UI customization

This guide covers concepts, configuration, and usage procedures for customizing the ForgeRock Access Management user interface.

Read this to apply your own look and feel to the end-user facing pages provided by Access Management.

About the user interface

Learn about the user interface (UI), which is split into the User UI and the Admin UI.

Download the UI

Download, rebuild, and deploy the UI project to customize.

Theme the UI

Learn how to change the look of the user-facing pages of the UI.

Deploy the UI

Build, test, and deploy the UI.

ForgeRock® Identity Platform serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see https://www.forgerock.com.

About the UI

AM includes the following browser-based UIs:

Admin UI
: Pages related to the administration of an AM server. Administrative login is delegated to the user UI.

User UI
: End user pages and login pages.

When you deploy AM to protect applications, you can redirect users to AM pages for login and logout. The end user pages have default styling and branding but are fully customizable. You can theme them, localize them, and change their layout.

To customize the UI, download the source code and change it to suit your environment. Deploy the modified UI in your AM instances as part of your environment pipelines or package it in your custom AM WAR files, then deploy them.

Customizing the page layout or the JavaScript resources requires a complete rebuild of the UI. You can make other customizations, such as localizing or theming the UI, by modifying the WAR file.

The UI contains the following types of resources:

UI resources

Resources	Location (Admin UI)	Location (User UI)
JavaScript source and configuration files	`openam-ui-admin/src/js`	`openam-ui-user/src/js`
CSS collections	`openam-ui-admin/src/resources/css`	`openam-ui-user/src/resources/css`
Fonts	`openam-ui-admin/src/resources/fonts`	`openam-ui-user/src/resources/fonts`
Images	`openam-ui-admin/src/resources/images`	`openam-ui-user/src/resources/images`
Localization files	`openam-ui-admin/src/resources/locales`	`openam-ui-user/src/resources/locales`
Themes	N/A	`openam-ui-user/src/resources/themes`

Resources	Location (Admin UI)	Location (User UI)
Partials	`openam-ui-admin/src/resources/partials`	`openam-ui-user/src/resources/themes/default/partials`
HTML pages	`openam-ui-admin/src/resources/templates`	`openam-ui-user/src/resources/themes/default/templates`

Download the UI source

To customize the layout and functionality of the UI, you must download, rebuild, and deploy the UI project. The build process for the UI uses the *Webpack* resource bundler to manage dependencies, optimize deliverables, and package the output.

Ping Identity provides the source code for the UI as a Maven project in the link:https://github .com/ForgeRock/am-external[am-external] public GitHub repository.

Follow these steps to get the UI source code:

1. Clone the `am-external` repository:

```
$ git clone https://github.com/ForgeRock/am-external.git
```

2. Check out the `releases/7.2.0` branch:

```
$ cd am-external
$ git checkout releases/7.2.0
```

   The End User UI project is in the `am-external/openam-ui` folder.

3. If you don't already have them, install the following prerequisites:
   - Yarn.
   - Node.js (12.x.x).

   TIP

   You can find information on building the UI as part of a Maven workflow for deployment inside a WAR file in How do I customize the XUI using source code in PingAM?.

4. Use the **`yarn`** command to download the dependencies to the project:

```
$ cd openam-ui/openam-ui-user
$ yarn install
[1/4] 🔍 Resolving packages…
[2/4] 🔍 Fetching packages…
[3/4] 🔍 Linking dependencies…
[4/4] 🔍 Building fresh packages…
✨ Done in 9.08s.
```

TIP

In some environments you might receive an error such as `gyp ERR! not ok` when downloading dependencies.

You can ignore such errors as they apply only to optional components in certain environments. You can also use the `yarn install --ignore-optional` command to suppress these errors.

Localize AM

ForgeRock provides the user-facing and administration UI pages in English only; however, you can customize and localize the user-facing text as required.

The `.json` files in the `openam-ui-user/src/resources/locales/en` directory provide the English text.

To localize the user-facing text to a new locale:

1. Copy the English locale directory (`locales/en`) to a new directory, for example, `locales/fr` .

   The name of the directory should follow RFC 5646 - Tags for Identifying Languages. For example, `en-GB` .

2. Edit the files and change the values of the elements for the required locale. Take care not to change the JSON structure or to render it invalid.
3. Rebuild the UI. For more information, see Test and deploy the UI.

   You can now redeploy the UI or pack it in your custom AM `.war` file.

Theme AM

The UI is built with the Bootstrap framework, and supports Bootstrap themes to customize the look and feel of the user interface.

Only user-facing UI pages support themes. The administration pages cannot be themed, although customizing the footer would alter the appearance for both user-facing and administration pages.

You can apply themes to specific realms, and also to specific authentication chains within those realms. AM includes a *default* theme, and an inverted *dark* theme.

Apply a theme to the UI

You can perform the steps of this procedure and edit the UI source code, or you can find the files as deployed inside the WAR file. The examples in the procedure use the source code paths, but you can find the referenced files in the `/path/to/tomcat/webapps/openam/XUI` directory of a deployed AM WAR file.

If you modify resources from the WAR file, you will see they have a hash value appended to them. For example, `ThemeConfiguration.`*`hash`*`.js`. The *hash* value is an alphanumeric value generated each time the UI is rebuilt, to reference the CSS files in the theme, and to map the theme to realms and authentication services.

Perform the following steps to apply a theme to the UI:

1. Copy your custom Bootstrap theme to a directory in `openam-ui-user/src/resources/themes`. A custom Bootstrap theme should consist of one or more CSS files, and optionally, media and font files.

   As an example, the *dark* theme is available in the `openam-ui-user/src/resources/themes/dark` directory.

2. Edit the `openam-ui-user/src/js/config/ThemeConfiguration.js` file:

   - Locate the `themes` element, and under it create a new element with the name of your theme.

     The following example adds a theme called `myTheme`:

     ```
     return {
         themes: {
             // There must be a theme named "default".
             "default": { ...
             },
             "fr-dark-theme": { ...
             },
             "myTheme": {}
         },
         mappings: [...]
     };
     ```

- In the new theme element, create a `stylesheets` array containing the theme's two CSS files, followed by the required `css/structure.css` file.

```
return {
    themes: {
        // There must be a theme named "default".
        "default": { ...
        },
        "fr-dark-theme": { ...
        },
        "myTheme": {
            stylesheets: [
                "themes/myTheme/css/bootstrap.min.css",
                "themes/myTheme/css/myTheme.css",
                "css/structure.css"
            ]
        }
    },
    mappings: [...]
};
```

  Note that you must specify paths relative to the `openam-ui-user/src/resources` directory.

  If required, specify additional settings specific to the new theme, such as the logos to use or the footer information. For information on the available settings, see ThemeConfiguration.js reference.

- Locate the `mappings` array, and create a new element under it to map your new theme to realms and authentication chains.

  Elements in the `mappings` array are evaluated in order from top to bottom. The first theme that matches the current realm and/or authentication chain is applied. Any subsequent mappings, even if true, are ignored once a match is found.

  If no match is found, the `default` theme is applied.

  - Create a `theme` element, and set the value to the name of your new theme:

```
return {
    themes: { ...
    },
    mappings: [{
        theme: "myTheme"
```

```
    }]
};
```

- Create a `realms` array, and include the realms the theme will apply to:

```
return {
    themes: { ...
    },
    mappings: [{
        theme: "myTheme",
        realms: ["/", "/test-realm", /^\/a/]
    }]
};
```

  You can use a regular expression to specify the realms the theme should apply to. For example `/^\/a/` will apply the theme to all realms that start with `/a`, including `/ab` and `/a/c`.

  If you do not include a realms array, the theme is applied to all realms.

- Create an `authenticationChains` array, and include any authentication service (chains or trees) the theme applies to when used:

```
return {
    themes: { ...
    },
    mappings: [{
        theme: "myTheme",
        realms: ["/", "/test-realm", /^\/a/],
        authenticationChains: ["auth-chain-one",
"Example-Tree"]
    }]
};
```

  If you specify both realms and authentication services, the theme is only applied when both criteria are true.

3. Compile the UI project using the **`yarn build`** command.
4. Start the development UI server to test the changes by following the steps detailed in To Test UI Pages in a Development Server.
5. Log in as a user to see the new theme applied:

Figure 1. UI with the Dark Theme

6. Once you are satisfied with the changes, deploy the output in the `build` directory to the `/path/to/tomcat/webapps/openam/XUI` directory of your AM instances.

   There is no need to restart the AM instance. Subsequent visits to the UI pages will use the rebuilt files.

Customize the UI layout

To perform basic customizations, edit the `openam-ui-user/src/js/config/ThemeConfiguration.js` and then rebuild the UI.

▼ ThemeConfiguration.js reference

The file contains a full configuration for the mandatory `default` theme. Additional themes should use a duplicate of the default theme's configuration. Any parameters that are not configured will inherit values from the mandatory `default` theme.

The available parameters for each theme in the file are as follows:

- `themes` : Title; also represents an array of theme objects.
  - *name*: Theme title.

- `stylesheets` : An ordered array of URLs to CSS stylesheet files that are applied to every page. It is highly recommended to include the variable `StructureStyle` as one of the entries to provide default styles for layout and structure.

  For example: `["themes/dark/css/bootstrap.min.css", StructureStyle, "themes/dark/css/theme-dark.css"]`
- `path` : A path relative to the `openam-ui-user/src/resources/themes` folder, that contains `templates` or `partials` directories, used for customizing the default layout of UI pages. The path must include a trailing slash character `/` .

  For example, `["theme-path/"]` .
- `icon` : URL to a resource to use as a favicon.
- `settings` : Configuration settings for the theme. Missing parameters inherit their value from the mandatory `default` theme.
  - `logo` : Parameters for the logo displayed on user profile pages.
    - `src` : Filename of the logo.
    - `title` : HTML `title` attribute of the logo.
    - `alt` : HTML `alt` attribute of the logo.
    - `height` : Logo height in CSS notation. For example: `75px` or `10%` .
    - `width` : Logo width in CSS notation. For example: `150px` or `25%` .
  - `loginLogo` : Parameters for the logo displayed on login pages.
    - `src` : Filename of the logo.
    - `title` : HTML `title` attribute of the logo.
    - `alt` : HTML `alt` attribute of the logo.
    - `height` : Logo height in CSS notation. For example: `75px` or `10%` .
    - `width` : Logo width in CSS notation. For example: `150px` or `25%` .
  - `footer` : Parameters to display in the footer of each UI page.
    - `mailto` : Email address.
    - `phone` : Telephone number.

This procedure demonstrates how to customize the layout of UI pages by copying the relevant files into a theme, editing the source, and making changes to the CSS.

1. Download the UI source code as explained in Download the UI source.

2. Modify the UI source as follows:

   - Copy the `src/resources/themes/dark` directory to a new directory with a suitable name for the theme that contains your customizations, for example, `myTheme` .

     TIP

     You can modify the original files, but it is recommended to create a new theme containing your changes.

   - Edit the `openam-ui-user/src/js/config/ThemeConfiguration.js` file, which contains the configuration the UI uses when applying a theme, and make a copy of the `"fr-dark-theme"` section, named after the directory you created in the previous step (`"myTheme"`).

     In this example, in the `footer` element, change the email address to `Feedback@example.com` , and add a new element called `icon` that points to an icon to display in the footer.

     Alter the `path` property to point to the name of the directory you created in the `themes` directory earlier, including a trailing slash. For example `myTheme/` .

     The new section will resemble the following:

```
"myTheme": {
    // An ordered list of URLs to stylesheets that will be
applied to every page.
    stylesheets: [
        "themes/myTheme/css/bootstrap.min.css",
        StructureStyle,
        "themes/myTheme/css/theme-dark.css"
    ],
    // A path that is prepended to every relative URL when
fetching resources (including images, stylesheets and
    // HTML template files). Must include a trailing
forward slash.
    path: "myTheme/",
    settings: {
        loginLogo: {
            src: "themes/myTheme/images/login-logo-
white.png",
            title: "ForgeRock",
            alt: "ForgeRock",
            height: "228px",
            width: "220px"
        },
```

```
        footer: {
            // A contact email address.
            mailto:"Feedback@example.com",
            // The footer icon.
            icon : "images/logo-horizontal.png"
        }
    }
}
```

IMPORTANT

Specify resource paths within the theme configuration as relative to the `openam-ui-user/src/resources/` directory.

- Edit the `mappings` section of the `ThemeConfiguration.js` file to apply the new theme to the required contexts.

  For example, to apply the theme to the Top Level Realm, use:

  ```
  mappings: [
      { theme: "myTheme", realms: ["/"] }
  ]
  ```

  For more information about the format of the `ThemeConfiguration.js` file, see ThemeConfiguration.js reference.

- Copy the template file that contains the HTML code for the footer, `openam-ui-user/src/resources/themes/default/templates/common/FooterTemplate.html`, into the same path but within the `myTheme` directory, for example:

  ```
  $ cp openam-ui-
  user/src/resources/themes/default/templates/common/FooterT
  emplate.html \
   openam-ui-
  user/src/resources/themes/myTheme/templates/common/FooterT
  emplate.html
  ```

- Add a table, containing an image, in the new `FooterTemplate.html` file, as follows:

  ```
  <div class="container">
      <p>
          <table align="center">
              <td>
                  <img src="{{theme.settings.footer.icon}}"
  alt="Horizontal ForgeRock Logo" align="middle"
  ```

```
class="footer_image">
            </td>
            <td>
                <b>ForgeRock XUI at Example.com</b>
                <br/>
                <a href="mailto:
{{theme.settings.footer.mailto}}">Send us Feedback!</a>
            </td>
        </table>
    </p>
</div>
```

The file contains variables expressed in double curly brackets (`{{}}`); they are the paths defined in the `ThemeConfiguration.js` file.

Notice that the image has the `footer_image` CSS class applied to simplify applying a style to it in the next step.

- Copy and edit any additional files that require customization from the `/themes/default/templates` and `/themes/default/partials` directories to the equivalent path in your `themes` directory.

  NOTE

  AM uses the partials and templates from the `/themes/default` directory if an equivalent file is not found in your customized theme.

- Edit the `/am-external/openam-ui/openam-ui-user/src/resources/themes/myTheme/css/theme-dark.css` file to specify the height of the image, and add padding to the `footer_image` class.

  For example:

```
.footer_image {
    height: 3em;
    padding-right: 1em;
}
```

3. Rebuild the UI by running the **`yarn build`** command.

   TIP

   If you receive errors during the build, you may be missing the required components locally. Try running **`yarn install`** , and then try again.

4. Test the UI pages by following the steps detailed in Test UI pages in a development server.

You can now see the customized template in the login page:

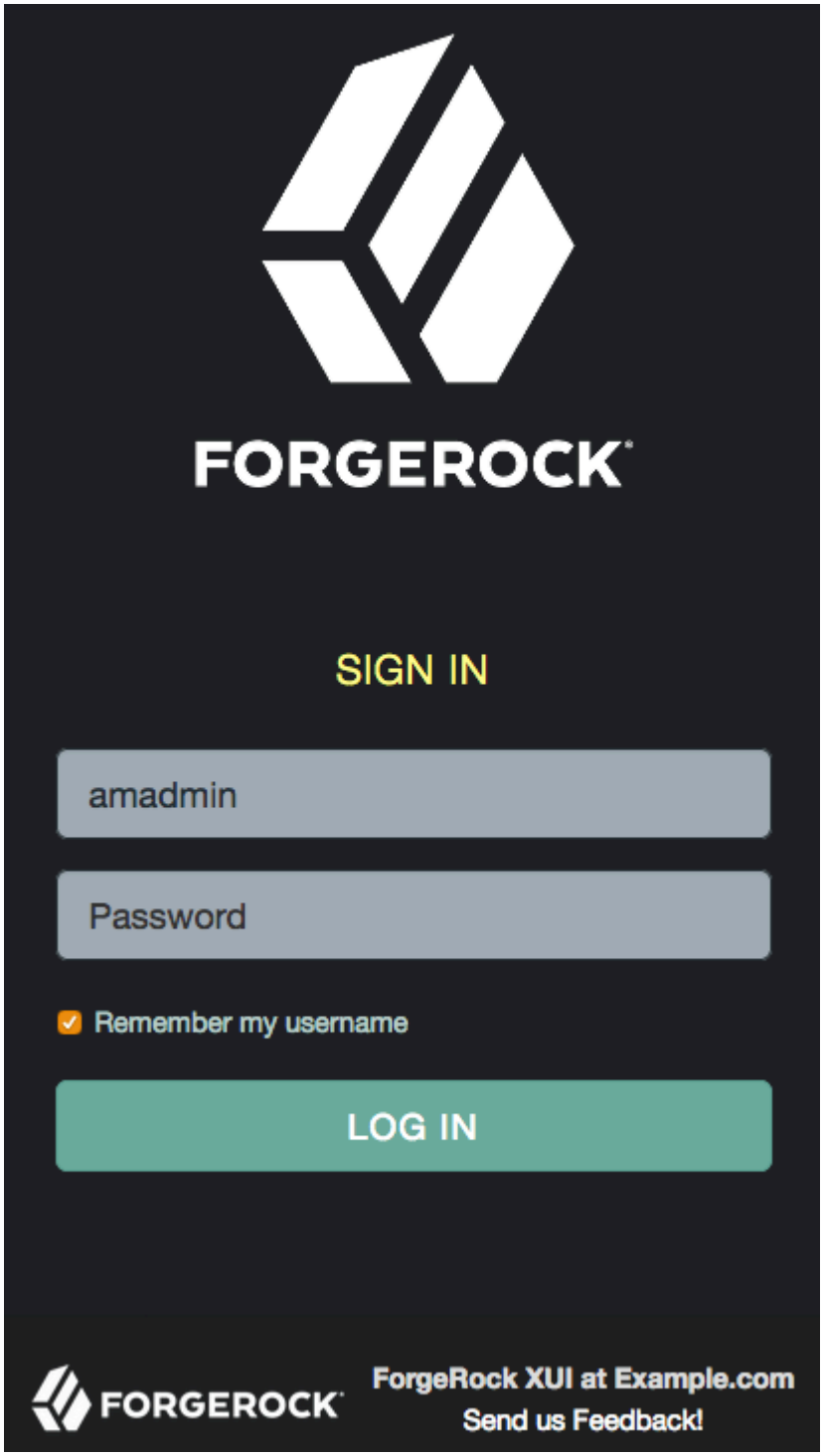

Figure 2. Example of a customized template

5. Once you are satisfied with the changes, deploy the output in the `build` directory to the `/path/to/tomcat/webapps/openam/XUI/` directory of your AM instances.

   There is no need to restart the AM instance. Subsequent visits to the UI pages will use the rebuilt files.

Test and deploy the UI

The Maven project for the UI contains a `package.json` configuration file, which specifies the available commands for building and testing the UI.

▼ Yarn commands available in package.json

`yarn build`
: Performs a one-time production-ready build of the UI. Outputs to the `build` directory.

`yarn test`
: Compiles and executes UI unit tests and continuously watches for source changes to trigger re-testing. Unit tests are located alongside the modules they are testing.

  Run this command in a separate terminal window when developing to continuously run unit tests against the code.

`yarn profile`
: Performs a one-time build of the UI, in production mode, with profiling enabled.

  Generates a report in the `build` directory named `report.html` detailing the structure of the bundles and chunks that are created as part of a production build.

`yarn start`
: Starts the UI in development mode, with automatic rebuilds and reloads enabled.

  The `start` script launches a dedicated Webpack development server to serve the UI during development. For more information, see Test UI pages in a development server.

The `package.json` file also lists the dependencies the UI uses, and the required versions. Dependencies are locked to specific versions. For example, `"lodash": "4.1.0"` specifies an explicit version without using `^` or `~` characters.

TIP

While customizing the UI, you can set the advanced server property, `org.forgerock.openam.core.resource.lookup.cache.enabled`, to `false` to allow AM immediately to pick up changes to the files as you customize them. This includes the XML callback files for authentication modules used by the UI.

Before using AM in production, however, set the property back to `true`.

Test UI pages in a development server

You can run the UI project on a dedicated development server to test customizations. The UI in the development server connects to a separate instance of AM running on a different port, but using the same base domain. Any HTTP requests the UI makes out to AM are proxied to port `8080`, by default.

You can override the default setting for the port AM is running on by using the `OPENAM_PORT` environment variable.

By separating the UI from the core AM server, the UI behaves as it would in production, except with the addition of development tooling, such as automatic browser refreshes when UI code is changed.

To run the UI on a development server:

1. Configure and start an AM instance. For example, `https://openam.example.com:8443/openam`.
2. Start a UI development server by using the **`yarn start`** command.

   The development server starts on this first available port counting up from port `8080`, which is usually port `8081`. Ensure you can access the development server using the same domain as the AM instance. For example, `http://xui.example.com:8081`.
3. In a web browser, go to the full URL of your *AM instance*, but use the port number of the UI development server.

   For example, go to `http://openam.example.com:8081/openam/XUI/#login`.

   Changes made to the UI project are rebuilt and redeployed to the development server automatically, and the browser refreshed to show the changes, or any errors that have occurred.

   NOTE

   The UI development server assumes the AM instaces has a deployment URI of `/openam`. The deployment URI and port numbers can be edited in the `config/webpack/development.js` file in the UI project.

Rebuild and deploy the UI

After making changes to the UI, such as editing the JavaScript or HTML templates, perform the following steps:

1. Rebuild the project using the **`yarn build`**.
2. Test the UI pages before deploying them to an instance.

   For more information, see Test UI pages in a development server.
3. Deploy the output in the `build` directory to the `/path/to/tomcat/webapps/openam/XUI/` directory in your AM instances.

   There is no need to restart the AM instance. Subsequent visits to the UI pages will use the rebuilt files.

Was this helpful?