed by Trac), `mod_auth_digest` (authentication of users using MD5 digests – this allows us to use a password file common to all of our development tools, see the Jenkins section below), `mod_rewrite` and `mod_ssl`.

These services are provided in a secure way (HTTPS protocol) on TCP port 443, by using the following `/etc/apache2/sites-available/default-ssl` configuration file:

```
ServerTokens Prod

<IfModule mod_ssl.c>
<VirtualHost _default_:443>
      ServerAdmin admin-dev-Open-PEOPLE@loria.fr

      DocumentRoot /srv/trac/htdocs
      # Le dossier "/srv/trac/htdocs" est vide...

      RewriteEngine on
      RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK)
      RewriteRule .*    -     [F]

      # Point d'entrée vers notre site Trac
      ScriptAlias / /srv/trac/cgi-bin/trac.fcgi/
      <Location "/">
            AuthType Digest
            AuthName "OP-devtools"
            AuthDigestProvider file
            AuthUserFile /srv/auth/htdigests
      </Location>

      # Point d'entrée vers le dépôt svn (accessible par WebDAV)
      <Location /svn>
            DAV svn
            SVNPath /srv/svn/depotOpenPEOPLE

            AuthType Digest
            AuthName "OP-devtools"
            AuthDigestProvider file
            AuthUserFile /srv/auth/htdigests
            AuthzSVNAccessFile /srv/svn/depotOpenPEOPLE/conf/authz
            Require valid-user
      </Location>
      ErrorLog /var/log/apache2/error.log

      LogLevel warn

      CustomLog /var/log/apache2/ssl_access.log combined


      SSLEngine on
      SSLCertificateFile    /etc/apache2/dev.open-people.fr.crt
      SSLCertificateKeyFile /etc/apache2/dev.open-people.fr.key

      <FilesMatch "\.(cgi|shtml|phtml|php)$">
            SSLOptions +StdEnvVars
      </FilesMatch>
      <Directory /usr/lib/cgi-bin>
            SSLOptions +StdEnvVars
      </Directory>

      BrowserMatch ".*MSIE.*" \
            nokeepalive ssl-unclean-shutdown \
            downgrade-1.0 force-response-1.0
</VirtualHost>
</IfModule>
```

We also open the standard HTTP port 80, only to allow for redirection of any query on the HTTPS port 443, thanks to `mod_rewrite`; we do this using the following `/etc/apache2/sites-available/default-ssl` configuration file:

```
ServerTokens Prod

<VirtualHost *:80>
      ServerAdmin admin-dev-Open-PEOPLE@loria.fr

      DocumentRoot /srv/trac/htdocs
      # Le dossier "/srv/trac/htdocs" est vide...

      <IfModule mod_rewrite.c>
            Options +FollowSymLinks
            Options +Indexes
            RewriteEngine On
            RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK)
            RewriteRule .*    -     [F]
            RewriteCond %{SERVER_PORT}    ^80$
            RewriteRule ^/hudson    https://dev.open-people.fr:8443/   [R,L]
            RewriteRule ^/svn       https://dev.open-people.fr/svn/    [R,L]
            RewriteRule ^.*$        https://dev.open-people.fr/        [R,L]
      </IfModule>

      ErrorLog /var/log/apache2/error.log

      LogLevel warn

      CustomLog /var/log/apache2/access.log combined

</VirtualHost>
```

Finally, we use the following `/etc/apache2/ports.conf` configuration file:

```
NameVirtualHost *:80
Listen 80

<IfModule mod_ssl.c>
    # SSL name based virtual hosts are not yet supported, therefore no
    # NameVirtualHost statement here
    Listen 443
</IfModule>
```

4.4. Subversion (SVN) source code repository

SVN is accessed via the Apache server, through the WebDAV protocol.

It has also been installed "the Debian way", simply by using the package offered by the Debian-stable distribution (`subversion 1.6.12dfsg`).

The Open-PEOPLE source repository is installed on the biggest partition available on the system (in `/srv/svn/depotOpenPEOPLE/`). Since its installation depends on Apache, the whole repository is owned by the '`www-data`' system user and its corresponding group.

To allow for a fine-grained tuning of users permissions on this repository, an AuthZ-type file has been written for use by our SVN service (see Apache configuration in the previous section). This file, named `/srv/svn/depotOpenPEOPLE/conf/authz`, has the following content:

```
[groups]
secured = user1, user2, user3
# and so on ...

[/]
@secured = rw
* =
```

NOTE: for evident security reasons, the real user logins have been replaced by dummy, unexploitable names ('usernn'...) in the previous listing.

4.5. Trac: integrated multi-function development tool

Trac is accessed via the Apache server; it is implemented as a FCGI (Fast Common Gateway Interface) application.

It has also been installed "the Debian way", simply by using the package offered by the Debian-stable distribution (`trac 0.11.7`).

The Open-PEOPLE Trac environment is installed on the biggest partition available on the system (in `/srv/trac/`). Since its installation depends on Apache, this whole environment is owned by the '`www-data`' system user and its corresponding group.

In order to tailor the feature set of Trac according to our needs, we installed different publicly available plug-ins into its environment, and especially the following:

- **IniAdmin (version 0.2):** this plugin allows Trac to be administrated directly from its web interface, rather than needing direct modifications into its configuration files
- **TracAccountManager (version 0.2.1):** this plugin allows user account management to be done directly from Trac web interface, the used user account database being common to all of our development tools – it is implemented as the `/srv/auth/htdigests` file, as shown in the Apache configuration section above
- **PeerReviewPlugin (version 2.0.dev):** this plugin adds source code "peer-reviewing" features to Trac

We also installed a custom Python source file on the Trac environment; this little Python program – `authz_policy.py` – allows us to define fine-grained user permissions for the different features offered by Trac:

```
# -*- coding: utf-8 -*-
#
# Copyright (C) 2007-2009 Edgewall Software
# Copyright (C) 2007 Alec Thomas <alec@swapoff.org>
# All rights reserved.
#
# This software is licensed as described in the file COPYING, which
# you should have received as part of this distribution. The terms
# are also available at http://trac.edgewall.org/wiki/TracLicense.
#
# This software consists of voluntary contributions made by many
# individuals. For the exact contribution history, see the revision
# history and logs, available at http://trac.edgewall.org/log/.
#
# Author: Alec Thomas <alec@swapoff.org>

revision = "$Rev$"
url = "$URL$"

import os
from fnmatch import fnmatch
from trac.core import *
from trac.config import Option
from trac.util.compat import set, groupby
from trac.perm import PermissionSystem, IPermissionPolicy
from configobj import ConfigObj
```

```
class AuthzPolicy(Component):

    implements(IPermissionPolicy)

    authz_file = Option('authz_policy', 'authz_file', None,
                        'Location of authz policy configuration file.')

    authz = None
    authz_mtime = None

    # IPermissionPolicy methods

    def check_permission(self, action, username, resource, perm):
        if self.authz_file and not self.authz_mtime or \
                os.path.getmtime(self.get_authz_file()) > self.authz_mtime:
            self.parse_authz()
        resource_key = self.normalise_resource(resource)
        self.log.debug('Checking %s on %s', action, resource_key)
        permissions = self.authz_permissions(resource_key, username)
        if permissions is None:
            return None                 # no match, can't decide
        elif permissions == ['']:
            return False                # all actions are denied

        # FIXME: expand all permissions once for all
        ps = PermissionSystem(self.env)
        for deny, perms in groupby(permissions,
                                   key=lambda p: p.startswith('!')):
            if deny and action in ps.expand_actions([p[1:] for p in perms]):
                return False            # action is explicitly denied
            elif action in ps.expand_actions(perms):
                return True            # action is explicitly granted

        return None                    # no match for action, can't decide

    # Internal methods

    def get_authz_file(self):
        f = self.authz_file
        return os.path.isabs(f) and f or os.path.join(self.env.path, f)

    def parse_authz(self):
        self.env.log.debug('Parsing authz security policy %s' %
                           self.get_authz_file())
        self.authz = ConfigObj(self.get_authz_file())
        self.groups_by_user = {}
        for group, users in self.authz.get('groups', {}).iteritems():
            if isinstance(users, basestring):
                users = [users]
            for user in users:
                self.groups_by_user.setdefault(user, set()).add('@' + group)
        self.authz_mtime = os.path.getmtime(self.get_authz_file())
    def normalise_resource(self, resource):
        def flatten(resource):
            if not resource or not (resource.realm or resource.id):
                return []
            # XXX Due to the mixed functionality in resource we can end up with
            # ticket, ticket:1, ticket:1@10. This code naively collapses all
            # subsets of the parent resource into one. eg. ticket:1@10
            parent = resource.parent
            while parent and (resource.realm == parent.realm or \
                    (resource.realm == parent.realm and resource.id ==
parent.id)):
                parent = parent.parent
            if parent:
                parent = flatten(parent)
            else:
                parent = []
            return parent + ['%s:%s@%s' % (resource.realm or '*',
                                           resource.id or '*',
                                           resource.version or '*')]
        return '/'.join(flatten(resource))

    def authz_permissions(self, resource_key, username):
        # TODO: Handle permission negation in sections. eg. "if in this
        # ticket, remove TICKET_MODIFY"
        valid_users = ['*', 'anonymous']
        if username and username != 'anonymous':
            valid_users += ['authenticated', username]
        for resource_section in [a for a in self.authz.sections
                                 if a != 'groups']:
            resource_glob = resource_section
            if '@' not in resource_glob:
                resource_glob += '@*'

            if fnmatch(resource_key, resource_glob):
                section = self.authz[resource_section]
                for who, permissions in section.iteritems():
                    if who in valid_users or \
                            who in self.groups_by_user.get(username, []):
                        self.env.log.debug('%s matched section %s for user %s'
                                % (resource_key, resource_glob, username))
                        if isinstance(permissions, basestring):
                            return [permissions]
                        else:
                            return permissions
        return None
```

Thus, we use a dedicated AuthZ-type file to divide our Trac environment into several parts, on which the different kind of users of our development tools (project developers, registered partners, anonymous visitors, etc.) have different rights.

4.6. Jenkins continuous integration server

Contrary to the other development tools, Jenkins is not a prepackaged application provided by the Debian-stable distribution. It is a server-side Java (JEE) application.

It comes with its own embedded web server (Winstone), and as such does not rely on Apache; its configuration is thus totally independent from the other tools. However, we purposedly managed to force these different program to use the same user & password database.

In order to maintain the maximum coherency with the other services, we chose to install Jenkins is the same partition (in `/srv/jenkins/`), and to execute it as the classical '`www-data`' user.

It runs through the HTTPS protocol – like our other tools – but on its own TCP port (8443).

As we did with Trac, we added different publicly available plug-ins to Jenkins to best suit our needs, the most important of them being:

- **Jenkins Emma plugin:** integrates the code coverage reports generated by the EMMA tool into our Jenkins installation, thus allowing us to mesure how well the tests we write do exercise our source code
- **Subversion tagging plugin:** allows us to “tag” automatically, in the SVN repository, each version of the source code that builds and passes all the tests successfully

But we also ***developed*** and and installed two custom plug-ins for Jenkins, to meet the following specific requirement of ours:

- **BuildCompleteHandler:** this plugin performs any useful action(s) – at the discretion of the Jenkins administrator – when a build finishes, according whether the build succeeded or failed; this Jenkins plugin is part of the mechanism – named OPCIM and registered by INRIA[†] – we developed to ensure that appropriate measures can be taken every time that some unaccurate source code is commited into the protected branches of our source code repository
- **`htdigest` Security Realm:** this plugin allows Jenkins to fetch user credentials from an Apache `htdigest`-type file; it allows us to have a common user database between Jenkins and our other, Apache-based development tools

We plan to publish these two plugins and their source code as an open-source contribution to the Jenkins community, as soon as the legal departments of the partners of the OP project have settled all the legal issues...

[†] APP deposit IDDN #FR.001.150008.000.S.P.2010.000.10000

5. User access to the provided development services

5.1. User accounts

As we stated before, all of our development tools use – purposedly – the same user & password database.

Consequently, management of user accounts is made through a single tool, that is: Trac. The Trac account manager plugin we installed allows every user to manage its account, and especially to change his password, define his mail address for contact, etc.

The OP project administrators can, thanks to the same mechanism, easily add, modify or delete user accounts, reset user passwords when needed (e.g. when a user forgets his/her password).

5.2. Services URLs

Each of the three main development tools we provide possesses a canonical URL, as well as some shortcuts that are easy to remember by the users.

The URLs are the following:

Tool	Canonical URL	Shortcuts
Trac	`https://dev.open-people.fr/`	`trac.open-people.fr` `dev.open-people.fr/trac`
Subversion	`https://dev.open-people.fr/svn/`	`svn.open-people.fr` `dev.open-people.fr/svn`
Jenkins	`https://dev.open-people.fr:8443/`	`jenkins.open-people.fr` `dev.open-people.fr/jenkins`

Note that canonical addresses are strictly HTTPS (since all of the services we provide are secured), while shortcuts also work with the "plain" HTTP protocol.