Access Control Lists (ACL)s

DokuWiki — like most wikis — is very open by default. Everyone is allowed to create, edit and delete pages. However sometimes it makes sense to restrict access to certain or all pages. This is when the *Access Control List* (ACL) comes into play. This page gives an overview of how ACLs work in DokuWiki and how they are configured.

Configuration and Setup

ACLs can be enabled in the installer and an initial ACL policy is set there as well. To manually enable ACLs, switch on the useacl option and create a copy of the example files `conf/acl.auth.php.dist` and `conf/users.auth.php.dist`. Rename the files to `conf/acl.auth.php` and `conf/users.auth.php` respectively.

Example of a minimal `conf/users.auth.php` file for a user `admin` with the password `admin`. If you use it, be sure to change the password afterwards.

conf/users.auth.php

```
# login:passwordhash:Real Name:email:groups (comma[,] separated)

admin:$2y$10$P5YH8uIM2uAE9snRq32yAuHMb4/XAzksFd5Cakqqtsw9BWeSsyLZq:admin:admin@admin.com:admin,user
```

See also

There are a few more config options and features that relate to authentication, user registration and ACL setup. Please check their respective wiki pages to get more information:

- Config option useacl – enable ACL usage
- Config option superuser – setup superusers with ACL granting rights
- Config option defaultgroup – the default group to which new users are added
- User Manager – managing users
- Authentication Backends – identify users from different data sources
- FAQ: How to disable open user registration – replaces $conf[openregister]

WARNING: DokuWiki's ACL feature has been included for some time and should be pretty stable. However, if you are concerned about the risk of unauthorized users accessing information in your wiki, you should never put it on a computer accessible from the Internet.

Access Restrictions

Access restrictions can be bound to pages and namespaces. There are seven permissions: *none*, *read*, *edit*, *create*, *upload*, *delete* and *admin*. Each higher permission contains the lower ones, with read being the lowest and delete the highest one. You should note that create, upload and delete permissions can only be assigned to namespaces.

Rules that were set to namespaces apply on media namespaces as well as for page namespaces.

When DokuWiki checks which rights it should give to a user, it uses all rules matching the user's name or the groups he or she is in. The rule that provides a user's permission is chosen according to the following process:

- Rules which match closer to the namespace:page are preferred over rules which match further away—we call this «specific matching».
- When more than one rule matches at the same level, the rule giving the highest access level is preferred.

Users are in the groups they were assigned to in the user manager (or the auth backend). However there are two **groups** that are somewhat special:

- **@ALL** Everyone, even users not logged in, is a member of the ALL group. You can use this group to restrict access for all users (as a default setting) and then relax the permissions for some selected users.
- **@user** All self-registered users are by default automatically a member of the group 'user'. Use this to give permissions to 'logged-in' users. The name of this group is configured through the defaultgroup option. Unlike the virtual «ALL» group, the «user» group is a real group to which all users are added automatically when using the plain auth backend. If you use a different backend you need to use the groups provided by this backend.

Groups are represented internally and in the ACL manager by a prepended @ character to the group name.

Editing ACLs

To easily add new or change existing access rules, you should use the ACL Manager which is available from the Administration menu. A detailed description of its interface can be found here.

Basically there are three steps to add a new ACL rule:

1. select the namespace or page to restrict from the upper left tree navigation
2. choose to whom the ACL rule should apply
    - by selecting a known group or user from the dropdown menu
    - or by selecting «User:» or «Group:» and entering the group or user name in the field
3. set the appropriate permissions

Existing rules can be modified or deleted in the table at the bottom of the ACL manager.

ACLs by Example

In this section we will explain how access rules work, using a fictional example setup that looks like this in the ACL manager:

Let's have a look at each line:

1. This sets permission for everyone in the main namespace, allowing everybody to edit and create pages. However upload is not allowed.
2. User *bigboss* is given full rights.
3. Now the access for the `devel` namespace is restricted. Nobody is allowed to do anything.
4. Well not nobody really—we give members of the *devel* group almost full rights here. Deleting files however is not allowed.
5. User *bigboss* however is allowed full access to the `devel` namespace. He/she can even delete uploaded files.
6. The *marketing* group may read everything in the `devel` namespace, but cannot edit or create pages.
7. However the devel team doesn't want their boss to see the `funstuff` page—remember exact pagematches override namespace permissions.
8. And finally the *marketing* group is allowed to edit the `devel:marketing` page as well. (This page could however not have been created by them.)
9. Then the permissions for the namespace `marketing` are set. All members of the *marketing* group are allowed to upload there
    - other users will be matched by line #1 so they can still create and edit.
    - Rights for *bigboss* are inherited from line #2 so this user can still upload and delete files. (No wonder that everyone would like to be the *bigboss*.)
10. The last line finally restricts the start page to readonly for everyone. Even for *bigboss*. Only superusers will be able to ever edit that page.

Let's have a look at a second example to better understand **specific matching**:

This time we look what rules will match for different users when trying to access the page `private:bobspage`.

1. abby, a regular user
    - three rules match, #1, #2, #4
    - rule #4 is closest, it matches at the namespace level so it takes precedence over the other three
    - abby's permissions level is `None`
2. bob, a regular user
    - four rules match, #1, #2, #4, #6
    - rule #6 wins as its an exact match
    - bob's permission level is `Delete`
3. bob forgets to login and tries to access his page
    - two rules match, #1 & #4
    - rule #4 is closer, it wins
    - bob's permission level while not logged in is `None`
4. charlie, a staff member

- five rules match, #1–#5
- two rules match at namespace level, #5 gives charlie the higher permission so it wins
- charlie's permission level is `Delete`

Note rule #5, which appears to duplicate rule #3. Without it, staff members wouldn't be able to access the private namespace as rule #4 would keep them out.

Background Info

Access restrictions are saved in a file called `conf/acl.auth.php`, which should be writable by the webserver if you want to use the ACL admin interface described above. It is not recommended to edit this file manually. Use the admin interface instead.

Empty lines and shell-style comments are ignored. Each line contains 3 whitespace separated fields:

- The resource to restrict. This can either be a pagename or a namespace. Namespaces are marked by an additional asterisk (see examples below).
- A group or user name. Groupnames are marked by a leading @ character.
- A permission level (see below).

There are 7 permission levels represented by an integer. Higher levels include lower ones. If you can edit you can read, too. However the *admin* permission of *255* can not be used in the `conf/acl.auth.php` file. It is only used internally by matching against the superuser option.

Name	Level	applies to	Permission	DokuWiki constant
none	0	pages, namespaces	no permission—complete lock out	AUTH_NONE
read	1	pages, namespaces	read permission	AUTH_READ
edit	2	pages, namespaces	existing pages may be edited	AUTH_EDIT
create	4	namespaces	new pages can be created	AUTH_CREATE
upload	8	namespaces	mediafiles may be uploaded	AUTH_UPLOAD
delete	16	namespaces	mediafiles may be overwritten or deleted	AUTH_DELETE
admin	255	admin plugins	superuser[1] can change admin settings	AUTH_ADMIN

Here is an example setup matching the first example given above:

```
*                   @ALL        4
*                   bigboss    16
devel:*             @ALL        0
devel:*             @devel      8
devel:*             bigboss    16
devel:*             @marketing  1
devel:funstuff      bigboss     0
devel:marketing     @marketing  2
marketing:*         @marketing  8
start               @ALL        1
```

Please note that **order does not matter** in the file. The file is parsed as whole, then a perfect match for the current page/user combo is searched for. When a match is found further matching is aborted. If no match is found, group permissions for the current page are checked. If no match is found the

check continues in the next higher namespace.

Note: The delete permission affects media files only. Pages can be deleted (and restored) by everyone with at least edit permission. Someone who has upload permissions but no delete permissions can only overwrite existing media files if the media revisions option is enabled.

User/Group Encoding

Because the ACL configuration uses a few special characters to denote special functionality (like @ prefixes, spaces, etc), user and group names need to be encoded when they contain certain characters to avoid clashes.

When you use the ACL Manager you don't have to think about this, it will do it automatically for you.

When manually editing ACLs, user and group names need to be encoded. Internally this is done using the auth_nameencode() method.

The encoding uses URL encoding for all non-letter/number ASCII chars (special chars in the lower 128 byte range). UTF-8 Multibytechars are not encoded.

Example: `Herbert.Müller` becomes `Herbert%2eMüller`

User Wildcards

It is possible to use user and group wildcards in the ACLs. This can be useful for Wikis with many registered users, if you want to give each user or group a personal namespace where only he/she has write access, and you don't want to edit the ACLs for each of them. To accomplish that **%USER%** is replaced by the username of the currently logged in user and **%GR0UP%** by all the groups of this user.

In the following example a logged-in user gains full access (upload/delete) permissions for the user's namespace `user:<username>:*` and revoke all access from other namespaces located in `user:*`.

In this case logged-in user has access to own namespace only and have not access to users namespaces (even view names of namespaces) of other users.

```
#
# Grant full access to logged in user's namespace
user:%USER%:*          %USER%  16
#
# Allow to browse own namespace via the index
user:                  %USER%  1
#
# Allow read only access to start page located in "user" namespace
user:start             %USER%  1
#
# Disable all access to user's home namespaces not owned by logged in user
# (include view namespaces via the index)
user:*                 @user   0
```

```
#
# Allow members of 'group' to edit pages in the 'group' namespace.
# BE CAREFUL, if you have a 'user' namespace, all members of the default
group
# will gain access to it since %GROUP% will be replaced literally
%GROUP%:*                %GROUP% 2
```

Note: version 2009-12-25c «Lemming» has some caveat. If you add, update or remove ACL entries from the admin interface then DokuWiki will replace %USER% in the second field of the ACL to %25USER%25 (this is bug FS#1955). To avoid this, change permissions manually only (by editing: `conf/acl.auth.php`) or correct them manually after each operation in the admin interface because %25USER%25 does not work as expected, only %USER% should be used in the `conf/acl.auth.php`. This bug is fixed in newer versions.

Note: The wildcard changed from @ to % in December 2008 – if you are upgrading from an older version you need to adjust your ACL setup accordingly.

1)

see superuser