UI Customization Guide

/ ForgeRock Access Management 7.1.4

Latest update: 7.1.4

ForgeRock AS.
201 Mission St., Suite 2900
San Francisco, CA 94105, USA
+1 415-599-1100 (US)
www.forgerock.com

Abstract

Guide showing you how to customize the user interface in ForgeRock® Access Management (AM). ForgeRock Access Management provides intelligent authentication, authorization, federation, and single sign-on functionality.

Table of Contents

Overview iv
1. About the UI 1
2. Downloading the UI 3
3. Localizing AM 5
4. Theming AM 6
5. Customizing the UI Layout 10
6. Testing and Deploying the UI 16
Glossary 19

Overview

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

This guide is written for anyone wanting to apply their own look and feel to the end-user facing pages provided by Access Management.

Quick Start

 About the User Interface Learn about the User Interface (UI), which is split into the User UI and the Admin UI.	 Downloading the UI Download, rebuild, and deploy the UI project to customize the layout and functionality of the UI.
 Theming the UI Learn how to change the look of the user-facing pages of the UI.	 Building, Testing and Deploying the UI Compile and execute UI unit tests and continuously watch for source changes to trigger re-testing.

About ForgeRock Identity Platform™ Software

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.

Chapter 1

About the UI

The UI is split in the following ways:

- User UI, which contains any end user pages and login pages.
- Admin UI, which contains any pages related to the administration of an AM server. Note, administrative logins are delegated to the user UI.

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

Note

The Admin UI cannot be customized, themed, or localized. However, changes in in the common components of the UI, such as the footer, do also appear in it.

ForgeRock provides the source code of the UI so you can customize it to fit your organization requirements. Customization of the layout of the pages or the JavaScript resources require a complete rebuild of the UI, but other customization, such as localizing or theming the UI, can be performed in the WAR file instead.

The recommended approach to customize the UI is to download the source code, modify it to suit your environment, and then either deploy it in your AM instances as part of your environment pipelines or pack it in your custom AM WAR files, then deploy them.

The UI contains the following types of resources:

UI Resources

Resources	Location
JavaScript source and configuration files	`openam-ui-user/src/js`
CSS collections	`openam-ui-user/src/resources/css`
Fonts	`openam-ui-user/src/resources/fonts`
Images	`openam-ui-user/src/resources/images`
Localization files	`openam-ui-user/src/resources/locales`
Themes	`openam-ui-user/src/resources/themes`
Partials	`openam-ui-user/src/resources/themes/default/partials`

Resources	Location
HTML pages	`openam-ui-user/src/resources/themes/default/templates`

Some of the resources, such as the CSS collections, are subdivided in directories depending on if they apply to the administrator pages, user pages, or to both.

Chapter 2
Downloading the UI

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

ForgeRock provides the source code for the UI as a Maven project located in the `am-external` repository, accessible to ForgeRock BackStage accounts that are added to a subscription.

If you do not already have a ForgeRock BackStage account, get one from https://backstage.forgerock.com. For more information about adding your account to a subscription, see Getting access to product support in the *ForgeRock Knowledge Base*.

To Download the UI Source

Perform the following steps to get the source of the UI:

1. Clone the `am-external` repository:

   ```
   $ git clone https://myBackStageUsername@stash.forgerock.org/scm/openam/am-external.git
   ```

   URL-encode your BackStage Username if it contains special characters. For example `:` becomes `%3A` and `@` becomes `%40`.

   Enter your BackStage password when prompted to do so.

2. Check out the `releases/7.1.4` branch:

   ```
   $ cd am-external
   $ git checkout releases/7.1.4
   ```

   The UI project is located in the `am-external/openam-ui/openam-ui-user/` folder.

3. If you do not already have them, install the following prerequisites:

   - Yarn.
   - Node.js (version 10.x.x).

Tip

For information on building the UI as part of a Maven workflow for deployment inside a WAR file, see How do I customize the XUI using source code in AM (All versions) and OpenAM 12.x, 13.x? in the *ForgeRock Knowledge Base*.

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

```
$ cd openam-ui/openam-ui-user
$ yarn
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 may receive an error such as `gyp ERR! not ok` when downloading dependencies.

You can safely ignore such errors as they only apply to optional components in certain environments, or you could try the **yarn install --ignore-optional** command.

Chapter 3
Localizing AM

Both user-facing and the administration pages are provided only in English, but you can customize and localize the user-facing text as required.

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

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 be specified as per rfc5646 - Tags for Identifying Languages. For example, `en-GB`.

2. Edit the files and change the values of the elements for the required locale taking care not to change the JSON structure or to render it invalid.

3. Rebuild the UI. For more information, see "To Rebuild and Deploy the UI".

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

Chapter 4
Theming 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.

To 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:

   a. 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: [...]
};
```

b. 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.

c. 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.

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

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

ii. (Optional) 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.

iii. (Optional) 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"]
    }]
};
```

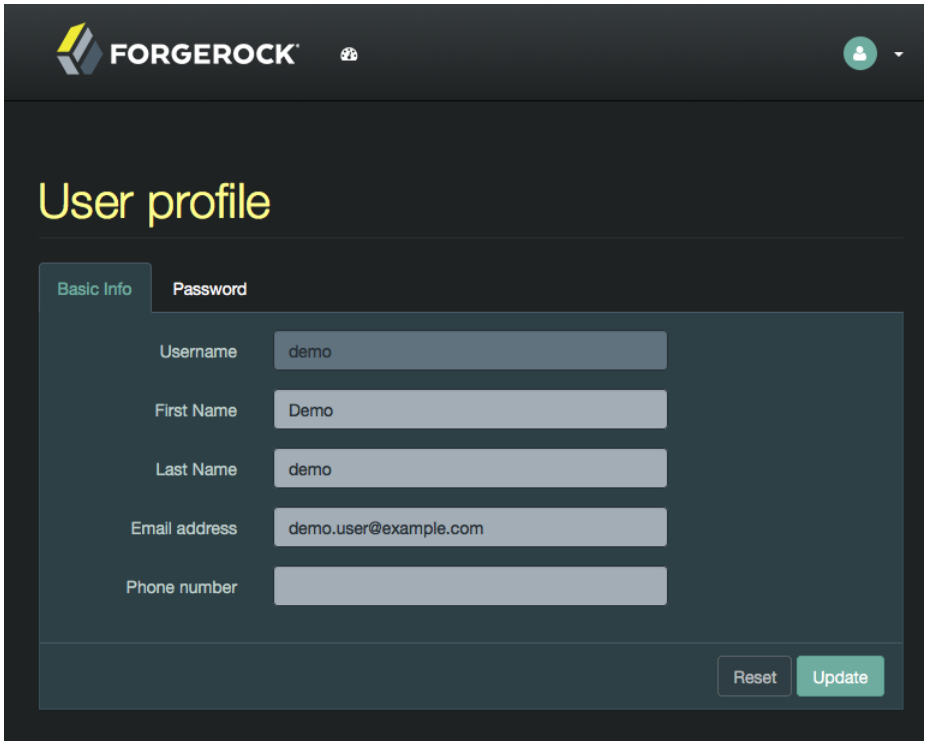
If you specify both realms and authentication services, the theme is only applied when both criteria are true.

3. (Optional) Compile the UI project using the **yarn build** command.

4. (Optional) 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:

UI with the Dark Theme



6. (Optional) 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.

Chapter 5

Customizing 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.

To Change UI Page Layouts

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 "*Downloading the UI*".
2. Modify the UI source as follows:
   a. 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.

   b. 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.

c. 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.

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

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

e. 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.

f. 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.

g. 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.

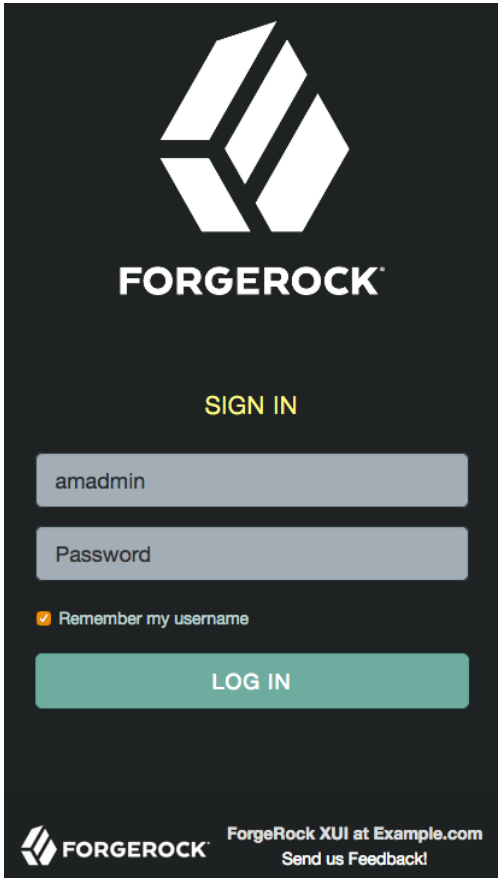
Tip

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

4. (Optional) Test the UI pages by following the steps detailed in "To Test UI Pages in a Development Server".

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

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.

Chapter 6
Testing and Deploying 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 "To 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`.

To Test UI Pages in a Development Server

You can run the UI project on a dedicated development server for testing 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, navigate to the full URL of your *AM instance*, but use the port number of the UI development server.

   For example, navigate 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.

To 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. (Optional) Test the UI pages before deploying them to an instance. For more information, see "To 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.

Glossary

Access control	Control to grant or to deny access to a resource.
Account lockout	The act of making an account temporarily or permanently inactive after successive authentication failures.
Actions	Defined as part of policies, these verbs indicate what authorized identities can do to resources.
Advice	In the context of a policy decision denying access, a hint to the policy enforcement point about remedial action to take that could result in a decision allowing access.
Agent administrator	User having privileges only to read and write agent profile configuration information, typically created to delegate agent profile creation to the user installing a web or Java agent.
Agent authenticator	Entity with read-only access to multiple agent profiles defined in the same realm; allows an agent to read web service profiles.
Application	In general terms, a service exposing protected resources. In the context of AM policies, the application is a template that constrains the policies that govern access to protected resources. An application can have zero or more policies.
Application type	Application types act as templates for creating policy applications. Application types define a preset list of actions and functional logic, such as policy lookup and resource comparator logic.

	Application types also define the internal normalization, indexing logic, and comparator logic for applications.
Attribute-based access control (ABAC)	Access control that is based on attributes of a user, such as how old a user is or whether the user is a paying customer.
Authentication	The act of confirming the identity of a principal.
Authentication chaining	A series of authentication modules configured together which a principal must negotiate as configured in order to authenticate successfully.
Authentication level	Positive integer associated with an authentication module, usually used to require success with more stringent authentication measures when requesting resources requiring special protection.
Authentication module	AM authentication unit that handles one way of obtaining and verifying credentials.
Authorization	The act of determining whether to grant or to deny a principal access to a resource.
Authorization Server	In OAuth 2.0, issues access tokens to the client after authenticating a resource owner and confirming that the owner authorizes the client to access the protected resource. AM can play this role in the OAuth 2.0 authorization framework.
Auto-federation	Arrangement to federate a principal's identity automatically based on a common attribute value shared across the principal's profiles at different providers.
Bulk federation	Batch job permanently federating user profiles between a service provider and an identity provider based on a list of matched user identifiers that exist on both providers.
Circle of trust	Group of providers, including at least one identity provider, who have agreed to trust each other to participate in a SAML v2.0 provider federation.
Client	In OAuth 2.0, requests protected web resources on behalf of the resource owner given the owner's authorization. AM can play this role in the OAuth 2.0 authorization framework.
Client-based OAuth 2.0 tokens	After a successful OAuth 2.0 grant flow, AM returns a token to the client. This differs from CTS-based OAuth 2.0 tokens, where AM returns a *reference* to token to the client.
Client-based sessions	AM sessions for which AM returns session state to the client after each request, and require it to be passed in with the subsequent

	request. For browser-based clients, AM sets a cookie in the browser that contains the session information. For browser-based clients, AM sets a cookie in the browser that contains the session state. When the browser transmits the cookie back to AM, AM decodes the session state from the cookie.
Conditions	Defined as part of policies, these determine the circumstances under which which a policy applies. Environmental conditions reflect circumstances like the client IP address, time of day, how the subject authenticated, or the authentication level achieved. Subject conditions reflect characteristics of the subject like whether the subject authenticated, the identity of the subject, or claims in the subject's JWT.
Configuration datastore	LDAP directory service holding AM configuration data.
Cross-domain single sign-on (CDSSO)	AM capability allowing single sign-on across different DNS domains.
CTS-based OAuth 2.0 tokens	After a successful OAuth 2.0 grant flow, AM returns a *reference* to the token to the client, rather than the token itself. This differs from client-based OAuth 2.0 tokens, where AM returns the entire token to the client.
CTS-based sessions	AM sessions that reside in the Core Token Service's token store. CTS-based sessions might also be cached in memory on one or more AM servers. AM tracks these sessions in order to handle events like logout and timeout, to permit session constraints, and to notify applications involved in SSO when a session ends.
Delegation	Granting users administrative privileges with AM.
Entitlement	Decision that defines which resource names can and cannot be accessed for a given identity in the context of a particular application, which actions are allowed and which are denied, and any related advice and attributes.
Extended metadata	Federation configuration information specific to AM.
Extensible Access Control Markup Language (XACML)	Standard, XML-based access control policy language, including a processing model for making authorization decisions based on policies.
Federation	Standardized means for aggregating identities, sharing authentication and authorization data information between trusted providers, and

	allowing principals to access services across different providers without authenticating repeatedly.
Fedlet	Service provider application capable of participating in a circle of trust and allowing federation without installing all of AM on the service provider side; AM lets you create Java Fedlets.
Hot swappable	Refers to configuration properties for which changes can take effect without restarting the container where AM runs.
Identity	Set of data that uniquely describes a person or a thing such as a device or an application.
Identity federation	Linking of a principal's identity across multiple providers.
Identity provider (IDP)	Entity that produces assertions about a principal (such as how and when a principal authenticated, or that the principal's profile has a specified attribute value).
Identity repository	Data store holding user profiles and group information; different identity repositories can be defined for different realms.
Java agent	Java web application installed in a web container that acts as a policy enforcement point, filtering requests to other applications in the container with policies based on application resource URLs.
Metadata	Federation configuration information for a provider.
Policy	Set of rules that define who is granted access to a protected resource when, how, and under what conditions.
Policy agent	Java, web, or custom agent that intercepts requests for resources, directs principals to AM for authentication, and enforces policy decisions from AM.
Policy Administration Point (PAP)	Entity that manages and stores policy definitions.
Policy Decision Point (PDP)	Entity that evaluates access rights and then issues authorization decisions.
Policy Enforcement Point (PEP)	Entity that intercepts a request for a resource and then enforces policy decisions from a PDP.
Policy Information Point (PIP)	Entity that provides extra information, such as user profile attributes that a PDP needs in order to make a decision.
Principal	Represents an entity that has been authenticated (such as a user, a device, or an application), and thus is distinguished from other entities.

	When a Subject successfully authenticates, AM associates the Subject with the Principal.
Privilege	In the context of delegated administration, a set of administrative tasks that can be performed by specified identities in a given realm.
Provider federation	Agreement among providers to participate in a circle of trust.
Realm	AM unit for organizing configuration and identity information. Realms can be used for example when different parts of an organization have different applications and identity stores, and when different organizations use the same AM deployment. Administrators can delegate realm administration. The administrator assigns administrative privileges to users, allowing them to perform administrative tasks within the realm.
Resource	Something a user can access over the network such as a web page. Defined as part of policies, these can include wildcards in order to match multiple actual resources.
Resource owner	In OAuth 2.0, entity who can authorize access to protected web resources, such as an end user.
Resource server	In OAuth 2.0, server hosting protected web resources, capable of handling access tokens to respond to requests for such resources.
Response attributes	Defined as part of policies, these allow AM to return additional information in the form of "attributes" with the response to a policy decision.
Role based access control (RBAC)	Access control that is based on whether a user has been granted a set of permissions (a role).
Security Assertion Markup Language (SAML)	Standard, XML-based language for exchanging authentication and authorization data between identity providers and service providers.
Service provider (SP)	Entity that consumes assertions about a principal (and provides a service that the principal is trying to access).
Authentication Session	The interval while the user or entity is authenticating to AM.
Session	The interval that starts after the user has authenticated and ends when the user logs out, or when their session is terminated. For browser-based clients, AM manages user sessions across one or more applications by setting a session cookie. See also CTS-based sessions and Client-based sessions.

Session high availability	Capability that lets any AM server in a clustered deployment access shared, persistent information about users' sessions from the CTS token store. The user does not need to log in again unless the entire deployment goes down.
Session token	Unique identifier issued by AM after successful authentication. For a CTS-based sessions, the session token is used to track a principal's session.
Single log out (SLO)	Capability allowing a principal to end a session once, thereby ending her session across multiple applications.
Single sign-on (SSO)	Capability allowing a principal to authenticate once and gain access to multiple applications without authenticating again.
Site	Group of AM servers configured the same way, accessed through a load balancer layer. The load balancer handles failover to provide service-level availability. The load balancer can also be used to protect AM services.
Standard metadata	Standard federation configuration information that you can share with other access management software.
Stateless Service	Stateless services do not store any data locally to the service. When the service requires data to perform any action, it requests it from a data store. For example, a stateless authentication service stores session state for logged-in users in a database. This way, any server in the deployment can recover the session from the database and service requests for any user. All AM services are stateless unless otherwise specified. See also Client-based sessions and CTS-based sessions.
Subject	Entity that requests access to a resource When an identity successfully authenticates, AM associates the identity with the Principal that distinguishes it from other identities. An identity can be associated with multiple principals.
Identity store	Data storage service holding principals' profiles; underlying storage can be an LDAP directory service or a custom `IdRepo` implementation.
Web Agent	Native library installed in a web server that acts as a policy enforcement point with policies based on web page URLs.