



A premium module for DotNetNuke®

Reference Manual

Version 3.1.3

Last Updated: March 15, 2008

For the latest information and support, please visit:

<http://www.snapsis.com/support.aspx>

Contents

Chapter 1: About PageBlaster.....	1
Introduction	1
Why call it a Content Delivery Engine?.....	1
Chapter 2: PageBlaster Configuration.....	2
Installing PageBlaster	2
Common problems.....	4
Troubleshooting.....	4
Ok, it's installed. How do I use it?.....	5
Admin configuration.....	5
Page Specific Configuration.....	5
The Toolbar	6
Configuration Tree.....	7
Page Setup Section	8
Chapter 3: Performance Optimizations	12
Overview.....	12
Enabling the Static File Handler	12
Chapter 4: PageBlaster Replacements	14
Overview.....	14
Explanation	14

Snapsis PageBlaster - Content Delivery Engine

Page Rules 15

Portal Rules 15

Saved Rules 16

Replacement Tags 18

Appendix A: Replacement Tokens..... 20

Request Members 20

Request Members (Continued) 21

DNN.UserInfo 21

DNN.PageInfo 21

DNN.PortalInfo..... 22

Chapter 1: About PageBlaster

Introduction

The PageBlaster Http Module was developed as a solution to the problem inherent in most dynamic ASP.Net sites. Dynamic and readily updateable sites usually have to trade-off that functionality for performance. In other words, if we want to be able to update our web sites online in real time, or deliver personalized content to our users, we will also have to expect that our content will take longer to create each and every time a page is requested. PageBlaster solves this problem by dynamically caching each page and allowing a static “copy” of that page to also be saved on disk so that it can quickly be retrieved on the next request, and the next 1000’s of requests.

Why call it a Content Delivery Engine?

The PageBlaster Http Module is not just a control, or a module that sits within a web page. This Http Module actually makes it possible to get complete control over all of the output of all your website’s dynamically generated pages. This http module will allow you to deliver your content at rocket speed, and at the same time gives you the ability to manipulate every aspect of that content. Pages delivered with PageBlaster have multiple options for manipulating the output including dynamically setting the time that they are cached, compression, merging CSS & JS files, virtual page names (a.k.a. friendly Urls), and a very powerful replacement engine.

Have you ever wished you could change something within the page output like the doctype, or maybe just a CSS class name that is doing more harm than good, or how about when using SSL and you need to keep that pesky message box from showing? If so, you’ll love how easy PageBlaster makes these tasks, and all the while caching your pages so you don’t have to pay for that flexibility with performance.

Now, add to all that the ability to pull content in from the file system ([LoadFile](#)), or from another website ([WebCapture](#)) and the ability to Transform XML feeds ([TransformXML](#)), and you’ll have to agree that Content Delivery is what this module is all about.

Chapter 2: PageBlaster Configuration

Installing PageBlaster

PageBlaster for DNN can be installed by following the procedure outlined below:

1. Unzip the downloaded file to a folder on your hard drive.
2. If this is a new install, all you need to do is upload the module using the DNN installation process. Login to DNN as host and go to Host > Module Definitions > Install New Module. Browse for the folder that you unzipped the package to in step one, and choose the

Snapsis.DNN.PageBlaster_*.*.*_Install.zip file (where *.*.* = the version number)

After installation the module will automatically update the Web.config file to place the following lines into the http modules section. If you are upgrading then you may need to move this line so that it is above the UrlRewrite module instead of below it.

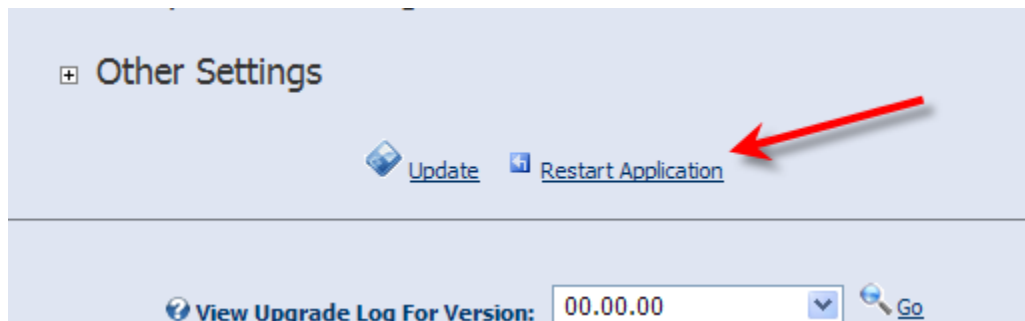
```
<!-- HttpModules for Common Functionality -->
<httpModules>
  <!-- add name="Authentication" type="DotNetNuke.HttpModules.AuthenticationModule, DotNetNuke.Ht
  <add name="ScriptModule" type="System.Web.Handlers.ScriptModule, System.Web.Extensions, Version:
  <add name="PageBlaster" type="Snapsis.HttpModules.PageBlaster.PageBlasterModule, Snapsis.HttpMo
  <add name="UrlRewrite" type="DotNetNuke.HttpModules.UrlRewriteModule, DotNetNuke.HttpModules" /:

  <add verb="*" path="*_AppService.axd" validate="false" type="System.Web.Script.Services.ScriptHand
  <add verb="GET,HEAD" path="ScriptResource.axd" type="System.Web.Handlers.ScriptResourceHandler, Sy
  <!-- <add verb="*" path="*.css,*.js,*.gif,*.jpg,*.jpeg,*.png"
  type="Snapsis.HttpModules.PageBlaster.StaticFileHandler, Snapsis.HttpModules.PageBlaster" /> -->
</httpHandlers>
```

The StaticFileHandler line is placed in the web.config, but left commented out so that you can enable it when you are ready. The Static File Handler will be required if you want to make virtual folders call up a specific page unless you use the “Save as Static File” option. Both of these options are explained in more detail later.

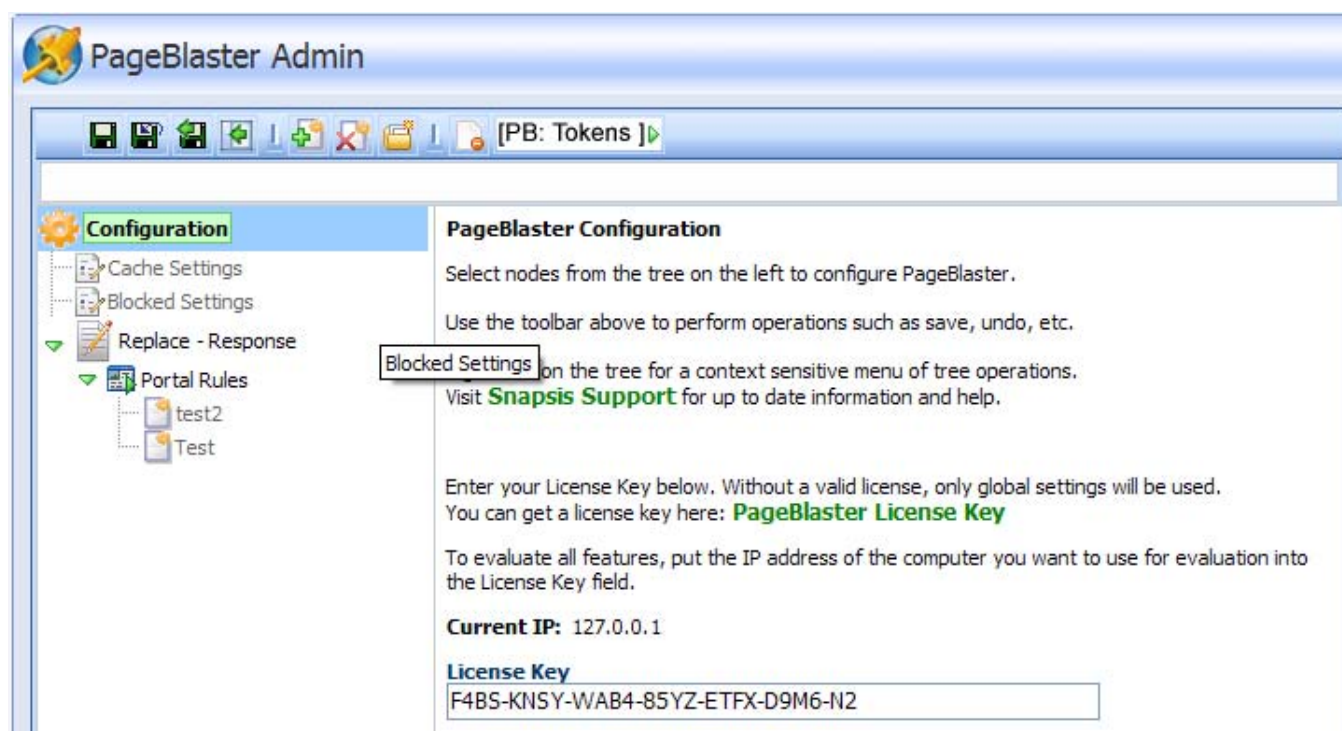
Snapsis PageBlaster - Content Delivery Engine for DNN®

3. Navigate to Host > Host Settings and click on “Restart Application” at the bottom of the page.



4. Navigate to Admin > PageBlaster and enter your license key. If you do not have a PageBlaster License key you can get one from:

<http://www.snapsis.com/PageBlaster.aspx>



Entering your Current IP address in the license key field will allow you to evaluate the module in Full Professional mode using a single computer.

Common problems

Erratic Page output

Some people experience erratic behavior with their site after installing PageBlaster. What they see is repeated output, or no output, or a continuous cycling of the page. This happens when the page output contains more than one `</html` tag. The closing html tag is a signal to PageBlaster that the end of the page has been reached. If there is more than one, then critical information may be lost from the content after the first one. To correct this problem, you will usually find that your skin has been created as a full html document. DNN skins should only contain the code inside and excluding the `<body` tags. Modifying your skin will fix this problem.

The page does not display correctly – missing CSS

There may be a problem in the merging of the CSS files, you can exclude the merge process by un-commenting the line in the config file that excludes the MergeCSS process.

```
<path pattern=".*" excludeFrom="MergeCSS" />
```

The page has javascript errors – bad JS

There may be a problem in the merging of the JS files, you can exclude the merge process by un-commenting the line in the config file that excludes the MergeJS process.

```
<path pattern=".*" excludeFrom="MergeJS" />
```

Medium Trust

You may see an error like the following:

```
[SecurityException: Request for the permission of type  
'System.Security.Permissions.SecurityPermission, mscorlib, Version=2.0.0.0, Culture=neutral,  
PublicKeyToken=b77a5c561934e089' failed.]
```

```
Snapsis.HttpModules.PageBlaster.PageBlasterModule.Init(HttpApplication app) +0
```

This means that your hosting environment does not allow the tracing diagnostics. Please contact Snapsis Support to get a compatible build for your environment.

Troubleshooting

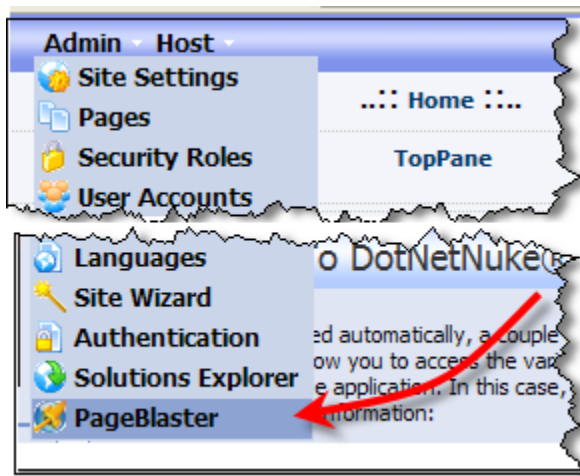
If the module has been installed correctly and still does not seem to be working you can turn on Debug tracing to send information to a log file. To do this locate the `<trace` node in the `Snapsis.PageBlaster.Config` file and set the `level="Debug"`. You can also change the file-path if you wish to log to a different file. After setting the level to Debug, you will need to restart the application to make this setting take effect. You can restart the application in Host > Host Settings at the bottom of the page is an option to restart application. If you use debugging you will want to remember to come back and turn it off because the file has a lot of information added to it on each request and can get large rather quickly. To turn off debugging, set the level attribute back to "None". You can also set the level to "Header" if you want to see PageBlaster response header information.

Ok, it's installed. How do I use it?

After installing PageBlaster for DNN, you will find it in the drop-down list of modules that can be added to a page, and also on the Admin Menu > PageBlaster.

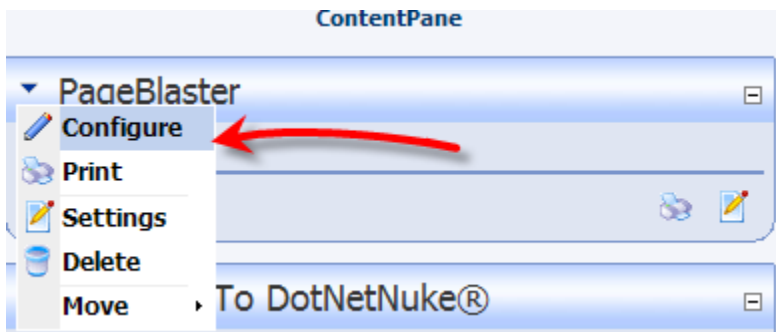
Admin configuration

The admin menu allows you to configure caching and replacements that will be executed on every page in the portal.



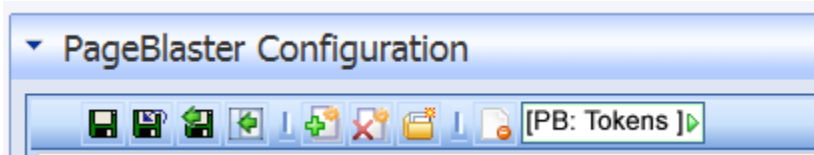
Page Specific Configuration

You will need to use the module on a page to set a virtual path, or to modify specific cache and replacement settings for a single page. These settings will over-ride the settings in the global config and/or the admin config. To do this, add the module to the page that needs specific settings, and edit the configuration by clicking on Config in the Action menu as shown in the image below.





You may also want to set the container to not be visible by adjusting the module settings since this module does not have any visible output.


The Toolbar





The toolbar is used for general functions. The icons from left to right do the following:


 **Save:** Each setting in the PageBlaster configuration can be changed without having to make a call to the server. This button saves the current configuration of all information to the server.


 **Undo Last Save:** If you have made some changes, and have not yet saved them then you can use this button to reset the form and pull the last saved configuration from the server.


 **Save and Exit:** Use this button to Save and simultaneously exit the configuration screen. This is handy for a quick change, but you will miss any messages that are returned from the server about the change that was made.


 **Exit without saving:** Use this button if you decide not to commit the changes to the server and want to exit (cancel).

 **Add Rule:** Use this button to add a new rule to the Tree when working in the replacements section.

 **Delete Rule:** Use this button to delete a rule. The rule will not actually be deleted from the configuration until you save, so if you make a mistake with this button, then you may want to choose this time to use the Undo Last Save button.

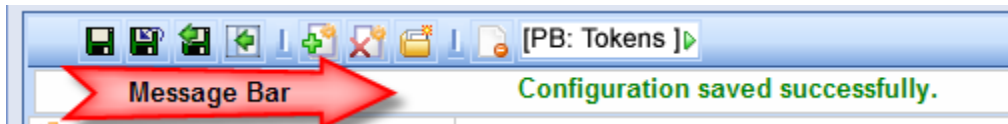
 **Saved Rules:** Use this button to load a new node in the configuration tree that will allow you to copy rules to be saved for use on other pages, or in the admin configuration. The saved rules do not get executed.

 **Delete Cache:** Use this button to remove the current page from the PageBlaster cache. If you are using the admin configuration then this button will remove all files from cache.

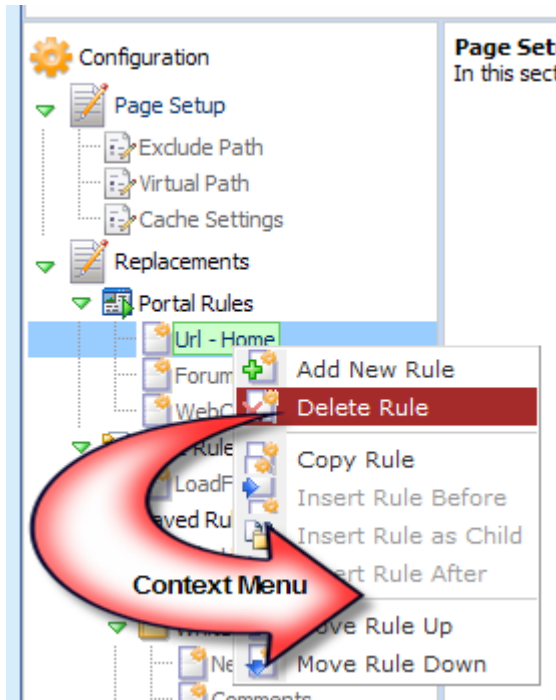
 **[PB: Tokens]:** This button will open a new window which contains a list of replacement tags that can be used in the Search For and Respond - Replace With fields of the replacement rules. You can find out more about replacement rules in [Chapter 3](#).

Message Bar

The message bar is directly below the toolbar. This space is used to display messages concerning the operation that last took place.



Configuration Tree



The configuration tree is used to quickly navigate to the desired configuration section and also provides a context menu and drag-drop operations to make setup a breeze.

Context Menu

When you right-click on the nodes in the replacements section you will be given options for dealing with the replacement rule that you clicked on. One main use of this menu is to re-order your rules, because all rules are executed in order from top to bottom. This gives you the ability to create more complex rules from building on a previously executed one.

You can also use the context menu to copy the rules into different folders. If a rule is placed inside another rule it will make the parent rule a folder. This helps to organize your rules into "RuleSets". The parent rule is no different

from the child rules except that it will be executed first, and then all of its child rules, then on to the next parent.

Drag-Drop

Rules can be clicked on a dragged to another location within the Replacements section. Dragging a rule between the three main areas (Page Rules, Portal Rules, and Saved Rules) will **copy** the rule to the section it is dropped on. This feature is implemented in this way to facilitate saving your rules for use in other PageBlaster instances, or for using a rule from the Saved Rules. Saved Rules are not executed. That folder is just a place to "save your work" for later use, so if you have a rule that isn't working correctly, or you just want to see what happens when it is not executed you could just drag it into the Save Rules section and then delete the original. If you drag and drop a rule in the same section then it will **move** that rule so that it is executed in a different order.

Page Setup Section



Exclude Path

Exclude Path Settings

Here you can set this page to be excluded from PageBlaster operations if the path in the querystring matches your expression below.
A Regular Expression pattern can be used or a simple literal string.

Exclude Path (pattern):

Exclude From:
None
None
Compression
Caching
CompressionAndCaching
Replacements
MergeCSS
MergeJS
MergeCSSAndJS
JSMin
CSSMin
All

Specify a portion of the Url to look for and PageBlaster will compare it using [Regular Expressions](#).

For example, if you wanted to exclude the replacements operation while editing content, and the Url has `ctl=edit` in it when this happens, then you could put **ctl=edit** in the exclude path and choose Replacements in the drop-down. Or if you wanted to exclude any Url with the word admin and the word edit in it, then you could use **admin.*edit** as your exclude path.

For the CSS and JS the path to the file is also used for comparison, so if for example you wanted to only exclude jQuery from minification then you would put jQuery in the exclude path and choose JSMin as the Exclude From option.

Virtual Path

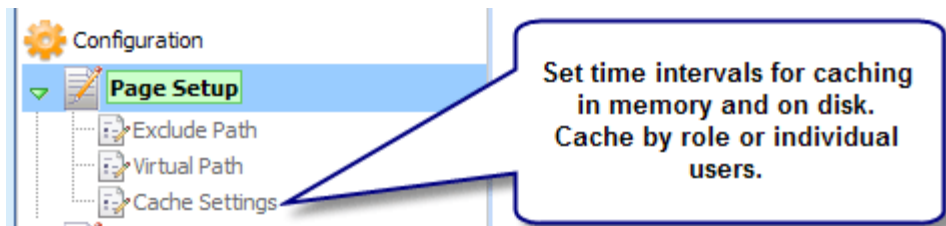


The Virtual Path setting is a way for you to specify a Url for reaching this page that is exactly how you think it will work best for humans and or search engines. It does not work off of a rewriting scheme where all the Urls get re-written automatically so it can be used alongside the “friendly” Url rewriter you may already be using. Keep in mind that the Virtual Path you enter here is only a mapping to a tabid. It is not a Url rewriter. So if you want to do a full Url rewrite then you will also need a replacement rule.

Save path as a static file

When creating a virtual path to a page, you also have the option to save it as a static file. The file name must have the .htm extension. This allows you to make a page that is accessible using only a virtual folder. If this option is checked then the physical folders will be created and a page will be created in the folder corresponding to the name you choose. Most websites use default.htm or index.htm as a default document that will be delivered if just the folder name is used on the url.

Cache Settings



Caching can be set for all pages in the portal by using the admin configuration or for each individual page if you put a PageBlaster module on it. This is something that is not possible in most content management systems because they run from a single ASP.Net page on the server.

Memory Interval:

Days: Hours: Minutes: Seconds:

Disk Interval:

Days: Hours: Minutes: Seconds:

Each page can be cached in memory and/or on disk. The settings you use will depend on your environment, but some general advice is given here to help in making those settings work for you.

In general, the more traffic you have, the lower you can set your cache times and still receive huge benefits. Conversely, if you have low traffic then you will want to increase the cache time to make it effective. The thing to think about is how long you need to set the cache interval to get the same page reused many times without it having to be rebuilt. PageBlaster allows you set this time very high and still be able to refresh the cache if updates are made to pages. Anytime a user is logged in and a post-back is made to the server (something that usually happens during an update), PageBlaster will automatically refresh the cache for that page. Also, if you want to force a refresh to the cache on a page at any time then you can do so from any browser by holding the CTRL key and refreshing the page (CTRL +f5 in Internet Explorer or Shift+f5 in FireFox).

Cache Parameters

Specify additional cache parameters to determine how PageBlaster stores the cached files, which in turn helps the engine to better update the cache.

Page Parameters

Additional Cache Key Parameters

Page Parameters

One or more querystring variables separated by semi-colons. Tabid is the default page parameter, but some modules in DNN produce many pages within a single tabid. For instance, if you have a Forum module on the page then you may want to set it up so

that each post is considered a unique page. The reason this is helpful is because when a page is removed from cache, all the different instances of that page will also be removed. So instead of making all of your posts get removed from cache you can specify additional page parameters here and only specific posts will be cleared when needed. If you look in the PageBlaster cache folder you can see that the value for each page parameter is used as the folder name.

Snapsis PageBlaster - Content Delivery Engine for DNN®

Additional Cache Key Parameters

The additional parameters can be specified with [PB:Request Tokens ([see the replacement tokens in Appendix A](#)), which allow you to specify cookies, form variables, etc. The additional parameters do not create additional folders in the disk cache, but they can help you to produce unique cached pages for other reasons, like when a browser does not support Javascript (spiders).

Cached Roles & Users

By default, PageBlaster only caches users that are not authenticated (not logged in). But you may find that you have certain roles or users that can also benefit from caching.

Specify the roles or users you want to cache in the following fields. These fields, like many others in the configuration also use Regular Expressions to make it very robust.

Cached Roles:

Cached Users:

To cache all users you could just put the expression `.*` into the Cached Roles field. Although if you have a high traffic site this is not recommended because you would certainly have a diminishing rate of return on the number of times each cached page would get used.

Blocked Settings

The blocked settings section allows you to block requests either by IP address or by user-agent. You can enter partial IP addresses, or use regular expressions in either of these fields. Separate multiple entries by using a semi-colon.

Chapter 3: Performance Optimizations

Overview

You may already realize that PageBlaster can do Http Compression and Caching which dramatically improves the performance of your dynamic website. New in version 3, PageBlaster can also automatically perform some additional optimizations to your website pages. Several of these optimizations are recommended by the Yahoo! Exceptional Performance team and can be measured using the YSlow! Plug-in for FireFox. If you haven't yet loaded this plug-in, then I highly recommend that you head on over to get it now: <http://developer.yahoo.com/yslow/> also be sure to read the great information on performance: <http://developer.yahoo.com/performance/>.

Minimizing Http Requests & Minimizing CSS & JS

PageBlaster will automatically merge all the files for CSS and JS links in the <head> section of your page. After merging these files together, PageBlaster will also perform Minimization on the content of those files by removing comments, line breaks and extra white space. After the files have been merged and minimized they will also be compressed and then cached. This effectively performs all the optimization on your CSS & JS files “on-the-fly” without having an impact on overall performance.

Only the files that are linked to in the head section are processed because it could cause un-intended problems if the links from the body of the page were moved in relation to their relative position in the page layout. However, you can still optimize the static files that are linked in throughout the page by using a replacement rule to move the link to the head section. If these links cannot be moved then you can also optimize these files individually by enabling the PageBlaster Static File Handler.

Enabling the Static File Handler

The Static File Handler is used if you want PageBlaster to also process .JS files & .CSS files that are not in the Head section and/or images. You will also need to enable the static file handler if you want to use Urls that do not end with an extension. To enable the static file handler you need to first decide what extensions you want PageBlaster to handle and then go into web.config and update the following line in the httpHandlers section.

```
<add verb="*" path="*_AppService.axd" validate="false" type="System.Web.Script.Services.ScriptHand
<add verb="GET,HEAD" path="ScriptResource.axd" type="System.Web.Handlers.ScriptResourceHandler, Sy
<!-- <add verb="*" path="*.css,*.js,*.gif,*.jpg,*.jpeg,*.png"
type="Snapsis.HttpModules.PageBlaster.StaticFileHandler, Snapsis.HttpModules.PageBlaster" /> -->
</httpHandlers>
```

After you have allowed these extensions to be processed by PageBlaster you will also need to modify the IIS settings for your website so that the Application Mappings for those extensions are sent to ASP.Net. The images below show how to setup application mappings in IIS5 on WinXP Pro. If you do not manage your own web server, or you are on a shared host, then you will need to have the application mapping completed by the person that manages your IIS server.

Snapsis PageBlaster - Content Delivery Engine for DNN®

3. Enter the extension. After entering extension you need to click back in the executable area to get the OK button to enable.

2. Click on Browse to find the aspnet_isapi.dll for the version of ASP.Net you are running

4. Click on OK.

1. Click on Add

The image shows three overlapping windows from the Snapsis PageBlaster application. The background window is the 'Application Configuration' dialog, with the 'Mappings' tab selected. It contains a table of application mappings and an 'Add' button. The middle window is the 'Add/Edit Application Extension Mapping' dialog, which has fields for 'Executable' (set to 'C:\WINDOWS\...aspnet_isapi.dll'), 'Extension' (set to '.css'), and 'Verbs' (set to 'All Verbs'). It also has checkboxes for 'Script engine' and 'Check that file exists', and 'OK', 'Cancel', and 'Help' buttons. The foreground window is an 'Open' file dialog showing the file system structure, with the 'v2.0.50727' folder selected under 'Microsoft.NET Framework'. The file 'aspnet_isapi.dll' is selected in the file list, and the 'Files of type' is set to 'Dynamic Link Libraries (*.dll)'. Red arrows point from the text instructions to the corresponding UI elements in the windows.

Chapter 4: PageBlaster Replacements

Overview

Ok, now that you have your site running at its optimum, we can now afford to spend a little time on making it more compliant and better for SEO. PageBlaster replacements are a powerful feature that allows you to change the output of your page right before it is sent to the browser.

The thing that makes PageBlaster replacements so flexible and powerful is that it uses Regular-Expressions which have been around for a long time and have been optimized to do this chore. If you don't know about Regular Expressions, I would suggest heading over to <http://www.regular-expressions.info/> to get your feet wet. RegEx can be a little intimidating, but if you give yourself a little time to get over the curve you'll be glad you added that tool to your toolbox. The nice thing is, you don't have to be an expert to use the Replacement features of PageBlaster, and if you ever need help, just head on over to <http://www.snapsis.com/support.aspx>.

Explanation

The simplest way to explain what the replacement rule feature can do is to relate it to a manual operation most of us do every day. When you use the standard search and replace function of any application like MS Word or your favorite Html editor, then you are essentially building a replacement rule. On the input form for an individual rule you will notice that there is a Search For field and a Replace With field.

Rule Settings
Set the properties of this rule to control how and when text in this page will be replaced.

Rule Name:

Search For:

Replace With:

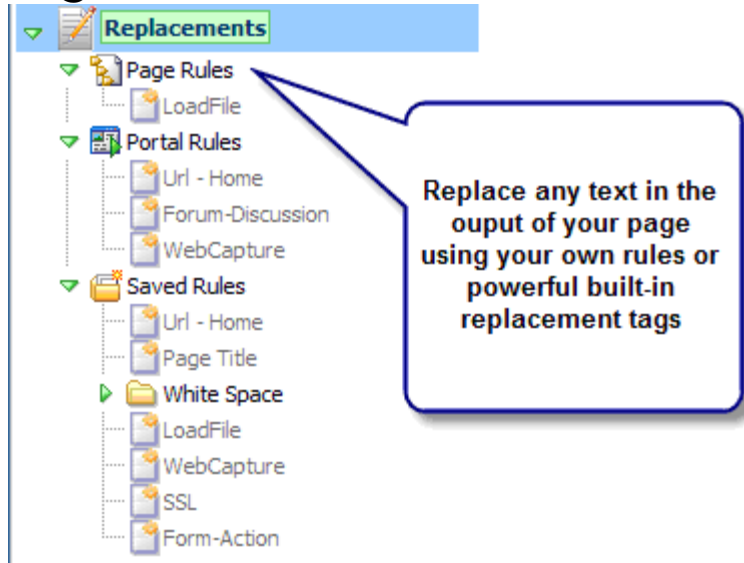
What PageBlaster does is allow you to save this search and replace operation to be executed automatically on a page before it is delivered to the browser. So taking for example the Rule in the image above, the engine will search for a for a friendly Url in the style output by DNN® and replace that with a much friendlier page name.

Snapsis PageBlaster - Content Delivery Engine for DNN®

Building on that simple example, the real power comes in when you have the ability to save many instances of these search and replace operations and can define their order of execution. This gives you the ability to create a “scrubbing filter” to clean up any errant html and make it W3C compliant if you like.

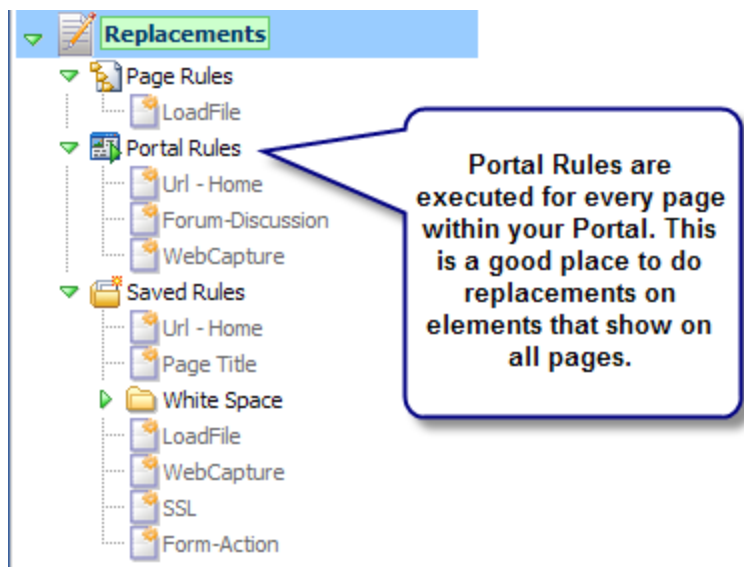
The configuration tree for replacements is broken into three sub-sections. Page Rules, Portal Rules, and Saved Rules.

Page Rules



Page Rules will be executed only on the page that this module is inserted into. If you wanted to define some rules that are security / role specific you could put the rules into this section and then set the module security so that the rules only get executed when that module is displayed on this page.

Portal Rules



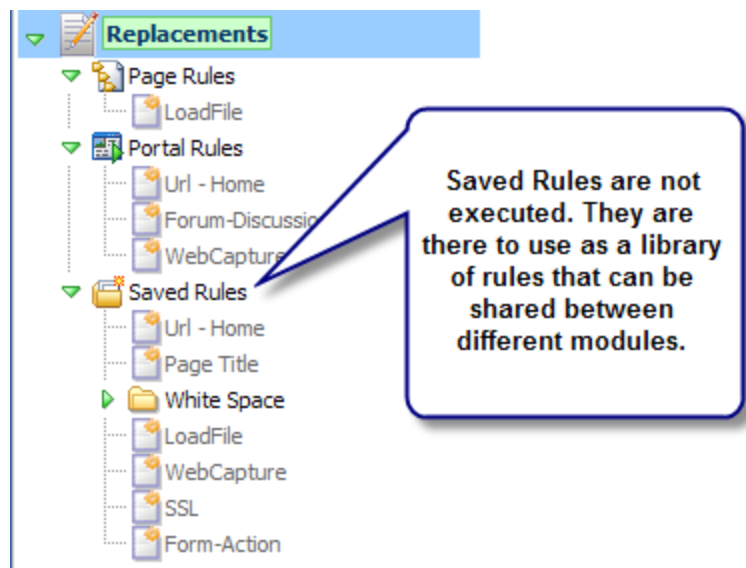
Portal Rules are executed on every page in the current portal (even the admin pages). This section will also abide by any security settings on the module, so if you want to make sure that the rules in this

Snapsis PageBlaster - Content Delivery Engine for DNN®

section are executed regardless of the security that was set for the Page Rules, then you may want to create a separate module to put your Portal Rules in.

There is no reason to display this module on all pages to get your rules to be executed for all pages of the portal. Although, if you wanted to make rules execute on all pages except the admin pages (which will not use the module even if it is set for all pages) then you could set a module to display on all pages and put your rules in the Page Rules section instead.

Saved Rules



Saved Rules are never executed as long as they are in this section. This section gives you a place to save any rules that you want to share between different instances of the module.

Anytime you want to use a saved rule, just drag it from this section and drop it into another section. You can also right click and use the context menu for more precise placement.

Conditional Rules

A conditional rule basically consists of a special PageBlaster token syntax that allows an IF condition to be evaluated before executing the rules that it contains. The syntax of an IF function is as follows:

```
[PB: IF( expression1 evaloperator expression2 ) { true condition } else { false condition } ]
```

Where:

expression1: can be a literal term or the result of an inner function

Inner functions are:

Match (*expression1, expresion2*) where *expression2* can use regular expressions

Len (*expresion1*)

Left (*expresion1,length*)

Right (*expresion1,length*)

Substring(*expresion1, start, length*)

evaloperator: can be any of the following operators: =, ==, !=, <>, >=, <=, <, >

equals, not equals, greater than or equal to, less than or equal to, less than, greater than

expression2: same as expression 1

- Expressions can be literal or you can use PB: tokens
- The condition statements are enclosed in curly braces
- The eval operator and expression are optional if you are using an inner function that evaluates to true or false.
- The else part with the false condition is optional

So an example conditional statement that could be used to match on a Url and do a 301 redirect would look like this:

```
[PB: IF( Match("[PB:Request.Url]","http://snapsis.com") ) {  
  
    [PB: Response.Status("301 Moved Permanently") ]  
  
    [PB: Response.AddHeader("Location","http://www.snapsis.com") ]  
  
    [PB: Response.End() ]  
  
} ]
```

As you can see the special function type tokens are used to write logical code much like you would use in programming and get direct access to the request and response object.

Replacement Tags

Replacement Tags can be used to get specific information from the server and used as part of your Search and Replace routines. A full listing of Replacement Tags is defined in [Appendix A](#).

Replacement Tags can also be used free-form in any part of your content. You can insert the tags directly into any module's output, use them as part of your skins, or put them directly into external pages that are pulled in from external sources.

The DNN Replacement Tags are pretty self-explanatory, but there are a few additional Tags that deserve a closer look because they will really help you in delivering and maintaining the output of your pages.

Load File [RE:LoadFile(*filepath*)]

The LoadFile Replacement Tag is used for pulling any file from the file system and displaying it as content. It requires one parameter for the path of the file which can be specified as a relative mapping (e.g. /portals/o/MyHtml.htm), or as a physical mapping (e.g. E:\Websites\MyWebsite\MyFile.htm)

The content of this file will also be parsed for replacement tags so you could use this to build a complete template system.

Web Capture

[RE:WebCapture(*url,searchFor,replaceWith, method,params*)]

The Web Capture Replacement Tag allows you to grab content from another Url and optionally parse the capture for a specific starting and ending tag.

The parameters for this method are:

url : *required* = the Url that you want to capture the output from.

searchFor: *optional* = the text that you want to search for within the content of this capture.

replaceWith: *optional* = the replacement that you want to use in this capture.

method: *optional* = GET, POST, or HEAD

params: *optional* = A parameter string of name=value pairs to send (used for a post method to send data).

The searchFor and replaceWith parameters of this method use regular expressions so that you can pinpoint the exact information you want to capture from a Url.

Transform XML [RE:TransformXML(xmlpath,xslpath)]

The Transform XML Replacement Tag works like the XML module in DotNetNuke®. The reason it is included in PageBlaster is because you will now be able to use this Replacement Tag in any content output from your pages. So you could have a skin that uses this tag to display a feed from your host site into your child portals, or any number of other uses. The parameters for this method are:

xmlpath: *required* = the path to the Xml document

xslpath: *optional* = the path to the Xsl document

Look for examples of these methods online at <http://www.snapsis.com/support.aspx>

Appendix A: Replacement Tokens

Request Members

```
[PB:Request.IsAuthenticated]
[PB:Request.IsSecureConnection]
[PB:Request.Params("variable")]
[PB:Request.Form("variable")]
[PB:Request.Url.IsLoopback]
[PB:Request.Url.AbsolutePath]
[PB:Request.Url.AbsoluteUri]
[PB:Request.Url.Authority]
[PB:Request.Url.Fragment]
[PB:Request.Url.Host]
[PB:Request.Url.HostNameType]
[PB:Request.Url.IsDefaultPort]
[PB:Request.Url.IsFile]
[PB:Request.Url.IsUnc]
[PB:Request.Url.PathAndQuery]
[PB:Request.Url.Query]
[PB:Request.Url.AbsoluteUri]
[PB:Request.QueryString("variable")]
[PB:Request.Cookies("variable")]
[PB:Request.Browser.ActiveXControls]
[PB:Request.Browser.AOL]
[PB:Request.Browser.BackgroundSounds]
[PB:Request.Browser.Beta]
[PB:Request.Browser.Browser]
[PB:Request.Browser.CDF]
[PB:Request.Browser.ClrVersion]
[PB:Request.Browser.Cookies]
[PB:Request.Browser.Crawler]
[PB:Request.Browser.EcmaScriptVersion]
[PB:Request.Browser.Frames.ToString()]
[PB:Request.Browser.JavaApplets]
[PB:Request.Browser.JavaScript]
[PB:Request.Browser.MajorVersion]
[PB:Request.Browser.MinorVersion]
[PB:Request.Browser.MSDomVersion]
[PB:Request.Browser.Platform]
[PB:Request.Browser.Tables]
[PB:Request.Browser.Type]
[PB:Request.Browser.VBScript]
[PB:Request.Browser.Version]
[PB:Request.Browser.W3CDomVersion]
[PB:Request.Browser.Win16]
[PB:Request.Browser.Win32]
[PB:Request.ServerVariables("variable")]
```

Request Members (Continued)

```
[PB:Request.RawUrl]
[PB:Request.UrlReferrer.AbsolutePath]
[PB:Request.UrlReferrer.AbsoluteUri]
[PB:Request.UrlReferrer.Authority]
[PB:Request.UrlReferrer.Host]
[PB:Request.UrlReferrer.IsFile]
[PB:Request.UrlReferrer.IsUnc]
[PB:Request.UrlReferrer.PathAndQuery]
[PB:Request.UrlReferrer.Query]
[PB:Request.UrlReferrer.AbsoluteUri]
```

DNN.UserInfo

```
FirstName = [PB:DNN.UserInfo.FirstName]
LastName = [PB:DNN.UserInfo.LastName]
DisplayName = [PB:DNN.UserInfo.DisplayName]
LastLoginDate = [PB:DNN.UserInfo.LastLoginDate]
Email = [PB:DNN.UserInfo.Email]
UserId = [PB:DNN.UserInfo.UserId]
Roles = [PB:DNN.UserInfo.Roles]
Username = [PB:DNN.UserInfo.Username]
Telephone = [PB:DNN.UserInfo.Telephone]
Street = [PB:DNN.UserInfo.Street]
Unit = [PB:DNN.UserInfo.Unit]
City = [PB:DNN.UserInfo.City]
Region = [PB:DNN.UserInfo.Region]
State = [PB:DNN.UserInfo.State]
Country = [PB:DNN.UserInfo.Country]
CreatedDate = [PB:DNN.UserInfo.CreatedDate]
```

DNN.PageInfo

```
AdministratorRoles = [PB:DNN.PageInfo.AdministratorRoles]
AuthorizedRoles = [PB:DNN.PageInfo.AuthorizedRoles]
```

Snapsis PageBlaster - Content Delivery Engine for DNN®

```
BreadCrumb = [PB:DNN.PageInfo.BreadCrumb]
ContainerPath = [PB:DNN.PageInfo.ContainerPath]
ContainerSrc = [PB:DNN.PageInfo.ContainerSrc]
Description = [PB:DNN.PageInfo.Description]
DisableLink = [PB:DNN.PageInfo.DisableLink]
EndDate = [PB:DNN.PageInfo.EndDate]
FullUrl = [PB:DNN.PageInfo.FullUrl]
HasChildren = [PB:DNN.PageInfo.HasChildren]
IconFile = [PB:DNN.PageInfo.IconFile]
IsAdminTab = [PB:DNN.PageInfo.IsAdminTab]
IsDeleted = [PB:DNN.PageInfo.IsDeleted]
IsSuperTab = [PB:DNN.PageInfo.IsSuperTab]
IsVisible = [PB:DNN.PageInfo.IsVisible]
Level = [PB:DNN.PageInfo.Level]
PageHeadText = [PB:DNN.PageInfo.PageHeadText]
ParentId = [PB:DNN.PageInfo.ParentId]
RefreshInterval = [PB:DNN.PageInfo.RefreshInterval]
SkinPath = [PB:DNN.PageInfo.SkinPath]
SkinSrc = [PB:DNN.PageInfo.SkinSrc]
StartDate = [PB:DNN.PageInfo.StartDate]
TabID = [PB:DNN.PageInfo.TabID]
TabName = [PB:DNN.PageInfo.TabName]
TabOrder = [PB:DNN.PageInfo.TabOrder]
TabPath = [PB:DNN.PageInfo.TabPath]
Title = [PB:DNN.PageInfo.Title]
```

DNN.PortalInfo

```
AdministratorId = [PB:DNN.PortalInfo.AdministratorId]
AdministratorRoleId = [PB:DNN.PortalInfo.AdministratorRoleId]
AdministratorRoleName = [PB:DNN.PortalInfo.AdministratorRoleName]
AdminTabId = [PB:DNN.PortalInfo.AdminTabId]
```

Snapsis PageBlaster - Content Delivery Engine for DNN®

```
BackgroundFile = [PB:DNN.PortalInfo.BackgroundFile]
BannerAdvertising = [PB:DNN.PortalInfo.BannerAdvertising]
Currency = [PB:DNN.PortalInfo.Currency]
DefaultLanguage = [PB:DNN.PortalInfo.DefaultLanguage]
Description = [PB:DNN.PortalInfo.Description]
Email = [PB:DNN.PortalInfo.Email]
ExpiryDate = [PB:DNN.PortalInfo.ExpiryDate]
FooterText = [PB:DNN.PortalInfo.FooterText]
HomeDirectory = [PB:DNN.PortalInfo.HomeDirectory]
HomeDirectoryMapPath = [PB:DNN.PortalInfo.HomeDirectoryMapPath]
HomeTabId = [PB:DNN.PortalInfo.HomeTabId]
Keywords = [PB:DNN.PortalInfo.KeyWords]
LoginTabId = [PB:DNN.PortalInfo.LoginTabId]
LogoFile = [PB:DNN.PortalInfo.LogoFile]
PortalId = [PB:DNN.PortalInfo.PortalId]
PortalName = [PB:DNN.PortalInfo.PortalName]
RegisteredRoleId = [PB:DNN.PortalInfo.RegisteredRoleId]
RegisteredRoleName = [PB:DNN.PortalInfo.RegisteredRoleName]
SiteLogHistory = [PB:DNN.PortalInfo.SiteLogHistory]
SplashTabId = [PB:DNN.PortalInfo.SplashTabId]
SuperTabId = [PB:DNN.PortalInfo.SuperTabId]
TimeZoneOffset = [PB:DNN.PortalInfo.TimeZoneOffset]
UserRegistration = [PB:DNN.PortalInfo.UserRegistration]
UserTabId = [PB:DNN.PortalInfo.UserTabId]
Version = [PB:DNN.PortalInfo.Version]
```