

# Docusaurus v2



# Table of contents:

- Empli-Embed
  - Installation
  - Backend initialization
  - Frontend initialization
    - Importing components
    - Setting tokens
      - Environment variables
      - Widget props
      - Default option
  - Executing app
- Components
  - Widget
    - Parameters
    - Returns
    - Example Usage
  - Widget Vision
    - Parameters
    - Returns
    - Example Usage
  - Omni-Studio terms
  - Empli-Embed terms
- ./docs
- ./src
  - ./src/client
  - ./src/server
  - ./src/index.js
  - ./src/backend.js
  - ./src/tests/\*
- ./script/postinstall.sh
- ./dist
- ./webpack.config.js
- Conventions
  - File Extension
  - Case Types
- Development

- Git cloning
- Downloading libraries
- Building new version
- Running unit tests
- Updating version on NPM
- Introduction
  - What Embedding Studio offers?
- Tokens
  - Creating Public API Token
  - Creating Omni Studio Token
  - Previewing
- Preview
  - How to preview ?
- Generating code
  - Generating Widget code
  - Generating .env values

# Empli-Embed

Empli-Embed 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 it to [milan.janoch@emplifi.io](mailto:milan.janoch@emplifi.io).

# Installation

Library can be installed by executing command

Npm installation

```
npm install empli-embed
```

or just

Npm installation

```
npm i empli-embed
```

# Backend initialization

Create new project using Express.js. In express, require `routes` from `empli-embed` and use prefix `/api` while assigning these routes. Make sure they are assigned to `/api` path.

```
const { routes } = require('empli-embed')
const app = express()

// rest of the code

app.use('/api', routes)
```

Now we've successfully assign routes that are being used by `<Widget/>` component for fetching data.

# Frontend initialization

## Importing components

On the client side, import one of the components by

```
import { WidgetVision } from 'empli-embed'
```

or both of them

```
import { Widget, WidgetVision } from 'empli-embed'
```

## Setting tokens

`<Widget>` component requires valid tokens for embedding - `Omni Studio token` and `Public API token`. There are two options how to require these tokens.

## Environment variables

First option is to set tokens into .env file or enviroment variables. This is the easier option. All you need to do is generate valid tokenS (e.g. via [Embdding Studio] (<https://web-dev-embedded-screen-prototype-x.eks-prod.us-w2.aws.ccl>)) and add them into .env file.

.env

```
ACCESS_TOKEN=$ACCESS_TOKEN      # Public API token
OMNI_API_TOKEN=$OMNI_TOKEN       # Omni Studio token
```

## Widget props

Second option is creating function that will return object with these tokens. E.g. in Embedding Studio, the tokens are taken from localStorage, so the function looks like

Creating object with tokens

```
const getTokensFromLocalStorage = () => {
  return {
    OMNI_API_TOKEN: localStorage.getItem('omni-studio-api-access-token'),
    ACCESS_TOKEN: localStorage.getItem('public-api-access-token'),
  }
}
```

and after that, just pass this function into props of `<Widget>` component

Usage

```
<Widget
  boardID={...}
  widgetID={...}
  // other props
  tokenFunc={getTokensFromLocalStorage}
/>
```

## Default option

If `tokenFunc` is not defined, library will automatically load tokens from .env file.

# Executing app

After previous steps, now you should be able to start application and embed widgets from Omni Studio.



# 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, required***): The ID of the board associated with the widget.
- `widgetID` (***number, required***): 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.
- `tokenFunc` (***func***): Function that returns ACCESS\_TOKEN and OMNI\_STUDIO\_TOKEN.

## Returns

A React element representing the embedded widget component.

## Example Usage

```
import React from 'react'
import { Widget } from 'empli-embed'

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 (payloads). Used only for debugging (WidgetVision is used by `<Widget>` component to mount visualizations) to visualize payload.
- `expandedDataRequests` (**object|array**): Expanded request(s) (if only one request - object, otherwise array). Used only for debugging (WidgetVision is used by `<Widget>` component to mount visualizations) to visualize payloads.

## Returns

A React element representing the visualization.

## Example Usage

```
import React from 'react'
import { WidgetVision } from 'empli-embed'

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

# 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 look like (axis, legends, color,...)
Data request	Payload that specifies which data will be downloaded from Emplifi Public API

# Empli-Embed terms

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

## **./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.

### **./src/backend.js**

File with exported routes of Express router.

### **./src/tests/\***

Folder containing all unit tests and files with mocked values.

## **./script/postinstall.sh**

Script for installing minified versions of libraries.

**./dist**

Folder with minified libraries (PreJSON, Vision) and with built library files.

**./webpack.config.js**

Config used for building library.

# 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

Cloning

```
git clone https://github.com/M1LNES/Embedding-Library.git
```

## Downloading libraries

Library requires minified versions of PreJSON and Vision. Firstly, add into `.env` file valid token and package url.

```
ACCESS_TOKEN=$ACCESS_TOKEN  
PACKAGE_URL=https://3348d0628f75ab43fe445e17eb650c1b.sbksapps.com/3/packages/ana  
version=v1.0.0
```

After that, just simply execute script

```
npm run download
```

that will download into `/dist` folder minified version of libraries.

## Building new version

Library is built by command:

```
npm run build
```

The result is in `/dist` folder and will contain two files - `empli-embed.js` and `empli-embed-backend.js`.



# Running unit tests

To execute all the tests, use command

```
npm test
```

## Updating version on NPM

New version of library can be pushed into npm by

```
npm publish
```

Before publishing, make sure that you re-built the new version of library.

`.npmignore` file contains file that are being ignored during releasing new version into npm. Make sure that if you added any new files that they are included in this file.

# Introduction


## What Embedding Studio offers?

1. Creating tokens
2. Previewing visualizations
3. Generating Widget code and .env variables


# Tokens

Previewing is possible only if the tokens are valid - tokens are store in localStorage and you can create them by two buttons located in footer.

## Creating Public API Token

For Public API Token, you must be connected to `suite account`. After clicking on the button, you will be redirected to page where you just allow the permissions. After that, token will be saved in your localStorage and the button should disappear and  appear.

## Creating Omni Studio Token

For Public API Token, you must be connected to `VPN`. After clicking on the button, you will be redirected to page where you again just allow the permissions. After that, token will be saved in your localStorage and the button should disappear and  appear.

## Previewing

If both tokens were successfully created, the disabled button for previewing should be now enabled and you can start previewing widget visualizations!

# Preview

You can preview widgets only if tokens in localStorage are valid.

## How to preview ?

1. Find board with widget in Omni Studio
2. Select values from fields (e.g. DateRange, profiles etc.)
3. After that, the widget in Omni Studio should be mounted
4. Copy boardID from URL to Embedding Studio
5. Copy widgetID - move mouse to top right corner - Edit config - copy number from Widget #NUMBER to Embedding Studio
6. Copy dependencies - move mouse to top right corner - Open debug tools - Dependencies - COPY AS BOARD PARAMS
7. Add additional specification - such as CSS style, width, height etc. (not required)
8. Click on Preview

# Generating code

## Generating Widget code

After successful preview, button in top right corner will appear. That button will open modal with previewed widget and with widget code located in TextField. You can copy that Widget code by the button below.

## Generating .env values

If both tokens were created successfully, you can copy these two tokens in .env format. Just press button in footer and copy these variables into clipboard.