App-Arena.com App-Manager API Documentation

Release 1.0

App-Arena.com | Sebastian Buckpesch

April 11, 2017

Contents

App-Arena.com provides an infrastructure to manage and scale web-apps. Developers can publish and sell own web-apps through our sales channels. Sales partners can integrate our Web-App-CMS in their third-party software solutions and sell all published apps of the platform under their name and brand.

The code of this documentation is open source, and available on github. We appreciate every contribution to make this documentation better.

The main documentation for the site is organized into a couple sections:

- *API*
- *SDK*

CHAPTER 1

API

API - Getting started

API key

To request restricted information from the API, you need to add an API key or a JSON Web Token (JWT) to your request header. Read more about it in Authorization

If you do not have access to the developer section yet, please drop us an email to s.buckpesch at app-arena.com with your contact data and will we get in touch with you and sent you an API key.

API Endpoint

All API URLs listed in this documentation are relative to `https://my.app-arena.com/api/v2/`.

HTTP Verbs

The App-Arena API is a RESTful API. All requests can be made using one of the following HTTP verbs

	Valid for	Description
GET	Entity, Collection	Request a single entity or a whole collection
POST	Collection	Creates a new entity in a collection
PUT	Entity	Updates an entity
DELETE	Entity	Deletes an entity

Passing Request Data

Request data is passed to the API by Posting JSON objects to the API endpoints with the appropriate parameters. The documentation for each API call will contain more detail on the parameters accepted by the call.

Additionally, the requests can be manipulated by query parameters. The first query must be preceded by a '?' while the following queries have to be separated by a '&' character.

Example:

http://route.to.api/collection/entity?query1=xxx&query2=yyy&query3=...

Note: You can find the available queries for each call in their respective section.

Available query options:

Query	Valid for	Description
lang	GET,PUT,DELETE	points the request to the desired language
fields	GET	receive only desired fields in the response, list fields comma separated example: fields=appId,expiryDate,name
exclude	GET	exclude fields from the response
type	GET	receive only media items of type image, audio or video
orderBy	GET	order the response items syntax: orderBy[dir]=target dir can be 'ASC' or 'DESC' (without ''), target is the field to order e.g.: orderBy[DESC]=appId
page	GET	sets the page of paginated results
items	GET	sets the amount of items of paginated results
version	GET, POST, PUT, DELETE	
filter	GET	filter the results, explanation in the Filter section below.
rel	GET	receive additional related information in a single request, explanation in the Relation section below

Note: Only collection requests on 'Apps', 'Templates', 'Projects', 'Versions', 'Companies' and 'Customers' have default pagination values. Elsewhere, when a query parameter is not defined, no action is performed.

The Filter Query Parameter

Requests can be refined with a filter query to shape the output exactly to what you want to receive. In combination with the pagination query parameters, the filter function make a great search tool for the data you want to display.

Note: The filter is applicable for all GET requests of collections and can be combined freely with the other available query parameter options.

The filter distinguishes between the different field types and allows only certain operator types for each type. You can find the types of the fields in the respective section of the calls. The field types in use are:

`bool` operators: [eq], [neq]

`integer` operators: [eq], [neq],[gt], [gte], [lt], [lte]

`text string`

operators: [eq], [neq], [match], [not]

`datetime` in the format Y-m-D (Year-Month-Day) operators: [eq], [neq], [gt], [gte], [lt], [lte]

Note: The filter function is case insensitive.

A field of an entity can be targeted for filtering with one of the following operators:

[eq] equals : receive a collection of entities where the target field value equals the submitted value applicable for all field types, in case of type `datetime` the filter returns all entities that match the day regardless of the time of the day

[neq] equals not : receive a collection of entities where the target field value does not equal the submitted value applicable for all field types, in case of type `datetime` the filter returns all entities that do not match the day regardless of the time of the day

[match] matches : receive a collection of entities where the target field value matches the submitted value partially e.g. : the match value 'test' for the target field 'name' returns all entities where the

'name' field contains the string 'test' independent of the location of the occurrence in the string like 'testApp', 'Apptest' or 'appTESTapp' applicable only for the field types 'string'

[not] matches not: receive a collection of entities where the target field value does not contain the submitted value applicable only for the field types 'string'

[gt] greater than : receive a collection of entities where the target field value is greater than the submitted value applicable for field types 'integer' and 'datetime'

[gte] greater than or equal: receive the same as [gt] inclusive the submitted value applicable for field types 'integer' and 'datetime'

[lt] lower than : receive a collection of entities where the target field value is lower than the submitted value applicable for field types 'integer' and 'datetime'

[lte] lower than or equal: receive the same as [gt] inclusive the submitted value applicable for field types 'integer' and 'datetime'

Syntax:

GET /{collection}?filter.{target}[{operator}]={value}

{collection} The {collection} is the route to the target collection, if we wanted to receive apps it would be just 'apps', for the config entities of that app it would be 'apps/:appId/configs'.

filter. The 'filter.' keyword at the beginning of the query parameter is mandatory, indicating the filter intention to the API.

{target} The {target} defines the field which is to be filtered. Find the available fields in the corresponding section of the call.

{operator} In the brackets follows the {operator} which is defining the mode of the filter (see operators list above).

{value} The {value} is mandatory and needs to follow a '=' character. Just put here the plain value without any '' or ', no matter integer or string.

Examples:

1.) If we wanted to get apps which are not yet expired and will not expire today, the request would look like this (on the 25th of november in 2016, the date this was written):

GET /apps?filter.expiryDate[gt]=2016-11-25

2.) If we wanted to get all the config entities of app '1' where the 'type' field is 'input' the request would be

GET /apps/1/configs?filter.type[eq]=input

Combine these request with pagination, order- and field filters to make precise and lightweight requests!

The Relation Query Parameter

As you can see in Chapter Data Structure, there are quite a few connections between different entities. Every App is owned by a company, connected to a Template while having multiple configs, translations, infos, etc... . Sometimes the information stored in these connected resources is needed because they contain vital information for element description and presentation. To gather these details, multiple API calls are necessary. To avoid excessive requesting, the relation query parameter got introduced.

With this request enhancement it is possible to gather any connected information in a single request for an entity. When fetching a collection this underlies a restriction.

General Usage

The syntax for this enhancement is quite simple: just add the query parameter 'rel' to the request with a comma separated list of desired relations as its value.

GET /{entity/collection}?rel={relation1},{relation2}, ...

{entity/collection} Request a single entity or a collection of entities. Valid targets are: 'apps', 'templates', 'projects', 'projects/:projectId/versions', 'companies' and 'customers'

rel= Mandatory keyword to indicate the relation intention to the API.

{relationX} The relation identifier. You can find the valid relations in the corresponding section of the call.

Note: While you can combine this query parameter as usual with the other query modifiers, the field filter influences only the base entities while the relations will always

---

be outputted completely. They will however never contain sensitive information like passwords when fetching user relations.

Entity Relations

There are no restrictions when fetching an entity. Add as much relations to the request as you want, the response format will look somewhat like this:

```
{
    "_embedded": {
        "data": {
            "entityProperty1": "value1",
            "entityProperty2": "value2",
                .
                .
                .
            "relation1": {                              //relation entity
                "relationProperty1_1": "relValue1_1",
                "relationProperty1_1": "relValue1_2",
                    .
                    .
                    .
            },
            "relation2": {                              //relation collection
                "relationIdentifier2_1": {
                    "relationProperty2_1": "relValue2_1",
                    "relationProperty2_1": "relValue2_1",
                        .
                        .
                        .
                },
                "relationIdentifier2_2": {
                    .
                    .
                    .
                },
            },
```

```
                "relation3": {
                    .
                    .
                    .
                },
                .
                .
                .
            }
        }
    }
```

You will find the relation data under its identifier key (already used in the request) in the "_embedded" -> "data" object. Relation entities data is in the first level of the relation object while relation collections data is in objects with their identifier as keys.

Warning: The relation query parameter directly modifies the query to the database, therefore you only receive properties directly connected to the base entity. No inheritance of entities from the App/Template/Version chain as you know it from the regular calls on those collections/entities takes place! Relation calls are not intended to substitute the regular calls, but rather to compress multiple requests into a single one to save time and reduce server load.

Collection Relations

When fetching a collection the relation query parameter underlies the restriction that no relation collections may be fetched. A relation collection can easily consist of tens to hundreds of entities which have to be fetched for tens to hundreds of base entities. The result can cause memory issues as well as heavy workload on the server, which is to be avoided. For this reason only relation entities are allowed with the exception of some small collections which are expected to be small enough to avoid these issues. You can find the allowed relations for each call in the corresponding section.

The response format of a collection request with relations looks something like this (exemplified by a request to 'apps'):

Warning: Fetching a lot of entities with relations can be very memory intensive. Even though we designed it the way these issues are unlikely, use collection fetching with entities restrictively and fetch only necessary relations!

Example request body

```
{
  "_links": {
    "next": {
      "href":   "https://my.app-arena.com/api/v2/apps?items=5&page=3"
    },
    "previous": {
      "href":   "https://my.app-arena.com/api/v2/apps?items=5&page=1"
    },
    "self": {
      "href":   "https://my.app-arena.com/api/v2/apps?items=5&page=2"
    }
  },
  "_embedded": {
    "data": {
      "100": {
```

```
    "appId":        100,
    "name":         "example App",
    "lang":         "en_US",
    "activated":    true,
    "expiryDate":   "2017-08-04 00:00:00",
    "companyId":    1,
    "templateId":   10,
    "relation1": {                              //relation entity
        "relationProperty1_1": "relValue1_1",
        "relationProperty1_2": "relValue1_1",
            .
            .
            .
    },
    "relation2": {                              //relation collection
        "relationIdentifier1_1": {
            "relationProperty2_1": "relValue2_1",
            "relationProperty2_2": "relValue2_2",
                .
                .
                .
        },
        "relationIdentifier1_2": {
                .
                .
                .
        },
    },
    "_links": {
      "app": {
        "href":     "https://my.app-arena.com/api/v2/apps/100"
      },
      "language": {
        "href":     "https://my.app-arena.com/api/v2/apps/100/languages/en_US"
      },
      "company": {
        "href":     "https://my.app-arena.com/api/v2/companies/1"
      },
      "template": {
        "href":     "https://my.app-arena.com/api/v2/templates/10"
      }
    }
  },
  "101": {
    "appId": 101,
        .
        .
        .
    "relation1": {                              //relation entity
        "relationProperty1": "relValue1",
        "relationProperty2": "relValue2",
            .
            .
            .
    },
    "relation2": {                              //relation collection
        "relationIdentifier1": {
            "relationProperty1": "relValue1",
```

```
                "relationProperty2": "relValue2",
                    .
                    .
                    .
            },
            "relationIdentifier2": {
                    .
                    .
                    .
            },
        },
        .
        .
        .
    },
    "102": {
      "appId": 102,
          .
          .
          .
    }
    },
    .
    .
    .
  }
},
"total_items": 10511,
"page_size": 5,
"page_count": 2103,
"page_number": 2
}
```

Note: Relation entities/collection do not contain links in the HAL-format style

Example Request

Let's assume that we want to display the App with the Id = 1, including all details of it. That means we need the name of the template as well as the name of the company that owns it. Additionally we want to display the activated languages and wen need some information stored in the info entities. In this case the request would look like this:

GET /apps/1?rel=template,company,languages,infos

The request output would be:

Example request body

```
{
  "_embedded": {
    "data": {
      "appId": 1,
      "name": "App name",
      "lang": "de_DE",
      "activated": true,
      "expiryDate": "2099-04-18 00:00:00",
      "companyId": 1,
```

```
"templateId": 22,
"template": {
  "templateId": 22,
  "name": "Template name",
  "lang": "de_DE",
  "parentId": 22,
  "versionId": 33,
  "companyId": 1,
  "public": true,
  "created": "2016-04-20 13:51:44",
  "updated": "2016-04-20 13:51:44",
  "deletedAt": null,
  "createdFromIp": "0.0.0.0",
  "updatedFromIp": "0.0.0.0",
  "createdBy": "some user",
  "updatedBy": "some user"
},
"company": {
  "companyId": 1,
  "subdomain": "subdomain",
  "name": "App-Arena GmbH",
  "address1": "Moltkestr. 123",
  "address2": "",
  "zip": "50674",
  "city": "Cologne",
  "country": "DE",
  "logo": "https://my.path.to/my/logo.png",
  "color1": "#478AB8",
  "color2": "#2D343D",
  "parentId": 1,
  "storeId": 1,
  "created": "2016-04-20 13:46:50",
  "updated": "2016-10-07 10:32:06",
  "deletedAt": null,
  "createdFromIp": "0.0.0.0",
  "updatedFromIp": "0.0.0.0",
  "createdBy": "some admin",
  "updatedBy": "some admin"
},
"languages": {
  "de_DE": {
    "lang": "de_DE",
    "activated": true,
    "appId": 1,
    "created": "2016-04-20 13:46:50",
    "updated": "2016-10-07 10:32:06",
    "createdFromIp": 0.0.0.0,
    "updatedFromIp": 0.0.0.0,
    "createdBy": "some user",
    "updatedBy": "some user"
  },
  "en_US": {
    "lang": "en_US",
    "activated": true,
    "appId": 1,
    "created": "2016-04-20 13:46:50",
    "updated": "2016-10-07 10:32:06",
    "createdFromIp": 0.0.0.0,
```

```
            "updatedFromIp": 0.0.0.0,
            "createdBy": "some user",
            "updatedBy": "some user"
          }
        },
        "infos": {
          "description": {
            "infoId": "description",
            "lang": "de_DE",
            "revision": 0,
            "value": "a nice description of this app",
            "meta": {},
            "appId": 1,
            "deletedAt": null,
            "createdBy": "some user",
            "createdFromIp": "0.0.0.0",
            "created": "2016-04-20 13:52:52"
          }
          "fb_page_url": {
            "infoId": "fb_page_url",
            "lang": "de_DE",
            "revision": 0,
            "value": "https://www.facebook.com/",
            "meta": {},
            "appId": 1,
            "deletedAt": null,
            "createdBy": "some user",
            "createdFromIp": "0.0.0.0",
            "created": "2016-04-20 13:52:55"
          }
        },
        "_links": {
          "app": {
            "href": "http://manager.local/api/v2/apps/1"
          },
          "language": {
            "href": "http://manager.local/api/v2/apps/1/languages"
          },
          "company": {
            "href": "http://manager.local/api/v2/companies/1"
          },
          "template": {
            "href": "http://manager.local/api/v2/templates/22"
          }
        }
      }
    }
  }
```


Response Formats

The API responds using HTTP Status Codes so that you can see immediately if you request was successful or an error occurred. A complete list of HTTP response formats you can find here: HTTP-Statuscodes

	HTTP-Response on Success
GET	200 (OK)
POST	201 (Created)
PUT	200 (OK)
DELETE	200 (OK)

The JSON output depends on the type of request and the data submitted. GET collection requests will output data in the **HAL-format_**, a format which provides links to the mentioned resources for easy resource browsing. As some of the requests are intended for listing items to the user, these requests will additionally output the data paginated. It comes in chunks of adjustable size for convenient item displaying. PUT and POST requests however output besides a status code the created/updated information without any links to the resources, as this information serves for verification and further processing. DELETE requests will always output a status and a message.

Response examples

GET **request HAL format**

The relevant data can be found in "_embedded" -> "data" and the status code is only submitted via HTTP. The keys of the contained objects are named after their characterizing item for easy processing and representation. This example shows the output of the 'App' 9999 entity GET request.

Example request body


```
{
  "_embedded": {
    "data": {
      "9999": {
        "appId":         9999,
        "name":          "Example App",
        "lang":          "en_US",
        "activated":     false,
        "expiryDate":    "2099-01-01 00:00:00",
        "companyId":     1,
        "templateId":    888,
        "_links": {
          "app": {
            "href":      "https://my.app-arena.com/api/v2/apps/9999"
          },
          "language": {
            "href":      "https://my.app-arena.com/api/v2/apps/9999/languages/en_US"
          },
          "company": {
            "href":      "https://my.app-arena.com/api/v2/companies/1"
          },
          "template": {
            "href":      "https://my.app-arena.com/api/v2/templates/888"
          }
        }
      }
    }
  }
}
```


GET **request HAL format paginated**

Pagination information is added and can be modified by the following queries:

- items : defines the number of objects to be sent per page
- page : defines the current page

Example request body


```
{
  "_links": {
    "next": {
      "href":   "https://my.app-arena.com/api/v2/apps?items=5&page=3"
    },
    "previous": {
      "href":   "https://my.app-arena.com/api/v2/apps?items=5&page=1"
    },
    "self": {
      "href":   "https://my.app-arena.com/api/v2/apps?items=5&page=2"
    }
  },
  "_embedded": {
    "data": {
      "100": {
        "appId":        100,
        "name":         "example App",
        "lang":         "en_US",
        "activated":    true,
        "expiryDate":   "2017-08-04 00:00:00",
        "companyId":    1,
        "templateId":   10,
        "_links": {
          "app": {
            "href":     "https://my.app-arena.com/api/v2/apps/100"
          },
          "language": {
            "href":     "https://my.app-arena.com/api/v2/apps/100/languages/en_US"
          },
          "company": {
            "href":     "https://my.app-arena.com/api/v2/companies/1"
          },
          "template": {
            "href":     "https://my.app-arena.com/api/v2/templates/10"
          }
        }
      },
      "101": {
        "appId": 101,
            .
            .
            .
        }
      },
      "102": {
        "appId": 102,
            .
            .
            .
        }
      },
```

```
            .
            .
            .
          }
        },
        "total_items": 10511,
        "page_size": 5,
        "page_count": 2103,
        "page_number": 2
    }
```

POST **or PUT request**
: The output of these types of requests contains the HTTP status and the created/updated information of the entity in the object "data".

Example request body

```
    {
      "status": 201,
      "data": {
        "appId":        11559,
        "templateId":   888,
        "companyId":    1,
        "lang":         "en_US",
        "name":         "example App",
        "activated":    false,
        "expiryDate":   "2016-08-23 12:24:12"
      }
    }
```

DELETE **request**
: The output of a delete request contains the status and a message.

Example request body

```
    {
        "status":   200,
        "message":  "App '9999' deleted."
    }
```

API - Authorization

You have two options to authenticate against our API: 1) API Key 2) Json Web Token

Both methods can be used be by sending an "Authorization" header.

API Key usage

The example request will return a list of all apps of the company, the api key is assigned to.

POST **/api/v2/apps**

Request header

```
Host: v2.app-arena.com
Content-Type: application/json
Authorization: **YOURAPIKEY**
```

Json Web Token (JWT)

The token will automatically expire after 2 hours. It is meant for the use in browser for example for your Ajax requests.

The example request will return a list of all apps of the company, the token is assigned to.

`POST` **`/api/v2/apps`**

Request header

```
Host: v2.app-arena.com
Content-Type: application/json
Authorization: Bearer **JWT-TOKEN**
```

Generate user token

A user token can act in behalf of a user through the API. The token has the same access rights as the user account the user is generated from.

`POST` **`/api/v2/auth/token`**

Request header

```
Host: v2.app-arena.com
Content-Type: application/json
```

Request body

```
{
    "email":"s.buckpesch@iconsultants.eu",
    "password":"1234"
}
```

Data

- **email** (*string*) – Email address of the user you want to generate a token for
- **password** (*string*) – Clear text password of the user you want to generate a token for

Generate api key token

An Api key token can be used independent from a user. The key advantage of an API key token is its limited access. You can define all access rights for api keys in the App-Manager Backend Developer Section.

`POST` **`/api/v2/auth/token`**

Request header

```
Host: v2.app-arena.com
Content-Type: application/json
```

Request body

```
{
    "apikey":"123456"
}
```

Data apikey (*string*) – Apikey you want to generate a token for

API - Data structure

App-Arena.com apps launched in the world wide web are designed to be highly customizable. Through a system of interconnected configuration sets, we make sure that every customer is able to make his app look and feel like a unique and individually crafted application.

In order to create new apps it is crucial to know the mechanics working in the background.

The components of an application

Every App-Arena.com application consists of three types of components:

- projects
- templates
- apps

All of these components are build up in the same way, consisting of a config, an info, a translation and a language section

Component	Component section	Route
project	• projectconfig • projectinfo • projecttranslation • projectlanguage	• /projects/:projectId/config • /projects/:projectId/info • /projects/:projectId/translation • /projects/:projectId/language
template	• templateconfig • templateinfo • templatetranslation • templatelanguage	• /templates/:templateId/config • /templates/:templateId/info • /templates/:templateId/translation • /templates/:templateId/language
app	• appconfig • appinfo • apptranslation • applanguage	• /apps/:appId/config • /apps/:appId/info • /apps/:appId/translation • /apps/:appId/language

Note: Variable route elements start with a ':'. These variables can be an integer or a string, depending on the context.

The requests to access the resources contained in the components are explained in their respective chapters.

- the config section holds the configuration parameters which define functionality, look and layout of the application
- the info section is a key : value storage for entries which frame the application in its web context
- the translation section contains the strings for the different languages
- the language section defines the available languages for the app

The project

At the foundation of every application is a project. Projects serve as the central entry point for a collection of version varieties, which occur over the course of a development. Every version contains the necessary settings and information an application needs to operate. Templates and apps connected to a version can only overwrite data which is present in the version.

Note: The distinctive versions of a project hold the entirety of customization options, hence defining the palette of available items templates and apps can access.

The version number consists of 3 components: MAJOR.MINOR.PATCH. Recommended use is to increment the:

- MAJOR version when you make incompatible changes,
- MINOR version when you add functionality in a backwards-compatible manner, and
- PATCH version when you make backwards-compatible bug fixes.

The template

In order to customize an application to their individual purpose there are configuration sets which can overwrite the default values contained in the root project.

On our platform, we call these configuration sets 'templates'. Templates can contain as little as a single configuration but may as well be used to configure, style and translate an entire application. However, the real power of the templates derives from the possibility to chain them together and thus, let you create vast amounts of apps of individual behaviour with minimal effort.

The app

Projects and templates are the space in which the designers of an application operate. The app however is the place where the customers have the power to make their application look and feel the way they want it.

Note: While templates mostly contain general configurations altering the core functionality of the project version, the app domain is used for storing what makes the application unique.

An example

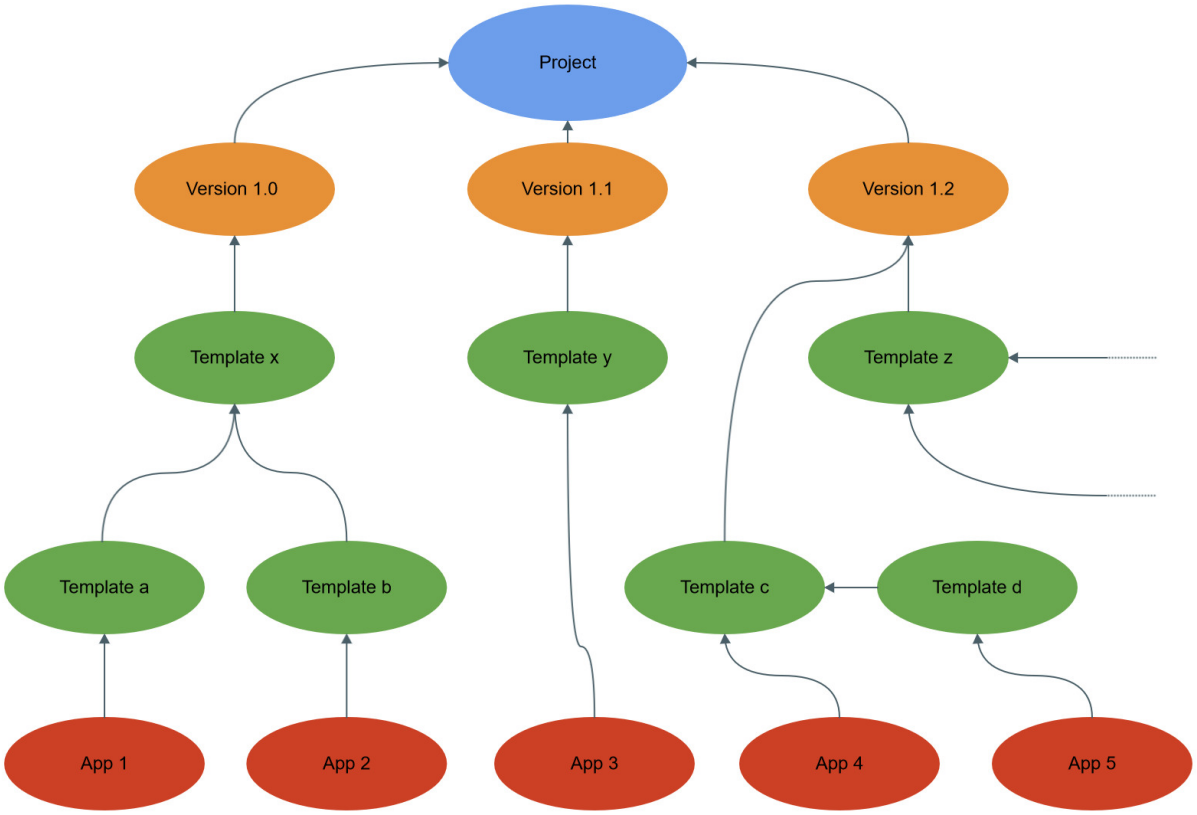
To clarify this concept, imagine a contest project in its initial version "1.0". It contains the logic necessary to let users pick a single item of their preference out of a greater heap of related items. These items can be anything from a picture or a video to a song, the core of the voting system stays the same, no matter what is being voted for. The difference between the contest modes is determined through the configuration sets, which 'shape' the project to the customers needs.

In this example we create a template which 'shapes' the contest-project to a video contest by altering the configuration accordingly. In the next step, we create three templates which alter the Headline to 'Band-Contest', 'Beauty-Contest' and 'Funny-Video-Contest' as well as setting the logos adequately. These templates now get chained to the template we created in the first place. This can be achieved by setting the parentId to the templateId of the initially created template.

This results in three different apps with individual look and feel while the core logic behind them stays the same. The customers now get a web interface in which they can choose things like the font, the colors of their corporate design and the content e.g. the text to be displayed after a user casted a vote for an item.

Should the developers of this contest project come up with new ideas and features, they can easily create a new version of the project (in this example it would be version "1.1"), copying whatever they want to adopt from the initial version to the newly created one. The new version can now be enhanced with new configurations and content while apps pointing to version "1.0" still work without any restriction.

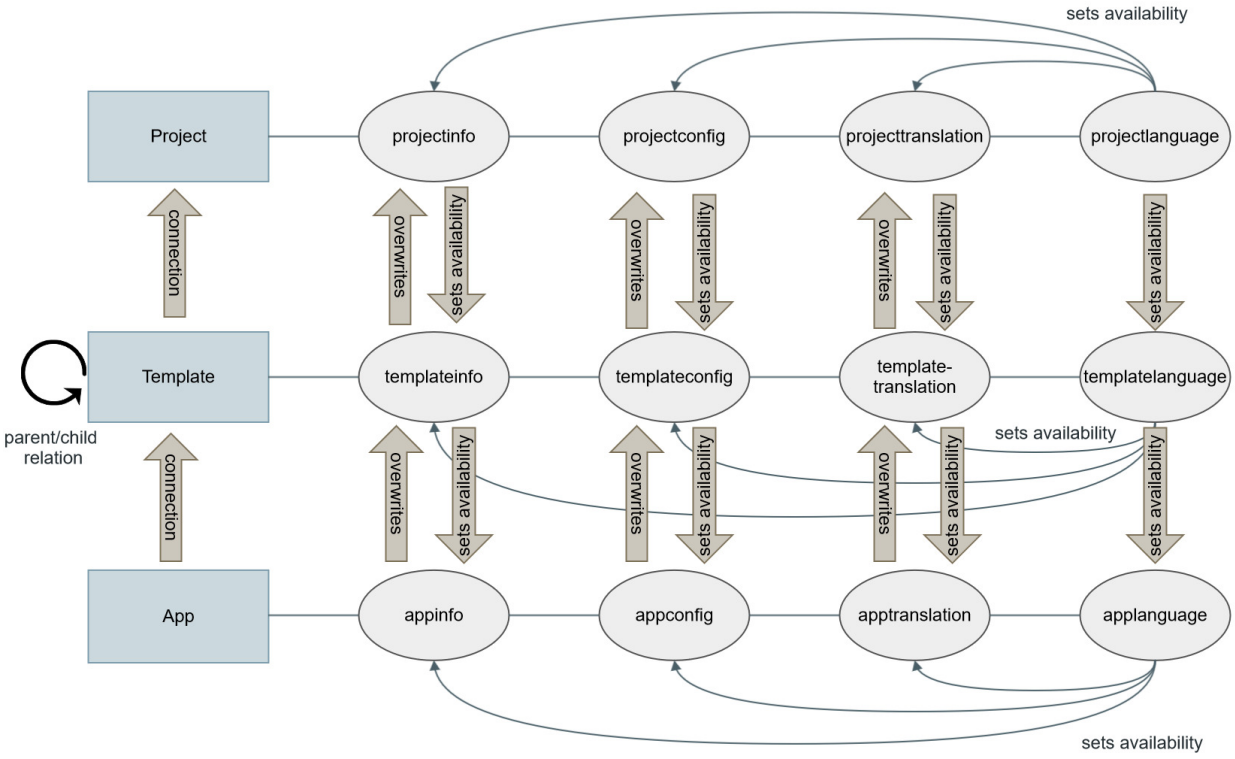
In image 1, the relation between 'apps', 'templates' and 'projects' is illustrated.



As seen in image 1, all versions point to their root project. Templates can point to a project version as well as to a another template. The difference is determined by the parentId: If the parentId points to itself, or in other words, if the parentId equals the templateId, the template points to the project version declared in it. In the case that templateId and parentId differ, the template points to its parent template.

Templates may only contain settings that are already present in the project version. They are therefore only capable of overwriting existing settings and do not create configurations on their own. The same rules apply for the app component. While templates can be used for multiple apps, the configuration of the app applies only for itself. This is why the app is mostly the customers domain where he can give the application his final personal touch.

Projects, templates and app settings are hierarchically structured. This means that settings in the app overwrite occurrences of the same setting from templates and the project. Likewise, template settings overwrite those of the parent template and project. Image 2 visualizes this behaviour.



The image shows how the different types of settings found in projects/templates/apps:

- info: Works as a key => value storage for general application information like e.g. domain name, facebook ID, app validity in days, ...
- config: Is used to configure the application itself like e.g. font, logo uri, images, html and css code, ... The different types of config values are categorized. See the different types of config values and their characteristics here.
- translation: Stores the translation strings used for multi language support.
- language: Sets the available/activated languages.

The hierarchy of the distinct sections lead to some basic rules in the design of an application:

- The project version dictates the range of configs, infos and languages available for templates and apps pointing to it.
- Templates and apps can therefore overwrite (by PUT request) configs, infos and translations in languages present in the project.
- New items can only be created in the project itself, which explains the absence of POST request for templates and apps.

Slightly different is the behaviour for the available languages. While it is possible to edit configs, infos and translations, for templates and apps, the languages present in the app represent the languages activated for the customer. Content can only be retrieved in activated languages.

API - Config data

Config types

Config values have a mandatory field 'type' which determines what kind of data is stored. When creating a config value in a project (-version) or updating (overwriting) it in a template or app, the 'value' field gets validated based on the contents of the 'type' field. This table shows the data types and what to keep in mind using them.

Config-Type	Valid input	Description
checkbox	`mixed` boolean values: 1, 0, "1", "0", true, false	Simple Checkbox which can be checked or unchecked
color	`string` Hex value: #FFFFFF - #000000	Sets a color in Hex format
css	`string` CSS contents	CSS content optionally with variables (config values), gets validated
date	`string` Date in format: "Y-m-d H:i:s"	Y = Year, m = Month, d = Day, H = Hour, i = Minute, s = Second
HTML	`string`	HTML contents in a string
image	`string` URI to an image e.g.: www.example.com/logo.jpg	Points to the location of an image
input	`string` any string	Can be any string the user inputs e.g. a custom name
select	`string` any value contained in the 'options' field in the meta data	Allows the user to select one item out of a list. The list is defined in the meta-field 'options' of the config value (see explanation below).
multiselect	`string` multiple values contained in the 'options' field in the meta data	Similar to 'select' but the user can select multiple items

Meta data behaviour

POST/PUT requests addressing non existing fields (e.g. changing a non existing field with a PUT request or a typo in a JSON key when POSTing a new entity) usually lead to an error with the status code 400 (bad request). This is not the case when PUTting a config or info entry, or more specific, when changing an entry containing the meta data field: unrecognized fields are saved into the meta data array.

Add meta data

To clarify this concept, we go through the process in simple example.

Lets assume a config entry of an app/template/project with the configId 'test_config' and the type 'input' which means it is a text field. The meta data field is initially empty. A GET request on that entry would yield something like:

```
{
  "_embedded": {
    "data": {
      "test_config": {
        "configId": "test_config",
        "lang": "de_DE",
        "name": "noname",
        "revision": 0,
        "value": "some_value",
        "meta": null,
        "type": "input",
        "description": "This is a test config entry.",
        "appId": 1,
        "_links": {
                .
                .
                .
          }
        }
      }
    }
  }
}
```

Now we want to give this entry some additional information in the meta data field, so we submit a PUT request. In this example the config entry lives in an app with the appId '1', so the request url is

PUT /apps/1/configs/test_config

```
{
    "author"    :   "John Doe",
    "content"   :   "this is an example"
}
```

Because the app config entry doesn't have fields called 'author' and 'content' they will be created in the meta data. A GET request on this config entry now yields

> **Warning:** In consequence, you cannot use meta-keys equal to existing fields of a config entry like: 'configId', 'lang', 'value', ...

```
{
  "_embedded": {
    "data": {
```

```
      "test_config": {
        "configId": "test_config",
        "lang": "de_DE",
        "name": "noname",
        "revision": 1,
        "value": "some_value",
        "meta": {
          "author": "John Doe",
          "content": "this is an example"
        },
        "type": "input",
        "description": null,
        "appId": 1,
        "_links": {
                .
                .
                .
        }
      }
    }
  }
}
```

Add meta objects

As you can see, the meta field became an object with the newly created information on the top level. To create sub-levels, an object can be submitted. This way it is possible to create objects with unlimited depth. An example of a sub-level object:

```
{
    "options": {"option1": "something", "option2": "something2"}
}
```

After this request, a GET on the 'test_config' yields:

```
{
  "_embedded": {
    "data": {
      "test_config": {
        "configId": "test_config",
        "lang": "de_DE",
        "name": "noname",
        "revision": 1,
        "value": "some_value",
        "meta": {
          "author": "John Doe",
          "content": "this is an example",
          "options": {
            "option1": "something",
            "option2": "something2"
          }
        },
        "type": "input",
        "description": null,
        "appId": 1,
        "_links": {
                .
```

```
                .
                .
            }
          }
        }
      }
    }
}
```

While it is possible to create deep level structures, you can only address the top-level entries. Keeping the meta object shallow is therefore recommended in order to avoid confusion and simplify the reading process.

Delete meta keys

To delete entries, send a PUT request with an empty value.

```
{
    "options": null
}
```

Now a GET request yields:

```
{
  "_embedded": {
    "data": {
      "test_config": {
        "configId": "test_config",
        "lang": "de_DE",
        "name": "noname",
        "revision": 1,
        "value": "some_value",
        "meta": {
          "author": "John Doe",
          "content": "this is an example"
        },
        "type": "input",
        "description": null,
        "appId": 1,
        "_links": {
                .
                .
                .
          }
        }
      }
    }
  }
}
```

Change meta data

In order to change existing meta data entries use the same approach as adding data.

> **Warning:** Changing a meta key will overwrite the existing data in that key completely!

API - App requests

Hint: While this documentation uses dummy names like config_1, info_1, etc., you are free to choose the ID of the values yourself as long as they contain only letters from a-z, numbers 0-9 and the underscore

/apps

The app component consists of the following fields:

app fields

appId `integer` the unique identifier this entity belongs to

templateId `integer` the template this app points to

companyId `integer` id of the owning company

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

name `string` give this entity a name

activated `bool` the status of the app

expiryDate `string` the date until the app is running or `integer` the number of days the app is running

common fields

created `dateTime` the date this entity was created

updated `dateTime` the date this entity was modified

createdFromIp `string` the IP of the creator of this entity

updatedFromIp `string` the IP of the person modifying this entity

createdBy `string` the username or apikeyId of the creator of this entity

updatedBy `string` the username or apikeyId of the person modifying this entity

deletedAt the entity got deleted when this field contains a `dateTime`, otherwise this field contains `null`

relations

company*, template*, channels, configs, translations, infos, languages*

* can be fetched via collection request

GET /apps

Receive a collection of apps owned by your company.

Query parameters

page
items
fields
exclude
orderasc/orderdesc

rel

Example response body


```
{
  "_links": {
    "next": {
      "href": "https://my.app-arena.com/api/v2/apps?page=2"
    },
    "self": {
      "href": "https://my.app-arena.com/api/v2/apps"
    }
  },
  "_embedded": {
    "data": {
      "1": {
        "appId": 1,
        "name": "Example app",
        "lang": "en_US",
        "activated": true,
        "expiryDate": "2016-11-30 00:00:00",
        "companyId": 1,
        "templateId": 888,
        "_links": {
          "app": {
            "href": "https://my.app-arena.com/api/v2/apps/1"
          },
          "appLanguage": {
            "href": "https://my.app-arena.com/api/v2/apps/1/languages/en_US"
          },
          "company": {
            "href": "https://my.app-arena.com/api/v2/companies/1"
          },
          "template": {
            "href": "https://my.app-arena.com/api/v2/templates/888"
          }
        }
      },
      "2": {
        "appId": 2,
        "name": "Example app 2",
                .
                .
                .
      },
      "3": {
                .
                .
                .
      },
        .
        .
        .
      "N": {
                .
                .
```

```
                    .
        }
      }
    },
    "total_items": 1000,
    "page_size": 20,
    "page_count": 50,
    "page_number": 1
}
```


GET /apps/:appId

Receive information about an app entity specified by :appId

Query parameters

fields

exclude

rel

Example response body


```
{
  "_embedded": {
    "data": {
      "appId": 1,
      "name": "Example app",
      "lang": "de_DE",
      "activated": false,
      "expiryDate": "2099-01-01 00:00:00",
      "companyId": 1,
      "templateId": 888,
      "_links": {
        "app": {
          "href": "https://my.app-arena.com/api/v2/apps/1"
        },
        "appLanguage": {
          "href": "https://my.app-arena.com/api/v2/apps/1/languages/de_DE"
        },
        "company": {
          "href": "https://my.app-arena.com/api/v2/companies/1"
        },
        "template": {
          "href": "https://my.app-arena.com/api/v2/templates/888"
        }
      }
    }
  }
}
```


POST /apps

Creates a new app

Query parameters
force

Example request body

```
{
    "templateId"    :   888,
    "name"          :   "created example app",
    "expiryDate"    :   60,
    "lang"          :   "de_DE"
}
```

Example response body

```
{
  "status": 201,
  "data": {
    "appId": 1,
    "templateId": 888,
    "companyId": 1,
    "lang": "de_DE",
    "name": "created example app",
    "activated": false,
    "expiryDate": "2016-08-26 10:39:00"
  }
}
```

Required data

name `string` give this entity a name

templateId `integer` The template ID this app is connected to

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

Optional data

companyId `integer` The ID of the owning company, if not specified, the app will be owned by the company used for authorization

expiryDate

`Integer` Sets the number of days the app is valid, 0 sets the app valid for 50 years.

`String` Sets a date for app expiration, needs to be in the format 'Y-m-d H:i:s' with Y=year, m=month, d=day, H=hour, i=minute, s=second

activated `bool` the status of the app

Copy an existing app

If you want to modify an existing template but keep the original, you can copy it by sending a POST request with the field "copyFrom" : "template" and the templateId

POST /apps

Example request body

```
{
    "copyFrom"      : 1
}
```

Example response body

```
{
  "status": 201,
  "data": {
    "appId": 2,
    "templateId": 1,
    "companyId": 1,
    "lang": "de_DE",
    "name": "App name [copy]",
    "activated": true,
    "expiryDate": "2016-10-03 13:16:52"
  }
}
```

Required data

copyFrom `integer``|``string` sets the app ID which is to be copied

Optional data

templateId `integer` sets the template the new app is pointing to

companyId `integer` sets a different company than your own as owner of the newly created app

expiryDate `string` sets the expiration date of the app `integer` sets the expiration date in days. A value of 30 means that the app will expire in 30 days from the day of execution

lang `string` sets the default language of the new app. This language must be present in the root project

name `string` defines the name of the new app. If not specified, the name of the original app with an additional "[copy]" string is used

activated `bool` sets the activation status of the new app

PUT /apps/:appId

Alters an app entry specified by :appId

Query parameters

force

Example request body

```
{
    "activated"    :   true,
}
```

Example response body

```
{
  "status": 200,
  "data": {
    "appId": 1,
    "templateId": 888,
    "companyId": 1,
    "lang": "de_DE",
    "name": "created Example app",
    "activated": true,
    "expiryDate": "2016-08-26 10:39:00"
  }
}
```


modifiable parameters

templateId `integer` The template ID this app is connected to

name `string` give this entity a name

activated `bool` the status of the app

expiryDate

`Integer` Sets the number of days the app is valid, 0 sets the app valid for 50 years.

`String` Sets a date for app expiration, needs to be in the format 'Y-m-d H:i:s' with Y=year, m=month, d=day, H=hour, i=minute, s=second

DELETE /apps/:appId

Deletes an app from the database specified by :appId

> **Warning:** This deletes all contained settings and translations as well!

Query parameters

none

Example response body


```
{
  "status": 200,
  "message": "app '1' deleted."
}
```


/apps/:appId/configs

The app config component consists of the following fields:

app config fields

appId/templateId/projectId `integer` the unique identifier this entity belongs to

configId `string` the unique identifier for this entity

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

type `string` has to be one of "facebook", "domain" or "website"

name `string` give this entity a name

value `string` the currently selected config entity content

description `string` an informational description of the purpose of this entity, displayed to the user

meta `string` additional information in custom format. See config meta section for information about the behaviour of meta data

common fields

revision `integer` instead of modifying an existing entity, a new entity is created while the revision reflects its actuality -> The highest revision marks the currently used entity

created `dateTime` the date this entity was created

createdFromIp `string` the IP of the creator of this entity

createdBy `string` the username or apikeyId of the creator of this entity

deletedAt the entity got deleted when this field contains a `dateTime`, otherwise this field in `null`

GET /apps/:appId/configs

Receive a collection of config values of an app specified by :appId

Query parameters

fields

exclude

lang

Example response body


```
{
  "_links": {
    "self": {
      "href": "http://my.app-arena.com/api/v2/apps/1/configs"
    }
  },
  "_embedded": {
    "data": {
      "config_1": {
        "configId": "config_1",
        "lang": "de_DE",
        "name": "config value 1",
        "revision": 0,
        "value": "some_value",
        "meta": {"meta_key":{"meta_inner":"meta_inner_value"}},
        "type": "input",
        "description": "This is an example of a app config value.",
        "appId": 1,
        "_links": {
          "app": {
```

```
            "href": "http://my.app-arena.com/api/v2/apps/1"
          },
          "config": {
            "href": "http://my.app-arena.com/api/v2/apps/1/configs/config_1"
          }
        }
      },
      "config_2": {
        "configId": "config_2",
            .
            .
            .
        }
      },
            .
            .
            .
      }
    }
  }
}
```

GET /apps/:appId/configs/:configId

Receive the information of a config value entity of an app specified by :appId and :configId

Query parameters

fields

exclude

lang

Example response body

```
{
  "_embedded": {
    "data": {
      "configId": "config_1",
      "lang": "de_DE",
      "name": "config value 1",
      "revision": 0,
      "value": "some_value",
      "meta": {
        "meta_key": {
          "meta_inner": "meta_inner_value"
        }
      },
      "type": "input",
      "description": "This is an example of a app config value.",
      "appId": 1,
      "_links": {
        "app": {
          "href": "http://my.app-arena.com/api/v2/apps/1"
        },
```

```
        "config": {
          "href": "http://my.app-arena.com/api/v2/apps/1/configs/config_1"
        }
      }
    }
  }
}
```

PUT /apps/:appId/configs/:configId

Alter a config value for an app specified by :appId and :configId

Query parameters

lang

Example request body

```
{
    "value"    :   "new value"
}
```

Example response body

```
{
  "status": 200,
  "data": {
    "appId": 1,
    "configId": "config_1",
    "lang": "de_DE",
    "type": "input",
    "name": "config value 1",
    "value": "new value",
    "description": "This is an example of a app config value.",
    "revision": 1,
    "meta": {"meta_key":{"meta_inner":"meta_inner_value"}}
  }
}
```

modifiable parameters

value `string` the currently selected config content. See config for more information

name `string` give this entity a name

description `string` an informational description of the purpose of this entity, displayed to the user

meta `string` additional information in custom format. See config meta section for information about the behaviour of meta data

DELETE /apps/:appId/configs/:configId

Deletes a config value of an app from the database specified by :appId and :configId

Query parameters
lang

Example response body

```
{
  "status": 200,
  "message": "Config 'config_1' deleted."
}
```

/apps/:appId/infos

The app info component consists of the following fields:

app info fields

appId/templateId/projectId `integer` the unique identifier this entity belongs to

info_id `string` the unique identifier for this entity

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

value `string` the currently selected config entity content

meta `string` additional information in custom format. See config meta section for information about the behaviour of meta data

common fields

revision `integer` instead of modifying an existing entity, a new entity is created while the revision reflects its actuality -> The highest revision marks the currently used entity

created `dateTime` the date this entity was created

createdFromIp `string` the IP of the creator of this entity

createdBy `string` the username or apikeyId of the creator of this entity

deletedAt the entity got deleted when this field contains a `dateTime`, otherwise this field in `null`

GET /apps/:appId/infos

Receive a collection of info values of an app specified by :appId

Query parameters
fields
exclude
lang

Example response body

```
{
  "_links": {
    "self": {
      "href": "http://my.app-arena.com/api/v2/apps/1/infos"
    }
  },
  "_embedded": {
    "data": {
      "info_1": {
        "infoId": "info_1",
        "lang": "de_DE",
        "revision": 0,
        "value": "some_value",
        "meta": {"meta_key":{"meta_inner":"meta_inner_value"}},
        "description": "This is an example of an app info value.",
        "appId": 1,
        "_links": {
          "app": {
            "href": "http://my.app-arena.com/api/v2/apps/1"
          },
          "info": {
            "href": "http://my.app-arena.com/api/v2/apps/1/infos/info_1"
          }
        }
      },
      "info_2": {
        "infoId": "info_2",
            .
            .
            .
        }
      },
            .
            .
            .
      }
    }
  }
}
```


GET /apps/:appId/infos/:infoId

Receive the information of an info entity of an app specified by :appId and :infoId

Query parameters

fields

exclude

lang

Example response body


```
{
  "_embedded": {
```

```
    "data": {
      "infoId": "info_1",
      "lang": "de_DE",
      "revision": 0,
      "value": "1234",
      "templateId": 888,
      "meta": {
        "type": "integer"
      },
      "_links": {
        "info": {
          "href": "http://my.app-arena.com/api/v2/apps/1/infos/info_1"
        },
        "template": {
          "href": "http://my.app-arena.com/api/v2/templates/888"
        }
      }
    }
  }
}
```

PUT /apps/:appId/infos/:infoId

Alter a info value for an app specified by :appId and :infoId

Query parameters

lang

Example request body

```
{
    "value"    :   "new value"
}
```

Example response body

```
{
  "status": 200,
  "data": {
    "appId": 1,
    "infoId": "info_1",
    "lang": "de_DE",
    "revision": 1,
    "value": "new value",
    "meta": {"type":"string"}
  }
}
```

modifiable parameters

value `string` the currently selected config content. See config for more information

meta `string` additional information in custom format. See config meta section for information about the behaviour of meta data

DELETE /apps/:appId/infos/:infoId

Deletes a info value of an app from the database specified by :appId and :infoId

Query parameters

lang

Example response body

```
{
  "status": 200,
  "message": "Info 'info_1' in app '1' deleted."
}
```

/apps/:appId/languages

The app language component consists of the following fields:

app language fields

appId/projectId `integer` the unique identifier this entity belongs to

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

common fields

created `dateTime` the date this entity was created

updated `dateTime` the date this entity was modified

createdFromIp `string` the IP of the creator of this entity

updatedFromIp `string` the IP of the person modifying this entity

createdBy `string` the username or apikeyId of the creator of this entity

updatedBy `string` the username or apikeyId of the person modifying this entity

deletedAt the entity got deleted when this field contains a `dateTime`, otherwise this field contains `null`

Note: Activated languages always belong to the app while the available languages derive from the corresponding project.

GET /apps/:appId/languages

Receive information about the available and activated languages specified by :appId

Query parameters
none

Example response body

```
{
  "activated": {
    "de_DE": {
      "lang": "de_DE",
      "appId": 1
    }
  },
  "available": {
    "de_DE": {
      "lang": "de_DE",
      "projectId": 1,
      "version": "1.1.0"
    },
    "en_US": {
      "lang": "en_US",
      "projectId": 1,
      "version": "1.1.0"
    }
  }
}
```

POST /apps/:appId/languages

Activate a language in an app specified by :appId and :lang

Query parameters
none

Example request body

```
{
    "lang"  : "en_US"
}
```

Example response body

```
{
  "status": 201,
  "data": {
    "appId": 1,
    "lang": "en_US",
  }
}
```

required data

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

/apps/:appId/translations

The app translation component consists of the following fields:

app translation fields

translationId `string` the unique identifier for this entity

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

appId `integer` the unique identifier this entity belongs to

translated `bool` shows if the string is translated (singular translation)

translation `string` the translated string (singular translation)

pluralized `bool` shows if the string is translated (plural translation)

translationPluralized `string` the translated string (plural translation)

common fields

revision `integer` instead of modifying an existing entity, a new entity is created while the revision reflects its actuality -> The highest revision marks the currently used entity

created `dateTime` the date this entity was created

createdFromIp `string` the IP of the creator of this entity

createdBy `string` the username or apikeyId of the creator of this entity

deletedAt the entity got deleted when this field contains a `dateTime`, otherwise this field in `null`

GET /apps/:appId/translations

Receive translations of an app specified by :appId

Query parameters

lang
fields
exclude
orderasc/orderdesc

Example response body

```
{
  "_links": {
    "self": {
      "href": "http://my.app-arena.com/api/v2/apps/1/translations"
    }
  },
  "_embedded": {
    "data": {
      "translation_1": {
        "translationId": "translation_1",
```

```
        "lang": "de_DE",
        "revision": 0,
        "translation": "translated_text",
        "translated": true,
        "translationPluralized": "translation_pluralized_text",
        "pluralized": true,
        "projectId": 1,
        "version": "1.1.0"
        "_links": {
          "version": {
            "href": "http://my.app-arena.com/api/v2/projects/1/versions/1"
          }
        }
      },
      "translation_2": {
        "translationId": "translation_2",
            .
            .
            .
      },
      "translation_3":{
            .
            .
            .
      },
        .
        .
        .
      "translation_N":{
            .
            .
            .
      }
    }
  }
}
```


PUT /apps/:appId/translations/:translationId

Change a translation for an app specified by :appId and :infoId

Query parameters

lang

Example request body


```
{
    "translation": "new translation"
}
```


Example response body

```
{
  "status": 200,
  "data": {
    "translationId": "translation_1",
    "lang": "de_DE",
    "appId": 1,
    "translation": "new translation",
    "translated": true,
    "translation_pluralized": "translation_pluralized_text",
    "pluralized": true,
    "revision": 1
  }
}
```


modifiable parameters

translation `string` the translated string (singular translation)

translated `bool` shows if the string is translated (singular translation)

translationPluralized `string` the translated string (plural translation)

pluralized `bool` shows if the string is translated (plural translation)

DELETE /apps/:appId/translations/:translationId

Deletes a translation of an app specified by :appId and :infoId

Query parameters

lang

Example response body


```
{
  "status": 200,
  "message": "Translation 'translation_1' deleted."
}
```


/apps/:appId/channels

Note: A Channel needs to be connected to an App in order to publish that App through it

GET /apps/:appId/channels

Returns a list of channels, this app has been published on.

Query parameters

none

Example response body


```
{
  "_links": {
    "self": {
      "href": "https://my.app-arena.com/api/v2/apps/18560/channels"
    }
  },
  "_embedded": {
    "data": [
      {
        "channelId": 6753,
        "companyId": 1,
        "type": "facebook",
        "name": "App-Arena Test Page 01",
        "value": "254789607927209",
        "_links": {
          "company": {
            "href": "https://my.app-arena.com/api/v2/companies/1"
          }
        }
      },
      ...
    ]
  }
}
```


POST /apps/:appId/channels

Connects an App with an existing channel of your company

Query parameters
none

Example request body


```
{
    "channelId": 1
}
```


Example response body


```
{
  "status": 201,
  "data": {
      "channelId": 1,
      "companyId": 1,
      "type": "domain",
      "name": "my domain channel",
      "value": "www.mydomain.com",
      "meta": {}
```

```
    }
}
```

Required data

channelId `integer` the channel you want to connect this app to

DELETE /apps/:appId/channels/:channelId

Removes a channel from an App

Query parameters

none

Example response body

```
[
    { "channelId": 1, "companyId": 1, "type": "domain", "name": "my domain channel", "value": "ww.mydomain.com", "meta": {}
    }
]
```

Example response body

```
{
  "status": 201,
  "message": "Channel '10' of app '13784' removed"
}
```

API - Template requests

Hint: While this documentation uses dummy names like config_1, info_1, etc., you are free to choose the ID of the values yourself as long as they contain only letters from a-z, numbers 0-9 and the underscore

Every template points to a parent template or a project. When the parentId is not equal to the templateId, the parent template is used. Only if the parentId equals the templateId, the project is the next step in the chain.

/templates

The template component consists of the following fields:

template fields

templateId `integer` the unique identifier this entity belongs to

projectId `integer` the project this template points to (if parentId equals templateId)

parentId `integer` the parent template this template points to

companyId `integer` the id of the owning company

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

name `string` give this entity a name

public `bool` determines if the template may be used by users of foreign companies

common fields

created `dateTime` the date this entity was created

updated `dateTime` the date this entity was modified

createdFromIp `string` the IP of the creator of this entity

updatedFromIp `string` the IP of the person modifying this entity

createdBy `string` the username or apikeyId of the creator of this entity

updatedBy `string` the username or apikeyId of the person modifying this entity

deletedAt the entity got deleted when this field contains a `dateTime`, otherwise this field contains `null`

relations

apps, company*, parentTemplate*, configs, translations, infos, versions*, languages*

* can be fetched via collection request

GET /templates

Receive a collection of templates owned by your company.

Query parameters

page
items
fields
exclude
orderasc/orderdesc
rel

Example response body

```
{
  "_links": {
    "next": {
      "href": "https://my.app-arena.com/api/v2/templates?page=2"
    },
    "self": {
      "href": "https://my.app-arena.com/api/v2/templates"
    }
  },
```

```
  "_embedded": {
    "data": {
      "1": {
        "templateId": 1,
        "name": "template_1",
        "lang": "de_DE",
        "parentId": 1,
        "projectId": 1,
        "version": "1.1.0"
        "companyId": 1,
        "public": true,
        "_links": {
          "template": {
            "href": "https://my.app-arena.com/api/v2/templates/1"
          },
          "language": {
            "href": "https://my.app-arena.com/api/v2/templates/1/languages"
          },
          "parent": {
            "href": "https://my.app-arena.com/api/v2/templates/1"
          },
          "version": {
            "href": "https://my.app-arena.com/api/v2/projects/1/versions/1"
          },
          "company": {
            "href": "https://my.app-arena.com/api/v2/companies/1"
          }
        }
      },
      "2": {
        "templateId": 2,
            .
            .
            .
      },
      .
      .
      .
      "N":{
            .
            .
            .
      }
    }
  },
  "total_items": 1000,
  "page_size": 20,
  "page_count": 50,
  "page_number": 1
}
```


GET /templates/:templateId

Receive information about a template entity specified by :templateId

Query parameters

fields
exclude
rel

Example response body

```
{
  "_embedded": {
    "data": {
      "templateId": 1,
      "name": "template_1",
      "lang": "de_DE",
      "parentId": 1,
      "projectId": 1,
      "version": "1.1.0",
      "companyId": 1,
      "public": true,
      "_links": {
        "template": {
          "href": "http://my.app-arena.com/api/v2/templates/1"
        },
        "language": {
          "href": "http://my.app-arena.com/api/v2/templates/1/languages"
        },
        "parent": {
          "href": "http://my.app-arena.com/api/v2/templates/1"
        },
        "version": {
          "href": "http://my.app-arena.com/api/v2/projects/1/versions/1"
        },
        "company": {
          "href": "http://my.app-arena.com/api/v2/companies/1"
        }
      }
    }
  }
}
```

POST /templates

Creates a new template

Query parameters
force

Example request body

```
{
    "projectId"     : 1,
    "version"       : 1.2,
```

```
    "name"          : "new template"
}
```

Example response body

```
{
  "status": 201,
  "data": {
    "templateId": 1,
    "projectId": 1,
    "version": "1.1.0"
    "parentId": 1190,
    "companyId": 1,
    "lang": "de_DE",
    "name": "test template for collectionrunner 1467211852",
    "public": false
  }
}
```

Required data

name `string` give this entity a name

projectId `integer` the unique identifier this entity belongs to

Optional data

parentId `integer` the template this template should be connected to, if left blank the newly created templateId is inserted

version `string` The version of the specified project the template should point to, if not specified the most recent version is used

companyId `integer` ID of the owning company, if not specified, app will be owned by the company used for authorization

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

public `bool` determines if the template may be used by users of foreign companies

Creating a template from an app

Sometimes it might be handy to convert an app into a template. In this case a new template is created and all config, info, translation and language entries are copied into it.

In order to execute this, make a regular POST request onto /templates, but instead of submitting the required information for creating a template, just send a field "copyFrom" : "app" and the appId of the app you want to convert.

To keep the response JSON small, only the basic template information is returned. Use a GET request on templates/:templateId/infos, .../configs, .../translations or .../languages to retrieve its contents.

POST /templates

Example request body

```
{
    "copyFrom"  : "app",
    "appId"     :   1
}
```

Example response body

```
{
  "status": 201,
  "data": {
    "templateId": 2,
    "version": "1.0.0",
    "projectId": 1,
    "parentId": 2,
    "companyId": 1,
    "lang": "en_US",
    "name": "App Name [copy]",
    "public": false
  }
}
```

Required data

copyFrom `string` must be "app"

appId `integer` specifies the app the template will be copied from

Optional data

companyId `integer` defines a different company than your own as owner of the newly created template

parentId `integer` defines the template, the newly created template should point to. If left out, the template to which the app pointed will be used, if set to '0', the template points to the project.

projectId `integer` defines the project the newly created template points to. If the parentId is not equal to the templateId, the template points to the parent template, meaning that this will have no effect if a parent template is defined.

version `string` if a projectId is submitted, you can specify the version here

lang `string` sets the default language of the new template. This language must be present in the root project.

name `string` defines the name of the new template. If not specified, the name of the app with an additional "[copy]" string is used

public `bool` sets the public status of the new template

Copy an existing template

If you want to modify an existing template but keep the original, you can copy it by sending a POST request with the field "copyFrom" : "template" and the templateId

POST /templates

Example request body

```
{
    "copyFrom"      : "template",
    "templateId"    :   1
}
```

Example response body

```
{
  "status": 201,
  "data": {
    "templateId": 2,
    "version": "1.0.0",
    "projectId": 1,
    "parentId": 2,
    "companyId": 1,
    "lang": "en_US",
    "name": "Template Name [copy]",
    "public": false
  }
}
```

Required data

copyFrom `string` must be "template"

templateId `integer``|``string` sets the template ID which is to be copied

Optional data

companyId `integer` defines a different company than your own as owner of the newly created template

parentId `integer` defines the template, the newly created template should point to. If left out, the template to which the app pointed will be used, if set to '0', the template points to the project.

projectId `integer` defines the project the newly created template points to. If the parentId is not equal to the templateId, the template points to the parent template, meaning that this will have no effect if a parent template is defined.

version `string` if a projectId is submitted, you can specify the version here

lang `string` sets the default language of the new template. This language must be present in the root project.

name `string` defines the name of the new template. If not specified, the name of the original template with an additional "[copy]" string is used

public `bool` sets the public status of the new template

PUT /templates/:templateId

Alters an template entry specified by :templateId

Query parameters

lang

Example request body

```
{
    "name"          : "new template name"
}
```

Example response body

```
{
  "status": 200,
  "data": {
    "templateId": 1,
    "projectId": 1,
    "version": "1.1.0"
    "parentId": 1,
    "companyId": 1,
    "lang": "de_DE",
    "name": "new template name",
    "public": false
  }
}
```


modifiable parameters

parentId `integer` the parent template this template points to

projectId `integer` the project this template points to (if parentId equals templateId)

version `string` The version of the specified project the template should point to, if not specified the most recent version is used (needs a projectId)

companyId `integer` the id of the owning company

name `string` give this entity a name

public `bool` determines if the template may be used by users of foreign companies

DELETE /templates/:templateId

Deletes an template from the database specified by :templateId

> **Warning:** This deletes all containing settings and translations as well!

Query parameters

none

Example response body


```
{
  "status": 200,
  "message": "Template '1' deleted."
}
```


/templates/:templateId/configs

The template config component consists of the following fields:

template config fields

templateId/projectId `integer` the unique identifier this entity belongs to

configId `string` the unique identifier for this entity

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

type `string` has to be one of "facebook", "domain" or "website"

name `string` give this entity a name

value `string` the currently selected config entity content

description `string` an informational description of the purpose of this entity, displayed to the user

meta `string` additional information in custom format. See config meta section for information about the behaviour of meta data

common fields

revision `integer` instead of modifying an existing entity, a new entity is created while the revision reflects its actuality -> The highest revision marks the currently used entity

created `dateTime` the date this entity was created

createdFromIp `string` the IP of the creator of this entity

createdBy `string` the username or apikeyId of the creator of this entity

deletedAt the entity got deleted when this field contains a `dateTime`, otherwise this field in `null`

GET /templates/:templateId/configs

Receive a collection of config values of an template specified by :templateId

Query parameters

fields

exclude

lang

Example response body

```
{
  "_links": {
    "self": {
      "href": "http://my.app-arena.com/api/v2/templates/1/configs"
    }
  },
  "_embedded": {
    "data": {
      "config_1": {
        "configId": "config_1",
        "lang": "de_DE",
        "revision": 0,
        "name": "template_config_name",
        "value": "some_value",
        "type": "input",
        "description": "This is an example of a template config value.",
        "templateId": 1,
        "meta": {"meta_key":{"meta_inner":"meta_inner_value"}},
```

```
          "_links": {
            "template": {
              "href": "http://my.app-arena.com/api/v2/templates/1"
            }
          }
        },
        "config_2": {
          "configId": "config_2",
              .
              .
              .
        },
          .
          .
          .
        "config_N":{
        }
      }
    }
}
```

GET /templates/:templateId/configs/:configId

Receive the information of a config value entity of an template specified by :templateId and :configId

Query parameters

fields
exclude
lang

Example response body

```
{
  "_embedded": {
    "data": {
      "configId": "config_1",
      "lang": "de_DE",
      "name": "template_config_name",
      "revision": 0,
      "value": "some_value",
      "meta": {
        "meta_key": {
          "meta_inner": "meta_inner_value"
        }
      },
      "type": "input",
      "description": "This is an example of a template config value.",
      "appId": 1,
      "_links": {
        "app": {
          "href": "http://my.app-arena.com/api/v2/apps/1"
        },
        "config": {
```

```
            "href": "http://my.app-arena.com/api/v2/apps/1/configs/config_1"
          }
        }
      }
    }
  }
```

PUT /templates/:templateId/configs/:configId

Alter a config value for an template specified by :templateId and :configId

Query parameters

lang

Example request body

```
{
    "value"    :   "new value"
}
```

Example response body

```
{
  "status": 200,
  "data": {
    "appId": 1,
    "configId": "config_1",
    "lang": "de_DE",
    "type": "input",
    "name": "config value 1",
    "value": "new value",
    "description": "This is an example of a app config value.",
    "revision": 1,
    "meta": {"meta_key":{"meta_inner":"meta_inner_value"}}
  }
}
```

modifiable parameters

value `string` the currently selected config entity content

name `string` give this entity a name

description `string` an informational description of the purpose of this entity, displayed to the user

meta `string` additional information in custom format. See config meta section for information about the behaviour of meta data

DELETE /templates/:templateId/configs/:configId

Deletes a config value of an template from the database specified by :templateId and :configId

Query parameters
lang

Example response body

```
{
  "status": 200,
  "message": "Config 'config_1' deleted."
}
```

/templates/:templateId/infos

The template info component consists of the following fields:

template info fields

templateId/projectId `integer` the unique identifier this entity belongs to

info_id `string` the unique identifier for this entity

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

value `string` the currently selected config entity content

meta `string` additional information in custom format. See config meta section for information about the behaviour of meta data

common fields

revision `integer` instead of modifying an existing entity, a new entity is created while the revision reflects its actuality -> The highest revision marks the currently used entity

created `dateTime` the date this entity was created

createdFromIp `string` the IP of the creator of this entity

createdBy `string` the username or apikeyId of the creator of this entity

deletedAt the entity got deleted when this field contains a `dateTime`, otherwise this field in `null`

GET /templates/:templateId/infos

Receive a collection of info values of an template specified by :templateId

Query parameters
fields
exclude
lang

Example response body

```
{
  "_links": {
    "self": {
      "href": "http://my.app-arena.com/api/v2/templates/1/infos"
    }
  },
  "_embedded": {
    "data": {
      "info_1": {
        "infoId": "info_1",
        "lang": "de_DE",
        "name": "info value 1",
        "revision": 0,
        "value": "some_value",
        "meta": {"meta_key":{"meta_inner":"meta_inner_value"}},
        "type": "input",
        "description": "This is an example of an template info value.",
        "templateId": 1,
        "_links": {
          "app": {
            "href": "http://my.app-arena.com/api/v2/templates/1"
          },
          "info": {
            "href": "http://my.app-arena.com/api/v2/templates/1/configs/info_1"
          }
        }
      },
      "info_2": {
        "infoId": "info_2",
            .
            .
            .
      }
      },
            .
            .
            .
    }
  }
}
```


GET /templates/:templateId/infos/:infoId

Receive the information of an info entity of an template specified by :templateId and :infoId

Query parameters

fields

exclude

lang

Example response body

```
{
  "_embedded": {
    "data": {
      "infoId": "info_1",
      "lang": "de_DE",
      "revision": 0,
      "value": "1234",
      "templateId": 888,
      "meta": {
        "type": "integer"
      },
      "_links": {
        "info": {
          "href": "http://my.app-arena.com/api/v2/apps/1/infos/info_1"
        },
        "template": {
          "href": "http://my.app-arena.com/api/v2/templates/888"
        }
      }
    }
  }
}
```


PUT /templates/:templateId/infos/:infoId

Alter a info value for an template specified by :templateId and :infoId

Query parameters

lang

Example request body


```
{
    "value"    :   "new value"
}
```


Example response body


```
{
  "status": 200,
  "data": {
    "templateId": 1,
    "infoId": "info_1",
    "lang": "de_DE",
    "revision": 1,
    "value": "new value",
    "meta": {"type":"string"}
  }
}
```


modifiable parameters

value `string` the currently selected config entity content

meta `string` additional information in custom format. See config meta section for information about the behaviour of meta data

DELETE /templates/:templateId/infos/:infoId

Deletes a info value of an template from the database specified by :templateId and :infoId

Query parameters

lang

Example response body

```
{
  "status": 200,
  "message": "Info 'info_1' in template '1' deleted."
}
```

/templates/:templateId/languages

The template language component consists of the following fields:

template language fields

appId/projectId `integer` the unique identifier this entity belongs to

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

common fields

created `dateTime` the date this entity was created

updated `dateTime` the date this entity was modified

createdFromIp `string` the IP of the creator of this entity

updatedFromIp `string` the IP of the person modifying this entity

createdBy `string` the username or apikeyId of the creator of this entity

updatedBy `string` the username or apikeyId of the person modifying this entity

deletedAt the entity got deleted when this field contains a `dateTime`, otherwise this field contains `null`

GET /templates/:templateId/languages

Receive information about the available and activated languages specified by :templateId

Query parameters

none

Example response body

```
{
  "available": {
    "de_DE": {
      "lang": "de_DE",
      "projectId": 1,
      "version": "1.1.0"
    },
    "en_US": {
      "lang": "en_US",
      "projectId": 1,
      "version": "1.1.0"
    }
  }
}
```

POST /templates/:templateId/languages

Activate a language in an template specified by :templateId and :lang

Query parameters

none

Example request body

```
{
    "lang"  : "en_US"
}
```

Example response body

```
{
  "status": 201,
  "data": {
    "templateId": 1,
    "lang": "en_US",
  }
}
```

required data

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

/templates/:templateId/translations

The template translation component consists of the following fields:

template translation fields

translationId `string` the unique identifier for this entity

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

templateId `integer` the unique identifier this entity belongs to

translated `bool` shows if the string is translated (singular translation)

translation `string` the translated string (singular translation)

pluralized `bool` shows if the string is translated (plural translation)

translationPluralized `string` the translated string (plural translation)

common fields

revision `integer` instead of modifying an existing entity, a new entity is created while the revision reflects its actuality -> The highest revision marks the currently used entity

created `dateTime` the date this entity was created

createdFromIp `string` the IP of the creator of this entity

createdBy `string` the username or apikeyId of the creator of this entity

deletedAt the entity got deleted when this field contains a `dateTime`, otherwise this field in `null`

GET /templates/:templateId/translations

Receive translations of an template specified by :templateId

Query parameters

lang
fields
exclude
orderasc/orderdesc

Example response body


```
{
  "_links": {
    "self": {
      "href": "http://my.app-arena.com/api/v2/templates/1/translations"
    }
  },
  "_embedded": {
    "data": {
      "translation_1": {
        "translationId": "translation_1",
        "lang": "de_DE",
        "revision": 0,
        "translation": "translated_text",
        "translated": true,
        "translationPluralized": "translation_pluralized_text",
        "pluralized": true,
        "projectId": 1,
        "version": "1.1.0"
        "_links": {
          "version": {
```

```
            "href": "http://my.app-arena.com/api/v2/projects/1/versions/1"
          }
        }
      },
      "translation_2": {
        "translationId": "translation_2",
            .
            .
            .
      },
      "translation_3":{
            .
            .
            .
      },
        .
        .
        .
      "translation_N":{
            .
            .
            .
      }
    }
  }
}
```


PUT /templates/:templateId/translations/:translationId

Change a translation for an template specified by :templateId and :infoId

Query parameters

lang

Example request body


```
{
    "translation": "new translation"
}
```


Example response body


```
{
  "status": 200,
  "data": {
    "translationId": "translation_1",
    "lang": "de_DE",
    "templateId": 1,
    "translation": "new translation",
    "translated": true,
    "translation_pluralized": "translation_pluralized_text",
    "pluralized": true,
```

```
    "revision": 1
  }
}
```

modifiable parameters

translation `string` the translated string (singular translation)

translated `bool` shows if the string is translated (singular translation)

translationPluralized `string` the translated string (plural translation)

pluralized `bool` shows if the string is translated (plural translation)

DELETE /templates/:templateId/translations/:translationId

Deletes a translation of an template specified by :templateId and :infoId

Query parameters

lang

Example response body

```
{
  "status": 200,
  "message": "Translation 'translation_1' in template '1' deleted."
}
```

API - Project requests

Hint: While this documentation uses dummy names like config_1, info_1, etc., you are free to choose the ID of the values yourself as long as they contain only alphanumerics and the underscore

The version concept

As you can see in image 1 of the structure chapter, all versions point to a root project. This was introduced in order to give every application a home instead of letting them coexist on the same level, which can lead to long unordered lists when displaying. With this approach, the list of existing projects is slimmed down drastically while keeping everything nicely accessible.

In order to work on a specific version, the project routes are used while the version can be selected through the query parameter 'version'.

`Example:` **GET /projects/1/configs?version=2.1.0** will return all config entries of the version '2.1.0' of project '1'

If no version query parameter is defined, the API automatically determines the highest version and performs the requested action on it.

The format of the version-number is based on semantic versioning. It is stored in a string and consists of three integers in the format 'Ma.Mi.P', where Ma stands for 'MAJOR', Mi for 'MINOR' and P for 'PATCH'.

Recommended usage is that you increment the:

- MAJOR version when you make incompatible changes,
- MINOR version when you add functionality in a backwards-compatible manner, and
- PATCH version when you make backwards-compatible bug fixes.

/projects

The project component consists of the following fields:

project fields

projectId `integer` the unique identifier this entity belongs to

companyId `integer` the id of the owning company

name `string` give this entity a name

description `string` an informational description of the purpose of this entity, displayed to the user

common fields

created `dateTime` the date this entity was created

updated `dateTime` the date this entity was modified

createdFromIp `string` the IP of the creator of this entity

updatedFromIp `string` the IP of the person modifying this entity

createdBy `string` the username or apikeyId of the creator of this entity

updatedBy `string` the username or apikeyId of the person modifying this entity

deletedAt the entity got deleted when this field contains a `dateTime`, otherwise this field contains `null`

relations for project requests (/projects[/:projectId])

company*, versions*

relations for version requests (/projects/:projectId/versions[/:versionId])

company*, project*, configs, translations, infos, languages*

* can be fetched via collection request

GET /projects

Receive a collection of projects owned by your company.

Query parameters

page
items
fields
exclude
orderasc/orderdesc
rel

Example response body


```
{
  "_links": {
    "next": {
      "href": "http://my.app-arena.com/api/v2/projects?page=2"
    },
    "self": {
      "href": "http://my.app-arena.com/api/v2/projects"
    }
  },
  "_embedded": {
    "data": {
      "1": {
        "projectId": 1,
        "name": "Project_1",
        "description": "This is a project description",
        "companyId": 1,
        "_links": {
          "project": {
            "href": "http://my.app-arena.com/api/v2/projects/1"
          },
          "company": {
            "href": "http://my.app-arena.com/api/v2/companies/1"
          }
        }
      },
      "2": {
        "projectId": 2,
                .
                .
                .
      },
        .
        .
        .
      "N":{
                .
                .
                .
      }
    }
  },
  "total_items": 100,
  "page_size": 20,
  "page_count": 5,
  "page_number": 1
}
```


GET /projects/:projectId

Receive information about a project entity specified by :projectId

Query parameters

fields

exclude
rel

Example response body

```
{
  "_embedded": {
    "data": {
      "projectId": 1,
      "name": "Project_1 name",
      "description": "This is s project description",
      "companyId": 1,
      "_links": {
        "project": {
          "href": "http://my.app-arena.com/api/v2/projects/1"
        },
        "company": {
          "href": "http://my.app-arena.com/api/v2/companies/1"
        }
      }
    }
  }
}
```

POST /projects

Creates a new project

Note: When creating a new project, a version '1.0' and the specified language will be created as well.

Query parameters
force

Example request body

```
{
    "name"      : "new project",
    "lang"      : "de_DE"
}
```

Example response body

```
{
  "status": 201,
  "data": {
    "projectId": 2,
    "companyId": 1,
    "name": "new project",
    "description": null,
    "version": {
```

```
      "projectId": 1,
      "version": "1.1.0"
      "projectId": 2,
      "companyId": 1,
      "name": "autogenerated initial version of project 'new project'.",
      "lang": "de_DE",
      "variant": 1,
      "public": false,
      "language": {
        "projectId": 1,
        "version": "1.1.0"
        "lang": "de_DE",
      }
    }
  }
}
```


Tip: You can change the name of the initial version with a PUT request to /projects/:projectId/versions/1.0

Required data

name `string` give this entity a name

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

sets the default language of the initial project version and makes the language available to all connected templates/apps

Optional data

companyId `integer` the id of the owning company

description `string` an informational description of the purpose of this entity, displayed to the user

Copy an existing project

This request makes a copy of an existing project. You can set

POST /projects

Example request body


```
{
    "copyFrom"  : 1,
    "version"   : "1.0.0"
}
```


Example response body


```
{
  "status": 201,
  "data": {
    "projectId": 2,
    "companyId": 1,
    "name": "dummy project [copy]",
    "description": null
  }
}
```

Required data

copyFrom `integer``|``string` sets the project ID which is to be copied

version

`string` `"all"` copies all existing versions of the project `"latest"` copies the most recent version of the project `"X.Y.Z"` copies the specified version of the project `["A.B.C","X.Y.Z",...]` copies all the declared versions of the project

Optional data

companyId `integer` defines a different company than your own as owner of the newly created template

name `string` defines the name of the new project. If not specified, the name of the original project with an additional "[copy]" string is used

description `string` describe the newly created project

PUT /projects/:projectId

Alters an project entry specified by :projectId

Query parameters

force

Example request body

```
{
    "name":         "new project name",
    "description":  "This is a new project description"
}
```

Example response body

```
{
  "status": 200,
  "data": {
    "projectId": 2,
    "companyId": 1,
    "name": "new project name",
    "description": "This is a new project description"
  }
}
```

modifiable parameters

name `string` give this entity a name

companyId `integer` the id of the owning company

description `string` an informational description of the purpose of this entity, displayed to the user

DELETE /projects/:projectId

Deletes an project from the database specified by :projectId

> **Warning:** This deletes all versions including all contained settings and translations as well!

Query parameters

none

Example response body

```
{
  "status": 200,
  "message": "Project '2' deleted."
}
```

/projects/:projectId/versions

Note: Every request regarding a version of a project uses 'projectId' and 'version' fields for indication. When handling the actual version entities, the field 'variant' is used.

The version component consists of the following fields:

version fields

projectId `integer` the unique identifier this entity belongs to

companyId `integer` the id of the owning company

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

name `string` give this entity a name

variant `string` the version number (is called 'variant' only in the version itself, all other components call this field 'version')

public `bool` determines if the template may be used by users of foreign companies

common fields

created `dateTime` the date this entity was created

updated `dateTime` the date this entity was modified

createdFromIp `string` the IP of the creator of this entity

updatedFromIp `string` the IP of the person modifying this entity

createdBy `string` the username or apikeyId of the creator of this entity

updatedBy `string` the username or apikeyId of the person modifying this entity

deletedAt the entity got deleted when this field contains a `dateTime`, otherwise this field contains `null`

GET /projects/:projectId/versions

Receive information about the versions of a project specified by :project_id

Query parameters

page
items
fields
exclude
orderasc/orderdesc
rel

Example response body

```
{
  "_links": {
    "self": {
      "href": "http://my.app-arena.com/api/v2/projects/1/versions"
    },
    "next": {
      "href": "http://my.app-arena.com/api/v2/projects/1/versions?page=2"
    },
  },
  "_embedded": {
    "data": {
      "1.0.0": {
        "projectId": 1,
        "name": "project version 1.0.0",
        "variant": "1.0.0",
        "public": false,
        "lang": "de_DE",
        "companyId": 1,
        "projectId": 1,
        "_links": {
          "version": {
            "href": "http://my.app-arena.com/api/v2/projects/1/versions/1.0"
          },
          "company": {
            "href": "http://my.app-arena.com/api/v2/companies/1"
          },
          "project": {
            "href": "http://my.app-arena.com/api/v2/projects/1"
          }
        }
      },
      "1.1.0": {
        "projectId": 2,
        "version": "1.1.0",
                .
                .
                .
      },
        .
```

```
        .
        .
      "M.m.P": {
                .
                .
                .
      }
    }
  },
  "total_items": 10,
  "page_size": 5,
  "page_count": 1,
  "page_number": 1
}
```

GET /projects/:projectId/versions/:version

Receive information about a project version specified by :projectId and :version

Note: Use the version number as :version e.g.: GET /projects/1/versions/1.1.0

Query parameters

fields

exclude

rel

Example response body

```
{
  "_embedded": {
    "data": {
      "name": "project version 1.1.0",
      "variant": "1.1.0",
      "public": false,
      "lang": "de_DE",
      "companyId": 1,
      "projectId": 1,
      "_links": {
        "version": {
          "href": "http://my.app-arena.com/api/v2/projects/1/versions/1.1"
        },
        "company": {
          "href": "http://my.app-arena.com/api/v2/companies/1"
        },
        "project": {
          "href": "http://my.app-arena.com/api/v2/projects/1"
        }
      }
    }
  }
}
```

POST /projects/:projectId/versions

Create a new version for a project, specified by :projectId

Note: The default language specified in the request body will be created automatically and is included in the response under the 'language' sub-object!

Query parameters

force

Example request body

```
{
    "name"      : "new project version",
    "lang"      : "de_DE"
}
```

Example response body

```
{
  "status": 200,
  "data": {
    "projectId": 1,
    "companyId": 1,
    "name": "new project version",
    "lang": "de_DE",
    "variant": "1.2.0",
    "public": false,
    "language": {
      "projectId": 3,
      "version": "1.2.0"
      "lang": "de_DE",
    }
  }
}
```

Required data

name `string` give this entity a name

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

Optional data

variant `string` the version number (is called 'variant' only in the version itself, all other components call this field 'version')

public `bool` determines if the template may be used by users of foreign companies

Copy an existing version

Copies a version inside a project specified by :projectId

POST /projects/:projectId/versions

Example request body

```
{
    "copyFrom"  : "1.0.0"
}
```

Example response body

```
{
  "status": 201,
  "data": {
    "projectId": 1,
    "variant": "1.0.1",
    "companyId": 1,
    "name": "dummy version of project 1 [copy]",
    "lang": "de_DE",
    "public": false
  }
}
```

Required data

copyFrom `string` specifies the version which is to be copied

Optional data

name `string` defines the name of the new version. If not specified, the name of the original version with an additional "[copy]" string is used

lang `string` sets the default language of the new version

variant `string` sets the version variant number ("A.B.C") of the new version. If not submitted, the last digit is increased by one

public `bool` sets the public status of the new version

PUT /projects/:projectId/versions/:version

Alters the properties of a version, specified by :projectId and :version

Query parameters

none

Example request body

```
{
    "name"      : "new version name"
}
```

Example response body

```
{
  "status": 200,
  "data": {
    "projectId": 1,
    "companyId": 1,
    "name": "new version name",
    "lang": "de_DE",
    "variant": "1.2.0",
    "public": false
  }
}
```


modifiable parameters

name `string` give this entity a name

public `bool` determines if the template may be used by users of foreign companies

DELETE /projects/:projectId/versions/:version

Deletes a version of an project from the database specified by :projectId and :version

> **Warning:** This deletes all containing settings and translations of the version as well!

Query parameters

lang

Example response body


```
{
  "status": 200,
  "message": "Version '111' deleted."
}
```


/projects/:projectId/configs

The project config component consists of the following fields:

project config fields

projectId `integer` the unique identifier this entity belongs to

version `string` the version number, format: "Ma.Mi.P" Ma=Major, Mi=Minor, P=Patch e.g.: "2.0.3"

configId `string` the unique identifier for this entity

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

type `string` has to be one of "facebook", "domain" or "website"

name `string` give this entity a name

value `string` the currently selected config entity content

description `string` an informational description of the purpose of this entity, displayed to the user

meta `string` additional information in custom format. See config **meta_** section for information about the behaviour of meta data

common fields

revision `integer` instead of modifying an existing entity, a new entity is created while the revision reflects its actuality -> The highest revision marks the currently used entity

created `dateTime` the date this entity was created

createdFromIp `string` the IP of the creator of this entity

createdBy `string` the username or apikeyId of the creator of this entity

deletedAt the entity got deleted when this field contains a `dateTime`, otherwise this field in `null`

Note: For all of the following requests, the query 'version' can be used to select a specific project-version. If it is left blank the operation will automatically use the most recent version

---

GET /projects/:projectId/configs

Receive a collection of config values of an project specified by :projectId

Query parameters

fields

exclude

lang

version

Example response body

```
{
  "_links": {
    "self": {
      "href": "http://my.app-arena.com/api/v2/projects/1/configs"
    }
  },
  "_embedded": {
    "data": {
      "config_1": {
        "configId": "config_1",
        "lang": "de_DE",
        "revision": 0,
        "type": "input",
        "name": "project config_1 name",
        "value": "some_value",
        "meta": {"meta_key":{"meta_inner":"meta_inner_value"}},
        "description": "This is a config value description",
        "projectId": 1,
        "version": "1.1.0"
        "_links": {
          "version": {
```

```
            "href": "http://my.app-arena.com/api/v2/projects/1/versions/1.0"
          }
        }
      },
      "config_2": {
        "configId": "config_2",
                .
                .
                .
      },
        .
        .
        .
      "config_N": {
                .
                .
                .
      }
    }
  }
}
```


GET /projects/:projectId/configs/:configId

Receive the information of a config value entity of a project specified by :templateId and :configId

Query parameters

fields

exclude

lang

Example response body


```
{
  "_embedded": {
    "data": {
      "configId": "bla",
      "lang": "de_DE",
      "revision": 0,
      "type": "input",
      "name": "bla",
      "value": "lala",
      "meta": null,
      "description": null,
      "projectId": 1,
      "version": "1.1.0",
      "_links": {
        "version": {
          "href": "http://my.app-arena.com/api/v2/projects/111/versions/384"
        }
      }
    }
```

```
    }
}
```

POST /projects/:projectId/configs

Creates a new config value

Query parameters

force

Example request body

```
{
    "name"      : "new config",
    "configId"  : "text_content",
    "type"      : "input"
}
```

Example response body

```
{
  "status": 201,
  "data": {
    "projectId": 1,
    "version": "1.1.0"
    "configId": "text_content",
    "lang": "de_DE",
    "type": "input",
    "name": "new config",
    "value": null,
    "description": null,
    "meta": null,
    "revision": 0
  }
}
```

Required data

name `string` give this entity a name

configId `string` the unique identifier for this entity

type `string` has to be one of "facebook", "domain" or "website"

Optional data

value `string` the currently selected config content. See config for more information

description `string` an informational description of the purpose of this entity, displayed to the user

meta `string` additional information in custom format. See config **meta_** section for information about the behaviour of meta data

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

PUT /projects/:projectId/configs/:configId

Alters the properties of a project config entry specified by :projectId and :configId

Query parameters

lang

version

Example request body

```
{
    "name":         "new config name",
    "meta_example": "meta_content",
}
```

Example response body

```
{
  "status": 200,
  "data": {
    "projectId": 1,
    "version": "1.1.0"
    "configId": "config_1",
    "lang": "de_DE",
    "type": "input",
    "name": "new config name",
    "value": "some_value",
    "description": null,
    "meta": "{\"meta_example\":\"meta_content\"}",
    "revision": 2
  }
}
```

modifiable parameters

description `string` an informational description of the purpose of this entity, displayed to the user

name `string` give this entity a name

value `string` the currently selected config entity content

meta `string` additional information in custom format. See config **meta_** section for information about the behaviour of meta data

DELETE /projects/:projectId/configs/:configId

Deletes a config entry of an project from the database specified by :projectId and :configId

Query parameters

lang

Example response body

```
{
  "status": 200,
  "message": "Config 'config_1' in project '1' deleted."
}
```

/projects/:projectId/infos

The project info component consists of the following fields:

project info fields

projectId `integer` the unique identifier this entity belongs to

version `string` the version number, format: "Ma.Mi.P" Ma=Major, Mi=Minor, P=Patch e.g.: "2.0.3"

info_id `string` the unique identifier for this entity

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

value `string` the currently selected config entity content

meta `string` additional information in custom format. See config **meta_** section for information about the behaviour of meta data

common fields

revision `integer` instead of modifying an existing entity, a new entity is created while the revision reflects its actuality -> The highest revision marks the currently used entity

created `dateTime` the date this entity was created

createdFromIp `string` the IP of the creator of this entity

createdBy `string` the username or apikeyId of the creator of this entity

deletedAt the entity got deleted when this field contains a `dateTime`, otherwise this field in `null`

GET /projects/:projectId/infos

Receive the collection of info values of a project specified by :projectId

Query parameters

fields

exclude

lang

Example response body

```
{
  "_links": {
    "self": {
      "href": "http://my.app-arena.com/api/v2/projects/1/infos"
    }
  },
  "_embedded": {
    "data": {
      "info_1": {
        "infoId": "info_1",
        "lang": "de_DE",
        "revision": 1,
        "value": "some_value",
        "projectId": 1,
        "version": "1.1.0"
        "meta": null,
        "_links": {
          "version": {
            "href": "http://my.app-arena.com/api/v2/projects/1/versions/1.0"
          }
        }
      },
      "info_2": {
        "infoId": "info_2",
                  .
                  .
                  .
      },
        .
        .
        .
      "info_N": {
                  .
                  .
                  .
      }
    }
  }
}
```


GET /projects/:projectId/infos/:infoId

Receive the information of an info entity of an project specified by :projectId and :infoId

Query parameters

fields

exclude

lang

Example response body


```
{
  "_embedded": {
```

```
    "data": {
      "infoId": "info_1",
      "lang": "de_DE",
      "revision": 1,
      "value": "some_value",
      "projectId": 1,
      "version": "1.1.0",
      "meta": {
        "type": "string"
      },
      "_links": {
        "version": {
          "href": "http://my.app-arena.com/api/v2/projects/1/versions/1.0"
        }
      }
    }
  }
}
```

POST /projects/:projectId/infos

Creates a new info entry

Query parameters

force

Example request body

```
{
    "name"      : "new info name",
    "infoId"    : "new info",
    "lang"      : "de_DE",
    "metakey"   : "metavalue"
}
```

Example response body

```
{
  "status": 200,
  "data": {
    "projectId": 1,
    "version": "1.1.0"
    "infoId": "new info",
    "lang": "de_DE",
    "value": null,
    "meta": {"metakey": "metavalue"},
    "revision": 0
  }
}
```

Required data

infoId `string` the unique identifier for this entity

Optional data

value `string` the currently selected config content. See config for more information

meta `string` additional information in custom format. See config **meta_** section for information about the behaviour of meta data

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

PUT /projects/:projectId/infos/:infoId

Alter a info value for an project specified by :projectId and :infoId

Query parameters

lang

Example request body

```
{
    "value":   "new value"
}
```

Example response body

```
{
  "status": 200,
  "data": {
    "projectId": 1,
    "version": "1.1.0"
    "infoId": "info_1",
    "lang": "de_DE",
    "value": "new value",
    "meta": "{\"type\":\"string\"}",
    "revision": 2
  }
}
```

modifiable parameters

value `string` the currently selected config content. See config for more information

meta `string` additional information in custom format. See config **meta_** section for information about the behaviour of meta data

DELETE /projects/:projectId/infos/:infoId

Deletes a info value of an project from the database specified by :projectId and :infoId

Query parameters

lang

Example response body

```
{
  "status": 200,
  "message": "Info 'info_1' in project '1' deleted."
}
```

/projects/:projectId/languages

The project language component consists of the following fields:

project language fields

projectId

version `string` the version number, format: "Ma.Mi.P" Ma=Major, Mi=Minor, P=Patch e.g.: "2.0.3"

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

common fields

created `dateTime` the date this entity was created

updated `dateTime` the date this entity was modified

createdFromIp `string` the IP of the creator of this entity

updatedFromIp `string` the IP of the person modifying this entity

createdBy `string` the username or apikeyId of the creator of this entity

updatedBy `string` the username or apikeyId of the person modifying this entity

deletedAt the entity got deleted when this field contains a `dateTime`, otherwise this field contains `null`

GET /projects/:projectId/languages

Receive information about the available languages specified by :projectId

Query parameters

none

Example response body

```
{
  "available": {
    "de_DE": {
      "lang": "de_DE",
      "projectId": 1,
      "version": "1.1.0"
    }
  }
}
```

POST /projects/:projectId/languages

Activate a language in an project specified by :projectId and :lang

Query parameters

none

Example request body

```
{
    "lang"  : "en_US"
}
```

Example response body

```
{
  "status": 201,
  "data": {
    "projectId": 1,
    "version": "1.1.0"
    "lang": "en_US"
  }
}
```

required data

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

/projects/:projectId/translations

The template translation component consists of the following fields:

template translation fields

translationId `string` the unique identifier for this entity

version `string` the version number, format: "Ma.Mi.P" Ma=Major, Mi=Minor, P=Patch e.g.: "2.0.3"

lang `string` the default language code. Syntax: de_DE for Germany, de_AT for Austrian german ...

projectId `integer` the unique identifier this entity belongs to

translated `bool` shows if the string is translated (singular translation)

translation `string` the translated string (singular translation)

pluralized `bool` shows if the string is translated (plural translation)

translationPluralized `string` the translated string (plural translation)

common fields

revision `integer` instead of modifying an existing entity, a new entity is created while the revision reflects its actuality -> The highest revision marks the currently used entity

created `dateTime` the date this entity was created

createdFromIp `string` the IP of the creator of this entity

createdBy `string` the username or apikeyId of the creator of this entity

deletedAt the entity got deleted when this field contains a `dateTime`, otherwise this field in `null`

GET /projects/:projectId/translations

Receive translations of a project specified by :projectId

Query parameters

lang
fields
exclude
orderasc/orderdesc

Example response body

```
{
  "_links": {
    "self": {
      "href": "http://my-app-arena.com/api/v2/projects/1/translations"
    }
  },
  "_embedded": {
    "data": {
      "translation_1": {
        "translationId": "translation_1",
        "lang": "de_DE",
        "revision": 0,
        "translation": "This is a translated text",
        "translated": true,
        "translationPluralized": null,
        "pluralized": false,
        "projectId": 1,
        "version": "1.1.0"
        "_links": {
          "version": {
            "href": "http://my-app-arena.com/api/v2/projects/1/versions/1.0"
          }
        }
      },
      "translation_2": {
        "translationId": "translation_2",
                    .
                    .
                    .
      },
        .
        .
        .
      "translation_N": {
                    .
                    .
```

```
                    .
        }
      }
    }
}
```

PUT /projects/:projectId/translations/:translationId

Change a translation for a project specified by :projectId and :infoId

Query parameters

lang

Example request body

```
{
    "translation":  "new translation"
}
```

Example response body

```
{
  "status": 200,
  "data": {
    "translationId": "translation_1",
    "lang": "de_DE",
    "projectId": 1,
    "version": "1.1.0"
    "translated": true,
    "translation": "new translation",
    "translationPluralized": null,
    "pluralized": false,
    "revision": 1
  }
}
```

modifiable parameters

translation `string` the translated string (singular translation)

translated `bool` shows if the string is translated (singular translation)

translationPluralized `string` the translated string (plural translation)

pluralized `bool` shows if the string is translated (plural translation)

DELETE /projects/:projectId/translations/:translationId

Deletes a translation of a project specified by :projectId and :infoId

Query parameters

lang

Example response body

```
{
  "status": 200,
  "message": "Translation 'translation_1' in project '1' deleted."
}
```

API - Company/Customer requests

Note: /customers and /companies request are similar (exceptions are highlighted) and are therefore presented combined here

/companies & /customers

The company component consists of the following fields:

company fields

//todo

relations

customers, apps, templates, projects, users, apikeys*, invitations, channels*

* can be fetched via collection request

GET /companies & GET /customers

Receive a list of your customers

Note: The GET /companies request is reserved to store owners

Query parameters

page
items
fields
exclude
orderasc/orderdesc
rel

Example response body

```
{
  "_links": {
    "next": {
      "href": "http://my.app-arena.com/api/v2/customers?page=2"
    },
    "self": {
```

```
        "href": "http://my.app-arena.com/api/v2/customers"
      }
    },
    "_embedded": {
      "data": {
        "1": {
          "companyId": 1,
          "subdomain": "fictional",
          "name": "Fictional Corp.",
          "address1": "Brainstormroad 333",
          "address2": null,
          "zip": "12345",
          "city": "Cloud City",
          "country": "DE",
          "logo": "www.fictional.com/logo.png",
          "color1": "#478AB8",
          "color2": "#2D343D",
          "parentId": 1,
          "storeId": 1,
          "_links": {
            "company": {
              "href": "http://my.app-arena.com/api/v2/companies/1"
            }
          }
        },
        "2": {
          "companyId": 2,
                       .
                       .
                       .
        },
          .
          .
          .
        "N": {
                       .
                       .
                       .
        }
      }
    },
    "total_items": 100,
    "page_size": 20,
    "page_count": 5,
    "page_number": 1
}
```


GET /customers/:companyId & GET /companies/:companyId

Receive information about one of your customers specified by :companyId

---

Note: From a database point of view there is no difference between a company and a customer. That is why there is used :companyId instead of :customerId in customer requests!

Query parameters

fields
exclude
orderasc/orderdesc
rel

Example response body

```
{
  "_embedded": {
    "data": {
      "1": {
        "companyId": 1,
        "subdomain": "fictional",
        "name": "Fictional Corp.",
        "address1": "Brainstormroad 333",
        "address2": null,
        "zip": "12345",
        "city": "Cloud City",
        "country": "DE",
        "logo": "www.fictional.com/logo.png",
        "color1": "#478AB8",
        "color2": "#2D343D",
        "parentId": 1,
        "storeId": 1,
        "_links": {
          "company": {
            "href": "http://my.app-arena.com/api/v2/companies/1"
          }
        }
      }
    }
  }
}
```

POST /companies & POST /customers

Creates a company or customer

Note: This request creates a new company with your own companyId as parentId which makes it a customer of your company. To create a company/customer for a different owner than yourself use POST /companies/:companyId/customers.

Query parameters
none

Example request body

```
{
    "name"      : "new customer"
}
```

Example response body

```
{
  "status": 201,
  "data": {
    "companyId": 2,
    "storeId": 1,
    "subdomain": null,
    "parentId": 1,
    "name": "new customer",
    "address1": null,
    "address2": null,
    "zip": null,
    "city": null,
    "country": "DE",
    "logo": null,
    "color1": "#478AB8",
    "color2": "#2D343D"
  }
}
```

Required data

name (string) The name of the company/customer

Optional data

subdomain (string) Sets the individual subdomain of the company

address1 & address2 (string) Sets address information of the company

zip (string) Sets the zip code

city (string)

country (string) Sets the country of the company with a two letter code e.g.: Germany -> DE, Autria -> AT etc.

logo (string) Sets the uri of the company logo

color1 & color2 (string) Set the company's default colors as Hex values

PUT /companies/:companyId & PUT /customer/:companyId

Alters a company entry specified by :companyId

Query parameters

none

Example request body

```
{
    "name":         "new company name",
}
```

Example response body

```
{
  "status": 200,
  "data": {
    "companyId": 1,
    "storeId": 1,
    "subdomain": null,
    "parentId": 1,
    "name": "new company name",
    "address1": null,
    "address2": null,
    "zip": null,
    "city": null,
    "country": "DE",
    "logo": null,
    "color1": "#478AB8",
    "color2": "#2D343D"
  }
}
```


modifiable parameters

name (string)

subdomain (string)

address1 & address 2 (string)

zip (string)

city (string)

country (string)

logo (string)

color1 & color2 (string)

DELETE /companies/:companyId & DELETE /customers/:companyId

Deletes an company from the database specified by :companyId

> **Warning:** This deletes every project, template or app this company owns!

Query parameters

none

Example response body


```
{
  "status": 200,
  "message": "Company '1' deleted."
}
```

/companies/:companyId/users & /customers/:companyId/users

GET /companies/:companyId/users & GET /customers/:companyId/users

Receive a list of the users of a company specified by :companyId

Query parameters

page
items
fields
exclude
orderasc/orderdesc

Example response body

```
{
  "_links": {
    "self": {
      "href": "http://my.app-arena.com/api/v2/companies/1/users"
    }
  },
  "_embedded": {
    "data": {
      "1": {
        "userId": 1,
        "username": "john_doe",
        "email": "john@doe.com",
        "gender": "male",
        "firstName": "John",
        "lastName": "Doe",
        "telephone": +555 12345678,
        "displayname": "John Doe",
        "avatar": null,
        "lang": "de_DE",
        "companyId": 1,
        "_links": {
          "company": {
            "href": "http://my.app-arena.com/api/v2/companies/1"
          }
        }
      },
      "2": {
        "userId": 2,
                .
                .
                .
      },
        .
        .
        .
      "N": {
                .
                .
```

```
                    .
          }
        }
      },
      "total_items": 10,
      "page_size": 5,
      "page_count": 2,
      "page_number": 1
}
```


GET /companies/:companyId/users/:userId & GET /customers/:companyId/users/:userId

Receive information about a user of a company specified by :companyId and :userId

Query parameters

fields
exclude
orderasc/orderdesc

Example response body


```
{
  "_embedded": {
    "data": {
      "1": {
        "userId": 1,
        "username": "john_doe",
        "email": "john@doe.com",
        "gender": "male",
        "firstName": "John",
        "lastName": "Doe",
        "telephone": +555 12345678,
        "displayname": "John_Doe",
        "avatar": null,
        "lang": "de_DE",
        "companyId": 1,
        "_links": {
          "company": {
            "href": "http://my.app-arena.com/api/v2/companies/1"
          }
        }
      }
    }
  }
}
```


POST /companies/:companyId/users & POST /customers/:companyId/users

Creates a user

Query parameters

none

Example request body

```
{
    "firstname"     : "Jane",
    "lastname"      : "Doe",
    "email"         : "Jane@doe.com",
    "password"      : "quite_secret_pw",
    "username"      : "Jane_Doe",
    "roles"         : "Administrator",
    "gender"        : "female"
}
```

Example response body

```
{
  "status": 201,
  "data": {
    "userId": 2,
    "companyId": 1,
    "email": "Jane@doe.com",
    "username": "Jane_Doe",
    "gender": female,
    "firstname": "Jane",
    "lastname": "Doe",
    "displayname": Jane_Doe,
    "telephone": null,
    "avatar": null,
    "lang": "de_DE",
    "roles": "Administrator"
  }
}
```

Required data

firstname (string)

lastname (string)

email (string)

username (string)

Optional data

gender (string) Sets the gender of the user. Valid strings: “male” or “female”

telephone (string)

avatar (string) Sets the uri to an avatar picture

lang (string) The default language of the version, if left blank, the default language of the project is used instead Syntax: de_DE for Germany, de_AT for Austrian german, en_US for american english ...

roles (string) Sets the roles of the user. Every role consists of a set of rights. A user can have as much roles as desired.

Syntax: “roles” : “Support” for a single role,

"roles" : ["Support","Translator", ...] for multiple roles. See config for available roles.

PUT /companies/:companyId/users/:userId & PUT /customer/:companyId/users/:userId

Alters a user entry specified by :companyId and :userId

Query parameters

force

Example request body

```
{
    "username"      : "new user name"
}
```

Example response body

```
{
  "status": 200,
  "data": {
    "userId": 2,
    "companyId": 1,
    "email": "Jane@doe.com",
    "username": "new user name",
    "gender": female,
    "firstname": "Jane",
    "lastname": "Doe",
    "displayname": Jane_Doe,
    "telephone": null,
    "avatar": null,
    "lang": "de_DE",
    "roles": "Administrator"
  }
}
```

modifiable parameters

email (string)

username (string)

gender (string)

firstname (string)

lastname (string)

displayname (string)

telephone (string)

avatar (string)

lang (string)

roles (string)

DELETE /companies/:companyId/users/:userId & DELETE /customers/:companyId/users/:userId

Deletes an user from the database specified by :companyId

Query parameters

none

Example response body

```
{
  "status": 200,
  "message": "User '1' deleted."
}
```

/companies/:companyId/customers

Note: The output of the following requests is similar to GET /customers. It is used to receive information about a customer of your customer companies.

GET /companies/:companyId/customers

Note: You can find the output format of this request here

GET /companies/:companyId/customers/:companyId

Note: You can find the output format of this request here

POST /companies/:companyId/customers

Note: You can find the output format of this request here

PUT /companies/:companyId/customers/:companyId

Note: You can find the output format of this request here

DELETE /companies/:companyId/customers/:companyId

Note: You can find the output format of this request here

/companies/:companyId/projects

Note: This request is similar to the GET /projects with the difference that it shows only projects owned by the specified company.

GET /companies/:companyId/projects

Receive a collection of projects owned by a company specified by :companyId.

Query parameters

page
items
fields
exclude
orderasc/orderdesc

Example response body


```
{
  "_links": {
    "next": {
      "href": "http://my.app-arena.com/api/v2/projects?page=2"
    },
    "self": {
      "href": "http://my.app-arena.com/api/v2/projects"
    }
  },
  "_embedded": {
    "data": {
      "1": {
        "projectId": 1,
        "name": "Project_1",
        "description": "This is a project description",
        "companyId": 1,
        "_links": {
          "project": {
            "href": "http://my.app-arena.com/api/v2/projects/1"
          },
          "company": {
            "href": "http://my.app-arena.com/api/v2/companies/1"
          }
        }
      },
      "2": {
        "projectId": 2,
                .
                .
                .
      },
        .
        .
```

```
        .
      "N":{
                .
                .
                .
      }
    }
  },
  "total_items": 100,
  "page_size": 20,
  "page_count": 5,
  "page_number": 1
}
```

/companies/:companyId/templates

Note: This request is similar to the GET /templates with the difference that it shows only templates owned by the specified company.

GET /companies/:companyId/templates

Receive a collection of templates owned by your company.

Query parameters

page

items

fields

exclude

orderasc/orderdesc

Example response body

```
{
  "_links": {
    "next": {
      "href": "https://my.app-arena.com/api/v2/templates?page=2"
    },
    "self": {
      "href": "https://my.app-arena.com/api/v2/templates"
    }
  },
  "_embedded": {
    "data": {
      "1": {
        "templateId": 1,
        "name": "template_1",
        "lang": "de_DE",
        "parentId": 1,
        "versionId": 1,
        "companyId": 1,
```

```
          "public": true,
          "_links": {
            "template": {
              "href": "https://my.app-arena.com/api/v2/templates/1"
            },
            "language": {
              "href": "https://my.app-arena.com/api/v2/templates/1/languages"
            },
            "parent": {
              "href": "https://my.app-arena.com/api/v2/templates/1"
            },
            "version": {
              "href": "https://my.app-arena.com/api/v2/projects/1/versions/1"
            },
            "company": {
              "href": "https://my.app-arena.com/api/v2/companies/1"
            }
          }
        },
        "2": {
          "templateId": 2,
              .
              .
              .
        },
        .
        .
        .
        "N":{
              .
              .
              .
        }
      }
    },
    "total_items": 1000,
    "page_size": 20,
    "page_count": 50,
    "page_number": 1
}
```


/companies/:companyId/apps

Note: This request is similar to the GET /apps with the difference that it shows only apps owned by the specified company.

GET /companies/:companyId/apps

Receive a collection of apps owned by a company specified by :companyId.

Query parameters

page

items

fields
exclude
orderasc/orderdesc

Example response body


```
{
  "_links": {
    "next": {
      "href": "https://my.app-arena.com/api/v2/apps?page=2"
    },
    "self": {
      "href": "https://my.app-arena.com/api/v2/apps"
    }
  },
  "_embedded": {
    "data": {
      "1": {
        "appId": 1,
        "name": "Example app",
        "lang": "en_US",
        "activated": true,
        "expiryDate": "2016-11-30 00:00:00",
        "companyId": 1,
        "templateId": 888,
        "_links": {
          "app": {
            "href": "https://my.app-arena.com/api/v2/apps/1"
          },
          "appLanguage": {
            "href": "https://my.app-arena.com/api/v2/apps/1/languages/en_US"
          },
          "company": {
            "href": "https://my.app-arena.com/api/v2/companies/1"
          },
          "template": {
            "href": "https://my.app-arena.com/api/v2/templates/888"
          }
        }
      },
      "2": {
        "appId": 2,
        "name": "Example app 2",
                .
                .
                .
      },
      "3": {
                .
                .
                .
      },
        .
        .
        .
      "N": {
```

```
                    .
                    .
                    .
          }
        }
      },
      "total_items": 1000,
      "page_size": 20,
      "page_count": 50,
      "page_number": 1
    }
```

/companies/:companyId/channels

The company channels consist of the following fields:

channel fields

channelId `integer` the unique identifier this entity belongs to

type `string` specifies the contents of the config entity

value `string` the currently selected config entity content

companyId `integer` the id of the owning company

meta `string` additional information in custom format. See config **meta_** section for information about the behaviour of meta data

name `string` give this entity a name

GET /companies/:companyId/channels

Receive an array of channels of the requested company

Query parameters

none

Example response body

```
[
  {
    "channelId": 1,
    "type": "domain",
    "name": "my channel",
    "value": "www.mydomain.com",
    "companyId": 1,
    "created": "2016-11-03 11:39:33",
    "updated": "2016-11-03 11:39:33",
    "createdFromIp": "127.0.0.1",
    "updatedFromIp": "127.0.0.1",
    "createdBy": "apikey_1",
    "updatedBy": "apikey_1",
    "meta": {}
```

```
    }
]
```

POST /companies/:companyId/channels

Creates a new Channel for the specified company

Query parameters
none

Example request body

```
{
    "type": "domain",
    "value": "www.mydomain.com",
    "name": "my channel"
}
```

Example response body

```
{
  "channelId": 1,
  "companyId": 1,
  "type": "domain",
  "name": "my channel",
  "value": "www.mydomain.com"
}
```

Required data

name `string` give this entity a name

type `string` specifies the contents of the config entity

value `string` stores channel information like a key, domain name, etc.

Optional data

meta `string` additional information in custom format. See config **meta_** section for information about the behaviour of meta data

PUT /companies/:companyId/channels/:channelId

Alters the channel information of a specified company

Query parameters
none

Example request body

```
{
    "name": "new channel name"
}
```

Example response body

```
{
  "channelId": 1,
  "companyId": 1,
  "type": "domain",
  "name": "new channel name",
  "value": "www.mydomain.com",
  "meta": {}
}
```

modifiable parameters

type `string` has to be one of “facebook”, “domain” or “website”

name `string` the channel name

value `string` stores channel information like a key, domain name, etc.

meta `string` stores meta data in JSON format

DELETE /companies/:companyId/channels/:channelId

Deletes a channel specified by companyId and channelId

Query parameters

none

Example response body

```
{
  "status": 200,
  "message": "Channel '1' deleted."
}
```

SDK

PHP-SDK

Getting started

You can find our PHP SDK on github.

Installation

```
composer require app-arena/php-sdk
```

Usage

Use the Composer autoloader to start using the App-Manager

```
// In your index.php
define("ROOT_PATH", realpath(dirname(__FILE__)));
require ROOT_PATH . '/vendor/autoload.php';

// Add App-Arena App-Manager
$am = new \AppArena\AppManager(
    array(
        "projectId" => 123, // Add your project ID here
        "root_path" => ROOT_PATH,
        "cache" => ["dir" => "/var/cache"],
        'apikey' => "ABCDEFGHIJKLMNOPQRSTUVW" // Add you API key here
    )
);
// Get all necessary instance information to start working
$configs      = $am->getApp()->getConfigs();
$translations = $am->getApp()->getTranslations();
$info         = $am->getApp()->getInfos();
```

Now the connection is build up and you can start using you App-Instance. The App-Manager SDK automatically tries to get your App ID (appId) from GET-Parameters or from Cookies. So your Url should be something like this:

```
http://www.domainofmyapp.com/mypage.php?appId=1234
```

Methods

Method	Description	Parameters	Response
addParams($params)	This will add parameters to the smartLink Url. These parameters will be available as GET-Parameters, when a user* clicks on the smartLink. The parameters will be available as GET parameters as well in the facebook page tab* or within an iframe	array $params Array of parameters which should be passed through	
cleanCache()	Cleans the cache, so that a fresh API call will be done		
getApp()	Returns the currently used App object		int
getBaseUrl()	Returns the BaseUrl your Sharing Url is generated with. By default it will use the currently used domain		string
getBrowser()	Returns user browser information		array
getBrowserName()	Returns the users browser name		string
getBrowserVersion()	Returns user browser major version		int
getCssFiles($css_configs)	Returns CSS Helper object to compile and concatenate Less, CSS and Config-Type (CSS) values	array $css_config Array to define the compilation process (@see CSS Config.)	array List of compiled CSS file path's
getDevice()	Returns user device information		array
getDeviceType()	Returns the device type of the current device `mobile`, `tablet`, `desktop`		string
getEnvironment()	Returns if the app currently running on a `website`, `facebook` or `direct` `website` means the app is embedded via iframe to a website `facebook` means the app is embedded in a facebook page tab `direct` means the app is being accessed directly without iframe embed		string
getFacebookInfo()	Returns all available Facebook information, like currently used fanpage and canvas information.		
getLang()	Returns the currently used Language as Language Code (e.g. de_DE, en_US, ...)		string
getParams()	Returns all params submitted to the SmartLink before redirection		array
getProjectId()	Returns the project ID of the currently selected app		int
getOperatingSystem()	Returns the operating system of the current user		string
getUrl()	Returns the SmartLink Url for Sharing		string
getUrlLong()	Returns the SmartLink Url without Url Shortener	bool $shorten Shorten URL using smartl.ink	string
renderSharePage ($debug = false)	Renders the complete HTML of the Share page including all meta tags and redirection.	bool $debug - Show debug information on the page?	string
setBaseUrl($base_url)	Sets a new base url for your sharing links (->getUrl()).	string $base_url New base url	void
setFilename($filename)	Sets the filename for the SmartLink (default: smartlink.php)	string $filename	void
setLang($lang)	Sets a new language for the current instance	string $lang 5 char Language Code,e .g. de_DE	

The app object

You can use the app object to retrieve config-values, translations and basic information about an app. The PHP SDK tries to get the current app ID from a REQUEST-Parameter or a previous set cookie.

To get the app-object call

```
// In your index.php
$app = $am->getApp();
$appId = $app->getId();
$appConfigs = $app->getConfigs();
$appTranslations = $app->getTranslations();
$appInfo = $appInfos();
```

Method	Description	Parameters	Response
getConfig($config_id, $attr = "value")	Returns the value of a config value	String $config_id Config identifier to get the data for String	string array $attr Attribute or Attributes which should be returned
getConfigs()	Returns all Config Elements of the current instance as array		array
getId()	Returns the currently used App ID		int
getInfo($attr)	Returns an attribute of the instance	String $attr Attribute you want to return	string
getInfos()	Returns all basic information of the current instance		array
getLanguages()	Returns all available and activated languages of the app		array
getTranslation($translation_id, $args = array()	Returns the translation for the submitted ID	String $translation_id Config identifier to get the data Array $args Array of values to replace in the translation (@see http://php.net/manual/de/function.vsprintf.php)	string
getTranslations()	Returns all translations for the currently set language		array
isAdmin($projectSecret)	Returns if the current request contains admin authentication information (GET-params)	String $projectSecret The project secret to validate the Hash	bool

PHP-SDK - CSS, LESS and SCSS

The App-Manager SDK provides a helper to enable you to compile, concatenate and minify all your CSS and Less-Files. As well config values (type CSS) can be included in the process. The output file will be automatically saved to your file cache to keep your app fast. The App-Manager SDK uses the great PHP Less compiler by Josh Schmidt and the PHP SCSS compiler by Leaf Corcoran.

The CSS-Helper Class provides the following functions:

- Compile Less and SCSS files (including @imports), CSS Files and Config value (CSS)
- Compiles Bootstrap CSS :-)
- Dynamically replace Variables with values you submit
- Search and Replace strings

Getting started

To request compiled CSS files, you need to tell the SDK, what files to include in your compiled files. Your "input" is defined in an array:

```
// And here is your initialization code:
define("ROOT_PATH", realpath(dirname(__FILE__)));
$am = new \AppManager\AppManager(
   $m_id,
   array(
        "root_path" => ROOT_PATH,
        "cache_dir" => "/var/cache"
    )
);

/**
 * This example config files will return 2 compiled css files: file1 and file2
 * array(
 *      files         -> Array list of files (using an absolute path) to include in the compilat
 *      config_values -> App-Manager config values of type CSS to include in the compilation
 *      variables     -> Less or Scss Variables which will be replaced in all files
 *      replacements  -> String replacements in all files (e.g. to fix a relative filepath')
 * )
 */
$css_config = array(
    'file1' => array(
        'files' => array(
            ROOT_PATH . '/css/less/bootstrap-custom.less',
            ROOT_PATH . '/css/scss/bootstrap-social.scss',
            ROOT_PATH . '/js/vendor_bower/font-awesome/css/font-awesome.min.css'
        ),
        'config_values' => array(),
        'variables' => array(
            'brand-primary' => $am->getConfig('color_primary'), // Use a config value to set a c
            'border-radius-base' => '0px',
            'border-radius-large' => '0px',
            'border-radius-small' => '0px',
        ),
        'replacements' => array(
            '../fonts/fontawesome' => '../../js/vendor_bower/font-awesome/fonts/fontawesome'
        ),
    ),
    'file2' => array(
        'files' => array(
            ROOT_PATH . '/css/style.css'
            ROOT_PATH . '/css/less/app.less',
        ),
        'config_values' => array( 'css_app', 'css_user' ), // A list of config value IDs to incl
        'variables' => array(
            'primary' => '#478AB8',
            'secondary' => '#2D343D',
```

```
            'highlight' => '#efefef',
        ),
        'replacements' => array()
    ),
);

$css_files = $am->getCssFiles($css_config);
```

Now you got an array containing two CSS file path's. To use them in your app just print them in your HTML head section:

```
<head>
...
<?php
foreach ($css_files as $css_file) {
    ?>
    <link type="text/css" rel="stylesheet" href="<?php echo $css_file ?>" />
    <?php
}
?>
...
</head>
```

Note: Test all available requests in our **API-Explorer_**.

Overwriting variables using SCSS

In you CSS-Config file you can set variables, which will be set before your source files are being compiled. When you use **SCSS source files**, please assure, that all variables you want to overwrite are flagged with **!default**

```
$primary: #112233 !default;

h1 {
    color: $primary;
}
```

With the above configuration for 'file2', the compiled CSS would look like this:

```
h1 {
    color: #478AB8;
}
```

PHP-SDK - SmartLink

To setup a SmartLink, which is responsible for all your redirects, copy the `smartlink.template.php` file to your root folder and rename it to `smartlink.template.php`. As you can see in the file you can customize Meta-Data for sharing by calling `setMetaData(..)`. Additional Parameters which will be passed to your app as GET-Parameters can be add via `addParams(..)` or by just adding your parameters as GET parameter to the "Smart-Link"-Url you will get, when calling `getUrl()`.

The **users device will be detected**, when the user is being redirected to your app target. So mobile and tablet devices will never be redirected to a facebook page tab, as it is not possible for them to display them (No Support from Facebook).

GET Parameters

The Smart-Link technology manages redirects for app users depending on their device, language and environment settings. The Smart-Link should be used for all sharing functionality in your app. It offers an easy way to generate sharing links and some `GET`-Parameters to modify the Redirect-Behaviour.

Note: You can add all of the listed parameters to the SmartLink Url to modify the Redirect behaviour.

Parameter	Description	Example
device	To simulate a different device type (mobile, tablet or desktop), just add a `device`-GET Parameter to your URL. The SmartLink will automatically use this device type then and respond with it. Allowed values are `mobile`, `tablet` and `desktop`.	Simulate the mobile view of an app: `https://www.my-web-app.com/?i_id=1234&device=m`
website	If your app is being embedded in a website via iframe, you should add a GET-Parameter called `website` containing the URL the app is embedded in to your smartlink.php Url. Your users will then be redirected the Website Url and not directly to your app. This will keep traffic up on your website. :-)	Redirect the user to a certain website the iframe with your app is embedded in e.g. https://www.app-arena.com/fotowettbewerb.html The SmartLink is:
fb_page_id	Submit this parameter to redirect to this Facebook fanpage Tab your app is embedded in. You can use this to have the same app installed on several fanpages and to control the redirects	
ref_app_env	Set this parameter to `fb` to force the redirection to the facebook fanpage the app is installed on.	`https://www.my-app.com/?i_id=1234&ref_app_env=`
lang	The language parameter controls the used language of the app	Show your app in french: `https://www.my-web-app.com/?i_id=1234&lang=fr_`
debug	Add the debug parameter to disable redirects and show Debug info on the smartlink.php page	Show the Debug-Page for the SmartLink: `https://www.my-web-app.com/smartlink.php?debug`

Embed an App via iframe ($_GET['website'])

Warning: Safari is blocking third-party cookies within iframes! You need to assure that users visiting your app will be redirected via SmartLink to your application, so the SmartLink can set a cookie as first-party. Within the iframe cookies from this domain will be allowed then. **DO NOT link directly to the page, your app is embedded in**. Always link to the SmartLink redirecting to the page your app is embedded in.

If your app is being embedded in a website via iframe, you should add a GET-Parameter called `website` containing the URL the app is embedded in to your smartlink.php Url. Your users will then be redirected the Website Url and not directly to your app. This will keep traffic up on your website. :-)

Use the `website`-GET Parameter for your iframe-Source is the easiest way to keep your visitors on the website. The App-Manager SDK automatically detects `website`-GET Parameters and set them to a cookie.

Here is an example to embed a photocontest-app to the website `https://www.app-arena.com/fotowettbewerb.html`

```
<iframe src="https://stage.fotowettbewerb.co/?i_id=9713&website=https%3A%2F%2Fwww.app-arena.com%2Ffot
width="100%" height="1200" frameBorder="0"></iframe>
```

Easy parameter passthrough

The SmartLink Technology makes it easy to pass parameters to your application no matter if the application is embedded via iframe or into a Facebook Page-Tab. All GET Parameters passed to your `smartlink.php` file will be written to a cookie (Cookie-key `aa_1234_smartlink`, 1234 is the instance id of your application). When you initialize the app manager object in your application again, then all parameters will be deleted from the cookie and written to the GET parameter again.

So you don't have to care about that... Pass GET parameters to your smartlink.php file and expect them in your app target file. :-)

JS-SDK

Getting started

You can find our JS SDK on github.

Installation

1. Clone the repository
2. npm install
3. npm run build (production & develop) | npm run build:prod (production) | npm run build:dev (develop)

Usage

Methods

Method	Description	Parameters	Response
init(params)	This will add parameters to the smartLink Url. These parameters will be available as GET-Parameters, when a user* clicks on the smartLink. The parameters will be available as GET parameters as well in the facebook page tab* or within an iframe	array $params Array of parameters which should be passed through	

CHAPTER 3

Reseller

Reseller guide

As a reseller you can create your own Whitelabel Web-App-Store running under your own domain. To setup a store and to integrate the store into your processes there is some information, which makes it easier for you to integrate. Read about it on this page.

Create a whitelabel store

To create your own store contact us for pricing and the setup: service@app-arena.com or +49 (0)221 177 340 - 00.

Let customers create free demos

That's an easy one. Just link to `https://www.yourstoredomain.com/app/create?templateId=123&templateName`

Parameter	type	description
templateId	(required) int	ID of the template you want the customer to crete an app of
templateName	(optional) string	Name of the template you want to be displayed in the header of the app creation screen. If you leave this field emplty, then the default template name will be used

When a customer clicks on that link, then he will be asked to register or login. After that he will be redirected to the app creation page.

D

E

G

H

P

R