NTT DATA Sitecore CDN Connector

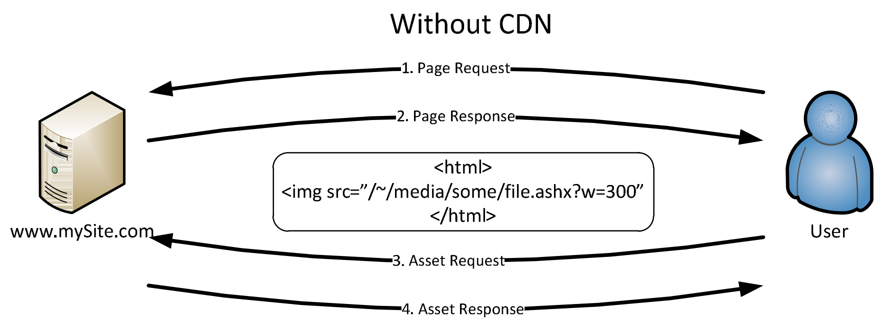
Overview

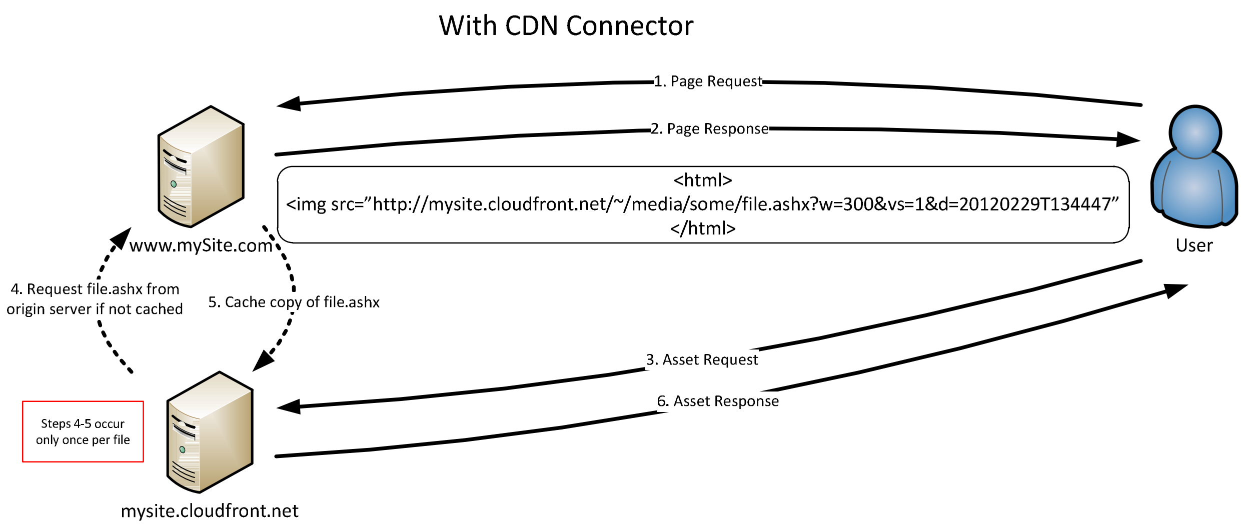
The CDN Connector for Sitecore allows developers to route all media requests (dynamic and static) through a proxy CDN. It is designed to be plug-n-play requiring no additional development effort other than configuration.

Media Rerouting

Media URLs that would traditionally be handled by the web server would be rewritten to point to a geographically appropriate edge server which will cache media items from the origin server as they are requested. This takes a large amount of load off the web (origin) server.

An example using AWS CloudFront would be:




With CDN Connector
1. Page Request
2. Page Response
<html>
<img src="http://mysite.cloudfront.net/~/media/some/file.ashx?w=300&vs=1&d=20120229T134447"
</html>
www.mySite.com
User
4. Request file.ashx from origin server if not cached
5. Cache copy of file.ashx
3. Asset Request
6. Asset Response
Steps 4-5 occur only once per file
mysite.cloudfront.net

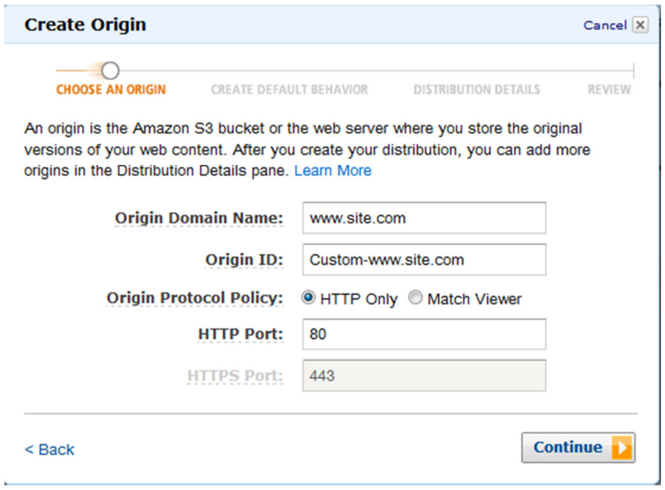
Setup

The package simply needs to be installed and configured for the desired CDN's distribution server.

If working with AWS CloudFront, you can follow the below steps for setting up a distribution.

Setting up the distribution in AWS

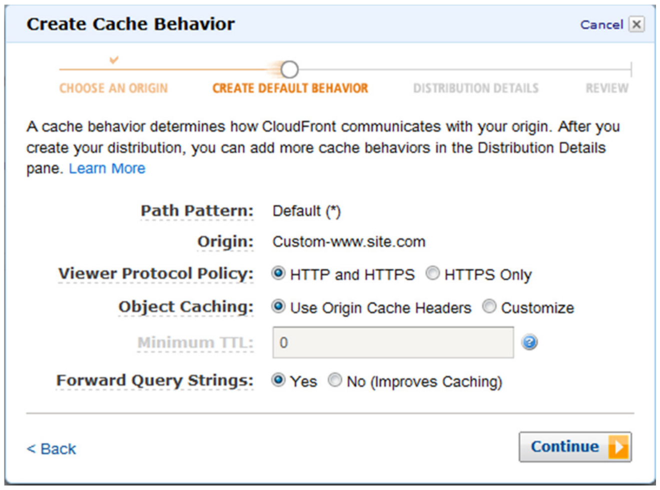
1. Sign into the AWS console, then the CloudFront tab.
2. Click "Create Distribution"
3. Choose the Download delivery method
4. Provide the hostname of your public facing web server as the Origin Domain Name and accept the defaults



An origin is the Amazon S3 bucket or the web server where you store the original versions of your web content. After you create your distribution, you can add more origins in the Distribution Details pane. Learn More

Origin Domain Name:	www.site.com
Origin ID:	Custom-www.site.com
Origin Protocol Policy:	[✓ HTTP Only] Match Viewer
HTTP Port:	80
HTTPS Port:	443

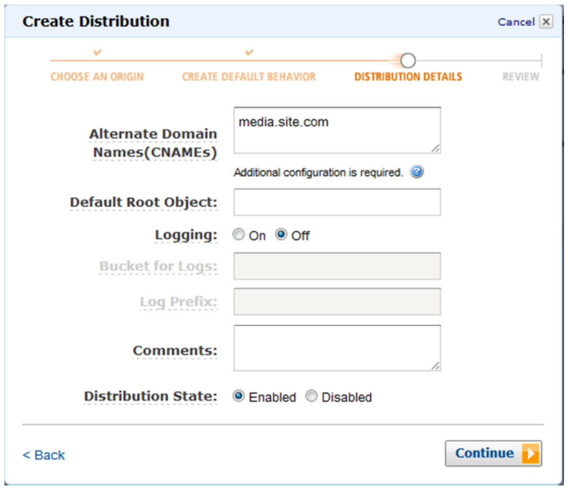
5. Choose "Yes" for Forward Query Strings. The rest of the default values are fine



A cache behavior determines how CloudFront communicates with your origin. After you create your distribution, you can add more cache behaviors in the Distribution Details pane. Learn More

Path Pattern:	Default (*)
Origin:	Custom-www.site.com
Viewer Protocol Policy:	[✓ HTTP and HTTPS] HTTPS Only
Object Caching:	[✓ Use Origin Cache Headers] Customize
Minimum TTL:	0
Forward Query Strings:	[✓ Yes] No (Improves Caching)

6. Accept the default values for the Distribution Details. CNAMEs are optional domain names you can set up from your DNS to point to the CloudFront domain name you'll receive later. (Example: "media.site.com")



Alternate Domain Names(CNAMEs)	media.site.com Additional configuration is required.
Default Root Object:	
Logging:	On [✓ Off]
Bucket for Logs:	
Log Prefix:	
Comments:	
Distribution State:	[✓ Enabled] Disabled

7. Click through the rest of the wizard and you'll be returned to the distribution list

8. Click on your new distribution and copy the “Domain Name”. This is the publicly provided hostname if you are not using your own custom CNAMEs. If you are using custom CNAMEs use this “Domain Name” as the CNAME’s target.
9. You will have to wait for the distributions status to change from “In Progress” to “Enabled” before you can test this.

Installing on the sitecore application

Ensure the following files are installed

1. /bin/NTTData.SitecoreCDN.dll
2. /App_config/Include/SitecoreCDN.config

In your web.config, do the following:

1. Under /system.webserver/handlers/
   ADD
   <add verb="*" path="aws_minify.ashx" type="NTTData.SitecoreCDN.Handlers.MinifyHandler, NTTData.SitecoreCDN" name="SitecoreCDN.Minify" />
   BEFORE
   <add verb="*" path="sitecore_handlers.ashx" type="Sitecore.Web.CustomHandlerFactory, Sitecore.Kernel" name="Sitecore.GenericHandler"/>
2. Under /system.web/httpHandlers/
   ADD
   <add path="aws_minify.ashx" verb="*" type="NTTData.SitecoreCDN.Handlers.MinifyHandler, NTTData.SitecoreCDN" />
   BEFORE
   <add verb="*" path="sitecore_handlers.ashx" type="Sitecore.Web.CustomHandlerFactory, Sitecore.Kernel"/>
3. In /App_Config/Includes/SitecoreCDN.config set the hostname of your CDN per site.

```
<sites>
   <site name="website">
      <patch:attribute name="cdnHostName">yourcdnhostname</patch:attribute>
   </site>
</sites>
```

Html Rewriting

The module rewrites the resulting html at the last moment by overriding the Response.Filter stream.

A pipeline processor `NTTData.SitecoreCDN.Pipelines.CDNAttachFilter` is located under

```
<httpRequestProcessed>
        …
        <processor type="NTTData.SitecoreCDN.Pipelines.CDNAttachFilter, NTTData.SitecoreCDN"/>
        …
</httpRequestProcessed>
```

This pipeline decides whether to attach the Response.Filter based on a few criteria

1. Sitecore.Context.Item != null (Item was resolved)
   a. OR args.Url.FilePathWithQueryString matches an entry in <processReqeusts> config
2. Sitecore.Context.Site != null (Context Site resolved)

3. Sitecore.Context.PageMode.IsNormal (Site is not being edited)
4. Sitecore.Context.Site has a "**cdnHostName**" attribute set (hostnames are per site)
5. Querystring url parameter "?**cdn**=true|false" (for overriding default behavior)

The rewriting filter then takes all the generated html from the page and replaces urls on these elements <img src="[url]" />, <script src="[url]" ></script>, and <link href="[url]"/> with transformed urls before returning the page to the client.

URL Rewriting

URLs are rewritten with 3 purposes:

1. Attaching the CDN hostname to make the url fully qualified and pointing to the proxy CDN
2. Attach version information and a datetime stamp to the query string **to guarantee uniqueness** so the CDN doesn't over-cache a file that's been modified.

Dynamic Media Urls

Dynamic urls are rewritten from
/~/media/path/to/file.ashx?w=300&as=1 to
http://cdnhostname/~/media/path/to/file.ashx?w=300&as=1&vs=3& d=20120101T120000

The &vs=3 is this media's sitecore Item version while &d=20120101T120000 is the last modified time of the item for un-versioned items. A CDN will treat these rewritten parameters as a unique filename when caching.

Static media Urls

Static urls are rewritten from
/path/to/file.ext to
http://cdnhostname/path/to/file.ext?d=20120101T120000

The d=20120101T120000 is the "Last Modified" date of the file on the file system. A CDN will treat these rewritten parameters as a unique filename when caching.

Excluded Urls

Certain URLS will be excluded from being rewritten, instead keeping their local url.

1. Any external urls
2. Urls matching any /sitecore/cdn/excludeUrls/regex in SitecoreCDN.config
3. Media Items that are not publicly accessible to [domain]\Anonymous
4. Media Items that use Analytics Tracking

Consuming Rewritten URLs

The rewritten URL requests will go to the closest edge server. If the server hasn't previously cached this file, it will go to the origin server to get the first copy. Sitecore must be able to recognize these rewritten urls and recognize the embedded filepath and querystring parameters.

A PreProcessRequest pipeline step `NTTData.SitecoreCDN.Pipelines.CDNInterceptPipeline` is located under

```
<preprocessRequest>
	…
	<processor type=" NTTData.SitecoreCDN.Pipelines.CDNInterceptPipeline, NTTData.SitecoreCDN"/>
	…
```

```
</preprocessRequest>
```

The preprocessor will assess whether the item is a minifiable CSS or JS file and rewrite the URL for the minify handler.

Performance Enhancements

Minification

Static .js and .css requests coming from the CDN can optionally be minified before being delivered. This will result in decrease overall page size and improve load times.
This feature can be enabled/disabled in the /app_config/include/SitecoreCDN.config

```
<cdn enabled="true" … minifyEnabled="true" …>
```

CSS Processing

If minification is enabled, the media handler will also do URL replacements on css url([url]) statements. This ensures that any media referenced within the css also gets the benefit of the rewritten media urls.

This feature can be enabled/disabled in the /app_config/include/ SitecoreCDN.config

```
<cdn enabled="true" … processCss="true" …>
```

JS Fastloading

When an .aspx page request is having its urls rewritten, you have the option to automatically move any <script> tags to just before the </body> tag. This allows the browser to parallel load scripts rather than the browser imposed limit of 2 simultaneous downloads of script tags located in the <head>.

This feature can be enabled/disabled in the /app_config/include/ SitecoreCDN.config

```
<cdn enabled="true" … fastLoadJsEnabled="true" …>
```

Configuration

Under /App_Config/Include/ SitecoreCDN.config

```
      <cdn enabled="true" filenameVersioningEnabled="true" minifyEnabled="true"
fastLoadJsEnabled="true" processCss="true">
        <provider type=" NTTData.SitecoreCDN.Providers.CDNProvider, NTTData.SitecoreCDN" />
        <!-- Incoming requests matching these urls will be processed
              .aspx is processed when Sitecore.Context.Item is resolved
        -->
        <processRequests>
          <regex pattern = "\.asmx" />
          <!-- matches any .asmx -->
        </processRequests>

        <!-- Incoming requests matching these urls will not be processed -->
        <excludeRequests>
          <!--<regex pattern = "Default\.aspx" />-->
        </excludeRequests>

        <!-- These regex patterns will prevent matching urls from being replaced in the outgoing html,
doesn't affect Incoming request processing -->
        <excludeUrls>
          <regex pattern = "\.axd" />
          <!-- this keeps ScriptResource.axd and WebResource.axd from being CDN'd-->
          <regex pattern = "VisitorIdentification.aspx" />
          <!-- this keeps the Sitecore Analytics request from being CDN'd -->
        </excludeUrls>
      </cdn>
```

On the <cdn> element, `enabled="true"` turns the HTML rewriting on and off. `filenameVersioningEnabled="true"` turns on/off the versioning features of the URL rewrite.

`minifyEnabled ="true"` turns on/off the css/js minification.

`fastLoadJsEnabled ="true"` turns on/off the javascript tag relocation.

`processCss ="true"` turns on/off url rewriting within css files

You can override any behavior of the CDNManager class by implementing your own CDNProvider and changing `<provider type="NTTData.SitecoreCDN.Providers.CDNProvider, NTTData.SitecoreCDN" />`

You can specify which incoming requests will get Url replacement in <processRequests> by matching regex. By default, ".aspx" is only processed when Sitecore.Context.Item is not null.

You can specify which incoming requests to exclude form Url replacement in <excludeRequests> by matching regex.

Even when a request is being processed, you can specify a list of URLs that will not be replaced under <excludeUrls> by matching regex. Some media item you want going directly to the origin server. For example: Robot identifying files for DMS and .net generated script .axd files.