


| | |
|---|--|
|  | Titre du document : Compte rendu d'étude |
| | Référence : L 1.4.1 |
| | Version du 24/09/2011 |

Contrôles d'accès et implémentations

| | |
|---------------------------------------|---------|
| Livrable du au titre du projet | COCLICO |
| Lot | 1 |
| Tache | 0.4 |
| Livrable | 1.4.1 |

| Rédacteur(s) | Vérificateur(s) | Approbateur(s) |
|---------------------|------------------------|-----------------------|
| C Bayle | | |




| | |
|---|--|
|  | Titre du document : Compte rendu d'étude |
| | Référence : L 1.4.1 |
| | Version du 24/09/2011 |

Table des matières

| | |
|--|---|
| I.Authorization | 4 |
| A.Authorization in Codendi | 4 |
| i.Introduction | 4 |
| ii.Server Configuration | 4 |
| iii.User classes | 4 |
| iv.Project visibility | 5 |
| v.User Permissions: delegating service administration rights | 5 |
| vi.User Groups: setting access rights on Codendi resources | 5 |
| a)User Groups | 5 |
| b)Additional Information on User Groups | 6 |
| c)Where can I use user groups to define permissions? | 6 |
| B.Authorization in Fusionforge | 7 |
| i.Software Design | 7 |
| a)Introduction | 7 |
| b)Site level | 7 |
| Server Configuration | 7 |
| Site Admin | 7 |
| c)Project level | 7 |
| Project visibility | 7 |
| Project admin | 8 |
| d)Service level | 8 |
| Permission config for services | 8 |
| Rights on each service | 8 |
| e)User level | 8 |
| f)Groups | 8 |
| ii.Implementation | 9 |
| a)Database | 9 |
| b)PHP Classes | 9 |

| | |
|---|--|
|  | Titre du document : Compte rendu d'étude |
| | Référence : L 1.4.1 |
| | Version du 24/09/2011 |

| | |
|----------------------------------|----|
| c)How does it work ? | 9 |
| C.Authorization Frameworks | 9 |
| i. Which frameworks ? | 9 |
| ii. Main principles | 10 |

| | |
|---|--|
|  | Titre du document : Compte rendu d'étude |
| | Référence : L 1.4.1 |
| | Version du 24/09/2011 |

I. Authorization

Authorization : <http://en.wikipedia.org/wiki/Authorization>

“**Authorization** (also spelt **Authorisation**) is the function of specifying access rights to resources, which is related to information security and computer security in general and to access control in particular. More formally, "to authorize" is to define access policy. For example, human resources staff are normally authorized to access employee records, and this policy is usually formalized as access control rules in a computer system. During operation, the system uses the access control rules to decide whether access requests from (authenticated) consumers shall be approved (granted) or disapproved (rejected). Resources include individual files' or items' data, computer programs, computer devices and functionality provided by computer applications. Examples of consumers are computer users, computer programs and other devices on the computer.”

A. Authorization in Codendi

i. Introduction

Codendi offers a wide range of permission mechanisms to allow very specific access customizations.

Those mechanisms are available on several levels: site level, user level, project level, service level.


ii. Server Configuration

At site level, it is possible to configure several permission settings (see `/etc/codendi/conf/local.inc`):

- anonymous users allowed (`$sys_allow_anon`): if this option is disabled, users will need to log in before accessing any content on the platform
- default project privacy settings (`$sys_is_project_public`): Projects can be made private by default (see below).
- enable restricted users (`$sys_allow_restricted_users`): if this option is enabled, you will be able to have restricted users on the server (see below).

iii. User classes

Depending in the server configuration (see above) there might be different user classes on the server: - anonymous users: if enabled, anonymous users will be able to browse all public projects on the server, and access site features (documentation, project tree, code snippets...). - active users: they have the same possibilities, but they can also become project members. As members, they will be able to submit or update project data (source code, artifacts, documents...) - restricted users: if enabled, those users may only access data from project they belong to. You can customize whether restricted users have access to the code snippet library, Codendi welcome page, site documentation,

| | |
|---|--|
|  | Titre du document : Compte rendu d'étude |
| | Référence : L 1.4.1 |
| | Version du 24/09/2011 |

etc.

iv. Project visibility

Projects may have to visibility status: - public: visible and accessible to all - private: visible and accessible only to project members - "fully public projects": public projects that are also visible and accessible to restricted users. This is the default status of the "Codendi Administration Project". This is configurable through the file site-content/en_US/include/restricted_user_permissions.txt.

v. User Permissions: delegating service administration rights

!Location: Project Administration Menu Project Administrators have the ability to grant different permissions to different project members. As an example, a project member can be granted full administration rights on the bug tracker and no rights at all on the Documentation Manager of the project.

In the User Permissions page, any project administrator can grant or revoke administration on Codendi services: - Project administration: a project administrator has full rights on all services of the project - Subversion: SVN admin may set permissions on the repository, define global settings and notifications. - Forums: Forum Moderators has the ability to moderate the Web Discussion forum that is to say create/delete discussion forums for the project, delete posted messages and update the Forum status (public/private) as well as the Forum description - Wiki: Wiki admin may manage wiki pages, attachments and permissions - News: News Writer may submit news, and News Admin may delete them - File manager: File admins may create/update/delete/restrict packages, releases and files. - Trackers: there is one admin role for each tracker. A Tracker Admin may update the tracker structure, values, permissions...

This is specified on a per-member basis


vi. User Groups: setting access rights on Codendi resources

! Location: Project Administration Menu

a) User Groups

A user group, sometimes called a "ugroup", is simply a group of Codendi users. User groups are used to set specific permissions to some project data (e.g. software releases and packages, documents, trackers and artifacts, SVN branches, ...). A user group is always attached to a project, but the users comprising the group do not necessarily belong to that project.

The "User Groups Admin" function of the Project Administration menu lists all available user groups, and provides a way to create new ones.

| | |
|---|--|
|  | Titre du document : Compte rendu d'étude |
| | Référence : L 1.4.1 |
| | Version du 24/09/2011 |

There are two different kinds of user groups:

- Pre-defined User Groups: These groups are defined for every project. Examples of pre-defined groups are: project_members, project_admins, registered_users, nobody, file_manager_admin, etc. These groups are dynamic: if you assign some permission to 'project_admins', and a new project administrator is defined, then this new user will automatically be granted the corresponding permission. - Custom User Groups are defined by project administrators. They are composed of a static list of users. The only requirement is that any member must be a registered Codendi user. This list can be modified at any time, but will not automatically be updated, except if a member is removed from the project or deleted from the system.

!Tip: Combining pre-defined groups with additional members

Sometimes, you might want to grant some permissions to all project members and some other Codendi users. In this case, you might be tempted to build a user group from the list of project members and to add the other users to the group. The issue with this solution is that if new members join the project, they will have to be manually added to the group. So it is more convenient to create a group containing only the users that are not member of the project. And then, permissions should be granted to this group and to the pre-defined "project members" group.

b)Additional Information on User Groups

It is possible to know all user groups one individual project member belongs to. Simply display the User Permissions page. However, please note that only user groups belonging to the current project are displayed. The user might also be a member of additional user groups in other projects.

The bottom of the User Group Edit page also lists all the permissions granted to this group, e.g. packages and releases this user group is granted access to.


When a project member is removed from a project, or quits a project, they are also automatically removed from all project user groups for safety reasons.

Similarly, when a user is deleted (not just suspended) by the site administrator, they are removed from all user groups in all projects.

Please note that if a user group was specifically granted some permission, deleting the user group might be dangerous. Indeed, if a group is the only one allowed to access a package and this group is deleted, the permission is also deleted and reset to default, so any registered user can access the package.

c)Where can I use user groups to define permissions?

- Subversion: you can set permissions with user groups on any svn directory of the project repository. Mostly used to create read-only or private branches. - Documents: There are three permission levels on documents and folders: read, write and manage. "Manage" is "write" plus the

| | |
|---|--|
|  | Titre du document : Compte rendu d'étude |
| | Référence : L 1.4.1 |
| | Version du 24/09/2011 |

right to set permissions on the item. User groups are needed to set permissions. - Wiki: It is possible to set permissions on wiki pages using user groups. Please note that if a wiki page is pointed from the document manager, it inherits from the permissions set in the document manager. - Files: you can use user groups to set permissions on packages and releases. - Trackers. There are three different permission mechanisms where you may use user groups:

- tracker access: defines who can access the whole tracker
- field permissions: defines who can read, submit or update any tracker field
- artifact permissions: you can set specific permissions on individual artifacts (e.g. to hide an artifact that contains private data).

B. Authorization in Fusionforge

i. Software Design

a) Introduction

Fusionforge offers a wide range of permission mechanisms to allow very specific access customizations. Those mechanisms are available on several levels: site level, project level, service level, user level.

b) Site level

• *Server Configuration*

In the server configuration, it is possible to configure several permission settings (see `/etc/gforge/local.inc`):

- project registration (`$sys_project_reg_registered`) : if this option is enabled, only site admin can register project
- user registration (`$sys_user_reg_registered`) : if this option is enabled, only site admin can register user

• *Site Admin*


The site admin sees an "Administration" tab in his interface. In this admin page, he can manage users, groups (add new, validate, revoke...) and configure forge global parameters (files types, themes, cron, configuration (edit local.inc)). Furthermore he has all project admin rights.

c) Project level

• *Project visibility*

Projects may have two visibility status:

- *public*: visible and accessible to all
- *private*: visible and accessible only to project members

| | |
|---|--|
|  | Titre du document : Compte rendu d'étude |
| | Référence : L 1.4.1 |
| | Version du 24/09/2011 |

•*Project admin*

He manages the access and permissions to the project. In his project interface, an "Administration" sub-tab is present. He has the ability to create and configure the different roles that exist in the project. By default there are 5 roles :

- Admin
- Senior Developer
- Junior Developer
- Doc Writer
- Support tech.

d)Service level

For each service (wiki, forum, tracker...) , we can configure different access : anonymous access, read, post, admin, moderate... This is done by project admin or members of project that have been granted admin rights on the service.

•*Permission config for services*

Accessible only by service admin:

- Forum : allow anonymous message, public/private, manage user
- Tracker : allow anonymous message, public/private, description
- Tasks : add tasks...

...

•*Rights on each service*


- File release system : (read/write)
- Forum admin : (none/admin)
 - Forum : (post/read/admin/no access)
- Tracker admin : (none/admin)
 - Tracker : (tech/read/admin only/tech & admin/no access)
- Tasks admin : (tech/read/admin only/tech & admin/no access)
- Webcal (no access/modify/see)

e)User level

For each project, the project admin manages users (add, remove, validate, attribute a role...). Project admin has the ability to grant different roles to different project members. This is specified on a per-member basis. The project admin has to manage right for observers too (visitors not logged in or not member of the project).

f)Groups

In Fusionforge, a project is a group. For instance in Administration page, the site admin may know

| | |
|---|--|
|  | Titre du document : Compte rendu d'étude |
| | Référence : L 1.4.1 |
| | Version du 24/09/2011 |

for one user which groups he belongs to. There are special groups such as Site Amin with only the *Local GForge Admin* as member.

It is not possible to grant permissions to all members group at the same time, it is a one-by-one operation.

ii. Implementation

The implementation of these features is done with some tables in the database (that define roles, permission, associate user and role...) and with some php classes.

a) Database

- **role**(role_id,group_id,role_name) : lists all the roles that are defined in each group. *role_id* is unik but two different groups can be associated with same *role_name*. (e.g "Admin" is a *role_name* defined in evry *group_id*)
- **role_setting**(role_id,section_name,ref_id,value) : for each *role_id* associated with a *section_name* (trackeradmin, docman...) defines the permission (*value*)
- **user_group**(user_group_id,user_id,group_id,admin_flags,forum_flags,...,tracker_flags,member_role,role_id): for each user member of a group defines the permission (flags) for every services.

b) PHP Classes

There are all in /common/include directory.

- **Role.class.php** : defines the default roles (Admin, developer, writer...) and lets us define new roles, update existing roles...
- **Permission.class.php** : read the database to answer question like (is member? is siteAdmin? is forumAdmin? etc.)

c) How does it work ?

Each service of a project calls `$perm = $this_group->getPermission($user)` then `$perm->isMember` , `$perm->isAdmin...` to define the user access.


C. Authorization Frameworks

i. Which frameworks ?

a) Zend Acl

Here we have more informations about ACL in general and advanced features of Zend_ACL: Zend_ACL CodeUtopia

b) Apache Shiro project previously know as Jsecurity

| | |
|---|--|
|  | Titre du document : Compte rendu d'étude |
| | Référence : L 1.4.1 |
| | Version du 24/09/2011 |

Apache Shiro is a Java security framework that performs authentication, authorization, cryptography, and session management.

c)SimpleSAMLphp

Based on SAML 2.0, service provider, identity provider.

ii.Main principles

These frameworks distinguish three parts : roles, ressources and privileges. A user has a role (project member, admin...), wants to access a ressource (webpage, file, forum...) with some privileges (read, download, post...)

Thus we need files to define rules and then for each access request we call `$user->isAllowed($ressource,$privilege)`