EmpliEmbed docs

Table of contents:

EmpliEmbed

EmpliEmbed is JavaScript library that enables embedding widgets from omni-studio to third party applications (outside of VPN) via Emplifi Public API.

If you find a bug, please report to milan.janoch@emplifi.io.

Installation

Library can be installed by executing command

Npm installation

```
npm install empliembed
```

or just

Npm installation

```
npm i empliembed
```

Importing

Library (so far) contains two important components. Their usage will be explained in next chapter. These components can be imported via

Library import

```
import { Widget, WidgetVision } from 'empliembed'
```

Omni-Studio terms

Term	Explanation
Widget	Component in Omni-Studio (chart, select box,...)
Board	Page with Widgets
PreJSON	Library used for expanding schemas of data requests or widget configurations
Vision	Library used for mounting charts
Widget config	Payload that specifies how the widget will looks like (axis, legends, color,...)
Data request	Payload that specifies which data will be downloaded from Emplifi Public API

EmpliEmbed terms

Term	Explanation
Acess Token	Token for Emplifi Public API
Omni API Token	Token for Omni Studio API

Create Project / edit existing one

Create new project build by `webpack` or edit the existing one. Make sure that you can build and execute the client and server side.

Add .env variables

To be able to fetch data from APIs, you need to add into .env file (or enviroment varaibles furing deploy) two tokens - `ACCESS_TOKEN` for Emplifi Public API and `OMNI_API_TOKEN` for Omni Studio API. Both of these tokens needs to be valid. You also need to specify `PACKAGE_URL` which is endpoint for downloading minimalized versions of libraries.

.env

```
ACCESS_TOKEN=4eYo9qFmZw2oRXtZnIyOHhUy3TlPLp2JbrsA1MRc7Fx
OMNI_API_TOKEN=ZnU7lNwYd1nCgVJFyO1qYdKJw9IaZVq5EUxYedRp93O
PACKAGE_URL=https://3348d0628f75ab43fe445e17eb650c1b.sbksapps.com/3/packages/analy
```

These tokens can be generated e.g. via `playground`.

Download the external libraries

Postinstall script

Firstly, you need to specify the postinstall script that will be executed after installing dependencies. In `package.json`, simply add to `"scripts"`

package.json

```
"postinstall": "./scripts/postinstall.sh",
```

Now create in the root of the project folder `scripts` and create file `postinstall.sh`

postinstall.sh

```
#!/bin/bash
set -o errexit
set -o pipefail

mkdir -p assets

# For DEV execution - parses env variables from .env file.

# ACCESS_TOKEN=$(cat .env | grep ACCESS_TOKEN | cut -d '=' -f2-)
# PACKAGE_URL=$(cat .env | grep PACKAGE_URL | cut -d '=' -f2- | sed 's/\"//g')

# For deployment - takes env variables from global variables.
ACCESS_TOKEN=$ACCESS_TOKEN
PACKAGE_URL=$PACKAGE_URL
curl -H "Authorization: Bearer ${ACCESS_TOKEN}" -o ./assets/embedding.js
${PACKAGE_URL}
```

Sometimes (happened on Mac) these script could not be executed due to permissions. If you are facing this problem, just use `chmod 777 ./scripts/postinstall` command to set execution permission.

Run install script

After creating post install script, execute installation of dependencies once again.

```
npm i
```

Now, it should execute the post install script and from Emplifi Public API download PreJSON, PreJSONTime and Vision libraries.

Check if folder `assets` was created

If folder `assets` with file `embedding.js` was created, the installation was successful.

Ready to use

If everything was successful, the library is now ready to be used.

Components

Keep in mind that these components are resizable - if you don't specify width/height, the mounted component will use the given space. To do that, in the outer component, you need to specify `display=flex` and pass style prop to the Widget component `style={{"flex":1}}`.

Widget

The most important component. This component allows to embed existing widgets from Omni-Studio. So far, it allows only vision widgets.

Parameters

- `boardID` *(**number**)*: The ID of the board associated with the widget.
- `widgetID` *(**number**)*: The ID of the specific widget to embed.
- `style` *(**object**)*: JSON object defining the CSS styling for the component.
- `className` *(**string**)*: Name of the styling class to apply to the component (Will be used in future, now it's just prepared for future development).
- `params` *(**array**)*: An array of pre-JSON objects for expanding from Omni-Studio.
- `width` *(**number**)*: The width of the widget component.
- `height` *(**number**)*: The height of the widget component.

Returns

A React element representing the embedded widget component.

Example Usage

```
import React from 'react'
import Widget from 'empliembed'

function App() {
        return (
                <div>
                        <Widget
```

```
                                boardID={1671}
                                widgetID={25353}
                                style={{ backgroundColor: 'white', border: '1px
solid black' }}
                                params={[
                                        {
                                                value: 'P30D/now[sD]',
                                                name: 'cas',
                                                type: 'daterange',
                                        },
                                        {
                                                value: 35,
                                                name: 'alertid',
                                                type: 'number',
                                        },
                                ]}
                                width={300}
                                height={200}
                        />
                </div>
        )
}

export default App
```

Mandatory parameters are only boardID and widgetID. If widget is depended on some Omni-Studio params, they must be specified as well.

Widget Vision

This component renders a visualization based on fetched data and configuration - the advantage of this component is that it no longer needs APIs to fetch the data - data and expanded prejson config is passed via params.

Parameters

- `data` (**object|array**): Fetched data in Omni format (array for multiple requests, object for only one).
- `prejson` (**object**): Expanded widget config (**required**).
- `showConfig` (**bool**): Visualize widget config and not expanded data requests. Default: `false`.
- `showConfigRevealed` (**bool**): Visualize widget config and expanded data requests. Default: `false`.

- `dataRequests` (**object**): Not expanded data requests.
- `expandedDataRequests` (**object|array**): Expanded request(s) (if only one request - object, otherwise array).

Returns

A React element representing the visualization.

Example Usage

```
import React from 'react'
import WidgetVision from 'empliembed'

function App() {
        return (
                <div>
                        <WidgetVision
                                data={...}
                                prejson={...}
                                showConfig={true}
                                showConfigRevealed={false}
                                dataRequests={...}
                                expandedDataRequests={...}
                        />
                </div>
        )
}
export default App
```

`./docs`

Contains functions for generating documentation.

`./src`

Library source code.

`./src/client`

Contains React components (such as `<Widget/> or <WidgetVision/>`) or files with functions/constants used in that frontend components.

`./src/server`

Contains routes for back-end endpoints and other functions for backend.

`./src/index,js`

File that exports React components.

Conventions

File Extension

Extension	Explanation
*.jsx	React components
*.js	Files with constants/functions for front-end or back-end routes
*test.js	Unit tests

Case Types

Type	Case Type
Back-end endpoints	kebab-case
File names	kebab-case
Variable names	camelCase

Development

Git cloning

To be continued when library will be build.

Npm scripts

To be continued when library will be build.