Advanced Access Manager

ConfigPress Tutorial

V 1.0

Content

Introduction

ConfigPress has been introduced couple versions before and the main goal is to configure the Advanced Access Manager (AAM) behavior in more specific way. The way you need it!

Think about ConfigPress as a Settings page for AAM but instead of clicking a lot of buttons and checkboxes, you type the configurations. And yes, this may make the configuration process more complicated, but also way more flexible.

The ConfigPress syntax is based on standard Windows INI file. So my suggestion will be to spend 10-15 minutes and understand how it works.

This tutorial is divided on four main sections:

AAM Configurations. Explains all available settings for configuring the plugin's core behavior. It is worth reading if you are advanced WordPress user. As well as it has some useful set of configurations for Multisite support.

Backend Configurations. List of all possible configurations to specify the way how AAM should handle access control for WordPress Backend.

Frontend Configurations. Very similar as Backend Configuration just for Frontend part.

Advanced Configurations. Explains how to use custom functions to extend the AAM functionality.

AAM Configurations

Overview

This section introduces you to the core AAM configurations. I'm suggesting to be careful with these configurations in fact it may cause incorrect WordPress behavior or reduce website performance.

All configurations below should be included in INI section **[aam].** Please make sure that you understand what does it mean INI section before your proceed with tutorial.

Caching

AAM has internal caching mechanism to store individual user settings to speed-up the system. From release 1.7 the cache is stored to MySQL database table **Options**. Before all caching was based on Zend Framework module Zend_Cache.

To turn ON/OFF the caching use next configurations:

```
[aam]

caching = "true" ;turn on caching

caching = "false" ;turn off caching
```

Delete Capabilities

By default, there is no possibility to delete any capability under Capabilities Tab.

Before you made a decision to delete any capability, please make sure that you know what you are doing. If you delete any capability from the list, it'll delete it from all User Roles and this may cause you a lot of troubles.

```
[aam]

delete_capabilities = "true" ;turn on feature

delete_capabilities = "false" ;turn off feature
```

License Key

In case you purchased the Premium Version, you probably already received an email with license key. After you put a correct key, the AAM will make a web service request to our server and the necessary add-on will be received as the response.

Please do not try to put a fake license key. This does not make any difference for our web service but will slow down your website in fact each new request to it will trigger request to our server. And our server will response with failure.

The license key is 40 characters unique sha1 hash which is generated automatically after payment processed successfully.

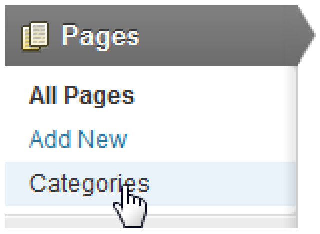
```
[aam]

license_key = “your license key here”
```

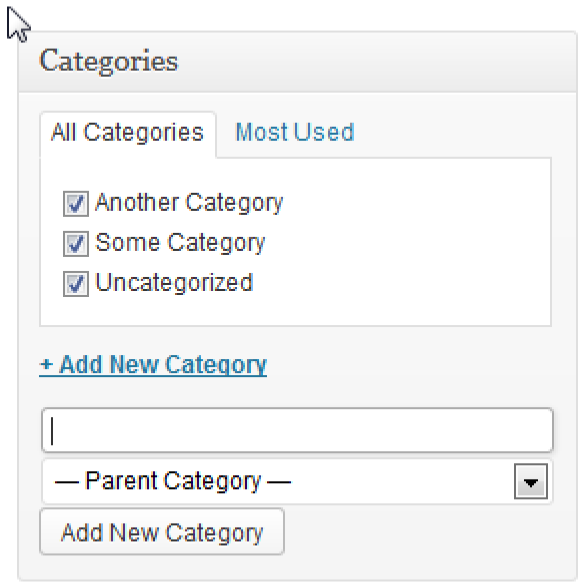
Page Category

This feature is available only in Premium Version. It allows you to group your Pages into Categories.

Very useful feature if you want to manage access to the group of Pages but not individually.

As soon as you activate the Page Category, you’ll find additional link under Pages menu as well as new metabox in Edit Page Screen.





```
[aam]

page_category = ”true” ;turn on feature

page_category = “false” ;turn off feature
```

Multisite support

There are couple setting for multisite support. So if you are not using multisite, you may skip this section.

There is a way to specify default Site Template for your WordPress multisite. So let’s say you created a new site and named it “Default Template”. The ID of this site is 5 (to find the ID the of the site just hover on it and get the ID from the URL). Now you can specify default site setting and each new Site you are going to create in future will copy all settings from Default Template site.

```
[aam]

multisite.default_site = “5” ;ID of Default Template site
```

In case you want to manage all your sites from one, you may set the Apply All settings and this will copy all settings from currently editing site to all other. This feature is turned off by default.

```
[aam]

multisite.apply_all = “true” ;turn on feature

multisite.apply_all = “false” ;turn off feature
```

And now even more flexible set of feature. You can specify exactly what you want to apply. For example if you want to copy the settings of Current Role to all other sites you may specify next configuration:

```
[aam]

multisite.apply[] = “current_role”
```

By default next set of settings are transferred to other sites, only if Apply All feature is on: Menu, Menu Order, Metaboxes, Capabilities and Current Role.

And this is available list of configurations:

```
[aam]

multisite.apply[] = “current_role” ;transfer current role

multisite.apply[] = “menu” ;menu settings from Menu Tab

multisite.apply[] = “menu_order” ;menu order

multisite.apply[] = “metaboxes” ;metaboxes settings

multisite.apply[] = “capabilities” ;current role capabilities

multisite.apply[] = “role_list” ;transfer all role
```

Backend Configurations

Overview

This part of tutorial shows you how to configure the backend access behavior.

Please be sure that all settings below are included inside the [backend] section.

All configurations below should be included in INI section **[backend].** Please make sure that you understand what does it mean INI section before your proceed with tutorial.

Deny Access Behavior

Sometimes it is important to define the default procedure for request or action which is not authorized. Let's say you want to redirect user to the WordPress Dashboard if he is not allowed to see the list of posts.

In this case you may to configure the denial procedure with next lines of configurations:

```
[backend]

access.deny.redirect = "http://yourdomain/wp-admin"
```

In case you want to display just message you may use next configurations:

```
[backend]

access.deny.message = "Your message here"
```

Taxonomy Access

There is a way to configure the different level of taxonomy accessing. So this is the next list of possible configurations:

```
[backend]

access.taxonomy.default.list = "deny or allow"

access.taxonomy.default.browse = "deny or allow"

access.taxonomy.default.edit = "deny or allow"
```

Post Access

In the similar way as for taxonomies, there is a set of configurations to control access to posts and pages (please take in consideration that Post & Page or Media for WordPress internal is the same type of object which is called POST).

```
[backend]

access.post.default.list = “deny or allow”

access.post.default.trash = “deny or allow”

access.post.default.delete = “deny or allow”

access.post.default.edit = “deny or allow”

access.post.default.publish = “deny or allow”
```

Role and User Access

Now let’s make the configuration part even more flexible. Please notice that you can use the Posts & Pages Tab in AAM to do the same. But the beauty of ConfigPress configurations is that you can specify the default behavior for all posts and taxonomies (so in case you have many posts this is very useful).

Ok. So to control access to specific User Role you have to know the internal WordPress Role ID. Usually it is the lowercase representation of the User Role name. Let’s say User Role **Administrator**, internally will have ID **administrator**. In case the User Role has couple words, the white spaces may be replaced with underscore “**_**”. So the User Role Hello World will be hello_world.

```
[backend]

role.all.access.taxonomy.default.list = “deny or allow”

role.editor.access.taxonomy.default.edit = “deny or allow”
```

The same scenario if for specific user. To get the User ID, go to Users and hover on the username. The URL contains the user_id parameter which you may use:

```
[backend]

user.5.access.post.default.publish = “allow or deny”
```

Frontend Configurations

Overview

Frontend access control is pretty much the same as Backend. So I'm not going to repeat myself the only difference is the list of possible access configurations for taxonomies and posts.

All configurations below should be included in INI section **[frontend].** Please make sure that you understand what does it mean INI section before your proceed with tutorial.

Taxonomy Access

For taxonomies there are two different types of access. So you can specify if user has access to list the taxonomies and to browse them.

```
[frontend]

access.taxonomy.default.browse = "allow or deny"

access.taxonomy.default.list = "allow or deny"
```

Post Access

This is the next possible list of configurations for posts and pages.

```
[frontend]

access.post.default.list = "allow or deny"

access.post.default.read = "allow or deny"

access.post.default.comment = "allow or deny"
```

Advanced configurations

Overview

For every single parameter in ConfigPress you may specify the custom user function and trigger it. To make myself clear, let's show on two examples.

Example #1. Subscription

Let's say you have a website where users (with the User Role **Subscriber**) may subscribe on different categories. The subscription expires in a year. So you would like to know if current user still has the active subscription. If not, then redirect him to the payment page.

The ConfigPress will be next:

```
[frontend]

subscriber.taxonomy.access.default.browse.userFunc =
"MySubsciptionClass::checkSubscription"
```

This tells to AAM that there is a user function **checkSubscription** inside the **MySubscriptionClass** which will return value allow or deny. Please notice that you, as developer, has to be sure that this class is included to the global scope and is accessible.

Inside that function you can get the current user ID, taxonomy ID or post ID and based on this information perform the check, as well as perform necessary redirect.

Example #2. License Key

You purchased the Premium version of AAM and would like to hide the license key, so other administrators will not be able to see it. In this case you make create a custom function which will return the key:

```
[aam]

license_key.userFunc = "returnMyLicenseKeyFunction"
```