# Siteaccess Matching



Siteaccess matching is done through **eZ\Publish\MVC\SiteAccess\Matcher** objects. You can configure this matching and even develop custom matchers.

## Configuration

You can configure siteaccess matching in your main **ezpublish/config/ezpublish.yml** :

**ezpublish.yml**

```
ezpublish:
    siteaccess:
        default_siteaccess: ezdemo_site
        list:
            - ezdemo_site
            - eng
            - fre
            - fr_eng
            - ezdemo_site_admin
        groups:
            ezdemo_site_group:
                - ezdemo_site
                - eng
                - fre
                - fr_eng
                - ezdemo_site_admin
        match:
            Map\URI:
                ezdemo_site: ezdemo_site
                fre: fre
                ezdemo_site_admin: ezdemo_site_admin
```

You need to set several parameters:

- **ezpublish.siteaccess.default_siteaccess**
- **ezpublish.siteaccess.list**
- *(optional)* **ezpublish.siteaccess.groups**
- **ezpublish.siteaccess.match**

**ezpublish.siteaccess.default_siteaccess** is the default siteaccess that will be used if matching was not successful. This ensures that a siteaccess is always defined.

**ezpublish.siteaccess.list** is the list of all available siteaccesses in your website.

*(optional)* **ezpublish.siteaccess.groups** defines which groups siteaccesses are member of. This is useful when you want to mutualize settings between several siteaccesses and avoid config duplication. Siteaccess groups are considered as regular siteaccesses as far as configuration is concerned.

A siteaccess can be part of several groups.

A siteaccess configuration has always precedence on the group configuration.

**ezpublish.siteaccess.match** holds the matching configuration. It consists in a hash where the key is the name of the matcher class. If the matcher class doesn't start with a ****, it will be considered relative to `eZ\Publish\MVC\SiteAccess\Matcher` (e.g. `Map\Host` will refer to `eZ\Publish\MVC\SiteAccess\Matcher\Map\Host`)

Every **custom matcher** can be specified with a **fully qualified class name** (e.g. `\My\SiteAccess\Matcher`) or by a **service identifier prefixed by @** (e.g. `@my_matcher_service`).

- In the case of a fully qualified class name, the matching configuration will be passed in the constructor.
- In the case of a service, it must implement `eZ\Bundle\EzPublishCoreBundle\SiteAccess\Matcher`. The matching configuration will be passed to `setMatchingConfiguration()`.

## Available matchers

| Name | Description | Configuration | Example |
|---|---|---|---|
| `URIElement` | Maps a URI element to a siteaccess.<br>This is the default matcher used when choosing<br>URI matching in setup wizard. | The element number you want to match (starting from 1).<br>`ezpublish:`<br>`siteaccess:`<br>`match:`<br>`URIElement: 1`<br>**Important:** When using a value > 1,<br>it will concatenate the elements with _ | **URI:** `/ezdemo_site/foo/bar`<br>Element number: 1<br>Matched siteaccess: ezdemo_site<br>Element number: 2<br>Matched siteaccess: ezdemo_site_foo |
| `URIText` | Matches URI using *pre* and/or *post* sub-strings<br>in the first URI segment | The prefix and/or suffix (none are required)<br>`ezpublish:`<br>`siteaccess:`<br>`match:`<br>`URIText:`<br>`prefix: foo`<br>`suffix: bar` | **URI:** `/footestbar/my/content`<br>Prefix: foo<br>Suffix: bar<br>Matched siteaccess: test |

| | | | |
|---|---|---|---|
| `HostElement` | Maps an element in the host name to a siteaccess. | The element number you want to match (starting from 1).<br>`ezpublish:`<br>`siteaccess:`<br>`match:`<br>`HostElement: 2` | **Host name:** `www.example.com`<br>Element number: 2<br>Matched siteaccess: example |
| `HostText` | Matches a siteaccess in the host name,<br>using *pre* and/or *post* sub-strings<br>. | The prefix and/or suffix (none are required)<br>`ezpublish:`<br>`siteaccess:`<br>`match:`<br>`HostText:`<br>`prefix: www.`<br>`suffix: .com` | **Host name**: `www.foo.com`<br>Prefix: www.<br>Suffix: .com<br>Matched siteaccess: foo |
| `Map\Host` | Maps a host name to a siteaccess. | A hash map of host/siteaccess<br>`ezpublish:`<br>`siteaccess:`<br>`match:`<br>`Map\Host:`<br>`www.foo.com: foo_front`<br>`adm.foo.com: foo_admin`<br>`www.stuff.fr: bar_front`<br>`adm.stuff.fr: bar_admin` | **Map**:<br>• www.foo.com => foo_front<br>• admin.foo.com => foo_admin<br>**Host name**: www.example.com<br>Matched siteaccess: foo_front |

| | | | |
|---|---|---|---|
| `Map\URI` | Maps a URI to a siteaccess | A hash map of URI/siteaccess<br>`ezpublish:`<br>`siteaccess:`<br>`match:`<br>`Map\URI:`<br>`something: ezdemo_site`<br>`foobar: ezdemo_site_admin` | **URI:** `/something/my/content`<br>Map:<br>• something => ezdemo_site<br>• foobar => ezdemo_site_admin<br>Matched siteaccess: ezdemo_site |
| `Map\Port` | Maps a port to a siteaccess | A has map of Port/siteaccess<br>`ezpublish:`<br>`siteaccess:`<br>`match:`<br>`Match\Port:`<br>`80: foo`<br>`8080: bar` | **URL:** `http://ezpublish.dev:8080/my/content`<br>Map:<br>• 80: foo<br>• 8080: bar<br>Matched siteaccess: bar |
| `Regex\Host` | Matches against a regexp and extract a portion of it | The regexp to match against and the captured element to use<br>`ezpublish:`<br>`siteaccess:`<br>`match:`<br>`Regex\Host:`<br>`regex: "^(\\w+_sa)$"`<br>`# Default is 1`<br>`itemNumber: 1` | **Host name:** `example_sa`<br>regex: `^(\\w+)_sa$`<br>itemNumber: 1<br>Matched siteaccess: example |

| | | | |
|---|---|---|---|
| `Regex\URI` | Matches against a regexp and extract a portion of it | The regexp to match against and the captured element to use<br><br>```<br>ezpublish:<br><br>siteaccess:<br><br>match:<br><br>Regex\URI:<br><br>regex:<br>"^/foo(\\w+)b<br>ar"<br><br># Default is<br>1<br><br>itemNumber: 1<br>``` | **URI:** `/footestbar/somethin`<br>`g`<br><br>regex: ^/foo(\\w+)bar<br>itemNumber: 1<br><br>Matched siteaccess: test |

## Compound siteaccess matcher

The Compound siteaccess matcher allows to combine several matchers together, which is performed using the legacy ***host_uri***.

For that, the following matchers can be used in your siteaccess match:

- `Compound\LogicalAnd`
- `Compound\LogicalOr`

**ezpublish.yml**

```
ezpublish:
    siteaccess:
        default_siteaccess: ezdemo_site
        list:
            - ezdemo_site
            - eng
            - fre
            - fr_eng
            - ezdemo_site_admin
        groups:
            ezdemo_site_group:
                - ezdemo_site
                - eng
                - fre
                - fr_eng
                - ezdemo_site_admin
        match:
            Map\URI:
                ezdemo_site: ezdemo_site
                fre: fre
                ezdemo_site_admin: ezdemo_site_admin
            # The following mimics host_uri for 2 different configuration.
            Compound\LogicalAnd:
                my_compound_rule_1:
                    # Nested matchers, with their configuration.
                    # No need to precise their matching values (true will suffice).
                    matchers:
                        Map\URI:
                            the_front: true
                        Map\Host:
                            ezpublish.dev: true
                    # The siteaccess to match in the end
                    match: ezdemo_site
                my_compound_rule_2:
                    matchers:
                        Map\URI:
                            the_admin: true
                        Map\Host:
                            ezpublish.dev: true
                    match: ezdemo_site_admin
            # Matching hosts as host: siteaccess
            Map\Host:
                ezpublish.dev: ezdemo_site
                admin.ezpublish.dev: ezdemo_site_admin
```

## Matching by request header

It is possible to define which siteaccess to use by setting a **X-Siteaccess** header in your request. This can be useful for REST requests.

In such case, **X-Siteaccess** must be the **siteaccess name** (e.g. *ezdemo_site*).

## Matching by environment variable

It is also possible to define which siteaccess to use directly via an **EZPUBLISH_SITEACCESS** environment variable.

This is recommended if you want to get **performance gain** since no matching logic is done in this case.

You can define this environment variable directly from your web server configuration:

**Apache VirtualHost example**

```
# This configuration assumes that mod_env is activated
<VirtualHost *:80>
    DocumentRoot "/path/to/ezpublish5/web/folder"
    ServerName example.com
    ServerAlias www.example.com
    SetEnv EZPUBLISH_SITEACCESS ezdemo_site
</VirtualHost>
```

This can also be done via PHP-FPM configuration file, if you use it. See PHP-FPM documentation for more information.

**Note about precedence**
The precedence order for siteaccess matching is the following (the first matched wins):

1. Request header
2. Environment variable
3. Configured matchers

## URILexer and semanticPathinfo

In some cases, after matching a siteaccess, it is neecessary to modify the original request URI. This is for example needed with URI-based matchers since the siteaccess is contained in the original URI and it is not part of the route itself.

The problem is addressed by *analyzing* this URI and by modifying it when needed through the **URILexer** interface.

**URILexer interface**

```
/**
 * Interface for SiteAccess matchers that need to alter the URI after matching.
 * This is useful when you have the siteaccess in the URI like
"/<siteaccessName>/my/awesome/uri"
 */
interface URILexer
{
    /**
     * Analyses $uri and removes the siteaccess part, if needed.
     *
     * @param string $uri The original URI
     * @return string The modified URI
     */
    public function analyseURI( $uri );
    /**
     * Analyses $linkUri when generating a link to a route, in order to have the
siteaccess part back in the URI.
     *
     * @param string $linkUri
     * @return string The modified link URI
     */
    public function analyseLink( $linkUri );
}
```

Once modified, the URI is stored in the ***semanticPathinfo*** request attribute, and the original pathinfo is not modified.