

# Introduction

INVOX Medical SDK offers a set of web components that allows you to **integrate all the available functionality in an agile and simple way**.

INVOX Medical SDK Integrations

These components are available in the `libs/integrations` folder of the SDK.

- *inbox-bar/* : contains the minified code and styles of the **Dictation Bar Component**.
- *inbox-dialogs/* : contains the minified code and styles for the **Dictionary, Templates and Transformations Components**.

Check how to include these components in your web application:

- Dictation Bar Component
- Dictionary Component
- Templates Component
- Transformations Component

## Dictation Bar Component

Dictation Bar Component when session has been started

This component is available in the `libs/integrations/inbox-bar` folder of the SDK.

Dictation Bar is a high-level component **designed to be integrated quickly and easily** in your web application. This web component is created using the customizing components API.

## Getting started!

### 1. Import resources.

Import the resources in the *head* of your `index.html`.

- *inbox.min.js*: contains the API of INVOX Medical.
- *inbox-bar-component.min.js*: contains the Dictation Bar Component integration.
- *inbox-bar-component.css*: contains the styles of the Dictation Bar Component integration.

```
<!-- INVOX Medical SDK - API -->
```

```
<script type="text/javascript" src="libs/inbox.min.js" charset="UTF-8"></script>
```

```
<!-- INVOX Medical SDK - Dictation Bar Component -->
```

```
<script type="text/javascript" src="libs/integrations/inbox-bar/inbox-bar-component.min.js" charset="UTF-8"></script>
```

```
<link rel="stylesheet" href="libs/integrations/inbox-bar/inbox-bar-component.css">
```

### 2. Initialize the component.

1. Create a container in the DOM with a unique identifier, for example “invox-bar”:

```
<div id="invox-bar"></div>
```

2. Create the component:

```
// The following method creates the component in the DOM under the container with id "invox-bar"
const invoxbar = INVOXMDComponent_Bar.create("invox-bar");

// And then you can print the component
invoxbar.show();
```

If we check the DOM we can see the Dictation Bar Component under the established container.

Dictation Bar Component element in the DOM

### 3. Login.

Connect to the INVOX Medical service by passing the credentials and connection parameters. The following code is implemented to attempt connection to the Local Dictation Service first. If no service is found locally, it then tries to connect to the Remote Dictation Service. **This approach allows for automated integration with both services.**

```
const credentials = {
  user: "provided_username",
  password: "provided_password"
}

const connectToLocal = {
  host: "localhost",    // Default host when "useDictationService" is false
  port: "8443",        // Default port
  useDictationService: false
}

const connectToRemote = {
  host: "dev.invoxmedical.com",
  port: "8443",
  useDictationService: true
}

try {
  await invoxbar.login(credentials, connectToLocal);
  INVOX.LogInfo("Local Dictation Service connected successfully.");
} catch (error) {
  INVOX.LogWarn("Local Dictation Service not found. Trying to connect to Remote Dictation Service...");
}
```

```

    try {
      await invoxbar.login(credentials, connectToRemote);
      INVOX.LogInfo("Remote Dictation Service connected successfully.");
    } catch (error) {
      INVOX.LogError("Remote Dictation Service not found. Make sure that connection parameters are correct.");
    }
  }
}

```

#### 4. Logout.

Disconnect from INVOX Medical service.

```

invoboxbar.logout()
  .then((result) => console.log("Successful logout!"))
  .catch((error) => console.error(error));

```

## Public methods

### Create

Function	Description	Parameters	Returns
<b>create</b>	Creates the element in the DOM. <b>Not visible to user by default.</b>	String	void

**Description:** The *create* function allows the developer to create Dictation Bar Component in the DOM under a specified element.

### Parameters:

- **containerId:** String. Container element identification.

### Usage:

```

// Create element
const invoboxbar = new INVOXMDComponent_Bar.create("invobox-bar");

```

### Show

Function	Description	Parameters	Returns
<b>show</b>	Displays component in the user interface.	-	<b>void</b>

**Description:** The ***show*** function allows the developer to show Dictation Bar Component when required.

**Parameters:** No parameters required.

**Usage:**

```
// Show element
inboxbar.show();
```

## Hide

Function	Description	Parameters	Returns
<b>hide</b>	Hides Dictation Bar Component to user.	-	<b>void</b>

**Description:** The ***hide*** function allows developer to hide Dictation Bar Component if required.

**Parameters:** No parameters required.

**Usage:**

```
// Hide element
inboxbar.hide();
```

## Remove

Function	Description	Parameters	Returns
<b>remove</b>	Removes the element from the DOM.	-	<b>void</b>

**Description:** The ***remove*** function allows developer to remove the Dictation Bar Component from the DOM if required.

**Parameters:** No parameters required.

## Usage:

```
// Remove element
inboxbar.remove();
```

## Login

Function	Description	Parameters	Returns
<b>login</b>	Connect to INVOX Medical service.	Object, Object	Promise<void>

**Description:** The *login* function allows developer to start user session in INVOX Medical with user credentials and .

## Parameters:

- **credentials:** Object. Contains the user's credentials.
  - **user:** String. Contains the login name of the user.
  - **password:** String. Contains the user's password.
- **connectionConfig:** Object. Contains the parameters necessary to establish the connection with the dictation service.
  - **host:** String. Sets the host.
  - **port:** String. Sets the port.
  - **useDictationService:** Boolean. If **True** connects to the Remote Dictation Service, else connects to the Local Dictation Service.

## Usage:

```
const credentials = {
  user: "example",
  password: "example",
}
const connectionConfig = {
  host: "example.cloud.net",
  port: "8443",
  useDictationService: true
}

// Connect to INVOX Medical service
inboxbar.login(credentials, connectionConfig)
  .then(() => console.log("Successful login!"))
  .catch((e) => console.error(e));
```

## Logout

Function	Description	Parameters	Returns
<b>logout</b>	Disconnect from INVOX Medical service.	-	Promise<boolean>

**Description:** Save user session and disconnect from INVOX Medical service.

**Parameters:** No parameters required.

**Usage:**

```
inboxbar.logout()  
  .then((result) => console.log("Successful logout!"))  
  .catch((e) => console.error(e));
```

## Customize design

At the beginning of the Dictation Bar Component CSS file `libs/integrations/invox-bar/invox-bar-component.css` you can find the `:root` selector with some variables for easy customization.

Feel free to modify this stylesheet to adjust the component to your web design.

Variable	Description	Default value
<code>-invox-bar-size__width</code>	Width	100%
<code>-invox-bar-size__height</code>	Height	30px
<code>-invox-bar-size__padding</code>	Padding	2px
<code>-invox-bar-size__border-radius</code>	General border radius	5px
<code>-invox-bar-size__inner-border-radius</code>	Border radius from some internal components	3px
<code>-invox-bar-size__vumeter-width</code>	Volume indicator width	50px
<code>-invox-bar-color__primary</code>	Primary color	#52619f
<code>-invox-bar-color__primary-hover</code>	Primary hover color	#415491
<code>-invox-bar-color__secondary</code>	Secondary color	#6d7cba
<code>-invox-bar-color__error</code>	Error color	#ff9191
<code>-invox-bar-color__microphone</code>	Microphone button color	red
<code>-invox-bar-color__microphone-hover</code>	Microphone button hover color	#e30000
<code>-invox-bar-color__microphone-disabled</code>	Microphone button disabled color	#ff6060
<code>-invox-bar-color__microphone-focus</code>	Microphone button focus color	#f3a2b4
<code>-invox-bar-color__vumeter-bars</code>	Volume indicator bars color	white

Variable	Description	Default value
-invox-bar-color___vumeter-bars-active	Volume indicator color when recognizing	#3adf3a
-invox-bar-font___color	Font color	white
-invox-bar-font___family	Font family	sans-serif, Arial, Verdana, “Trebuchet MS”
-invox-bar-font___size	Font size	0.9rem
-invox-bar-icons___size	Icons size	1rem
-invox-bar-icons___color	Icons color	white
-invox-bar-menu-font___color	Menu font color	-invox-bar-color___primary
-invox-about-dialog___z-index	About dialog z-index	700
-invox-profile-dialog___z-index	Profile dialog z-index	700
-invox-dialogs-color___primary	Dialogs primary color	#52619f
-invox-dialogs-color___primary-hover	Dialogs primary hover color	#415491
-invox-dialogs-size___width	Common dialog width	400px
-invox-dialogs-size___height	Common dialog height	400px
-invox-dialogs-size___min-width	Minimum common dialog width	300px
-invox-dialogs-size___min-height	Minimum common dialog height	400px
-invox-dialogs-size___max-width	Maximum common dialog height	100vw
-invox-dialogs-size___max-height	Maximum common dialog height	100vh
-invox-dialogs-size___border-radius	Common dialog border radius	5px

## How to use

### Dictation Bar Component elements

1. **Loading progress percentage.** Displays the load percentage at login.
2. **Progress status.** Displays messages about the status of the session initialization.
3. **Dictation button (Recognizer).** Allows you to start or stop voice recognition.
4. **Audio level indicator.** Displays the audio level activity during the session.
5. **Status bar & Hypthesis viewer.** Displays messages about the status of the session and during dictation shows the recognized hypothesis.
6. **Dictionary button.** Displays dictionary dialog to allow user to create, read, update and remove (CRUD) words.
7. **Templates button.** Displays templates dialog to allow user to create, read, update and remove (CRUD) templates.
8. **Transformations button.** Displays transformations dialog to allow user to create, read, update and remove (CRUD) transformations.
9. **Menu button.** Displays more features.
10. **Profile button.** Displays the user information.

11. **Help button.** Redirect to user help guide.
12. **Support button.** Redirect to *INVOX Medical* support center.
13. **About button.** Displays information about *INVOX Medical*.
14. **Adaptation indicator.** Displays the status of the recognition model adaptation process.
  - 14.a. **Loading status.** The adaptation process is in progress.
  - 14.b. **Finished status.** The adaptation process has finished successfully.

## How to dictate

Dictation Bar Component microphone button

1. Click on the Dictation button to start recognizer.
2. Dictate your medical report...
3. Click again on the Dictation button to stop recognizer.

## How to create words

Open the words management view by clicking on the Dictionary button in the Dictation Bar Component.

1. Click on the “New” button to open the creation view.
2. Fill the gap with the new word.
3. Click on the “Save” button to save changes.

## How to create templates

Open the templates management view by clicking on the Templates button in the Dictation Bar Component.

1. Click on the “New” button to open the creation view.
2. Fill the gaps with the new template.
3. Click on the “Save” button to save changes.

## How to create transformations

Open the transformation management view by clicking on the Transformations button in the Dictation Bar Component.

1. Click on the “New” button to open the creation view.
2. Fill the gaps with the new transformation.
3. Click on the “Save” button to save changes.



## Reset component at runtime

In some cases, it may be necessary to regenerate the Dictation Bar without completely reloading the page, especially in single-page applications (SPA).

To do this you must be able to capture the exact moment where the error occurs and then depending on your needs you can for example remove the Dictation Bar instance from your page and restart it using the create and show methods.

```
function resetComponent(credentials, connectionConfig) {  
  
    // Step 1. Logout is absolutely necessary even if an error has occurred in the session.  
    invoxbar.logout()  
        .then((result) => console.log("Successful logout!"))  
        .catch((e) => console.error(e))  
        .finally(() => {  
  
        // Step 2. When logout finishes we can remove from the DOM.  
        invoxbar.remove();  
  
        // Step 3. Create the dictation bar again.  
        invoxbar = new INVOXMDComponent_Bar.create("invox-bar");  
  
        // Step 4. Connect to INVOX Medical service again.  
        invoxbar.login(credentials, connectionConfig)  
            .then(() => console.log("Successful login!"))  
            .catch((e) => console.error(e));  
    })  
}
```

## Dictionary Component

Dictionary dialog main view

This component is available in the `libs/integrations/invox-dialogs` folder of the SDK.

The Dictionary Component is a modal window that appears in front of the application content to allow word management. This high-level component **is designed to be integrated quickly and easily** in your web application.

### Getting started!

#### 1. Import resources.

Import the resources in the *head* of your `index.html`.

- *inbox.min.js*: contains the API of INVOX Medical.
- *inbox-dictionary-component.min.js*: contains the Dictionary Component integration.
- *inbox-dictionary-component.css*: contains the styles of the Dictionary Component integration.

```
<!-- INVOX Medical SDK - API -->
<script type="text/javascript" src="libs/inbox.min.js" charset="UTF-8"></script>

<!-- INVOX Medical SDK - Dictionary Component -->
<script type="text/javascript" src="libs/integrations/inbox-dialogs/inbox-dictionary-component.min.js" charset="UTF-8"></script>
<link rel="stylesheet" href="libs/integrations/inbox-dialogs/inbox-dictionary-component.css">
```

## 2. Initialize component.

1. Create a container in the DOM with a unique identifier, for example “inbox-dialogs-container”. **You can use the same container for all dialogs:**

```
<div id="inbox-dialogs-container"></div>
```

2. Create the component:

```
// The following method creates the element in the DOM under the container with id "inbox-dialogs-container"
const dictionaryDialog = INVOXMDCComponent_Dictionary.create("inbox-dialogs-container");
```

## 3. Show component.

By default the component is hidden. You must call the following function to print the component in the user interface:

```
// The following method displays the component to the user
dictionaryDialog.show();
```

## 4. Hide component (optional).

This method is not required. **The dialog can be closed by pressing ESC or by clicking the Close button at the header.** But if required, you can call the following function to force it to close.

```
// The following method hides the component to the user
dictionaryDialog.hide();
```

## 5. Remove component (optional).

This method is not required. **The dialog element can be removed from the DOM.** You should call the following function to remove it.

```
// The following method removes the element from the DOM
dictionaryDialog.remove();
```

## Public methods

### Create

Function	Description	Parameters	Returns
<b>create</b>	Creates the element in the DOM. <b>Not visible to user by default.</b>	<b>String</b>	<b>void</b>

**Description:** The *create* function allows the developer to create Dictionary Component in the DOM under a specified element.

**Parameters:**

- **containerId:** **String.** Container element identification.

**Usage:**

```
// Create element
const dictionaryDialog = INVOXMDComponent_Dictionary.create("invox-dialogs-container");
```

**Show**

Function	Description	Parameters	Returns
<b>show</b>	Displays component in the user interface.	-	<b>void</b>

**Description:** The *show* function allows the developer to show the Dictionary Component when required.

**Parameters:** No parameters required.

**Usage:**

```
// Show element
dictionaryDialog.show();
```

**Hide**

Function	Description	Parameters	Returns
<b>hide</b>	Hides component in the user interface.	-	<b>void</b>

**Description:** The *hide* function allows developer to hide the Dictionary Component if required.

**Parameters:** No parameters required.

**Usage:**

```
// Hide element  
dictionaryDialog.hide();
```

**Remove**

Function	Description	Parameters	Returns
<b>remove</b>	Removes the element from the DOM.	-	<b>void</b>

**Description:** The *remove* function allows developer to remove the Dictionary Component from the DOM if required.

**Parameters:** No parameters required.

**Usage:**

```
// Remove element  
dictionaryDialog.remove();
```

## Customize design

At the beginning of the Dictionary Component CSS file `libs/integrations/invox-dialogs/invox-dictionary-component.css` you can find the `:root` selector with some variables for easy customization.

Feel free to modify this stylesheet to adjust the component to your web design.

Variable	Description	Default value
<code>--invox-dictionary-dialog__z-index</code>	Dictionary dialog z-index	700
<code>--invox-dialogs-color__primary</code>	Primary color	#52619f
<code>--invox-dialogs-color__primary-hover</code>	Primary hover color	#415491
<code>--invox-dialogs-size__width</code>	Common dialog width	400px
<code>--invox-dialogs-size__height</code>	Common dialog height	400px
<code>--invox-dialogs-size__min-width</code>	Minimum common dialog width	300px
<code>--invox-dialogs-size__min-height</code>	Minimum common dialog height	400px
<code>--invox-dialogs-size__max-width</code>	Maximum common dialog height	100vw
<code>--invox-dialogs-size__max-height</code>	Maximum common dialog height	100vh

Variable	Description	Default value
<code>-inbox-dialogs-size__border-radius</code>	Common dialog border radius	5px

## How to use

Dictionary Component elements

1. **Help button.** Displays the help view.
2. **Fullscreen button.** Enlarges the view to full screen.
3. **Close button.** Close the dialog.
4. **List selector.** Displays user items, speciality items or all items.
5. **Search bar.** Allows user to search items by the element name.
6. **Shared item.** Icon indicating that the word is shared with users of the same speciality.
7. **Edit button.** Navigates to the edition view.
8. **Close button.** Close the dialog.
9. **New button.** Navigates to the creation view.
10. **Resize dialog.** Allows user to resize dialog.

## How to create words

1. Click on the “New” button to navigate to the creation view.
2. Fill the gap with the new word.
3. Click on the “Save” button to save changes.

## How to remove words

1. Select the word you want to remove.
2. Click on the “Edit” button to navigate to the edition view.
3. Click on the “Remove” button.
4. Click again on the “Remove” button to confirm.

## How to edit words

1. Select the word you want to edit.
2. Click on the “Edit” button to navigate to the edition view.
3. Edit the word. . .

4. Click on the “Save” button to save changes.

## Templates Component

Templates Component main view

This component is available in the `libs/integrations/invox-dialogs` folder of the SDK.

The Templates Component is a modal window that appears in front of the application content to allow templates management. This high-level component **is designed to be integrated quickly and easily** in your web application.

### Getting started!

1. Import resources.

Import the resources in the *head* of your `index.html`.

- *invox.min.js*: contains the API of INVOX Medical.
- *invox-templates-component.min.js*: contains the Templates Component integration.
- *invox-templates-component.css*: contains the styles of the Templates Component integration.

```
<!-- INVOX Medical SDK - API -->
<script type="text/javascript" src="libs/invox.min.js" charset="UTF-8"></script>

<!-- INVOX Medical SDK - Templates Component -->
<script type="text/javascript" src="libs/integrations/invox-dialogs/invox-templates-component.min.js" charset="UTF-8"></script>
<link rel="stylesheet" href="libs/integrations/invox-dialogs/invox-templates-component.css">
```

2. Initialize component.

1. Create a container in the DOM with a unique identifier, for example “invox-dialogs-container”. **You can use the same container for all dialogs:**

```
<div id="invox-dialogs-container"></div>
```

2. Create the component:

```
// The following method creates the element in the DOM under the container with id "invox-dialogs-container"
const templatesDialog = INVOXMDCComponent_Templates.create("invox-dialogs-container");
```

3. Show component.

By default the component is hidden. You must call the following function to print the component in the user interface:

```
// The following method hides the component to the user
templatesDialog.show();
```

#### 4. Hide component (optional).

This method is not required. **The dialog can be closed by pressing ESC or by clicking the Close button at the header.** But if required, you can call the following function to force it to close.

```
// The following method creates the dialog in DOM and shows to the user
templatesDialog.hide();
```

#### 5. Remove component (optional).

This method is not required. **The dialog element can be removed from the DOM.** You should call the following function to remove it.

```
// The following method removes the element from the DOM
templatesDialog.remove();
```

## Public methods

### Create

Function	Description	Parameters	Returns
<b>create</b>	Creates the element in the DOM. <b>Not visible to user by default.</b>	String	void

**Description:** The *create* function allows the developer to create Templates Component component in the DOM under a specified element.

### Parameters:

- **containerId:** String. Container element identification.

### Usage:

```
// Create element
const templatesDialog = INVOXMDComponent_Templates.create("invox-dialogs-container");
```

### Show

Function	Description	Parameters	Returns
<b>show</b>	Displays component in the user interface.	-	void

**Description:** The *show* function allows the developer to show the Templates Component when required.

**Parameters:** No parameters required.

**Usage:**

```
// Show element
templatesDialog.show();
```

## Hide

Function	Description	Parameters	Returns
<b>hide</b>	Hides component in the user interface.	-	<b>void</b>

**Description:** The ***hide*** function allows developer to hide the Templates Component if required.

**Parameters:** No parameters required.

**Usage:**

```
// Hide element
templatesDialog.hide();
```

## Remove

Function	Description	Parameters	Returns
<b>remove</b>	Removes the element from the DOM.	-	<b>void</b>

**Description:** The ***remove*** function allows developer to remove the Templates Component from the DOM if required.

**Parameters:** No parameters required.

**Usage:**

```
// Remove element
templatesDialog.remove();
```



## Customize design

At the beginning of the Templates Component CSS file `libs/integrations/invox-dialogs/invox-templates-component.css` you can find the `:root` selector with some variables for easy customization.

Feel free to modify this stylesheet to adjust the component to your web design.

Variable	Description	Default value
<code>-invox-templates-dialog__z-index</code>	Templates Component z-index	700
<code>-invox-dialogs-color__primary</code>	Primary color	<code>#52619f</code>
<code>-invox-dialogs-color__primary-hover</code>	Primary hover color	<code>#415491</code>
<code>-invox-dialogs-size__width</code>	Common dialog width	400px
<code>-invox-dialogs-size__height</code>	Common dialog height	400px
<code>-invox-dialogs-size__min-width</code>	Minimum common dialog width	300px
<code>-invox-dialogs-size__min-height</code>	Minimum common dialog height	400px
<code>-invox-dialogs-size__max-width</code>	Maximum common dialog height	100vw
<code>-invox-dialogs-size__max-height</code>	Maximum common dialog height	100vh
<code>-invox-dialogs-size__border-radius</code>	Common dialog border radius	5px

## How to use

Templates Component elements

1. **Help button.** Displays the help view.
2. **Fullscreen button.** Enlarges the view to full screen.
3. **Close button.** Close the dialog.
4. **Search bar.** Allows user to search items by the element name or content.
5. **Use button.** Allows the user to use the template without having to dictate it.
6. **Edit button.** Navigates to the edition view.
7. **List selector.** Displays user items, speciality items or all items.
8. **Text template name.** The name of the text template. *What the user has to say.*
9. **Text template replacement.** The replacement of the text template. *What the user expects.*
10. **Shared item.** Icon indicating that the item is shared with users of the same speciality.
11. **No pause required.** Icon indicating that it is not necessary to pause during dictation to activate the template.
12. **Close button.** Close the dialog.

13. **New button.** Navigates to the creation view.
14. **Resize dialog.** Allows user to resize dialog.

### How to create templates

1. Click on the “New” button to navigate to the creation view.
2. Fill gaps of the new template.
3. Click on the “Save” button to save changes.

### How to remove templates

1. Select the template you want to remove.
2. Click on the “Edit” button to navigate to the edition view.
3. Click on the “Remove” button.
4. Click again on the “Remove” button to confirm.

### How to edit templates

1. Select the template you want to edit.
2. Click on the “Edit” button to navigate to the edition view.
3. Edit the template...
4. Click on the “Save” button to save changes.

## Transformations Component

Transformations Component main view

This component is available in the `libs/integrations/invox-dialogs` folder of the SDK.

The Transformations Component is a modal window that appears in front of the application content to allow transformations management. This high-level component **is designed to be integrated quickly and easily** in your web application.

### Getting started!

1. **Import resources.**

Import the resources in the *head* of your `index.html`.

- *invox.min.js*: contains the API of INVOX Medical.
- *invox-transformations-component.min.js*: contains the Transformations Component integration.
- *invox-transformations-component.css*: contains the styles of the Transformations Component integration.

```

<!-- INVOX Medical SDK - API -->
<script type="text/javascript" src="libs/invox.min.js" charset="UTF-8"></script>

<!-- INVOX Medical SDK - Transformations Component -->
<script type="text/javascript" src="libs/integrations/invox-dialogs/invox-transformations-component.min.js" charset="UTF-8"></script>
<link rel="stylesheet" href="libs/integrations/invox-dialogs/invox-transformations-component.css">

```

## 2. Initialize component.

1. Create a container in the DOM with a unique identifier, for example “invox-dialogs-container”. **You can use the same container for all dialogs:**

```
<div id="invox-dialogs-container"></div>
```

2. Create the component:

```

// The following method creates the element in the DOM under the container with id "invox-dialogs-container"
const transformationsDialog = INVOXMDComponent_Transformations.create("invox-dialogs-container");

```

## 3. Show component.

By default the component is hidden. You must call the following function to print the component in the user interface:

```

// The following method displays the component to the user
transformationsDialog.show();

```

## 4. Hide component (optional).

This method is not required. **The dialog can be closed by pressing ESC or by clicking the Close button at the header.** But if required, you can call the following function to force it to close.

```

// The following method hides the component to the user
transformationsDialog.hide();

```

## 5. Remove component (optional).

This method is not required. **The dialog element can be removed from the DOM.** You should call the following function to remove it.

```

// The following method removes the element from the DOM
transformationsDialog.remove();

```

## Public methods

### Create

Function	Description	Parameters	Returns
<b>create</b>	Creates the element in the DOM. <b>Not visible to user by default.</b>	<b>String</b>	<b>void</b>

**Description:** The *create* function allows the developer to create Transformations Component component in the DOM under a specified element.

**Parameters:**

- **containerId:** **String.** Container element identification.

**Usage:**

```
// Create element
const transformationsDialog = INVOXMDComponent_Transformations.create("invoy-dialogs-container");
```

**Show**

Function	Description	Parameters	Returns
<b>show</b>	Displays component in the user interface.	-	<b>void</b>

**Description:** The *show* function allows the developer to show the Transformations Component when required.

**Parameters:** No parameters required.

**Usage:**

```
// Show element
transformationsDialog.show();
```

**Hide**

Function	Description	Parameters	Returns
<b>hide</b>	Hides component in the user interface.	-	<b>void</b>

**Description:** The *hide* function allows developer to hide the Transformations Component if required.

**Parameters:** No parameters required.

**Usage:**

```
// Hide element
transformationsDialog.hide();
```

**Remove**

Function	Description	Parameters	Returns
<b>remove</b>	Removes the element from the DOM.	-	<b>void</b>

**Description:** The *remove* function allows developer to remove the Transformations Component from the DOM if required.

**Parameters:** No parameters required.

**Usage:**

```
// Remove element
transformationsDialog.remove();
```

## Customize design

At the beginning of the Transformations Component CSS file `libs/integrations/invox-dialogs/invox-transformations-component.css` you can find the `:root` selector with some variables for easy customization.

Feel free to modify this stylesheet to adjust the component to your web design.

Variable	Description	Default value
<code>-invox-transformations-dialog__z-index</code>	Transformations Component z-index	700
<code>-invox-dialogs-color__primary</code>	Primary color	#52619f
<code>-invox-dialogs-color__primary-hover</code>	Primary hover color	#415491
<code>-invox-dialogs-size__width</code>	Common dialog width	400px
<code>-invox-dialogs-size__height</code>	Common dialog height	400px
<code>-invox-dialogs-size__min-width</code>	Minimum common dialog width	300px
<code>-invox-dialogs-size__min-height</code>	Minimum common dialog height	400px
<code>-invox-dialogs-size__max-width</code>	Maximum common dialog height	100vw
<code>-invox-dialogs-size__max-height</code>	Maximum common dialog height	100vh

Variable	Description	Default value
<code>-inbox-dialogs-size___border-radius</code>	Common dialog border radius	5px

## How to use

Transformations Component elements

1. **Help button.** Displays the help view.
2. **Fullscreen button.** Enlarges the view to full screen.
3. **Close button.** Close the dialog.
4. **Search bar.** Allows user to search items by the element name or content.
5. **Edit button.** Navigates to the edition view.
6. **Transformation name.** The name of the transformation. *What the user says.*
7. **Transformation replacement.** The replacement of the transformation. *What the user expects.*
8. **Close button.** Close the dialog.
9. **New button.** Navigates to the creation view.
10. **Resize dialog.** Allows user to resize dialog.

## How to create transformations

1. Click on the “New” button to navigate to the creation view.
2. Fill gaps of the new transformation.
3. Click on the “Save” button to save changes.

## How to remove transformations

1. Select the transformation you want to remove.
2. Click on the “Edit” button to navigate to the edition view.
3. Click on the “Remove” button.
4. Click again on the “Remove” button to confirm.

## How to edit transformations

1. Select the transformation you want to edit.
2. Click on the “Edit” button to navigate to the edition view.
3. Edit the transformation. . .

4. Click on the “Save” button to save changes.