projectfacts API

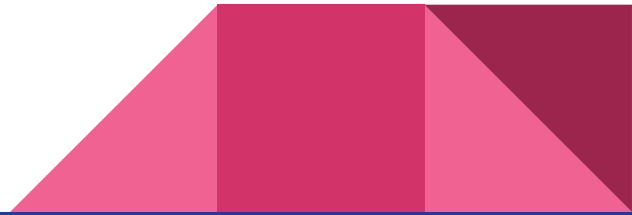
Compact Guide

Version 0.36, requires projectfacts 6.00 oder later

2021-03-09

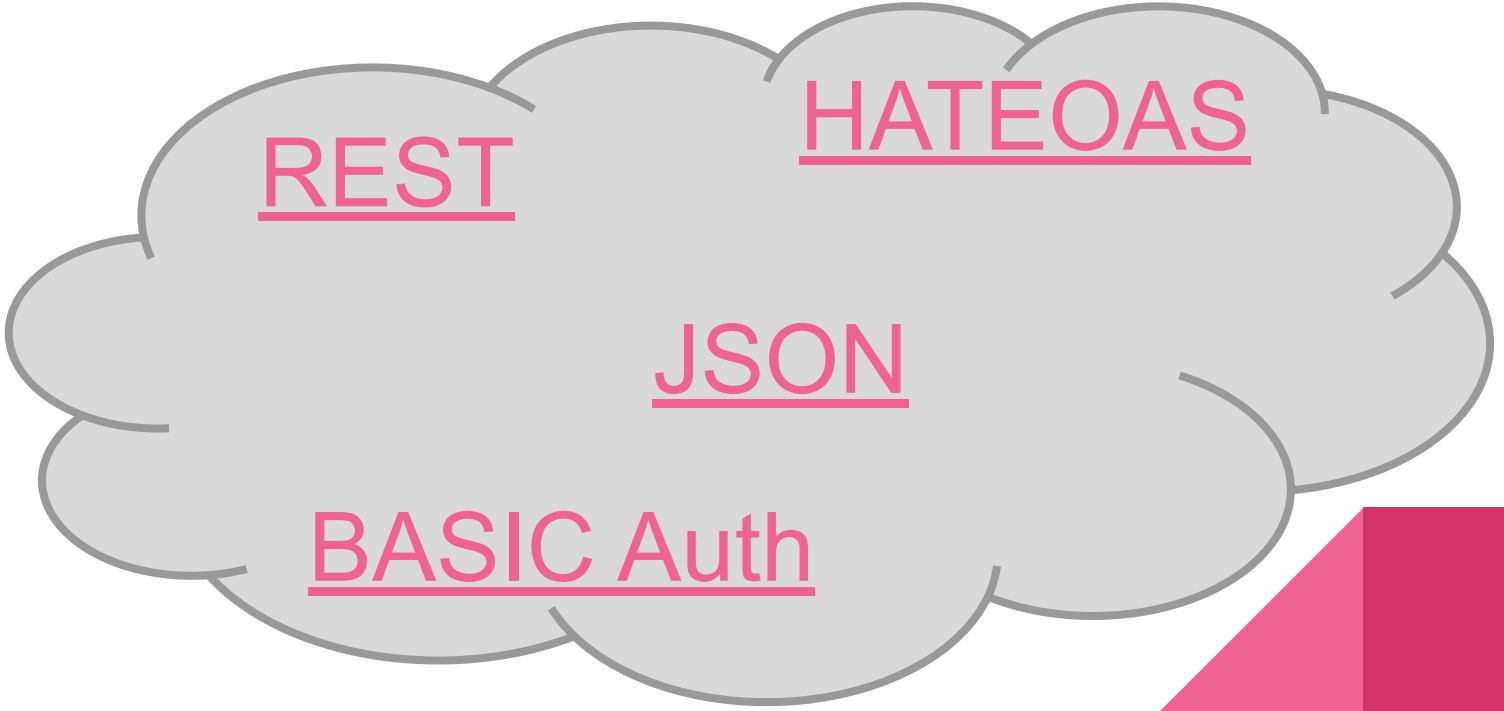
Overview

- Buzzwords: Which standards and best practices are used?
- Authorization: What can be done via the API
- Authentication: Security concept of the API
  - Login-Auth
  - Token-Auth
  - Interface-Auth
- projectfacts API explorer
- Example: Update of a company’s address
- Things to keep in mind



Buzzwords

Before we go on: You should be familiar with these topics...



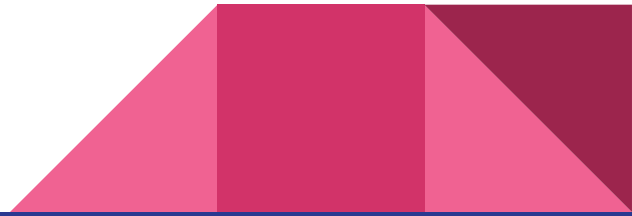
Authorisation: What can be done via API?

In general, an API user has the same permissions as in the web app (projectfacts)

! If a user can delete projects in the web app, he can do it also via API. This cannot be prevented. Also, you cannot keep that user from developing his own app to delete projects:
You cannot prevent an user from using the API at all!

If an user may not book times in the web app, you cannot allow it via API either.

* Exception: Usage of the API via *projectfacts interfaces*
(more later...)



Authentication: Security concept of the API

You cannot use the API just with your projectfacts password:

1. Transfer password with every request?
   ➜ Risk of it getting tapped😬
2. One client gets compromised?
   ➜ Change password
      ➜ All clients would get disconnected.😞
3. The password would be stored on the Client! 😲

Authentication: Login Authentication

The only resource you can access using your projectfacts credentials (login authentication) is the device resource. It has to be created (PUT) when logging in from your App the first time:

POST on **https://sync.projectfacts.de/api/device/**

```
{
  "email": "user@provider.com",
  "password":"projectfactsPassword",
  "deviceName": "YourAppName",
  "deviceType": "de.fivepoint.other",
}
```

Authentication: Login Authentication

If there is a user with matching email and password, the server responds with the new device resource.
It contains the DeviceID and the API-Token, you have to authenticate all following requests with:

```
{
  "_id":"10001234",
  "token": "D1C2B3A4",
  "deviceName": "NameOfYo   App ",
  [...]
}
```

These have to be stored by your app.
Never store the original password!

Authentication: Login Authentication

If there is a user with matching email and password, the server responds with the new device resource.
It contains the DeviceID and the API-Token, you have to authenticate all following requests with:

```
{
  "_id":"10001234",
  "token": "D1C2B3A4",
  "deviceName": "NameIhr
  [...]
}
```

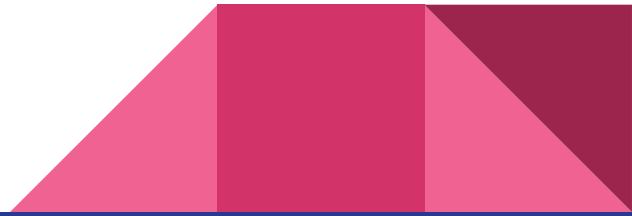
You get the Token **only this one time!**

Authentication: Token Authentication

Your apps local storage:

```
{
  "_id":"10001234",
  "token": "D1C2B3A4",
  [...]
}
```

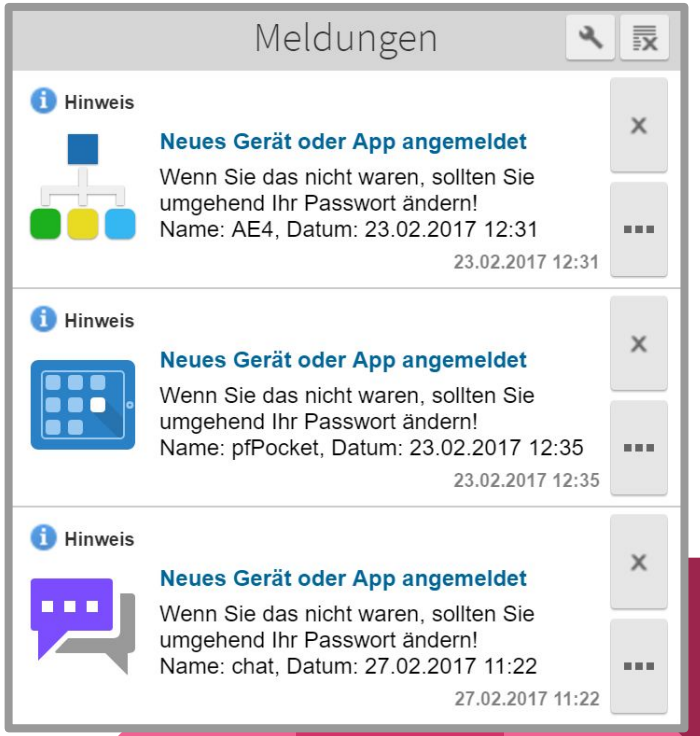
```
// API-Request in Javascript
var credsB64 = window.btoa(device._id + ':' + device.token)
var xhr = new XMLHttpRequest();
xhr.open('GET', url, true);
xhr.setRequestHeader('Authorization', 'Basic ' + credsB64);
xhr.onload = function(response) { /*...*/};
xhr.send();
```



Authentication: Token Authentication

All regular requests are secured with **BASIC-Auth** using Device ID and Device Token: The user of your app as well as the device he is using.

- ➔ A devices permission can be withdrawn by deleting the device within projectfacts (“My devices”).
- ➔ Other devices / apps are not affected.



Authentication: Interface Authentication

- Use for **non-personal** apps, which do not act “on behalf” of a user (e.g. Check-In/Out-Terminal)
- **BASIC-Auth** using Interface-ID and -Password
- Password is set in **projectfacts configuration** (Interfaces)
- **No user-permissions**. Permissions result from the interfaces settings.
- Restriction of connecting IP-address is possible.

projectfacts API explorer

- https://sync.projectfacts.de/htdocs/apps/apiexplorer/index.html
  or root of your pf-Servers + /htdocs/...
- HTML5 / AngularJS-App
- Browse our API
  - How are resources and relationen represented?
  - How to use queries?
- Track communication using the browser's developer tools

Find JSON-Representants as Typescript-Classes:
https://sync.projectfacts.de/api/api/dto.ts

You should take a look!

Example: Update a company's address

We search for “Example corp” in our app and update its address.

These steps are necessary:

1. GET on `/api/contact` including a search query
2. GET on the desired contact resource
3. GET on the contacts main address
4. PUT on that address

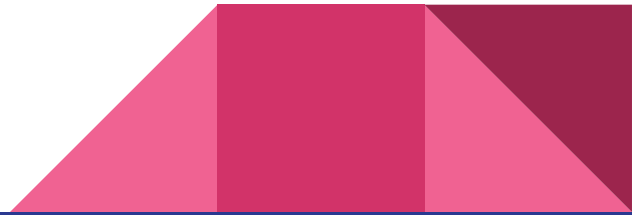
Example: Update a company's address - Step 1

We look for “Example corp.” (The user typed “examp” into our app)

Perform **GET** using deviceID and token as BASIC-Auth-Header on

`https://sync.projectfacts.de/api/contact?caption*=examp`

Resource to search: **contact** Contact resource
Field, we filter on: **caption** Caption of the element
Operator: ***=** “contains” opposed to **=** “equals”
search value: **examp** (op **=** is case insensitive)



Example: Update a company's address - Step 1

We look for “Example corp.” (The user typed “examp” into our app)

```
GET https://sync.projectfacts.de/api/contact?caption*=examp
```

Result (excerpt)


```
{
 "size": 3, "offset":0, "limit":100,
 "items": [
  {"caption":"Max TexaMP","href":"https://…/api/contact/1234","value":1234},
  {"caption":"Ben Examplename","href":"https://…/api/contact/2345","value":2345},
  {"caption":"Example corp.","href":"https://…/api/contact/3456","value":3456}
 ]
}
```

Example: Update a company's address - Step 1

We look for “Example corp.” (The user typed “examp” into our app)

```
GET https://sync.projectfacts.de/api/contact?caption*=examp
```

Result (excerpt)

```
{
 "size": 3, "offset":0, "limit":100,
 "items": [
  {"caption":"Max TexaMP","href":"https://…/api/contact/1234","value"    34},
  {"caption":"Ben Examplename","href":"https://…/api/contact/2345"
  {"caption":"Example corp.","href":"https://…/api/contact/3456",
 ]
}
```

HATEOAS: continue here

Example: Update a company's address - Step 2

We load “Example corp.” (the user selected the third result entry)

```
GET https://sync.projectfacts.de/api/contact/3456
```

Result (excerpt)


```
{
 "_id":3456, "_idKey":"39", _lastModifiedDate:"2015-11-19T10:44:26.000+0000"
 "active":true,
 "person":false,
 "caption":"Example corp.",
 "firstname":null,
 "lastname":"Example corp.",
 "description":"Address outdated\nMoved?",
 "mainAddress":{"caption":";;Examplestreet 7;Examplecity;;122456;Deutschland",
                "href":"…/api/contactfield/173880993","rel":"contactfield",
                "title":"preferred postal address",
                "value":173880993,"idKey":"174"}
}
```

Example: Update a company's address - Step 2

We load “Example corp.” (the user selected the third result entry)

```
GET https://sync.projectfacts.de/api/contact/3456
```

Result (excerpt)

```
{
 "_id":28883816, "_idKey":"39", _lastModifiedDate:"2015-11-19T10:44
 "active":true,
 "person":false,
 "caption":"Example corp.",
 "firstname":null,
 "lastname":"Example corp.",
 "description":"Address outdated\nMoved?",
 "mainAddress":{"caption":";;Examplestreet 7;Examplecity;;122456;Deutschland",
                "href":"…/api/contactfield/173880993","rel":"contactfield",
                "title":"preferred postal address",
                "value":173880993,"idKey":"174"}
}
```

Example: Update a company's address - Step 2

We load “Example corp.” (the user selected the third result entry)

GET https://sync.projectfacts.de/api/contact/3456

Result (excerpt)

```
{
 "_id":28883816, "_idKey":"39"   _lastModifiedDate:"2015-11-19T10:44:26.000+0000"
 "active":true,
 "person":false,
 "caption":"Example corp
 "firstname":null,
 "lastname":"Example corp.
 "description":"Address outda    d\nMoved?",
 "mainAddress":{"caption":";;Examplestreet 7;Examplecity;;122456;Deutschland",
                "href":"…/api/contactfield/173880993","rel":"contactfield",
                "title":"preferred postal address",
                "value":173880993,"idKey":"174"}
}
```

Contacts can be persons as well as organizations

Example: Update a company's address - Step 2

We load “Example corp.” (the user selected the third result entry)

GET https://sync.projectfacts.de/api/contact/3456

Result (excerpt)

```
{
 "_id":28883816, "_idKey":"39", _lastModifiedDate:"2015-11-19T10:44:26.000+0000"
 "active":true,
 "person":false,
 "caption":"Example corp.",
 "firstname":null,
 "lastname":"Example corp.",
 "description":"Address outdated\nMoved?",
 "mainAddress":{"caption":";;Examplestreet 7;Exampl
                "href":"…/api/contactfield/1738809
                "title":"preferred postal address",
                "value":173880993,"idKey":"174"}
}
```

Complex data as link object
(HATEOAS again)
`caption` as “preview” of the resource

Example: Update a company's address - Step 3

We load the **`contactfield`** resource, referenced as **`mainAddress`**

`GET` `https://sync.projectfacts.de/api/contactfield/173880993`

Result (excerpt)

```
{
 "_id":173880993,"_idKey":"174",
 "value":";;Examplestreet 7;Examplecity;;122456;Deutschland",
 "customlabel":null,
 "type":{"caption":"Address", "value":"ADR",
         "href":"https://ws-cp/api/enum/contactfieldtype/ADR",
         "optionsUrl":"…/api/enum/contactfieldtype"},
 "subtype":{"caption":"Work","value":"WORK",
            "href":"…/api/enum/contactfieldsubtype/WORK",
            "optionsUrl":"https://ws-cp/api/enum/contactfieldsubtype"},
 "contact":{"href":"…/api/contact/3456","title":"Contact",
            "value":3456,"idKey":"39"}
}
```

Example: Update a company's address - Step 3

We load the **`contactfield`** resource, referenced as **`mainAddress`**

`GET` `https://sync.projectfacts.de/api/contactfield/173880993`

Result (excerpt)


```
{
 "_id":173880993,"_idKey":"174",
 "value":";;Examplestreet 7;Examplecity;;122456;Deutschla
 "customlabel":null,
 "type":{"caption":"Address", "value":"ADR",
         "href":"https://ws-cp/api/enum/contactfieldtype/ADR
         "optionsUrl":"…/api/enum/contactfieldtype"},
 "subtype":{"caption":"Work","value":"WORK",
           "href":"…/api/enum/contactfieldsubtype/WORK",
           "optionsUrl":"https://ws-cp/api/enum/contactfieldsubtype"},
 "contact":{"href":"…/api/contact/3456","title":"Contact",
           "value":3456,"idKey":"39"}
}
```


Address in vCard format
incl. option for custom label

Example: Update a company's address - Step 3

We load the **`contactfield`** resource, referenced as **`mainAddress`**

`GET` `https://sync.projectfacts.de/api/contactfield/173880993`

Result (excerpt)

```
{
 "_id":173880993,"_idKey":"174",
 "value":";;Musterstr;Musterstadt;;122456;Deutschland",
 "customlabel":null,
 "type":{"caption":"Address", "value":"ADR",
         "href":"…/api/enum/contactfieldtype/ADR",
         "optionsUrl":"…/api/enum/contactfieldtype"},
 "subtype":{"caption":"Arbeit","value":"WORK",
            "href":"…/api/enum/contactfieldsubtype/WORK",
            "optionsUrl":"…/api/enum/contactfieldsubtype"},
 "contact":{"href":"…/api/contact/28883816","title":"Contact"
            "value":28883816,"idKey":"39"}
}
```

The type of that contact field is ‘ADR’ as in vCard

Again, a link object referencing to the details. It links a collection of possible values too!

Example: Update a company's address - Step 4

After updating the value, we store the modified **`contactfield`** resource

`PUT https://sync.projectfacts.de/api/contactfield/173880993`

Request data (excerpt)

```
{
 "_id":173880993,"_idKey":"174",
 "value":";;Saalbaustraße 27;Darmstadt;;64283;Deutschland",
 "customlabel":null,
 "type":{"caption":"Address", "value":"ADR",
         "href":"…/api/enum/contactfieldtype/ADR",
         "optionsUrl":"…/api/enum/contactfieldtype"},
 "subtype":{"caption":"Arbeit","value":"WORK",
           "href":"…/api/enum/contactfieldsubtype/WORK",
           "optionsUrl":"…/api/enum/contactfieldsubtype"},
 "contact":{"href":"…/api/contact/28883816","title":"Contact",
           "value":28883816,"idKey":"39"}
}
```

simple data types like string, can be modified easily

Example: Update a company's address - Bonus

How to update those link objects? **`contactfield`**-Element

`PUT` `https://sync.projectfacts.de/api/contactfield/173880993`

Request data (excerpt)

```
{
 "_id":173880993,"_idKey":"174",
 "value":";;Saalbaustraße 27;Darmstadt;;64283;De schland",
 "customlabel":null,
 "type":{"caption":"Address", "value":"ADR",
         "href":"…/api/enum/contactfieldtype/A
         "optionsUrl":"…/api/enum/contactfieldty "},

 "subtype":"HOME",

 "contact":{"href":"…/api/contact/28883816","title":"Contact",
            "value":28883816,"idKey":"39"}
}
```

But what about these link objects?
Only the value matters...

Example: Update a company's address - Bonus

How to update those link objects? **`contactfield`**-Element

`PUT` `https://sync.projectfacts.de/api/contactfield/173880993`

Request data (excerpt)

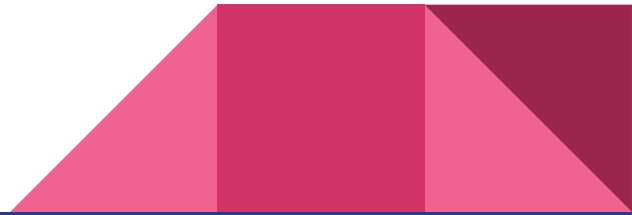
```
{
 "_id":173880993,"_idKey":"174",
 "value":";;Saalbaustraße 27;Darmstadt;;64283;De schland",
 "customlabel":null,
 "type":{"caption":"Adresse", "value":"ADR",
         "href":"…/api/enum/contactfieldtype/A
         "optionsUrl":"  /api/enum/contactfieldty  "},

 "subtype":"HOME",

 "contact":{"href":"…/a i/contact/28883816","title":"Contact",
            "value":28883816,"idKey":"39"}
}
```

But what about these link objects?
Only the value matters...

...so this works too!

Things to keep in mind

- Our API supports you to use **Caching**. Take advantage of it!
- **The first thing** an app should do on its launch **is to load its device resource** - so you know from the start, if the token is still valid
- Remember to **store the token securely in your app**
- Our API is **not complete**, but will be extended with additional resources from time to time
- Our **projectfacts chat** is another example, implemented in typescript
- If you plan to develop a public app, contact us. **We will support you!**



Cheat sheet for queries (collections)

Match	…?field=value	name=”bob ross”&car=volvo&age=3	use quotation where needed (string only)
Contains	…?field*=val	name*=”bo”&car=vol	Names containing “bo”
Range	…?field=1..3	name=”alice”..”bob”&age=20..30	ages of 20, 30 and between
Or	…?field=A,B,C	name=alice,”bob ross”,charlie	all alices, charlies and bob rosses
not empty	…?field	name=bob&car	Search Bobs, owning a car
empty	…?!field	name=bob&!car	Search Bobs, not owning a car

Cheat sheet for matrix parameters (collections)

Sort Ascending	...;sort=name?...	Sort elements by name in ascending order.
Sort Descending	...;sort=!name?...	Sort elements by name in descending order
Result size	...;limit=100?...	load first 100 items after offset
Result offset	...;offset=200?...	skip first 200 items
Subtree	...;parent=1234?...	load childs of item 1234 (tree structures only)
Depth **(maybe SLOW!)**	...;depth=1?...	resolve first generation of references **(SLOW!)**
Show hints	...;showhints=true?...	Displays available sortings (sort by xyz)

Cheat sheet for matrix parameters (items)

7 day caching header	…;lck=value?...	Response is returned with a 7 day caching header “Value” should represent some kind of version to the resource. As long it does not change, your http client will use its cache instead of fetching the resource from the server. Tip: Use the “_lastModifiedDate” found in the collection resource to ensure you get a fresh item, when it is changed on the server.

We wish you success

Head of development:
Christoph Preißer
christoph.preisser@5point.de