**Dispatcher Overview**

Dispatcher versions are independent of AEM. You may have been redirected to this page if you followed a link to the Dispatcher documentation that is embedded in the documentation for a previous version of AEM.

Dispatcher is Adobe Experience Manager's caching and/or load balancing tool. Using AEM's Dispatcher also helps to protect your AEM server from attack. Therefore, you can increase the security of your AEM instance by using the Dispatcher in conjunction with an enterprise-class web server.

The process for deploying a Dispatcher is independent of the web server and the OS platform chosen:

1. Learn about Dispatcher (this page). Also, see [frequently asked questions about dispatcher](https://helpx.adobe.com/experience-manager/using/dispatcher-faq.html).
2. Install a [supported web server](https://helpx.adobe.com/experience-manager/6-3/sites/deploying/using/technical-requirements.html) according to the web server documentation.
3. [Install the Dispatcher module](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/dispatcher-install.html) on your web server and configure the web server accordingly.
4. [Configure Dispatcher](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html) (the dispatcher.any file).
5. [Configure AEM](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/page-invalidate.html) so that content updates invalidate the cache.

To gain a better undestanding of how Dispatcher works with AEM see [Ask the AEM Community Experts for July 2017](https://bit.ly/ATACE0717).

Use the following information as required:

* [The Dispatcher Security Checklist](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/security-checklist.html)
* [The Dispatcher Knowledge Base](https://helpx.adobe.com/cq/kb/index/dispatcher.html)
* [Optimizing a Website for Cache Performance](https://helpx.adobe.com/experience-manager/6-4/sites/deploying/using/configuring-performance.html)
* [Using Dispatcher with Multiple Domains](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-domains.html)
* [Using SSL with Dispatcher](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-ssl.html)
* [Implementing Permission-Sensitive Caching](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/permissions-cache.html)
* [Troubleshooting Dispatcher Problems](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/troubleshooting/dispatcher-troubleshooting.html)
* [Dispatcher Top Issues FAQ](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/troubleshooting/dispatcher-faq.html)

**The most common use of Dispatcher** is to cache responses from an AEM **publish instance**, to increase the responsiveness and security of your externally facing published website. Most of the discussion focuses on this case.

But, the Dispatcher can also be used to increase the responsiveness of your **author instance**, particularly if you have a large number users editing and updating your website. For details specific to this case see [Using a Dispatcher with an Author Server](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/dispatcher.html#using-a-dispatcher-with-an-author-server), below.

Why use Dispatcher to implement Caching?

There are two basic approaches to web publishing:

* **Static Web Servers**: such as Apache or IIS, are very simple, but fast.
* **Content Management Servers**: which provide dynamic, real-time, intelligent content, but require much more computation time and other resources.

The Dispatcher helps realize an environment that is both fast and dynamic. It works as part of a static HTML server, such as Apache, with the aim of:

* storing (or "caching") as much of the site content as is possible, in the form of a static website
* accessing the layout engine as little as possible.

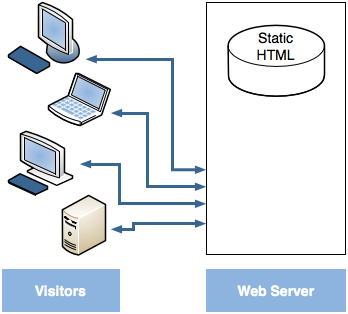
Which means that:

* **static content** is handled with exactly the same speed and ease as on a static web server;*additionally you can use the administration and security tools available for your static web server(s)*.
* **dynamic content** is generated as needed, without slowing the system down any more than absolutely necessary.

The Dispatcher contains mechanisms to generate, and update, static HTML based on the content of the dynamic site. You can specify in detail which documents are stored as static files and which are always generated dynamically.

This section illustrates the principles behind this.

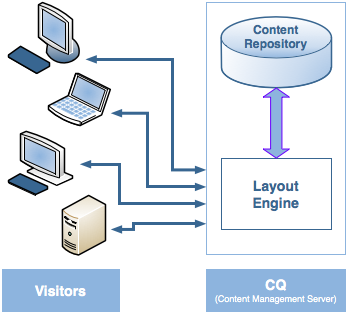
Static Web Server



A static web server, such as Apache or IIS, serves static HTML files to visitors of your website. Static pages are created once, so the same content will be delivered for each request.

This process is very simple, and thus extremely efficient. If a visitor requests a file (e.g. a HTML page), the file is usually taken directly from memory, at worst it is read from the local drive. Static web servers have been available for quite some time, so there is a wide range of tools for administration and security management, and they are very well integrated with network infrastructures.

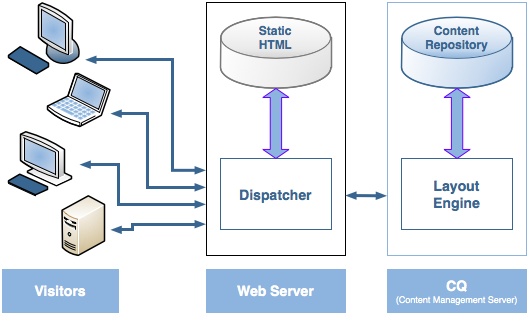
Content Management Servers



If you use a Content Management Server, such as AEM, an advanced layout engine processes the request from a visitor. The engine reads content from a repository which, combined with styles, formats and access rights, transforms the content into a document that is tailored to the visitor's needs and rights.

This allows you to create richer, dynamic content, which increases the flexibility and functionality of your website. However, the layout engine requires more processing power than a static server, so this setup may be prone to slowdown if many visitors use the system.

How Dispatcher performs Caching



**The Cache Directory** For caching, the Dispatcher module uses the web server's ability to serve static content. The Dispatcher places the cached documents in the document root of the web server.

When lacking the configuration for HTTP Header Caching, the Dispatcher stores only the HTML code of the page - it does not store the HTTP headers. This can be an issue if you use different encodings within your website, as these may get lost. To enable HTTP Header Caching, see [Configuring the Dispatcher Cache.](https://helpx.adobe.com/experience-manager/dispatcher/using/dispatcher-configuration.html)

Locating the document root of your web server on network-attached storage (NAS) causes performance degradation. Also, when a document root that is located on NAS is shared between multiple web servers, intermittant locks can occur when replication actions are performed.

The Dispatcher stores the cached document in a structure equal to the requested URL.

There can be OS-level limitations for length of file name; i.e. if you have an URL with a lot of selectors.

Methods for Caching

The Dispatcher has two primary methods for updating the cache content when changes are made to the website.

* **Content Updates** remove the pages that have changed, as well as files that are directly associated with them.
* **Auto-Invalidation** automatically invalidates those parts of the cache that may be out of date after an update. i.e. it effectively flags relevant pages as being out of date, without deleting anything.

Content Updates

In a content update, one or more AEM documents change. AEM sends a syndication request to the Dispatcher, which updates the cache accordingly:

1. It deletes the modified file(s) from the cache.
2. It deletes all files that start with the same handle from the cache. For example, if the file /en/index.html is updated, all the files that start with /en/index. are deleted. This mechanism allows you to design cache-efficient sites, especially in regard to picture navigations.
3. It *touches* the so-called **statfile**; this updates the timestamp of the statfile to indicate the date of the last change.

The following points should be noted:

* Content Updates are typically used in conjunction with an authoring system which "knows" what must be replaced.
* Files that are affected by a content update are removed, but not replaced immediately. The next time such a file is requested, the Dispatcher fetches the new file from the AEM instance and places it in the cache, thereby overwriting the old content.
* Typically, automatically generated pictures that incorporate text from a page are stored in picture files starting with the same handle - thus ensuring that the association exists for deletion. For example, you may store the title text of the page mypage.html as the picture mypage.titlePicture.gif in the same folder. This way the picture is automatically deleted from the cache each time the page is updated, so you can be sure that the picture always reflects the current version of the page.
* You may have several statfiles, for example one per language folder. If a page is updated, AEM looks for the next parent folder containing a statfile, and *touches* that file.

Auto-invalidation

Auto-invalidation automatically invalidates parts of the cache - without physically deleting any files. At every content update, the so-called statfile is touched, so its timestamp reflects the last content update.

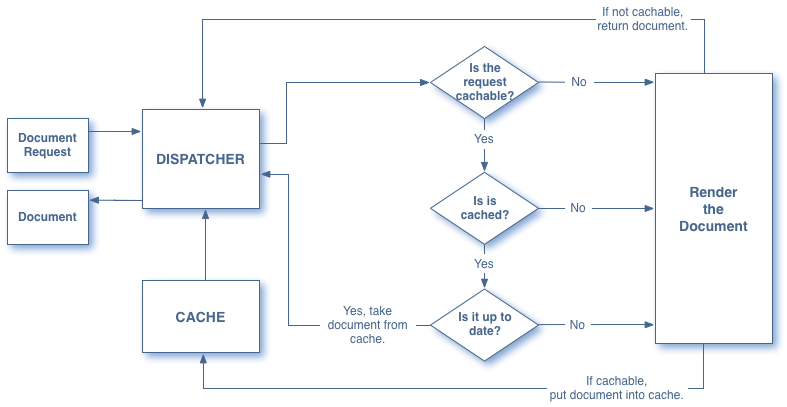
The Dispatcher has a list of files that are subject to auto-invalidation. When a document from that list is requested, the Dispatcher compares the date of the cached document with the timestamp of the statfile:

* if the cached document is newer, the Dispatcher returns it.
* if it is older, the Dispatcher retrieves the current version from the AEM instance.

Again, certain points should be noted:

* Auto invalidation is typically used when the inter-relations are complex e.g. for HTML pages. These pages contain links and navigation entries, so they usually have to be updated after a content update. If you have automatically generated PDF or picture files, you may choose to auto-invalidate those too.
* Auto-invalidation does not involve any action by the dispatcher at update time, except for touching the statfile. However, touching the statfile automatically renders the cache content obsolete, without physically removing it from the cache.

How Dispatcher returns Documents



Determining whether a document is subject to caching

You can [define which documents the Dispatcher caches in the configuration file](https://helpx.adobe.com/experience-manager/dispatcher/using/dispatcher-configuration.html). The Dispatcher checks the request against the list of cacheable documents. If the document is not in this list, the Dispatcher requests the document from the AEM instance.

The Dispatcher *always* requests the document directly from the AEM instance in the following cases:

* If the request URI contains a question mark "?". This usually indicates a dynamic page, such as a search result, which does not need to be cached.
* The file extension is missing. The web server needs the extension to determine the document type (the MIME-type).
* The authentication header is set (this can be configured)

The GET or HEAD (for the HTTP header) methods are cacheable by the Dispatcher. For additional information on response header caching, see the [Caching HTTP Response Headers](https://helpx.adobe.com/experience-manager/dispatcher/using/dispatcher-configuration.html) section.

Determining if a document is cached

The Dispatcher stores the cached files on the web server as if they were part of a static website. If a user requests a cacheable document the Dispatcher checks whether that document exists in the web server's file system:

* if the document is cached, Dispatcher returns the file.
* if it is not cached, the Dispatcher requests the document from the AEM instance.

Determining if a document is up-to-date

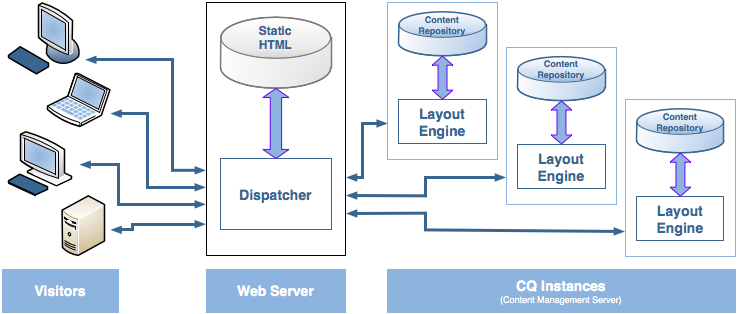
To find out if a document is up to date, the Dispatcher performs two steps:

1. It checks whether the document is subject to auto-invalidation. If not, the document is considered up-to-date.
2. If the document is configured for auto-invalidation, the Dispatcher checks whether it is older or newer than the last change available. If it is older, the Dispatcher requests the current version from the AEM instance and replaces the version in the cache.

Documents without **auto-invalidation** remain in the cache until they are physically deleted; e.g. by a content update on the web site.

The Benefits of Load Balancing

Load Balancing is the practice of distributing the computational load of the website across several instances of AEM.



You gain:

* **increased processing power** In practice this means that the Dispatcher shares document requests between several instances of AEM. Because each instance now has fewer documents to process, you have faster response times. The Dispatcher keeps internal statistics for each document category, so it can estimate the load and distribute the queries efficiently.
* **increased fail-safe coverage** If the Dispatcher does not receive responses from an instance, it will automatically relay requests to one of the other instance(s). Thus, if an instance becomes unavailable, the only effect is a slowdown of the site, proportionate to the computational power lost. However, all services will continue.
* you can also manage different websites on the same static web server.

While load balancing spreads the load efficiently, caching helps to reduce the load. Therefore, try to optimize caching and reduce the overall load before you set up load balancing. Good caching may increase the load balancer's performance, or render load balancing unnecessary.

While a single Dispatcher will usually be able to saturate the capacity of the available Publish instances, for some rare applications it can make sense to additionally balance the load between two Dispatcher instances. Configurations with multiple Dispatchers need to be considered carefully, since an additional Dispatcher will increase the load on the available Publish instances and can easily decrease performance in most applications.

How the Dispatcher performs Load Balancing

Performance Statistics

The Dispatcher keeps internal statistics about how fast each instance of AEM processes documents. Based on this data, the Dispatcher estimates which instance will provide the quickest response time when answering a request, and so it reserves the necessary computation time on that instance.

Different types of requests may have differing average completion times, so the Dispatcher allows you to specify document categories. These are then considered when computing the time estimates. For example, you can make a distinction between HTML pages and images, as the typical response times may well differ.

If you use an elaborate search function, you may create a new category for search queries. This helps the Dispatcher send search queries to the instance that responds fastest. This prevents a slower instance from stalling when it receives several "expensive" search queries, while the others get the "cheaper" requests.

Personalized content (Sticky Connections)

Sticky connections ensure that documents for one user are all composed on the same instance of AEM. This is important if you use personalized pages and session data. The data is stored on the instance, so subsequent requests from the same user must return to that instance or the data is lost.

Because sticky connections restrict the Dispatcher's ability to optimize the requests, you should use them only when needed. You can specify the folder that contains the "sticky" documents, thus ensuring all documents in that folder are composed on the same instance for each user.

For most pages that use sticky connections you have to switch off caching - otherwise the page looks the same to all users, regardless of the session content.

For a *few* applications, it can be possible to use both sticky connections and caching; for example, if you display a form that writes data to the session.

Using Multiple Dispatchers

In complex setups, you may use multiple Dispatchers. For example, you may use:

* one Dispatcher to publish a website on the Intranet
* a second Dispatcher, under a different address and with different security settings, to publish the same content on the Internet.

In such a case, make sure that each request goes through only one Dispatcher. A Dispatcher does not handle requests that come from another Dispatcher. Therefore, make sure that both Dispatchers access the AEM website directly.

Using Dispatcher with a CDN

A content delivery network (CDN), such as Akamai Edge Delivery or Amazon Cloud Front, deliver content from a location close to the end user. By that it

* speeds up response times for end users
* takes load off your servers

As an HTTP infrastructure component, a CDN works much like Dispatcher: when a CDN node receives a request, it serves the request from its cache if possible (the resource is available in the cache and is valid). Otherwise, it reaches out to the next closest server to retrieve the resource and cache it for further requests if appropriate.

The "next closest server" depends on your specific setup. For example, in an Akamai setup the request can take the following path:

* The Akamai Edge Node
* The Akamai Midgres Layer
* Your firewall
* Your load balancer
* Dispatcher
* AEM

In most cases, Dispatcher is the next server that might serve the document from a cache and influence the response headers returned to the CDN server.

Controlling a CDN cache

There are a numer of ways to control for how long a CDN will cache a resource before it re-fetches is from Dispatcher.

1. Explicit configurationConfigure, how long particular resources are held in the CDN's cache, depending on mime type, extension, request type, etc.
2. Expiration and cache-control headersMost CDNs will honor Expires: and Cache-Control: HTTP Headers if sent by the upstream server. This can be achieved e.g. by using the [mod\_expires](https://httpd.apache.org/docs/2.2/mod/mod_expires.html) Apache Module.
3. Manual invalidationCDNs allow resources to be removed from the cache through web interfaces.
4. API based invalidationMost CDNs also offer a REST and/or SOAP API that allows resources to be removed from the cache.

In a typical AEM setup, configuration by extension and/or path, which can be achieved through points 1 and 2 above, offers possibilities to set reasonable caching periods for often-used resources that do not change often, such as design images and client libraries. When new releases are deployed, typically a manual invalidation is required.

If this approach is used to cache managed content, it implies that content changes are only visible to end users once the configured caching period is expired and the document is fetched from Dispatcher again.

For finer-grained control, API based invalidation allows you to invalidate a CDN's cache as the Dispatcher cache is invalidated. Based on the CDNs API, you can implement your own [ContentBuilder](https://docs.adobe.com/docs/en/cq/current/javadoc/com/day/cq/replication/ContentBuilder.html) and [TransportHandler](https://docs.adobe.com/docs/en/cq/current/javadoc/com/day/cq/replication/TransportHandler.html) (if the API is not REST-based) and set up a Replication Agent that will use these to invalidate the CDN's cache.

See also [AEM (CQ) Dispatcher Security and CDN+Browser Caching](https://www.slideshare.net/andrewmkhoury/dispatcher-caching-aemgemspart2jan2015) and recorded presentation on [Dispatcher Caching](https://docs.adobe.com/content/ddc/en/gems/dispatcher-caching---new-features-and-optimizations.html).

Using a Dispatcher with an Author Server

if you are using [AEM with Touch UI](https://helpx.adobe.com/experience-manager/6-3/sites/developing/using/touch-ui-concepts.html) you should **not** cache author instance content. If caching was enabled for the author instance, you need to disable it and delete the contents of the cache directory. To disable caching, you should edit the author\_dispatcher.any file and modify the /rule property of the /cache section as follows:

/rules

{

/0000

{ /type "deny" /glob "\*"}

}

A Dispatcher can be used in front of an author instance to improve authoring performance. To configure an authoring Dispatcher, do the following:

1. Install a Dispatcher in a web server (this could be Apache or IIS web server, see [Installing Dispatcher](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/dispatcher-install.html)).
2. You may wish to test the newly installed Dispatcher against a working AEM publish instance, to ensure that a baseline correct install has been acheived.
3. Now make sure that the Dispatcher is able to connect via TCP/IP to your author instance.
4. Replace the sample dispatcher.any file with the author\_dispatcher.any file provided with the [Dispatcher download](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/release-notes.html#downloads).
5. Open the author\_dispatcher.any in a text editor and make the following changes:
   1. Change the /hostname and /port of the /renders section to point to your author instance.
   2. Change the /docroot of the /cache section to point to a cache directory. In case you are using [AEM with Touch UI](https://helpx.adobe.com/experience-manager/6-3/sites/developing/using/touch-ui-concepts.html), see the warning above.
   3. Save the changes.
6. Delete all existing files in the /cache > /docroot directory which you configured above.
7. Restart the web server.

Please note that with the provided author\_dispatcher.any configuration, when you install a CQ5 feature pack, hotfix, or application code package that affects any content under /libs or /apps then you must delete the cached files under those directories in your dispatcher cache to ensure that the next time they are requested the newly upgraded files are fetched, and not the old cached ones.

If you have used the previously configured author dispatcher and enabled a *dispatcher flushing agent* then please do the following:

1. Delete or disable the **author dispatcher's** flushing agent on your AEM author instance.
2. Re-do the author dispatcher configuration by following the new instructions above.

**Installing Dispatcher**

Dispatcher versions are independent of AEM. You may have been redirected to this page if you followed a link to the Dispatcher documentation that is embedded in the documentation for a previous version of AEM.

Use the [Dispatcher Release Notes](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/release-notes.html) page to obtain the latest Dispatcher installation file for your operating system and web server. Dispatcher release numbers are independent of the Adobe Experience Manager release numbers and are compatible with Adobe Experience Manager 6.x, 5.x and Adobe CQ 5.x releases.

The following file naming convention is used:

dispatcher-<web-server>-<operating-system>-<dispatcher-version-number>.<file-format>

For example, the dispatcher-apache2.4-linux-x86\_64-ssl-4.3.1.tar.gz file contains Dispatcher version 4.3.1 for an Apache 2.4 web server that runs on Linux i686 and is packaged using the **tar** format.

The following table lists the web server identifier that is used in file names for each web server:

| **Web Server** | **Installation Kit** |
| --- | --- |
| Apache 2.4 | dispatcher-apache**2.4**-<other parameters> |
| Apache 2.2 | dispatcher-apache**2.2**-<other parameters> |
| Microsoft Internet Information Server 7.5, 8, 8.5 | dispatcher-**iis**-<other parameters> |
| Sun Java Web Server iPlanet | dispatcher-**ns**-<other parameters> |

You should install the latest version of Dispatcher that is available for your platform. On a yearly basis, you should upgrade your Dispatcher instance to use the latest version to take advantage of product improvements.

Each archive contains the following files:

* the Dispatcher modules
* an example configuration file
* the README file that contains installation instructions and last-minute information
* the CHANGES file that lists issues fixed in current and past releases

Please check the README file for any last-minute changes / platform specific notes before starting the installation.

Microsoft Internet Information Server

For information on how to install this web server, see the following resources:

* Microsoft's own documentation on the Internet Information Server
* ["The Official Microsoft IIS site"](https://www.iis.net/)

Required IIS Components

IIS versions 8.5 and 10 require that the following IIS components are installed:

* ISAPI Extensions

Also, you must add the Web Server (IIS) role. Use Server Manager to add the role and components.

Microsoft IIS - Installing the Dispatcher module

The required archive for Microsoft Internet Information System is:

* dispatcher-iis-<operating-system>-<dispatcher-release-number>.zip

The ZIP file contains the following files:

| **File** | **Description** |
| --- | --- |
| disp\_iis.dll | The Dispatcher dynamic link library file. |
| disp\_iis.ini | Configuration file for the IIS. This example can be updated with your requirements. **Note**: The ini file must have the same name-root as the dll. |
| dispatcher.any | An example configuration file for the Dispatcher. |
| author\_dispatcher.any | An example configuration file for Dispatcher working with the author instance. |
| README | Readme file that contains installation instructions and last-minute information. **Note**: Please check this file before starting the installation. |
| CHANGES | Changes file that lists issues fixed in current and past releases. |

Use the following procedure to copy the Dispatcher files to the correct location.

1. Use Windows Explorer to create the <IIS\_INSTALLDIR>/Scripts directory, for example, C:\inetpub\Scripts.
2. Extract the following files from the Dispatcher package into this Scripts directory:
   * disp\_iis.dll
   * disp\_iis.ini
   * One of the following files depending on if Dispatcher is working with an AEM author instance or publish instance:
     + Author instance: author\_dispatcher.any
     + Publish instance: dispatcher.any

Microsoft IIS - Configure the Dispatcher INI File

Edit the disp\_iis.ini file to configure the Dispatcher installation. The basic format of the .ini file is as follows:

[main]

configpath=<path to dispatcher.any>

loglevel=1|2|3

servervariables=0|1

replaceauthorization=0|1

The following table describes each property.

| **Parameter** | **Description** |
| --- | --- |
| configpath | The location of dispatcher.any within the local file system (absolute path). |
| logfile | The location of the dispatcher.log file. If this is not set then log messages go to the windows event log. |
| loglevel | Defines the Log Level used to output messages to the event log. The following values may be specified:Log level for the log file:  0 - error messages only.  1 - errors and warnings.  2 - errors, warnings and informational messages  3 - errors, warnings, informational and debug messages.  **Note**: It is recommended to set the log level to 3 during installation and testing, then to 0 when running in a production environment. |
| replaceauthorization | Specifies how authorization headers in the HTTP request are handled. The following values are valid:  0 - Authorization headers are not modified.  1 - Replaces any header named "Authorization" other than "Basic" with its Basic <IIS:LOGON\\_USER> equivalent. |
| servervariables | Defines how server variables are processed.  0 - IIS server variables are sent to neither the Dispatcher nor AEM.  1 - all IIS server variables (such as LOGON\\_USER, QUERY\\_STRING, ...) are sent to the Dispatcher, together with the request headers (and also to the AEM instance if not cached).  Server variables include AUTH\\_USER, LOGON\\_USER, HTTPS\\_KEYSIZE and many others. See the IIS documentation for the full list of variables, with details. |
| enable\_chunked\_transfer | Defines whether to enable (1) or disable (0) chunked transfer for the client response. The default value is 0. |

An example configuration:

[main]

configpath=C:\Inetpub\Scripts\dispatcher.any

loglevel=1

servervariables=1

replaceauthorization=0

Configuring Microsoft IIS

Configure IIS to integrate the Dispatcher ISAPI module. In IIS you use wildcard application mapping.

Configuring Anonymous Access - IIS 8.5 and 10

The default Flush replication agent on the Author instance is configured so that it does not send security credentials with flush requests. Therefore, the website that you are using a the Dispatcher cache must allow anonymous access.

If your website uses an authentication method, the Flush replication agent must be configured accordingly.

1. Open IIS Manager and select the website that you are using as the Disptcher cache.
2. Using Features View mode, in the IIS section double click Authentication.
3. If Anonymous Authentication is not enabled, select Anonymous Authentication and in the Actions area click Enable.

Integrating the Dispatcher ISAPI Module - IIS 8.5 and 10

Use the following procedure to add the Dispatcher ISAPI Module to IIS.

1. Open IIS Manager.
2. Select the website that you are using as the Dispatcher Cache.
3. Using Features View mode, in the IIS section double click Handler Mappings.
4. In the Actions panel of the Handler Mappings page, click Add Wildcard Script Map, add the following property values and then click OK:
   * Request Path:
   * Executable: The absolute path of the disp\_iis.dll file, for example C:\inetpub\Scripts\disp\_iis.dll.
   * Name: A descriptive name for the handler mapping, for example Dispatcher.
5. In the dialog box that appears, to add the disp\_iis.dll library to the ISAPI and CGI Restrictions list, click Yes.

For IIS 7.0 and 7.5 the configuration is complete. Continue with the remaining steps if you are configuring IIS 8.0.

1. (IIS 8.0) In the list of handler mappings, select the one that you just created, and in the Actions area click Edit.
2. (IIS 8.0) In the Edit Script Map dialog box, click the Request Restrictions button.
3. (IIS 8.0) To ensure that the handler is used for files and folders that are not yet cached, deselect Invoke Handler Only If Request Is Mapped To, and then click OK.
4. (IIS 8.0) On the Edit Script Map dialog box, click OK.

Configuring Access to the Cache - IIS 8.5 and 10

Provide the default App Pool user with write-access to the folder that is being used as the Dispatcher cache.

1. Right-click the root folder of the website that you are using as the Dispatcher cache and click Properties, such as C:\inetpub\wwwroot.
2. On the Security tab, click Edit, and then on the Permissions dialog box, click Add. A dialog box opens for selecting user accounts. Click the Locations button, select your computer name, and then click OK.

Keep this dialog box open while you complete the next step.

1. in IIS Manager, select the IIS site that you are using for the Dispatcher cache, and on the right side of the window click Advanced Settings.
2. Select the value of the Application Pool property and copy it to the clipboard.
3. Return to the open dialog box. In the Enter The Object Names To Select box, type IIS AppPool\ and then paste the contents of your clipboard. The value should look like the following example:

IIS AppPool\DefaultAppPool

1. Click the Check Names button. When Windows resolves the user account, click OK.
2. In the Permissions dialog box for the dispatcher folder, select the account that you just added, enable all of the permissions for the account **except for Full Control** and click OK. Click OK to close the folder Properties dialog box.

Registering the JSON Mime Type - IIS 8.5 and 10

Use the following procedure to register the JSON MIME type, when you want Dispatcher to allow JSON calls.

1. In IIS Manager, select your web site and using Features View, double-click Mime Types.
2. If the JSON extension is not in the list, in the Actions panel click Add, enter the following property values, and then click OK:
   * File Name Extension: .json
   * MIME Type: application/json

Removing the bin Hidden Segment - IIS 8.5 and 10

Use the following procedure to remove the bin hidden segment. Web sites that are not new can contain this hidden segment.

1. In IIS Manager, select your web site and using Features View, double-click Request Filtering.
2. Select the bin segment, click Remove, and in the confirmation dialog box click Yes.

Logging IIS Messages to a File - IIS 8.5 and 10

Use the following procedure to write Dispatcher log messages to a log file instead of to the Windows Event log. You need to configure Dispatcher to use the log file, and provide IIS with write-access to the file.

1. Use Windows Explorer to create a folder named dispatcher below the logs folder of the IIS installation. The path of this folder for a typical installation is C:\inetpub\logs\dispatcher.
2. Right-click the dispatcher folder and click Properties.
3. On the Security tab, click Edit, and then on the Permissions dialog box, click Add. A dialog box opens for selecting user accounts. Click the Locations button, select your computer name, and then click OK.

Keep this dialog box open while you complete the next step.

1. in IIS Manager,select the IIS site that you are using for the Dispatcher cache, and on the right side of the window click Advanced Settings.
2. Select the value of the Application Pool property and copy it to the clipboard.
3. Return to the open dialog box. In the Enter The Object Names To Select box, type IIS AppPool\ and then paste the contents of your clipboard. The value should look like the following example:

IIS AppPool\DefaultAppPool

1. Click the Check Names button. When Windows resolves the user account, click OK.
2. In the Permissions dialog box for the dispatcher folder, select the account that you just added, enable all of the permissions for the account\*\* except for Full Control,\*\* and click OK. Click OK to close the folder Properties dialog box.
3. Use a text editor to open the disp\_iis.ini file.
4. Add a line of text similar to the following example to configure the location of the log file and then save the file:
5. logfile=C:\inetpub\logs\dispatcher\dispatcher.log

Next Steps

Before you can start using the Dispatcher you must know:

* [Configure](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html) Dispatcher
* [Confgure AEM](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/page-invalidate.html) to work with Dispatcher.

Apache Web Server

Instructions for installation under both **Windows** and **Unix** are covered here. Please be careful when performing the steps.

Installing Apache Web Server

For Information about how to install an Apache Web Server read the installation manual - either [online](https://httpd.apache.org/) or in the distribution.

If you are creating an Apache binary by compiling the source files, make sure that you turn on **dynamic modules support**. This can be done by using any of the **--enable-shared** options. At a minimum include the mod\_so module.

More information can be found in the Apache Web Server installation manual.

Also see the Apache HTTP Server [Security Tips](https://httpd.apache.org/docs/2.2/misc/security_tips.html) and [Security Reports](https://httpd.apache.org/security_report.html).

Apache Web Server - Add the Dispatcher Module

The Dispatcher comes as either:

* **Windows**: a Dynamic Link Library (DLL)
* **Unix**: a Dynamic Shared Object (DSO)

The installation archive files contains the following files - dependent on whether you have selected Windows or Unix:

| **File** | **Description** |
| --- | --- |
| disp\_apache<x.y>.dll | Windows: The Dispatcher dynamic link library file. |
| dispatcher-apache<x.y>-<rel-nr>.so | Unix: The Dispatcher shared object library file. |
| mod\_dispatcher.so | Unix: An example link. |
| http.conf.disp<x> | An example configuration file for the Apache server. |
| dispatcher.any | An example configuration file for the Dispatcher. |
| README | Readme file that contains installation instructions and last-minute information. **Note**: Please check this file before starting the installation. |
| CHANGES | Changes file that lists issues fixed in the current and past releases. |

Use the following steps to add Dispatcher to your Apache Web Server:

1. Place the Dispatcher file in the appropriate Apache module directory:
   * **Windows**: Place disp\_apache<x.y>.dll<APACHE\_ROOT>/modules
   * **Unix**: Locate either the <APACHE\_ROOT>/libexec or <APACHE\_ROOT>/modulesdirectory according to your installation.Copy dispatcher-apache<options>.so into this directory.To simplify long-term maintenance you can also create a symbolic link named mod\_dispatcher.so to the Dispatcher:ln -s dispatcher-apache<x>-<os>-<rel-nr>.so mod\_dispatcher.so
2. Copy the dispatcher.any file to the <APACHE\_ROOT>/conf directory.

**Note:** You can place this file in a different location, as long as the DispatcherLog property of the Dispatcher module is configured accordingly. (See Dispatcher-Specific Configuration Entries below.)

Apache Web Server - Configure SELinux Properties

If you are running Dispatcher on RedHat Linux Kernel 2.6 with SELinux enabled, you might run into error messages like this in the dispatcher logfile.

Mon Jun 30 00:03:59 2013] [E] [16561(139642697451488)] Unable to connect to backend rend01 (10.122.213.248:4502): Permission denied

This is likely due to an enabled SELinux security. Then you you need perform the following tasks:

* Configure the SELinux context of the dispatcher module file.
* Enable HTTPD scripts and modules to make network connections.
* Configure the SELinux context of the docroot, where the cached files are stored.

Enter the following commands in a terminal window, replacing [path to the dispatcher.so file] with the path to the Dispatcher module that you installed to Apache Web Server, and *path to the docroot* with the path where the docroot is located (e.g. /opt/cq/cache):

semanage fcontext -a -t httpd\_modules\_t [path to the dispatcher.so file]

setsebool -P httpd\_can\_network\_connect on

chcon -R --type httpd\_sys\_content\_t [path to the docroot]

semanage fcontext -a -t httpd\_sys\_content\_t "[path to the docroot](/.\*)?"

Apache Web Server - Configure Apache Web Server for Dispatcher

The Apache Web Server needs to be configured, using httpd.conf. In the Dispatcher installation kit you will find an example configuration file named httpd.conf.disp<x>.

These steps are compulsory:

1. Navigate to <APACHE\_ROOT>/conf.
2. Open httpd.conffor editing.
3. The following configuration entries must be added, in the order listed:
   * **LoadModule** to load the module on start up.
   * Dispatcher-specific configuration entries, including **DispatcherConfig, DispatcherLog** and **DispatcherLogLevel**.
   * **SetHandler** to activate the Dispatcher. **LoadModule**.
   * **ModMimeUsePathInfo** to configure behavior of **mod\_mime**.
4. (Optional) It is recommended that you change the owner of the htdocs directory:
   * The apache server starts as root, though the child processes start as daemon (for security purposes). The DocumentRoot (<APACHE\_ROOT>/htdocs) must belong to the user daemon:
   * cd <APACHE\_ROOT>
   * chown -R daemon:daemon htdocs

**LoadModule**

The following table lists examples that can be used; the exact entries are according to your specific Apache Web Server:

|  |  |
| --- | --- |
| Windows | ... LoadModule dispatcher\_module modules\disp\_apache.dll ... |
| Unix (assuming symbolic link) | ... LoadModule dispatcher\_module libexec/mod\_dispatcher.so ... |

The first parameter of each statement must be written exactly as in the above examples.

See the example configuration files provided and the Apache Web Server documentation for full details about this command.

**Dispatcher specific configuration entries**

The Dispatcher-specific configuration entries are placed after the LoadModule entry. The following table lists an example configuration that is applicable for both Unix and Windows:

**Windows & Unix**

...

<fModule disp\_apache2.c>

DispatcherConfig conf/dispatcher.any

DispatcherLog logs/dispatcher.log DispatcherLogLevel 3

DispatcherNoServerHeader 0 DispatcherDeclineRoot 0

DispatcherUseProcessedURL 0

DispatcherPassError 0

DispatcherKeepAliveTimeout 60

</IfModule>

...

The individual configuration parameters:

| **Parameter** | **Description** |
| --- | --- |
| DispatcherConfig | Location and name of the Dispatcher configuration file.  When this property is located in the main server configuration, all virtual hosts inherit the property value. However, virtual hosts can include a DispatcherConfig property to override the main server configuration. |
| DispatcherLog | Location and name of the log file. |
| DispatcherLogLevel | Log level for the log file:  0 - Errors  1 - Warnings  2 - Infos  3 - Debug  **Note**: It is recommended to set the log level to 3 during installation and testing, then to 0 when running in a production environment. |
| DispatcherNoServerHeader | *This parameter is deprecated and no longer has any effect.*  Defines the Server Header to be used:   * undefined or 0 - the HTTP server header contains the AEM version. * 1 - the Apache server header is used. |
| DispatcherDeclineRoot | Defines whether to decline requests to the root "/":  **0** - accept requests to /  **1** - requests to / are not handled by the dispatcher; use mod\_alias for the correct mapping. |
| DispatcherUseProcessedURL | Defines whether to use pre-processed URLs for all further processing by Dispatcher:  **0** - use the original URL passed to the web server.  **1** - the dispatcher uses the URL already processed by the handlers that precede the dispatcher (i.e. mod\_rewrite) instead of the original URL passed to the web server. For example, either the original or the processed URL is matched with Dispatcher filters. The URL is also used as the basis for the cache file structure. See the Apache web site documentation for information about mod\_rewrite; for example, Apache 2.2. When using mod\_rewrite, it is advisable to use the flag 'passthrough |
| DispatcherPassError | Defines how to support error codes for ErrorDocument handling:  **0** - Dispatcher spools all error responses to the client.  **1** - Dispatcher does not spool an error response to the client (where the status code is greater or equal than 400), but passes the status code to Apache, which e.g. allows an ErrorDocument directive to process such a status code.  **Code Range** - Specify a range of error codes for which the response is passed to Apache. Other error codes are passed to the client. For example, the following configuration passes responses for error 412 to the client, and all other errors are passed to Apache: DispatcherPassError 400-411,413-417 |
| DispatcherKeepAliveTimeout | Specifies the keep-alive timeout, in seconds. Starting with Dispatcher version 4.2.0 the default keep-alive value is 60. A value of 0 disables keep-alive. |
| DispatcherNoCanonURL | Setting this parameter to On will pass the raw URL to the backend instead of the canonicalised one and will override the settings of DispatcherUseProcessedURL. The default value is Off.  **Note**: The filter rules in the Dispatcher configuration will always be evaluated against the sanitised URL not the raw URL. |

Path entries are relative to the root directory of the Apache Web Server.

The default settings for the Server Header are: ServerTokens Full DispatcherNoServerHeader 0Which shows the AEM version (for statistical purposes). If you want to disable such information being available in the header you can set: ServerTokens ProdSee the [Apache Documentation about ServerTokens Directive (for example, for Apache 2.2)](https://httpd.apache.org/docs/2.2/mod/core.html) for more information.

**SetHandler**

After these entries you must add a **SetHandler** statement to the context of your configuration ( <Directory>, <Location>) for the Dispatcher to handle the incoming requests. The following example configures the Dispatcher to handle requests for the complete website:

**Windows and Unix**

...

<Directory />

<IfModule disp\\_apache2.c>

SetHandler dispatcher-handler

</IfModule>

Options FollowSymLinks

AllowOverride None

</Directory>

...

The following example configures the Dispatcher to handle requests for a virtual domain:

**Windows**

...

<VirtualHost 123.45.67.89>

ServerName www.mycompany.com

DocumentRoot \_\[cache-path\]\_\\docs

<Directory \_\[cache-path\]\_\\docs>

<IfModule disp\\_apache2.c>

SetHandler dispatcher-handler

</IfModule>

AllowOverride None

</Directory>

</VirtualHost>

...

**Unix**

...

<VirtualHost 123.45.67.89>

ServerName www.mycompany.com

DocumentRoot /usr/apachecache/docs

<Directory /usr/apachecache/docs>

<IfModule disp\\_apache2.c>

SetHandler dispatcher-handler

</IfModule>

AllowOverride None

</Directory>

</VirtualHost>

...

The parameter of the **SetHandler** statement must be written *exactly as in the above examples*, as this is the name of the handler defined in the module.

See the example configuration files provided and the Apache Web Server documentation for full details about this command.

**ModMimeUsePathInfo**

After the **SetHandler** statement you should also add the **ModMimeUsePathInfo** definition.

The ModMimeUsePathInfo parameter should only be used and configured if you are using Dispatcher version 4.0.9, or higher.

(Note that Dispatcher version 4.0.9 has been released in 2011. If you are using an older version, upgrading to a recent Dispatcher version would be appropriate).

The **ModMimeUsePathInfo** parameter should be set On for all Apache configurations:

ModMimeUsePathInfo On

The mod\_mime module (see for example, [Apache Module mod\_mime](https://httpd.apache.org/docs/2.2/mod/mod_mime.html)) is used to assign content metadata to the content selected for an HTTP response. The default setup means that when mod\_mime determines the content type, only the part of the URL that maps to a file or directory will be considered.

When On, the ModMimeUsePathInfo parameter specifies that mod\_mime is to determine the content type based on the *complete* URL; this means that virtual resources will have metainformation applied based on their extension.

The following example activates **ModMimeUsePathInfo**:

**Windows and Unix**

...

<Directory />

<IfModule disp\\_apache2.c>

SetHandler dispatcher-handler

ModMimeUsePathInfo On

</IfModule>

Options FollowSymLinks

AllowOverride None

</Directory>

...

Enable Support for HTTPS (Unix and Linux)

Dispatcher uses OpenSSL to implement secure communication over HTTP. Starting from Dispatcher version **4.2.0**, OpenSSL 1.0.0 and OpenSSL 1.0.1 are supported. Dispatcher uses OpenSSL 1.0.0 by default. To use OpenSSL 1.0.1, use the following procedure to create symbolic links so that Dispatcher uses the OpenSSL libraries that are installed.

1. Open a terminal and change the current directory to the directory where the OpenSSL libraries are installed, for example:

cd /usr/lib64

1. To create the symbolic links, enter the following commands:
2. ln -s libssl.so libssl.so.1.0.1

ln -s libcrypto.so libcrypto.so.1.0.1

If you are using a customized version of Apache, make sure Apache and Dispatcher are compiled using the same version of [OpenSSL](https://www.openssl.org/source/).

Next Steps

Before you can start using the Dispatcher you must now:

* [Configure](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html) Dispatcher
* [Confgure AEM](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/page-invalidate.html) to work with Dispatcher.

Sun Java System Web Server / iPlanet

Instructions for both Windows and Unix environments are covered here.

Please be careful when selecting which to execute.

Sun Java System Web Server / iPlanet - Installing your Web Server

For full information on how to install these web servers, please refer to their respective documentation:

* Sun Java System Web Server
* iPlanet Web Server

Sun Java System Web Server / iPlanet - Add the Dispatcher Module

The Dispatcher comes as either:

* **Windows**: a Dynamic Link Library (DLL)
* **Unix**: a Dynamic Shared Object (DSO)

The installation archive files contains the following files - dependent on whether you have selected Windows or Unix:

| **File** | **Description** |
| --- | --- |
| disp\_ns.dll | Windows: The Dispatcher dynamic link library file. |
| dispatcher.so | Unix: The Dispatcher shared object library file. |
| dispatcher.so | Unix: An example link. |
| obj.conf.disp | An example configuration file for the iPlanet / Sun Java System web server. |
| dispatcher.any | An example configuration file for the Dispatcher. |
| README | Readme file that contains installation instructions and last-minute information. Note: Please check this file before starting the installation. |
| CHANGES | Changes file that lists issues fixed in the current and past releases. |

Use the following steps to add the Dispatcher to your web server:

1. Place the Dispatcher file in the web server's plugin directory:

Sun Java System Web Server / iPlanet - Configure for the Dispatcher

The web server needs to be configured, using obj.conf. In the Dispatcher installation kit you will find an example configuration file named obj.conf.disp.

1. Navigate to <WEBSERVER\_ROOT>/config.
2. Open obj.conffor editing.
3. Copy the line that starts:Service fn="dispService"from obj.conf.disp to the initialization section of obj.conf.
4. Save the changes.
5. Open magnus.conf for editing.
6. Copy the two lines that start:Init funcs="dispService, dispInit"andInit fn="dispInit"from obj.conf.disp to the initialization section of magnus.conf.
7. Save the changes.

The following configurations should all be on one line and the $(SERVER\_ROOT) and $(PRODUCT\_SUBDIR) must be replaced by the respective values.

**Init**

The following table lists examples that can be used; the exact entries are according to your specific web server:

**Windows and Unix**

...

Init funcs="dispService,dispInit" fn="load-modules" shlib="$(SERVER\\_ROOT)/plugins/dispatcher.so"

Init fn="dispInit" config="$(PRODUCT\\_SUBDIR)/dispatcher.any" loglevel="1" logfile="$(PRODUCT\\_SUBDIR)/logs/dispatcher.log"

keepalivetimeout="60"

...

where:

| **Parameter** | **Description** |
| --- | --- |
| config | Location and name of the configuration file dispatcher.any. |
| logfile | Location and name of the log file. |
| loglevel | Log level for when writing messages to the log file:  **0** Errors  **1** Warnings  **2** Infos  **3** Debug  **Note:** It is recommended to set the log level to 3 during installation and testing and to 0 when running in a production environment. |
| keepalivetimeout | Specifies the keep-alive timeout, in seconds. Starting with Dispatcher version 4.2.0 the default keep-alive value is 60. A value of 0 disables keep-alive. |

Depending on your requirements you can define the Dispatcher as a service for your objects. To configure the Dispatcher for your entire website modify the default object:

**Windows**

...

NameTrans fn="document-root" root="$(PRODUCT\\_SUBDIR)\\dispcache"

...

Service fn="dispService" method="(GET|HEAD|POST)" type="\\*\\\\*"

...

**Unix**

...

NameTrans fn="document-root" root="$(PRODUCT\\_SUBDIR)/dispcache"

...

Service fn="dispService" method="(GET|HEAD|POST)" type="\\*/\\*"

...

Next Steps

Before you can start using the Dispatcher you must now:

* [Configure](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html) Dispatcher
* [Confgure AEM](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/page-invalidate.html) to work with Dispatcher.

**The Dispatcher Security Checklist**

The dispatcher as a front end system offers an extra layer of security to your Adobe Experience Manager infrastructure. Adobe strongly recommends that you complete the following checklist before going on production.

You must also complete the Security Checklist of your version of AEM before going live. Please refer to the corresponding [Adobe Experience Manager documentation](https://helpx.adobe.com/experience-manager/6-3/sites/administering/using/security-checklist.html).

Use the Latest Version of Dispatcher

You should install the latest available version that is available for your platform. You should upgrade your Dispatcher instance to use the latest version to take advantage of product and security enhancements. See [Installing Dispatcher](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/dispatcher-install.html).

You can check the current version of your dispatcher installation by looking at the dispatcher log file.

[Thu Apr 30 17:30:49 2015] [I] [23171(140735307338496)] Dispatcher initialized (build 4.1.9)

To find the log file, inspect the dispatcher configuration in your httpd.conf.

Restrict Clients that Can Flush Your Cache

Adobe recommends that you [limit the clients that can flush your cache.](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#limiting-the-clients-that-can-flush-the-cache)

Enable HTTPS for transport layer security

Adobe recommends to enable HTTPS transport layer on both author and publish instances.

Restrict Access

When configuring the Dispatcher you should restrict external access as much as possible. See [Example /filter Section](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#main-pars_184_1_title) in the Dispatcher documentation.

Make Sure Access to Administrative URLs is Denied

Make sure you use filters to block external access to any administrative URLs, such as the Web Console.

See [Testing Dispatcher Security](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#testing-dispatcher-security) for a list of URLs that need to be blocked.

Use Whitelists Instead Of Blacklists

Whitelists are a better way of providing access control since inherently, they assume that all access requests should be denied unless they are explicitly part of the whitelist. This model provides more restrictive control over new requests that might not have been reviewed yet or taken into consideration during a certain configuration stage.

Run Dispatcher with a Dedicated System User

When configuring the Dispatcher you should ensure that the web server is ran by a dedicated user with least privileges. It is recommended to only grant write acess to the dispatcher cache folder.

Additionnaly, IIS users need to configure their website as follows:

1. In the physical path setting for your web site, select **Connect as specific user**.
2. Set the user.

Prevent Denial of Service (DoS) Attacks

A denial of service (DoS) attack is an attempt to make a computer resource unavailable to its intended users.

At the dispatcher level, there are two methods of configuring to prevent DoS attacks: [filter](https://docs.adobe.com/content/docs/en/dispatcher.html#/filter)

* Use the mod\_rewrite module (for example, [Apache 2.2](https://httpd.apache.org/docs/2.2/mod/mod_rewrite.html)) to perform URL validations (if the URL pattern rules are not too complex).
* Prevent the dispatcher from caching URLs with spurious extensions by using [filters](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#configuring-access-to-conten-tfilter).For example, change the caching rules to limit caching to the expected mime types, such as:
  + .html
  + .jpg
  + .gif
  + .swf
  + .js
  + .doc
  + .pdf
  + .ppt

An example configuration file can be seen for [restricting external access](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/security-checklist.html#restrict-access), this includes restrictions for mime types.

To safely enable full functionality on the publish instances, configure filters to prevent access to the following nodes:

* /etc/
* /libs/

Then, configure filters to allow access to the following node paths:

* /etc/designs/\*
* /etc/clientlibs/\*
* /etc/segmentation.segment.js
* /libs/cq/personalization/components/clickstreamcloud/content/config.json
* /libs/wcm/stats/tracker.js
* /libs/cq/personalization/\* (JS, CSS and JSON)
* /libs/cq/security/userinfo.json (CQ user information)
* /libs/granite/security/currentuser.json (**data must not be cached**)
* /libs/cq/i18n/\* (Internalization)

Configure Dispatcher to prevent CSRF Attacks

AEM provides a [framework](https://helpx.adobe.com/experience-manager/6-3/sites/administering/using/security-checklist.html#verification-steps) aimed at preventing Cross-Site Request Forgery attacks. In order to properly make use of this framework, you need to whitelist CSRF token support in the dispatcher. You can do this by:

1. Creating a filter to allow the /libs/granite/csrf/token.json path;
2. Add the CSRF-Token header to the clientheaders section of the Dispatcher configuration.

Prevent Clickjacking

To prevent clickjacking we recommend that you configure your webserver to provide the X-FRAME-OPTIONS HTTP header set to SAMEORIGIN.

For more [information on clickjacking please see the OWASP site](https://www.owasp.org/index.php/Clickjacking).

Perform a Penetration Test

Adobe strongly recommends to perform a penetration test of your AEM infrastructure before going on production.

# Configuring Dispatcher

Dispatcher versions are independent of AEM. You may have been redirected to this page if you followed a link to the Dispatcher documentation that is embedded in the documentation for a previous version of AEM.

The following sections describe how to configure various aspects of the Dispatcher.

## Support for IPv4 and IPv6

All elements of AEM and Dispatcher can be installed in both IPv4 and IPv6 networks. See [IPV4 and IPV6](https://helpx.adobe.com/experience-manager/6-3/sites/deploying/using/technical-requirements.html#AdditionalPlatformNotes).

## Dispatcher Configuration Files

By default the Dispatcher configuration is stored in the dispatcher.any text file, though you can change the name and location of this file during installation.

The configuration file contains a series of single-valued or multi-valued properties that control the behavior of Dispatcher:

* Property names are prefixed with a forward slash /.
* Multi-valued properties enclose child items using braces { }.

An example configuration is structured as follows:

# name of the dispatcher

/name "internet-server"

# each farm configures a set off (loadbalanced) renders

/farms

{

# first farm entry (label is not important, just for your convenience)

/website

{

/clientheaders

{

# List of headers that are passed on

}

/virtualhosts

{

# List of URLs for this Web site

}

/sessionmanagement

{

# settings for user authentification

}

/renders

{

# List of AEM instances that render the documents

}

/filter

{

# List of filters

}

/vanity\_urls

{

# List of vanity URLs

}

/cache

{

# Cache configuration

/rules

{

# List of cachable documents

}

/invalidate

{

# List of auto-invalidated documents

}

}

/statistics

{

/categories

{

# The document categories that are used for load balancing estimates

}

}

/stickyConnectionsFor "/myFolder"

/health\_check

{

# Page gets contacted when an instance returns a 500

}

/retryDelay "1"

/numberOfRetries "5"

/unavailablePenalty "1"

/failover "1"

}

}

You can include other files that contribute to the configuration:

* If your configuration file is large you can split it into several smaller files (that are easier to manage) then include these.
* To include files that are generated automatically.

For example, to include the file myFarm.any in the /farms configuration use the following code:

/farms

{

$include "myFarm.any"

}

Use the asterisk ("\*") as a wildcard to specify a range of files to include.

For example, if the files farm\_1.any through to farm\_5.any contain the configuration of farms one to five, you can include them as follows:

/farms

{

$include "farm\_\*.any"

}

## Using Environment Variables

You can use environment variables in string-valued properties in the dispatcher.any file instead of hard-coding the values. To include the value of an environment variable, use the format ${variable\_name}.

For example, if the dispatcher.any file is located in the same directory as the cache directory, the following value for the [docroot](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#main-pars-title-30) property can be used:

/docroot "${PWD}/cache"

As another example, if you create an environment variable named PUBLISH\_IP that stores the hostname of the AEM publish instance, the following configuration of the [/renders](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#main-pars-127-25-0008) property can be used:

/renders {

/0001 {

/hostname "${PUBLISH\_IP}"

/port "8443"

}

}

## Naming the Dispatcher Instance

Use the /name property to specify a unique name to identify your Dispatcher instance. The /name property is a top-level property in the configuration structure.

## Defining Farms

The /farms property defines one or more sets of Dispatcher behaviors, where each set is associated with different web sites or URLs. The /farmsproperty can include a single farm or multiple farms:

* Use a single farm when you want Dispatcher to handle all of your web pages or web sites in the same way.
* Create multiple farms when different areas of your web site or different web sites require different Dispatcher behavior.

The /farms property is a top-level property in the configuration structure. To define a farm, add a child property to the /farms property. Use a property name that uniquely identifies the farm within the Dispatcher instance.

The /\*farmname\* property is multi-valued, and contains other properties that define Dispatcher behavior:

* The URLs of the pages that the farm applies to.
* One or more service URLs (typically of AEM publish instances) to use for rendering documents.
* The statistics to use for load-balancing multiple document renderers.
* Several other behaviors, such as which files to cache and where.

The value can have include any alphanumeric (a-z, 0-9) character. The following example shows the skeleton definition for two farms named /daycom and /docsdaycom:

#name of dispatcher

/name "day sites"

#farms section defines a list of farms or sites

/farms

{

/daycom

{

...

}

/docdaycom

{

...

}

}

If you use more than one render farm, the list is evaluated bottom-up. This is particularly relevant when defining [Virtual Hosts](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#main-pars-117-15-0006) for your websites.

Each farm property can contain the following child properties:

| **Property name** | **Description** |
| --- | --- |
| [/homepage](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#specify-a-default-page-iis-only-homepage) | Default homepage (optional)(IIS only) |
| [/clientheaders](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#specifying-the-http-headers-to-pass-through-clientheaders) | The headers from the client HTTP request to pass through. |
| [/virtualhosts](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#identifying-virtual-hosts-virtual-hosts) | The virtual hosts for this farm. |
| [/sessionmanagement](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#enabling-secure-sessions-session-management) | Support for session management and authentication. |
| [/renders](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#defining-page-renderers-renders) | The servers that provide rendered pages (typically AEM publish instances). |
| [/filter](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#configuring-access-to-content-filter) | Defines the URLs to which Dispatcher enables access. |
| [/vanity\_urls](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#enabling-access-to-vanity-urls-vanity-urls) | Configures access to vanity URLs. |
| [/propagateSyndPost](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#forwarding-syndication-requests-propagate-syndpost) | Support for the forwarding of syndication requests. |
| [/cache](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#configuring-the-dispatcher-cache-cache) | Configures caching behavior. |
| [/statistics](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#configuring-load-balancing-statistics) | Defining statistic categories for load-balancing calculations. |
| [/stickyConnectionsFor](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#identifying-a-sticky-connection-folder-sticky-connections-for) | The folder that contains sticky documents. |
| [/health\_check](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#specifying-a-health-check-page) | The URL to use to determine server availability. |
| [/retryDelay](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#specifying-the-page-retry-delay) | The delay before retrying a failed connection. |
| [/unavailablePenalty](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#reflecting-server-unavailability-in-dispatcher-statistics) | Penalties that affect statistics for load-balancing calculations. |
| [/failover](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#using-the-fail-over-mechanism) | Resend requests to different renders when the original request fails. |

## Specify a Default Page (IIS Only) - /homepage

The /homepageparameter (IIS only) no longer works. Instead, you should use the [IIS URL Rewrite Module](https://docs.microsoft.com/en-us/iis/extensions/url-rewrite-module/using-the-url-rewrite-module).

If you are using Apache, you should use the mod\_rewrite module. See the Apache web site documentation for information about mod\_rewrite (for example, [Apache 2.4](https://httpd.apache.org/docs/current/mod/mod_rewrite.html)). When using mod\_rewrite, it is advisable to use the flag \*\* ['passthrough|PT' (pass through to next handler)](https://helpx.adobe.com/dispatcher/kb/DispatcherModReWrite.html)\*\* to force the rewrite engine to set the uri field of the internal request\_rec structure to the value of the filename field.

## Specifying the HTTP Headers to Pass Through

The /clientheaders property defines a list of HTTP headers that Dispatcher passes from the client HTTP request to the renderer (AEM instance).

By default Dispatcher forwards the standard HTTP headers to the AEM instance. In some instances, you might want forward additional headers, or remove specific headers:

* Add headers, such as custom headers, that your AEM instance expects in the HTTP request.
* Remove headers, such as authentication headers, that are only relevant to the web server.

If you customize the set of headers to pass through, you must specify an exhaustive list of headers, including those that are normally included by default.

For example, a Dispatcher instance that handles page activation requests for publish instances requires the PATH header in the /clientheaders section. The PATH header enables communication between the replication agent and the dispatcher.

The following code is an example configuration for /clientheaders:

/clientheaders

{

"CSRF-Token"

"X-Forwarded-Proto"

"referer"

"user-agent"

"authorization"

"from"

"content-type"

"content-length"

"accept-charset"

"accept-encoding"

"accept-language"

"accept"

"host"

"if-match"

"if-none-match"

"if-range"

"if-unmodified-since"

"max-forwards"

"proxy-authorization"

"proxy-connection"

"range"

"cookie"

"cq-action"

"cq-handle"

"handle"

"action"

"cqstats"

"depth"

"translate"

"expires"

"date"

"dav"

"ms-author-via"

"if"

"lock-token"

"x-expected-entity-length"

"destination"

"PATH"

}

## Identifying Virtual Hosts

The /virtualhosts property defines a list of all hostname/URI combinations that Dispatcher accepts for this farm. You can use the asterisk ("\*") character as a wildcard. Values for the / virtualhosts property use the following format:

[scheme]host[uri][\*]

* scheme: (Optional) Either https:// or https://.
* host: The name or IP address of the host computer and the port number if necessary. (See <https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.23>)
* uri: (Optional) The path to the resources.

The following example configuration handles requests for the .com and .ch domains of myCompany, and all domains of mySubDivision:

/virtualhosts

{

"www.myCompany.com"

"www.myCompany.ch"

"www.mySubDivison.\*"

}

The following configuration handles *all* requests:

/virtualhosts

{

"\*"

}

### Resolving the Virtual Host

When Dispatcher receives an HTTP or HTTPS request, it finds the virtual host value that best-matches the host,uri, and scheme headers of the request. Dispatcher evaluates the values in the virtualhosts properties in the following order:

* Dispatcher begins at the lowest farm and progresses upward in the dispatcher.any file.
* For each farm, Dispatcher begins with the topmost value in the virtualhosts property and progresses down the list of values.

Dispatcher finds the best-matching virtual host value in the following manner:

* The first-encountered virtual host that matches all three of the host, the scheme, and the uri of the request is used.
* If no virtualhosts values has scheme and uri parts that both match the scheme and uri of the request, the first-encountered virtual host that matches the host of the request is used.
* If no virtualhosts values have a host part that matches the host of the request, the topmost virtual host of the topmost farm is used.

Therefore, you should place your default virtual host at the top of the virtualhosts property in the topmost farm of your dispatcher.any file.

### Example Virtual Host Resolution

The following example represents a snippet from a dispatcher.any file that defines two Dispatcher farms, and each farm defines a virtualhosts property.

/farms

{

/myProducts

{

/virtualhosts

{

"www.mycompany.com"

}

/renders

{

/hostname "server1.myCompany.com"

/port "80"

}

}

/myCompany

{

/virtualhosts

{

"www.mycompany.com/products/\*"

}

/renders

{

/hostname "server2.myCompany.com"

/port "80"

}

}

}

Using this example, the following table shows the virtual hosts that are resolved for the given HTTP requests:

| **Request URL** | **Resolved virtual host** |
| --- | --- |
| https://www.mycompany.com/products/gloves.html | www.mycompany.com/products/\*; |
| https://www.mycompany.com/about.html | www.mycompany.com |

## Enabling Secure Sessions - /sessionmanagement

/allowAuthorized**must** be set to "0" in the /cache section in order to enable this feature.

Create a secure session for access to the render farm so that users need to log in to access any page in the farm. After logging in, users can access all pages in the farm. See [Creating a Closed User Group](https://helpx.adobe.com/experience-manager/6-3/sites/administering/using/cug.html#CreatingTheUserGroupToBeUsed) for information about using this feature with CUGs.

The /sessionmanagement property is a subproperty of /farms.

If sections of your website use different access requirements, you need to define multiple farms.

**/sessionmanagement** has several sub-parameters:

**/directory** (mandatory)

The directory that stores the session information. If the directory does not exist, it is created.

**/encode** (optional)

How the session information is encoded. Use "md5" for encryption using the md5 algorithm, or "hex" for hexadecimal encoding. If you encrypt the session data, a user with access to the file system cannot read the session contents. The default is "md5".

**/header** (optional)

The name of the HTTP header or cookie that stores the authorization information. If you store the information in the http header, use HTTP:<\*header-name\*>. To store the information in a cookie, use Cookie:<header-name>. If you do not specify a value HTTP:authorization is used.

**/timeout** (optional)

The number of seconds until the session times out after it has been used last. If not specified "800" is used, so the session times out a little over 13 minutes after the last request of the user.

An example configuration looks as follows:

/sessionmanagement

{

/directory "/usr/local/apache/.sessions"

/encode "md5"

/header "HTTP:authorization"

/timeout "800"

}

## Defining Page Renderers

The /renders property defines the URL to which Dispatcher sends requests to render a document. The following example /renders section identifies a single AEM instance for rendering:

/renders

{

/myRenderer

{

# hostname or IP of the renderer

/hostname "aem.myCompany.com"

# port of the renderer

/port "4503"

# connection timeout in milliseconds, "0" (default) waits indefinitely

/timeout "0"

}

}

The following example /renders section identifies an AEM instance that runs on the same computer as Dispatcher:

/renders

{

/myRenderer

{

/hostname "127.0.0.1"

/port "4503"

}

}

The following example /renders section distributes render requests equally among two AEM instances:

/renders

{

/myFirstRenderer

{

/hostname "aem.myCompany.com"

/port "4503"

}

/mySecondRenderer

{

/hostname "127.0.0.1"

/port "4503"

}

}

### Renders options

**/timeout**

Specifies the connection timeout accessing the AEM instance in milliseconds. The default is "0", causing the Dispatcher to wait indefinitely.

**/receiveTimeout**

Specifies the time in milliseconds that a response is allowed to take. The default is "600000", causing Dispatcher to wait for 10 Minutes. A setting of "0" eliminates the timeout completely.If the timeout is reached while parsing response headers, an HTTP Status of 504 (Bad Gateway) is returned. If the timeout is reached while the response body is read, the Dispatcher will return the incomplete response to the client but delete any cache file that might have been written.

**/ipv4**

Specifies whether Dispatcher uses the getaddrinfo function (for IPv6) or the gethostbyname function (for IPv4) for obtaining the IP address of the render. A value of 0 causes getaddrinfo to be used. A value of 1 causes gethostbyname to be used. The default value is 0.

The getaddrinfo function returns a list of IP addresses. Dispatcher iterates the list of addresses until it establishes a TCP/IP connection. Therefore, the ipv4 property is important when the render hostname is associated with mutliple IP addresses and the host, in response to the getaddrinfo function, returns a list of IP addresses that are always in the same order. In this situation, you should use the gethostbyname function so that the IP address that Dispatcher connects with is randomized.

Amazon Elastic Load Balancing (ELB) is such a service that responds to getaddrinfo with a potentially same-ordered list of IP addresses.

**/secure**

If the /secure property has a value of "1" Dispatcher uses HTTPS to communicate with the AEM instance. For additional details, also see [Configuring Dispatcher to Use SSL](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-ssl.html#configuring-dispatcher-to-use-ssl).

**/always-resolve**

With Dispatcher version **4.1.6**, you can configure the /always-resolve property as follows:

* When set to "1" it will resolve the host-name on every request (the Dispatcher will never cache any IP address). There may be a slight performance impact due to the additional call required to get the host information for each request.
* If the property is not set, the IP address will be cached by default.

Also, this property can be used in case you run into dynamic IP resolution issues, as shown in the following sample:

/rend {

/0001 {

/hostname "host-name-here"

/port "4502"

/ipv4 "1"

/always-resolve "1"

}

}

## Configuring Access to Content

Use the /filter section to specify the HTTP requests that Dispatcher accepts. All other requests are sent back to the web server with a 404 error code (page not found). If no /filter section exists, all requests are accepted.

**Note:** Requests for the [statfile](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#main-pars-title-28) are always rejected.

See the [Dispatcher Security Checklist](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/security-checklist.html) for further considerations when restricting access using Dispatcher. Also, read the [AEM Security Cheklist](https://helpx.adobe.com/experience-manager/6-3/sites/administering/using/security-checklist.html) for additional security details regarding your AEM installation.

The /filter section consist of a series of rules that either deny or allow access to content according to patterns in the request-line part of the HTTP request. You should use a whiltelist strategy for your /filter section:

* First, deny access to everything.
* Allow access to content as needed.

### Defining a Filter

Each item in the /filter section includes a type and a pattern that is matched with a specific element of the request line or the entire request line. Each filter can contain the following items:

* **Type**: The /type indicates whether to allow or deny access for the requests that match the pattern. The value can be either allow or deny.
* **Element of the Request Line:** Include /method, /url, /query, or /protocol and a pattern for filtering requests according to these specific parts of the request-line part of the HTTP request. Filtering on elements of the request line (rather than on the entire request line) is the preferred filter method.
* **Advanced Elements of the Request Line:** Starting with Dispatcher 4.2.0, four new filter elements are available for use. These new elements are /path, /selectors, /extension and /suffix respectively. Include one or more of these items to further control URL patterns.

For more information about what part of the request line each of these elements references, see the [Sling URL Decomposition](https://sling.apache.org/documentation/the-sling-engine/url-decomposition.html) wiki page.

* **glob Property**: The /glob property is used to match with the entire request-line of the HTTP request.

Filtering with globs is deprecated in Dispatcher. As such, you should avoid using globs in the /filter sections since it may lead to security issues. So, instead of:

/glob "\* \*.css \*"

you should use

/url "\*.css"

#### The request-line Part of HTTP Requests

HTTP/1.1 defines the [request-line](https://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html) as follows:

*Method Request-URI HTTP-Version*<CRLF>

The <CRLF> characters repesent a carriage return followed by a line feed. The following example is the request-line that is recieved when a client requests the en page of the Geometrixx-Outoors site:

GET /content/geometrixx-outdoors/en.html HTTP.1.1<CRLF>

Your patterns must take into account the space characters in the request-line and the <CRLF> characters.

#### Double-quotes vs Single-quotes

When creating your filter rules, use double quotation marks "pattern" for simple patterns. If you are using Dispatcher 4.2.0 or later and your pattern includes a regular expression, you must enclose the regex pattern '(pattern1|pattern2)' within single quotation marks.

#### Regular Expressions

After Dispatcher 4.2.0, you can include POSIX Extended Regular Expressions in your filter patterns.

#### Troubleshooting Filters

If your filters are not triggering in the way you would expect, enable [Trace Logging](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#trace-logging) on dispatcher so you can see which filter is intercepting the request.

#### Example Filter: Deny All

The following example filter section causes Dispatcher to deny requests for all files. You should deny access to all files and then allow access to specific areas.

/0001 { /glob "\*" /type "deny" }

Requests to an explicitly denied area result in a 404 error code (page not found) being returned.

#### Example Filter: Deny Acess to Specific Areas

Filters also allow you to deny access to various elements for example ASP pages and sensitive areas within a publish instance. The following filter denies access to ASP pages:

/0002 { /type "deny" /url "\*.asp" }

#### Example Filter: Enable POST Requests

The following example filter allows submitting form data by the POST method:

/filter {

/0001 { /glob "\*" /type "deny" }

/0002 { /type "allow" /method "POST" /url "/content/[.]\*.form.html" }

}

#### Example Filter: Allow Access to the Workflow Console

The following example shows a filter used to deny external access to the Workflow console:

/filter {

/0001 { /glob "\*" /type "deny" }

/0002 { /type "allow" /url "/libs/cq/workflow/content/console\*" }

}

If your publish instance uses a web application context (for example publish) this can also be added to your filter definition.

/0003 { /type "deny" /url "/publish/libs/cq/workflow/content/console/archive\*" }

If you still need to access single pages within the restricted area, you can allow access to them. For example, to allow access to the Archive tab within the Workflow console add the following section:

/0004 { /type "allow" /url "/libs/cq/workflow/content/console/archive\*" }

When multiple filters patterns apply to a request, the last filter pattern that applies is effective.

#### Example filter: Using Regular Expressions

This filter enables extensions in non-public content directories using a regular expression, defined here between single quotes:

/005 { /type "allow" /extension '(css|gif|ico|js|png|swf|jpe?g)' }

#### Example filter: Filter Additional Elements of a Request URL

Below is a rule example that blocks content grabbing from the /content path and its subtree, using filters for path, selectors and extensions:

/006 {

/type "deny"

/path "/content/\*"

/selectors '(feed|rss|pages|languages|blueprint|infinity|tidy)'

/extension '(json|xml|html)'

}

### Example /filter section

When configuring Dispatcher you should restrict external access as much as possible. The following example provides minimal access for external visitors:

* /content
* miscellaneous content such as designs and client libraries; for example:
  + /etc/designs/default\*
  + /etc/designs/mydesign\*

After you create filters, [test page access](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#main-pars-title-19) to ensure your AEM instance is secure.

The following /filter section of the dispatcher.any file can be used as a basis in your [Dispatcher configuration](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html) file.

This example is based on the default configuration file that is provided with Dispatcher and is intended as an example for use in a production environment. Items prefixed with # are deactivated (commented out), care should be taken if you decide to activate any of these (by removing the # on that line) as this can have a security impact.

You should deny access to everything, then allow access to specific (limited) elements:

/filter

{

# Deny everything first and then allow specific entries

/0001 { /type "deny" /glob "\*" }

# Open consoles

# /0011 { /type "allow" /url "/admin/\*" } # allow servlet engine admin

# /0012 { /type "allow" /url "/crx/\*" } # allow content repository

# /0013 { /type "allow" /url "/system/\*" } # allow OSGi console

# Allow non-public content directories

# /0021 { /type "allow" /url "/apps/\*" } # allow apps access

# /0022 { /type "allow" /url "/bin/\*" }

/0023 { /type "allow" /url "/content\*" } # disable this rule to allow mapped content only

# /0024 { /type "allow" /url "/libs/\*" }

# /0025 { /type "deny" /url "/libs/shindig/proxy\*" } # if you enable /libs close access to proxy

# /0026 { /type "allow" /url "/home/\*" }

# /0027 { /type "allow" /url "/tmp/\*" }

# /0028 { /type "allow" /url "/var/\*" }

# Enable extensions in non-public content directories, using a regular expression

/0041

{

/type "allow"

/extension '(css|gif|ico|js|png|swf|jpe?g)'

}

# Enable features

/0062 { /type "allow" /url "/libs/cq/personalization/\*" } # enable personalization

# Deny content grabbing, on all accessible pages, using regular expressions

/0081

{

/type "deny"

/selectors '((sys|doc)view|query|[0-9-]+)'

/extension '(json|xml)'

}

# Deny content grabbing for /content and its subtree

/0082

{

/type "deny"

/path "/content/\*"

/selectors '(feed|rss|pages|languages|blueprint|infinity|tidy)'

/extension '(json|xml|html)'

}

# /0087 { /type "allow" /method "GET" /extension 'json' "\*.1.json" } # allow one-level json requests

}

When used with Apache, design your filter URL patterns according to the DispatcherUseProcessedURL property of the Dispatcher module. (See [Apache Web Server - Configure your Apache Web Server for Dispatcher](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/dispatcher-install.html#main-pars-55-35-1022).)

Filters 0030 and 0031 regarding Dynamic Media are applicable to AEM 6.0 and higher.

Consider the following recommendations if you do choose to extend access:

* External access to /admin should always be *completely* disabled if you are using CQ version 5.4 or an earlier version.
* Care must be taken when allowing access to files in /libs. Access should be allowed on an individual basis.
* Deny access to the replication configuration so it cannot be seen:
  + /etc/replication.xml\*
  + /etc/replication.infinity.json\*
* Deny access to the Google Gadgets reverse proxy:
  + /libs/opensocial/proxy\*

Depending on your installation, there might be additional resources under /libs, /apps or elsewhere, that must be made available. You can use the access.log file as one method of determining resources that are being accessed externally.

Access to consoles and directories can present a security risk for production environments. Unless you have explicit justifications they should remain deactivated (commented out).

If you are [using reports in a publish environment](https://helpx.adobe.com/experience-manager/6-3/sites/administering/using/reporting.html#UsingReportsinaPublishEnvironment) you should configure Dispatcher to deny access to /etc/reports for external visitors.

### Restricting Query Strings

Since Dispatcher version 4.1.5, use the /filter section to restrict query strings. It is highly recommended to explicitly allow query strings and exclude generic allowance through allow filter elements.

A single entry can have either *glob* or some combination of *method*,*url*,*query* and *version* but not both. The following example allows the a=\* query string and denies all other query strings for URLs that resolve to the /etc node:

/filter {

/0001 { /type "deny" /method "POST" /url "/etc/\*" }

/0002 { /type "allow" /method "GET" /url "/etc/\*" /query "a=\*" }

}

If a rule contains a /query, it will only match requests that contain a query string and match the provided query pattern.

In above example, if requests to /etc that have no query string should be allowed as well, the following rules would be required:

/filter {

>/0001 { /type "deny" /method “\*" /url "/path/\*" }

>/0002 { /type "allow" /method "GET" /url "/path/\*" }

>/0003 { /type “deny" /method "GET" /url "/path/\*" /query "\*" }

>/0004 { /type "allow" /method "GET" /url "/path/\*" /query "a=\*" }

}

### Testing Dispatcher Security

Dispatcher filters should block access to the following pages and scripts on AEM publish instances. Use a web browser to attempt to open the following pages as a site visitor would and verify that a code 404 is returned. If any other result is obtained, adjust your filters.

Note that you should see normal page rendering for /content/add\_valid\_page.html?debug=layout.

* /admin
* /system/console
* /dav/crx.default
* /crx
* /bin/crxde/logs
* /jcr:system/jcr:versionStorage.json
* /\_jcr\_system/\_jcr\_versionStorage.json
* /libs/wcm/core/content/siteadmin.html
* /libs/collab/core/content/admin.html
* /libs/cq/ui/content/dumplibs.html
* /var/linkchecker.html
* /etc/linkchecker.html
* /home/users/a/admin/profile.json
* /home/users/a/admin/profile.xml
* /libs/cq/core/content/login.json
* /content/../libs/foundation/components/text/text.jsp
* /content/.{.}/libs/foundation/components/text/text.jsp
* /apps/sling/config/org.apache.felix.webconsole.internal.servlet.OsgiManager.config/jcr%3acontent/jcr%3adata
* /libs/foundation/components/primary/cq/workflow/components/participants/json.GET.servlet
* /content.pages.json
* /content.languages.json
* /content.blueprint.json
* /content.-1.json
* /content.10.json
* /content.infinity.json
* /content.tidy.json
* /content.tidy.-1.blubber.json
* /content/dam.tidy.-100.json
* /content/content/geometrixx.sitemap.txt
* /content/add\_valid\_page.query.json?statement=//\*
* /content/add\_valid\_page.qu%65ry.js%6Fn?statement=//\*
* /content/add\_valid\_page.query.json?statement=//\*[#](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html)/(@transportPassword%20|%20@transportUri%20|%20@transportUser)
* /content/add\_valid\_path\_to\_a\_page/\_jcr\_content.json
* /content/add\_valid\_path\_to\_a\_page/jcr:content.json
* /content/add\_valid\_path\_to\_a\_page/\_jcr\_content.feed
* /content/add\_valid\_path\_to\_a\_page/jcr:content.feed
* /content/add\_valid\_path\_to\_a\_page/pagename.\_jcr\_content.feed
* /content/add\_valid\_path\_to\_a\_page/pagename.jcr:content.feed
* /content/add\_valid\_path\_to\_a\_page/pagename.docview.xml
* /content/add\_valid\_path\_to\_a\_page/pagename.docview.json
* /content/add\_valid\_path\_to\_a\_page/pagename.sysview.xml
* /etc.xml
* /content.feed.xml
* /content.rss.xml
* /content.feed.html
* /content/add\_valid\_page.html?debug=layout
* /projects
* /tagging
* /etc/replication.html
* /etc/cloudservices.html
* /welcome

Issue the following command in a terminal or command prompt to determine whether anonymous write access is enabled. You should not be able to write data to the node.

curl -X POST "https://anonymous:anonymous@hostname:port/content/usergenerated/mytestnode"

Issue the following command in a terminal or command prompt to attempt to invalidate the Dispatcher cache, and ensure that you recieve a code 404 response:

curl -H "CQ-Handle: /content" -H "CQ-Path: /content" https://yourhostname/dispatcher/invalidate.cache

## Enabling Access to Vanity URLs

Configure Dispatcher to enable access to vanity URLs that are configured for your CQ or AEM pages.

When access to vanity URLs is enabled, Dispatcher periodically calls a service that runs on the render instance to obtain a list of vanity URLs. Dispatcher stores this list in a local file. When a request for a page is denied due to a filter in the /filter section, Dispatcher consults the list of vanity URLs. If the denied URL is on the list, Dispatcher allows access to the vanity URL.

To enable access to vanity URLs, add a /vanity\_urls section to the /farms section, similar to the following example:

/vanity\_urls {

/url "/libs/granite/dispatcher/content/vanityUrls.html"

/file "/tmp/vanity\_urls"

/delay 300

}

The /vanity\_urls section contains the following properties:

* /url: The path to the vanity URL service that runs on the render instance. The value of this property must be "/libs/granite/dispatcher/content/vanityUrls.html".
* /file: The path to the local file where Dispatcher stores the list of vanity URLs. Make sure that Dispatcher has write-access to this file.
* /delay: (Seconds) The time between calls to the vanity URL service.

If your render is an instance of AEM you must install the [VanityURLS-Components](https://www.adobeaemcloud.com/content/marketplace/marketplaceProxy.html?packagePath=/content/companies/public/adobe/packages/cq600/component/vanityurls-components) package to install the vanity URL service. (See [Signing In to Package Share](https://helpx.adobe.com/experience-manager/6-3/sites/administering/using/package-manager.html#SigningIntoPackageShare).)

Use the following procedure to enable access to vanity URLs.

1. If your render service is an AEM instance, install the com.adobe.granite.dispatcher.vanityurl.content package on the publish instance (see the note above).
2. For each vanity URL that you have configured for an AEM or CQ page, ensure that the [/filter](dispatcher-configuration.md#main-pars\_134\_32\_0009) configuration denies the URL. If necessary, add a filter that denies the URL.
3. Add the /vanity\_urls section below /farms.
4. Restart Apache web server.

## Forwarding Syndication Requests - /propagateSyndPost

Syndication requests are usually intended for Dispatcher only, so by default they are not sent to the renderer (for example, an AEM instance).

If necessary, set the /propagateSyndPost property to "1" to forward syndication requests to Dispatcher. If set, you must make sure that POST requests are not denied in the filter section.

## Configuring the Dispatcher Cache - /cache

The /cache section controls how Dispatcher caches documents. Configure several sub-properties to implement your caching strategies:

* /docroot
* /statfile
* /serveStaleOnError
* /allowAuthorized
* /rules
* /statfileslevel
* /invalidate
* /invalidateHandler
* /allowedClients
* /ignoreUrlParams
* /headers
* /mode
* /gracePeriod

An example cache section might look as follows:

/cache

{

/docroot "/opt/dispatcher/cache"

/statfile "/tmp/dispatcher-website.stat"

/allowAuthorized "0"

/rules

{

# List of files that are cached

}

/invalidate

{

# List of files that are auto-invalidated

}

}

For permission-sensitive caching, read [Caching Secured Content](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/permissions-cache.html).

### Specifying the Cache Directory

The /docroot property identifies the directory where cached files are stored.

The value must be the exact same path as the document root of the web server so that Dispatcher and the web server handle the same files.The web server is responsible for delivering the correct status code when the dispatcher cache file is used, that's why it is important that it can find it as well.

If you use multiple farms, each farm must use a different document root.

### Naming the Statfile

The /statfile property identifies the file to use as the statfile. Dispatcher uses this file to register the time of the most recent content update. The statfile can be any file on the web server.

The statfile has no content. When content is updated, Dispatcher updates the timestamp. The default statfile is named .stat and is stored in the docroot. Dispatcher blocks access to the statfile.

If /statfileslevel is configured, Dispatcher ignores the /statfile property and uses .stat as the name.

### Serving Stale Documents When Errors Occur

The /serveStaleOnError property controls whether Dispatcher returns invalidated documents when the render server returns an error. By default, when a statfile is touched and invalidates cached content, Dispatcher deletes the cached content the next time it is requested.

If /serveStaleOnError is set to "1", Dispatcher does not delete invalidated content from the cache unless the render server returns a successful response. A 5xx response from AEM or a connection timeout causes Dispatcher to serve the outdated content and respond with and HTTP Status of 111 (Revalidation Failed).

### Caching When Authentication is Used

The /allowAuthorized property controls whether requests that contain any of the following authentication information are cached:

* The authorization header.
* A cookie named authorization.
* A cookie named login-token.

By default, requests that include this authentication information are not cached because authentication is not performed when a cached document is returned to the client. This configuration prevents Dispatcher from serving cached documents to users who do not have the necessary rights.

However, if your requirements permit the caching of authenticated documents, set /allowAuthorized to one:

/allowAuthorized "1"

To enable session management (using the /sessionmanagement property), the /allowAuthorized property must be set to "0".

### Specifying the Documents to Cache

The /rules property controls which documents are cached according to the document path. Regardless of the /rules property, Dispatcher never caches a document in the following circumstances:

* If the request URI contains a question mark ("?").This usually indicates a dynamic page, such as a search result that does not need to be cached.
* The file extension is missing.The web server needs the extension to determine the document type (the MIME-type).
* The authentication header is set (this can be configured)
* If the AEM instance responds with the following headers:
  + no-cache
  + no-store
  + must-revalidate

The GET or HEAD (for the HTTP header) methods are cacheable by the Dispatcher. For additional information on response header caching, see the [Caching HTTP Response Headers](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#caching-http-response-headers) section.

Each item in the /rules property includes a [glob](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#designing-patterns-for-glob-properties) pattern and a type:

* The glob pattern is used to match the path of the document.
* The type indicates whether to cache the documents that match the glob pattern. The value can be either allow (to cache the document) or deny (to always render the document).

If you do not have dynamic pages (beyond those already excluded by the above rules), you can configure Dispatcher to cache everything. The rules section for this looks as follows:

/rules

{

/0000 { /glob "\*" /type "allow" }

}

For information about glob properties, see [Designing Patterns for glob Properties](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#designing-patterns-for-glob-properties).

If there are some sections of your page that are dynamic (for example a news application) or within a closed user group, you can define exceptions:

Closed user groups must not be cached as user rights are not checked for cached pages.

/rules

{

/0000 { /glob "\*" /type "allow" }

/0001 { /glob "/en/news/\*" /type "deny" }

/0002 { /glob "\*/private/\*" /type "deny" }

}

**Compression**

On Apache web servers you can compress the cached documents. Compression allows Apache to return the document in a compressed form if so requested by the client. Compression is done automatically by enabling the Apache module mod\_deflate, for example:

AddOutputFilterByType DEFLATE text/plain

The module is installed by default with Apache 2.x.

### Invalidating Files by Folder Level

Use the /statfileslevel property to invalidate cached files according to their path:

* Dispatcher creates .statfiles in each folder from the docroot folder to the level that you specify. The docroot folder is level 0.
* Files are invalidated by touching the .stat file. The .stat file's last modification date is compared to the last modification date of a cached document. The document is re-fetched if the .stat file is newer.
* When a file located at a certain level is invalidated then **all**.stat files from the docroot **to** the level of the invalidated file or the configured statsfilevel (whichever is smaller) will be touched.
  + For example: if you set the statfileslevel property to 6 and a file is invalidated at level 5 then every .stat file from docroot to 5 will be touched. Continuing with this example, if a file is invalidated at level 7 then every . stat file from docroot to 6 will be touched (since /statfileslevel = "6").

Only resources\*\* along the path\*\* to the invalidated file are affected. Consider the following example: a website uses the structure /content/myWebsite/xx/. If you set statfileslevel as 3, a .statfile is created as follows:

* docroot
* /content
* /content/myWebsite
* /content/myWebsite/\*xx\*

When a file in /content/myWebsite/xx is invalidated then every .stat file from docroot down to /content/myWebsite/xxis touched. This would be the case only for /content/myWebsite/xx and not for example /content/myWebsite/yy or /content/anotherWebSite.

Invalidation can be prevented by sending an additional Header CQ-Action-Scope:ResourceOnly. This can be used to flush particular resources without invalidating other parts of the cache. See [this page](https://adobe-consulting-services.github.io/acs-aem-commons/features/dispatcher-flush-rules/index.html) and [Manually Invalidating the Dispatcher Cache](https://helpx.adobe.com/experience-manager/dispatcher/using/page-invalidate.html) for additional details.

If you specify a value for the /statfileslevel property, the /statfile property is ignored.

### Automatically Invalidating Cached Files

The /invalidate property defines the documents that are automatically invalidated when content is updated.

With automatic invalidation, Dispatcher does not delete cached files after a content update, but checks their validity when they are next requested. Documents in the cache that are not auto-invalidated will remain in the cache until a content update explicitly deletes them.

Automatic invalidation is typically used for HTML pages. HTML pages often contain links to other pages, making it difficult to determine whether a content update affects a page. To make sure that all relevant pages are invalidated when content is updated, automatically invalidate all HTML pages. The following configuration invalidates all HTML pages:

/invalidate

{

/0000 { /glob "\*" /type "deny" }

/0001 { /glob "\*.html" /type "allow" }

}

For information about glob properties, see [Designing Patterns for glob Properties](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#designing-patterns-for-glob-properties).

This configuration causes the following activity when /content/geometrixx/en is activated:

* All the files with pattern en.\* are removed from the /content/geometrixx/ folder.
* The /content/geometrixx/en/\_jcr\_content folder is removed.
* All the other files that match the /invalidate configuration are not immediately deleted. These files are deleted when the next request occurs. In our example /content/geometrixx.html is not deleted, it will be deleted when /content/geometrixx.html is requested.

If you offer automatically generated PDF and ZIP files for download, you might have to automatically invalidate these as well. A configuration example this looks as follows:

/invalidate

{

/0000 { /glob "\*" /type "deny" }

/0001 { /glob "\*.html" /type "allow" }

/0002 { /glob "\*.zip" /type "allow" }

/0003 { /glob "\*.pdf" /type "allow" }

}

The AEM integration with Adobe Analytics delivers configuration data in an analytics.sitecatalyst.js file in your website. The example dispatcher.any file that is provided with Dispatcher includes the following invalidation rule for this file:

{

/glob "\*/analytics.sitecatalyst.js" /type "allow"

}

### Using custom invalidation scripts

The /invalidateHandler property allows you to define a script which is called for each invalidation request received by Dispatcher.

It is called with the following arguments:

* HandleThe content path that is invalidated
* ActionThe replication Action (e.g. Activate, Deactivate)
* Action ScopeThe replication Action's Scope (empty, unless a header of CQ-Action-Scope: ResourceOnly is sent, see [Invalidating Cached Pages from AEM](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/page-invalidate.html) for details)

This can be used to cover a number of different use cases, such as invalidating other application specific caches, or to handle cases where the externalized URL of a page and its place in the docroot does not match the content path.

Below example script logs each invalidate request to a file.

/invalidateHandler "/opt/dispatcher/scripts/invalidate.sh"

#### sample invalidation handler script

#!/bin/bash

printf "%-15s: %s %s" $1 $2 $3>> /opt/dispatcher/logs/invalidate.log

### Limiting the Clients That Can Flush the Cache

The /allowedClients property defines specific clients that are allowed to flush the cache. The globbing patterns are matched against the IP.

The following example:

1. denies access to any client
2. explicitly allows access to the localhost

/allowedClients

{

/0001 { /glob "\*.\*.\*.\*" /type "deny" }

/0002 { /glob "127.0.0.1" /type "allow" }

}

For information about glob properties, see [Designing Patterns for glob Properties](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#designing-patterns-for-glob-properties).

It is recommended that you define the /allowedClients.

If this is not done, any client can issue a call to clear the cache; if this is done repeatedly it can severely impact the site performance.

### Ignoring URL Parameters

The ignoreUrlParams section defines which URL parameters are ignored when determining whether a page is cached or delivered from cache:

* When a request URL contains parameters that are all ignored, the page is cached.
* When a request URL contains one or more parameters that are not ignored, the page is not cached.

When a parameter is ignored for a page, the page is cached the first time that the page is requested. Subsequent requests for the page are served the cached page, regardless of the value of the parameter in the request.

To specify which parameters are ignored, add glob rules to the ignoreUrlParams property:

* To ignore a parameter, create a glob property that allows the parameter.
* To prevent the page to be cached, create a glob property that denies the parameter.

The following example causes Dispatcher to ignores the "q" parameter, so that request URLs that include the q parameter are cached:

/ignoreUrlParams

{

/0001 { /glob "\*" /type "deny" }

/0002 { /glob "q" /type "allow" }

}

Using the example ignoreUrlParams value, the following HTTP request causes the page to be cached because the q parameter is ignored:

GET /mypage.html?q=5

Using the example ignoreUrlParams value, the following HTTP request causes the page to **not** be cached because the p parameter is not ignored:

GET /mypage.html?q=5&p=4

For information about glob properties, see [Designing Patterns for glob Properties](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#designing-patterns-for-glob-properties).

### Caching HTTP Response Headers

This feature is avaiable with version **4.1.11** of the Dispatcher.

The /headers property allows you to define the HTTP header types that are going to be cached by the Dispatcher. On the first request to an uncached resource, all headers matching one of the configured values (see the configuration sample below) are stored in a separate file, next to the cache file. On subsequent requests to the cached resource, the stored headers are added to the response.

Presented below is a sample from the default configuration:

/cache {

...

/headers {

"Cache-Control"

"Content-Disposition"

"Content-Type"

"Expires"

"Last-Modified"

"X-Content-Type-Options"

"Last-Modified"

}

}

Also, be aware that file globbing characters are not allowed. For more details, see [Designing Patterns for glob Properties](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#designing-patterns-for-glob-properties).

If you need Dispatcher to store and deliver ETag response headers from AEM, do the following:

* Add the header name in the /cache/headerssection.
* Add the following [Apache directive](https://httpd.apache.org/docs/2.4/mod/core.html#fileetag) in the Dispatcher related section:

FileETag none

### Dispatcher Cache File Permissions

The mode property specifies what file permissions are applied to new directories and files in the cache. This setting is restricted by the umask of the calling process. It is an octal number constructed from the sum of one or more of the following values:

* 0400 Allow read by owner.
* 0200 Allow write by owner.
* 0100 Allow the owner to search in directories.
* 0040 Allow read by group members.
* 0020 Allow write by group members.
* 0010 Allow group members to search in the directory.
* 0004 Allow read by others.
* 0002 Allow write by others.
* 0001 Allow others to search in the directory.

The default value is 0755 which allows the owner to read, write or search and the group and others to read or search.

### Throttling .stat file touching

With the default /invalidate property, every activation effectively invalidates all .html files (when their path matches the /invalidate section). On a website with considerable traffic, multiple, subsequent activations will increase the cpu load on the backend. In such a scenario, it would be desirable to "throttle" .stat file touching to keep the website responsive. You can do this by using the /gracePeriod property.

The /gracePeriod property defines the number of seconds a stale, auto-invalidated resource may still be served from the cache after the last occuring activation. The property can be used in a setup where a batch of activations would otherwise repeatedly invalidate the entire cache. The recommended value is 2 seconds.

For additional details, also read the /invalidate and /statfileslevelsections above.

## Configuring Time Based Cache Invalidation - /enableTTL

If set, the enableTTL property will evaluate the response headers from the backend, and if they contain a Cache-Control max-age or Expires date, an auxiliary, empty file next to the cache file is created, with modification time equal to the expiry date. When the cached file is requested past the modification time it is automatically re-requested from the backend.

You can enable the feature by adding this line to the dispatcher.any file:

/enableTTL "1"

This feature is avaiable with version **4.1.11** of the Dispatcher.

## Configuring Load Balancing - /statistics

The /statistics section defines categories of files for which Dispatcher scores the responsiveness of each render. Dispatcher uses the scores to determine which render to send a request.

Each category that you create defines a glob pattern. Dispatcher compares the URI of the requested content to these patterns to determine the category of the requested content:

* The order of the categories determines the order in which they are compared to the URI.
* The first category pattern that matches the URI is the category of the file. No more category patterns are evaluated.

Dispatcher supports a maximum of 8 statistics categories. If you define more than 8 categories, only the first 8 are used.

**Render Selection**

Each time Dispatcher requires a rendered page, it uses the following algorithm to select the render:

1. If the request contains the render name in a renderid cookie, Dispatcher uses that render.
2. If the request includes no renderid cookie, Dispatcher compares the render statistics:
   1. Dispatcher determines the cateogry of the request URI.
   2. Dispatcher determines which render has the lowest response score for that category, and selects that render.
3. If no render is selected yet, use the first render in the list.

The score for a render's category is based on previous response times, as well as previous failed and successful connections that Dispatcher attempts. For each attempt, the score for the category of the requested URI is updated.

If you do not use load balancing, you can omit this section.

### Defining Statistics Categories

Define a category for each type of document for which you want to keep statistics for render selection. The /statistics section contains a /categories section. To define a category, add a line below the /categories section that has the following format:

/name { /glob "pattern"}

The category name must be unique to the farm. The pattern is described in the [Designing Patterns for glob Properties](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#designing-patterns-for-glob-properties) section.

To determine the category of a URI, Dispatcher compares the URI with each category pattern until a match is found. Dispatcher begins with the first category in the list and cointinues in order. Therefore, place categories with more specific patterns first.

For example, Dispatcher the default dispatcher.any file defines an HTML category and an others category. The HTML category is more specific and so it appears first:

/statistics

{

/categories

{

/html { /glob "\*.html" }

/others { /glob "\*" }

}

}

The following example also includes a category for search pages:

/statistics

{

/categories

{

/search { /glob "\*search.html" }

/html { /glob "\*.html" }

/others { /glob "\*" }

}

}

### Reflecting Server Unavailability in Dispatcher Statistics

The /unavailablePenalty property sets the time (in tenths of a second) that is applied to the render statistics when a connection to the render fails. Dispatcher adds the time to the statistics category that matches the requested URI.

For example, the penalty is applied when the TCP/IP connection to the designated hostname/port cannot be established, either because AEM is not running (and not listening) or because of a network-related problem.

The /unavailablePenalty property is a direct child of the /farm section (a sibling of the /statistics section).

If no /unavailablePenalty property exists, a value of "1" is used.

/unavailablePenalty "1"

## Identifying a Sticky Connection Folder - /stickyConnectionsFor

The /stickyConnectionsFor property defines one folder that contains sticky documents; this will be accessed using the URL. Dispatcher sends all requests, from a single user, that are in this folder to the same render instance. Sticky connections ensure that session data is present and consistent for all documents. This mechanism uses the renderid cookie.

The following example defines a sticky connection to the /products folder:

/stickyConnectionsFor "/products"

When a page is composed of content from several content nodes, include the /paths property that lists the paths to the content. For example, a page contains content from /content/image, /content/video, and /var/files/pdfs. The following configuration enables sticky connections for all content on the page:

/stickyConnections {

/paths {

"/content/image"

"/content/video"

"/var/files/pdfs"

}

}

### httpOnly

When sticky connections are enabled, the dispatcher module sets the renderid cookie. This cookie doesn't have the httponly flag, which should be added in order to enhance security. You can do this by setting the httpOnly property in the /stickyConnections node of a dispatcher.anyconfiguration file. The property's value (either 0 or 1) defines whether the renderid cookie has the HttpOnly attribute appended. The default value is 0, which means the attribute will not be added.

For additional information about the httponly flag, read [this page](https://www.owasp.org/index.php/HttpOnly).

### secure

When sticky connections are enabled, the dispatcher module sets the renderid cookie. This cookie doesn't have the **secure** flag, which should be added in order to enhance security. You can do this by setting the secure property in the /stickyConnections node of a dispatcher.any configuration file. The property's value (either 0 or 1) defines whether the renderid cookie has the secure attribute appended. The default value is 0, which means the attribute will be added if\* \*the incoming request is secure. If the value is set to 1 then the secure flag will be added regardless of whether the incoming request is secure or not.

## Handling Render Connection Errors

Configure Dispatcher behavior when the render server returns a 500 error, or is unavailable.

### Specifying a Health Check Page

Use the /health\_check property to specify a URL that is checked when a 500 status code occurs. If this page also returns a 500 status code the instance is considered to be unavailable and a configurable time penalty ( /unavailablePenalty) is applied to the render before retrying.

/health\_check

{

# Page gets contacted when an instance returns a 500

/url "/health\_check.html"

}

### Specifying the Page Retry Delay

The / retryDelay property sets the time (in seconds) that Dispatcher waits between rounds of connection attempts with the farm renders. For each round, the maximum number of times Dispatcher attempts a connection to a render is the number of renders in the farm.

Dispatcher uses a value of "1" if /retryDelay is not explicitly defined. The default value is appropriate in most cases.

/retryDelay "1"

### Configuring the Number of Retries

The /numberOfRetries property sets the maximum number of rounds of connection attempts that Dispatcher performs with the renders. If Dispatcher cannot successfully connect to a render after this number of retries, Dispatcher returns a failed response.

For each round, the maximum number of times Dispatcher attempts a connection to a render is the number of renders in the farm. Therefore, the maximum number of times that Dispatcher attempts a connection is ( /numberOfRetries) x (the number of renders).

If the value is not explicitly defined, the default value is **5**.

/numberOfRetries "5"

### Using the Failover Mechanism

Enable the failover mechanism on your Dispatcher farm to resend requests to different renders when the original request fails. When failover is enabled, Dispatcher has the following behavior:

* When a request to a render returns HTTP status 503 (UNAVAILABLE), Dispatcher sends the request to a different render.
* When a request to a render returns HTTP status 50x (other than 503), Dispatcher sends a request for the page that is configured for the health\_check property.
  + If the health check returns 500 (INTERNAL\_SERVER\_ERROR), Dispatcher sends the original request to a different render.
  + If the healtch check returns HTTP status 200, Dispatcher returns the initial HTTP 500 error to the client.

To enable failover, add the following line to the farm (or website):

/failover "1"

To retry HTTP requests that contain a body, Dispatcher sends a Expect: 100-continue request header to the render before spooling the actual contents. CQ 5.5 with CQSE then immediately answers with either 100 (CONTINUE) or an error code. Other servlet containers should support this as well.

## Ignoring Interruption Errors - /ignoreEINTR

This option is not usually needed. You only need to use this when you see the following log messages:

Error while reading response: Interrupted system call

Any file system oriented system call can be interrupted EINTR if the object of the system call is located on a remote system accessed via NFS. Whether these system calls can time out or be interrupted is based on how the underlying file system was mounted on the local machine.

Use the /ignoreEINTR parameter if your instance has such a configuration and the log contains the following message:

Error while reading response: Interrupted system call

Internally, Dispatcher reads the response from the remote server (i.e. AEM) using a loop that can be represented as:

while (response not finished) { read more data }

Such messages can be generated when the EINTR occurs in the " read more data" section and are caused by the reception of a signal before any data was received.

To ignore such interrupts you can add the following parameter to dispatcher.any (before /farms):

/ignoreEINTR "1"

Setting /ignoreEINTR to "1" causes Dispatcher to continue to attempt to read data until the complete response is read. The default value is 0 and deactivates the option.

## Designing Patterns for glob Properties

Several sections in the Dispatcher configuration file use glob properties as selection criteria for client requests. The values of glob properties are patterns that Dispatcher compares to an aspect of the request, such as the path of the requested resource, or the IP address of the client. For example, the items in the /filter section use glob patterns to identify the paths of the pages that Dispatcher acts on or rejects.

The glob values can include wildcard characters and alphanumeric characters to define the pattern.

| **Wildcard character** | **Description** | **Examples** |
| --- | --- | --- |
| \* | Matches zero or more contiguous instances of any character in the string. The final character of the match is determined by either of the following situations:  A character in the string matches the next character in the pattern, and the pattern character has the following characteristics:   * Not a \* * Not a ? * A literal character (including a space) or a character class. * The end of the pattern is reached.   Inside a character class, the character is interpreted literally. | \*/geo\* Matches any page below the /content/geometrixx node and the /content/geometrixx-outdoors node. The following HTTP requests match the glob pattern:   * "GET /content/geometrixx/en.html" * "GET /content/geometrixx-outdoors/en.html"   \*outdoors/\*  Matches any page below the /content/geometrixx-outdoors node. For example, the following HTTP request matches the glob pattern:   * "GET /content/geometrixx-outdoors/en.html" |
| ? | Matches any single character. Use outside character classes. Inside a character class, this character is interpreted literally. | \*outdoors/??/\*  Matches the pages for any language in the geometrixx-outdoors site. For example, the following HTTP request matches the glob pattern:   * "GET /content/geometrixx-outdoors/en/men.html"   The following request does not match the glob pattern:   * "GET /content/geometrixx-outdoors/en.html" |
| [ and ] | Demarks the beginning and end of a character class. Character classes can include one or more character ranges and single characters.  A match occurs if the target character matches any of the characters in the character class, or within a defined range.  If the closing bracket is not included, the pattern produces no matches. | \*[o]men.html\*  Matches the following HTTP request:   * "GET /content/geometrixx-outdoors/en/women.html"   Does not match the following HTTP request:   * "GET /content/geometrixx-outdoors/en/men.html"   \*[o/]men.html\*  Matches the following HTTP requests:   * "GET /content/geometrixx-outdoors/en/women.html" * "GET /content/geometrixx-outdoors/en/men.html" |
| - | Denotes a range of characters. For use in character classes. Outside of a character class, this character is interpreted literally. | \*[m-p]men.html\* Matches the following HTTP request:   * "GET /content/geometrixx-outdoors/en/women.html"   Does not match the following HTTP request:   * "GET /content/geometrixx-outdoors/en/men.html" |
| ! | Negates the character or character class that follows. Use only for negating characters and character ranges inside character classes. Equivalent to the ^ wildcard.  Outside of a character class, this character is interpreted literally. | \*[!o]men.html\*  Matches the following HTTP request:   * "GET /content/geometrixx-outdoors/en/men.html"   Does not match the following HTTP request:   * "GET /content/geometrixx-outdoors/en/women.html"   \*[!o!/]men.html\*  Does not match the following HTTP request:   * "GET /content/geometrixx-outdoors/en/women.html" or "GET /content/geometrixx-outdoors/en/men. html" |
| ^ | Negates the character or character range that follows. Use for negating only characters and character ranges inside character classes. Equivalent to the ! wildcard character.  Outside of a character class, this character is interpreted literally. | The examples for the ! wildcard character apply, substituting the ! characters in the example patterns with ^ characters. |

## Logging

In the web server configuration, you can set:

* The location of the Dispatcher log file.
* The log level.

Refer to the web server documentation and the readme file of your Dispatcher instance for more information.

**Apache Rotated / Piped Logs**

If using an **Apache** web server you can use the standard functionality for rotated and/or piped logs. For example, using piped logs:

DispatcherLog "| /usr/apache/bin/rotatelogs logs/dispatcher.log%Y%m%d 604800"

This will automatically rotate:

* the dispatcher log file; with a timestamp in the extension (logs/dispatcher.log%Y%m%d).
* on a weekly basis (60 x 60 x 24 x 7 = 604800 seconds).

Please see the Apache web server documentation on Log Rotation and Piped Logs; for example [Apache 2.2](https://httpd.apache.org/docs/2.2/logs.html).

Upon installation the default log level is high (i.e. level 3 = Debug), so that the Dispatcher logs all errors and warnings. This is very useful in the initial stages.

However, this requires additional resources, so when the Dispatcher is working smoothly *according to your requirements*, you can(should) lower the log level.

### Trace Logging

Amongst other enhancements for the Dispatcher, version 4.2.0 also introduces Trace Logging.

This is a higher level than Debug logging, showing additional information in the logs. It adds logging for:

* The values of the forwarded headers;
* The rule that is being applied for a certain action.

You can enable Trace Logging by setting the log level to 4 in your web server.

Below is an example of logs with tracing enabled:

[Thu Mar 03 16:05:38 2016] [T] [17183] request.headers[Host] = "localhost:8443"

[Thu Mar 03 16:05:38 2016] [T] [17183] request.headers[User-Agent] = "curl/7.43.0"

[Thu Mar 03 16:05:38 2016] [T] [17183] request.headers[Accept] = "\*/\*"

[Thu Mar 03 16:05:38 2016] [T] [17183] request.headers[X-Forwarded-SSL-Client-Cert] = "(null)"

[Thu Mar 03 16:05:38 2016] [T] [17183] request.headers[Via] = "1.1 localhost:8443 (dispatcher)"

[Thu Mar 03 16:05:38 2016] [T] [17183] request.headers[X-Forwarded-For] = "::1"

[Thu Mar 03 16:05:38 2016] [T] [17183] request.headers[X-Forwarded-SSL] = "on"

[Thu Mar 03 16:05:38 2016] [T] [17183] request.headers[X-Forwarded-SSL-Cipher] = "DHE-RSA-AES256-SHA"

[Thu Mar 03 16:05:38 2016] [T] [17183] request.headers[X-Forwarded-SSL-Session-ID] = "ba931f5e4925c2dde572d766fdd436375e15a0fd24577b91f4a4d51232a934ae"

[Thu Mar 03 16:05:38 2016] [T] [17183] request.headers[X-Forwarded-Port] = "8443"

[Thu Mar 03 16:05:38 2016] [T] [17183] request.headers[Server-Agent] = "Communique-Dispatcher"

And an event logged when a file that matches a blocking rule is requested:

[Thu Mar 03 14:42:45 2016] [T] [11831] 'GET /content.infinity.json HTTP/1.1' was blocked because of /0082

## Confirming Basic Operation

To confirm basic operation and interaction of the web server, Dispatcher and AEM instance you can use the following steps:

1. Set the loglevel to 3.
2. Start the web server; this also starts the Dispatcher.
3. Start the AEM instance.
4. Check the log and error files for your web server and the Dispatcher.Depending on your web server you should see messages such as:[Thu May 30 05:16:36 2002] [notice] Apache/2.0.50 (Unix) configuredand:[Fri Jan 19 17:22:16 2001] [I] [19096] Dispatcher initialized (build XXXX)
5. Surf the website via the web server. Confirm that content is being shown as required.For example, on a local installation where AEM runs on port 4502 and the web server on 80 access the Websites console using both: https://localhost:4502/libs/wcm/core/content/siteadmin.html https://localhost:80/libs/wcm/core/content/siteadmin.html The results should be identical. Confirm access to other pages with the same mechanism.
6. Check that the cache directory is being filled.
7. Activate a page to check that the cache is being flushed correctly.
8. If everything is operating correctly you can reduce the loglevel to 0.

## Using Multiple Dispatchers

In complex setups, you may use multiple Dispatchers. For example, you may use:

* one Dispatcher to publish a website on the Intranet
* a second Dispatcher, under a different address and with different security settings, to publish the same content on the Internet.

In such a case, make sure that each request goes through only one Dispatcher. A Dispatcher does not handle requests that come from another Dispatcher. Therefore, make sure that both Dispatchers access the AEM website directly.

## Debugging

When adding the header X-Dispatcher-Info to a request, Dispatcher answers whether the target was cached, returned from cached or not cacheable at all. The response header X-Cache-Info contains this information in a readable form. You can use these response headers to debug issues involving responses cached by the Dispatcher.

This functionality is not enabled by default, so in order for the response header X-Cache-Info to be included, the farm must contain the following entry:

/info "1"

For example,

/farm

{

/mywebsite

{

# Include X-Cache-Info response header if X-Dispatcher-Info is in request header

/info "1"

}

}

Also, the X-Dispatcher-Info header does not need a value, but if you use curl for testing you must supply a value in order to send the header, such as:

curl -v -H "X-Dispatcher-Info: true" https://localhost/content/we-retail/us/en.html

Below is a list containing the response headers that X-Dispatcher-Info will return:

* **cached**The target file is contained in the cache and the dispatcher has determined that it is valid to deliver it.
* **caching**The target file isn't contained in the cache and the dispatcher has determined that it is valid to cache the output and deliver it.
* **caching: stat file is more recent** The target file is contained in the cache, however, it is invalidated by a more recent stat file. The dispatcher will delete the target file, recreate it from the output and deliver it.
* **not cacheable: no document root** The farm's configuration doesn't contain a document root (configuration element cache.docroot).
* **not cacheable: cache file path too long**The target file - the concatenation of document root and URL file - exceeds the longest possible file name on the system.
* **not cacheable: temporary file path too long**The temporary file name template exceeds the longest possible file name on the system. The dispatcher creates a temporary file first, before actually creating or overwriting the cached file. The temporary file name is the target file name with the characters \_YYYYXXXXXX appended to it, where the Y and X will be replaced to create a unique name.
* **not cacheable: request URL has no extension**The request URL has no extension, or there is a path following the file extension, for example: /test.html/a/path.
* **not cacheable: request wasn't a GET or HEAD** The HTTP method is neither a GET nor a HEAD. The dispatcher assumes that the output will contain dynamic data that shouldn't be cached.
* **not cacheable: request contained a query string**The request contained a query string. The dispatcher assumes that the output depends on the query string given and therefore doesn't cache.
* **not cacheable: session manager didn't authenticate**The farm's cache is governed by a session manager (the configuration contains a sessionmanagement node) and the request didn't contain the appropriate authentication information.
* **not cacheable: request contains authorization**The farm is not allowed to cache output ( allowAuthorized 0) and the request contains authentication information.
* **not cacheable: target is a directory**The target file is a directory. This might point to some conceptual mistake, where a URL and some sub-URL both contain cacheable output, for example if a request to /test.html/a/file.ext comes first and contains cacheable output, the dispatcher will not be able to cache the output of a subsequent request to /test.html.
* **not cacheable: request URL has a trailing slash**The request URL has a trailing slash.
* **not cacheable: request URL not in cache rules**The farm's cache rules explicitly deny caching the output of some request URL.
* **not cacheable: authorization checker denied access**The farm's authorization checker denied access to the cached file.
* **not cacheable: session not valid** The farm's cache is governed by a session manager (configuration contains a sessionmanagement node) and the user's session is not or no longer valid.
* **not cacheable: response contains no\_cache**The remote server returned a Dispatcher: no\_cache header, forbidding the dispatcher to cache the output.
* **not cacheable: response content length is zero** The content length of the response is zero; the dispatcher will not create a zero-length file.

**Invalidating Cached Pages From AEM**

When using Dispatcher with AEM, the interaction must be configured to ensure effective cache management. Depending on your environment, the configuration can also increase performance.

Setting up AEM User Accounts

The default admin user account is used to authenticate the replication agents that are installed by default. You should create a dedicated user account for use with replication agents. [security-checklist.html#VerificationSteps](https://helpx.adobe.com/experience-manager/6-3/sites/administering/using/security-checklist.html#VerificationSteps)

For more information see the [Configure Replication and Transport Users](https://helpx.adobe.com/experience-manager/6-3/sites/administering/using/security-checklist.html#VerificationSteps) section of the AEM Security Checklist.

Invalidating Dispatcher Cache from the Authoring Environment

A replication agent on the AEM author instance sends a cache invalidation request to Dispatcher when a page is published. The request causes Dispatcher to eventually refresh the file in the cache as new content is published.

Use the following procedure to configure a replication agent on the AEM author instance for invalidating the Dispatcher cache upon page activation:

1. Open the AEM Tools console. (https://localhost:4502/miscadmin#/etc)
2. Open the required replication agent below Tools/replication/Agents on author. You can use the Dispatcher Flush agent that is installed by default.
3. Click Edit, and in the Settings tab ensure that **Enabled** is selected.
4. (optional) To enable alias or vanity path invalidation requests select the **Alias update** option.
5. On the Transport tab, enter the URI needed to access Dispatcher.If you are using the standard Dispatcher Flush agent you will probably need to update the hostname and port; for example, https://<*dispatcherHost*>:<*portApache*>/dispatcher/invalidate.cache

**Note:** For Dispatcher Flush agents, the URI property is used only if you use path-based virtualhost entries to differentiate between farms. You use this field to target the farm to invalidate. For example, farm #1 has a virtual host of www.mysite.com/path1/\* and farm #2 has a virtual host of www.mysite.com/path2/\*. You can use a URL of /path1/invalidate.cache to target the first farm and /path2/invalidate.cache to target the second farm. For more information, see [Using Dispatcher with Multiple Domains](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-domains.html).

1. Configure other parameters as required.
2. Click OK to activate the agent.

Alternatively, you can also access and configure the Dispatcher Flush agent from the [AEM Touch UI](https://helpx.adobe.com/experience-manager/6-2/sites/deploying/using/replication.html#ConfiguringaDispatcherFlushagent).

For additional details on how to enable access to vanity URLs, see [Enabling Access To Vanity URLs](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#enabling-access-to-vanity-urls-vanity-urls).

The agent for flushing dispatcher cache does not have to have a user name and password, but if configured they will be sent with basic authentication.

There are two potential issues with this approach:

* The Dispatcher must be reachable from the authoring instance. If your network (e.g. the firewall) is configured such that access between the two is restricted, this may not be the case.
* Publication and cache invalidation take place at the same time. Depending on the timing, a user may request a page just after it was removed from the cache, and just before the new page is published. AEM now returns the old page, and the Dispatcher caches it again. This is more of an issue for large sites.

Invalidating Dispatcher Cache from a Publishing Instance

Under certain circumstances performance gains can be made by transferring cache management from the authoring environment to a publishing instance. It will then be the publishing environment (not the AEM authoring environment) that sends a cache invalidation request to Dispatcher when a published page is received.

Such circumstances include:

* Preventing possible timing conflicts between Dispatcher and the publish instance (see [Invalidating Dispatcher cache from the Authoring Environment](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/page-invalidate.html#invalidating-dispatcher-cache-from-the-authoring-environment)).
* The system includes several publishing instances that reside on high performance servers, and only one authoring instance.

The decision to use this method should be made by an experienced AEM administrator.

The dispatcher flush is controlled by a replication agent operating on the publish instance. However, the configuration is made on the authoring environment and then transferred by activating the agent:

1. Open the AEM Tools console.
2. Open the required replication agent below Tools/replication/Agents on publish. You can use the Dispatcher Flush agent that is installed by default.
3. Click Edit, and in the Settings tab ensure that **Enabled** is selected.
4. (optional) To enable alias or vanity path invalidation requests select the **Alias update** option.
5. On the Transport tab, enter the URI needed to access Dispatcher.If you are using the standard Dispatcher Flush agent you will probably need to update the hostname and port; for example, http://<dispatcherHost>:<portApache>/dispatcher/invalidate.cache

**Note:** For Dispatcher Flush agents, the URI property is used only if you use path-based virtualhost entries to differentiate between farms. You use this field to target the farm to invalidate. For example, farm #1 has a virtual host of www.mysite.com/path1/\* and farm #2 has a virtual host of www.mysite.com/path2/\*. You can use a URL of /path1/invalidate.cache to target the first farm and /path2/invalidate.cache to target the second farm. For more information, see [Using Dispatcher with Multiple Domains](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-domains.html).

1. Configure other parameters as required.
2. Repeat for every publish instance affected.

After configuring, when you activate a page from author to publish, this agent initiates a standard replication. The log includes messages indicating requests coming from your publish server, similar to the following example:

1. <publishserver> 13:29:47 127.0.0.1 POST /dispatcher/invalidate.cache 200

Manually Invalidating the Dispatcher Cache

To invalidate (or flush) the Dispatcher cache without activating a page, you can issue an HTTP request to the dispatcher. For example, you can create an AEM application that enables administrators or other applications to flush the cache.

The HTTP request causes Dispatcher to delete specific files from the cache. Optionally, the Dispatcher then refreshes the cache with a new copy.

Delete cached files

Issue an HTTP request that causes Dispatcher to delete files from the cache. Dispatcher caches the files again only when it receives a client request for the page. Deleting cached files ins this manner is appropriate for web sites that are not likely to receive simultaneous requests for the same page.

The HTTP request has the following form:

POST /dispatcher/invalidate.cache HTTP/1.1

CQ-Action: Activate

CQ-Handle: path-pattern

Content-Length: 0

Dispatcher flushes (deletes) the cached files and folders that have names that match the value of the CQ-Handler header. For example, a CQ-Handle of /content/geomtrixx-outdoors/en matches the following items:

* All files (of any file extension) named en in the geometrixx-outdoors directory
* Any directory named " \_jcr\_content" below the en directory (which, if it exists, contains cached renderings of sub-nodes of the page)

All other files in the dispatcher cache (or up to a particular level, depending on the /statfileslevel setting) are invalidated by touching the .stat file. This file's last modification date is compared to the last modification date of a cached document and the document is re-fetched if the .stat file is newer. See [Invalidating Files by Folder Level](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#main-pars_title_26) for details.

Invalidation (i.e. touching of .stat files) can be prevented by sending an additional Header CQ-Action-Scope: ResourceOnly. This can be used to flush particular resources without invalidating other parts of the cache, like JSON data that is dynamically created and requires regular flushing independent of the cache (e.g. representing data that is obtained from a third-party system to display news, stock tickers, etc.).

Delete and recache files

Issue an HTTP request that causes Dispatcher to delete cached files, and immediately retrieve and recache the file. Delete and immediately re-cache files when web sites are likely to receive simultaneous client requests for the same page. Immediate recaching ensures that Dispatcher retrieves and caches the page only once, instead of once for each of the simultaneous client requests.

**Note:** Deleting and recaching files should be performed on the publish instance only. When performed from the author instance, race conditions occur when attempts to recache resources occur before they have been published.

The HTTP request has the following form:

POST /dispatcher/invalidate.cache HTTP/1.1

CQ-Action: Activate

`Content-Type: text/plain

CQ-Handle: path-pattern

Content-Length: numchars in bodypage\_path0

page\_path1

...

page\_pathn

The page paths to immediately recache are listed on separate lines in the message body. The value of CQ-Handle is the path of a page that invalidates the pages to recache. (See the /statfileslevel parameter of the [Cache](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#main-pars_146_44_0010) configuration item.) The following example HTTP request message deletes and recaches the /content/geometrixx-outdoors/en.html page:

POST /dispatcher/invalidate.cache HTTP/1.1

CQ-Action: Activate

Content-Type: text/plain

CQ-Handle: /content/geometrixx-outdoors/en/men.html

Content-Length: 36

/content/geometrixx-outdoors/en.html

Example flush servlet

The following code implements a servlet that sends an invalidate request to Dispatcher. The servlet receives a request message that contains handle and page parameters. These parameters provide the value of the CQ-Handle header and the path of the page to recache, respectively. The servlet uses the values to construct the HTTP request for Dispatcher.

When the servlet is deployed to the publish instance, the following URL causes Dispatcher to delete the /content/geometrixx-outdoors/en.html page and then cache a new copy.

10.36.79.223:4503/bin/flushcache/html?page=/content/geometrixx-outdoors/en.html&handle=/content/geometrixx-outdoors/en/men.html

This example servlet is not secure and only demonstrates the use of the HTTP Post request message. Your solution should secure access to the servlet.

package com.adobe.example;

import org.apache.felix.scr.annotations.Component;

import org.apache.felix.scr.annotations.Service;

import org.apache.felix.scr.annotations.Property;

import org.apache.sling.api.SlingHttpServletRequest;

import org.apache.sling.api.SlingHttpServletResponse;

import org.apache.sling.api.servlets.SlingSafeMethodsServlet;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.apache.commons.httpclient.\*;

import org.apache.commons.httpclient.methods.PostMethod;

import org.apache.commons.httpclient.methods.StringRequestEntity;

@Component(metatype=true)

@Service

public class Flushcache extends SlingSafeMethodsServlet {

@Property(value="/bin/flushcache")

static final String SERVLET\_PATH="sling.servlet.paths";

private Logger logger = LoggerFactory.getLogger(this.getClass());

public void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response) {

try{

//retrieve the request parameters

String handle = request.getParameter("handle");

String page = request.getParameter("page");

//hard-coding connection properties is a bad practice, but is done here to simplify the example

String server = "localhost";

String uri = "/dispatcher/invalidate.cache";

HttpClient client = new HttpClient();

PostMethod post = new PostMethod("https://"+host+uri);

post.setRequestHeader("CQ-Action", "Activate");

post.setRequestHeader("CQ-Handle",handle);

StringRequestEntity body = new StringRequestEntity(page,null,null);

post.setRequestEntity(body);

post.setRequestHeader("Content-length", String.valueOf(body.getContentLength()));

client.executeMethod(post);

post.releaseConnection();

//log the results

logger.info("result: " + post.getResponseBodyAsString());

}

}catch(Exception e){

logger.error("Flushcache servlet exception: " + e.getMessage());

}

}

}

**Caching Secured Content**

Permission-sensitive caching enables you to cache secured pages. Dispatcher checks user's access permissions for a page before delivering the cached page.

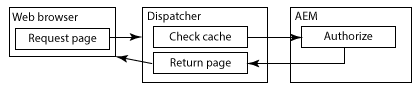
Dispatcher includes the AuthChecker module that implements permission-sensitive caching. When the module is activated, the render calls an AEM servlet to perform user authentication and authorization for the requested content. The servlet response determines whether the content is delivered to the web browser.

Because the methods of authentication and authorization are specific to the AEM deployment, you are required to create the servlet.

Use deny filters to enforce blanket security restrictions. Use permission-sensitive caching for pages that are configured to permit access to a subset of users or groups.

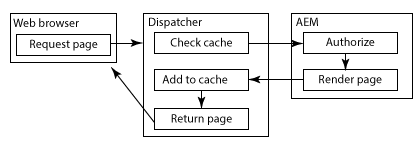
The following diagrams illustrate the order of events that occur when a web browser requests a page for which permission-sensitive caching is used.

Page is cached and user is authorized



1. Dispatcher determines that the requested content is cached and valid.
2. Dispatcher sends a request message to the render. The HEAD section includes all of the header lines from the browser request.
3. The render calls the authorizer to perform the security check and responds to Dispatcher. The response message includes an HTTP status code of 200 to indicate that the user is authorized.
4. Dispatcher sends a response message to the browser that consists of the header lines from the render response and the cached content in the body.

Page is not cached and user is authorized



1. Dispatcher determines that the content is not cached or requires updating.
2. Dispatcher forwards the original request to the render.
3. The render calls the authorizer servlet to perform a security check. When the user is authorized, the render includes the rendered page in the body of the response message.
4. Dispatcher forwards the response to the browser. Dispatcher adds the body of the render's response message to the cache.

User is not authorized

https://docs.adobe.com/content/dam/help/experience-manager-dispatcher.en/help/using/assets/chlimage_1-2.png

1. Dispatcher checks the cache.
2. Dispatcher sends a request message to the render that includes all header lines from the browser's request.
3. The render calls the authorizer servlet to perform a security check which fails, and the render forwards the original request to Dispatcher.

Implementing permission-sensitive caching

To implement permission-sensitive caching, perform the following tasks:

* Develop a servlet that performs authentication and authorization
* Configure the Dispatcher

Typically, secure resources are stored in a separate folder than unsecure files. For example, /content/secure/

Create the authorization servlet

Create and deploy a servlet that performs the authentication and authorization of the user who requests the web content. The servlet can use any authentication and authorization method, such as the AEM user account and repository ACLs, or an LDAP lookup service. You deploy the servlet to the AEM instance that Dispatcher uses as the render.

The servlet must be accessible to all users. Therefore, your servlet should extend the org.apache.sling.api.servlets.SlingSafeMethodsServlet class, which provides read-only access to the system.

The servlet recieves only HEAD requests from the render, so you only need to implement the doHead method.

The render includes the URI of the requested resource as a parameter of the HTTP request. For example, an authorization servlet is accessed via /bin/permissioncheck. To perform a security check on the /content/geometrixx-outdoors/en.html page, the render includes the following URL in the HTTP request:

/bin/permissioncheck?uri=/content/geometrixx-outdoors/en.html

The servlet response message must contain the following HTTP status codes:

* 200: Authentication and authorization passed.

The following example servlet obtains the URL of the requested resource from the HTTP request. The code uses the Felix SCR Property annotation to set the value of the sling.servlet.paths property to /bin/permissioncheck. In the doHeadmethod, the servlet obtains the session object and uses the checkPermission method to determine the appropriate response code.

The value of the sling.servlet.paths property must be enabled in the Sling Servlet Resolver (org.apache.sling.servlets.resolver.SlingServletResolver) service.

Example servlet

package com.adobe.example;

import org.apache.felix.scr.annotations.Component;

import org.apache.felix.scr.annotations.Service;

import org.apache.felix.scr.annotations.Property;

import org.apache.sling.api.SlingHttpServletRequest;

import org.apache.sling.api.SlingHttpServletResponse;

import org.apache.sling.api.servlets.SlingSafeMethodsServlet;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import javax.jcr.Session;

@Component(metatype=false)

@Service

public class AuthcheckerServlet extends SlingSafeMethodsServlet {

@Property(value="/bin/permissioncheck")

static final String SERVLET\_PATH="sling.servlet.paths";

private Logger logger = LoggerFactory.getLogger(this.getClass());

public void doHead(SlingHttpServletRequest request, SlingHttpServletResponse response) {

try{

//retrieve the requested URL

String uri = request.getParameter("uri");

//obtain the session from the request

Session session = request.getResourceResolver().adaptTo(javax.jcr.Session.class);

//perform the permissions check

try {

session.checkPermission(uri, Session.ACTION\_READ);

logger.info("authchecker says OK");

response.setStatus(SlingHttpServletResponse.SC\_OK);

} catch(Exception e) {

logger.info("authchecker says READ access DENIED!");

response.setStatus(SlingHttpServletResponse.SC\_FORBIDDEN);

}

}catch(Exception e){

logger.error("authchecker servlet exception: " + e.getMessage());

}

}

}

Configure Dispatcher for permission-sensitive caching

The auth\_checker section of the dispatcher.any file controls the behavior of permission-sensitive caching. The auth\_checker section includes the following subsections:

* url: The value of the sling.servlet.paths property of the servlet that performs the security check.
* filter: Filters that specify the folders to which permission-sensitive caching is applied. Typically, a deny filter is applied to all folders, and allow filters are applied to secured folders.
* headers: Specifies the HTTP headers that the authorization servlet includes in the response.

When Dispatcher starts, the Dispatcher log file includes the following debug-level message:

AuthChecker: initialized with URL 'configured\_url'.

The following example auth\_checker section configures Dispatcher to use the servlet of the prevoius topic. The filter section causes permission checks to be performed only on secure HTML resources.

Example configuration

/auth\_checker

{

# request is sent to this URL with '?uri=<page>' appended

/url "/bin/permissioncheck"

# only the requested pages matching the filter section below are checked,

# all other pages get delivered unchecked

/filter

{

/0000

{

/glob "\*"

/type "deny"

}

/0001

{

/glob "/content/secure/\*.html"

/type "allow"

}

}

# any header line returned from the auth\_checker's HEAD request matching

# the section below will be returned as well

/headers

{

/0000

{

/glob "\*"

/type "deny"

}

/0001

{

/glob "Set-Cookie:\*"

/type "allow"

}

}

}

**Using Dispatcher with Multiple Domains**

Dispatcher versions are independent of AEM. You may have been redirected to this page if you followed a link to the Dispatcher documentation that is embedded in AEM or CQ documentation.

Use Dispatcher to process page requests in multiple web domains while supporting the following conditions:

* Web content for both domains is stored in a single AEM repository.
* The files in the Dispatcher cache can be invalidated separately for each domain.

For example, a company publishes websites for two of their brands: Brand A and Brand B. The content for the website pages are authored in AEM, and stored in the same repository workspace:

/

| - content

| - sitea

| - content nodes

| - siteb

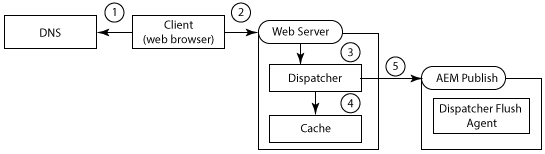
| - content nodes

Pages for BrandA.com are stored below /content/sitea. Client requests for the URL https://BrandA.com/en.html are returned the rendered page for the /content/sitea/en node. Similarly, pages for BrandB.com are stored below /content/siteb.

When using Dispatcher to cache content, associations must be made between the page URL in the client HTTP request, the path of the corresponding file in the cache, and the path of the corresponding file in the repository.

Client requests

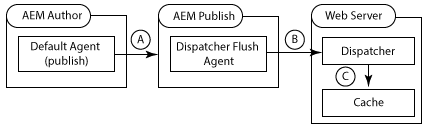
When clients send HTTP requests to the web server, the URL of the requested page must be resolved to the content in the Dispatcher cache, and eventually to the content in the repository.



1. The domain name system discovers the IP address of the web server which is registered for the domain name in the HTTP request.
2. The HTTP request is sent to the web server.
3. The HTTP request is passed to Dispatcher.
4. Dispatcher determines whether the cached files are valid. If valid, the cached files are served to the client.
5. If cached files are not valid, Dispatcher requests newly-rendered pages from the AEM publish instance.

Cache Invalidation

When Dispatcher Flush replication agents request that Dispatcher invalidates cached files, the path of the content in the repository must resolve to the content in the cache.



1. A page is activated on the AEM author instance and the content is replicated to the publish instance.
2. The Dispatcher Flush Agent calls Dispatcher to invalidate the cache for the replicated content.
3. Dispatcher touches one or more .stat files to invalidate the cached files.

To use Dispatcher with multiple domains, you need to configure AEM, Dispatcher, and your web server. The solutions described on this page are general and apply to most environments. Due to the complexity of some AEM topologies, your solution can require further custom configurations to resolve particular issues. You will likely need to adapt the examples to satisfy your existing IT infrastructure and management policies.

URL Mapping

To enable domain URLs and content paths to resolve to cached files, at some point in the process a file path or page URL must be translated. Descriptions of the following common strategies are provided, where path or URL translations occur at different points in the process:

* (Recommended) The AEM publish instance uses Sling mapping for resource resolution to implement internal URL rewriting rules. Domain URLs are translated to content repository paths. (See [AEM Rewrites Incoming URLs](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-domains.html#main-pars-title-2).)
* The web server uses internal URL rewriting rules that translate Domain URLs to cache paths. (See [The Web Server Rewrites Incoming URLs](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-domains.html#main-pars-title-1).)

It is generally desirable to use short URLs for web pages. Typically, page URLs mirror the structure of the repository folders that contain the web content. However, the URLs do not reveal the topmost repository nodes, such as /content. The client is not necessarily aware of the structure of the AEM repository.

General Requirements

Your environment must implement the following configurations to support Dispatcher working with multiple domains:

* Content for each domain resides in separate branches of the repository (see the example environment below).
* The Dispatcher Flush replication agent is configured on the AEM publish instance. (See [Invalidating Dispatcher Cache from a Publishing Instance](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/page-invalidate.html).)
* The domain name system resolves the domain names to the IP address of the web server.
* The Dispatcher cache mirrors the directory structure of the AEM content repository. The file paths below the document root of the web server are the same as the paths of the files in the repository.

Environment for the Provided Examples

The example solutions that are provided apply to an environment with the following characteristics:

* The AEM author and publish instances are deployed on Linux systems.
* Apache HTTPD is the web server, deployed on a Linux system.
* The AEM content repository and the document root of the web server use the following file structures (the document root of the Apache web server is /usr/lib/apache/httpd-2.4.3/htdocs):

**Repository**

| - /content

| - sitea

| | - content nodes

| - siteb

| - conent nodes

**Document root of the web server**

| - /usr

| - lib

| - apache

| - httpd-2.4.3

| - htdocs

| - content

| - sitea

| - content nodes

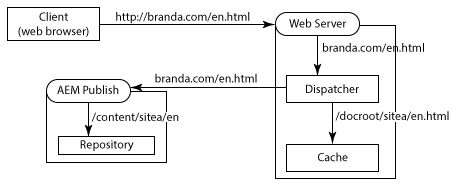
| - siteb

| - content nodes

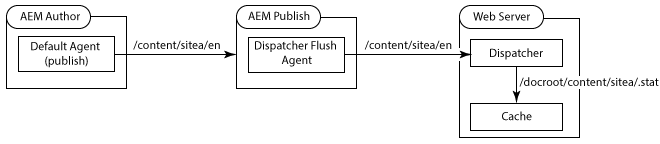
AEM Rewrites Incoming URLs

Sling mapping for resource resolution enables you to associate incoming URLs with AEM content paths. Create mappings on the AEM publish instance so that render requests from Dispatcher resolve to the correct content in the repository.

Dispatcher requests for page rendering identify the page using the URL that it is passed from the web server. When the URL includes a domain name, Sling mappings resolve the URL to the content. The following graphic illustrates a mapping of the branda.com/en.html URL to the /content/sitea/en node.



The Dispatcher cache mirrors the repository node structure. Therefore, when page activations occur the resulting requests for invalditing the cached page require no URL or path translations.



Define virtual hosts on the web server

Define virtual hosts on the web server so that a different document root can be assigned to each web domain:

* The web server must define a virtual domain for each of your web domains.
* For each domain, configure the document root to coincide with the folder in the repository that contains the domain's web content.
* Each virtual domain must also include Dispatcher-related configurations, as described on the [Installing Dispatcher](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/dispatcher-install.html)page.

The following example httpd.conf file configures two virtual domains for an Apache web server:

* The server names (which coincide with the domain names) are branda.com (line 16) and brandb.com (line 30).
* The document root of each virtual domain is the directory in the Dispatcher cache that contains the site's pages. (lines 17 and 31)

With this configuration, the web server performs the following actions when it recieves a request for https://branda.com/en/products.html:

* Associates the URL with the virtual host that has a ServerName of branda.com.
* Forwards the URL to Dispatcher.

httpd.conf

# load the Dispatcher module

LoadModule dispatcher\_module modules/mod\_dispatcher.so

# configure the Dispatcher module

<IfModule disp\_apache2.c>

DispatcherConfig conf/dispatcher.any

DispatcherLog logs/dispatcher.log

DispatcherLogLevel 3

DispatcherNoServerHeader 0

DispatcherDeclineRoot 0

DispatcherUseProcessedURL 0

DispatcherPassError 0

</IfModule>

# Define virtual host for brandA.com

<VirtualHost \*:80>

ServerName branda.com

DocumentRoot /usr/lib/apache/httpd-2.4.3/htdocs/content/sitea

<Directory /usr/lib/apache/httpd-2.4.3/htdocs/content/sitea>

<IfModule disp\_apache2.c>

SetHandler dispatcher-handler

ModMimeUsePathInfo On

</IfModule>

Options FollowSymLinks

AllowOverride None

</Directory>

</VirtualHost>

# define virtual host for brandB.com

<VirtualHost \*:80>

ServerName brandB.com

DocumentRoot /usr/lib/apache/httpd-2.4.3/htdocs/content/siteb

<Directory /usr/lib/apache/httpd-2.4.3/htdocs/content/siteb>

<IfModule disp\_apache2.c>

SetHandler dispatcher-handler

ModMimeUsePathInfo On

</IfModule>

Options FollowSymLinks

AllowOverride None

</Directory>

</VirtualHost>

# document root for web server

DocumentRoot "/usr/lib/apache/httpd-2.4.3/htdocs"

Note that virtual hosts inherit the [DispatcherConfig](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/dispatcher-install.html#main-pars-67-table-7) property value that is configured in the main server section. Virtual hosts can include their own DispatcherConfig property to override the main server configuration.

Configure Dispatcher to Handle Multiple Domains

To support URLs that include domain names and their corresponding virtual hosts, define the following Dispatcher farms:

* Configure a Dispatcher farm for each virtual host. These farms process requests from the web server for each domain, check for cached files, and request pages from the renders.
* Configure a Dispatcher farm that is used for invalidating content the cache, regardless of which domain the content belongs to. This farm handles file invalidation requests from Flush Dispatcher replication agents.

Create Dispatcher farms for virtual hosts

Farms for virtual hosts must have the following configurations so that the URLs in client HTTP requests are resolved to the correct files in the Dispatcher cache:

* The /virtualhosts property is set to the domain name. This property enables Dispatcher to associate the farm with the domain.
* The /filter property allows access to the path of the request URL truncated after the domain name part. For example, for the https://branda.com/en.html URL, the path is interpreted as /en.html, so the filter must allow access to this path.
* The /docroot property is set to the path of the root directory of the domain's site content in the Dispatcher cache. This path is used as the prefix for the concatenated URL from the original request. For example, the docroot of /usr/lib/apache/httpd-2.4.3/htdocs/sitea causes the request for https://branda.com/en.html to resolve to the /usr/lib/apache/httpd-2.4.3/htdocs/sitea/en.html file.

Additionally, the AEM publish instance must be designated as the render for the virtual host. Configure other farm properties as required. The following code is an abbreviated farm configuration for the branda.com domain:

/farm\_sitea {

...

/virtualhosts { "branda.com" }

/renders {

/rend01 { /hostname "127.0.0.1" /port "4503" }

}

/filter {

/0001 { /type "deny" /glob "\*" }

/0023 { /type "allow" /glob "\*/en\*" }

...

}

/cache {

/docroot "/usr/lib/apache/httpd-2.4.3/htdocs/content/sitea"

...

}

...

}

Create a Dispatcher farm for cache invalidation

A Dispatcher farm is required for handling requests for invalidating cached files. This farm must be able to access .stat files in the docroot directories of each virtual host.

The following property configurations enable Dispatcher to resolve files in the AEM content repository from files in the cache:

* The /docroot property is set to the default docroot of the web server. Typically, this is the directory where the /content folder is created. An example value for Apache on Linux is /usr/lib/apache/httpd-2.4.3/htdocs.
* The /filter property allows access to files below the /content directory.

The /statfileslevelproperty must be high enough so that .stat files are created in the root directory of each virtual host. This property enables the cache of each domain to be invalidated separately. For the example setup, a /statfileslevelvalue of 2 creates .stat files in the \*docroot\*/content/sitea directory and the \*docroot\*/content/siteb directory.

Additionally, the publish instance must be designated as the render for the virtual host. Configure other farm properties as required. The following code is an abbreviated configuration for the farm that is used for invalidating the cache:

/farm\_flush {

...

/virtualhosts { "invalidation\_only" }

/renders {

/rend01 { /hostname "127.0.0.1" /port "4503" }

}

/filter {

/0001 { /type "deny" /glob "\*" }

/0023 { /type "allow" /glob "\*/content\*" }

...

}

/cache {

/docroot "/usr/lib/apache/httpd-2.4.3/htdocs"

/statfileslevel "2"

...

}

...

}

When you start the web server, the Dispatcher log (in debug mode) indicates the initialization of all farms:

Dispatcher initializing (build 4.1.2)

[Fri Nov 02 16:27:18 2012] [D] [24974(140006182991616)] farms[farm\_sitea].cache.docroot = /usr/lib/apache/httpd-2.4.3/htdocs/content/sitea

[Fri Nov 02 16:27:18 2012] [D] [24974(140006182991616)] farms[farm\_siteb].cache.docroot = /usr/lib/apache/httpd-2.4.3/htdocs/content/siteb

[Fri Nov 02 16:27:18 2012] [D] [24974(140006182991616)] farms[farm\_flush].cache.docroot = /usr/lib/apache/httpd-2.4.3/htdocs

[Fri Nov 02 16:27:18 2012] [I] [24974(140006182991616)] Dispatcher initialized (build 4.1.2)

Configure Sling Mapping for Resource Resolution

Use Sling mapping for resource resolution so that domain-based URLs resolve to content on the AEM publish instance. The resource mapping translates the incoming URLs from Dispatcher (originally from client HTTP requests) to content nodes.

To learn about Sling resource mapping, see [Mappings for Resource Resolution](https://sling.apache.org/site/mappings-for-resource-resolution.html) in the Sling documentation.

Typically, mappings are required for the following resources, although additional mappings can be necessary:

* The root node of the content page (below /content)
* The design node that the pages use (below /etc/designs)
* The /libs folder

After you create the mapping for the content page, to discover additional required mappings use a web browser to open a page on the web server. In the error.log file of the publish instance, locate messages about resources that are not found. The following example message indicates that a mapping for /etc/clientlibs is required:

01.11.2012 15:59:24.601 \*INFO\* [10.36.34.243 [1351799964599] GET /etc/clientlibs/foundation/jquery.js HTTP/1.1] org.apache.sling.engine.impl.SlingRequestProcessorImpl service: Resource /content/sitea/etc/clientlibs/foundation/jquery.js not found

The linkchecker transformer of the default Apache Sling rewriter automatically modifies hyperlinks in the page to prevent broken links. However, link rewriting is performed only when the link target is an HTML or HTM file. To update links to other file types, create a transformer component and add it to an HTML rewriter pipeline.

Example resource mapping nodes

The following table lists the nodes that implement resource mapping for the branda.com domain. Similar nodes are created for the brandb.com domain, such as /etc/map/http/brandb.com. In all cases, mappings are required when references in the page HTML to not resolve correctly in the context of Sling.

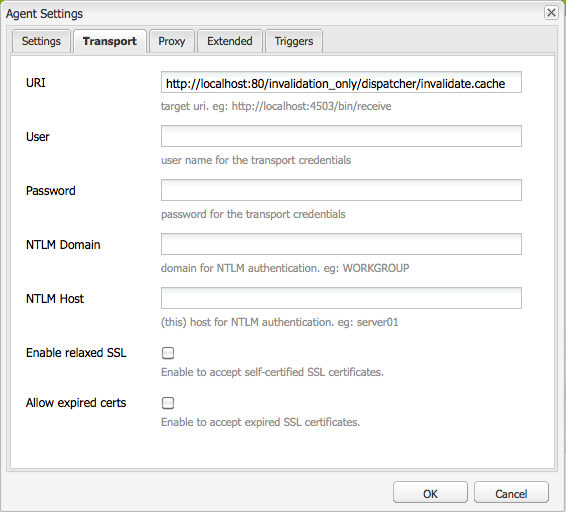
| **Node path** | **Type** | **Property** |
| --- | --- | --- |
| /etc/map/http/branda.com | sling:Mapping | Name: sling:internalRedirect Type: String Value: /content/sitea |
| /etc/map/http/branda.com/libs | sling:Mapping | Name: sling:internalRedirect  Type: String  Value: /libs |
| /etc/map/http/branda.com/etc | sling:Mapping |  |
| /etc/map/http/branda.com/etc/designs | sling:Mapping | Name: sling:internalRedirect  VType: String  VValue: /etc/designs |
| /etc/map/http/branda.com/etc/clientlibs | sling:Mapping | Name: sling:internalRedirect  VType: String  VValue: /etc/clientlibs |

Configuring the Dispatcher Flush replication agent

The Dispatcher Flush replication agent on the AEM publish instance must send invalidation requests to the correct Dispatcher farm. To target a farm, use the URI property of the Dispatcher Flush replication agent (on the Transport tab). Include the value of the /virtualhost property for the Dispatcher farm that is configured for invalidating the cache:

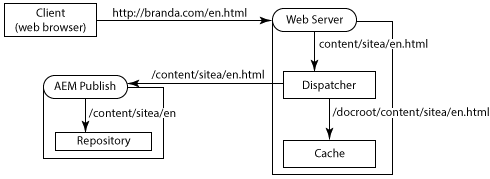
https://\*webserver\_name\*:\*port\*/\*virtual\_host\*/dispatcher/invalidate.cache

For example, to use the farm\_flush farm of the previous example, the URI is https://localhost:80/invalidation\_only/dispatcher/invalidate.cache.

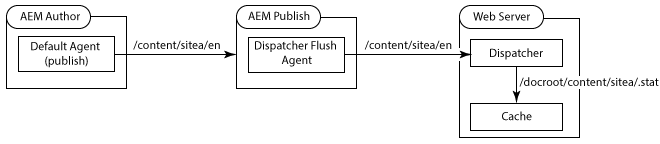


The Web Server Rewrites Incoming URLs

Use the internal URL rewriting feature of your web server to translate domain-based URLs to file paths in the Dispatcher cache. For example, client requests for the https://brandA.com/en.html page are translated to the content/sitea/en.htmlfile in the document root of the web server.



The Dispatcher cache mirrors the repository node structure. Therefore, when page activations occur the resulting requests for invalidating the cached page require no URL or path translations.



Define virtual hosts and rewrite rules on the Web server

Configure the following aspects on the web server:

* Define a virtual host for each of your web domains.
* For each domain, configure the document root to coincide with the folder in the repository that contains the domain's web content.
* For each virtual domain, create a URL renaming rule that translates the incoming URL to the path of the cached file.
* Each virtual domain must also include Dispatcher-related configurations, as described on the [Installing Dispatcher](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/dispatcher-install.html)page.
* The Dispatcher module must be configured to use the URL that the web server has rewritten. (See the DispatcherUseProcessedURL property in [Installing Dispatcher](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/dispatcher-install.html).)

The following example httpd.conf file configures two virtual hosts for an Apache web server:

* The server names (which coincide with the domain names) are brandA.com (line 16) and brandB.com (line 32).
* The document root of each virtual domain is the directory in the Dispatcher cache that contains the site's pages. (lines 20 and 33)
* The URL rewrite rule for each virtual domain is a regular expression that prefixes the path of the requested page with the path to the pages in the cache. (lines 19 and 35)
* The DispatherUseProcessedURL property is set to 1. (line 10)

For example, the web server performs the following actions when it receives a request with the https://brandA.com/en/products.html URL:

* Associates the URL with the virtual host that has a ServerName of brandA.com.
* Rewrites the URL to be /content/sitea/en/products.html.
* Forwards the URL to Dispatcher.

httpd.conf

# load the Dispatcher module

LoadModule dispatcher\_module modules/mod\_dispatcher.so

# configure the Dispatcher module

<IfModule disp\_apache2.c>

DispatcherConfig conf/dispatcher.any

DispatcherLog logs/dispatcher.log

DispatcherLogLevel 3

DispatcherNoServerHeader 0

DispatcherDeclineRoot 0

DispatcherUseProcessedURL 1

DispatcherPassError 0

</IfModule>

# Define virtual host for brandA.com

<VirtualHost \*:80>

ServerName branda.com

DocumentRoot /usr/lib/apache/httpd-2.4.3/htdocs/content/sitea

RewriteEngine on

RewriteRule ^/(.\*)\.html$ /content/sitea/$1.html [PT]

<Directory /usr/lib/apache/httpd-2.4.3/htdocs/content/sitea>

<IfModule disp\_apache2.c>

SetHandler dispatcher-handler

ModMimeUsePathInfo On

</IfModule>

Options FollowSymLinks

AllowOverride None

</Directory>

</VirtualHost>

# define virtual host for brandB.com

<VirtualHost \*:80>

ServerName brandB.com

DocumentRoot /usr/lib/apache/httpd-2.4.3/htdocs/content/siteb

RewriteEngine on

RewriteRule ^/(.\*)\.html$ /content/siteb/$1.html [PT]

<Directory /usr/lib/apache/httpd-2.4.3/htdocs/content/siteb>

<IfModule disp\_apache2.c>

SetHandler dispatcher-handler

ModMimeUsePathInfo On

</IfModule>

Options FollowSymLinks

AllowOverride None

</Directory>

</VirtualHost>

# document root for web server

DocumentRoot "/usr/lib/apache/httpd-2.4.3/htdocs"

Configure a Dispatcher Farm

When the web server rewrites URLs, Dispatcher requires a single farm defined according to [Configuring Dispatcher](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html). The following configurations are required to support the web server virtual hosts and URL renaming rules:

* The /virtualhosts property must include the ServerName values for the all VirtualHost definitions.
* The /statfileslevel property must be high enough to create .stat files in the directories that contain the content files for each domain.

The following example configuration file is based on the example dispatcher.any file that is installed with Dispatcher. The following changes are required to support the web server configurations of the previous httpd.conf file:

* The /virtualhosts property causes Dispatcher to handle requests for the brandA.com and brandB.com domains. (line 12)
* The /statfileslevel property is set to 2, so that stat files are created in each directory that contains the domain's web content (line 41): /statfileslevel "2"

As usual, the document root of the cache is the same as the document root of the web server (line 40): /usr/lib/apache/httpd-2.4.3/htdocs

dispatcher.any

/name "testDispatcher"

/farms

{

/dispfarm0

{

/clientheaders

{

"\*"

}

/virtualhosts

{

"brandA.com" "brandB.com"

}

/renders

{

/rend01 { /hostname "127.0.0.1" /port "4503" }

}

/filter

{

/0001 { /type "deny" /glob "\*" }

/0023 { /type "allow" /glob "\*/content\*" } # disable this rule to allow mapped content only

/0041 { /type "allow" /glob "\* \*.css \*" } # enable css

/0042 { /type "allow" /glob "\* \*.gif \*" } # enable gifs

/0043 { /type "allow" /glob "\* \*.ico \*" } # enable icos

/0044 { /type "allow" /glob "\* \*.js \*" } # enable javascript

/0045 { /type "allow" /glob "\* \*.png \*" } # enable png

/0046 { /type "allow" /glob "\* \*.swf \*" } # enable flash

/0061 { /type "allow" /glob "POST /content/[.]\*.form.html" } # allow POSTs to form selectors under content

/0062 { /type "allow" /glob "\* /libs/cq/personalization/\*" } # enable personalization

/0081 { /type "deny" /glob "GET \*.infinity.json\*" }

/0082 { /type "deny" /glob "GET \*.tidy.json\*" }

/0083 { /type "deny" /glob "GET \*.sysview.xml\*" }

/0084 { /type "deny" /glob "GET \*.docview.json\*" }

/0085 { /type "deny" /glob "GET \*.docview.xml\*" }

/0086 { /type "deny" /glob "GET \*.\*[0-9].json\*" }

/0090 { /type "deny" /glob "\* \*.query.json\*" }

}

/cache

{

/docroot "/usr/lib/apache/httpd-2.4.3/htdocs"

/statfileslevel "2"

/allowAuthorized "0"

/rules

{

/0000 { /glob "\*" /type "allow" }

}

/invalidate

{

/0000 { /glob "\*" /type "deny" }

/0001 { /glob "\*.html" /type "allow" }

}

/allowedClients

{

}

}

/statistics

{

/categories

{

/html { /glob "\*.html" }

/others { /glob "\*" }

}

}

}

}

Because a single Dispatcher farm is defined, the Dispatcher Flush replication agent on the AEM publish instance requires no special configurations.

Rewriting Links to Non-HTML Files

To rewrite references to files that have extensions other than .html or .htm, create a Sling rewriter transformer component and add it to the default rewriter pipeline.

Rewrite references when resource paths do not resolve correctly in the web server context. For example, a transformer is required when image-generating components create links such as /content/sitea/en/products.navimage.png. The topnav component of the [How to Create a Fully Featured Internet Website](https://helpx.adobe.com/experience-manager/6-3/sites/developing/using/the-basics.html) creates such links.

The [Sling rewriter](https://sling.apache.org/documentation/bundles/output-rewriting-pipelines-org-apache-sling-rewriter.html) is a module that post-processes Sling output. SAX pipeline implementations of the rewriter consist of a generator, one or more transformers, and a serializer:

* **Generator:** Parses the Sling output stream (HTML document) and generates SAX events when it encounters specific element types.
* **Transformer:** Listens for SAX events and consequently modifies the event target (an HTML element). A rewriter pipeline contains zero or more transformers. Transformers are executed in sequence, passing the SAX events to the next transformer in the sequence.
* **Serializer:** Serializes the output, including the modifications from each transformer.

https://docs.adobe.com/content/dam/help/experience-manager-dispatcher.en/help/using/assets/chlimage_1-15.png

The AEM Default Rewriter Pipeline

AEM uses a default pipeline rewriter that processes documents of type text/html:

* The generator parses HTML documents and generates SAX events when it encounters a, img, area, form, base, link, script, and body elements. The generator alias is htmlparser.
* The pipeline includes the following transformers: linkchecker, mobile, mobiledebug, contentsync. The linkchecker transformer externalizes paths to referenced HTML or HTM files to prevent broken links.
* The serializer writes the HTML output. The serializer alias is htmlwriter.

The /libs/cq/config/rewriter/default node defines the pipeline.

Creating a Transformer

Perform the following tasks to create a transformer component and use it in a pipeline:

1. Implement the org.apache.sling.rewriter.TransformerFactory interface. This class creates instances of your transformer class. Specify values for the transformer.type property (the transformer alias) and configure the class as an OSGi service component.
2. Implement the org.apache.sling.rewriter.Transformer interface. To minimize the work, you can extend the org.apache.cocoon.xml.sax.AbstractSAXPipe class. Override the startElement method to customize the rewriting behavior. This method is called for every SAX event that is passed to the transformer.
3. Bundle and deploy the classes.
4. Add a configuration node to your AEM application to add the transformer to the pipeline.

You can instead configure the TransformerFactory to that the transformer is inserted into every rewriter that is defined. Consequently you do not need to configure a pipeline:

* Set the pipeline.mode property to global.
* Set the service.ranking property to a positive integer.
* Do not include a pipeline.type property.

Use the [multimodule](https://helpx.adobe.com/experience-manager/aem-previous-versions.html) archetype of the Content Package Maven Plugin to create your Maven project. The POMs automatically create and install a content package.

The following examples implement a transformer that rewrites references to image files.

* The MyRewriterTransformerFactory class instantiates MyRewriterTransformer objects. The pipeline.type property sets the transformer alias to mytransformer. To include the alias in a pipeline, the pipeline configuration node includes this alias in the list of transformers.
* The MyRewriterTransformer class overrides the startElement method of the AbstractSAXTransformer class. The startElement method rewrites the value of src attributes for img elements.

The examples are not robust and should not be used in a production environment.

Example TransformerFactory implementation

package com.adobe.example;

import org.apache.felix.scr.annotations.Component;

import org.apache.felix.scr.annotations.Service;

import org.apache.felix.scr.annotations.Property;

import org.apache.sling.rewriter.Transformer;

import org.apache.sling.rewriter.TransformerFactory;

@Component

@Service

public class MyRewriterTransformerFactory implements TransformerFactory {

/\* Define the alias \*/

@Property(value="mytransformer")

static final String PIPELINE\_TYPE ="pipeline.type";

public Transformer createTransformer() {

return new MyRewriterTransformer ();

}

}

Example Transformer implementation

package com.adobe.example;

import java.io.IOException;

import org.apache.cocoon.xml.sax.AbstractSAXPipe;

import org.apache.sling.api.SlingHttpServletRequest;

import org.apache.sling.rewriter.ProcessingComponentConfiguration;

import org.apache.sling.rewriter.ProcessingContext;

import org.apache.sling.rewriter.Transformer;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.xml.sax.Attributes;

import org.xml.sax.SAXException;

import org.xml.sax.helpers.AttributesImpl;

import javax.servlet.http.HttpServletRequest;

public class MyRewriterTransformer extends AbstractSAXPipe implements Transformer {

private static final Logger log = LoggerFactory.getLogger(MyRewriterTransformer.class);

private SlingHttpServletRequest httpRequest;

/\* The element and attribute to act on \*/

private static final String ATT\_NAME = new String("src");

private static final String EL\_NAME = new String("img");

public MyRewriterTransformer () {

}

public void dispose() {

}

public void init(ProcessingContext context, ProcessingComponentConfiguration config) throws IOException {

this.httpRequest = context.getRequest();

log.debug("Transforming request {}.", httpRequest.getRequestURI());

}

@Override

public void startElement (String nsUri, String localname, String qname, Attributes atts) throws SAXException {

/\* copy the element attributes \*/

AttributesImpl linkAtts = new AttributesImpl(atts);

/\* Only interested in EL\_NAME elements \*/

if(EL\_NAME.equalsIgnoreCase(localname)){

/\* iterate through the attributes of the element and act only on ATT\_NAME attributes \*/

for (int i=0; i < linkAtts.getLength(); i++) {

if (ATT\_NAME.equalsIgnoreCase(linkAtts.getLocalName(i))) {

String path\_in\_link = linkAtts.getValue(i);

/\* use the resource resolver of the http request to reverse-resolve the path \*/

String mappedPath = httpRequest.getResourceResolver().map(httpRequest, path\_in\_link);

log.info("Tranformed {} to {}.", path\_in\_link,mappedPath);

/\* update the attribute value \*/

linkAtts.setValue(i,mappedPath);

}

}

}

/\* return updated attributes to super and continue with the transformer chain \*/

super.startElement(nsUri, localname, qname, linkAtts);

}

}

Adding the Transformer to a Rewriter Pipeline

Create a JCR node that defines a pipeline that uses your transformer. The following node definition creates a pipeline that processes text/html files. The default AEM generator and parser for HTML are used.

If you set the Transformer property pipeline.mode to global, you do not need to configure a pipeline. The globalmode inserts the transformer into all pipelines.

Rewriter configuration node - XML representation

<?xml version="1.0" encoding="UTF-8"?>

<jcr:root xmlns:jcr="https://www.jcp.org/jcr/1.0" xmlns:nt="https://www.jcp.org/jcr/nt/1.0"

jcr:primaryType="nt:unstructured"

contentTypes="[text/html]"

enabled="{Boolean}true"

generatorType="htmlparser"

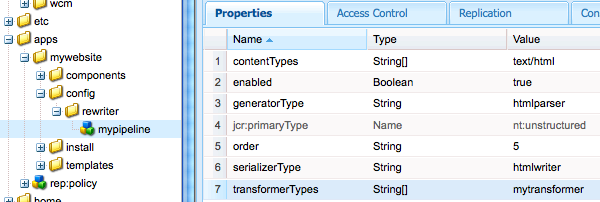
order="5"

serializerType="htmlwriter"

transformerTypes="[mytransformer]">

</jcr:root>

The following graphic shows the CRXDE Lite representation of the node:



# Using SSL with Dispatcher

Use SSL connections between Dispatcher and the render computer:

* [One-way SSL](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-ssl.html#main-pars-title-1)
* [Mutual SSL](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-ssl.html#main-pars-title-2)

Operations related to the SSL certificates are bound to third party products. They are not covered by the Adobe Platinum Maintenance and Support contract.

## Use SSL When Dispatcher Connects to AEM

Configure Dispatcher to communicate with the AEM or CQ render instance using SSL connections.

Before you configure Dispatcher, configure AEM or CQ to use SSL:

* AEM 6.2: [Enabling HTTP Over SSL](https://helpx.adobe.com/experience-manager/6-2/sites/deploying/using/config-ssl.html)
* AEM 6.1: [Enabling HTTP Over SSL](https://docs.adobe.com/content/docs/en/aem/6-1/deploy/configuring/config-ssl.html)
* Older AEM versions: see [this page](https://helpx.adobe.com/experience-manager/aem-previous-versions.html).

### SSL-Related Request Headers

When Dispatcher recieves an HTTPS request, Dispatcher includes the following headers in the subsequent request that it sends to AEM or CQ:

* X-Forwarded-SSL
* X-Forwarded-SSL-Cipher
* X-Forwarded-SSL-Keysize
* X-Forwarded-SSL-Session-ID

A request through Apache-2.2 with mod\_ssl includes headers that are similar to the following example:

X-Forwarded-SSL: on

X-Forwarded-SSL-Cipher: DHE-RSA-AES256-SHA

X-Forwarded-SSL-Session-ID: 814825E8CD055B4C166C2EF6D75E1D0FE786FFB29DEB6DE1E239D5C771CB5B4D

### Configuring Dispatcher to Use SSL

To configure Dispatcher to connect with AEM or CQ over SSL, your [dispatcher.any](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html) file requires the following properties:

* A virtual host that handles HTTPS requests.
* The renders section of the virtual host includes an item that identifies the host name and port of the CQ or AEM instance that uses HTTPS.
* The renders item includes a property named secure of value 1.

Note: Create another virtual host for handling HTTP requests if required.

The following example dispatcher.any file shows the property values for connecting using HTTPS to a CQ instance that is running on host localhost and port 8443:

/farms

{

/secure

{

/virtualhosts

{

# select this farm for all incoming HTTPS requestss

"https://\*"

}

/renders

{

/0001

{

# hostname or IP of the render

/hostname "localhost"

# port of the render

/port "8443"

# connect via HTTPS

/secure "1"

}

}

# the rest of the properties are omitted

}

/non-secure

{

/virtualhosts

{

# select this farm for all incoming HTTP requests

"https://\*"

}

/renders

{

/0001

{

# hostname or IP of the render

/hostname "localhost"

# port of the render

/port "4503"

}

}

# the rest of the properties are omitted

}

## Configuring Mutual SSL Between Dispatcher and AEM

Configure the connections between Dispatcher and the render computer (typically an AEM or CQ publish instance) to use Mutual SSL:

* Dispatcher connects to the render instance over SSL.
* The render instance verifies the validity of Dispatcher's certificate.
* Dispatcher verifies that the CA of the the render instance's certificate is trusted.
* (Optional) Dispatcher verifies that the certificate of the render instance matches the render instance's server address.

To configure mutual SSL, you require certificates that are signed by a trusted certificate authority (CA). Self-signed certificates are not adequate. You can either act as the CA or use the services of a third-party CA to sign your certificates. To configure mutual SSL, you require the following items:

* Signed certificates for the render instance and Dispatcher
* The CA certificate (if you are acting as the CA)
* OpenSSL libraries for generating the CA, certificates, and certificate requests.

Perform the following steps to configure mutual SSL:

1. [Install](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/dispatcher-install.html) the latest version of Dispatcher for your platform. Use a Dispatcher binary that supports SSL (SSL is in the file name, such as dispatcher-apache2.4-linux-x86-64-ssl10-4.1.7.tar).
2. [Create or obtain CA-signed certificate](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-ssl.html#main-pars-title-3) for Dispatcher and the render instance.
3. [Create a keystore containing render certificate](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-ssl.html#main-pars-title-6) and configure the render's HTTP service to use it.
4. [Configure the Dispatcher web server module](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-ssl.html#main-pars-title-4) for mutual SSL.

### Creating or Obtaining CA-Signed Certificates

Create or obtain the CA-signed certificates that authenticate the publish instance and Dispatcher.

#### Creating Your CA

If you are acting as the CA, use [OpenSSL](https://www.openssl.org/) to create the Certificate Authority that signs the server and client certificates. (You must have the OpenSSL libraries installed.) If you are using a third-party CA, do not perform this procedure.

1. Open a terminal and change the current directory to the directory that contiains the CA.sh file, such as /usr/local/ssl/misc.
2. To create the CA, enter the following command and then provide values when promtped:

./CA.sh -newca

Several properties in the openssl.cnf file control the behavior of the CA.sh script. You should modify this file as required before you create your CA.

#### Creating the Certificates

Use OpenSSL to create the certificate requests to send to the third-party CA or to sign with your CA.

When you create a certificate, OpenSSL uses the Common Name property to identify the certificate holder. For the certificate of the render instance, use the instance computer's host name as the Common Name if you are configuring Dispatcher to accept the certificate only if it matches the hostname of the Publish instance. (See the [DispatcherCheckPeerCN](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-ssl.html#main-pars-title-11) property.)

1. Open a terminal and change the current directory to the directory that contains the CH.sh file of your OpenSSL libraries.
2. Enter the following command and provide values when prompted. If required, use the host name of the publish instance as the the Common Name. The host name is DNS-resolvable name for the IP address of the render:

./CA.sh -newreq

If you are using a third-party CA, send the newreq.pem file to the CA to sign. If you are acting as the CA, continue to step 3.

1. Enter the following command to sign the certificate using the certificate of your CA:

./CA.sh -sign

Two files named newcert.pem and newkey.pem are created in the directory that contains your CA management files. These are the public certificate and private key for the render computer, respectively.

1. Rename newcert.pem to rendercert.pem, and rename newkey.pem to renderkey.pem.
2. Repeat steps 2 and 3 to create a new certificate and a new public key for the Dispatcher module. Ensure that you use a Common Name that is specific to the Dispatcher instance.
3. Rename newcert.pem to dispcert.pem, and rename newkey.pem to dispkey.pem.

### Configuring SSL on the Render Computer

Configure SSL on the render instance using the rendercert.pem and renderkey.pem files.

#### Converting the Render Certificate to JKS Format

Use the following commmand to convert the render certificate, which is a PEM file, to a PKCS#12 file. Also include the certificate of the CA that signed the render certificate:

1. In a terminal window, change the current directory to the location of the render certificate and private key.
2. Enter the following commmand to convert the render certificate, which is a PEM file, to a PKCS#12 file. Also include the certificate of the CA that signed the render certificate:

openssl pkcs12 -export -in rendercert.pem -inkey renderkey.pem -certfile demoCA/cacert.pem -out rendercert.p12

1. Enter the following command to convert PKCS#12 file to Java KeyStore (JKS) format:

keytool -importkeystore -srckeystore servercert.p12 -srcstoretype pkcs12 -destkeystore render.keystore

1. The Java Keystore is created using a default alias. Change the alias if desired:

keytool -changealias -alias 1 -destalias jettyhttp -keystore render.keystore

#### Adding the CA Cert to the Render's Truststore

If you are acting as the CA, import your CA certificate into a keystore. Then, configure the JVM that runs the render instance to trust the keystore.

1. Use a text editor to open the cacert.pem file and remove all the text that precedes the followign line:

-----BEGIN CERTIFICATE-----

1. Use the following command to import the certificate into a keystore:

keytool -import -keystore cacerts.keystore -alias myca -storepass password -file cacert.pem

1. To configure the JVM that runs your render instance to trust the keystore, use the following system property:

-Djavax.net.ssl.trustStore=<location of cacerts.keystore>

For example, if you use the crx-quickstart/bin/quickstart script to start your publish instance you can modify the CQ\_JVM\_OPTS property:

CQ\_JVM\_OPTS='-server -Xmx2048m -XX:MaxPermSize=512M -Djavax.net.ssl.trustStore=/usr/lib/cq6.0/publish/ssl/cacerts.keystore'

#### Configuring the Render Instance

Use the render certificate with the instructions in the *Enable SSL on the Publish Instance* section to configure the HTTP service of the render instance to use SSL:

* AEM 6.2: [Enabling HTTP Over SSL](https://helpx.adobe.com/experience-manager/6-2/sites/deploying/using/config-ssl.html)
* AEM 6.1: [Enabling HTTP Over SSL](https://docs.adobe.com/content/docs/en/aem/6-1/deploy/configuring/config-ssl.html)
* Older AEM versions: see [this page.](https://helpx.adobe.com/experience-manager/aem-previous-versions.html)

### Configuring SSL for the Dispatcher Module

To configure Dispatcher to use mutual SSL, prepare the Dispatcher certificate and then configure the web server module.

### Creating a Unified Dispatcher Certificate

Combine the dispatcher certificate and the unencrypted private key into a single PEM file. Use a text editor or the cat command to create a file that is similar to the following example:

1. Open a terminal and change the current directory to the location of the dispkey.pem file.
2. To decrypt the private key, enter the following command:

openssl rsa -in dispkey.pem -out dispkey\_unencrypted.pem

1. Use a text editor or the cat command to combine the unencrypted private key and the certificate in a single file that is similar to the following example:
2. -----BEGIN RSA PRIVATE KEY-----
3. MIICxjBABgkqhkiG9w0B...
4. ...M2HWhDn5ywJsX
5. -----END RSA PRIVATE KEY-----
6. -----BEGIN CERTIFICATE-----
7. MIIC3TCCAk...
8. ...roZAs=

-----END CERTIFICATE-----

### Specifying the Certificate to Use for Dispatcher

Add the following properties to the [Dispatcher module configuration](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/dispatcher-install.html#main-pars-55-35-1022) (in httpd.conf):

* DispatcherCertificateFile: The path to the Dispatcher unified certificate file, containing the public certificate and the unencrypted private key. This file is used when SSL server requests the Dispatcher client certificate.
* DispatcherCACertificateFile: The path to the CA certificate file, used if the SSL server presents a CA that is not trusted by a root authority.
* DispatcherCheckPeerCN: Whether to enable ( On) or disable ( Off) host name checking for remote server certificates.

The following code is an example configuration:

<IfModule disp\_apache2.c>

DispatcherConfig conf/dispatcher.any

DispatcherLog logs/dispatcher.log

DispatcherLogLevel 3

DispatcherNoServerHeader 0

DispatcherDeclineRoot 0

DispatcherUseProcessedURL 0

DispatcherPassError 0

DispatcherCertificateFile disp\_unified.pem

DispatcherCACertificateFile cacert.pem

DispatcherCheckPeerCN On

</IfModule>

**Configuring Dispatcher to Prevent CSRF Attacks**

AEM provides a framework aimed at preventing Cross-Site Request Forgery attacks. In order to properly make use of this framework, you need to make the following changes to your dispatcher configuration:

Be sure to update the rule numbers in the examples below based on your existing configuration. Remember that dispatchers will use the last matching rule to grant an allow or deny, so place the rules near the bottom of your existing list.

1. In the /clientheaders section of your author-farm.any and publish-farm.any, add the following entry to the bottom of the list:CSRF-Token
2. In the /filters section of your author-farm.any and publish-farm.any or publish-filters.any file, add the following line to allow requests for /libs/granite/csrf/token.json through the dispatcher./0999 { /type "allow" /glob " \* /libs/granite/csrf/token.json\*" }
3. Under the /cache /rules section of your publish-farm.any, add a rule to block the dispatcher from caching the token.json file. Typically authors bypass caching, so you should not need to add the rule into your author-farm.any./0999 { /glob "/libs/granite/csrf/token.json" /type "deny" }

To validate that the configuration is working, watch the dispatcher.log in DEBUG mode to validate that the token.json file is not being cached and is not being blocked by filters. You should see messages similar to:... checking [/libs/granite/csrf/token.json] ... request URL not in cache rules: /libs/granite/csrf/token.json... cache-action for [/libs/granite/csrf/token.json]: NONE

You can also validate that requests are succeeding in your apache access\_log. Requests for ``/libs/granite/csrf/token.json should return an HTTP 200 status code.

**Troubleshooting Dispatcher Problems**

Dispatcher versions are independent of AEM, however the Dispatcher documentation is embedded in the AEM documentation. Always use the Dispatcher documentation that is embedded in the documentation for the latest version of AEM.

You may have been redirected to this page if you followed a link to the Dispatcher documentation that is embedded in the documentation for a previous version of AEM.

Please also check the [Dispatcher Knowledge Base](https://helpx.adobe.com/cq/kb/index/dispatcher.html), [Troubleshooting Dispatcher Flushing Issues](https://helpx.adobe.com/adobe-cq/kb/troubleshooting-dispatcher-flushing-issues.html) and the [Dispatcher Top Issues FAQ](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/troubleshooting/dispatcher-faq.html) for further information.

Check the Basic Configuration

As always the first steps are to check the basics:

* [Confirm Basic Operation](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/troubleshooting/dispatcher-troubleshooting.html#ConfirmBasicOperation)
* Check all log files for your web server and dispatcher. If necessary increase the loglevel used for the dispatcher [logging](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/troubleshooting/dispatcher-troubleshooting.html#Logging).
* [Check your configuration](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/troubleshooting/dispatcher-troubleshooting.html#ConfiguringtheDispatcher):
  + Do you have multiple Dispatchers?
    - Have you determined which Dispatcher is handling the website / page you are investigating?
  + Have you implemented filters?
    - Are these impacting the matter you are investigating?

IIS Diagnostic Tools

IIS provides various trace tools, dependent on the actual version:

* IIS 6 - IIS diagnostic tools can be downloaded and configured
* IIS 7 - tracing is fully integrated

These can help you monitor activity.

IIS and 404 Not Found

When using IIS you might experience 404 Not Found being returned in various scenarios. If so, see the following Knowledge Base articles.

* [IIS 6/7: POST method returns 404](https://helpx.adobe.com/dispatcher/kb/IIS6IsapiFilters.html)
* [IIS 6: Requests to URLs that contain the base path](https://helpx.adobe.com/dispatcher/kb/RequestsToBinDirectoryFailInIIS6.html)

You should also check that the dispatcher cache root and the IIS document root are set to the same directory.

Problems Deleting Workflow Models

**Symptoms**

Problems trying to delete workflow models when accessing an AEM author instance through the Dispatcher.

**Steps to reproduce:**

1. Log in to your author instance (confirm that requests are being routed through the dispatcher).
2. Create a new workflow; for example, with the Title set to workflowToDelete.
3. Confirm that the workflow was successfully created.
4. Select and right click on the workflow, then click **Delete**.
5. Click **Yes** to confirm.
6. An error message box will appear showing:" ERROR 'Could not delete workflow model!!".

**Resolution**

Add the following headers to the /clientheaders section of your dispatcher.any file:

* x-http-method-override
* x-requested-with

{ { /clientheaders { ... "x-http-method-override" "x-requested-with" }

Interference with mod\_dir (Apache)

This describes how the dispatcher interacts with mod\_dir inside the Apache webserver, as this can lead to various, potentially unexpected effects:

Apache 1.3

In Apache 1.3 mod\_dir handles every request where the URL maps to a directory in the file system.

It will either:

* redirect the request to an existing index.html file
* generate a directory listing

When the dispatcher is enabled, it processes such requests by registering itself as a handler for the content type httpd/unix-directory.

Apache 2.x

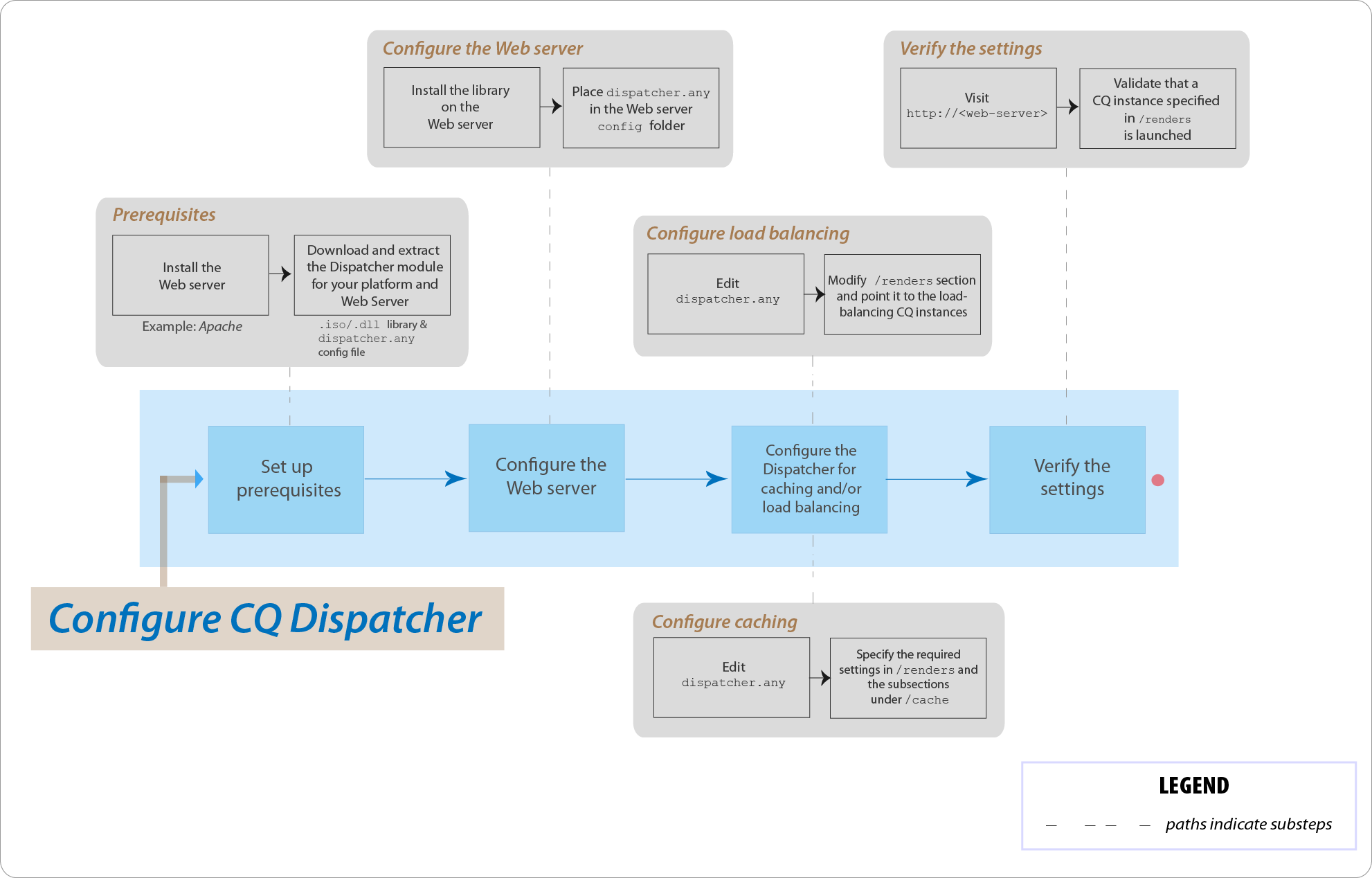
In Apache 2.x things are different. A module can handle different stages of the request, such as URL fixup. mod\_dir handles this stage by redirecting a request (when the URL maps to a directory) to the URL with a / appended.

Dispatcher does not intercept the mod\_dir fixup, but completely handles the request to the redirected URL (i.e. with / appended). This might pose a problem if the remote server (e.g. AEM) handles requests to /a\_path differently to requests to /a\_path/ (when /a\_path maps to an existing directory).

If this happens you must either:

* disable mod\_dir for the Directory or Location subtree handled by the dispatcher
* use DirectorySlash Off to configure mod\_dir not to append /

**AEM Dispatcher Top Issues FAQ**



Introduction

What is the Dispatcher?

The Dispatcher is Adobe Experience Manager's caching and/or load balancing tool that helps realize a fast and dynamic Web authoring environment. For caching, the Dispatcher works as part of an HTTP server, such as Apache, with the aim of storing (or "caching") as much of the static website content as possible and accessing the website's layout engine as infrequently as possible. In a load balancing role, the Dispatcher distributes user requests (load) across different AEM instances (renders).

For caching, the Dispatcher module uses the Web server's ability to serve static content. The Dispatcher places the cached documents in the document root of the Web server.

How does the Dispatcher perform caching?

The Dispatcher uses the web server's ability to serve static content. The Dispatcher stores cached documents in the web server’s document root. The Dispatcher has two primary methods for updating the cache content when changes are made to the website.

* **Content Updates** remove the pages that have changed, as well as files that are directly associated with them.
* **Auto-Invalidation** automatically invalidates those parts of the cache that may be out of date after an update. For example, it effectively flags relevant pages as being out of date, without deleting anything.

What are the benefits of load-balancing?

Load Balancing distributes user requests (load) across several AEM instances.The following list describes the advantages for load balancing:

* **Increased processing power**: In practice this means that the Dispatcher shares document requests between several instances of AEM. Because each instance has fewer documents to process, you have faster response times. The Dispatcher keeps internal statistics for each document category, so it can estimate the load and distribute the queries efficiently.
* **Increased fail-safe coverage**: If the Dispatcher does not receive responses from an instance, it will automatically relay requests to one of the other instance(s). Thus, if an instance becomes unavailable, the only effect is a slowdown of the site, proportionate to the computational power lost.

For further details, see the [Dispatcher Overview page](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/dispatcher.html)

Install and Configure

Where do I download the Dispatcher module from?

You can download the latest Dispatcher module from the [Dispatcher Release Notes](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/release-notes.html) page.

How do I install the Dispatcher module?

Refer to the [Installing Dispatcher](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/dispatcher-install.html) page

How do I configure the Dispatcher module?

See the [Configuring Dispatcher](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html) page.

How do I configure the Dispatcher for the author instance?

See [Using Dispatcher with an Author Instance](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/dispatcher.html#using-a-dispatcher-with-an-author-server) for the detailed steps.

How do I configure the Dispatcher with multiple domains?

You can configure the CQ Dispatcher with multiple domains, provided the domains satisfy the following conditions:

* The Web content for both the domains is stored in a single AEM repository
* The files in the Dispatcher cache can be invalidated separately for each domain

Read [Using Dispatcher with Multiple Domains](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-domains.html) for further details.

How do I configure the Dispatcher, such that all requests from a user are routed to the same Publish instance?

You can use the [sticky connections](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#identifying-a-sticky-connection-folder-stickyconnectionsfor) feature, which ensures that all documents for a user are processed on the same instance of AEM. This feature is important if you use personalized pages and session data. The data is stored on the instance. Therefore, subsequent requests from the same user must return to that instance or the data is lost.

Because sticky connections restrict the Dispatcher's ability to optimize requests, you should use this approach only when necessary. You can specify the folder that contains the "sticky" documents, thus ensuring all documents in that folder are processed on the same instance for a user.

Can I use sticky connections and caching in tandem?

For most pages that use sticky connections, you should turn off caching. Otherwise, the same instance of the page is displayed to all users, regardless of the session content.

For some applications, it can be possible to use both sticky connections and caching. For example, if you display a form that writes data to a session, you can use sticky connections and caching in tandem.

Can a Dispatcher and an AEM Publish instance reside on the same physical machine?

Yes, if the machine is sufficiently powerful. However, it is recommended that you set up the Dispatcher and the AEM Publish instance on different machines.

Usually, the Publish instance resides inside the firewall and the the Dispatcher resides in the DMZ. If you decide to have both the Publish instance and Dispatcher on the same physical machine, ensure that the firewall settings prohibit direct access to the Publish instance from external networks.

Can I cache only files with specific extensions?

Yes. For example, if you want to cache only GIF files, specify \*.gif in the cache section of the dispatcher.any configuration file.

How do I delete files from the cache?

You can delete files from the cache by using an HTTP request. When the HTTP request is received, Dispatcher deletes the files from the cache. Dispatcher caches the files again only when it receives a client request for the page. Deleting cached files in this manner is appropriate for web sites that are not likely to receive simultaneous requests for the same page.

The HTTP request has the following syntax:

POST /dispatcher/invalidate.cache HTTP/1.1

CQ-Action: Activate

CQ-Handle: path-pattern

Content-Length: 0

Dispatcher deletes the cached files and folders that have names that match the value of the CQ-Handle header. For example, a CQ-Handle of /content/geomtrixx-outdoors/en matches the following items:

All files (of any file extension) named en in the geometrixx-outdoors directory Any directory named \_jcr\_content below the en directory (which, if it exists, contains cached renderings of sub-nodes of the page) The directory en will only be deleted if the CQ-Action is Delete or Deactivate.

For further details on this topic, see [Manually Invalidating the Dispatcher Cache](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/page-invalidate.html).

How do I implement permission-sensitive caching?

See the [Caching Secure Content](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/permissions-cache.html) page.

How do I secure communications between the Dispatcher and CQ instances?

See the [Dispatcher Security Checklist](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/getting-started/security-checklist.html) and the [AEM Security Checklist](https://helpx.adobe.com/experience-manager/6-4/sites/administering/using/security-checklist.html) pages.

Dispatcher issue jcr:content changed to jcr%3acontent

**Question**: We have recently faced a problem at dispatcher level wherein one of the ajax call which was getting some data form CQ repository had jcr:content in it and that got encoded to jcr%3acontent resulting in wrong result set.

**Answer**: Please use ResourceResolver.map() method to get a 'Friendly' URL to be used / issued get requests from and also to solve the caching issue with Dispatcher. The map() method encodes the : colon to underscores and the resolve() method decodes them back to SLING JCR readable format.You need to use the map() method to generate the URL that is used in the Ajax call.

Further read: <https://sling.apache.org/documentation/the-sling-engine/mappings-for-resource-resolution.html#namespace-mangling>

Flush the Dispatcher

How do I configure Dispatcher flush agents on a Publish instance?

See the [Replication](https://helpx.adobe.com/content/help/en/experience-manager/6-4/sites/deploying/using/replication.html#ConfiguringyourReplicationAgents) page.

How do I troubleshoot Dispatcher flushing issues?

[Refer to this troubleshooting article](https://helpx.adobe.com/content/help/en/experience-manager/kb/troubleshooting-dispatcher-flushing-issues.html) that answers the following questions:

* How should I debug a situation where no content is getting saved in the Dispatcher cache?
* How do I debug a situation where cache files are not getting updated?
* How do I debug a situation where nothing related to Dispatcher flushing is working?

If Delete operations are causing the Dispatcher to flush, [use the workaround in this community blog post by Sensei Martin](https://mkalugin-cq.blogspot.in/2012/04/i-have-been-working-on-following.html).

How do I flush DAM assets from the Dispatcher cache?

You can use the "chain replication" feature. With this feature enabled, the dispatcher flush agent sends a flush request when a replication is received from author.

To enable it:

1. [Follow the steps here](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/page-invalidate.html#invalidating-dispatcher-cache-from-a-publishing-instance) to create flushing agents on publish
2. Go to each of those agent's configuration and on the **Triggers** tab check the **On Receive** box.

Miscellaneous

How does the Dispatcher determine if a document is up-to-date? To determine whether a document is up-to-date, the Dispatcher performs these actions:

It checks whether the document is subject to auto-invalidation. If not, the document is considered up-to-date. If the document is configured for auto-invalidation, the Dispatcher checks whether it is older or newer than the last change available. If it is older, the Dispatcher requests the current version from the AEM instance and replaces the version in the cache.

How does the Dispatcher return documents?

You can define whether the Dispatcher caches a document by using the [Dispatcher configuration](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html) file, dispatcher.any. The Dispatcher checks the request against the list of cacheable documents. If the document is not in this list, the Dispatcher requests the document from the AEM instance.

The /rules property controls which documents are cached according to the document path. Regardless of the /rules property, Dispatcher never caches a document in the following circumstances:

* If the request URI contains a question mark (?).
* This usually indicates a dynamic page, such as a search result that does not need to be cached.
* The file extension is missing.
* The web server needs the extension to determine the document type (the MIME-type).
* The authentication header is set (this can be configured)
* If the AEM instance responds with the following headers:
  + no-cache
  + no-store
  + must-revalidate

The Dispatcher stores cached files on the web server as if they were part of a static website. If a user requests a cached document, the Dispatcher checks whether the document exists in the web server's file system. If so, the Dispatcher returns the documents. If not, the Dispatcher requests the document from the AEM instance.

The GET or HEAD (for the HTTP header) methods are cacheable by the Dispatcher. For additional information on response header caching, see the [Caching HTTP Response Headers](https://docs.adobe.com/content/help/en/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html#caching-http-response-headers) section.

Can I implement multiple Dispatchers in a setup?

Yes. In such cases, ensure that both the Dispatchers can access the AEM website directly. A Dispatcher cannot handle requests coming from another Dispatcher.