Our.Shield

Umbraco active security modules

# Installation

### Installing via NuGet

This Umbraco package can be installed via NuGet

The first part is the Shield framework, which coordinates the different security apps, which can be found here

<https://www.nuget.org/packages/Our.Shield.Core/>



And the second part is the Shield apps, which provide the active security. Note, there are no restriction on the number of shield apps that can be installed. If you want, install them all using NuGet, to gain the full benefits of what Shield can provide.

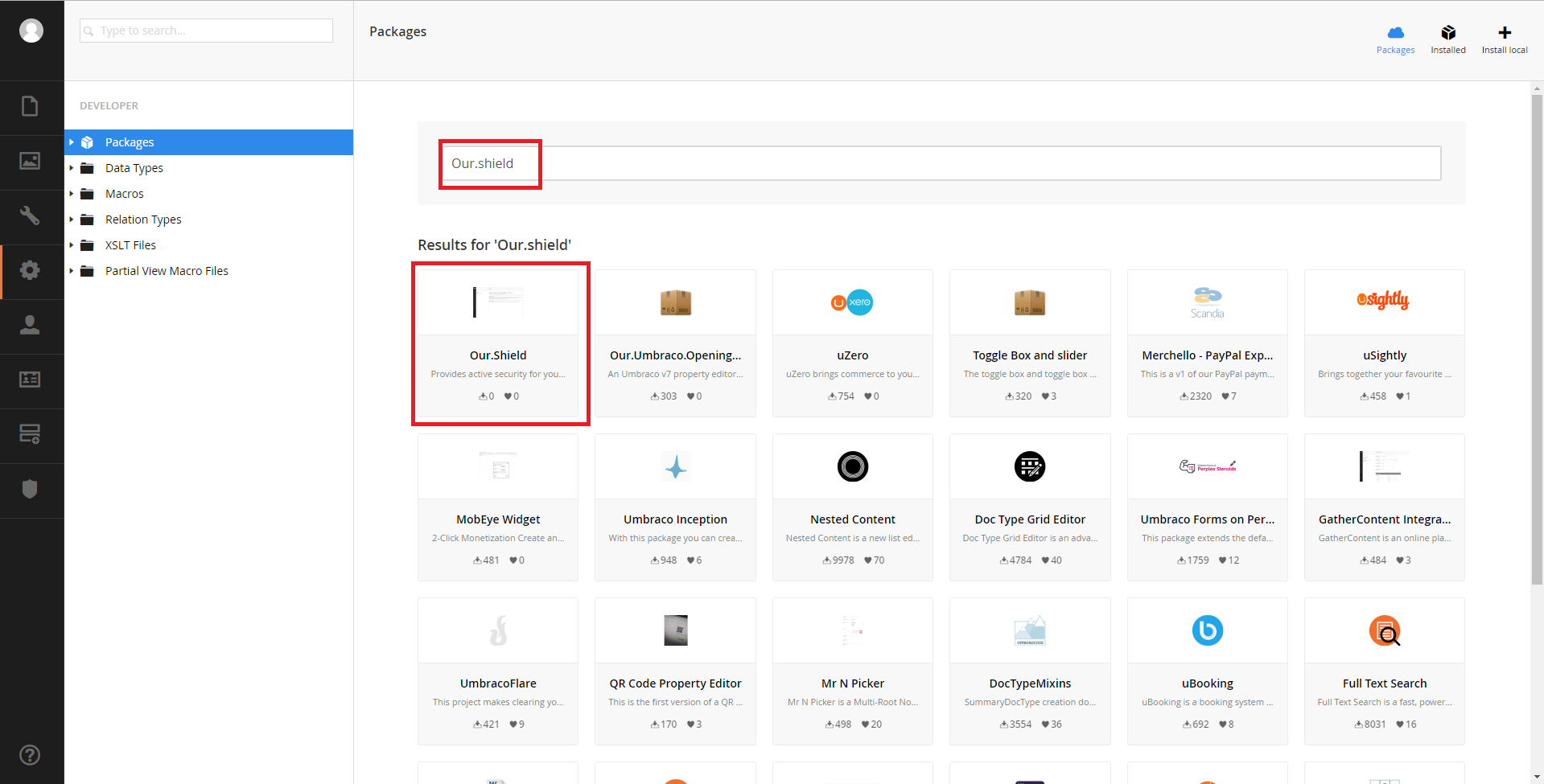
* **Backoffice Access**   
  Gives you the ability to configure and restrict access to the backoffice access URL.  
    
  <https://www.nuget.org/packages/Our.Shield.BackofficeAccess>  
    
  
* Media Protection  
  Disable [Hotlinking](https://simple.wikipedia.org/wiki/Hotlinking) and to secure your media to only be accessed by authenticated members.  
    
  <https://www.nuget.org/packages/Our.Shield.MediaProtection>  
    
  

### Installing via Umbraco Package Manager

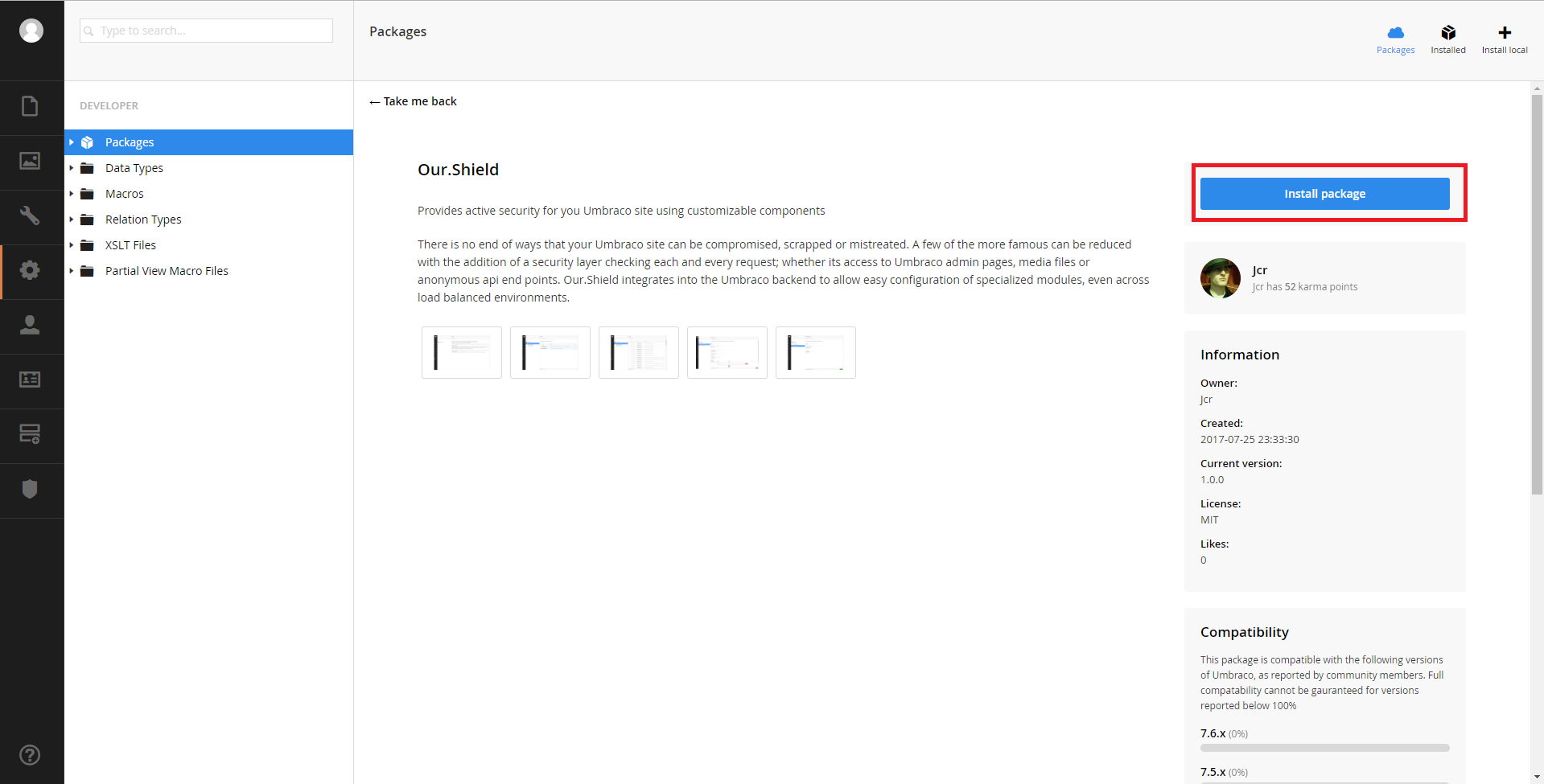
<https://our.umbraco.org/projects/backoffice-extensions/ourshield/>

This installation contains the Shield framework, and all available Shield apps

First, navigate to the developer section of Umbraco, click on the packages node and search for Our.Shield



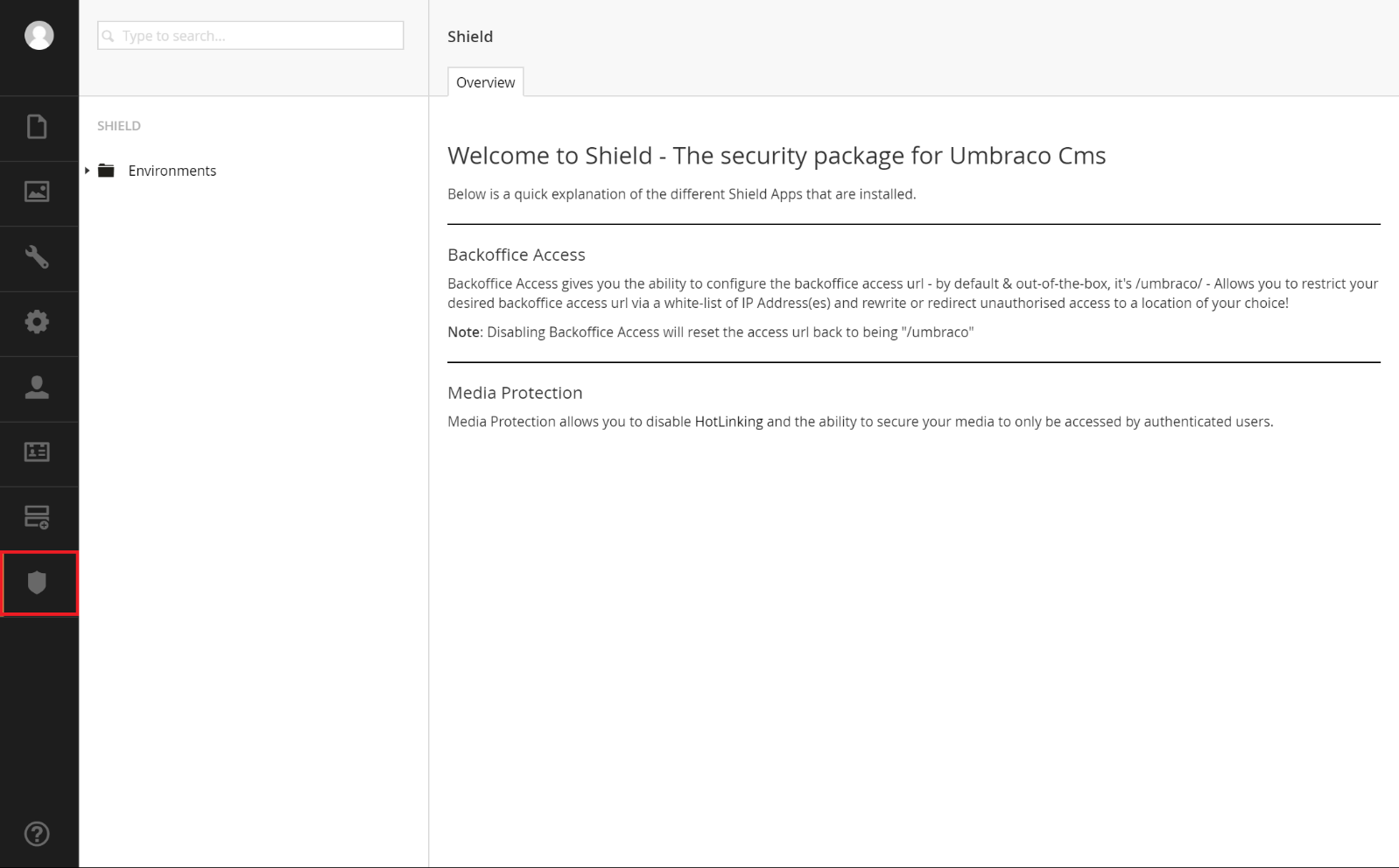
Next, Click on the package, and then the install button

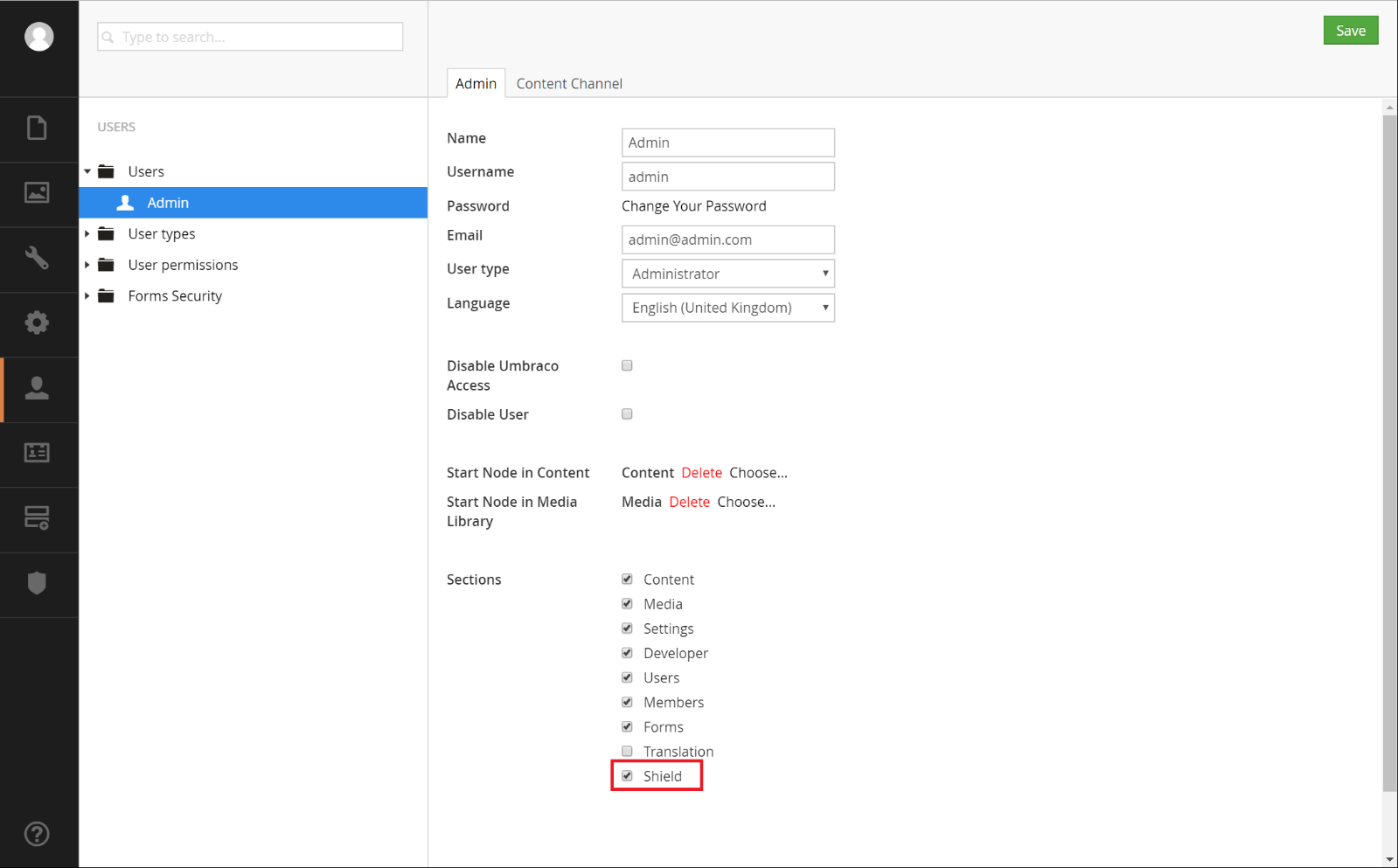


Afterwards, Our.Shield should be installed

# Our.Shield.Core

Our.Shield.Core is the framework for the Our.Shield Umbraco package. It contains the custom section to be displayed in Umbraco and does the ‘heavy’ lifting for the installed app(s).

Once installed, you should see a new custom section within the backoffice of Umbraco.

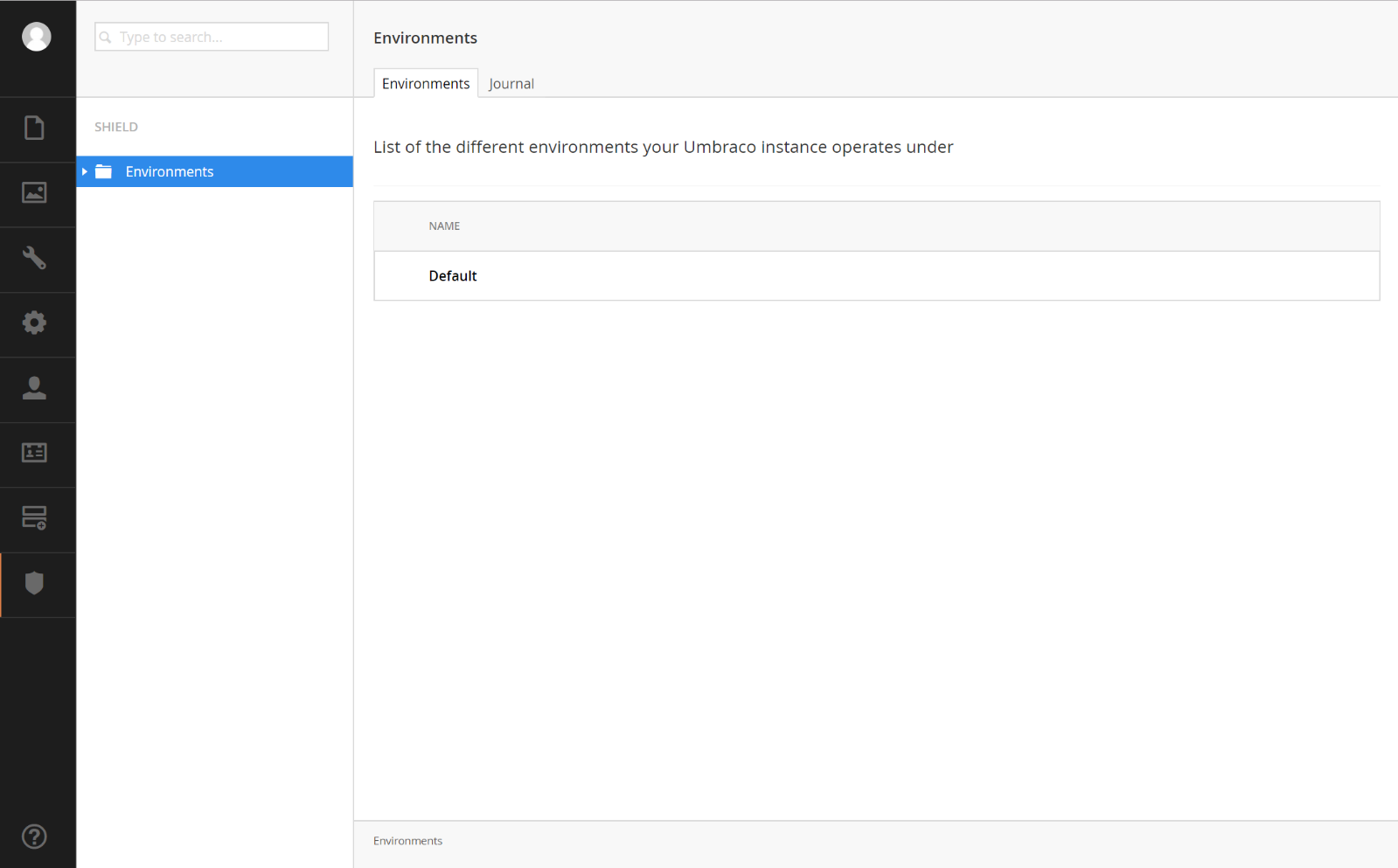
If the new section doesn’t display, you’ll need to allow the currently logged in user to have access to the Shield section via the users’ section:

## Environments Node

A List of all active Environments and a list of all messages, errors and warnings logged within Shield.

### Environments Tab

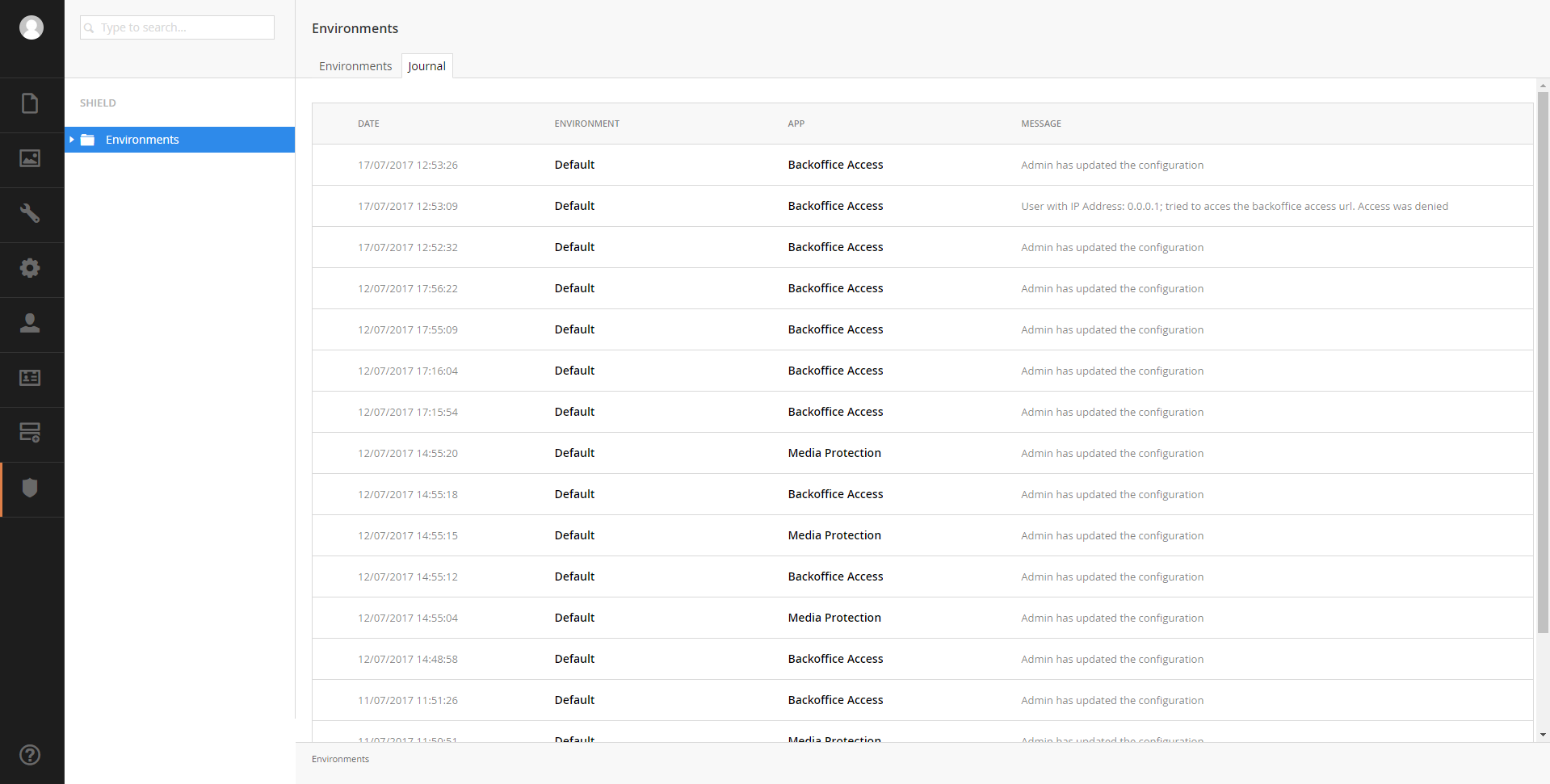
The environments tab will display a table listing the environments that have been created. Currently there is only one environment called “Default” which is used across your development, staging and production environments. At present, there isn’t the ability to create more environment(s) for Shield. The purpose of an environment will allow you to have different Shield apps that are installed to be configured differently dependant on the environment’s purpose, with the ability to define the domain(s) that are relevant for that environment.



### Journal Tab

The Journal tab will display all journal items (logs) that have been created by the different environment(s) & shield app(s). A Journal is composed of the following:

* Date & time of when the Journal item was created
* The environment of the app that created the Journal item
* The app that created the Journal item
* A message of why the Journal item was created.

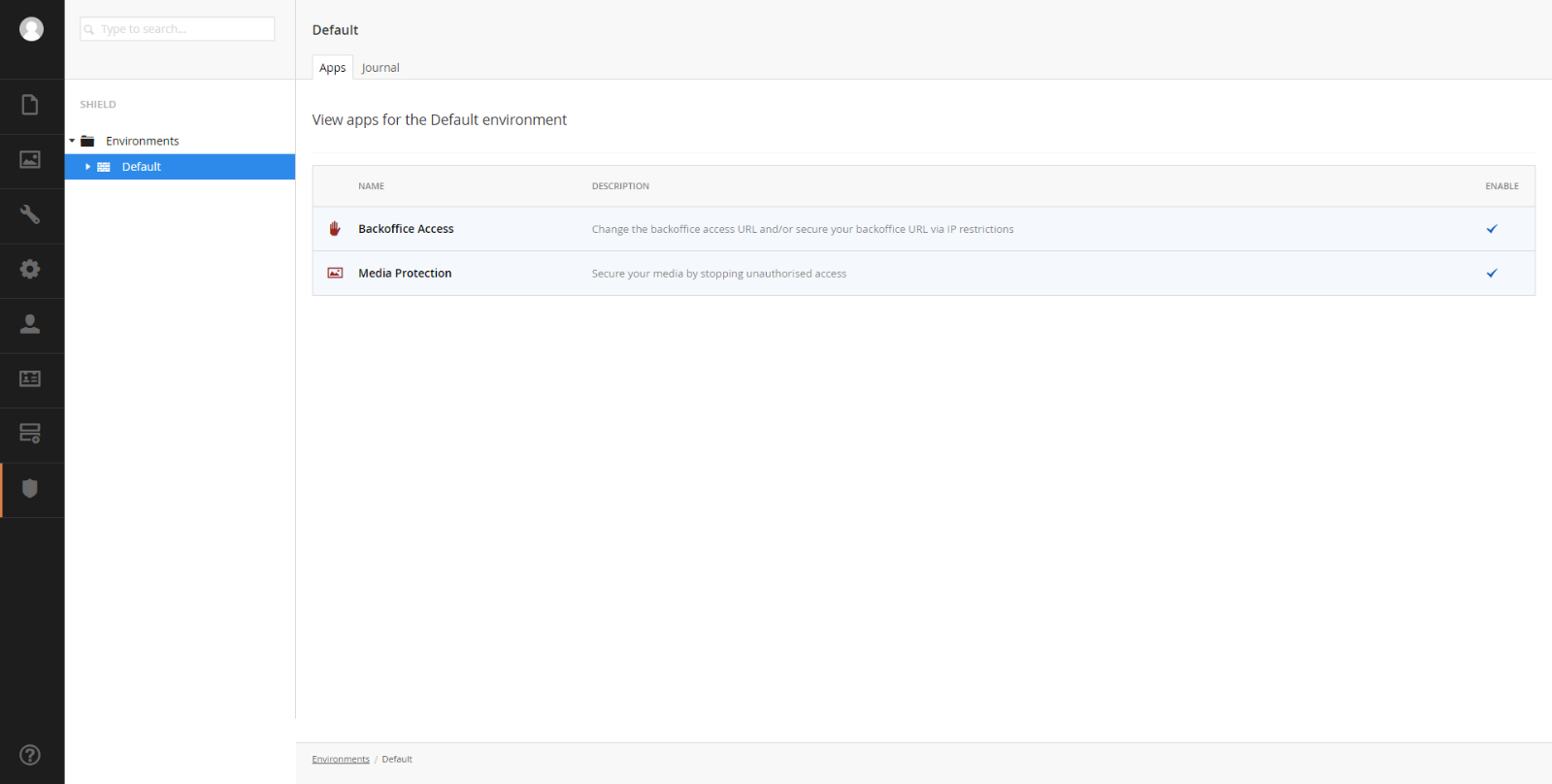


## Environment Node

The Environment node will display all installed apps on the “Apps” tab and a Journal on the “Journal” tab, see below for an explanation of these 2 tabs.

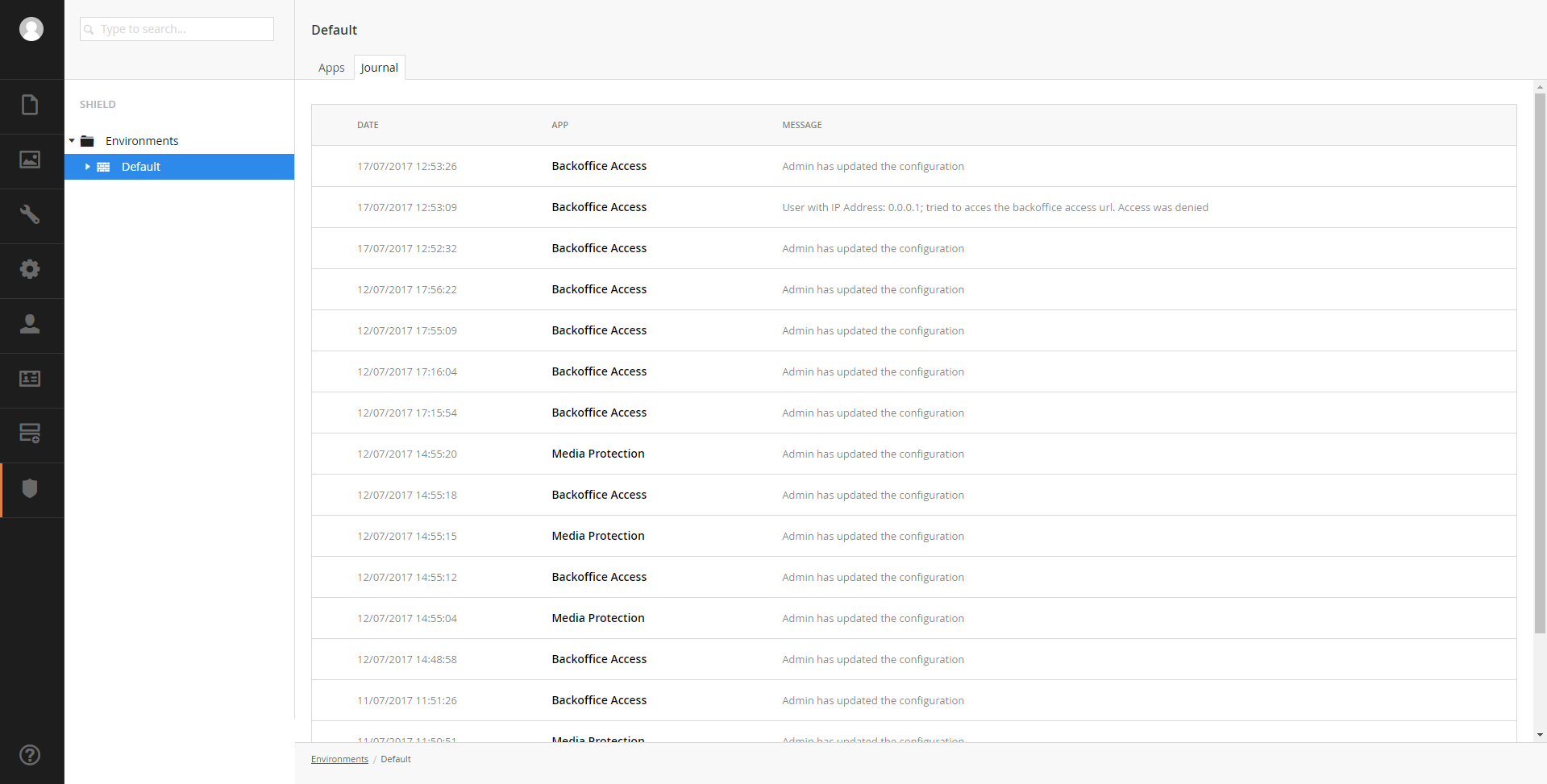
### Apps Tab

The Apps tab will display a listing of the Shield apps that are installed, showing the name, description and whether or not the app is enabled. Clicking on the app name will open up the app’s configuration.



### Journal Tab

Similar to the Environments node’s Journal tab, this will display the Journal items only for the selected environment. The difference being, the environment column is not included.

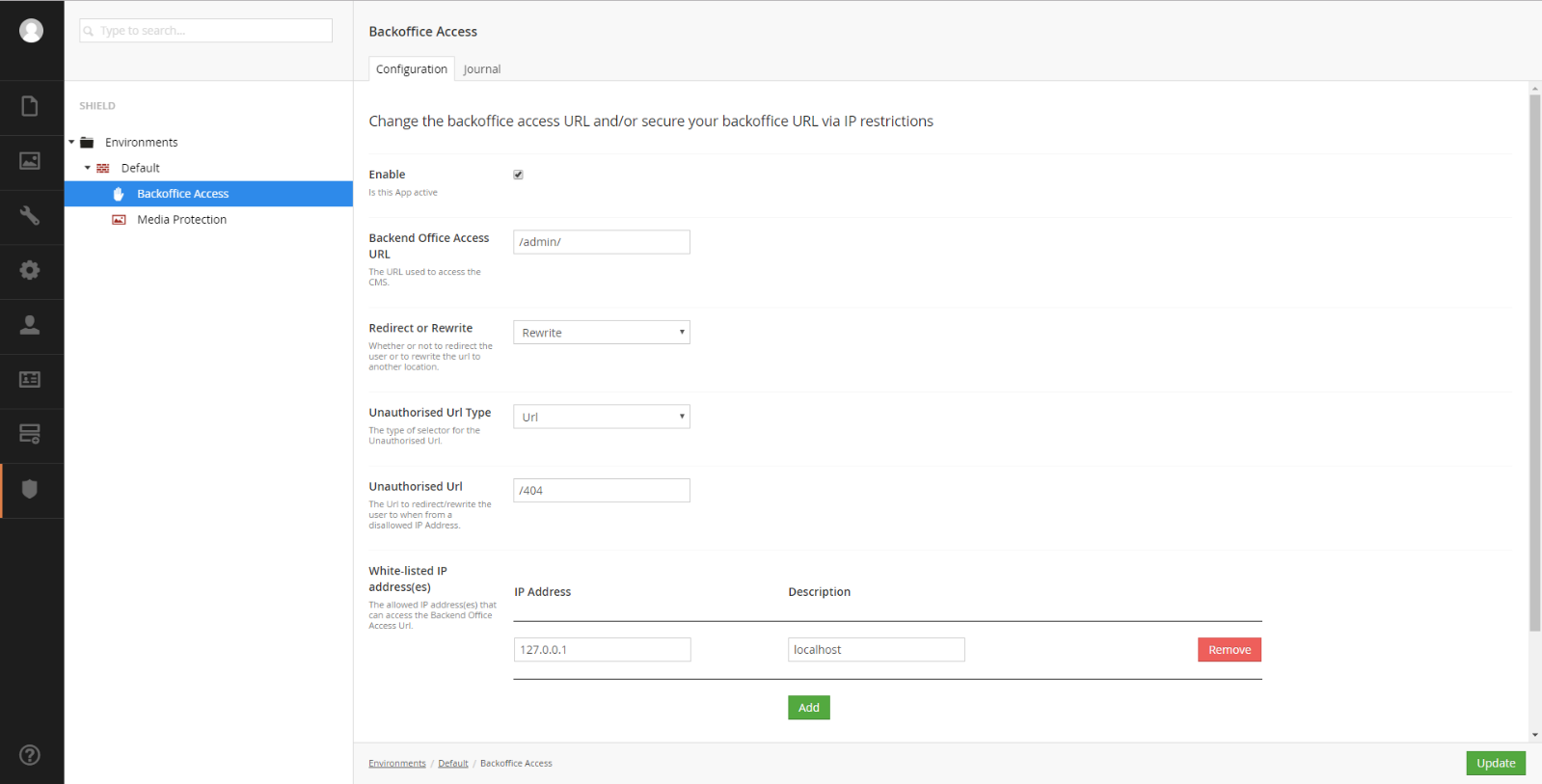


# Our.Shield.BackofficeAccess

Backoffice Access grants you the ability to change the backoffice access URL to a URL you desire, with the ability to restrict who can access the URL by a white-list of IP Addresses.

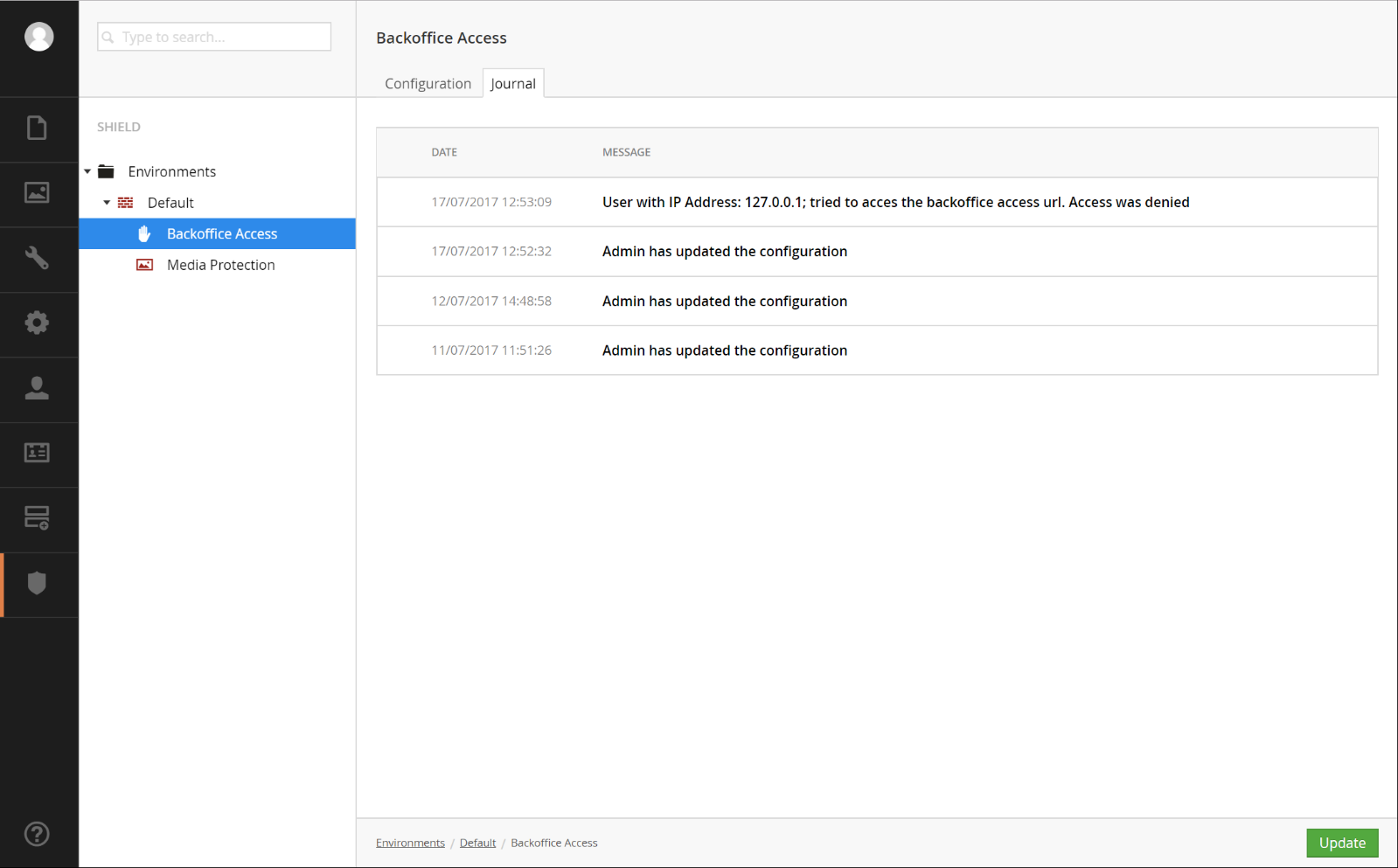
## Configuration Tab

* Enable or disable this app. When disabled the URL will return to the predefined default which is “/umbraco”.
* The backoffice access Url you wish to use to access the admin area of Umbraco. This can be any valid combination of letters or numbers, non-case sensitive. You are not allowed white space, symbols or special characters.
* The unauthorised action for users that are not within the white list of valid IP addresses. You can select between redirecting or rewriting to another webpage. The difference is that for the end user they will see their Url change within the address bar of their browser when redirecting, where rewriting the Url will not change.
* What type of page unauthorised users are directed to
* URL – Textbox to allow you to specify the page. i.e. /404
* XPath – Textbox to allow you to specify the path to the page. i.e. //standardPage[@isDoc AND @nodeName=’404 Not Found’]  
  OR
* Content Picker – Provides you a content section content picker to select the desired page.
* A white-list of either IP4 and/or IP6 Addresses of whom can access the desired backoffice access URL – if you required localhost, then you will need to add “127.0.0.1”. For your future reference a short description can be placed against each entry.



## Journal Tab

A list of warning, messages and errors that have occurred within this app. This includes all unauthorised attempts to gain access to the backoffice.

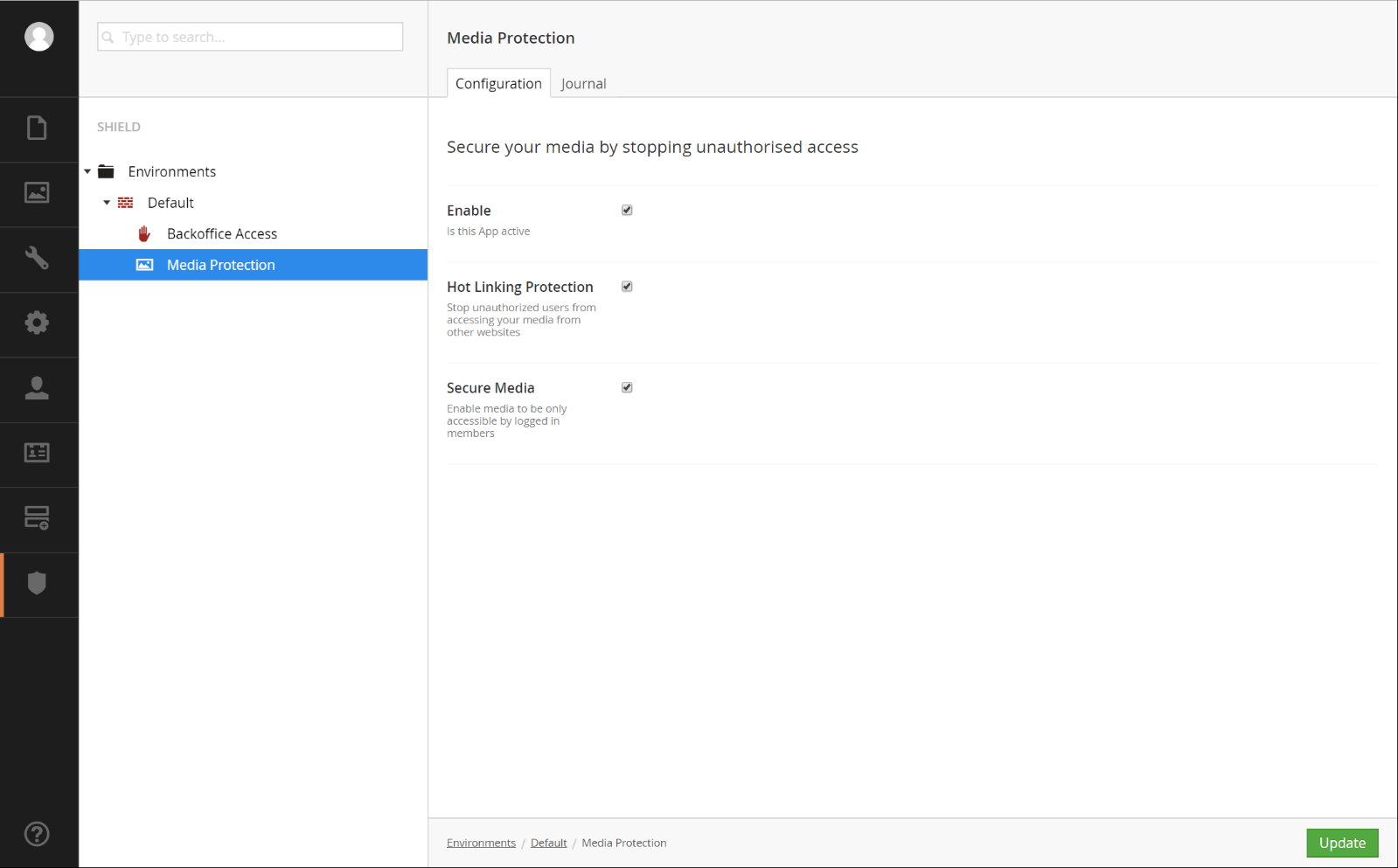


# Our.Shield.MediaProtection

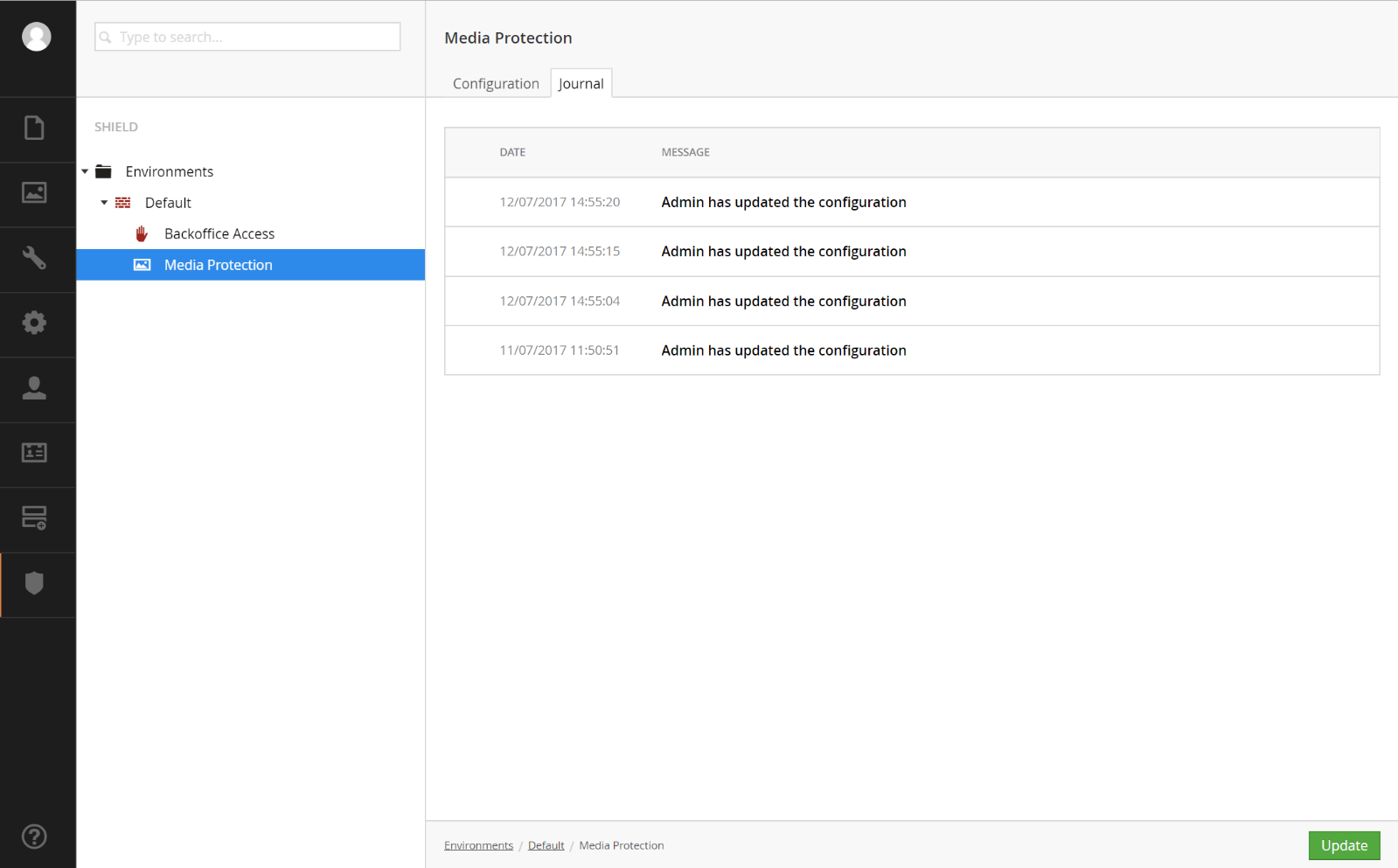
Media Protection gives you the ability to stop other websites from hot linking your media assets and allows you to assign media to only be viewed by authenticated members.

## Configuration

* Enable or disable this app. When disabled there will be no active security on your media assets.
* When Hot linking protection is enabled, no access is allowed to your media assets from any third party websites.
* When Secure Media is enabled, then any Secure Folder or File media items that they themselves have been specifically set to be Members only are restricted to your front-end users that have logged in.



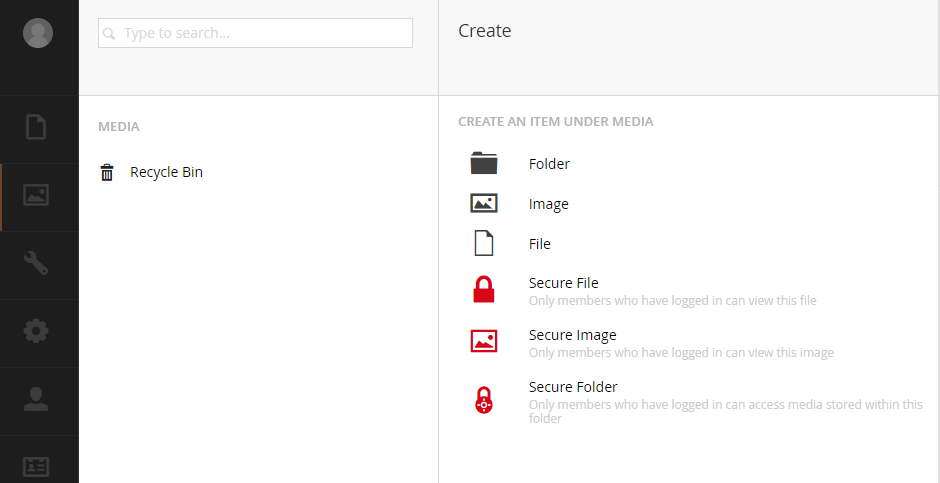
## Journal

A list of all errors, warning and messages from this app. 

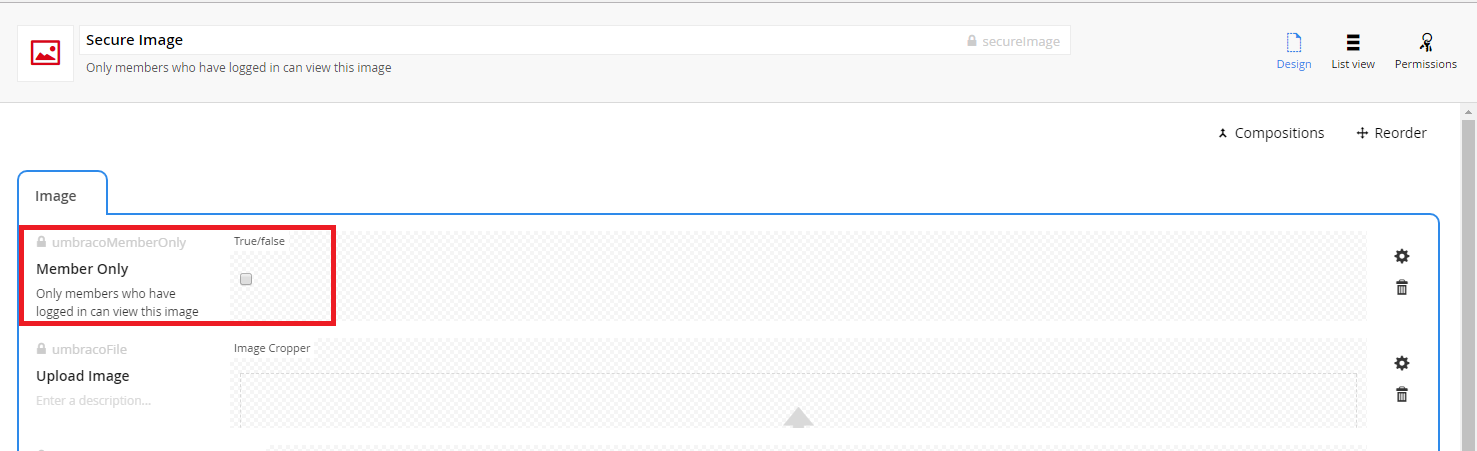
## Media Types

Once Media Protection has been installed, you should have 3 new media types to use:

1. Secure File
2. Secure Image
3. Secure Folder



These 3 new media types are used in conjunction with the Configuration’s “Secure Media” option. You’re able to create more secure media types by creating a new media type and having a property with a special alias of “umbracoMemberOnly” as type “True/False”.



## Secure Media

To enable the Secure Media to work as expected, you’ll need to create some new media items using one of the above mentioned new media types (or your custom secure media type(s) if you created any). Once the media items have been created, and the “Member Only” tickbox is checked, then the configuration’s Secure Media option is enable, only authenticated members can view the media item where the “Member Only” tickbox is checked.

Disabling Secure Media configuration option will allow access to the media items regardless of whether or not the “Member Only” tickbox on a media item is checked.

If you create a Secure Folder media item, and place all your media items in this secure folder, you’ll only need to check the “Member Only” tickbox on the Secure Folder item. Media Protection will look at the media item’s ancestors (parent nodes), and if an ancestor has the “Members Only” tickbox checked, then all its children are as well. For example, if you had the following media setup:

-Secure Folder

---Secure Image

---Image

---Image

And on the Secure Folder, you have the “Member Only” tickbox checked, then all the children, (the x1 Secure Image & the x2 Image) will only be accessible by authenticated users. The x1 Secure Image item itself doesn’t need the “Member Only” tickbox checked.

# Extending Shield with your own app

Within Visual Studio (or something similar) you’ll need to do the following:

1. Create a new project

2. Install UmbracoCMS & Our.Shield.Core into your project via NuGet

3. Create a class which will be the configuration for your app; the configuration is where you allow the app to be enabled/disabled, allow the end user to edit the configuration, to work the way they desire within the parameters of which your app offers.  
  
Create a class inheriting from **Our.Shield.Core.Models.Configuration**, you will need to decorate the class with Json attributes as this will be serialized/deserialized to and from your view and the database:   
  
Using Media Protection as an example  
  
[JsonObject(MemberSerialization.OptIn)]  
public class MediaProtectionConfiguration: Configuration  
{   
 [JsonProperty("enableHotLinkingProtection")]  
 public bool EnableHotLinkingProtection { get; set; }  
  
 [JsonProperty("enableMemberOnlyMedia")]  
 public bool EnableMembersOnlyMedia { get; set; }  
}

4. Create a class that will be your app, which uses your configuration created in the previous step.  
  
The new class will need to inherit from **Our.Shield.Core.Models.App** passing your configuration as a generic; using media protection as an example:

[AppEditor("/App\_Plugins/Shield.MediaProtection/Views/MediaProtection.html")]  
public class MediaProtectionApp : App<MediaProtectionConfiguration>  
{  
 public override string Id => "MediaProtection"  
  
 public override string Name => "Media Protection";  
  
 public override string Description => "Secure your media by stopping unauthorised access";  
  
 public override string Icon => "icon-picture red";  
  
 public override IConfiguration DefaultConfiguration  
 {  
 get  
 {  
 return new MediaProtectionConfiguration  
 {  
 EnableHotLinkingProtection = true,  
 EnableMembersOnlyMedia = true  
 }  
 }  
 }

public override bool Execute(IJob job, IConfiguration c)  
 {  
 var config = c as MediaProtectionConfiguration;  
  
 //We’ll come back to this in a later step  
   
 return true;  
 }  
}

5. Create the html page, as defined by the relative url in your AppEditor attribute, for best practices, I’d advise you to follow the guide for creating a custom package for Umbraco. i.e. all files should be stored within the ~/App\_Plugins/ directory with a package.manifest file

**[AppEditor("/App\_Plugins/Shield.MediaProtection/Views/MediaProtection.html")]**  
public class MediaProtectionApp : App<MediaProtectionConfiguration>  
{  
 ...  
}

And populate with the html that defines your configuration panel

<div ng-controller="Shield.Editors.MediaProtection.Edit as vm">  
 <div class="umb-el-wrap control-group umb-control-group">  
 <label for="hotlinking" class="control-label">  
 <span>Hot Linking Protection</span>  
 <small>  
 Stop unauthorized users from accessing your media from other websites  
 </small>  
 </label>  
 <div class="controls">  
 <input type="checkbox" id="hotlinking"

ng-model="vm.configuration.enableHotLinkingProtection" />  
 </div>  
 </div>  
  
 <div class="umb-el-wrap control-group umb-control-group">

<label for="securemedia" class="control-label">

<span>Secure Media</span>

<small>

Enable media to be only accessible by logged in members

</small>

</label>

<div class="controls">

<input type="checkbox" id="securemedia"

ng-model=" vm.configuration.enableMemberOnlyMedia" />

</div>

</div>  
</div>

6. Create a package.manifest file and populate with your js and css file(s) required by your html page above.

{

"javascript": [

"~/App\_Plugins/Shield.MediaProtection/Scripts/MediaProtection.js",

]

}

7. Create the angular module within the js file. This should also contain any validation your configuration requires.

angular.module('umbraco').controller('Shield.Editors.MediaProtection.Edit',  
 ['$scope', function ($scope) {  
 var vm = this;  
 angular.extend(vm, {  
 configuration: $scope.$parent.configuration  
 });  
 }]  
);

$scope.$parent.configuration will contain your serialized configuration created in step 3.

8. Now to return to the Execute method created in step 4.  
  
public class MediaProtectionApp : App<MediaProtectionConfiguration>  
{  
 ...

public override bool Execute(IJob job, IConfiguration c)  
 {  
 var config = c as MediaProtectionConfiguration;  
  
 ...  
  
 return true;  
 }  
}  
  
First off, you’ll want to return true when your app is disabled, this informs the Shield framework that your app has successfully ‘set up’ to provide security. If we were to return false instead, this will inform the framework to try again, we only want to return false in the worst-case scenarios:

public override bool Execute(IJob job, IConfiguration c)  
{  
 var config = c as MediaProtectionConfiguration;  
   
 if (config.Enable == false)  
 {

return true;

}  
   
 //this is where you’d want to put your security logic  
  
 ...  
  
 return true;  
}

9. Replace the comment “//this is where you’d want to put your security logic”, with whatever your needs are. Bear in mind though, you cannot access *UmbracoContext.Current* and certain other *Umbraco* classes within the method. However, adding a watcher on the HttpModule provides you a HttpApplication as one of the arguments to the anonymous function of which Job.WatchWebRequest requires.

public override bool Execute(IJob job, IConfiguration c)  
{  
 var config = c as MediaProtectionConfiguration;  
   
 //Unsubscribe any watches on the HttpModule, in-case the app is being disabled  
 job.UnwatchWebRequests();  
  
 if (config.Enable == false)  
 {

return true;

}  
   
 //Check if hotlinking protection is enabled  
 //and if so, subscribe a watch to the HttpModule  
 if (config.EnableHotLinkingProtection)

{  
 var mediaFolder = VirtualPathUtility.ToAbsolute(new  
 Uri(Umbraco.Core.IO.SystemDirectories.Media, UriKind.Relative).ToString()) + "/";

job.WatchWebRequests(new Regex(mediaFolder, RegexOptions.IgnoreCase), 50, (count, httpApp) =>

{

var referrer = httpApp.Request.UrlReferrer;

if (referrer == null

|| String.IsNullOrWhiteSpace(referrer.Host)

|| referrer.Host.Equals(httpApp.Request.Url.Host,  
 StringComparison.InvariantCultureIgnoreCase))

{

//This media is being accessed directly,

//or from a browser that doesn't pass referrer info,

//or from our own domain

//so allow access

return WatchCycle.Continue;

}  
  
 //Someone is trying to hotlink our media

job.WriteJournal(new JournalMessage($"Access was denied, {referrer.Host} is trying to hotlink  
 your media assets"));

httpApp.Response.StatusCode = (int) HttpStatusCode.Forbidden;

httpApp.Response.End();

return WatchCycle.Stop;

});

}  
  
 return true;  
}

You could attach your app to the content and/or media services:

public override bool Execute(IJob job, IConfiguration c)  
{  
 Umbraco.Core.Services.MediaService.Saved += MediaService\_Saved;  
 Umbraco.Core.Services.ContentService.Publishing += ContentService\_Publishing;  
 return true;  
}

private void MediaService\_Saved(IMediaService sender, Umbraco.Core.Events.SaveEventArgs<IMedia> e)

{

...

}  
  
private void ContentService\_Publishing(Umbraco.Core.Publishing.IPublishingStrategy sender,  
 Umbraco.Core.Events.PublishEventArgs<IContent> e)

{

...

}