User guide

This guide describes how to use ForgeRock Access Management Web Agent.

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.

About Web Agent

Web Agent is an Access Management add-on component that operates as a Policy Enforcement Point (PEP) or policy agent for applications deployed in a web server.

Web Agents intercept inbound requests to applications. Depending on the *filter mode* configuration, Web Agents interact with AM to:

- Ensure that clients provide appropriate authentication.
- Enforce AM resource-based policies.

  For information about how to enforce user authentication only, refer to SSO-only mode.

This chapter covers how Web Agent works and how it protects applications.

Agent components

Web Agent includes the following main components:

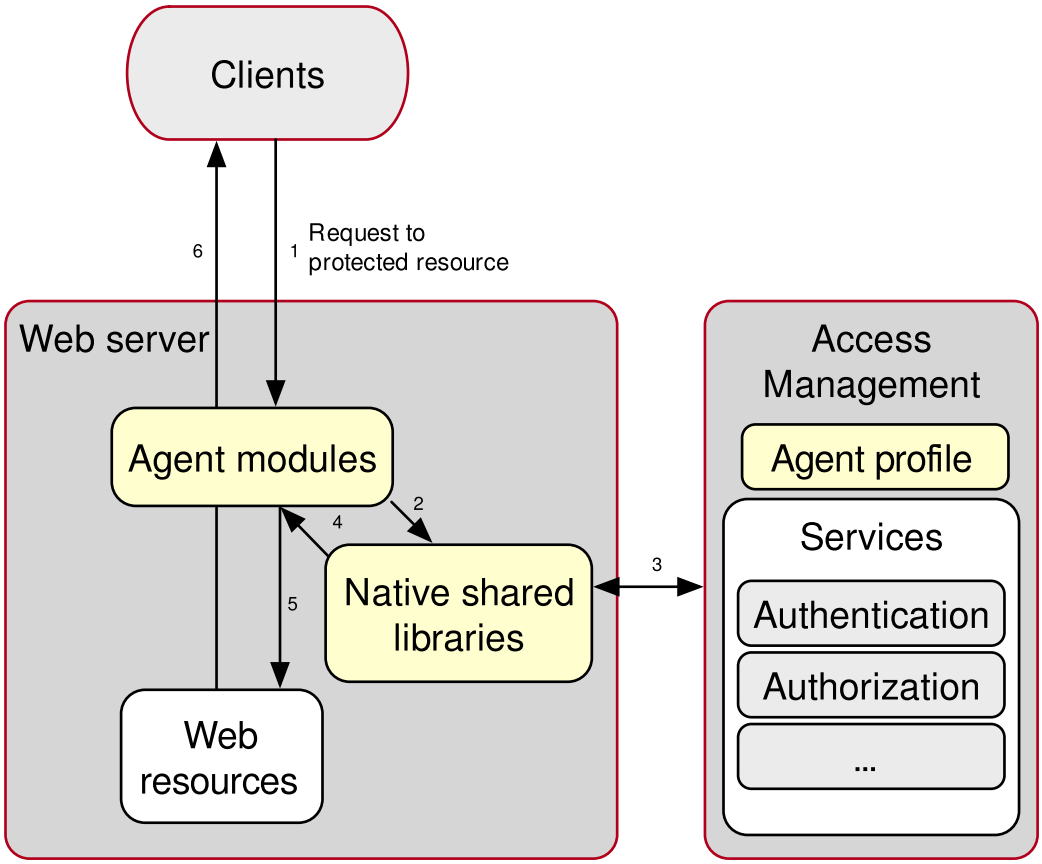
Agent Modules
: Intercepts and processes inbound requests to protected resources.

Native Shared Libraries
: Enables agents to interact with AM.

Agent Profile
: The agent profile is not strictly part of Web Agent, but plays an important part in the agent operation. It contains a set of configuration properties that define the agent

behavior.

The following image shows the Web Agent components when the agent profile is stored in the AM configuration store:



Configuration location

Web Agent configuration properties determine the behavior of the agent. AM stores configuration properties either centrally or locally:

Centralized configuration

AM stores the agent properties in the AM configuration store. Storing the agent configuration centrally allows you to configure your agents using the AM admin UI, the `ssoadm` command, and the REST API.

To access the centralized web agent configuration, on the AM admin UI go to **REALMS** > Realm Name > **Applications** > **Agents** > **Web** > Agent Name.

Configure properties that are not present in the AM admin UI as *custom properties*, on the **Advanced** tab of the console. For a list of property names, refer to the Properties reference.

When properties and value pairs are defined as custom properties, they take precedence for that property. Therefore, to prevent configuration errors, do not configure a property as a custom property if it has a UI counterpart.

For more information about creating centrally-stored agent profiles, refer to Creating agent profiles.

Local configuration (`agent.conf`)

The Web Agent installer creates the following file to store agent bootstrap and configuration properties:
`/web_agents/agent_type/instances/agent_nnn/config/agent.conf`

The installer populates `agent.conf` with enough information to make the agent start. To manage the configuration, edit `agent.conf` to add properties, remove properties, and change values. You cannot update `agent.conf` using the AM admin UI, the **`ssoadm`** command, or the REST API.

The `agent.conf` file must contain at least the following properties:

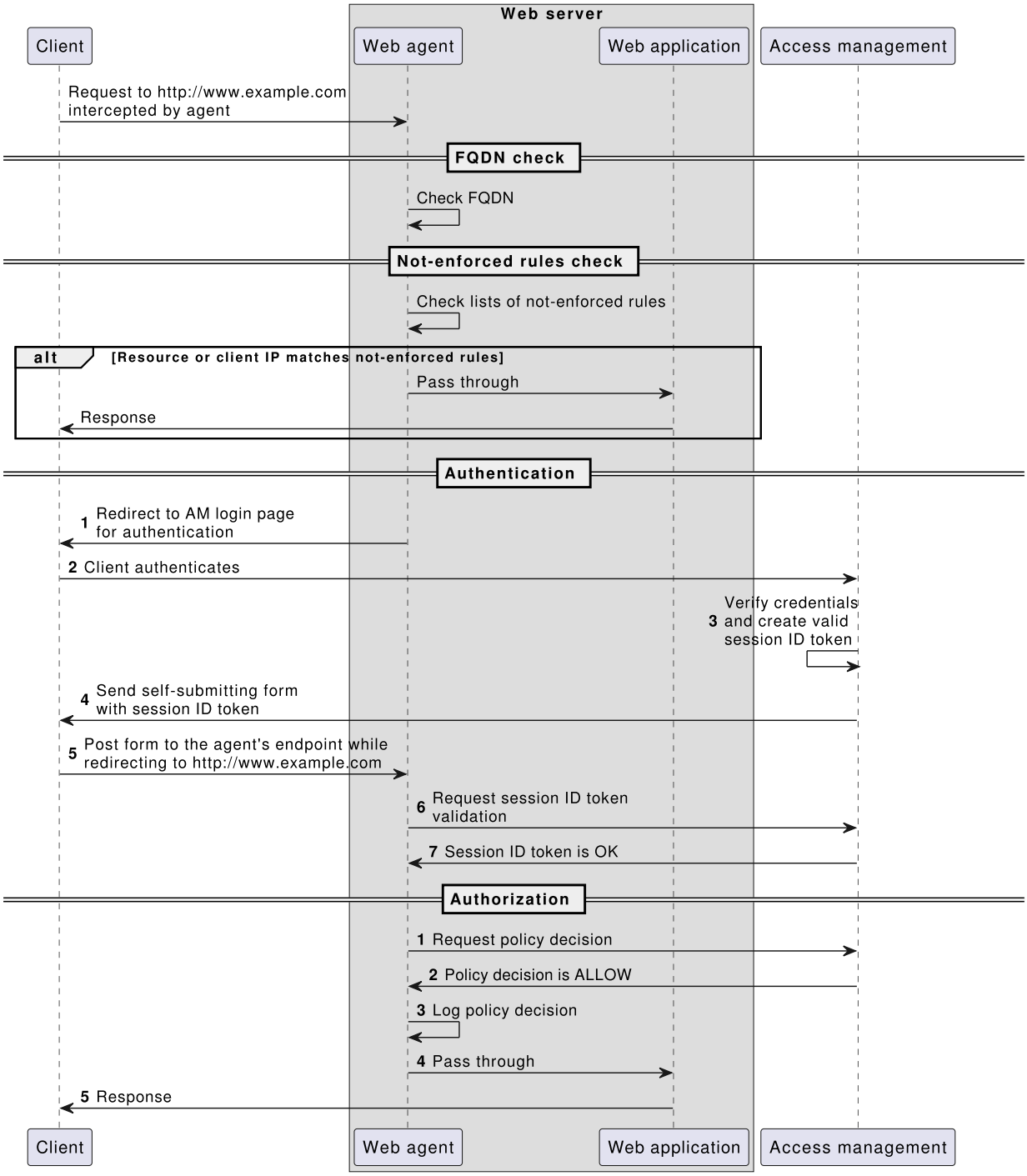
```
# Bootstrap properties
com.sun.identity.agents.config.organization.name = /
com.sun.identity.agents.config.username = ApacheAgentProfile
com.sun.identity.agents.config.password = o7OuvnaDnQ==
com.sun.identity.agents.config.key = OGM...Yg=
com.sun.identity.agents.config.naming.url =
https://am.example.com:8443/am

# Configuration properties
com.sun.identity.agents.config.repository.location = local
org.forgerock.openam.agents.config.jwt.name = am-auth-jwt
com.sun.identity.agents.config.cdsso.redirect.uri = agent/cdsso-
oauth2
org.forgerock.openam.agents.config.policy.evaluation.application
= iPlanetAMWebAgentService
org.forgerock.openam.agents.config.policy.evaluation.realm = /
com.sun.identity.agents.config.polling.interval = 60
com.sun.identity.agents.config.sso.cache.polling.interval = 3
com.sun.identity.agents.config.policy.cache.polling.interval = 3
com.sun.identity.agents.config.cookie.name = iPlanetDirectoryPro
com.sun.identity.agents.config.debug.file.size = 0
com.sun.identity.agents.config.local.logfile =
/web_agents/agent_type/instances/agent_name/logs/debug/debug.log
com.sun.identity.agents.config.local.audit.logfile =
/web_agents/agent_type/instances/agent_name/logs/audit/audit.log
com.sun.identity.agents.config.debug.level = Error
```

The properties are provided with an example value. For information about each of these properties, refer to the Properties reference.

Request process flow

When a client requests access to an application resource, the Web Agent intercepts the request. AM then validates the identity of the client, and their authorization to access the protected resource.

The following simplified data flow occurs when an unauthenticated client requests a resource protected by a Web Agent and AM. The flow assumes that requests must meet the requirements of an AM policy. For a detailed diagram, refer to Single sign-on in AM's *Authentication and SSO guide*.



FQDN check

When FQDN checking is enabled, the agent can redirect requests to different domains, depending on the hostname of the request. For more information, refer to FQDN checking.

Not-enforced rules check

The agent evaluates whether the requested resource or the client IP address matches a not-enforced rule.

- *Alternate flow*. The requested resource or the client IP address matches a not-enforced rule. The agent allows access to the resource.
- *Alternate flow*. The client receives a response from `www.example.com`. The flow ends.
- The requested resource or the client IP address does not match a not-enforced rule. The agent redirects the client to log in to AM.

For more information, refer to Not-enforced rules.

Authentication

1-2: The client authenticates to AM.

During client authentication, and to protect against reply attacks, the agent issues a pre-authentication cookie, named `agent-authn-tx`. The agent uses the cookie to track the authentication request to AM, and deletes it immediately after authentication.

Depending on the configuration, the agent can either issue a single cookie to track all concurrent authentication requests, or one cookie for each request.

The pre-authentication cookie expires after 5 minutes, or after the time specified in Profile Attributes Cookie Maxage.

If POST data preservation is enabled, the request expires after the time specified in POST Data Entries Cache Period, which is by default 10 minutes. In this case, consider increasing Profile Attributes Cookie Maxage to at least 10 minutes.

3: AM's authentication service verifies the client credentials and creates a valid OIDC JWT, with session information.

4: AM sends the client a self-submitting form with the OIDC JWT.

5: The client posts the self-submitting form to the agent endpoint, and the Web Agent consumes it.

6: The agent contacts AM to validate the session contained in the ID token.

7: AM validates the session.

Authorization

1: The agent contacts AM's policy service, requesting a decision about whether the client is authorized to access the resource.

2: AM's policy service returns `ALLOW`.

3: The agent writes the policy decision to the audit log.

4: The agent enforces the policy decision. Because the Policy Service returned `ALLOW`, the agent performs a pass-through operation to return the resource to the client.

5: The client accesses the resource.

Realms

Agent profile realm

The *agent profile realm* is the AM realm in which the agent profile is stored. The agent profile stores a set of configuration properties that define the behavior of the agent.

During agent installation, the installer prompts for the agent profile realm, and populates the property Agent Profile Realm in the bootstrap properties file. By default, the agent profile realm is set to the top-level realm.

The agent profile realm can be different to the user realm and policy evaluation realm. Groups of agents can use the same agent profile realm, which can be separate from the user realm and policy evaluation realm.

For information about creating agent profiles in the top-level realm or other realms, refer to Create agent profiles.

Policy evaluation realm

The *policy evaluation realm* is the realm that the agent uses to request policy decisions from AM. In most circumstances, the policy evaluation realm is the same as the user realm.

The policy evaluation realm is configured by Policy Evaluation Realm, and defaults to the top-level realm. The policy set to use is configured by Policy Set.

In AM, only the top-level realm has a default policy set, called iPlanetAMWebAgentService. If you use a policy evaluation realm that is in a subrealm of the top-level realm, you must also define a policy set and policies in the equivalent realm in AM.

User realm

The *user realm* is the realm in which a user is authenticated. In most circumstances, the user evaluation realm is the same as the policy evaluation realm.

By default, users authenticate to AM in the top-level realm, however, the agent can authenticate users in different realms depending on the request domain, path, or resource.

When a user logs out, the agent maintains the user realm. The agent obtains the realm info from the JWT, if one is available, or by calling `sessioninfo`. When the user logs out, the stored realm is passed to the logout endpoint automatically.

The first time an authenticated user requests a resource from the agent, the agent establishes the user realm from the session. It permanently associates the realm with the session in the session cache. When the session ends, the agent automatically passes the realm to the logout endpoint.

For more information about changing the user realm, refer to Login redirect.

Cross-domain single sign-on

Cross-domain single sign-on (CDSSO) is an AM capability that lets users access multiple independent services from a single login session, using the agent to transfer a validated session ID on a single DNS domain or across domains.

Without AM's CDSSO, SSO cannot be implemented across domains; the session cookie from one domain would not be accessible from another domain. For example, in a configuration where the AM server (`am.example.com`) is in a different DNS domain than the web agent (`myapp.website.com`), single sign-on would not be possible.

Web Agent works in CDSSO mode by default, regardless of the DNS domain of the AM servers and the DNS domain of the web agents.

For more information, refer to Single sign-on and Implementing CDSSO in AM's *Authentication and SSO guide*.

Policy enforcement

The agent evaluates policies as defined by the Policy evaluation mode (AM_POLICY_CACHE_MODE) environment variable. For information about caching policy decisions, refer to Caching.

This example sets up AM as a policy decision point for requests processed by Web Agent. Before you start, install a Web Agent as described in the Installation guide, with the following values:

- AM server URL: `http://am.example.com:8088/am`
- Agent URL: `http://agent.example.com:80`
- Agent profile name: `web-agent`
- Agent profile realm: `/`
- Agent profile password: `/secure-directory/pwd.txt`

Enforce a policy decision from AM

1. Using the ForgeRock Access Management docs for information, log in to AM as an administrator, and make sure you are managing the `/` realm.
2. Add a Web Agent profile:
    a. In the AM admin UI, select **Applications** > **Agents** > **Web**.
    b. Add an agent with the following values:
        - Agent ID: `web-agent`
        - Agent URL: `http://agent.example.com:80`
        - Server URL: `http://am.example.com:8088/am`
        - Password: `password`
3. Add a policy set and policy:
    a. In the AM admin UI, select **Authorization** > **Policy Sets**, and add a policy set with the following values:
        - **Id** : `PEP`
        - **Resource Types** : `URL`
    b. In the policy set, add a policy with the following values:
        - **Name** : `PEP-policy`
        - **Resource Type** : `URL`
        - **Resources** : `*://*:*/*`
    c. On the **Actions** tab, add actions to allow HTTP `GET` and `POST`.
    d. On the **Subjects** tab, remove any default subject conditions, add a subject condition for all `Authenticated Users`.
4. Assign the new policy set to the agent profile:
    a. In the AM admin UI, Select **Applications** > **Agents** > **Web**, and select your agent.
    b. On the agent page, select the **AM Services** tab.
    c. Set **Policy Set** to `PEP`, and then click **Save**.
5. Test the setup:

a. In the AM admin UI, select **Identities** > **+ Add Identity, and add a user with the following values:**

- **Username** : `demo`
- **First name** : `demo`
- **Last name** : `user`
- **Email Address** : `demo@example.com`
- **Password** : `Ch4ng31t`

b. Log out of AM, and clear any cookies.

c. Go to `http://agent.example.com:80` . The AM login page is displayed.

d. Log in to AM as user `demo` , password `Ch4ng31t` , to access the web page protected by the Web Agent.

Retrieve advice or response attributes from policy decisions

When AM makes a policy decision, it communicates an entitlement to the agent, which can optionally include advice and response attributes.

When AM denies a request with advice, the agent uses the advice to take remedial action. For example, when AM denies a request because the authentication level is too low, it can send advice to increase the authentication level. The agent then prompts the user to reauthenticate at a higher level, for example, by using a one-time password.

When AM allows a request it can include the following types of response attributes in the entitlement:

- Subject response attributes: Any LDAP user attribute configured for the identity store where AM looks up the user's profile. For more information, refer to Identity stores in AM's *Setup guide*.

  The agent adds the listed attributes to the response.

- Static response attributes: Any key:value pair, for example, `FrequentFlyerStatus` : `gold` .

  Depending on the value of Response Attribute Map, and Response Attribute Fetch Mode, the agent adds the listed attributes to HTTP headers or HTTP cookies in the response.

This example builds on the example in Enforce a policy decision from AM. Set up and test that example first.

1. Configure subject response attributes and static response attributes in the AM policy you created earlier:

a. In the AM admin UI, select the `PEP-policy`, and go to the **Response Attributes** tab.

b. In the **SUBJECT ATTRIBUTES** frame, select one or more of the available attributes. For example, select `cn`.

c. In the **STATIC ATTRIBUTES** frame, add a response attribute pair. For example, add the following pair:

- **PROPERTY NAME**: `FrequentFlyerStatus`
- **PROPERTY VALUE**: `gold`

d. Click **Save Changes**.

2. In the AM admin UI, select the `web-agent` you created earlier.

   The agent must use the AM policy set and realm where the response attributes are configured.

   If the response attributes are not present in the policy decision from AM, the agent does not create the corresponding HTTP header or cookie.

3. In the **Application** tab, set **Response Attribute Fetch Mode** to `HTTP-HEADER` or `HTTP-COOKIE` to select whether to map response attribute names to HTTP header names or HTTP cookie names.

   For more information, refer to Response Attribute Fetch Mode.

4. In the **Response Attribute Map** field, map the subject response attributes you selected in AM:

   - **Key**: cn
   - **Value**: `CUSTOM-name`

     The name of the AM response attribute `cn` is mapped to the HTTP header or cookie called `CUSTOM-name`. The value is taken from the user profile.

     For more information, refer to Response Attribute Fetch Mode.

5. In the **Response Attribute Map** field, map the static response attributes you added in AM:

   - **Key**: `FrequentFlyerStatus`
   - **Value**: `CUSTOM-flyer-status`

     The name of the AM response attribute `FrequentFlyerStatus` is mapped to the HTTP header or cookie called `CUSTOM-flyer-status`. The value is `gold`.

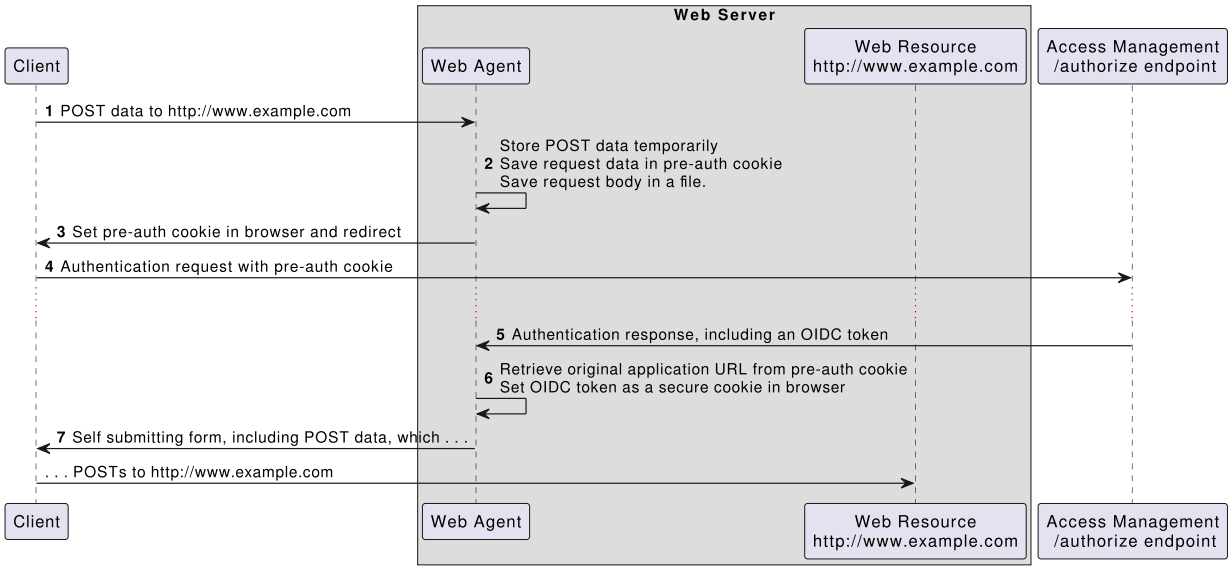
     For more information, refer to Response Attribute Map

POST data preservation

Use POST data preservation in environments where clients submit form data, and have short-lived sessions.

When POST data preservation is enabled, and an unauthenticated client posts data to a protected resource, the agent stores the POST data temporarily, and redirects the client to the login screen. The data can be any POST content, such as HTML form data or a file upload. After successful authentication, the agent recovers the stored POST data, and automatically submits it to the protected resource.

The following image shows a simplified data flow, when an unauthenticated client POSTs data to a protected web application:



Web Agent guarantees the integrity of the data, and the authenticity of the client as follows:

1. An unauthenticated client requests a POST to a protected resource.
2. The agent stores the POST data temporarily in the directory defined by POST Data Storage Directory, and saves data about the request in a standard pre-authentication cookie.
3. The agent sets a pre-authentication cookie in the browser, and redirects to the `/authorize` endpoint in AM.
4. The client authenticates with AM.
5. AM sends an authentication response to the registered redirect URI.
6. The agent retrieves the original application URL from the pre-authentication cookie, and replays the request with its body content to the server. The authentication response includes an OIDC token, which the agent sets as a secure cookie in the browser.

7. The agent sends a self-submitting form to the client browser, that includes the form data the user attempted to post in step 1. The self-submitting form POSTs to the protected resource.

For information about configuration properties, refer to POST data preservation.

Security considerations for POST data preservation

POST data is stored temporarily in the agent file system before a user is authenticated. Therefore, any unauthenticated user can POST a file that is then stored by the agent. Consider the following points when you configure POST data preservation:

- Payloads from unauthenticated users are stored in the agent files system. If your threat evaluation does not accept this risk, do not use POST data preservation; set Enable POST Data Preservation to `false`.
- By default, POST data is stored in the installation log directory, `web_agents/agent_type/instances/agent_n/log`. Although the log directory is considered secure, using it for POST data can cause the following risks:
    - Permissive access to POST data.

      Log directories can have relatively permissive read or copy permissions compared to other directories.
    - Leakage of personally identifiable information (PII).

      If POST data in log directories is processed by log applications, it could be included in a log view.

    To store POST data in a dedicated directory, set POST Data Storage Directory.
- POST data is stored for the time defined by POST Data Entries Cache Period and then deleted. To identify threats in POST data before it is deleted, make sure Intrusion Detection Systems inspect the data within the specified time.

Defend against CSRF attacks when using POST data preservation

WARNING

WARNING

Cross-site request forgery attacks (CSRF or XSRF) can be a cause of serious vulnerabilities in web applications. It is the responsibility of the protected application to implement countermeasures against such attacks, because Web Agent cannot provide generic protection against CSRF. ForgeRock recommends following the latest guidance from the OWASP CSRF Prevention Cheat Sheet.

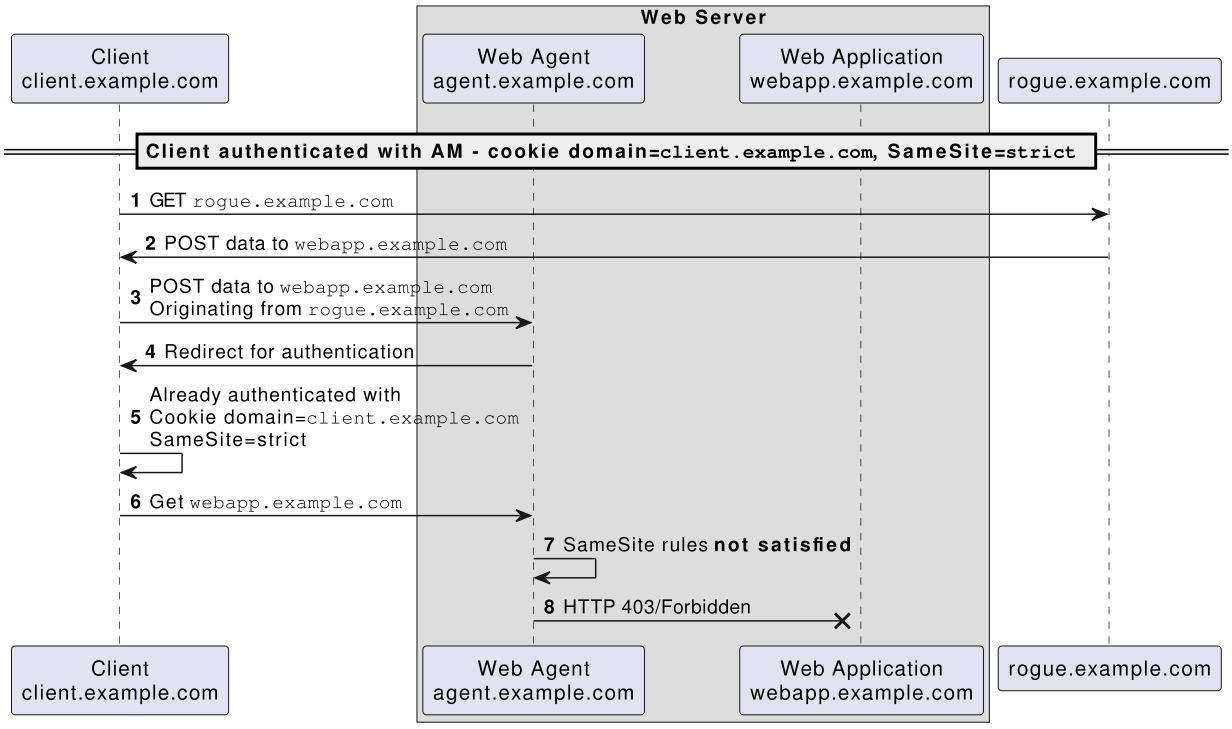
When POST data preservation is enabled, captured POST data that is replayed appears to come from the same origin as the protected application, not from the site that originated the request. Therefore, CSRF defenses that rely solely on checking the origin of requests, such as SameSite cookies or Origin headers, are not reliable.

To defend against CSRF attacks when POST data preservation is enabled, the agent uses a secure cookie and a nonce. The nonce must correspond to the authentication response from AM. This defense during authentication is specified in Cross-Site Request Forgery.

ForgeRock strongly recommend using token-based mitigations against CSRF, and relying on other measures only as a defense in depth, in accordance with OWASP guidance.

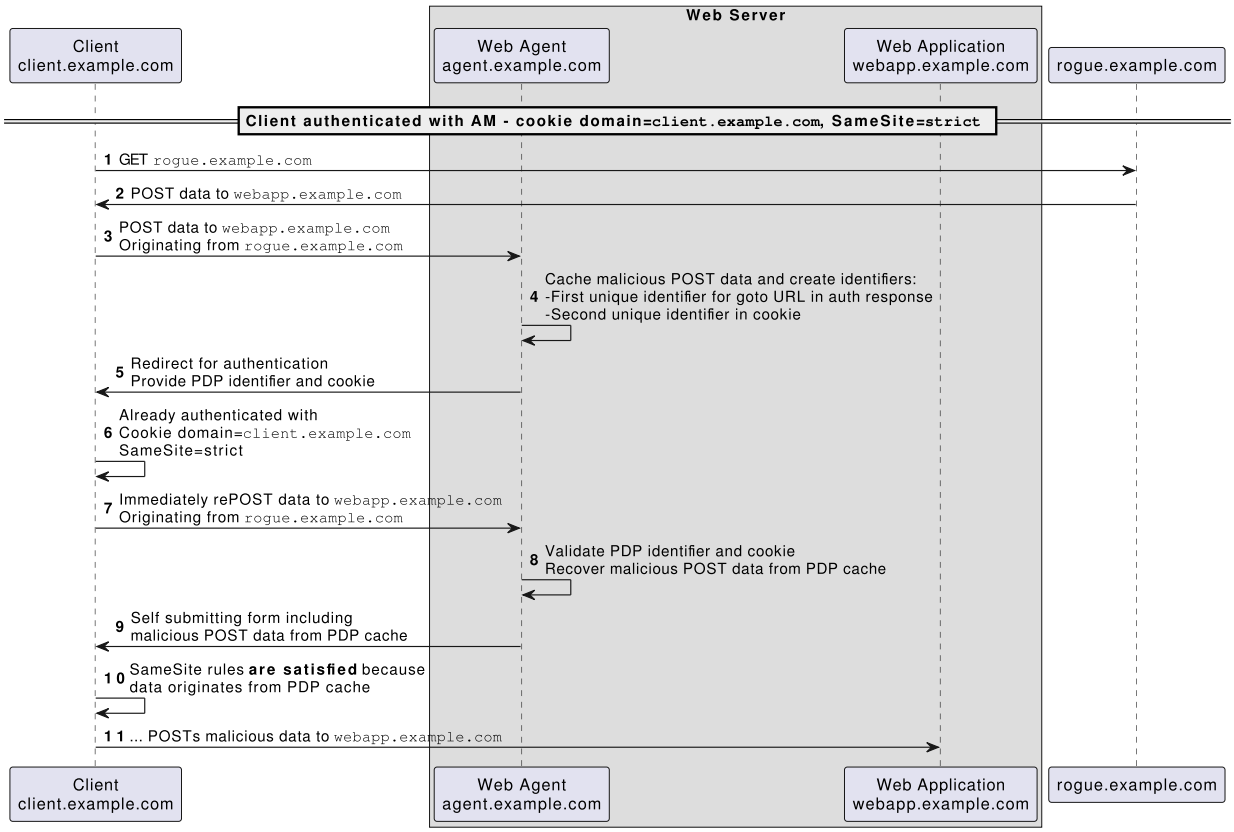
CSRF attack when POST data preservation is disabled

The following image shows a simplified data flow during a CSRF attack on an authenticated client when POST data preservation is disabled. In this limited scenario, the agent SameSite setting is enough to defend the web application:



CSRF attack when POST data preservation is enabled

The following image shows a simplified data flow during a CSRF attack on an authenticated client when POST data preservation is enabled. In this scenario, the SameSite setting **is not** enough to defend the web application:



Login redirect

When an unauthenticated user requests access to a protected resource, the agent redirects the user to log in. The choice of the login endpoint, and the parameters it receives, is defined by the login redirect mode and whether the agent accepts SSO tokens and ID tokens as session cookies.

Configure login redirect options as follows:

	Accept SSO Token	
	0	1

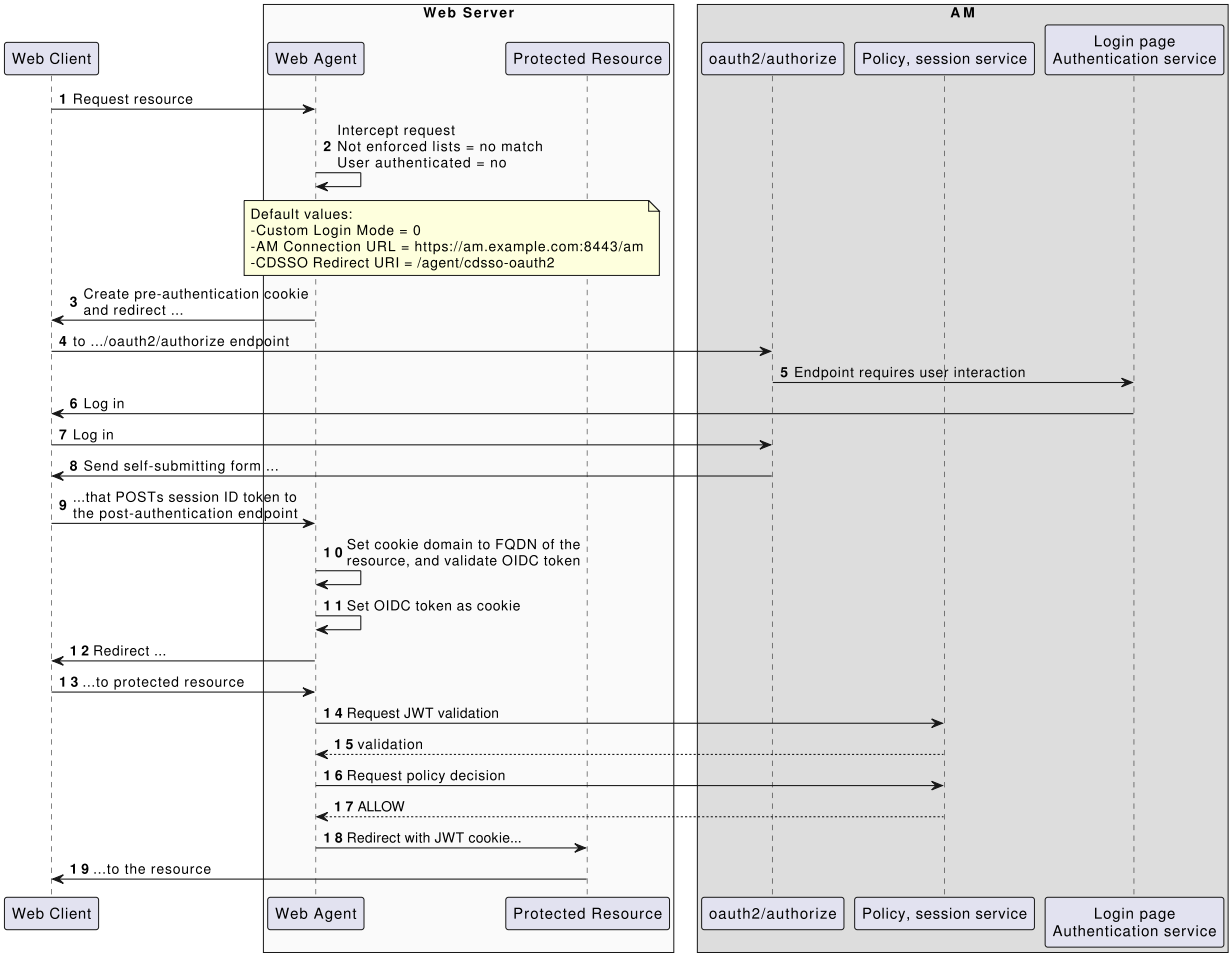
<table>
<tr><td rowspan="3">Enable Custom Login Mode</td><td>0</td><td colspan="2">Default login redirect:
Do not configure AM Login URL.
(Optional) Configure AM Conditional Login URL to point to the AM admin UI or the AM <code>oauth2/authorize</code> URL.</td></tr>
<tr><td></td><td>Does not accept SSO tokens as session cookies.</td><td>Accepts SSO tokens and ID tokens as session tokens during and after the login flow.
Does not convert SSO tokens to ID tokens.</td></tr>
<tr><td>1</td><td>Same domain custom login redirect or Cross domain custom login redirect:
Redirects login to the originally requested resource.
Converts SSO tokens to ID tokens.
Configure AM Login URL to point to a custom login page.
(Optional) Configure AM Conditional Login URL to point to the same URL as AM Login URL with additional parameters such as the login realm. Requests are logged conditionally to this URL.
Do not configure AM Conditional Login URL or AM Login URL to point to the AM admin UI or the AM <code>oauth2/authorize</code> URL.</td><td>Not supported.</td></tr>
</table>

	2	• Not supported.	Same domain custom login redirect: • This non-standard flow has limitations. Use only for environments migrating from earlier versions of the agent. • Redirects login with a `goto` query parameter to the originally requested resource • Accepts SSO tokens • Configure AM Login URL to point to a custom login page • (Optional) Configure AM Conditional Login URL to point to the same URL as AM Login URL with additional parameters such as the login realm. Requests are logged conditionally to this URL. • Do not configure AM Conditional Login URL or AM Login URL to point to the AM admin UI or the AM `oauth2/authorize` URL.

Default login redirect

In default login redirect, the agent redirects users for authentication to a page on the AM admin UI. The agent uses OpenID Connect ID JTWs as session tokens. Default login always redirects users to the top-level realm, irrespective of the user realm.

The following image shows the flow of data during a default login redirect. When an unauthenticated user requests access to a protected resource. The agent wraps the SSO session token inside an OpenID Connect JWT. Authentication requires access to the `/oauth2/authorize` endpoint, which invokes the AM admin UI and other endpoints such as `oauth2/authorize`, `json/authenticate`, `json/sessions`, `json/serverinfo`, and `XUI/*`.



Custom login redirect

In custom login redirect, the agent can redirect login in the following ways:

- Redirect login to custom login pages, in the same or a different realm
- Conditionally redirect login to different AM instances, AM sites, or authentication realms, based on the request URL.
- Use AM-specific SSO tokens as session tokens

Same domain custom login redirect

The agent redirects unauthenticated users to a custom login page in the same domain, adding the `original_request_url` parameter to the redirect. The parameter records the requested URL, which can then be used by custom login application or page.

The custom login page posts an SSO token to `agent/custom-login-response`, with the realm as an optional parameter, and sets an SSO token in the same domain as the agent.

The agent attempts to validate the SSO token against the AM endpoints.

At the end of the login flow, the agent can do the final redirect to the protected resource, or to the originally requested resource, with a `goto` query parameter

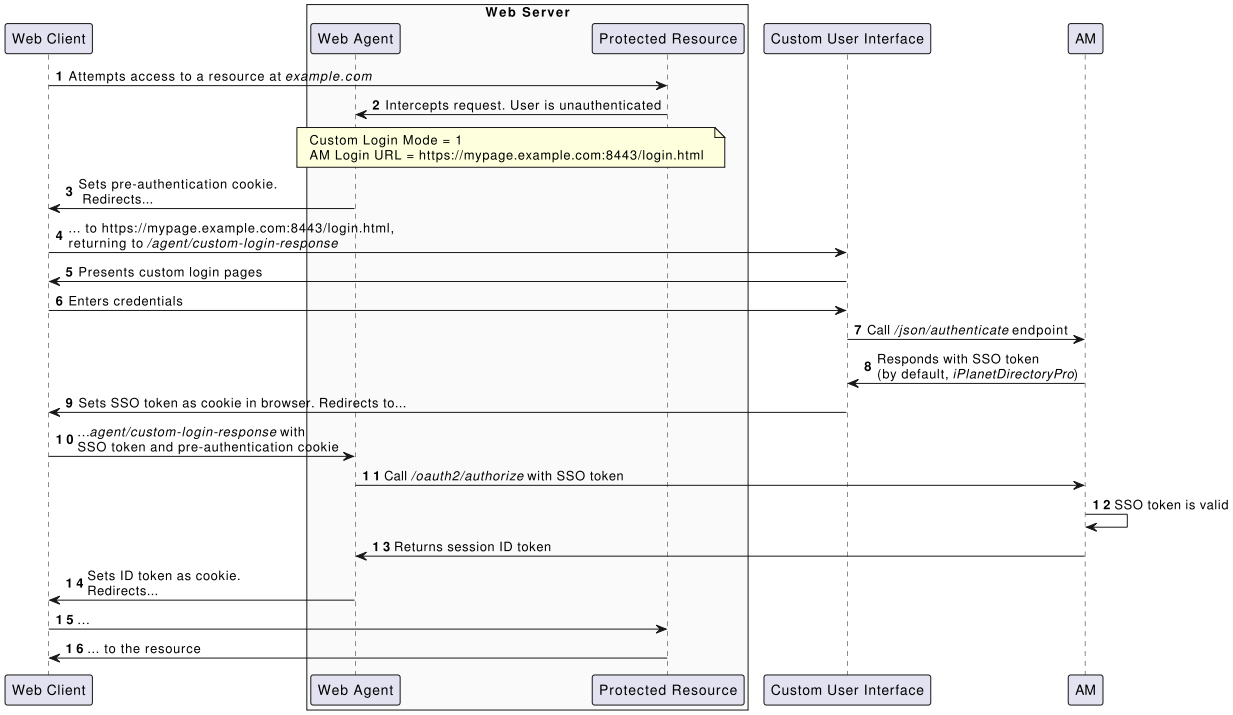
Same domain custom login redirect with final redirect to the protected resource

Set the following properties:

- Enable Custom Login Mode as `1`, to use the OIDC-compliant custom login redirection mode
- AM Login URL, to the URL of the custom login page

If the custom login page is a part of the agent's protected application, add the custom login pages to the not-enforced lists.

The following image shows the data flow for a custom login redirect when the custom login pages are in the same domain as the agent, and the agent redirects the the request to the originally requested resource.



Same domain custom login redirect with final redirect to the originally requested resource (migration mode)

CAUTION

Use this mode to migrate from Web Agent 4 to Web Agent 5.

Set the following properties:

- Enable Custom Login Mode as `2`
- Accept SSO Token

The agent redirects the client with a `goto` query parameter to the originally requested resource. This mode only operates on HTTP GET requests; POST requests are not supported.

The custom login pages obtain the SSO token from AM, but the agent does not create the pre-authentication cookie, which is used (among other things) to protect against CSRF attacks.

Because part of this flow happens outside the agent's control, the SSO token can expire or become invalid before the agent validates it. If so, the user/client must re-authenticate.

Cross-domain custom login redirect

The agent redirects unauthenticated users to a custom login page in a different domain, including the `original-request-uri` parameter in the redirect. The parameter records the requested URL, which can then be used by custom login application or page.

The custom login page provides a `custom-login-response` , and sets an SSO token, which can be accessed only in that domain. Because the agent cannot access the cookie, it redirects to AM for the Default login redirection mode.

Depending on your environment, the agent can contact AM to validate the cookie even if it cannot detect it. In other cases, you need to configure an additional property.

If AM can validate the SSO token, it returns an ID token as part of the default redirection login flow.

Consider the following points:

- Ensure that the login pages **do not** set the SSO token cookie with the `SameSite=Strict` attribute.
- If AM cannot validate the SSO token (for example, because it cannot recognize the domain set for the cookie), it redirects the end user to authenticate again using the Default login redirection mode.
- AM must be visible to the custom login pages, either because they both are in the same network/domain, or because you exposed the relevant AM endpoints using a proxy:

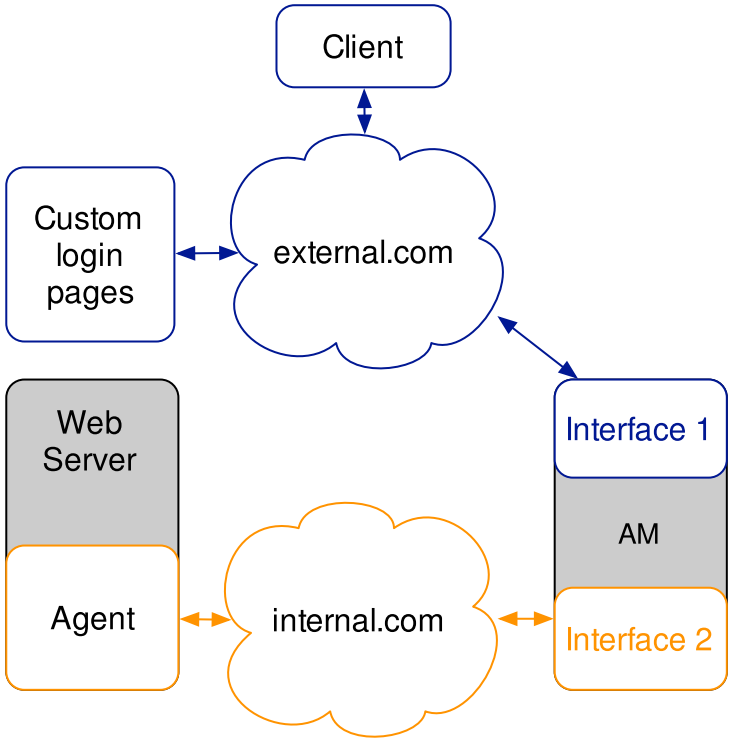
Cross-domain custom login redirect on a shared network

On a shared network, the server where AM is running has two interfaces: one connected to the internal network, where the agent is, and another connected to the external network, where the custom login pages are.

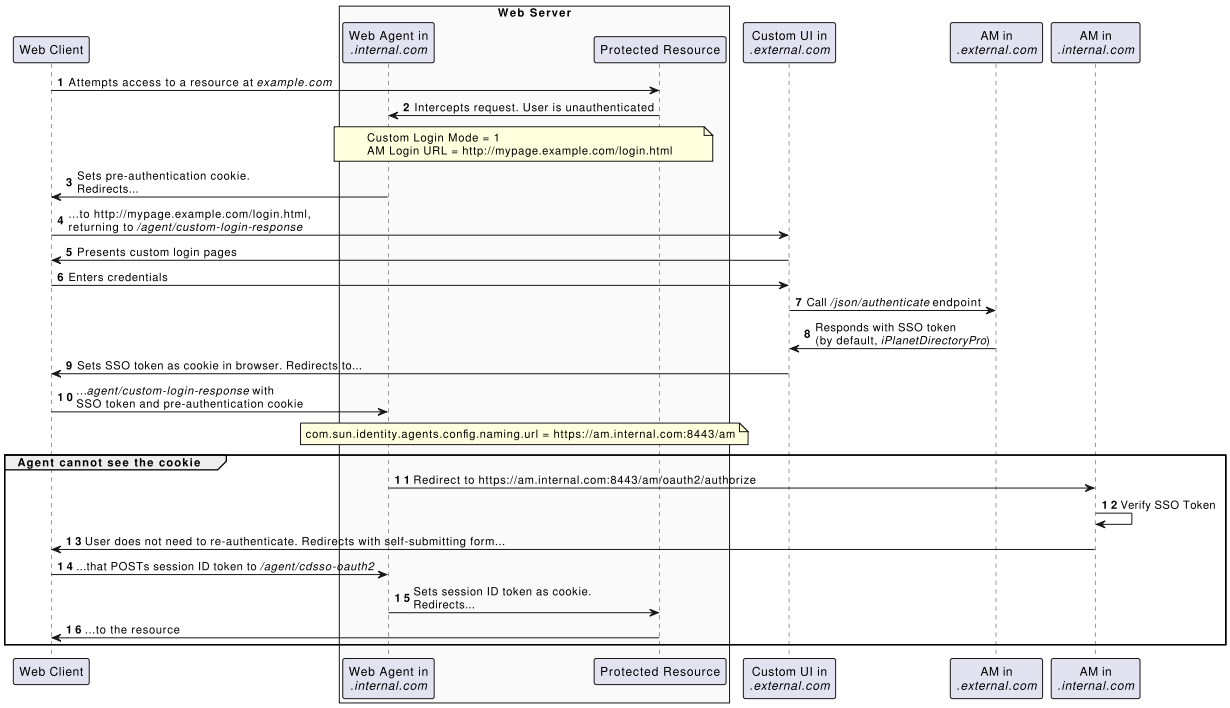
Use the following properties to configure this scenario:

- Enable Custom Login Mode
- AM Login URL

The following image illustrates the environment. The web server housing the protected resources can be connected to the external network in different ways; with two interfaces, or through a proxy. It is not important for the purposes of custom login, so it is not shown.



The following image shows the data flow:



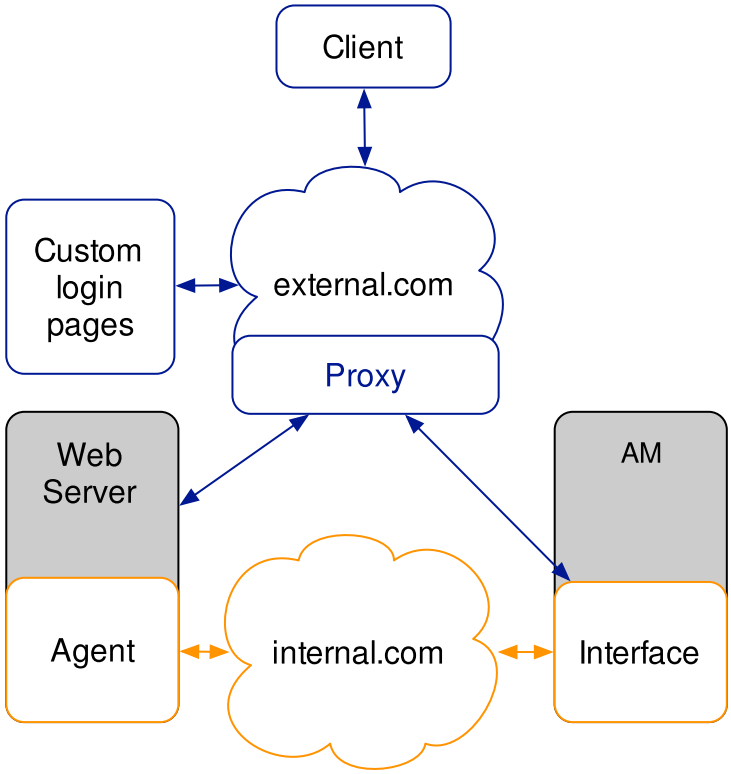
Cross-domain custom login redirect with AM behind a proxy

The server where AM is running has one interface to the internal network, where the agent is. A proxy hides AM from the external network, which forwards traffic to the `/oauth2/authorize` endpoint.

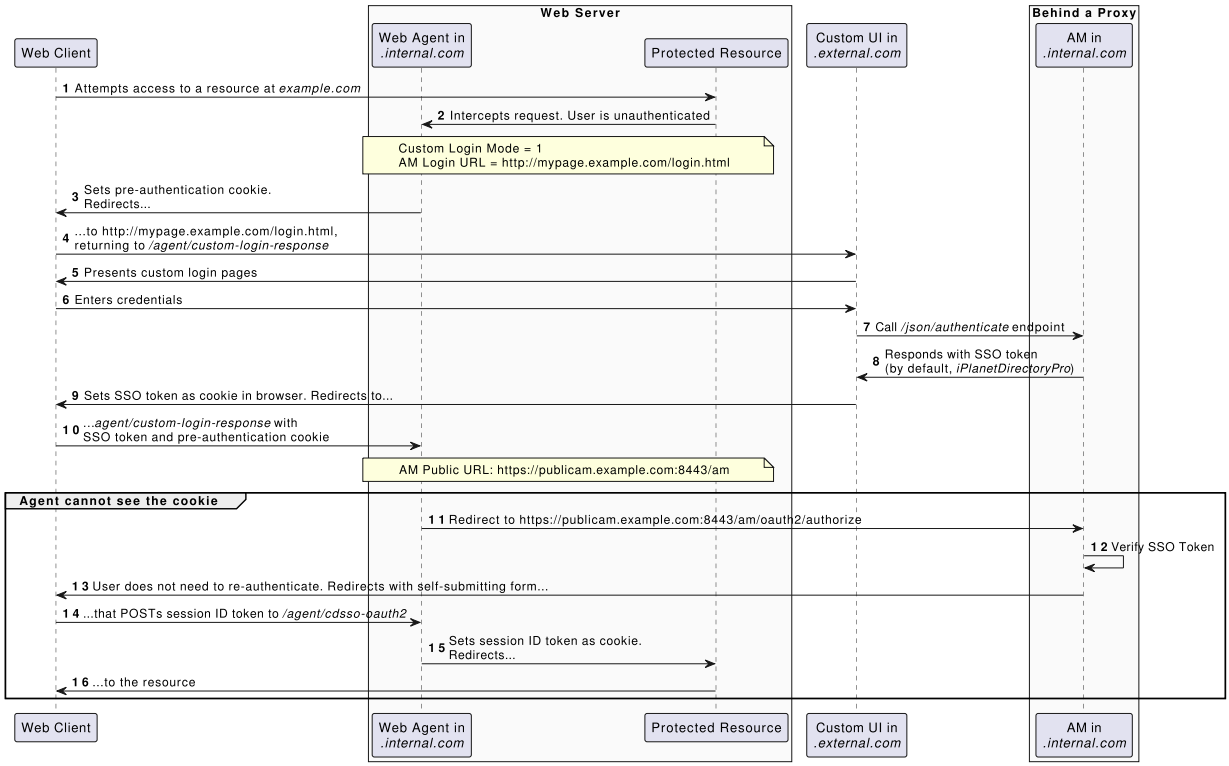
Use the following properties to configure this scenario:

- Enable Custom Login Mode
- AM Login URL
- Public AM URL

The following image illustrates the environment. The web server where the protected resources are may be connected to the external network in different ways; with two interfaces, or through a proxy. It is not important for the purposes of custom login, so it is not shown in the following diagram:



The following image shows the data flow:



Conditional login redirect

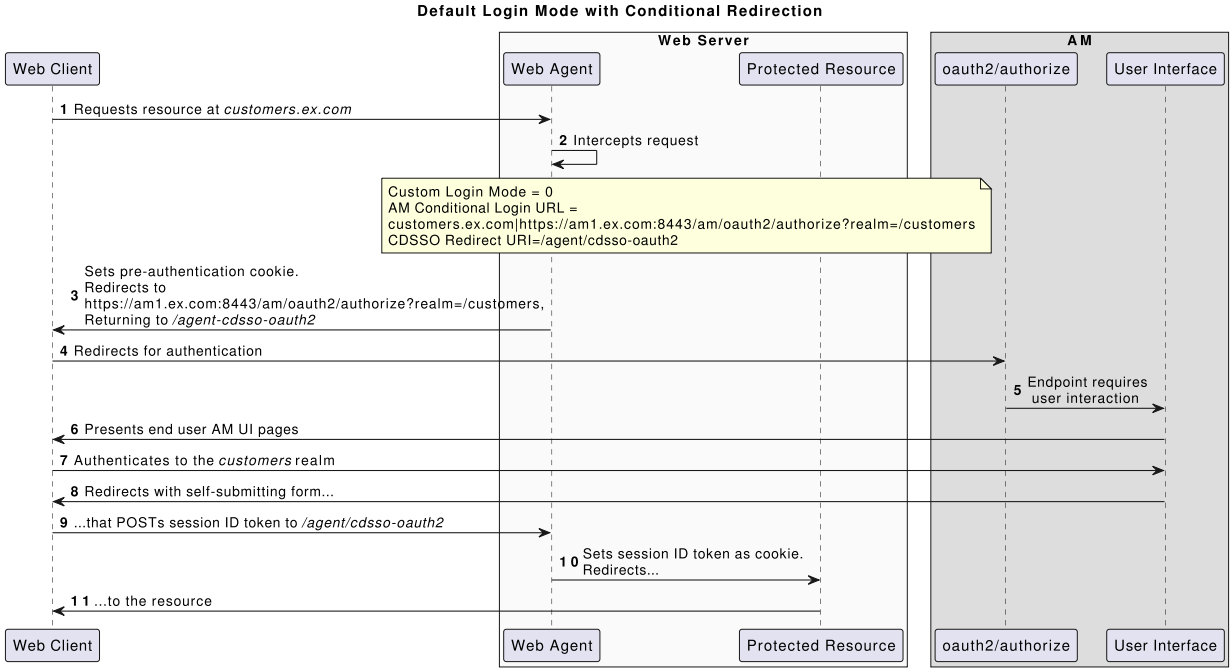
Use conditional redirects to redirect the end user to different AM instances or sites, or to different custom pages, depending on the incoming request URL.

Conditionally redirect login by domain name

When the incoming request URL matches a specified domain name, configure the following properties to redirect to a specified URL:

- Enable Custom Login Mode = 0
- AM Conditional Login URL
- CDSSO Redirect URI

The following image shows the data flow when a requests to the domain `customers.example.com` are redirected to the `customers` realm. Other requests are redirected to the top-level realm.



Conditionally redirect login by matching regular expressions

When the incoming request URL matches a regular expression, configure the following properties to redirect to a specified URL:

- Enable Custom Login Mode = 0
- Regular Expression Conditional Login Pattern
- Regular Expression Conditional Login URL

In the following example, when the request matches the regular expression `.*shop`, the agent redirects it to the `sales` realm for authentication:

```
org.forgerock.openam.agents.config.allow.custom.login = 0
org.forgerock.agents.config.conditional.login.pattern[0] = .*shop
org.forgerock.agents.config.conditional.login.url[0] =
http://am.example.com/am/oauth2/authorize?realm=sales
```

Redirect login to a custom URL configured in AM

AM's **OAuth2 Provider** service can be configured to use a custom URL to handle login, to override the default AM login page. When a custom login page is configured in AM, configure the agent to ensure that it redirects the login to that page.

1. In the AM admin UI, go to **Services** > **OAuth2 Provider** > **Advanced** > **Custom Login URL Template**, and note the custom URL.
2. Go to **Applications** > **Agents** > **Web**, and select your Web Agent.

3. On the **AM Services** tab set the following properties:
    - Enable Custom Login Mode: Set to `1`.
    - AM Conditional Login URL: Set to the custom URL in step 1.

Logout redirect

On logout, the Web Agent can redirect users to a specific AM page, or to a custom logout page in your web server. For properties to configure logout redirect, refer to Logout redirect.

The agent maintains the user realm for each session, obtaining it from the JWT or `sessioninfo` endpoint. When a user logs out, the agent automatically passes the stored realm to the logout endpoint. To log out to a landing page in a specific realm, specify the realm in AM Logout URL.

Logout redirect is triggered when Disable Logout Redirection is `false`, and the incoming URL matches a value in Logout URL List or Agent Logout URL Regular Expression.

When the incoming URL matches a logout URL, the agent redirects the web client to a URL configured in Logout Redirect URL.

IMPORTANT

To ensure the end user can access the logout redirect URL, add it to the Not-Enforced URL List.

If Enable Invalidate Logout Session is `true`, the agent invalidates the session in AM. Configure this if Logout URL List is set to a page in your application, and your application *does not handle* the session invalidation process.

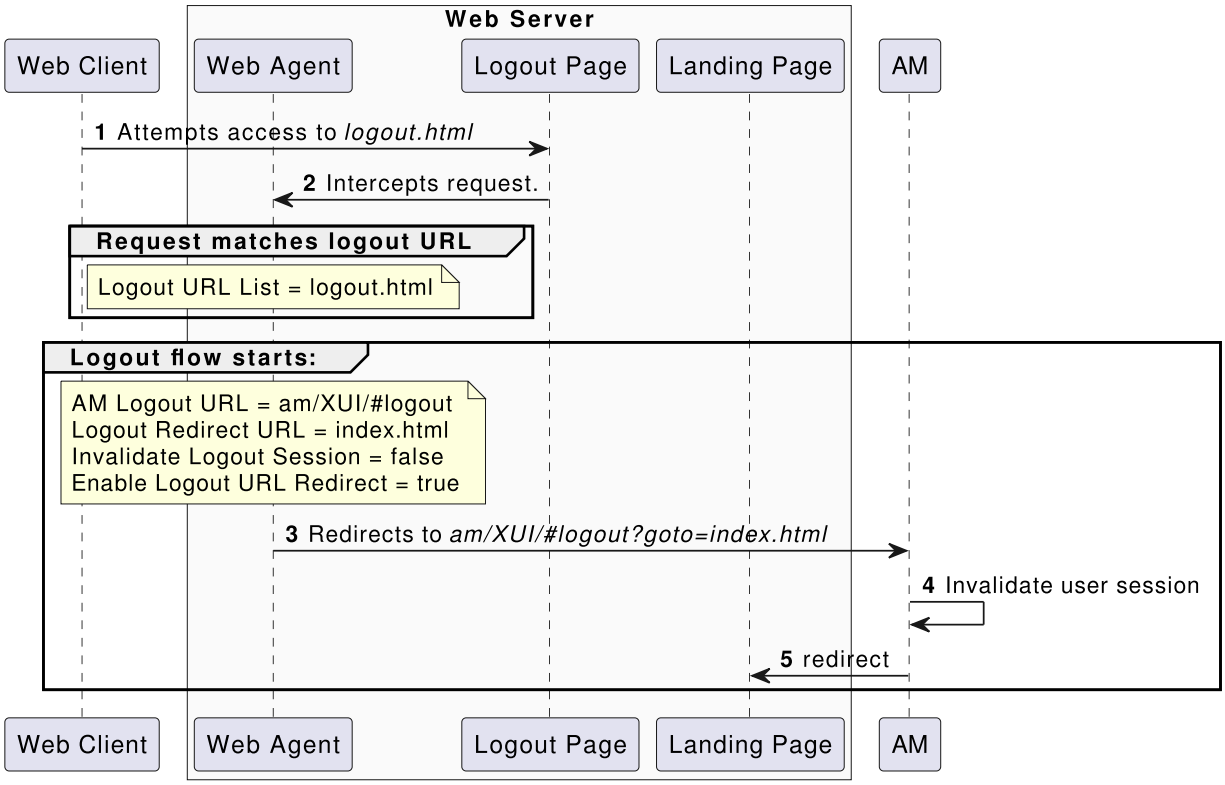
If Enable Invalidate Logout Session is `false`, the logout page is responsible for invalidating the user session. Configure this if the Logout URL List is a SAML v2.0 logout page, the AM logout page, or a page in your application that can handle the session invalidation process.

If Disable Logout Redirection is `true`, the agent does not add the `goto` parameter, and the web client remains in the logout page.

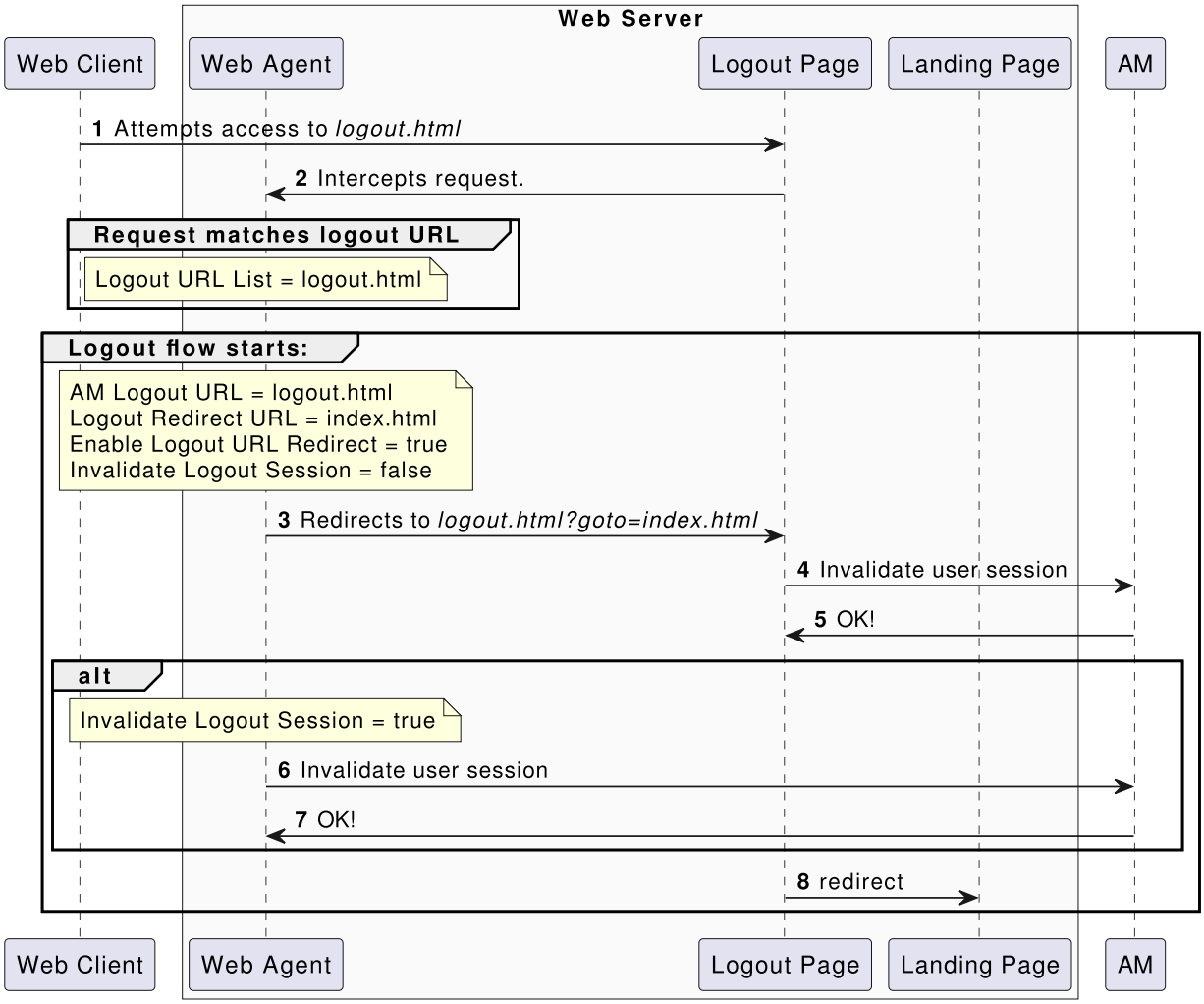
TIP

To reset specified cookies during logout, configure Reset Cookies on Logout List.

Example logout flow with AM as the logout page



Example logout flow with the application serving the logout page



Not-enforced rules

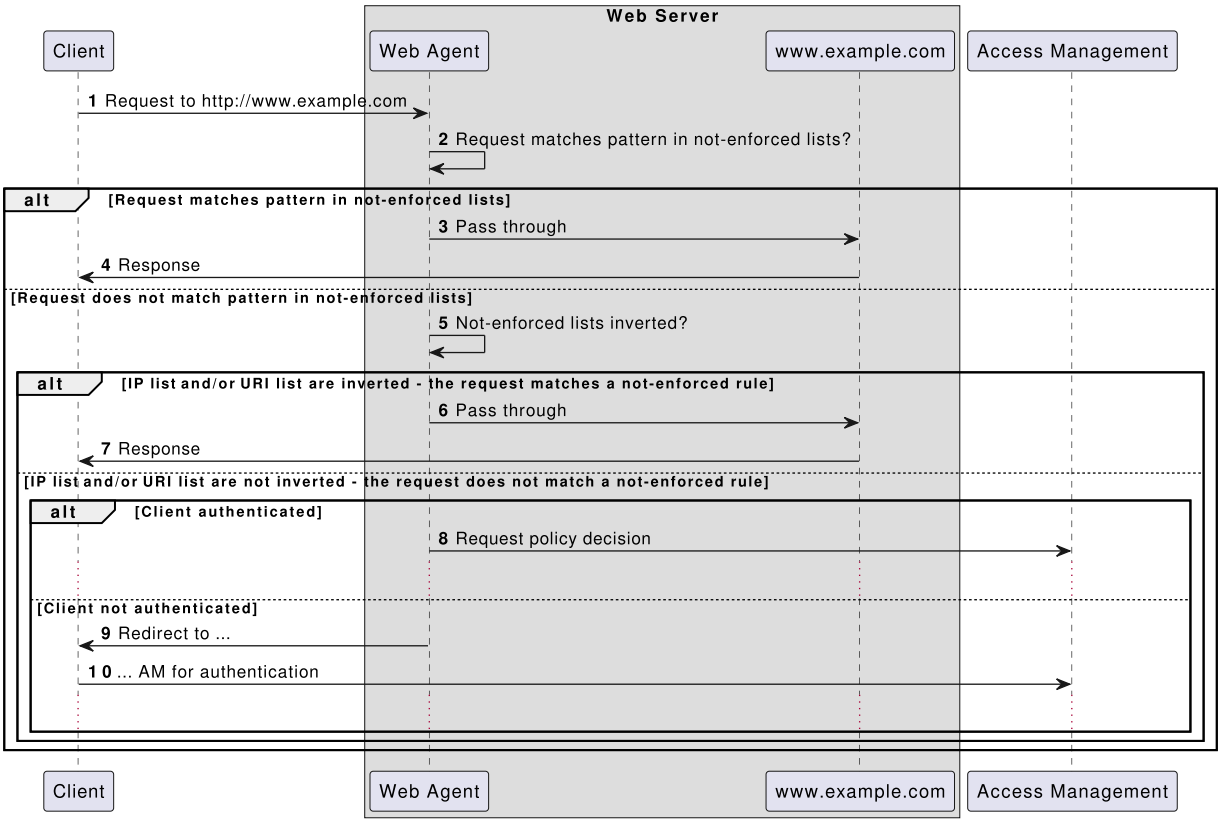
Some resources, such as the "public" directory of a web application, contain data that is not sensitive. It can be accessed by any, authenticated or unauthenticated, clients. The agent uses lists of *not-enforced rules* to identify these resources in the web application.

The agent matches incoming requests to the lists of not-enforced rules. When a request matches a not-enforced rule, the agent bypasses the call to AM:

- If an unauthenticated user sent the request, the agent does not redirect the user to log in.
- If an authenticated user sent the request, the agent does not request a policy evaluation from AM.

Use not-enforced rules to reduce the number of unnecessary calls to AM, and therefore improve the performance and speed of your application.

The following image shows the data flow when Web Agent evaluates not-enforced rules for a request:



1-2. A client requests a resource and the agent checks whether the request matches a rule in a not-enforced list.

3-5. If the request matches a rule, the agent passes the request without requiring authentication or policy decisions. Otherwise, the agent checks whether rules are inverted.

6-10. If the request matches an inverted rule, the agent passes the request without requiring authentication or policy decisions. Otherwise, the agent enforces authentication and policy decisions.

Conventions for not-enforced rules

Consider the following points about not-enforced rules:

- Web servers normalize request URLs as described in RFC 3986: Normalization and comparison before passing them to the agent. The agent compares the normalized URL to the not-enforced rule.
- Trailing forward-slashes `/` can represent a directory. Therefore, `/images/` does not match `/images`, but does match `/images/index.html`

Invert not-enforced URL rules

Invert all rules in a not-enforced URL list by setting Invert Not-Enforced URLs to `true`.

Consider the following points when you invert all rules:

- If Not-Enforced URL List is empty, all URLs are enforced.
- At least one URL must be enforced. To allow access to any URL without authentication, consider disabling the agent.

Wildcards

For more information about using wildcards, refer to Specifying resource patterns with wildcards.

Multi-level wildcard (*)

The following list summarizes the behavior of the multi-level wildcard (`*`):

- Matches zero or more occurrences of any character except for the question mark (`?`).
- Spans multiple levels in a URL.
- Cannot be escaped. Therefore, the backslash (`\`) or other characters cannot be used to escape the asterisk, as such `\*`.
- Cannot be used in the same rule as the one-level wildcard (`-*-`) or regular expression.
- Explicit patterns are required to match URL parameters. For example:
  - URL patterns ending with `/foo*` do not match URLs with parameters
  - URL patterns ending with `/foo*?*` match any parameter

Multi-level wildcard for not-enforced IP rules

Rules in Not-Enforced IP List	Matches request IP	Does not match request IP
192.168.1.*	192.168.1.0 192.168.1.0/24	192.168.0.1

Multi-level wildcard for not-enforced URI rules

Rules in Not-Enforced URL List	Matches request URL	Does not match request URL
http://A-examp.com:8080/*	http://A-examp.com:8080/ http://A-examp.com:8080/index.html http://A-examp.com:8080/x.gif	http://B-examp.com:8080/ http://A-examp.com:8090/index.html http://A-examp.com:8080/a?b=1
http://A-examp.com:8080/*.html	http://A-examp.com:8080/index.html http://A-examp.com:8080/pub/ab.html http://A-examp.com:8080/pri/xy.html	http://A-examp.com/index.html http://A-examp.com:8080/x.gif http://B-examp.com/index.html
http://A-examp.com:8080/*/ab	http://A-examp.com:8080/pri/xy/ab/xy/ab http://A-examp.com:8080/xy/ab	http://A-examp.com/ab http://A-examp.com/ab.html http://B-examp.com:8080/ab

Rules in Not-Enforced URL List	Matches request URL	Does not match request URL

`http://A-examp.com:8080/ab/*/de`	`http://A-examp.com:8080/ab/123/de`
	
	`http://A-examp.com:8080/ab/ab/de`
	
	`http://A-examp.com:8080/ab/de/ab/de`
		`http://A-examp.com:8080/ab/de`
		
		`http://A-examp.com:8090/ab/de`
		
		`http://B-examp.com:8080/ab/de/ab/de`

One-level wildcard (`-*-`)

The following list summarizes the behavior of the one-level wildcard (`-*-`):

- Matches zero or more occurrences of any character except for the forward-slash (`/`) and the question mark (`?`).
- Does not span across multiple levels in a URL.
- Cannot be escaped. Therefore, the backslash (`\`) or other characters cannot be used to escape the hyphen-asterisk-hyphen, like this `\-*-` .
- Cannot be used in the same rule as the multi-level wildcard (`*`) or regular expression.

One-level wildcard for not-enforced URI rules

Rules in Not-Enforced URL List	Matches request URL	Does not match request URL

Rules in Not-Enforced URL List	Matches request URL	Does not match request URL

`http://A-examp.com:8080/b/-*-`	`http://A-examp.com:8080/b/`
	
	`http://A-examp.com:8080/b/cd`
		`http://A-examp.com:8080/b`
		
		`http://A-examp.com:8080/b/cd/` (This URL should match the rule, but does not because of the known issue AMAGENTS-4672.)
		
		`http://A-examp.com:8080/b/c?d=e`
		
		`http://A-examp.com:8080/b/cd/e`
		
		`http://A-examp.com:8090/b/`

`http://A-examp.com:8080/b/-*-/f`	`http://A-examp.com:8080/b/c/f`
	
	`http://A-examp.com:8080/b/cde/f`
		`http://A-examp.com:8080/b/c/e/f`
		
		`http://A-examp.com:8080/f/`

`http://A-examp.com:8080/b/c-*-/f`	`http://A-examp.com:8080/b/cde/f`
	
	`http://A-examp.com:8080/b/cd/f`
	
	`http://A-examp.com:8080/b/c/f`
		`http://A-examp.com:8080/b/c/e/f`
		
		`http://A-examp.com:8080/b/c/`
		
		`http://A-examp.com:8080/b/c/fg`

Multiple wildcards

When multiple wildcards are included in the same rule of a Not-Enforced URL List, the agent matches the parameters in any order that they appear in a resource URI.

For example, the following rule applies to any resource URI that contains a `member_level` and `location` query parameter, in any order:

```
com.sun.identity.agents.config.notenforced.url[1]=http://www.exam
```

```
ple.com:8080/customers/*?*member_level=*&location=*
```

In following example, the requests would be not-enforced:

```
https://www.example.com/customers/default.jsp?
member_level=silver&location=fr
https://www.example.com/customers/default.jsp?
location=es&member_level=silver
https://www.example.com/customers/default.jsp?
location=uk&vip=true&member_level=gold
```

Regular expressions

IMPORTANT

Regular expressions are evaluated differently by different engines. When you use regular expressions in not-enforced lists, make sure that the expressions are evaluated in the way you expect. Double check that the correct URLs are enforced and not enforced.

Set Regular Expressions for Not-Enforced URLs to `true`, and consider the following points for using regular expressions in not-enforced rules:

- Wildcards cannot be used. The asterisk `*` is not treated as a wildcard, but is treated as part of the expression, representing repetition of the last character 0-n times.
- The following formats cannot be used:
  - Netmask CIDR notation
  - IP address ranges

  However, regular expressions can match a range of IP addresses, such as:

  ```
  com.sun.identity.agents.config.notenforced.ip[1]=192\.168\.10\
  .(10|\d)
  ```
- If an invalid regular expression is specified in a rule, the rule is dropped, and an error message is logged.

HTTP Methods

Rules that apply an HTTP method filter are configured as custom properties in AM.

Add one or more HTTP method keywords to the not-enforced rule to apply it when the incoming request uses the HTTP method. Keywords include but are not restricted to `GET`, `HEAD`, `POST`, `PUT`, `PATCH`, `DELETE`, and `OPTIONS`.

By default, no HTTP method is specified for a rule, and all methods are not-enforced for that rule. When one or more HTTP methods are specified, only those methods are not-enforced; methods that are not specified are enforced.

The following example does not require authentication for OPTIONS requests to your scripts, but does require authentication for other HTTP methods:

```
com.sun.identity.agents.config.notenforced.url[OPTIONS,1]=http://
www.example.com:8080/scripts/*
```

To specify a list of methods, add multiple rules:

```
com.sun.identity.agents.config.notenforced.url[OPTIONS]=http://ww
w.example.com:8080/scripts/*
com.sun.identity.agents.config.notenforced.url[PATCH]=http://www.
other.com:8080/scripts/*
com.sun.identity.agents.config.notenforced.url[PATCH1,PATCH2]=htt
p://www.example.com:8080/scripts/*
```

Unrecognized methods can invalidate a rule.

Compound rules

Configure compound rules in Not-Enforced URL from IP Processing List.

In the following example, the agent does not enforce HTTP requests from the IP addresses `192.6.8.0/24` to any file in `/public`, or any files or directories that start with the string `login` in the directory `/free_access URI`:

```
org.forgerock.agents.config.notenforced.ipurl[1]=192.6.8.0/24|htt
p://www.example.com:8080/public/* */free_access/login*
```

Encoding non-ASCII characters in rules

Percent-encode resources that use non-ASCII characters.

For example, to match resources to the URI `http://www.example.com/forstå`, specify the following percent-encoded rule:

```
/forst%C3%A5/*
```

Continuous security

When a user requests a resource through AM, excluding proxies and load balancers, the Web Agent is usually the first point of contact. Because Web Agent is closer to the user than AM, and outside the firewalls that separate the user and AM, the Web Agent can sometimes gather information about the request, which AM cannot access.

When Web Agent requests a policy decision from AM, it can include the additional information in an *environment map*, a set of name/value pairs that describe the request IP and DNS name, along with other, optional, information. The additional information can then be included in the policy, for example, to allow only incoming requests that contain the `InternalNetwork`.

In AM, use server-side authorization scripts to access the environment map, and write scripted conditions based on cookies and headers in the request. For information about server-side authorization scripts, refer to Scripting a policy condition in AM's *Authorization guide*.

Environment maps with customizable keys

In Web Agent, use the continuous security properties Continuous Security Cookie Map and Continuous Security Header Map to configure an environment map with the following parts:

requestIp
The IP address of the inbound request, determined as follows:

- If Client IP Address Header is configured, Web Agent extracts the IP address from the header.
- Otherwise, Web Agent uses the web server connection information to determine the client IP address.

This entry is always created in the map.

requestDNSName
The host name address of the inbound request, determined as follows:

- If Client Hostname Header is configured, Web Agent extracts the host name from the header.
- Otherwise, Web Agent uses the web server connection information to determine the client's host name.

This entry is always created in the map.

Other variable names

An array of cookie or header values. An entry is created for each value specified in the continuous security properties.

In the following example, the continuous security properties are configured to map values for the `ssid` cookie and `User-Agent` header to fields in an environment map:

```
org.forgerock.openam.agents.config.continuous.security.cookies[
ssid]=mySsid
org.forgerock.openam.agents.config.continuous.security.headers[
User-Agent]=myUser-Agent
```

If the incoming request contains an `ssid` cookie and a `User-Agent` header, the environment map takes the value of the cookie and header, as shown in this example:

```
requestIp=192.16.8.0.1
requestDnsName=client.example.com
mySsid=77xe99f4zqi1l99z
myUser-Agent=Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0)
like Gecko
```

Caching

Web Agent supports the following caches to speed up agent operations:

Configuration cache

The configuration cache stores web agent configuration properties.

When a Web Agent starts up, it either makes a call to AM to retrieve a copy of the agent profile (centralized configuration mode) or reads the agent profile from the local configuration file (local configuration mode). Then, the agent stores the configuration in its cache. The cached information is valid until one of the following events occur:

- AM notifies the agent of changes to hot-swappable agent configuration properties. This only applies to deployments that use centralized configuration mode.
- The information in the cache reaches the expiration time specified by Configuration Reload Interval.

  When a configuration property in the cache is invalid, the agent clears the cached property value and rereads it from the agent profile.

Session and policy decision cache

Stored in the shared memory pool defined by the `AM_MAX_SESSION_CACHE_SIZE` environment variable, the session and policy decision cache stores session information, and the results of previous policy decisions.

The default size of the cache is 16 MB, but you may need to increase its size if you plan to hold many active sessions in the cache at any given time. For more information about the environment variable, refer to Environment variables.

After authentication, AM presents the client with an ID token containing session information. The web agent stores part of that session information in the cache. When a client attempts to access a protected resource, the agent checks whether there is a policy decision cached for the resource:

- If there is a cached policy decision, the agent reuses it without contacting AM.
- If there is no cached policy decision, the validity of the client's session determines the agent's behavior:
    - If the client's session is valid, the web agent requests a policy decision from AM, caches it, and then enforces it.
    - If the client's session is not valid, the agent redirects the client to AM for authentication regardless of why the session is invalid. The agent does not specify the reason why the client needs to authenticate.

      After the client authenticates, and the session is cached, the agent requests a policy decision from AM, caches it, and then enforces it.

  Session and policy decisions are valid in the cache until one of the following events occur:

Session and policy decision validity in cache

Event	What is invalidated?
Session contained in the ID token expires	Session and policy decisions related to the session
Client logs out from AM (and session notifications are enabled)	Session and policy decisions related to the session
Session reaches the expiration time specified by SSO Cache Polling Period.	Session
Policy decision reaches the expiration time specified by Policy Cache Polling Period.	Policy decision

Event	What is invalidated?
Administrator makes a change to policy configuration (and policy notifications are enabled)	All sessions and all policy decisions

IMPORTANT

A Web Agent that loses connectivity with AM cannot request policy decisions. Therefore, the agent denies access to inbound requests that do not have a policy decision cached until the connection is restored(*).

Policy cache

The policy cache builds upon the session and policy decision cache. It downloads and stores details about policies from AM, and uses the downloaded policies to make authorization decisions, without contacting AM each time.

Web Agent uses the policy cache without contacting AM in the following situations:

- A requested resource matches the resource pattern of a policy that has been cached due to a previous evaluation.
- A requested resource **does not** match a pattern of policy rules downloaded locally. In this case, the agent denies access.
- A requested resource matches the resource pattern of a simple policy that applies to the `All Authenticated Users` virtual group.

  If the resource matches the policy used for a previous policy decision, the agent does not request policy evaluation from AM. Therefore, policy conditions based on scripts, LDAP filter conditions, or session properties, which rely on attributes that can vary during a session, may not be enforced.

  To reduce this risk, you should:
- Enable session property change notifications, as described in Notifications.
- Reduce the amount of time that sessions can remain in the agent session cache.

Consider the following caveats when using the policy cache:

- If you have a large number of policies, for example more than one million in an UMA deployment, the time to download the policies and the memory consumption of the agent may affect performance.
- The agent downloads the policy rules, and uses them to evaluate policies locally. If a policy is customized in AM in a way that changes the way it is evaluated (for example, a wildcard or delimiter is changed), the policy decision made by the agent might not match the policy defined in AM.

- Even though delimiters and wildcards are configurable in AM (**Configure** > **Global Services** > **Policy Configuration** > **Global Attributes** > **Resource Comparator**), the policy cache supports only the default configuration.

Do not enable the agent's policy cache if your policies use custom delimiters and/or wildcards.

Enable the policy cache by creating an environment variable named `AM_POLICY_CACHE_MODE`.

Change the location of the policy cache by creating an environment variable named `AM_POLICY_CACHE_DIR`.

For more information about properties related to the policy cache, refer to Environment variables.

Attribute fetch modes

For information about properties to configure attribute fetching, refer to Attribute processing.

Web Agent can fetch user information, and inject it into HTTP request headers and cookies, and pass them on to the protected client applications. The client applications can then personalize content using these attributes in their web pages or responses.

IMPORTANT

When injecting information into HTTP headers, do not use underscores (_) in the header name. Underscores are incompatible with systems that run CGI scripts, and the header can be silently dropped.

You can configure the type of attributes to be fetched, and the associated mappings for the attributes names used in AM, to those values used in the web server. The agent securely fetches the user and session data from the authenticated user, as well as policy response attributes.

For example, you can have a web page that addresses the user by name retrieved from the user profile, for example "Welcome Your-Name!". AM populates part of the request (header, form data) with the CN from the user profile, and the website consumes and displays it.

SSO-only mode

Web Agent intercepts all inbound client requests to access a protected resource and processes the request based on the Enable SSO Only Mode property.

The configuration setting determines the mode of operation that should be carried out on the intercepted inbound request, as follows:

- When `true`, the agent manages user authentication only. The filter invokes the AM Authentication Service to verify the identity of the user. If the identity is verified, the user is issued a session token through AM's session service.
- When `false`, which is the default, the agent also manages user authorization, by using the policy engine in AM.

FQDN checking

When FQDN checking is enabled, the agent checks the FQDN of a request before it evaluates the authentication or authorization of the request, as follows:

- If the request matches the default domain, or the value of a mapped domain, the agent passes the request on to the next step without changing the domain.
- Otherwise, the agent redirects the request to the specified domain before passing it on to the next step.

Use this feature to prevent the redirect of requests in the following scenarios:

- Where resource URLs differ from the FQDNs in AM policies, for example, in load balanced and virtual host environments.
- Where hostnames are virtual, allocated dynamically, or match a pattern, for example in a Kubernetes deployment.
- Where hostnames are partial.

FQDN checking requires Enable FQDN Check to be `true`, FQDN Default to be set to a suitable value, and optionally, FQDN Virtual Host Map to map incoming URLs to valid outgoing domains.

The agent maps FQDNs as follows:

1. If the request matches the domain in FQDN Default, the agent passes the request to the next step without changing the request domain.
2. Otherwise, if the request matches a mapped domain (map value) in FQDN Virtual Host Map, the agent passes the request to the next step without changing the request domain.
3. Otherwise, if the request matches a mapped host (map key) in FQDN Virtual Host Map, the agent redirects the request to the mapped domain before passing it to the next step.
4. Otherwise, the agent redirects the request to the domain in FQDN Default before passing it to the next step.

Examples

The following example configuration and requests illustrate how the agent checks and remaps FQDNs:

Example configuration

- Agent Root URL for CDSSO:

  ```
  sunIdentityServerDeviceKeyValue=agent.example.com
  ```

- Not-Enforced URL List

  ```
  com.sun.identity.agents.config.notenforced.url[0]=http://www.agent1*.example.com
  ```

- Enable FQDN Check:

  ```
  com.sun.identity.agents.config.fqdn.check.enable=true
  ```

- FQDN Default:

  ```
  com.sun.identity.agents.config.fqdn.default=agent.default.com
  ```

- FQDN Virtual Host Map:

  ```
  com.sun.identity.agents.config.fqdn.mapping[agent.example.com]=agent.example.com
  ```

  ```
  com.sun.identity.agents.config.fqdn.mapping[agent.example.com]=agent-*
  ```

  ```
  com.sun.identity.agents.config.fqdn.mapping[any.value.com]=ag*.example.com
  ```

  ```
  com.sun.identity.agents.config.fqdn.mapping[agent.othertest.com]=other.example.com
  ```

Example requests

- `http://agent.default.com/app` : The request URL matches the domain of the default mapping, so the agent does not redirect the request.
- `https://agent.example.com/app` : The request URL matches the value (domain) in the first mapping, so the agent does not redirect the request.
- `http://agent-4738294739287492/foo/bar/` : The request URL matches the value (domain) in the second mapping, using the wildcard, so the agent does not redirect the request. Note that the value of the key is irrelevant in this match.
- `https://agent123.example.com/app` : The request URL matches the value (domain) in the third mapping, so the agent does not redirect the request.
- `https://agent.othertest.me/app` : The request URL matches the key (host) in the fourth mapping, so the agent redirects the request to

`https://other.example.com/app`.

- `https://agent.othertest2.me/app`: The request URL doesn't match any mapping, so the agent redirects the request to the default domain, `https://agent.example.com/app`.

Cookie reset

Web Agent can reset cookies before redirecting the client to a login page, by issuing a Set-Cookie header to the client to reset the cookie values.

Cookie reset is typically used when multiple parallel authentication mechanisms are in play with the web agent and another authentication system. The agent can reset the cookies set by the other mechanism before redirecting the client to a login page.

NOTE

To set and reset secure or HTTP Only cookies, in addition to the cookie reset properties, set the relevant cookie option, as follows:

- To reset secure cookies, enable the `com.sun.identity.agents.config.cookie.secure` property.
- To reset HTTP only cookies, enable the `com.sun.identity.cookie.httponly` property.

If you have enabled attribute fetching by using cookies to retrieve user data, it is good practice to use cookie reset, which will reset the cookies when accessing an enforced URL without a valid session.

Configure load balancers and reverse proxies

Most environments deploy a load balancer and reverse proxy between the agent and clients, and another between the agent and AM, as shown in the following diagram:

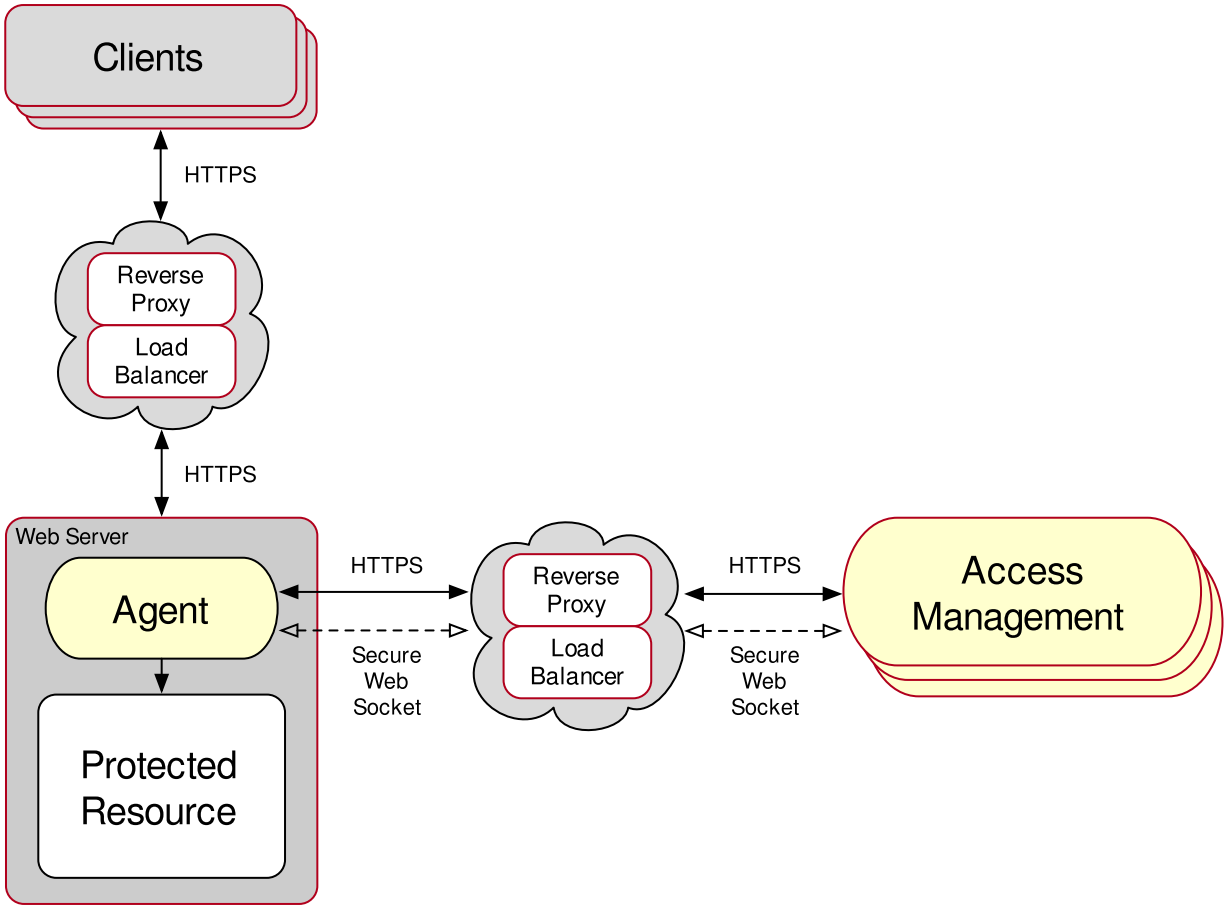


Figure 1. Environments with Load Balancers and Reverse Proxies

The reverse proxy and the load balancer can be the same entity. In complex environments, multiple load balancers and reverse proxies can be deployed in the network.

Identifying clients behind load balancers and reverse proxies

When a load balancer or reverse proxy is situated in the request path between the agent and a client, the agent does not have direct access to the IP address or hostname of the client. The agent cannot identify the client.

For load balancers and reverse proxies that support provision of the client IP and hostname in HTTP headers, configure the following properties:

- Client IP Address Header
- Client Hostname Header

When there are multiple load balancers or reverse proxies in the request path, the header values can include a comma-separated list of values, where the first value represents the client, as in `client,next-proxy,first-proxy`.

Agent connection to AM through a load balancer/reverse proxy

When a reverse proxy is situated between the agent and AM, it can be used to protect the AM APIs.

When a load balancer is situated between the agent and AM, it can be used to regulate the load between different instances of AM.

Consider the points in this section when installing Web Agent in an environment where AM is behind a load balancer or a reverse proxy.

Agent's IP address and/or FQDN

The load balancer or reverse proxy conceals the IP addresses and FQDNs of the agent and of AM. Consequently, AM cannot determine the agent base URL.

To prevent problems during installation or redirection, do the following:

- Configure the load balancer or reverse proxy to forward the agent IP address and/or FQDN in a header.
- Configure AM to recover the forwarded headers. For more information, see Configuring AM to Use Forwarded Headers.
- Install the agent using the IP address or FQDN of the load balancer or reverse proxy as the point of contact for the AM site.

AM sessions and session stickiness

Improve the performance of policy evaluation by setting AM's sticky cookie (by default, `amlbcookie`) to the AM's server ID. For more information, refer to Configuring site sticky load balancing in AM's *Setup guide*.

When configuring multiple agents, consider the impact on sticky load balancer requirements of using one or multiple agent profiles:

- If agents are configured with multiple agent profiles, configure sticky load balancing. The agent profile name is contained in the OpenID Connect JWT, used by the agent and AM for communication. Without session stickiness, it is not possible to make sure the appropriate JWT ends in the appropriate agent instance.

  To have multiple agent profiles without sticky load balancing, disable validation of the `aud` claim in the session ID token. Either enable Disable Audience Claim Validation, or configure Agent Profile ID Allow List.

  For security reasons, agents should validate all claims in session ID tokens. Therefore, use this approach sparingly and mostly for migrations.
- If multiple agents are configured with the same agent profile, decide whether to configure sticky load balancing depending on other requirements of your environment.

WebSockets

For communication between the agents and the AM servers, the load balancers and reverse proxies must support the WebSocket protocol. For more information, refer to the load balancer or proxy documentation.

TIP

For an example of how to configure Apache HTTP as a reverse proxy, refer to Configure Apache HTTP Server as a reverse proxy.

Configuring AM to use forwarded headers

When a load balancer or reverse proxy is situated between the agent and AM, configure AM to recover the forwarded headers that expose the agents' real IP address or FQDN.

To configure how AM obtains the base URL of web agents, use the Base URL Source service:

1. Log in to the AM admin UI as an administrative user, such as `amAdmin`.
2. Select **REALMS** > Realm Name > **Services**.
3. Select **Add a Service** > **Base URL Source**, and create a default service, leaving the fields empty.
4. Configure the service with the following properties, leaving the other fields empty:
   - **Base URL Source**: `X-Forwarded-* headers`

     This property allows AM to retrieve the base URL from the `Forwarded` header field in the HTTP request. The Forwarded HTTP header field is specified in RFC 7239: Forwarded HTTP Extension.
   - **Context path**: AM's deployment URI. For example, `/am`.

   For more information, refer to Base URL source in AM's *Reference*.
5. Save your changes.

Agent connection to clients through a load balancer/reverse proxy

When a reverse proxy is situated between the agent and client, it can be used to anonymize the client traffic that enters the network.

When a load balancer is situated between the agent and client, it can be used to regulate the load between the agents and the web application servers.

Consider the points in this section when installing Web Agent in an environment where clients are behind a load balancer or a reverse proxy.

Client's IP Address and/or FQDNs

The load balancer or reverse proxy conceals the IP addresses and FQDNs of the agent and clients. Consequently, the agent cannot determine the client base URL.

Configure the load balancer or reverse proxy to forward the client IP address and/or the client FQDN in a header. Failure to do so prevents the agent from performing policy evaluation, and applying not-enforced and conditional login/logout rules.

For more information, refer to Configuring client identification properties.

POST data preservation and sticky load balancing

For POST data preservation, use sticky load balancing to ensure that after login the client hits the same agent as before and can therefore get their saved POST data.

Agents provide properties to set either sticky cookie or URL query string for load balancers and reverse proxies.

For more information, refer to Configure POST Data Preservation for Load Balancers or Reverse Proxies.

Mapping agent host name to a load balancer or reverse proxy

In the following diagram, the agent and load balancer/reverse proxy use the same protocol and port, but different FQDNs:

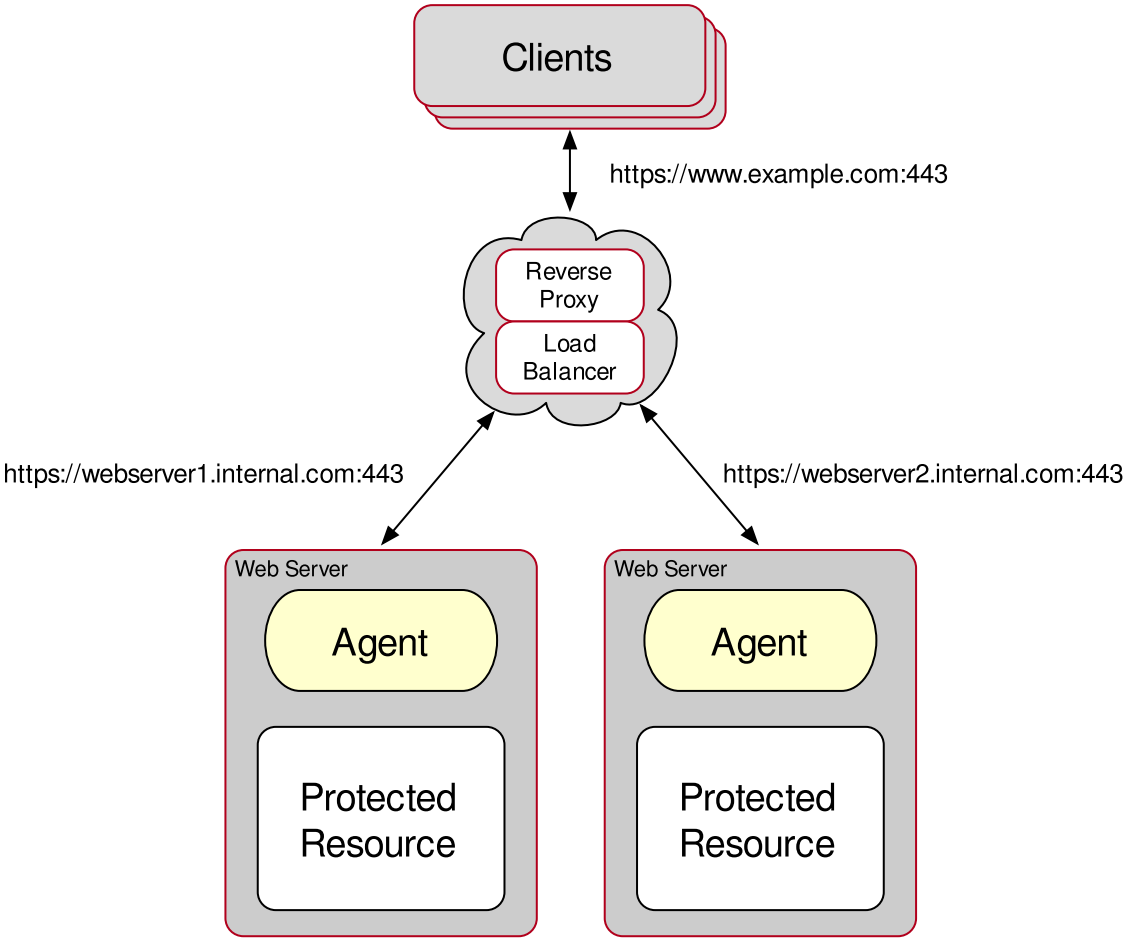


Figure 2. Same protocol and port, different FQDN

Map the host name of the agent to that of the load balancer or reverse proxy.

1. Log in to the AM admin UI as an administrative user with rights to modify the web agent profile.
2. Go to **REALMS** > Realm Name > **Applications** > **Agents** > **Web** > Agent Name.
3. Set the following options in the **Global** tab:
   - **FQDN Check**: Enable

     The equivalent property setting is Enable FQDN Check = `true`.
   - **FQDN Default**: Set to the FQDN of the load balancer or proxy, for example `lb.example.com`. Do not set it to the protected server FQDN where the agent is installed.

     The equivalent property setting is FQDN Default = `lb.example.com`.
   - **Agent Root URL for CDSSO**: Set to the FQDN of the load balancer or proxy, for example `https://lb.example.com:80/`.

     The equivalent property setting is Agent Root URL for CDSSO = `lb.example.com`.

- **FQDN Virtual Host Map**: Map the load balancer or proxy FQDN to the FQDN where the agent is installed. For example,
  - **Key**: `agent.example.com` (protected server)
  - **Value**: `lb.example.com` (load balancer or proxy)

    The equivalent property setting is `com.sun.identity.agents.config.fqdn.mapping[agent.example.com]=lb.example.com`.

4. Save your work.

Overriding protocol, host, and port

The load balancer or reverse proxy forwards requests and responses between clients and protected web servers. In the following diagram, the protocol, port, and FQDN configured on the load balancer and reverse proxy are different than those on the protected web server:

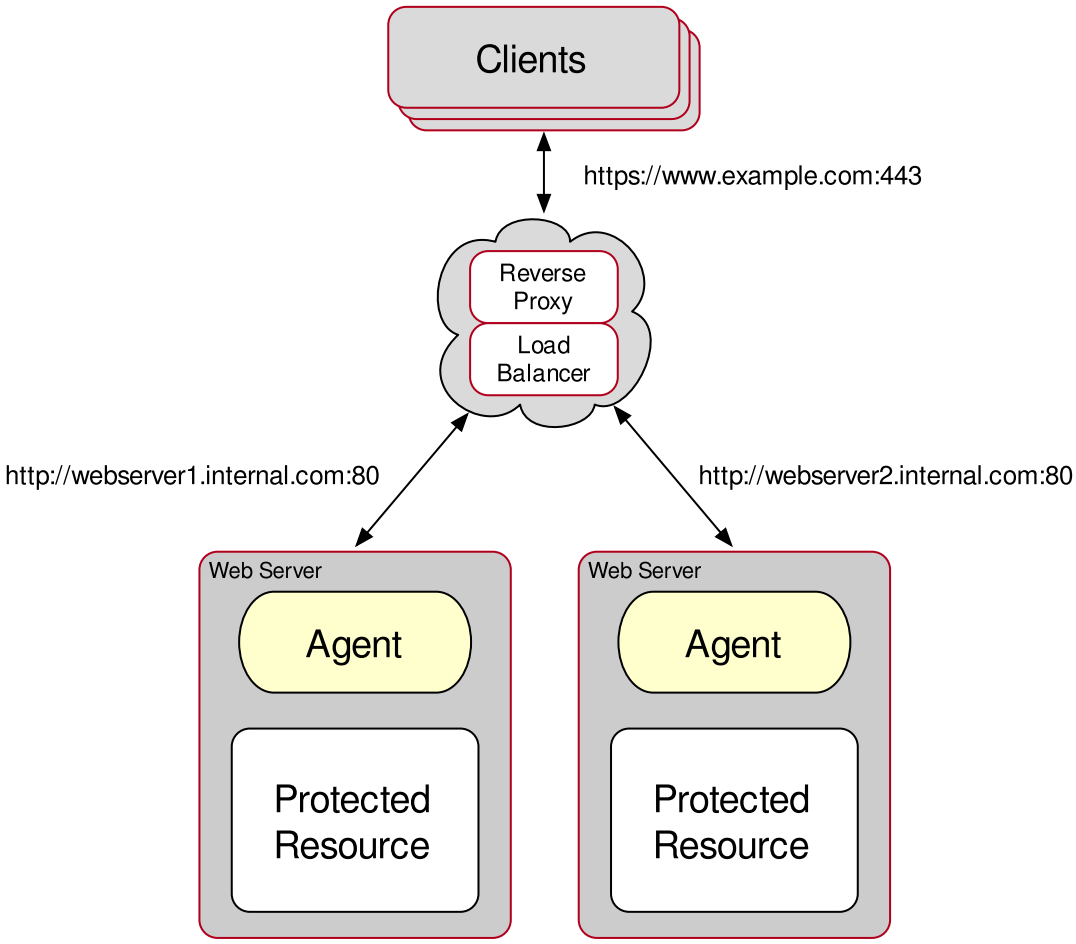


Figure 3. Different protocol, port, and FQDN

IMPORTANT

IMPORTANT

Agent configuration for TLS offloading prevents FQDN checking and mapping. Consequently, URL rewriting and redirection do not work correctly when the agent is accessed directly, instead of through the load balancer or proxy. This should not be a problem for client traffic, but could be a problem for web applications accessing the protected web server directly, from behind the load balancer.

NOTE

When the following headers are defined on the proxy or load-balancer, they override the value of Agent Deployment URI Prefix:

- `X-Forwarded-Proto`
- `X-Forwarded-Host`
- `X-Forwarded-Port`

If you are using these headers, do not configure the agent to override its hostname, port, or protocol.

Use Agent Deployment URI Prefix to override the agent protocol, host, and port with that of the load balancer or reverse proxy.

1. Log in to the AM admin UI as an administrative user with rights to modify the agent profile.
2. Go to **REALMS** > Realm Name > **Applications** > **Agents** > **Web** > Agent Name.
3. Set the following options in the **Global** tab:
   - **Agent Deployment URI Prefix**: Set to the URI of the load balancer or proxy. For example, `https://external.example.com:443`.

     This value is used to override the protocol, host, and port of the protected server.

     The equivalent property setting is Agent Deployment URI Prefix = `https://external.example.com:443`.
   - **Agent Root URL for CDSSO**: Set to the FQDN of the load balancer or proxy, for example `https://lb.example.com:80/`.

     The equivalent property setting is Agent Root URL for CDSSO = `lb.example.com`.
4. Enable the following options in the **Advanced** tab:
   - **Override Request URL Protocol**

     The equivalent property setting is Enable Override Request URL Protocol = `true`.

- **Override Request URL Host**

  The equivalent property setting is Enable Override Request URL Host = `true`.

- **Override Request URL Port**

  The equivalent property setting is Enable Override Request URL Port = `true`.

5. Save your work.

Configuring client identification properties

After configuring proxies or load balancers to forward the client FQDN and/or IP address, configure the agents to check the appropriate headers.

This procedure explains how to configure the client identification properties for a centralized web agent profile configured in the AM admin UI. The steps also mention the properties for web agent profiles that rely on local, file-based configurations:

1. Log in to the AM admin UI with a user that has permissions to modify the web agent profile.
2. Go to **REALMS** > Realm Name > **Applications** > **Agents** > **Web** > Agent Name.
3. Set the following options in the **Advanced** tab:
   - **Client IP Address Header**: Configure the name of the header containing the IP address of the client. For example, `X-Forwarded-For`.

     The equivalent property setting is Client IP Address Header = `X-Forwarded-For`.

     Configure this property if any of the following points are true:

     - AM policies are IP address-based.
     - The agent is configured for not-enforced IP rules.
     - The agent is configured take any decision based on the client's IP address.

   - **Client Hostname Header**: Configure the name of the header containing the FQDN of the client. For example, `X-Forwarded-Host`.

     The equivalent property setting is Client Hostname Header = `X-Forwarded-Host`.

     Configure this property if any of the following points are true:

     - AM policies are URL address-based.

- The agent is configured for not-enforced URL rules.
- The agent is configured take decisions based on the client's URL address.

4. Save your changes.

Configuring POST Data Preservation for Load Balancers or Reverse Proxies

Use one of the following procedures to configure post data preservation for load balancers or reverse proxies.

Map one agent profile to one agent instance when POST data preservation is enabled

In this procedure, a separate agent profile must created in AM for each agent instance. For scalable deployments, where resources are dynamically created and destroyed, use Map one agent profile to multiple agent instances when POST data preservation is enabled instead.

1. Configure your load balancer or reverse proxy to ensure session stickiness when the cookie or URL query parameter are present.
2. Log in to the AM admin UI as a user that has permissions to modify the agent profile.
3. Go to **REALMS** > Realm Name > **Applications** > **Agents** > **Web** > Agent Name.
4. Set the following options in the **Advanced** tab:
   - POST Data Sticky Load Balancing Mode:
     - **COOKIE**: The agent creates a cookie for POST data preservation session stickiness. The content of the cookie is configured in the next step.
     - **URL**: The agent appends to the URL a string specified in the next step.

       The equivalent property setting is `com.sun.identity.agents.config.postdata.preserve.stickysession.mode=COOKIE` or `com.sun.identity.agents.config.postdata.preserve.stickysession.mode=URL`.
   - POST Data Sticky Load Balancing Value: Configure a key-pair value separated by the `=` character.

     The agent creates the value when it evaluates POST Data Sticky Load Balancing Mode. For example, specifying `lb=myserver` either sets a cookie called `lb` with `myserver` as a value, or appends `lb=myserver` to the URL query string.

The equivalent property setting is `com.sun.identity.agents.config.postdata.preserve.stickysession.value=lb=myserver`.

5. Save your changes.

Map one agent profile to multiple agent instances when POST data preservation is enabled

Use this procedure for scalable deployments, where resources can be dynamically created or destroyed. For example, use it in deployments with load balancing, or environments running Kubernetes.

1. Configure your load balancer or reverse proxy to ensure session stickiness when the cookie or URL query parameter are present.
2. For each agent instance, configure post data preservation in the agent configuration file `agent.conf`. This configuration overrides the configuration in AM.

   In the following example, the settings in `agent.conf` configure two agents behind a load balancer to use the same agent profile, and provide uniqueness to the load balancer:

   - Agent 1:

     ```
     com.sun.identity.agents.config.postdata.preserve.stickysession.mode = COOKIE
     com.sun.identity.agents.config.postdata.preserve.stickysession.value = EXAMPLE=Agent1
     ```

   - Agent 2:

     ```
     com.sun.identity.agents.config.postdata.preserve.stickysession.mode = COOKIE
     com.sun.identity.agents.config.postdata.preserve.stickysession.value = EXAMPLE=Agent2
     ```

   For information about the values to use, refer to the following properties:

   - POST Data Sticky Load Balancing Mode
   - POST Data Sticky Load Balancing Value

3. Restart the web server where the agent is installed.

Configure Apache HTTP Server as a reverse proxy

This section contains an example configuration of Apache as a reverse proxy between AM and Web Agent. You can use any reverse proxy that supports the WebSocket protocol.

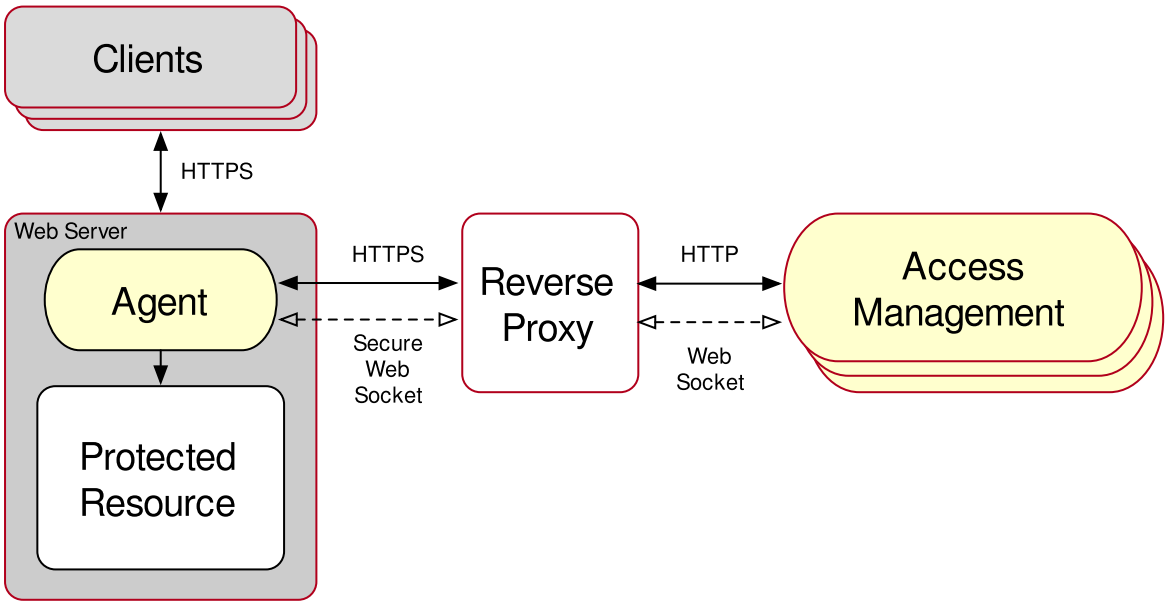


Figure 4. Reverse proxy configured between the agent and AM

For information about how to configure Apache for load balancing, and other requirements for your environment, refer to the Apache documentation.

1. Locate the `httpd.conf` file in your deployed reverse proxy instance.
2. Add the modules required for a proxy configuration, as follows:

```
# Modules required for proxy
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_wstunnel_module
modules/mod_proxy_wstunnel.so
```

   The `mod_proxy_wstunnel.so` module is required to support the WebSocket protocol used for communication between AM and the agents.

3. Add the proxy configuration inside the `VirtualHost` context. Consider the following directives:

```
<VirtualHost 192.168.1.1>
...
# Proxy Config
RequestHeader set X-Forwarded-Proto "https" (1)
ProxyPass "/openam/notifications"
```

```
"ws://am.example.com:8080/am/notifications"
Upgrade=websocket (2)
ProxyPass "/openam" "http://am.example.com:8080/am" (3)
ProxyPassReverseCookieDomain "openam.internal.example.com"
"proxy.example.com" (4)
ProxyPassReverse "/openam" "http://am.example.com:8080/am"
(5)
...
</VirtualHost>
```

(1) RequestHeader: Set to `https` or `http`, depending on the proxy configuration. If the proxy is configured for https, as in the above example, set to `https`. Otherwise, set `http`. In a later step, you configure AM to recognize the forwarded header and use it in the `goto` parameter for redirecting back to the agent after authentication.

(2) ProxyPass: Set to allow WebSocket traffic between AM and the agent. If HTTPS is configured between the proxy and AM, set to use the `wss` protocol instead of `ws`.

(3) ProxyPass: Set to allow HTTP traffic between AM and the agent.

(4) ProxyPassReverseCookieDomain: Set to rewrite the domain string in `Set-Cookie`headers in the format internal domain (AM's domain) public domain (proxy's domain).

(5) ProxyPassReverse: Set to the same value configured for the `ProxyPass` directive.

For more information about configuring Apache as a reverse proxy, refer to the Apache documentation.

4. Restart the reverse proxy instance.
5. Configure AM to recover the forwarded header you configured in the reverse proxy. Also, review other configurations that may be required in an environment that uses reverse proxies. For more information, refer to Regarding communication between AM and agents

Environment variables

Configure environment variables to affect the user that is running the web server, virtual host, or location that the agent protects.

This section describes Web Agent properties that are configured by environment variables. After setting an environment variable, restart Web Agent.

For information about environment variables for installation, refer to Installation environment variables.

For information about allowing environment variables to be used in NGINX, refer to the env directive in the *NGINX Core functionality documentation*.

AM_IPC_BASE

(Unix only) The base number for IPC identifiers used by the agent. The shared memory semaphore ID range used by the agent starts at the specified value. Set this variable only if you detect that the agent semaphores are clashing with those of other processes in your environment.

Default: Arbitrary value

AM_MAX_AGENTS

The maximum number of agent instances in the installation. The higher the number, the more shared memory the agent reserves.

When the maximum is reached, an additional agent instances that starts will log an error, and will not protect resources.

Default: `32`

AM_MAX_SESSION_CACHE_SIZE

The maximum size in bytes of the shared memory for the session and policy cache:

- Not set, or set to `0`: `16777216` (16 MB)
- Maximum value: `1073741824` (1 GB)
- Minimum value `1024` (1 MB)

For multiple concurrent sessions, consider using a higher value.

AM_NET_TIMEOUT

The number of seconds for which the agent installer can contact AM during agent configuration validation.

If the installer takes longer than this value to contact AM and validate the configuration, installation fails.

Default: 4 seconds

Policy evaluation mode (AM_POLICY_CACHE_MODE)

Policy evaluation mode:

- `off` or `0` (default): When a request requires a policy decision, the agent contacts AM for the decision.
- `on` or `1`: The agent downloads all policies from AM at start up. When a request requires a policy decision, the agent uses the downloaded policies to make the

policy decision.

In both modes, the agent caches the policy decision. If a request requires the same policy decision again, the agent uses the cached decision.

(Optional) Use the `AM_POLICY_CACHE_DIR` environment variable to specify a directory in which to store the policy cache.

AM_POLICY_CACHE_DIR

The directory in which to store the policy cache. The agent must be able to write to this directory.

For example, `/path/to/web_agents/agent_type/log`.

AM_RESOURCE_PERMISSIONS

(Unix only) The permissions that the agent sets for its runtime resources.

Allowed values:

- `0600`
- `0660`
- `0666`

The `AM_RESOURCE_PERMISSIONS` environment variable requires the `umask` value to allow these permissions for the files.

Consider an example where the Apache agent is running with the `apache` user. The `umask` value is `0022` and the `AM_RESOURCE_PERMISSIONS` is `0666`. The agent runtime resources have the following permissions:

Resource Permissions Example in Linux

Resource	Permission	Owner
`/path/to/web_agents/agent_type/log/system_n.log`	644	apache
`/path/to/web_agents/agent_type/log/monitor_n.log`	644	apache
`/path/to/web_agents/agent_type/instances/agent_n/conf/agent.conf`	640	apache
`/path/to/web_agents/agent_type/instances/agent_n/logs/debug/debug.log`	644	apache
`/dev/shm/am_cache_0`	644	apache
`/dev/shm/am_log_data_0`	644	apache

Any semaphores owned by the `apache` user have `644` permissions as well.

Consider another example where `umask` is `0002` and `AM_RESOURCE_PERMISSIONS` is `0666`. The files are created with `664` permissions, which allows them to be read and written by the members of the group, as well.

AM_SSL_OPTIONS

Overrides the default SSL/TLS protocols for the agent, set in the Security Protocol List bootstrap property.

Specifies a space-separated list of security protocols preceded by a dash (-) that will *not* be used when connecting to AM.

Supported protocols:

- SSLv3
- TLSv1
- TLSv1.1
- TLSv1.2 (Enabled)
- TLSv1.3 (Enabled)

For example, to configure `TLSv1.1`, set the environment variable to `AM_SSL_OPTIONS = -SSLv3 -TLSv1 -TLSv1.2`.

AM_SYSTEM_LOG_LEVEL

The log level of garbage collector statistics for all Web Agent instances in the web server. The logs are written to the `system_n.log` file, where `n` indicates the agent group number. Consider an environment with two Apache HTTP Server installations:

- `Apache_1` has two agent instances configured, `agent_1` and `agent_2`, configured to share runtime resources (AmAgentId is set to 0). Both agent instances write to the `system_0.log` file.
- `Apache_2` has one agent instance configured, `agent_3`, with AmAgentId set to 1. The instance write to the `system_1.log` file.

The `system_n.log` file can contain the following information:

- Agent version information, written when the agent instance starts up.
- Logs for the agent background processes.
- WebSocket connection errors.
- Cache stats and removal of old POST data preservation files.
- Agent notifications.

Valid values:

- All

- Message
- Warning
- Error (default)
- Info

AM_SYSTEM_LOG_PATH
The full path and filename to the `system_n.log` file.

Default: `/path/to/web_agents/agent_type/log/system_n.log`

AM_SYSTEM_LOG_SIZE
The size in bytes of the `system_n.log` file.

Valid range: 0 (unlimited log file size) to 4294967295 bytes (4GB)

Default: `0`

AM_SYSTEM_PIPE_DIR
(Unix only) The directory where agent instances store temporary pipe files.

Default: `/path/to/web_agents/agent_type/log/`

Troubleshooting

ForgeRock provides support services, professional services, training through ForgeRock University, and partner services to help you set up and maintain your deployments. For information about getting support, refer to Getting support.

When you are trying to solve a problem, save time by asking the following questions:

- How do you reproduce the problem?
- What behavior do you expect, and what behavior do you see?
- When did the problem start occurring?
- Are their circumstances in which the problem does not occur?
- Is the problem permanent, intermittent, getting better, getting worse, or staying the same?

If you contact ForgeRock for help, include the following information with your request:

- Description of the problem, including when the problem occurs and its impact on your operation.
- The product version and build information.
- Steps you took to reproduce the problem.

- Relevant access and error logs, stack traces, and core dumps.
- Description of the environment, including the following information:
  - Machine type
  - Operating system and version
  - Web server and version
  - Java version
  - Patches or other software that might affect the problem

The following items are some common issues and solutions:

TIP

The **`agentadmin`** command offers a validation mode for the agent that can help you troubleshoot issues in your environment; for example, after an agent upgrade or a network change. For more information, refer to the `--V[i]` option in agentadmin command.

▼ HTTP headers are ignored

Question

When I map a response or attribute to an HTTP header, using the following properties, why is the header ignored:

- Session Attribute Map
- Response Attribute Map
- Profile Attribute Map

Answer

When injecting information into HTTP headers, do not use underscores (_) in the header name. Underscores are incompatible with systems that run CGI scripts, and the header can be silently dropped.

▼ Error installing Agents on Windows

Question

I am trying to install a web agent on Windows, which will connect to an AM server running over HTTPS, but the installer reports the following error:

```
init_ssl(): ssleay32.dll is not available (error: 87)
init_ssl(): libeay32.dll is not available (error: 87)
```

Answer

If OpenSSL is correctly installed, on Windows 7 or Windows Server 2008 R2 systems, apply the update provided in Microsoft knowledge base article KB2533623. See Microsoft Security Advisory: Insecure library loading could allow remote code execution.

▼ Error installing Agents With SELinux

Question

I am trying to install Web Agent on a server with SELinux enabled in `enforcing` mode and I am getting error messages after installation, or the web server does not start up. What happened?

Answer

When installing Web Agent on Linux or Unix servers, you must ensure that the user that runs the web server process has read and write permissions for the agent installation directory and files.

If SELinux is enabled in `enforcing` mode, you must also ensure that SELinux is configured to allow the web server process to perform read and write operations to the agent installation directory and files. By default, SELinux only allows the web server process to read files in well-known authorized locations, such as the `/var/www/html` directory.

For environments where security can be more relaxed, consider setting SELinux or the `httpd_t` context in `permissive` mode for troubleshooting purposes.

For information about configuring SELinux, refer to the Linux documentation.

▼ Logs Not Written

Question

Why are logs not being written to `/log/system_0.log` and `/log/monitor_0.pipe` files? I am seeing this error:

```
unable to open event channel
```

Answer

It is likely that the agent does not have permission to be able to write to the `/log/system_0.log` and `/log/monitor_0.pipe` log files.

This can occur if you used the **`agentadmin --V[i]`** validator command using a user account that is different to the account used to run your web server.

Run the validator command as the same user that runs the web server, for example, by using the **`sudo`** command.

To fix the issue, change the ownership of these files to match the user or group that is running your web server.

▼ Port 80 Used as Default

Question

My Apache HTTP server is not using port 80. When I install Web Agent it defaults to port 80. How do I fix this?

Answer

You probably set `ServerName` in the Apache HTTP Server configuration to the host name, but did not specify the port number.

Instead, set both the host name and port number for `ServerName` in the configuration. For example, if you have Apache HTTP Server configured to listen on port 8080, then set `ServerName` appropriately as in the following excerpt:

```
<VirtualHost *:8080>
ServerName www.localhost.example:8080
```

▼ Cannot rotate logs

Question

My web server and Web Agent are installed as root, and the agent cannot rotate logs. I am seeing this error:

```
Could not rotate log file ... (error: 13)
```

What should I do?

Answer

If the web server is running with a non-root user, for example, the `daemon` user, you must ensure that user has the following permissions:

- Read Permission:
  - `/web_agents/agent_name/lib`
- Read and Write Permission:
  - `/web_agents/agent_name/instances/agent_nnn`
  - `/web_agents/agent_name/log`

Apply execute permissions on the folders listed above, recursively, for the user that runs the web server.

For IIS web agents, change the ownership of the files using the **`agentadmin --o`** command. For more information, refer to agentadmin command.

TIP

You may also see similar issues if SELinux is enabled in `enforcing` mode, and it is not configured to allow access to agent directories.

▼ Protection Against Phishing Attacks

Question

How do I increase security against possible phishing attacks through open redirect?

Answer

You can specify a list of valid URL resources against which AM validates the `goto` and `gotoOnFail` URL using the Valid `goto` URL Resource service.

AM only redirects a user if the `goto` and `gotoOnFail` URL matches any of the resources specified in this setting. If no setting is present, it is assumed that the `goto` and `gotoOnFail` URL is valid.

To set the Valid `goto` URL Resources, use the AM admin UI, and go to **REALMS** > Realm Name > **Services** > **Add** > **Validation Service**, and then add one or more valid `goto` URLs.

You can use the "*" wildcard to define resources, where "*" matches all characters except "?". For example, you can use the wildcards, such as `https://website.example.com/*` or `https://website.example.com/*?*`. For more specific patterns, use resource names with wildcards as described in Configuring success and failure redirection URLs.

▼ Apache Agent Start Up

Question

I have installed the Unix Apache Web Agent, and neither Apache HTTP Server nor the agent start up or log any message. If I remove the agent, the Apache HTTP Server starts again. What can be the problem?

Answer

To troubleshoot Web Agent or a web server that does not start, set the agent logging level to the maximum by performing the following steps:

1. Set the environment variable `AM_SYSTEM_LOG_LEVEL` to `All` in your command line session. For example:

```
$ export AM_SYSTEM_LOG_LEVEL=ALL
```

2. Restart the Apache HTTP Server.
3. Check the logs generated in the `web_agent/apache24_agent/log/system_n.log`.

   Web Agent reserves memory for the policy and session cache based on the `AM_MAX_SESSION_CACHE_SIZE` environment variable. If the server where the agent is installed does not have enough shared memory available, the web agent may log messages like the following:

```
017-11-10 12:06:00.492 +0000   DEBUG [1:7521]
[source/shared.c:1451]am_shm_create2() about to create
block-clusters_0, size 1074008064
```

```
2017-11-10 12:06:00.492 +0000   ERROR
[1:7521]am_shm_create2(): ftruncate failed, error: 28
```

The error message means the web agent tries to reserve 1074008064 bytes of memory, but there is not enough shared memory available. Several reasons may explain why the shared memory is running low, such as:

- A new application or additional workload may be stretching the server resources to the limit.

  In this case, ensure that the server has enough shared memory available to satisfy the need of all the applications.

- A web agent may not have been able to release its shared memory after stopping. Therefore, even if the shared memory is technically not in use, it is still reserved and cannot be reassigned unless freed.

  Different operating systems manage the shared memory in different ways. Refer to your operating system documentation for information about checking shared memory usage.

  You can reduce the amount of memory the web agent reserves for the session and policy cache by setting the `AM_MAX_SESSION_CACHE_SIZE` environment variable to a value between 1048576 (1 MB) and 1074008064 bytes (1 GB). For more information, refer to Environment variables.

  Troubleshooting a component that does not start and does not generate logs may be difficult to diagnose. Contact the ForgeRock Support team for more help and information.

▼ Infinite Redirection Loops

Question

I have client-based (stateless) sessions configured in AM, and I am getting infinite redirection loops. In the `debug.log` file I can see messages similar to the following:

```
... +0000 ERROR [c5319caa-beeb-5a44-a098-d5575e768348]state
identifier not present in authentication state
2018-03-15 16:23:10.538 +0000 WARNING [c5319caa-beeb-5a44-
a098-d5575e768348]unable to verify pre-authentication cookie
... +0000 WARNING [c5319caa-beeb-5a44-a098-
d5575e768348]convert_request_after_authn_post(): unable to
retrieve pre-authentication request data
... +0000 DEBUG [c5319caa-beeb-5a44-a098-d5575e768348] exit
status: forbidden (3), HTTP status: 403, subrequest 0
```

What is happening?

Answer

The redirection loop happens because the client-based (stateless) session cookie is surpassing the maximum supported browser header size. Since the cookie is incomplete, AM cannot validate it.

To ensure the session cookie does not surpass the browser supported size, configure either signing and compression or encryption and compression.

For more information, refer to AM's Security guide.

▼ Errors After Upgrade

Question

I have upgraded my agent and, in the logs, I can see errors similar to the following:

```
redirect_uri_mismatch. The redirection URI provided does not
match a pre-registered value.
com.iplanet.sso.SSOException: Invalid Agent Root URL
com.iplanet.sso.SSOException: Goto URL not valid for the agent
Provider ID
```

What should I do?

Answer

Web Agent accepts only requests sent to the URL specified by the Agent Root URL for CDSSO property. For example, `http://agent.example.com:8080/`.

As a security measure, Web Agent prevents you from accessing the agent on URLs not defined in the Agent Root URL for CDSSO property. Add entries to this property when:

- Accessing the agent through different protocols. For example, `http://agent.example.com/` and `https://agent.example.com/`.
- Accessing the agent through different virtual host names. For example, `http://agent.example.com/` and `http://internal.example.com/`.
- Accessing the agent through different ports. For example, `http://agent.example.com/` and `http://agent.example.com:8080/`.

▼ Configuration Not Updated After Upgrade

Question

I have upgraded my Unix Apache or IBM HTTP Server Web Agent, and even though notifications are enabled, the agent does not update its configuration. What is happening?

Answer

Set the web agent logging level to the maximum by performing the following steps:

1. Set the environment variable `AM_SYSTEM_LOG_LEVEL` to `ALL` in your command line session. For example:

```
$ export AM_SYSTEM_LOG_LEVEL=ALL
```

2. Restart the Apache or IBM HTTP server.
3. Check the logs generated in the `web_agent/agent_type/log/system_n.log` file.

   Sometimes stopping or upgrading an agent does not clean the pipe file the agent uses to communicate with AM. If the newly started agent cannot create the pipe to communicate with AM because it already exists, the agent would log messages like the following:

```
... UTC   DEBUG [1:10551398][source/monitor.c:503]monitor
startup
... UTC   ERROR [102:10551398]monitor unable to get
semaphore
... UTC   DEBUG [304:10551398]
[source/config.c:295]config_initialise():  agent
configuration read from cache, agent: / wpa-aix7-Httpd7-
32bit
```

If you see similar error messages, perform the following steps to delete the pipe file:

1. Stop the Apache or IBM HTTP server.
2. Change directories to the `/tmp` directory.
3. Delete the `monitor.pipe` file.
4. Restart the Apache or IBM HTTP server.

▼ Custom Pages Not Displayed

Question

After upgrade, the default Apache welcome page appears instead of my custom error pages. What should I do?

Answer

Check your Apache `ErrorDocument` configuration. If the custom error pages are not in the document root of the Apache HTTP Server, enclose the `ErrorDocument` directives in `Directory` elements. For example:

```
<Directory "/web/docs">
   ErrorDocument 403 myCustom403Page.html
</Directory>
```

For more information about `ErrorDocument`, refer to the Apache documentation.

Failure After Installation

Question

After starting a web agent installation, I see a failure in the logs:

```
[../resources/troubleshooting/troubleshooting.bash:#web-agent-
install]
```

Answer

Web Agent installation, can fail if AM's validation of the agent configuration exceeds the default timeout of 4 seconds.

You can set the `AM_NET_TIMEOUT` environment variable to change the default timeout, and then rerun the installation.

Agent Not Protecting a Website

Question

My Web Agent is not protecting my website. In the logs, I can see errors similar to the following:

```
... -0500  ERROR [86169084-5648-6f4d-a706-
30f5343d9220]config_fetch():  failed to load configuration for
agent: myagent myagent, error -24
... -0500  ERROR [86169084-5648-6f4d-a706-
30f5343d9220]amagent_auth_handler(): failed to get agent
configuration instance, error: invalid agent session*
```

What is happening?

Answer

The Web Agent is unable to log in to AM. Possible causes are:

- Network connection between the agent and AM is unavailable.
- The AM Connection URL property, which specifies the AM URL may be misconfigured.

Agent Not Protecting a Website

Question

My Web Agent is not protecting my website. In the `debug.log` file I can see messages similar to the following:

```
... GMT DEBUG [162ba6eb-cf88-3d7f-f92c-ee8b21971b4c]:
(source/oidc.c:265) agent_realm does not have the expected
value: JWT
```

```
   {
   "sub":"demo",
   "auditTrackingId":"267d1f56-0b97-4830-ae91-6be4b8b7099f-
5840",
   "iss":"https://am.example.com:8443/am/oauth2/Customers",
   "tokenName":"id_token",
   "nonce":"D3AE96656D6D634489AF325D90C435A2",
   "aud":"webagent",
   "s_hash":"rxwxIoqDFiwt4MxSwiBa-w",
   "azp":"webagent",
   "auth_time":1561600459,
   "forgerock":{
   "ssotoken":"wi8tHql...MQAA*",
   "suid":"267d1f56-0b97-4830-ae91-6be4b8b7099f-5647"
   },
   "realm":"/Customers",
   "exp":1561607661,
   "tokenType":"JWTToken",
   "iat":1561600461,
   "agent_realm":"/Customers"
   }
... GMT WARNING [162ba6eb-cf88-3d7f-f92c-ee8b21971b4c]:
redirect_after_authn(): unable to validate JWT
```

What is happening?

Answer

If you configured the agent profile in a realm other than AM's top-level realm (/), you must configure the agent `com.sun.identity.agents.config.organization.name` bootstrap property with the realm where the agent profile is located. For example, `/Customers` .

Realm names are case-sensitive. Failure to set the realm name exactly as configured in AM causes the agent to fail to recognize the realm.

▼ Cannot Access Protected Resources

I am getting HTTP 403 Forbidden messages when accessing protected resources, and I can see errors similar to the following in the `debug.log` file:

```
... GMT WARNING [69d4632c-82af-b853-0f340vb7b754]: too many
pending authentications
... GMT ERROR  [69d4632c-82af-76da-b853-0f340vb7b754]:
save_pre_authn_state(): unable to save state for request
```

What is happening?

Answer

Agents store the progress of authentication with AM in the pre-authentication cookie, `agent-authn-tx`. This cookie has a maximum size of 4096 bytes, and can fill up if the agent receives many parallel unauthenticated requests to access protected resources.

For more information, refer to Enable Multivalue for Pre-Authn Cookie.

▼ Cannot Access Protected Resources

Question

I am getting HTTP 403 Forbidden messages when accessing the Web Agent.

Answer

Make sure the Web Agent is executable:

1. In the terminal where the Web Agent is running, go to `/opt/web_agents`.
2. Review and, if necessary, change the permissions for the directory:

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 group

A group of agent instances that share runtime resources and shared memory.

Agent profile

A set of configuration properties that define the behavior of the agent.

Agent profile realm
: The AM realm in which the agent profile is stored.

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.

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.

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.

Centralized configuration mode

AM stores the agent properties in the AM configuration store. See also local configuration mode.

The configuration mode is defined by Location of Agent Configuration Repository.

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 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 AM.

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.

Identity store
: Data storage service holding principals' profiles; underlying storage can be an LDAP directory service or a custom `IdRepo` implementation.

Java agent
: Java web application installed in a java container that acts as a policy enforcement point, filtering requests to other applications in the container, with policies based on application resource URLs.

Local configuration mode
: The Web Agent installer creates the file `/web_agents/agent_type/instances/agent_nnn/config/agent.conf` to store the agent configuration properties. See also centralized configuration mode.

  The configuration mode is defined by Location of Agent Configuration Repository.

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
: 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 policy decision point.

Policy evaluation realm
The AM realm that the agent uses to request policy decisions from AM.

Policy Information Point (PIP)
Entity that provides extra information, such as user profile attributes, that a policy decision point 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 an AM identity successfully authenticates, AM associates the identity 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).

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 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.

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.

User realm
: The AM realm in which a user is authenticated.

Web Agent
: Native library installed in a web server that acts as a policy enforcement point with policies based on web page URLs.