UI Customization Guide

ForgeRock Access Management 5

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 ForgeRock# Access Management user interface.

Table of Contents

Preface

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 ForgeRock Access Management.

About ForgeRock Identity Platform™ Software

ForgeRock Identity Platform™ is the only offering for access management, identity management, user-managed access, directory services, and an identity gateway, designed and built as a single, unified platform.

The platform includes the following components that extend what is available in open source projects to provide fully featured, enterprise-ready software:

- ForgeRock Access Management (AM)
- ForgeRock Identity Management (IDM)
- ForgeRock Directory Services (DS)
- ForgeRock Identity Gateway (IG)

Chapter 1

Introducing the User Interface

When you deploy OpenAM to protect your web-based applications, users can be redirected to OpenAM pages for login and logout.

The end user pages have ForgeRock styling and branding by default. You likely want to change at least the images to reflect your organization. You might want different customizations for different realms. This chapter addresses how to get started customizing OpenAM end user pages for your organizations and supported locales.

You may want to change the default styling and branding as well as customize different realms.

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

You can set advanced server properties in the AM console under Deployment > Servers > *Server Name* > Advanced. Before using OpenAM in production, set `org.forgerock.openam.core.resource.lookup.cache.enabled` back to the default setting, `true`.

Chapter 2

Customizing the User Interface

This chapter covers customizing the default user interface, known as the XUI.

2.1. Theming the XUI

This section explains how to use themes to alter the appearance of user-facing XUI pages.

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

Only user-facing XUI pages support themes. The OpenAM administration console cannot be themed.

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

Procedure 2.1. To Apply a Theme to the XUI

This procedure demonstrates adding a custom Bootstrap theme to the XUI.

1. Copy your custom Bootstrap theme to a directory in `/path/to/tomcat/webapps/openam/XUI/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: `/path/to/tomcat/webapps/openam/XUI/themes/dark/`.

2. Edit the `/XUI/config/ThemeConfiguration.js` file, to reference the CSS files in the theme, and to map the theme to realms and authentication chains:

   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`:

   ```
   define("config/ThemeConfiguration", {
       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.

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

Note that you must specify paths relative to the `XUI` 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 Section 3.2, "XUI Configuration Parameters".

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:

```
define("config/ThemeConfiguration", {
    themes: { ... },
    mappings: [
        {
            theme: "myTheme"
        }
    ]
});
```

ii. Optionally, create a `realms` array, and include the realms the theme will apply to:

```
define("config/ThemeConfiguration", {
    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. Optionally, create an `authenticationChains` array, and include the authentication chains the theme will apply to when used:

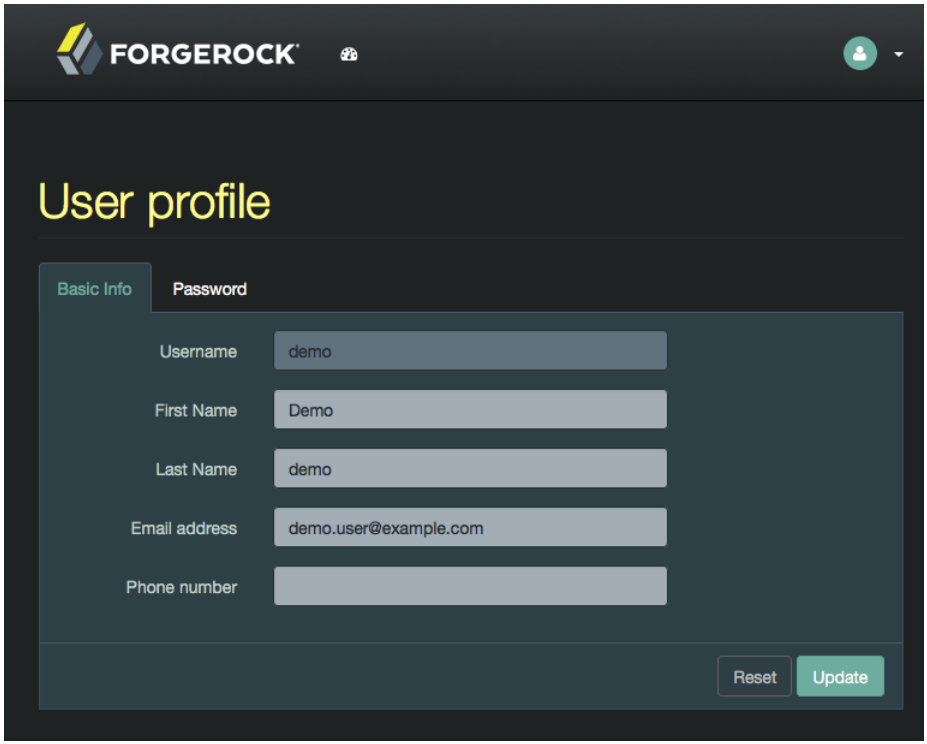
```
define("config/ThemeConfiguration", {
    themes: { ... },
    mappings: [
        {
            theme: "myTheme",
            realms: ["/", "/test-realm", /^\/a/],
            authenticationChains: ["auth-chain-one"]
        }
    ]
});
```

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

3. Save your work.

The next time a user logs in to the XUI they will see the new theme applied:

Figure 2.1. XUI with the Dark Theme



2.2. Customizing XUI Layout

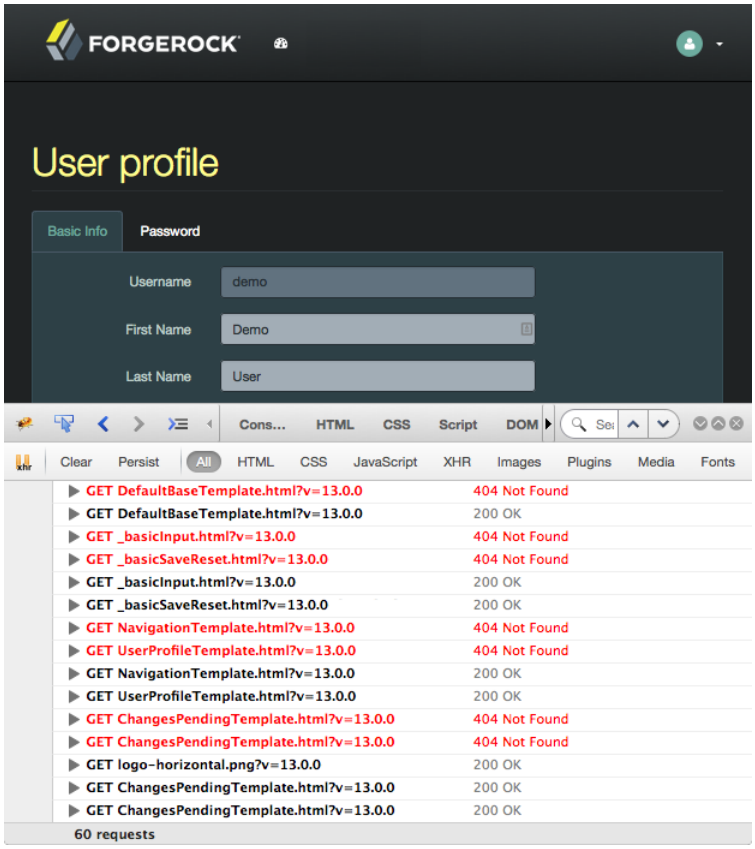
This section explains how to alter the layout of end user-facing XUI pages.

XUI pages are built with HTML templates, which in turn may contain reusable snippets of HTML stored in files referred to as *partials*.

The XUI stores the default templates in `/path/to/tomcat/webapps/openam/XUI/templates` and the default partials in `/path/to/tomcat/webapps/openam/XUI/partials`. You can override some, or all of these files by making duplicates containing edits and instructing the XUI to use the duplicates in place of the defaults.

If you provide a subset of the templates and partials provided with OpenAM, the XUI will fall back to the default set if a customized version is not provided. Note however that this will result in HTTP 404 Not Found errors in the background, which are visible in browser developer tools, but not visible to the end user:

Figure 2.2. Missing Customization Files Causing 404 Errors



To avoid HTTP 404 Not Found errors when customizing XUI layouts, duplicate the entire `/XUI/templates` and `/XUI/partials` directories into your custom theme directory, rather than only copying files that will be edited.

Procedure 2.2. To Customize XUI Layout

This procedure demonstrates customizing the default XUI layout by overriding a partial file.

Follow these steps on the server where OpenAM is deployed:

1. Copy the directories containing the templates and partials you want to customize to a directory in `/path/to/tomcat/webapps/openam/XUI/themes/`, ensuring that you maintain the same directory structure.

The following example copies the directory containing the default partials used for login pages into the `dark` theme directory, maintaining the `/partials/login/` directory structure:

```
$ cd /path/to/tomcat/webapps/openam/XUI
$ mkdir -p themes/dark/partials
$ cp -r partials/login/ themes/dark/partials/
```

2. Edit the copied template or partial files with the changes you require.

   For example, to include an HTML `<hr/>` tag to create a horizontal line that renders above password fields on login pages, edit the following file: `/path/to/tomcat/webapps/openam/XUI/themes/dark/partials/login/_Password.html`

```
<hr />
<label for="{{id}}" class="aria-label sr-only">{{prompt}}</label>
<input type="password"
    id="{{id}}"
    name="callback_{{index}}"
    class="form-control input-lg"
    placeholder="{{prompt}}"
    value="{{value}}"
    data-validator="required"
    required
    data-validator-event="keyup"
    {{#equals index 0}}autofocus{{/equals}}>
```

3. Edit the `/path/to/tomcat/webapps/openam/XUI/config/ThemeConfiguration.js` file, and add a `path` element that points to the newly edited templates or partials within the theme they will apply to.

   The following example alters the `fr-dark-theme` to use the custom login partials:

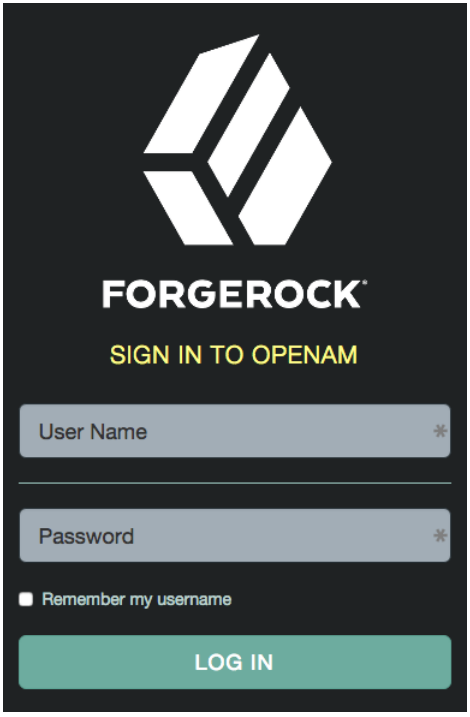
```
"fr-dark-theme": {
     path: "themes/dark/",
     stylesheets: [ ... ],
     settings: { ... }
 }
```

   Note that the trailing slash in the `path` value is required.

4. Save your work.

   The next time a user visits the login page in the XUI they will see the new partial applied, with the horizontal line above the password field:

Figure 2.3. XUI Login Page with Custom Partial



2.3. Localizing the XUI

This section explains how to localize the text that is generated for the user-facing XUI pages.

The text the XUI displays comes from from `translation.json` files located in locale-specific directories.

To customize the English text, edit `/path/to/tomcat/webapps/openam/XUI/locales/en/translation.json` under the directory where OpenAM is deployed.

To prepare a translation for a new locale, copy the provided `/path/to/tomcat/webapps/openam/XUI/locales/en` directory to `/path/to/tomcat/webapps/openam/XUI/locales/locale`, and edit the duplicate by changing the values, and taking care not to change the JSON structure or to render it invalid.

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

Chapter 3
Reference

This reference chapter covers the languages and locales supported by OpenAM, as well as configuration parameters for OpenAM's user interface, named the XUI.

3.1. Localization

This section lists languages and locales supported for OpenAM.

The XUI interface pages are localized for the following languages:

- English

You can localize the XUI for other languages as you require. For more information, see Section 2.3, "Localizing the XUI".

3.2. XUI Configuration Parameters

The configuration of the XUI is based on settings in the `ThemeConfiguration.js` file. This file can be found in the `/path/to/webapps/openam/XUI/config/` directory. 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 `"css/structure.css"` as one of the entries to provide default styles for layout and structure.

      For example: `["css/myTheme.css", "css/structure.css"]`
    - `path`: A relative path to a directory containing `templates` or `partials` directories, used for customizing the default layout of XUI pages.

      For more information, see Section 2.2, "Customizing XUI Layout".

- `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 XUI page.
    - `mailto`: Email address.
    - `phone`: Telephone number.

For more information, see Section 2.1, "Theming the XUI".

Appendix A. Getting Support

For more information or resources about OpenAM and ForgeRock Support, see the following sections:

A.1. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

- The ForgeRock Knowledge Base offers a large and increasing number of up-to-date, practical articles that help you deploy and manage ForgeRock software.
- ForgeRock core documentation, such as this document, aims to be technically accurate and complete with respect to the software documented. It is visible to everyone and covers all product features and examples of how to use them.

Core documentation therefore follows a three-phase review process designed to eliminate errors:

- Product managers and software architects review project documentation design with respect to the readers' software lifecycle needs.
- Subject matter experts review proposed documentation changes for technical accuracy and completeness with respect to the corresponding software.
- Quality experts validate implemented documentation changes for technical accuracy, completeness in scope, and usability for the readership.

The review process helps to ensure that documentation published for a ForgeRock release is technically accurate and complete.

Fully reviewed, published core documentation is available at http://backstage.forgerock.com/. Use this documentation when working with a ForgeRock Identity Platform release.

A.2. Joining the ForgeRock Community

Visit the Community resource center where you can find information about each project, download trial builds, browse the resource catalog, ask and answer questions on the forums, find community events near you, and find the source code for open source software.

A.3. Getting Support and Contacting ForgeRock

ForgeRock provides support services, professional services, classes through ForgeRock University, and partner services to assist you in setting up and maintaining your deployments. For a general overview of these services, see https://www.forgerock.com.

ForgeRock has staff members around the globe who support our international customers and partners. For details, visit https://www.forgerock.com, or send an email to ForgeRock at info@forgerock.com.

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 subjects 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 policy agent profile configuration information, typically created to delegate policy agent profile creation to the user installing a policy 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 OpenAM 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	OpenAM 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. OpenAM 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. OpenAM can play this role in the OAuth 2.0 authorization framework.
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 OpenAM configuration data.
Cross-domain single sign-on (CDSSO)	OpenAM capability allowing single sign-on across different DNS domains.
Delegation	Granting users administrative privileges with OpenAM.
Entitlement	Decision that defines which resource names can and cannot be accessed for a given subject 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 OpenAM.
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 OpenAM on the service provider side; OpenAM lets you create Java Fedlets.
Hot swappable	Refers to configuration properties for which changes can take effect without restarting the container where OpenAM 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 EE policy agent	Java web application installed in a web container that acts as a policy agent, 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	Agent that intercepts requests for resources, directs principals to OpenAM for authentication, and enforces policy decisions from OpenAM.
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, OpenAM associates the Subject with the Principal.
Privilege	In the context of delegated administration, a set of administrative tasks that can be performed by specified subjects in a given realm.
Provider federation	Agreement among providers to participate in a circle of trust.
Realm	OpenAM unit for organizing configuration and identity information. Realms can be used for example when different parts of an organization have different applications and user data stores, and when different organizations use the same OpenAM 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 OpenAM 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).
Session	The interval that starts with the user authenticating through OpenAM and ends when the user logs out, or when their session is terminated. For browser-based clients, OpenAM manages user sessions across one or more applications by setting a session cookie. See also Stateful session and Stateless session.
Session high availability	Capability that lets any OpenAM 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 OpenAM after successful authentication. For a Stateful session, 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 OpenAM servers configured the same way, accessed through a load balancer layer. The load balancer handles failover to provide service-level availability. Use sticky load balancing based on `amlbcookie` values to improve site performance. The load balancer can also be used to protect OpenAM services.
Standard metadata	Standard federation configuration information that you can share with other access management software.
Stateful session	An OpenAM session that resides in the Core Token Service's token store. Stateful sessions might also be cached in memory on one or

	more OpenAM servers. OpenAM tracks stateful 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.
Stateless session	An OpenAM session for which state information is encoded in OpenAM and stored on the client. The information from the session is not retained in the CTS token store. For browser-based clients, OpenAM sets a cookie in the browser that contains the session information.
Subject	Entity that requests access to a resource When a subject successfully authenticates, OpenAM associates the subject with the Principal that distinguishes it from other subjects. A subject can be associated with multiple principals.
User data store	Data storage service holding principals' profiles; underlying storage can be an LDAP directory service, a relational database, or a custom `IdRepo` implementation.
Web policy agent	Native library installed in a web server that acts as a policy agent with policies based on web page URLs.