Last updated 🗓 Date

Embodee Configurator

Draft version 0.3

Installation and Setup

Include the Embodee loader script:

```
<script src='https://beta.embodee.com/__/configurator/loader.js'></script>
```

The loader script will pull all dependencies required to run the Configurator, initialize the 3D model renderer, pull the 3D assets from Embodee Platform and perform the initial rendering of the model.

In your app, initialize the Loader by calling the **`EmbodeeLoader.init`** method with an options object to obtain an instance of the Embodee Configurator class:

```
const Configurator =
     await EmbodeeLoader.init({
       workspaceID: WORKSPACE_ID,
       productCode: PRODUCT_CODE,
       width: 640,
       height: 640,
     });
```

Available configuration options

Option Name	**Type**	**Description**
workspaceID	string	The ID of the Product Line containing the model in the Embodee Platform
productCode	string	The reference code of the model in the Embodee Platform
designID	string	The Design ID (DID) of a saved design. Replaces the product code parameter if specified.
containerID	string	The ID of the 3D viewer container element

variant	string	The name of a product variant to load (if omitted, the Primary variant will be loaded)
width	int	The width of the 3D viewer container in pixels
height	int	The height of the 3D viewer container in pixels
orbitControls	object	Settings that can be used to control the model zoom, tilt and rotation - see the Orbit Controls section for more information
bgColor	string	The background color for the 3D viewer, omit for transparent
show3DIcon	bool	Toggles the 3D icon shown when the 3D model can be interacted with
showLoader	bool	Toggles the loading indicator
quality	string	Specifies the maximum texture quality. Possible values are "high", "medium", "low" or "original". "high" - up to 2048x2048 pixels "medium" - up to 1024x1024 pixels "low" - up to 512x512 pixels "original" (default) - preserve the original 3D model settings

Once an instance of Configurator class has been obtained, you can subscribe for various events. Listen for the productReady event to obtain a callback called when the 3D model and all its assets have been successfully loaded. The full list of events is available from the **eventIDs** property:

```
const events = Configurator.eventIDs;
```

See the Configurator Events section for more information about the available events.

The "**productReady**" event signals the successful completion of the 3D model loading and initialization. When this event is fired, all the Configurator data is available to the client application.

```
Configurator.subscribe(events.productReady, () => {

    // retrieve the UI structure
    const uiTree = Configurator.uiStructure;

    // retrieve the color and fonts library
    const library = Configurator.library;
    // retrieve the viewer configuration
    const config = Configurator.config;

});
```

Configurator API

See the Configurator API document for detailed information about its properties and methods.

Product Structure

The product configuration set up in Embodee Platform consists of various customizable elements. Each element has a unique component code and is assigned a type as described in the UI Node Structure section. The component code can be obtained from the respective UI Structure node element.

Customizing an Element

Changing a customizable property of an element is done via a call to the Configurator's **setComponentProperty** method.

```
Configurator.setComponentValue(componentCode, property, value, isCustom);
```

Customize with Custom Values

When using arbitrary colors, graphics, prints, etc., call the **setComponentValue** method with the "**isCustom**" argument set to **true**.

Example: set the color of a component with code "c01" to 100% red:

```
Configurator.setComponentValue("c01", "color", "FF0000", true);
```

Example: set the graphic of a component with code "p02" to custom url:

```
Configurator.setComponentValue("p02", "material",
"https://example.com/my-graphic.png", true);
```

Please note that the graphics and prints are treated as materials in the context of the 3D model. That's why they can be changed by using the material property of the component.

When passing a custom URL to a print or graphic component, make sure that its CORS headers are set up to allow access from Embodee domains.

The supported graphic formats are PNG, SVG, JPEG, WebP, TIFF, BMP, AVIF with a limited support for PSD, AI, PDF, KTX2 and HDR formats.

Customize with Pre-Defined Values

When using pre-configured customization options, set the "isCustom" argument to false when calling the setComponentValue method.

The library items are set up in the Library section of Embodee Platform and then added to the product components during the design process. The Embodee Platform Library contains colors, fonts, prints, graphics and custom materials. Each library item represents an object that has a name, value and an optional code field.

Example: obtain a list of available colors for a component with code "c01"

```
// get a reference to the component's color property
const prop = Configurator.getComponentProperty('c01', 'color');

// get a list of available colors
const colors = prop.items();
```

The result will contain a list of library colors assigned to this component:

```
{
  "col_48": {
    "id": "col_48",
    "name": "Tan",
    "value": "CBBC8F"
  },
  "col_16": {
    "id": "col_16",
    "name": "Dark Red",
    "value": "7D1919"
  },
  "col_34": {
    "id": "col_34",
    "name": "Water",
    "value": "B2F2EA"
  }, ...
```

In a reactive client application it is advisable to use the node elements of the UI structure to do the same:

```
// given a reference to a node element
const colors = node.props.color.items();
```

Then, to set the component color to "**Dark Red**" library color call:

```
// col_16 is the ID of the Dark Red color; set the last argument to false
Configurator.setComponentValue("c01", "color", "col_16", false);
```

Rotating the Model to a Pre-Defined Angle

The **gotoView** method provides means for rotating the model programmatically.
Pass the node object to the gotoView method to initiate the animation.

```
Configurator.gotoView(node);
```

Applying a Product Variant

The Product Variants are pre-defined designs set up in Embodee Platform.

Example: get a list of available variants and apply the first variant in the list:

```
const variants = Configurator.variants;

// apply the first variant in the list
Configurator.applyVariant(variants[0]);
```

Obtaining Product Snapshots

Each product in Embodee Platform is automatically assigned eight turntable views that can be used to generate product snapshots from various angles.

The `takeSnapshot` method is used for generating snapshots:

```
Configurator.takeSnapshot(pixelSize, startFrame, endFrame, totalFrames, imageFormat,
backgroundColor);
```

Example: obtain the front view snapshot of a product at 2048px resolution in PNG format with transparent background

```
const snapshots = await Configurator.takeSnapshot(2048, 0, 0, 8, 'png');
const frontView = snapshots[0];
```

Example: obtain a full set of product snapshots at 120px resolution in JPEG format with white background.

```
const snapshots = await Configurator.takeSnapshot(120, 0, 7, 8, 'jpg',
'#ffffff');
```

Obtaining Print-Ready Files

The createPrintReadyFiles method is used to obtain a set of print-ready files in PNG format with a specified size and background color.

Example: obtain a full set of print-ready files at 4K resolution with white background:

```
const files = await Configurator.createPrintReadyFiles(4096, '#FFFFFF',
{imageFormat: "png"});
```

This method returns an array of base64-encoded images.

Additional options can be passed as an object in the third argument:

```
{
      imageFormat: The image format (png, svg) - default is png;
      dataFormat: The data format of the result (dataUrl, blob) - default is
dataUrl;
      showOnlyGraphicsAndPrints: Show only the graphics and prints on transparent
background (default is false);
      showOutline: Show an outline around the product part (default is false);
            outlineColor: The color of the outline (default is #000000)
}
```

The UI Structure

The "uiStructure" property returns a tree of nodes containing the folders, components, prints and graphics of the product as designed in Embodee Platform. It can be used to build a reactive client UI to drive the customization process.

For more information about the node objects returned by the "uiStructure" property see the "UI Node Types" and "UI Node Structure" sections of this document.