Cloud Application and Network Security

Contents

Cache Settings

Define content caching policies and caching rules for your website.

Learn more: Caching Duration

Note: After making changes to cache settings, you may want to manually clear the cache. For details, see Purge the cache below.

Where do I find it?

Log into your my.imperva.com account.

1. On the top menu bar, click **Application**.
2. On the sidebar, click **Websites** and click a website name.
3. On the sidebar, click **CDN > Cache**.

Permissions: The **Edit cache settings** permission is required for modifying the cache settings. By default, the account admin user can edit these settings, and can add the required permission to roles assigned to additional users.

Purge the cache

The following options are available under **Operations**:

Purge Cache	Purge the entire cache. After a major change to your website, such as a version update, you may want to clear all resources in the cache immediately, without waiting for the caching period to expire. Note: You can automate the purge cache action using the API. For details, see the Site Management API here: Cloud Application Security v1/v3 API Definition.
Purge Specific Resource	Purge a subset of the site's cached resources. You can purge resources that match a specified tag or URL. **Tag:** You can purge resources according to tag names. Resources can be tagged in the following ways: • According to a response header value in the origin resources. For details, see the **Tag the**

	Response According to the Value of this Header option under Response. • Using a custom cache rule. For details, see Create custom cache rules. You can select custom tags from the drop-down list or enter tags manually. The list includes tags that were defined by the **Create Tag** and **Enrich Cache Key** cache rules. The list does not contain tags set by the origin. To specify multiple tags, enter a comma-separated list. Only resources that include all specified tags will be purged. **URL:** You can purge all resources that match a specified string. • Enter a complete URL, such as /intro/settings/logo.png • For one of the other rules, enter a specific URL string that will be matched as a prefix, infix, or suffix of the URL, depending on the rule.

Note: Imperva Audit Trail tracks and displays the **Cache purged** and **Specific resources purged** audit events. For more details, see Audit Trail.

Debug caching

The **XRAY Access URL**, located under **Operations**, enables you to view specialized response headers for a single browser session.

Gain visibility into Imperva CDN and caching behavior for your sites using Imperva XRAY debug headers. Troubleshoot inconsistencies in displayed site content.

For details, see XRAY Debug Headers.

Set cache mode

Configure the overall caching policy for your website.

Note: We remove some non-essential HTTP response headers from your resources before storing them in our cache. If there are specific response headers that you want sent to clients, you can specify that they should be cached along with the resource using the **Cache Response Headers** option. For details, see Response.

The following options are available under **Cache Mode**:

No caching	Turns off all caching for the website. Any existing custom cache rules are ignored.
Custom caching	Cache according to custom cache rules only. For details, see Create custom cache rules. **Custom HTTPS caching:** Enables caching of HTTPS resources according to your custom cache rules. If there are no custom cache rules defined for the site, no caching is performed and all requested content is retrieved from your origin web server. **Note:** If you switch from another cache mode to **Custom caching**, some options will be reset to "off" and will be unavailable while caching is disabled.
Standard caching	Cache according to standard HTTP headers. Only content that was marked by the site's developer / web server as static using standard HTTP headers is cached. **Standard HTTPS caching:** (Available for SSL sites) Enables HTTPS caching of images, css files, JS files, and resources defined with the **Cache-Control: public** header, according to standard HTTP headers. • **Include HTML resources.** When selecting this option, you can introduce the risk of returning HTML resources that contain personal information, such as PII, ePHI, and PAN data. • **Include all resource types.** Enables HTTPS caching for all resource types, including HTML resources.
Smart caching	Also profile dynamic pages to identify and cache static content that was not marked as static. In addition to content that was marked by the site's developer / web server as static using standard HTTP headers, Imperva also profiles other resources to identify and cache static content that was not marked as such. The time period (in minutes, hours, days or weeks) that you set

	for this option determines how often the cache is refreshed. **Smart HTTPS caching:** (Available for SSL sites) Enables HTTPS caching of images, css files, JS files, and resources defined with the **Cache-Control: public** header, according to standard HTTP headers and Imperva's smart profiling. • **Include HTML resources.** When selecting this option, you can introduce the risk of returning HTML resources that contain personal information, such as PII, ePHI, and PAN data. • **Include all resource types.** Enables HTTPS caching for all resource types, including HTML resources.
Cache all	Cache every resource on the web server for the specified amount of time. All site content is cached. The time period (in minutes, hours, days or weeks) that you set for this option determines how often the cache is refreshed (TTL). **Cache all HTTPS resources (including HTML):** (Available for SSL sites) Enables caching of all HTTPS resources, including HTML resources. When selecting this option, you can introduce the risk of returning HTML resources that contain personal information, such as PII, ePHI, and PAN data.

Note:

- In addition to caching according to the selected mode, content is also cached as specified by any custom cache rules that are defined for your site. For details, see Create custom cache rules.
- Resources that include explicit caching directives against caching, as defined in the resources themselves using the **Cache-Control** or **Expires** HTTP headers, are not cached.

Create custom cache rules

Custom cache rules let you define specific exceptions to the caching rules that are set by the overall Cache Mode rules described above. You can define conditions for when and if specific resources should be cached.

Note: Cached objects are unaffected by newly added or modified rules. To fully apply a new or edited rule, purge the cache.

In the Custom Cache Rules section, click **Add Rule**.

Field/Option	Description
Rule Filter	Define a filter to determine when the rule is applied. The filter defines the conditions that trigger the rule action. If left empty, the rule is always run. For details, see Define the rule filter.
Rule Action	Define the action you want Imperva to take for every request that matches the rule. For details, see Select an action.
Rule Name	Give the rule a meaningful name. **Note:** The **Rule Name** may not contain special characters. Only alphanumeric, space, period, comma, colon, hyphen, and underscore characters are allowed.
Enable Rule	Enable or disable the rule.

Define the rule filter

Define a filter for the rule using predefined parameters.

The following rule filter parameters are available for cache rules:

- Client ID
- Cookie Exists
- Cookie Value
- Header Exists
- Header Value
- Param Exists
- Param Value
- URL

- User-Agent

For details on the parameters, see Rule Filter Parameters.

Matched object	Under **Apply This Rule If**, select the part of the request or the sessions to which the filter is applied. For example, Client IP or Country. For full details on the available parameters, see Rule Filter Parameters.
Operator	Defines how the filter value is matched. Most filter parameters support only a subset of the list of operators. For example, the **QueryString** filter parameter supports only the 'equal to' and 'not equal to' operators. When a filter parameter is selected (see **Matched object** above), only the supported operators will be displayed in the operator field. For the full list of filter parameters and the supported operators for each, see Rule Filter Parameters.
Value	The value to be matched.
Editor	When you define a filter and click **+Add**, the filter syntax is added to the Editor. You can add as many filters as required. The filters are added to the rule syntax using the AND logic. For information about combining filters using the OR logic, refer to the Syntax Guide. Alternatively, you can add filters directly using the native syntax. Every rule is checked for syntax validity before it is saved. For details, see the Syntax Guide.
Validate	Verifies the rule syntax. Validation is also performed automatically whenever you save a rule.

Select an action

Define the action you want to take for every request that matches the rule.

Action	Description
Cache Resource	Always cache the resource. Can be used with the following filter parameters only:

Action	Description
	Param Exists, **Param Value**, **URL**
Cache Resource on Client	Cache the resource on the client.
Don't Cache Resource	Never cache the resource.
Create Tag	Tag the resources that match the rule conditions. This enables you to subsequently purge those resources according to the tag name. Tag names can include the following characters only: alphanumeric (a-z, A-Z, 0-9), &,',^,-,$,!,`,#,%,.,+,~,_,| Spaces are not allowed. Note: Tags are added when the resource is cached. If you add or modify tags, purge the resource to enable retagging. For more details, see Purge the cache.
Differentiate Cache Key by...	By default, we create the cache key according to the requested URL. You can choose to create different cache keys based on protocol (http/s), header, cookie, or geolocation, so that the matching resources are cached as different resources. • **HTTP/HTTPS Scheme:** A resource is cached separately depending on whether it is accessed over HTTP or HTTPS. • **Header:** Specify a header name. • **Cookie:** Specify a cookie name. • **Geolocation:** Resources are cached separately based on geolocation of the request. Add locations to country code groups using standard 2-letter country codes. A resource is cached separately for each country code group, plus an additional entry for all other country codes. To add multiple geolocations to a country code group, enter a comma separated list, such as CN,CO,US.

Action	Description
Ignore Parameters in Cache Key	If the parameters do not affect which resource is returned, you can choose to ignore them. Add the name of a specific parameter, or select **Ignore All Parameters**.
Enrich Cache Key	Add text to enhance the cache key. A new, enriched cache key is then calculated using the additional input.
Cache Authenticated Resources	The **Authorization** request header contains credentials to authenticate user. By default, if this header is present, we do not return cached content and the request is forwarded to the origin server. Selecting this option returns cached content if available without authenticating the client.
Force Resource Validation	Validate with the origin server that the resource has not changed before returning the cached resource to the client. This might be used for the purpose of client authentication. If the client sends the Authorization header and this rule is triggered, Imperva forwards the request to your origin server and you can then verify client authorization and return the result as expected.
Serve Stale Content	Expired resources are returned from cache and refreshed asynchronously in the background. If the origin server cannot be reached to refresh the cache, stale content continues to be displayed for the amount of time specified for TTL. **Note:** When there is at least one enabled custom rule using this action, the global **Serve Stale Content** option is automatically enabled if it was not already enabled, but it does not apply globally and cannot be modified. The custom rules override the global setting. For more details, see the Serve Stale Content option below.

Precedence among caching rules

Because there may possibly be conflicts between the Cache Rules you define and the logic dictated by the Caching Mode, the following order of precedence is applied to the various types of rules:

1. **Don't Cache Resource** Cache Rules
2. **Cache Resource** Cache Rules
3. Caching Mode
4. Cache directives sent by the web server (in HTTP headers)

Note: If there is a conflict between the caching durations defined in the Caching Mode and in the Advanced Caching Rules, the longer period of the two is applied. The same goes for a conflict between the caching rule of a certain resource and one of its sub-resources.

Set advanced cache settings

The following options let you control HTTP features that may interfere with Imperva's caching behavior, thereby reducing performance. These are triggered by HTTP request and response headers that instruct the web server or client not to cache certain content, or by the browser's behavior.

These caching options are often enabled on web servers or browsers due to misconfiguration, so the default behavior is to ignore them. You can change this behavior using the settings below, located under **Advanced Settings**.

Cache key

Use the Same Cache for Full and Naked Domains	For example, use the same cached resource for www.example.com/a and example.com/a.
Comply with Vary	Cache resources in accordance with the **Vary** response header. Only resources with the Vary value 'Accept-Encoding' are cached. **Vary** is an HTTP response header that indicates how to match future request headers to decide if a cached response can be used. When this option is disabled, the **Vary** header is ignored.

Response

Cache Response Headers	**Cache All Headers:** By default, response headers are not cached. When the **Cache Response Headers** option is selected, all headers in the responses are cached.

	Custom: Specify which response headers should be cached along with the resource. Examples of commonly used headers: • Access-Control-Allow-Origin • Access-Control-Allow-Methods • Access-Control-Allow-Headers • Access-Control-Request-Method • Access-Control-Request-Headers If the **Custom** option is selected and no headers are specified, headers will not be cached.
Cache Shield	Adds an intermediate cache between other Imperva PoPs and your origin servers to protect your servers from redundant requests. For details, see Cache Shield.
Tag the Response According to the Value of this Header	Specify which origin response header contains the cache tags in your resources. This enables you to subsequently purge those resources according to the tag name. For details, see Purge the cache. If the specified header includes multiple tags separated by commas, the resources is tagged with multiple tags. **For example:** Header Name: Cache-Tag Header Value: “tag1,tag2,tag3” Tagging Result: The resource is tagged with 3 different tags - “tag1”, “tag2”, and “tag3”
Cache Empty Responses	Cache responses that don’t have a message body. By default, files that don’t have a message body are not cached (where content length = 0) . When the Cache Empty Responses option is selected, these resources are cached.

Cache 3xx Responses	When this option is checked Imperva will cache 301, 302, 303, 307, and 308 redirect response headers containing the target URI. • 301 and 308 response headers are cached for 2 hours. • The length of time that 302, 303, and 307 response headers are cached for is based on Cache-Control headers, according to our standard caching mechanism. For details, see Caching Duration.
Cache 404 Responses	Cache responses of unavailable resources for a specified amount of time.
Cache HTTP 1.0 responses	Cache HTTP 1.0 type responses that don't include the Content-Length header or chunking. By default, responses in HTTP 1.0 format are not cached. HTTP 1.0 sends a response and then closes the connection to indicate that it is finished. Because the connection may have been closed for other reasons, we cannot determine if the whole response has been received.
Serve Stale Content	Expired resources are returned from cache and refreshed asynchronously in the background. Stale content refers to cached resources whose TTL has expired. The first request after the cache period expires causes the resource to be retrieved from cache. The current cached version (stale) of the resource is displayed without delaying the response, while the cache is refreshed in the background. This option can also be useful when the origin server is temporarily unavailable and the cache cannot be refreshed. Stale content is served instead of displaying an error to end users, according to one of the following options: • **Adaptive:** Stale content is served for a duration of 2 to 24 hours based on the time passed since the resource was last updated. Frequent updates result in a shorter stale period. Infrequent updates result in a longer stale period. • **Custom:** Serve stale content for the specified amount of time.

TTL

Use Shortest Caching Duration in Case of Conflicts	By default, the longest duration is used in case of conflict between caching rules or modes. When this option is selected, Imperva uses the shortest duration in case of conflict.
Prefer 'Last Modified' over Etag	When this option is selected, Imperva prefers using Last Modified values (if available) over eTag values (recommended on multi-server setups).

Client-side caching

Enable Client-Side Caching	Cache content on client browsers or applications. When not enabled, content is cached only on the Imperva proxies.
Comply with No-Cache and Max-Age Directives in Client Requests	By default, these cache directives are ignored. Resources are dynamically profiled and re-configured to optimize performance. **no-cache** is a directive in the HTTP **Cache-Control** request header that instructs web proxies not to return cached content without first checking with the server to see if the content has changed. **max-age** is a directive in the HTTP **Cache-Control** request header that instructs web proxies not to return content that is over a certain expiration age.
Send Age Header	Send **Cache-Control: max-age** and **Age** headers. By default, in the **Cache-control: max-age** header, we send the value of: <max-age from origin> minus <age (time in cache)> and do not send the **Age** header. If you enable the Send Age Header option, we send the full **Cache-control: max-age** value from origin and also the **Age** header.

Cache Settings API

Define content caching policies and caching rules for your websites using the API.

For instructions on using the Cache Settings API, see Cache Settings API Definition.

The definition file presents a full, formatted, and interactive version of the APIs that you can use to learn about the APIs, or test them using your API ID and key. You can also download the definition file.

Last updated: 2024-10-20