

# Maintenance guide

---

This guide describes how to perform recurring administrative operations in Java Agent.

## Audit the deployment

---

Auditing is managed by the Commons Audit Framework to log audit events for security, troubleshooting, and regulatory compliance. Audit logs are written in UTF-8 format.

### Remote and local auditing

#### *Remote auditing*

In remote auditing, the agent logs audit events to the audit event handler configured in the AM realm. In an environment with several AM servers, the agent writes audit logs to the AM server that satisfies the agent request for client authentication or resource authorization.

The agent logs audit events remotely **only** when AM's global audit logging is enabled and configured in the realm where the agent runs.

Set up global audit logging in the AM admin UI:

1. In the AM admin UI, go to **Configure > Global Services > Audit logging**.
2. Enable **Audit logging**.
3. Enter values to include in **Field whitelist filters** or **Field blacklist filters**.

The following example path in the **Field whitelist filters** list includes the Accept-Language value in the **http.request.headers** field in *access* events:

```
/access/http/request/headers/accept-language
```

Learn more from AM's [Global audit logging](#).

#### *Local auditing*

In local auditing, the agent logs audit events in JSON format to a file in the agent installation directory, `/path/to/java_agents/agent_type/Agent_n/logs/audit`.

## Remote and local auditing

In remote and local auditing, the agent logs audit events in the following locations:

- To a file in the agent installation directory.
- To the audit event handler configured in the AM realm in which the agent profile is configured.

## Audit event logs

When a request matches a not-enforced rule, the agent doesn't log an audit event for that request. Otherwise, the agent logs an audit event according to the configuration described in [Configure auditing](#) and [Include or exclude elements from the audit logs](#).

The following example shows an audit event log for successful access to a resource:

```
{
  "_id": "...", //This ID is internal to AM and available only in
  remote logs.
  "timestamp": "...",
  "eventName": "AM-ACCESS-OUTCOME",
  "transactionId": "...",
  "trackingIds": [
    "...",
    "..."
  ],
  "userId": "id=demo,ou=user,dc=example,dc=com",
  "client": {
    "ip": "...",
    "port": ...
  },
  "server": {
    "ip": "...",
    "port": 8020
  },
  "http": {
    "request": {
      "secure": false,
      "method": "GET",
      "path": "http://my.example.com:8020/examples/",
      "headers": {
        "accept": [
```

```

"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
image/webp,image/apng,*/*;q=0.8,application/signed-
exchange;v=b3;q=0.7"
  ],
  "host": [
    "my.example.com:8020"
  ],
  "user-agent": [
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0
Safari/537.36"
  ]
}
},
"request": {
  "protocol": "HTTP/1.1",
  "operation": "GET"
},
"response": {
  "status": "SUCCESSFUL",
  "statusCode": "302",
  "elapsedTime": 25,
  "elapsedTimeUnits": "MILLISECONDS"
},
"component": "Java Policy Agent"
}

```

The audit log format uses the log structure shared by the Ping Identity Platform. Learn more from [Audit log format](#) in AM's *Security guide*.

Java Agent supports propagation of the transaction ID across the Ping Identity Platform, using the HTTP header `X-ForgeRock-TransactionId`. Learn more from [Trust transaction headers](#) in AM's *Security guide*.

## Configure auditing

In the AM admin UI, access the audit configuration at **Realms** > *Realm Name* > **Applications** > **Agents** > **Java** > *Agent Name* > **Global** > **Audit**. Use the following properties to configure auditing:

- [Audit Access Types](#): Select the type of messages to log. For example, select `LOG_ALL` to log access allowed and access denied events.

- Audit Log Location: Select whether to write the audit logs locally to the agent installation ( LOCAL ), remotely to AM ( REMOTE ), or to both places ( ALL ). For example, keep REMOTE to log audit events to the AM instances.
- Enable Local Audit Log Rotation: Select whether to rotate the audit logs when they reach a maximum size.
- Local Audit Log Rotation Size: Specify the maximum size of logs before rotation.

## Include or exclude elements from the audit logs

### IMPORTANT

Before you include non-safelisted audit event fields in the logs, consider the impact on security. Inclusion of some headers, query parameters, or cookies could cause credentials or tokens to be logged, and allow anyone with access to the logs to impersonate the holder of these credentials or tokens.

To prevent logging of sensitive data for an audit event, the Common Audit Framework uses a safelist to specify which audit event fields appear in the logs. By default, only safelisted audit event fields are included in the logs.

Use Audit Log Include Paths and Audit Log Exclude Paths to include or exclude elements from the audit logs.

Audit Log Exclude Paths takes precedence over Audit Log Include Paths. If a path is specified by both properties, the corresponding audit event field is excluded.

The following example excludes Header1 but includes Header2 and Cookie1:

```
org.forgerock.agents.audit.exclude.path.list[0]=/access/http/request/headers/Header1Name
org.forgerock.agents.audit.include.path.list[0]=/access/http/request/headers/Header2Name
org.forgerock.agents.audit.include.path.list[1]=/access/http/request/cookies/Cookie1Name
```

## Monitor services

The following sections describe how to set up and maintain monitoring in your deployment to ensure appropriate performance and service availability.

### NOTE

The Prometheus endpoint isn't currently supported for Jakarta builds of the agent because of known issue AMAGENTS-6809.

## Monitor with Prometheus

Java Agent automatically exposes a monitoring endpoint where Prometheus can scrape metrics in a standard Prometheus format (version 0.0.4).

The Prometheus endpoint is enabled by default. You can disable it if required by setting the [Enable Prometheus Monitoring](#) property to `false`.

You can find information about installing and running Prometheus in the [Prometheus documentation](#).

Prometheus performance metrics are provided by an endpoint configured in the protected web application's `web.xml` file. The endpoint must be accessible to the Prometheus server that uses the performance data.

### TIP

Tools such as Grafana are available to create customized charts and graphs based on the information collected by Prometheus. Learn more on the [Grafana website](#).

### *Expose the Prometheus endpoint*

Use the following procedure to expose the Prometheus endpoint.

1. For each protected web application that is to expose metrics, edit the `web.xml` file for the `agentapp` application.

The following example exposes a base endpoint named `/metrics`:

```
<servlet>
  <servlet-name>AgentMonitoring</servlet-name>
  <servlet-
class>org.forgerock.http.servlet.HttpFrameworkServlet</servlet
-class>
  <init-param>
    <param-name>application-loader</param-name>
    <param-value>service_loader</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>AgentMonitoring</servlet-name>
  <url-pattern>/metrics/*</url-pattern>
</servlet-mapping>
```

Choose a name for the exposed base endpoint that does not conflict with any of the built-in agent endpoints.

2. Allow access to the Prometheus endpoint that is protected by the agent, in one of the following ways:

- Configure [Not-Enforced URIs](#) to create a not-enforced URI rule for the base endpoint.

The following example rule allows access to the metrics base endpoint:

```
*/metrics/*
```

- Configure [Not-Enforced URIs](#), [Not-Enforced Client IP List](#), and [Not-Enforced Compound Rule Separator](#) to create a compound not-enforced rule for the base endpoint.

The rule allows access from only the IP addresses of the Prometheus server.

The following example rule allows access to the `/metrics` endpoint for HTTP requests that come from the IP address range from 192.168.1.1 to 192.168.1.3:

```
192.168.1.1-192.168.1.3 | */metrics/*
```

HTTP requests from other IP addresses are not able to access the metrics base endpoint.

- Create an authorization policy in Advanced Identity Cloud or AM to restrict access to the metrics base endpoint.

The metric base endpoint doesn't require login credentials. You can use a policy to ensure that requests to the endpoint are authenticated against the Advanced Identity Cloud or AM instance.

Learn more in [Policies](#) in Advanced Identity Cloud's *Authorization guide* or [Policies](#) in AM's *Authorization guide*.

3. If the Prometheus endpoint is protected by Advanced Identity Cloud or AM policies, include the required credentials.

### *Access the Prometheus endpoint*

When the example in [Expose the Prometheus endpoint](#) is configured, the Prometheus endpoint is available at

```
https://mydomain.example.com/agentapp/metrics/prometheus.
```

1. Access the Prometheus endpoint as follows:

```
$ curl https://mydomain.example.com/agentapp/metrics/prometheus
```

The metrics are displayed:

```
# HELP ja_expired_session_cache_eviction the eviction count
for the expired session cache
# TYPE ja_expired_session_cache_eviction gauge
ja_expired_session_cache_eviction 0.0
# HELP ja_expired_session_cache the hit count for the expired
session cache
# TYPE ja_expired_session_cache gauge
ja_expired_session_cache{outcome="hit",} 0.0
ja_expired_session_cache{outcome="miss",} 0.0
# HELP ja_expired_session_cache_load the total load count for
the expired session cache
# TYPE ja_expired_session_cache_load gauge
...
```

## Write metrics to CSV files

Configure [Export Monitoring Metrics to CSV](#) to write metric information to CSV files.

## Monitoring types

This section describes the data types used in monitoring:

### **Counter**

Cumulative metric for a numerical value that only increases.

### **Distinct counter**

Metric that provides an estimate of the number of *unique* values recorded.

For example, use this to estimate the number of unique users who have authenticated, or unique client IP addresses.

#### NOTE

The `DistinctCounter` type is reported as a `Gauge` type.

The `DistinctCounter` metric is calculated per instance of AM and can't be aggregated across multiple instances to get a site-wide view.

### **Gauge**

Metric for a numerical value that can increase or decrease.

The value for a gauge is calculated when requested and represents the state of the metric at that specific time.

### **Timer**

Metric that provides duration information.

The Prometheus endpoint doesn't provide rate-based statistics because rates can be calculated from the time-series data. Instead, the Prometheus endpoint includes summary metrics whose names have the following suffixes or labels:

- `count` : number of events recorded
- `sum` : sum of the durations recorded
- `{quantile="0.5"}` : 50% of the durations are at or below this value
- `{quantile="0.75"}` : 75% of the durations are at or below this value
- `{quantile="0.95"}` : 95% of the durations are at or below this value
- `{quantile="0.98"}` : 98% of the durations are at or below this value
- `{quantile="0.99"}` : 99% of the durations are at or below this value
- `{quantile="0.999"}` : 99.9% of the durations are at or below this value

#### NOTE

The `Timer` metric type is reported as a `Summary` type.

Duration-based quantile values are weighted towards newer data. By representing approximately the last five minutes of data, the timers make it easier to see recent changes in behavior, rather than a uniform average of recordings since the server was started.

## Metrics at the Prometheus endpoint

### *Audit handler metrics*

Java Agent exposes the following audit handler monitoring metrics:

Metric	Type	Description
<code>ja_audit_generate{topic=access}</code>	Timer	Time taken to generate an audit object.
<code>ja_audit{handler-type=<i>handler-type</i>, name=default, topic=access, outcome=<i>outcome</i>}</code>	Timer	Time taken to audit outcomes, both locally to the agent and remotely in Advanced Identity Cloud or AM.

### *Audit handler metrics labels*

Label	Values
-------	--------

Label	Values
<i>handler-type</i>	<ul style="list-style-type: none"> <li>am-delegate . Remote auditing performed by Advanced Identity Cloud or AM.  (Prometheus: am_delegate )</li> <li>json . Local audit logging using JSON.</li> </ul>
<i>outcome</i>	<ul style="list-style-type: none"> <li>success</li> <li>failure</li> </ul>

### *Endpoint and REST SDK metrics*

Java Agent exposes the following endpoint and REST SDK monitoring metrics:

Metric	Type	Description
ja_session_info	Timer	Time taken to retrieve user session information from Advanced Identity Cloud or AM.
ja_user_profile	Timer	Time taken to retrieve the user profile information from Advanced Identity Cloud or AM.
ja_policy_decision	Timer	Time taken to retrieve policy decisions from Advanced Identity Cloud or AM.

### *Expired session metrics*

Java Agent exposes the following expired session monitoring metrics:

Metric	Type	Description
ja_expired_session_cache_size	Gauge	Total number of entries in the expired session cache.
ja_expired_session_cache_eviction	Gauge	Eviction count for the expired session cache.
ja_expired_session_cache_load	Gauge	Load count for the expired session cache.

Metric	Type	Description
ja_expired_session_cache_load_time	Gauge	Load time for the expired session cache, in milliseconds.
ja_expired_session_cache{outcome=hit}	Gauge	Hit count for the expired session cache.
ja_expired_session_cache{outcome=miss}	Gauge	Miss count for the expired session cache.

### JSON Web Token (JWT) metrics

Java Agent exposes the following JWT-related monitoring metrics:

Metric	Type	Description
ja_jwt_cache_size	Gauge	Size of the JWT cache.
ja_jwt_cache_eviction	Gauge	The eviction count for the JWT cache.
ja_jwt_cache_load	Gauge	The load count for the JWT cache.
ja_jwt_cache_load_time	Gauge	The load time for the JWT cache, in milliseconds.
ja_jwt_cache{outcome=hit}	Gauge	The hit count for the JWT cache.
ja_jwt_cache{outcome=miss}	Gauge	The miss count for the JWT cache.

### JVM metrics

Java Agent exposes the following JVM-related monitoring metrics:

#### NOTE

Some `ja_jvm_*` metrics depend on the JVM version and configuration. In particular, GC-related metrics depend on the garbage collector the server uses. The GC metric names are *unstable*, and can change even in a minor JVM release.

Metric	Type	Description
ja_jvm_available_cpus	Gauge	Number of processors available to the Java virtual machine.

Metric	Type	Description
ja_jvm_class_loading_loaded	Gauge	Number of classes loaded since the Java virtual machine started.
ja_jvm_class_loading_unloaded	Gauge	Number of classes unloaded since the Java virtual machine started.
ja_jvm_free_used_memory_bytes	Gauge	Bytes of free memory allocated to the Java virtual machine
ja_jvm_garbage_collector_g1_old_generation_collection	Gauge	Number of collections performed by the "G1 old generation" garbage collection algorithm.
ja_jvm_garbage_collector_g1_old_generation_time	Gauge	Approximate accumulated time taken by the "G1 old generation" garbage collection algorithm.
ja_jvm_garbage_collector_g1_young_generation_collection	Gauge	Number of collections performed by the "G1 young generation" garbage collection algorithm.
ja_jvm_garbage_collector_g1_young_generation_time	Gauge	Approximate accumulated time taken by the "G1 young generation" garbage collection algorithm.
ja_jvm_max_memory_bytes	Gauge	Maximum amount of memory the Java virtual machine attempts to use.
ja_jvm_memory_usage_heap_init	Gauge	Amount of heap memory the Java virtual machine initially requested from the operating system.
ja_jvm_memory_usage_heap_max	Gauge	Maximum amount of heap memory the Java virtual machine attempts to use.
ja_jvm_memory_usage_heap_committed	Gauge	Amount of heap memory committed for the Java virtual machine to use.
ja_jvm_memory_usage_heap_used	Gauge	Amount of heap memory used by the Java virtual machine.

Metric	Type	Description
ja_jvm_memory_usage_heap_usage	Gauge	Amount of heap memory allocated to the Java virtual machine.
ja_jvm_memory_usage_total_init	Gauge	Amount of memory the Java virtual machine initially requested from the operating system.
ja_jvm_memory_usage_total_max	Gauge	Maximum amount of memory the Java virtual machine attempts to use.
ja_jvm_memory_usage_non_heap_init	Gauge	Amount of non-heap memory the Java virtual machine initially requested from the operating system.
ja_jvm_memory_usage_non_heap_max	Gauge	Maximum amount of non-heap memory the Java virtual machine attempts to use.
ja_jvm_memory_usage_non_heap_committed	Gauge	Amount of non-heap memory committed for the Java virtual machine to use.
ja_jvm_memory_usage_non_heap_used	Gauge	Amount of non-heap memory used by the Java virtual machine.
ja_jvm_memory_usage_non_heap_usage	Gauge	Amount of non-heap memory allocated to the Java virtual machine.
ja_jvm_memory_usage_pools_codeheap_non_nmethods_committed	Gauge	Amount of "non-method code cache heap" memory committed for the Java virtual machine to use.
ja_jvm_memory_usage_pools_codeheap_non_nmethods_init	Gauge	Amount of "non-method code cache heap" memory the Java virtual machine initially requested from the operating system.
ja_jvm_memory_usage_pools_codeheap_non_nmethods_max	Gauge	Maximum amount of "non-method code cache heap" memory the Java virtual machine attempts to use.

Metric	Type	Description
ja_jvm_memory_usage_pools_codeheap_non_nmethods_usage	Gauge	Amount of "non-method code cache heap" memory allocated to the Java virtual machine.
ja_jvm_memory_usage_pools_codeheap_non_nmethods_used	Gauge	Amount of "non-method code cache heap" memory used by the Java virtual machine.
ja_jvm_memory_usage_pools_codeheap_non_profiled_nmethods_committed	Gauge	Amount of "non-profiled code cache heap" memory committed for the Java virtual machine to use.
ja_jvm_memory_usage_pools_codeheap_non_profiled_nmethods_init	Gauge	Amount of "non-profiled code cache heap" memory the Java virtual machine initially requested from the operating system.
ja_jvm_memory_usage_pools_codeheap_non_profiled_nmethods_max	Gauge	Maximum amount of "non-profiled code cache heap" memory the Java virtual machine attempts to use.
ja_jvm_memory_usage_pools_codeheap_non_profiled_nmethods_usage	Gauge	Amount of "non-profiled code cache heap" memory allocated to the Java virtual machine.
ja_jvm_memory_usage_pools_codeheap_non_profiled_nmethods_used	Gauge	Amount of "non-profiled code cache heap" memory used by the Java virtual machine.
ja_jvm_memory_usage_pools_codeheap_profiled_nmethods_committed	Gauge	Amount of "profiled code cache heap" memory committed for the Java virtual machine to use.
ja_jvm_memory_usage_pools_codeheap_profiled_nmethods_init	Gauge	Amount of "profiled code cache heap" memory the Java virtual machine initially requested from the operating system.
ja_jvm_memory_usage_pools_codeheap_profiled_nmethods_max	Gauge	Maximum amount of "profiled code cache heap" memory the Java virtual machine attempts to use.
ja_jvm_memory_usage_pools_codeheap_profiled_nmethods_usage	Gauge	Amount of "profiled code cache heap" memory allocated to the Java virtual machine.

Metric	Type	Description
ja_jvm_memory_usage_pools_codeheap_profiled_nmethods_used	Gauge	Amount of "profiled code cache heap" memory used by the Java virtual machine.
ja_jvm_memory_usage_pools_compressed_class_space_init	Gauge	Amount of "compressed class space" memory the Java virtual machine initially requested from the operating system.
ja_jvm_memory_usage_pools_compressed_class_space_max	Gauge	Maximum amount of "compressed class space" memory the Java virtual machine attempts to use.
ja_jvm_memory_usage_pools_compressed_class_space_committed	Gauge	Amount of "compressed class space" memory committed for the Java virtual machine to use.
ja_jvm_memory_usage_pools_compressed_class_space_used	Gauge	Amount of "compressed class space" memory used by the Java virtual machine.
ja_jvm_memory_usage_pools_compressed_class_space_usage	Gauge	Amount of "compressed class space" memory allocated to the Java virtual machine.
ja_jvm_memory_usage_pools_g1_eden_space_committed	Gauge	Amount of "G1 eden space" memory committed for the Java virtual machine to use.
ja_jvm_memory_usage_pools_g1_eden_space_init	Gauge	Amount of "G1 eden space" memory the Java virtual machine initially requested from the operating system.
ja_jvm_memory_usage_pools_g1_eden_space_max	Gauge	Maximum amount of "G1 eden space" memory the Java virtual machine attempts to use.
ja_jvm_memory_usage_pools_g1_eden_space_usage	Gauge	Amount of "G1 eden space" memory allocated to the Java virtual machine.
ja_jvm_memory_usage_pools_g1_eden_space_used	Gauge	Amount of "G1 eden space" memory used by the Java virtual machine.

Metric	Type	Description
ja_jvm_memory_usage_pools_g1_eden_space_used_after_gc	Gauge	Amount of "G1 eden space" memory used by the Java virtual machine after garbage collection.
ja_jvm_memory_usage_pools_g1_old_gen_committed	Gauge	Amount of "G1 old generation" memory committed for the Java virtual machine to use.
ja_jvm_memory_usage_pools_g1_old_gen_init	Gauge	Amount of "G1 old generation" memory the Java virtual machine initially requested from the operating system.
ja_jvm_memory_usage_pools_g1_old_gen_max	Gauge	Maximum amount of "G1 old generation" memory the Java virtual machine attempts to use.
ja_jvm_memory_usage_pools_g1_old_gen_usage	Gauge	Amount of "G1 old generation" memory allocated to the Java virtual machine.
ja_jvm_memory_usage_pools_g1_old_gen_used	Gauge	Amount of "G1 old generation" memory used by the Java virtual machine.
ja_jvm_memory_usage_pools_g1_old_gen_used_after_gc	Gauge	Amount of "G1 old generation" memory used by the Java virtual machine after garbage collection.
ja_jvm_memory_usage_pools_g1_survivor_space_committed	Gauge	Amount of "G1 survivor space" memory committed for the Java virtual machine to use.
ja_jvm_memory_usage_pools_g1_survivor_space_init	Gauge	Amount of "G1 survivor space" memory the Java virtual machine initially requested from the operating system.
ja_jvm_memory_usage_pools_g1_survivor_space_max	Gauge	Maximum amount of "G1 survivor space" memory the Java virtual machine attempts to use.
ja_jvm_memory_usage_pools_g1_survivor_space_usage	Gauge	Amount of "G1 survivor space" memory allocated to the Java virtual machine.

Metric	Type	Description
ja_jvm_memory_usage_pools_g1_survivor_space_used	Gauge	Amount of "G1 survivor space" memory used by the Java virtual machine.
ja_jvm_memory_usage_pools_g1_survivor_space_used_after_gc	Gauge	Amount of "G1 survivor space" memory used by the Java virtual machine after garbage collection.
ja_jvm_memory_usage_pools_metaspace_init	Gauge	Amount of "metaspace" memory the Java virtual machine initially requested from the operating system.
ja_jvm_memory_usage_pools_metaspace_max	Gauge	Maximum amount of "metaspace" memory the Java virtual machine attempts to use.
ja_jvm_memory_usage_pools_metaspace_committed	Gauge	Amount of "metaspace" memory committed for the Java virtual machine to use.
ja_jvm_memory_usage_pools_metaspace_used	Gauge	Amount of "metaspace" memory used by the Java virtual machine.
ja_jvm_memory_usage_pools_metaspace_usage	Gauge	Amount of "metaspace" memory allocated to the Java virtual machine.
ja_jvm_memory_usage_total_committed	Gauge	Amount of memory committed for the Java virtual machine to use.
ja_jvm_memory_usage_total_used	Gauge	Amount of memory used by the Java virtual machine.
ja_jvm_thread_state_blocked_result	Gauge	Number of threads in the BLOCKED state.
ja_jvm_thread_state_result	Gauge	Number of live threads including both daemon and non-daemon threads.
ja_jvm_thread_state_daemon_result	Gauge	Number of live daemon threads.
ja_jvm_thread_state_new_result	Gauge	Number of threads in the NEW state.

Metric	Type	Description
ja_jvm_thread_state_runnable_result	Gauge	Number of threads in the RUNNABLE state.
ja_jvm_thread_state_terminated_result	Gauge	Number of threads in the TERMINATED state.
ja_jvm_thread_state_timed_waiting_result	Gauge	Number of threads in the TIMED_WAITING state.
ja_jvm_thread_state_waiting_result	Gauge	Number of threads in the WAITING state.
ja_jvm_used_memory_bytes	Gauge	Amount of memory used by the Java virtual machine

### *Not-enforced rule metrics*

Java Agent exposes the following not-enforced rule monitoring metrics:

Metric	Type	Description
ja_notenforced_uri_matched_cache_size	Gauge	Size of the not-enforced URI matched cache.
ja_notenforced_uri_matched_cache_eviction	Gauge	Eviction count for the not-enforced URI matched cache.
ja_notenforced_uri_matched_cache_load	Gauge	Load count for the not-enforced URI matched cache.
ja_notenforced_uri_matched_cache_load_time	Gauge	Load time for the not-enforced URI matched cache, in milliseconds.
ja_notenforced_uri_matched_cache{outcome=hit}	Gauge	Hit count for the not-enforced URI matched cache.
ja_notenforced_uri_matched_cache{outcome=miss}	Gauge	Miss count for the not-enforced URI matched cache.
ja_notenforced_uri_unmatched_cache_size	Gauge	Size of the not-enforced URI unmatched cache.
ja_notenforced_uri_unmatched_cache_eviction	Gauge	Eviction count for the not-enforced URI unmatched cache.

Metric	Type	Description
ja_notenforced_uri_unmatched_cache_load	Gauge	Load count for the not-enforced URI unmatched cache.
ja_notenforced_uri_unmatched_cache_load_time	Gauge	Load time for the not-enforced URI unmatched cache, in milliseconds.
ja_notenforced_uri_unmatched_cache{outcome=hit}	Gauge	Hit count for the not-enforced URI unmatched cache.
ja_notenforced_uri_unmatched_cache{outcome=miss}	Gauge	Miss count for the not-enforced URI unmatched cache.
ja_notenforced_ip_matched_cache_size	Gauge	Size of the not-enforced IP matched cache.
ja_notenforced_ip_matched_cache_eviction	Gauge	Eviction count for the not-enforced IP matched cache.
ja_notenforced_ip_matched_cache_load	Gauge	Load count for the not-enforced IP matched cache.
ja_notenforced_ip_matched_cache_load_time	Gauge	Load time for the not-enforced IP matched cache, in milliseconds.
ja_notenforced_ip_matched_cache{outcome=hit}	Gauge	Hit count for the not-enforced IP matched cache.
ja_notenforced_ip_matched_cache{outcome=miss}	Gauge	Miss count for the not-enforced IP matched cache.
ja_notenforced_ip_unmatched_cache_size	Gauge	Size of the not-enforced IP unmatched cache.
ja_notenforced_ip_unmatched_cache_eviction	Gauge	Eviction count for the not-enforced IP unmatched cache.
ja_notenforced_ip_unmatched_cache_load	Gauge	Load count for the not-enforced IP unmatched cache.
ja_notenforced_ip_unmatched_cache_load_time	Gauge	Load time for the not-enforced IP unmatched cache, in milliseconds.
ja_notenforced_ip_unmatched_cache{outcome=hit}	Gauge	Hit count for the not-enforced IP unmatched cache.

Metric	Type	Description
ja_notenforced_ip_unmatched_cache{outcome=miss}	Gauge	Miss count for the not-enforced IP unmatched cache.
ja_notenforced_compound_matched_cache_size	Gauge	Size of the not-enforced compound matched cache.
ja_notenforced_compound_matched_cache_eviction	Gauge	Eviction count for the not-enforced compound matched cache.
ja_notenforced_compound_matched_cache_load	Gauge	Load count for the not-enforced compound matched cache.
ja_notenforced_compound_matched_cache_load_time	Gauge	Load time for the not-enforced compound matched cache, in milliseconds.
ja_notenforced_compound_matched_cache{outcome=hit}	Gauge	Hit count for the not-enforced compound matched cache.
ja_notenforced_compound_matched_cache{outcome=miss}	Gauge	Miss count for the not-enforced compound matched cache.
ja_notenforced_compound_unmatched_cache_size	Gauge	Size of the not-enforced compound unmatched cache.
ja_notenforced_compound_unmatched_cache_eviction	Gauge	Eviction count for the not-enforced compound unmatched cache.
ja_notenforced_compound_unmatched_cache_load	Gauge	Load count for the not-enforced compound unmatched cache.
ja_notenforced_compound_unmatched_cache_load_time	Gauge	Load time for the not-enforced compound unmatched cache, in milliseconds.
ja_notenforced_compound_unmatched_cache{outcome=hit}	Gauge	Hit count for the not-enforced compound unmatched cache.
ja_notenforced_compound_unmatched_cache{outcome=miss}	Gauge	Miss count for the not-enforced compound unmatched cache.

### *Policy decision metrics*

Java Agent exposes the following policy decision monitoring metrics:

Metric	Type	Description
ja_policy_decision_cache_size	Gauge	Size of the policy decision cache.
ja_policy_decision_cache_eviction	Gauge	Eviction count for the policy decision cache.
ja_policy_decision_cache_load	Gauge	Load count for the policy decision cache.
ja_policy_decision_cache_load_time	Gauge	Load time for the policy decision cache, in milliseconds.
ja_policy_decision_cache{outcome=hit}	Gauge	Hit count for the policy decision cache.
ja_policy_decision_cache{outcome=miss}	Gauge	Miss count for the policy decision cache.

### *POST data preservation metrics*

Java Agent exposes the following POST data preservation monitoring metrics:

Metric	Type	Description
ja_pdp_cache_size	Gauge	Size of the POST data preservation cache.
ja_pdp_cache_eviction	Gauge	Eviction count for the POST data preservation cache.
ja_pdp_cache_load	Gauge	Load count for the POST data preservation cache.
ja_pdp_cache_load_time	Gauge	Load time for the POST data preservation cache, in milliseconds.
ja_pdp_cache{outcome=hit}	Gauge	Hit count for the POST data preservation cache.
ja_pdp_cache{outcome=miss}	Gauge	Miss count for the POST data preservation cache.

### *Request metrics*

Java Agent exposes the following request monitoring metrics:

Metric	Type	Description
<code>ja_requests{access=<i>access</i>, decision=<i>decision</i>}</code>	Timer	Rate of granted/denied requests and their decision.

### *Request metrics labels*

Label	Values
<i>access</i>	<ul style="list-style-type: none"> <li>granted</li> <li>denied</li> </ul>
<i>decision</i>	<ul style="list-style-type: none"> <li>not-enforced : Request matched a not-enforced rule.</li> <li>no-valid-token : Request didn't have a valid SSO token or an OpenID Connect JWT.</li> <li>allowed-by-policy : Request matched a policy that allowed access.</li> <li>denied-by-policy : Request matched a policy that denied access.</li> <li>am-unavailable : The Advanced Identity Cloud or AM instance wasn't reachable.</li> <li>bad-request : The request URI was badly formed.</li> <li>unexpected-exception : An unexpected exception occurred within the agent.</li> </ul>

### *Session information metrics*

Java Agent exposes the following session information monitoring metrics:

Metric	Type	Description
<code>ja_session_info_cache_size</code>	Gauge	Size of the session information cache.
<code>ja_session_info_cache_eviction</code>	Gauge	Eviction count for the session information cache.
<code>ja_session_info_cache_load</code>	Gauge	Load count for the session information cache.

Metric	Type	Description
ja_session_info_cache_load_time	Gauge	Load time for the session information cache, in milliseconds.
ja_session_info_cache{outcome=hit}	Gauge	Hit count for the session information cache.
ja_session_info_cache{outcome=miss}	Gauge	Miss count for the session information cache.

### *SSO token to JWT exchange metrics*

Java Agent exposes the following SSO token to JWT exchange monitoring metrics:

Metric	Type	Description
ja_sso_exchange_cache_size	Gauge	Size of the SSO token exchange cache.
ja_sso_exchange_cache_eviction	Gauge	Eviction count for the SSO token exchange cache.
ja_sso_exchange_cache_load	Gauge	Load count for the SSO token exchange cache.
ja_sso_exchange_cache_load_time	Gauge	Load time for the SSO token exchange, in milliseconds.
ja_sso_exchange_cache{outcome=hit}	Gauge	Hit count for the SSO token exchange cache.
ja_sso_exchange_cache{outcome=miss}	Gauge	Miss count for the SSO token exchange cache.

### *WebSocket metrics*

Java Agent exposes the following WebSocket-related monitoring metrics:

Metric	Type	Description
ja_websocket_last_received	Gauge	Number of milliseconds since anything was received over the WebSocket, for example, a ping or a notification.

Metric	Type	Description
ja_websocket_last_sent	Gauge	Number of milliseconds since anything was sent over the WebSocket.
ja_websocket_config_change_received_total	Counter	Count of WebSocket configuration change notifications received. <sup>(1)</sup>
ja_websocket_config_change_processed_total		Count of WebSocket configuration change notifications processed.
ja_websocket_policy_change_received_total	Counter	Count of WebSocket policy change notifications received. <sup>(1)</sup>
ja_websocket_policy_change_processed_total	Counter	Count of WebSocket policy change notifications processed.
ja_websocket_session_logout_received_total	Counter	Count of WebSocket session logout notifications received. <sup>(1)</sup>
ja_websocket_session_logout_processed_total	Counter	Count of WebSocket session logout notifications processed.
ja_websocket_ping_pong	Timer	Ping/pong round trip time.

<sup>(1)</sup> Some can be ignored if the realm or agent name isn't applicable.

## Manage logs

Logs in Java Agent and third-party dependencies are recorded using the Logback implementation of the Simple Logging Facade for Java (SLF4J) API.

By default, the logback configuration is stored in `agent-logback.xml` in `/path/to/java_agents/agent_type/Agent_n/config`. Learn about how to change the location from [Agent configuration](#).

The agent supports the following log levels:

- TRACE
- INFO
- WARN
- ERROR
- ON (deprecated, this setting uses the TRACE log level)

- OFF (deprecated, this setting uses the ERROR log level)

For a full description of logging options, refer to the [Logback website](#) .

## Set the log level

The agent maintains the **last** log level it reads from either `agent-logback.xml` or Agent Debug Level.

### TIP

To debug code during the property-reading phase of startup, set the log level in `agent-logback.xml` to TRACE .

## *Remote configuration mode*

In remote configuration mode, the agent sets the log level as follows:

- At start up, the agent reads `agent-logback.xml` , and uses the configured log level.
- When the agent receives the first request, it reads `AgentBootstrap.properties` . If org.forgerock.agents.debug.level specifies a different log level, the agent changes the log level.
- The agent then reads the AM properties. If Agent Debug Level specifies different log level, the agent changes the log level.
- While the agent is running, if either of the following events occurs the agent changes the log level to the value configured by the event:
  - Agent Debug Level is updated in AM. The log level is changed immediately.
  - `agent-logback.xml` is updated in the agent, and the update is taken into account as described in Change logging behavior.

## *Local configuration mode*

In local configuration mode, the agent sets the log level as follows:

- At start up, the agent reads `agent-logback.xml` , and uses the configured log level.
- When the agent receives the first request, it reads `AgentBootstrap.properties` and then `AgentConfiguration.properties` .
  - If either file specifies a log level in org.forgerock.agents.debug.level, the agent uses that log level.
  - If both files specify a log level, the agent uses the **last** log level it reads, from `AgentConfiguration.properties` .

- While the agent is running, if either of the following events occurs the agent changes the log level to the value configured by the event:
  - The property `org.forgerock.agents.debug.level` is changed in `AgentConfiguration.properties`, and the time configured by the property `org.forgerock.agents.config.reload.seconds` elapses.
  - `agent-logback.xml` is updated in the agent, and the update is taken into account as described in Change logging behavior.

## Default logging in `agent-logback.xml`

By default, `agent-logback.xml` is configured as follows:

- When the agent starts, log messages are recorded for the agent.
- Log messages are written to a text file in `/path/to/java_agents/agent_type/Agent_n/logs/debug`, where `/path/to/java_agents/agent_type/Agent_n` is the installation directory.
- The log level is `ERROR`.
- Logs are rolled over once a day, or more frequently if the file reaches 100 MB.

All aspects of the logs are defined by the following reference `agent-logback.xml` deployed by the agent installer:

```
<configuration debug="true" scan="true" scanPeriod="60 seconds">
  <appender name="ROLLING"
class="org.forgerock.agents.shaded.ch.qos.logback.core.rolling.RollingFileAppender">

    <file>/full/path/to/logs/debug/directory/log.txt</file>
    <rollingPolicy
class="org.forgerock.agents.shaded.ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">

      <!-- rollover daily -->
      <fileNamePattern>log-%d{yyyy-MM-dd}.%i.txt</fileNamePattern>

      <!-- each file should be at most 100MB, keep 30 days
worth of history, but at most 10GB -->
      <maxFileSize>100MB</maxFileSize>
      <maxHistory>30</maxHistory>
      <totalSizeCap>10GB</totalSizeCap>
    </rollingPolicy>
  </appender>
  <encoder>
```

```
        <pattern>%level %msg%n</pattern>
    </encoder>
</appender>

<root level="ERROR">
    <appender-ref ref="ROLLING" />
</root>
</configuration>
```

## Change logging behavior

To change the logging behavior, update `agent-logback.xml`.

To take the change into account, stop and restart the agent, or include a scan and an interval configuration in `agent-logback.xml`. In the following example, `agent-logback.xml` is scanned every 60 seconds:

```
<configuration scan="true" scanPeriod="60 seconds" ... >
    ...
```

For a full description of the options for logging, refer to [the Logback website](#).

Use shaded Logback classnames, to ensure that the agent uses a unique classname compared to other deployed filters and web applications in the container. For example, use

```
org.forgerock.agents.shaded.ch.qos.logback.core.rolling.RollingFileAppender
```

instead of

```
ch.qos.logback.core.rolling.RollingFileAppender
```

## Manage errors in `agent-logback.xml`

If `agent-logback.xml` is removed and not replaced, or if a replacement `agent-logback.xml` contains errors, logs are written to the standard output for the container. For example, Tomcat logs are written to `catalina.out`.

To log an error when `agent-logback.xml` fails to load, include a debug configuration:

```
<configuration debug="true" ... >
```

...

When `agent-logback.xml` contains errors, messages like these written to the standard output for the container.

```
ERROR in ch.qos.logback.core.joran.spi.Interpreter@20:72 ...  
ERROR in ch.qos.logback.core.joran.action.AppenderRefAction ...
```

## Change the log level in agent-logback.xml

The global log level is set by default to `ERROR` by the following line of the default `agent-logback.xml`:

```
<root level="ERROR">
```

To change the global log level, configure the same line with another log level and take the changed file into account, as described in [Change logging behavior](#).

To change the log level for a single object type without changing it for the rest of the configuration, add a logger defined by the shaded classname of the object, and set its log level.

The following line in `agent-logback.xml` changes the log level for evaluating not-enforced rules, but does not change the log level of other classes or packages:

```
<logger  
name="org.forgerock.agents.shaded.com.sun.identity.agents.common.No  
tEnforcedRuleHelper" level="TRACE" />
```

## Change the character set and format of log messages

By default, logs use the system default character set where the agent is running. To use a different character set, or change the pattern of the log messages, edit `agent-logback.xml` to change the `encoder` configuration.

The following lines add the date to log messages, and change the character set:

```
<encoder>  
  <pattern>%d{yyyyMMdd-HH:mm:ss} | %-5level | %thread |  
%logger{20} | %message%n%xException</pattern>  
  <charset>UTF-8</charset>  
</encoder>
```

## Limit repeated log messages

To keep log files clean and readable, and to prevent log overflow attacks, limit the number of repeat log messages. Add a custom `agent-logback.xml` with a `DuplicateMessageFilter`. This filter detects duplicate messages, and after the specified number of repetitions, drops repeated messages.

The following example allows 5 repetitions of a log message, holds the following 10 repeated messages in the cache, and then discards subsequent repeated messages:

```
<turboFilter
class="org.forgerock.agents.shaded.ch.qos.logback.classic.turbo.DuplicateMessageFilter" allowedRepetitions="5" CacheSize="10" />
```

The `DuplicateMessageFilter` has the following limitations:

- Filters out **all** duplicate messages. It does not filter per logger, or logger instance, or logger name.
- Detects repetition of raw messages, meaning that some non-identical messages are considered as repetition.

For example, when the agent is running at `TRACE` level, the following message is written to the logs every 20 seconds when there is no other activity over the Websocket:

```
TRACE 08:22:00 (20) Sending ping to check connection is still
alive
```

This message is produced by the logging statement:

```
logger.trace("{} ({{}) Sending ping to check connection is
still alive", timeStamp, lastSent.getSeconds());
```

Because the message text `{{}} ({{}) Sending ping to check connection is still alive` is always the same, it is considered as repeat. Only the first 5 repeats are logged.

- Does not limit the lifespan of the cache. After the specified number of repetitions is reached, the repeated log messages never appear again, even if they are frequently hit.

## Change the log level in the AM admin UI

1. In the AM admin UI, go to **Realms** > *Realm Name* > **Applications** > **Agents** > **Java**, and select your Java Agent.
2. On the **Global** tab, select **Agent Debug Level**, and select a level.

Alternatively, set [Agent Debug Level](#) as a custom property.

## Notifications

---

AM sends the following notifications to Java Agent through WebSockets:

### **Configuration notifications**

When the administrator makes a change to an agent configuration property, AM sends a notification to the agent. The agent flushes the configuration cache, and rereads the agent profile from AM. For more information about the cache, refer to [Configuration cache](#).

Configuration notifications apply when you store the agent profile in AM's configuration data store. For information, refer to [Enable Notifications of Agent Configuration Change](#).

### **Session notifications**

When a client logs out, or a CTS-based session expires, AM sends a notification to the agent to remove that entry from the session cache. For information, refer to [Enable Notification of Session Logout](#). For more information about the cache, refer to [Session Cache](#).

### **Policy notifications**

When an administrator changes a policy, AM sends a notification to the agent to flush the policy cache. For information, refer to [Enable Notification of Policy Changes](#). For more information about the cache, refer to [Policy Cache](#).

Notifications are enabled by default. To disable notifications, unsubscribe separately to each type of notification. If [Autonomous mode](#) is `true`, notifications and many other features are automatically disabled.

In configurations with load balancers and reverse proxies, make sure that the load balancers and reverse proxies support WebSockets.

For more information about properties that configure notifications, refer to [Notifications](#) in the *Properties reference*.

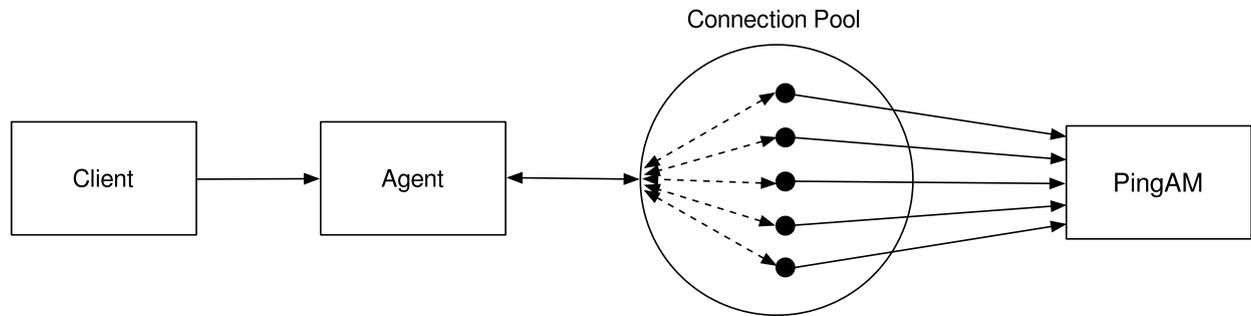
## Tune connections

---

Use a connection pool between Java Agent and AM to reuse connections, and so reduce the overhead of creating new connections.

Connection pooling can improve performance in some circumstances. Test and tune the performance of your deployment with connection pooling before you use it in a production environment.

The following image shows the architecture of a connection pool:

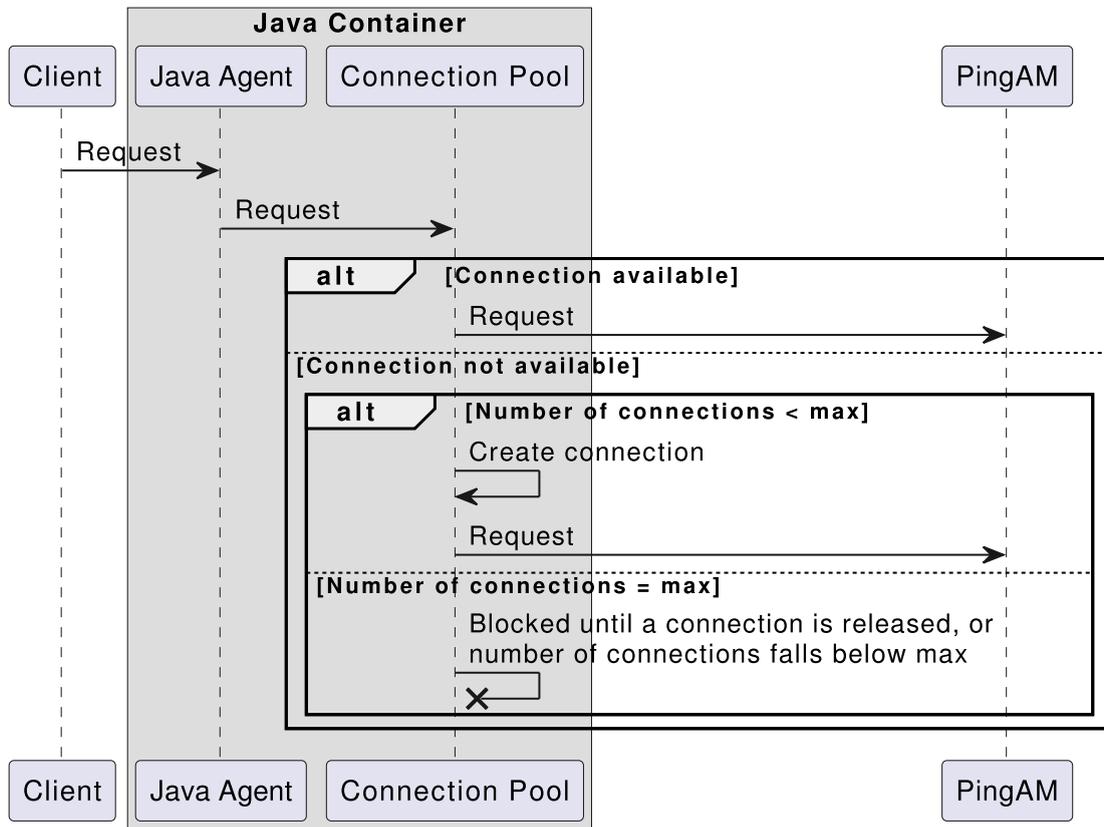


When a client makes a request, the agent intercepts the request, and uses the connection pool to connect to AM. If a connection is available, the agent uses that connection. The client is unaware of the connection reuse.

If a connection is not available, and the maximum number of connections is not reached, the agent creates and uses a new connection. When the maximum number of allowed connections is reached, the request waits until an existing connection is released, or a new connection can be created.

When the request is complete, the agent closes the connection to the pool but the physical connection is retained by the pool. The connection is then available to requests with the same connection parameters.

The following image shows the flow of information when a request is treated in a connection pool:



When Enable Connection Pooling is `true`, use the following properties to tune the connection pool:

Property	Description
<u>Max HTTP Connection Count</u>	<p>By default, the connection pool can contain up to 1000 HTTP connections. Change the value of this property according to the AM load, as follows:</p> <ul style="list-style-type: none"> <li>• If AM is likely to be overloaded, reduce the maximum number of connections to reduce the load on AM.</li> <li>• If AM is not likely to be overloaded, use the default or a higher value so that connections are available when they are requested.</li> </ul>
<u>HTTP Connection Timeout</u>	<p>By default, connections wait for up to 10 seconds for a response before they are considered stale.</p> <p>Tune this property to allow enough time for systems to respond, and therefore prevent unnecessary failures, but to minimize the time to wait after a network failure.</p>

Property	Description
<u>HTTP Socket Timeout</u>	<p>By default, when a continuous data flow between the connection pool and AM is interrupted, the socket is considered stale.</p> <p>Tune this property to allow enough time for systems to respond, and therefore prevent unnecessary failures, but to minimize the time to wait after a socket failure.</p>
<u>Enable HTTP Connection Reuse</u>	<p>By default, after a request is complete the connection is returned to the connection pool. The physical connection is retained and can be reused by another request with the same connection parameters.</p> <p>Reuse connections to reduce the overhead of creating a new connection each time one is requested.</p> <p>If you are using the connection pool only as a throttling mechanism, to limit the total number of simultaneous connections, it is not necessary to reuse connections.</p>
<u>Enable HTTP Connection State</u>	<p>When <u>Enable HTTP Connection Reuse</u> is <code>true</code>, connection reuse is enabled by default for Apache HTTP Client.</p>
<u>Enable HTTP Retry</u>	<p>By default, requests are retried up to three times after failure.</p> <p>If requests are likely to pass on retry, enable this property to reduce the overhead of unnecessarily killing and remaking connections.</p> <p>If requests are likely to fail on retry, disable this property to reduce the retry time.</p>

## Troubleshoot

---

Ping Identity provides support services, professional services, training, and partner services to help you set up and maintain your deployments. Learn more in [Getting support](#).

### Get information about the problem

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 there circumstances in which the problem does not occur?
- Is the problem permanent, intermittent, getting better, getting worse, or staying the same?

If you contact us 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 or container and version
  - Java version
  - Patches or other software that might affect the problem

## WebSocket issues

If you're experiencing issues with WebSocket connections, perform the following troubleshooting steps:

- Check the WebSocket jars are loading
- Test the WebSocket connection

### *Check the WebSocket jars are loading*

You can check the jars are being loaded on the AM Tomcat as follows:

1. Run the noisy command from the Tomcat `bin` directory, for example:

```
$ cd /path/to/tomcat/bin
$ lsof | grep websocket
```

- If the WebSocket jars are loading, you'll see responses similar to the following:

```
java 1014 root mem REG 8,1 225632 2097375
/usr/local/tomcat/lib/tomcat-websocket.jar
```

```
java 1014 root mem REG 8,1 36905 2097376
/usr/local/tomcat/lib/websocket-api.jar
java 1014 root 38r REG 8,1 36905 2097376
/usr/local/tomcat/lib/websocket-api.jar
...
```

- If the WebSocket jars aren't loading, you won't see them listed.
2. Add the following option to the Tomcat startup script ( `setenv.sh` ) to view further details about the Java WebSocket API loading:

```
JAVA_OPTS=-verbose:class
```

The Java WebSocket API is bundled with Tomcat.

3. Restart the web container.
4. Review the `catalina.out` log file for WebSocket details. For example, you'll see entries similar to the following if the WebSocket API is available:

```
$ cat ../logs/catalina.out | grep websocket

[Loaded
org.forgerock.openam.notifications.websocket.JsonValueDecoder
from file:/path/to/tomcat/webapps/am/WEB-INF/lib/openam-
notifications-websocket-7.5.0.jar]

[Loaded
org.forgerock.openam.notifications.websocket.NotificationsWebS
ocketConfigurator from file:/path/to/tomcat/webapps/am/WEB-
INF/lib/openam-notifications-websocket-7.5.0.jar]

[Loaded javax.websocket.EncodeException from
file:/path/to/tomcat/lib/websocket-api.jar]

...
```

#### NOTE

The `openam-notifications-websocket-x.x.x.jar` is required for WebSockets to work. If it's missing, you'll see `404` responses. To resolve this, verify your Tomcat configuration or contact your System Administrator for further assistance.

### *Test the WebSocket connection*

You can test the WebSocket connection by sending the agent's token to the notifications endpoint using curl.

1. Authenticate as the agent to return the agent's token:

```
$ curl \  
--request POST \  
--header "X-OpenAM-Username: agent-id" \ ①  
--header "X-OpenAM-Password: password" \ ②  
--header "Content-Type: application/json" \  
--header "Accept-API-Version: resource=2.1" \  
'https://am.example.com:8443/am/json/realms/root/realms/alpha/  
authenticate?auth-service' ③
```

- ① Replace *agent-id* with the ID of the agent profile you created.
- ② Replace *password* with the agent password.
- ③ Replace *auth-service* with either  
authIndexType=module&authIndexValue=Application or  
authIndexType=service&authIndexValue=Agent depending on whether  
you authenticate using the default non-configurable authentication module or  
a journey called Agent .

If authentication is successful, the response includes the `tokenId` that corresponds to the agent session and the URL to which the agent would normally be redirected. For example:

```
{  
  "tokenId": "AQIC5wM...TU30Q*"  
  "successUrl": "/am/console"  
  "realm": "/alpha"  
}
```

2. Send the agent's token to the notifications endpoint, for example:

```
$ curl \  
--verbose \  
--show-headers \  
--no-buffer \  
--header "Connection: Upgrade" \  
--header "Upgrade: websocket" \  
--header "Host: agent.example.com:443" \  
--header "Origin: https://notagent.example.com:8443" \  
--header "Sec-WebSocket-Key: SGVsbG8sIHdvcmxkIQ==" \  
--header "Sec-WebSocket-Version: 13" \  

```

```
--header "iPlanetDirectoryPro: AQIC5wM...TU30Q*"
'https://am.example.com:8443/am/notifications'
```

The notifications endpoint applies some login processing logic using a servlet filter and returns one of the following responses:

### **101 response**

A 101 response indicates everything is ok. It confirms the request is valid, it has been upgraded successfully, and the WebSockets connection is working correctly.

#### ▼ [Example 101 response](#)

```
* Trying 198.51.100.0...
* TCP_NODELAY set
* Connected to am.example.com (198.51.100.0) port 8443
(#0)
> GET /am/notifications HTTP/1.1
> Host: agent.example.com:443
> User-Agent: curl/8.71.0
> Accept: */*
> Connection: Upgrade
> Upgrade: websocket
> Origin: https://notagent.example.com:8443
> Sec-WebSocket-Key: SGVsbG8sIHdvcmxkIQ==
> Sec-WebSocket-Version: 13
> iPlanetDirectoryPro: AQIC5wM...TU30Q*

>

< HTTP/1.1 101
HTTP/1.1 101
< X-Frame-Options: SAMEORIGIN
X-Frame-Options: SAMEORIGIN
< Upgrade: websocket
Upgrade: websocket
< Connection: upgrade
Connection: upgrade
< Sec-WebSocket-Accept: qGEgH3En71di5rrssAZTmtRTyFk=
Sec-WebSocket-Accept: qGEgH3En71di5rrssAZTmtRTyFk=
< Date: Mon, 21 Oct 2024 17:38:15 GMT
Date: Mon, 21 Oct 2024 17:38:15 GMT
```

### **403 response**

A 403 Access Forbidden response means the agent has failed to establish a WebSocket connection with AM. You'll see 401 responses for this issue in the

logs.

#### **404 response**

A 404 response typically means the WebSocket request has not been upgraded but the request sent to the notifications endpoint is valid. Possible causes for a 404 response are:

- A network issue such as incorrectly configured load balancers or reverse proxies.
- A Tomcat issue such as a missing `openam-notifications-websocket-x.x.x.jar`.

## Common issues and solutions

### *Installation and upgrade*

#### ▼ [Cannot install over HTTPS](#)

##### **Question**

What should I do if I get an error like this when installing an agent with an HTTPS connection to AM:

```
AM server URL: https://am.example.com:8443/am
```

```
WARNING: Unable to connect to AM server URL....
```

```
If the AM server is TLS-enabled and the root CA certificate for the AM server certificate has been not imported into the installer JVMs key store (see installer-logs/debug/Agent.log for detailed exception), import the root CA certificate and restart the installer; or continue installation without verifying the AM server URL.
```

##### **Answer**

The Java platform includes certificates from many certificate authorities (CAs). If, however, you run your own CA, or you use self-signed certificates for HTTPS on the web application container where you run AM, then the **agentadmin** command cannot trust the certificate presented during connection to AM, and so cannot complete installation correctly.

After setting up the web application container where you run AM to use HTTPS, get the certificate to trust in a certificate file. The certificate you want is that of the CA

who signed the container certificate, or the certificate itself if the container certificate is self-signed.

Copy the certificate file to the system where you plan to install the Java agent. Import the certificate into a trust store that you will use during Java agent installation. If you import the certificate into the default trust store for the Java platform, then the **agentadmin** command can recognize it without additional configuration.

You can find details about export and import of self-signed certificates in [Configuring AM's container for HTTPS](#) in AM's *Installation guide*.

#### ▼ [Redirection URI error after upgrade](#)

##### **Question**

I have upgraded my agent and I see the following message in the agent log:

```
redirect_uri_mismatch. The redirection URI provided does not match a pre-registered value.
```

What should I do?

##### **Answer**

Java Agent accept only requests sent to the URL specified by the Agent Root URL for CDSSO property. For example, `https://agent.example.com/`.

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

- Configuring [Alternative Agent Protocol](#) to access the agent through different protocols. For example, `http://agent.example.com/` and `https://agent.example.com/`.
- Configuring [Alternative Agent Host Name](#) to access the agent through different virtual host names. For example, `https://agent.example.com/` and `https://internal.example.com/`.
- Configuring [Alternative Agent Port Number](#) to access the agent through different ports. For example, `https://agent.example.com/` and `https://agent.example.com:8443/`.

#### *Other issues*

#### ▼ [Detect the path of a resource loaded by classloader](#)

##### **Question**

Why does the agent fail to initialize with an error like this:

```
Servlet failed with an Exception:  
java.lang.NoSuchMethodError:  
com.sun.identity.shared.configuration.SystemPropertiesManager  
.getAsInt(Ljava/lang/String;I)I
```

### Answer

An out-of-date version of the SystemPropertiesManager class is deployed in another jar, that did not have the getAsInt function, and the classloader chose the outdated class instead of the agent's class.

Follow these steps to help locate jars containing outdated classes by showing where these classes are loaded from:

1. Add the property `-Ddisplay.classpath.mode.enabled=true` to the JVM properties in the container where the agent runs, and restart the container.

```
JAVA_OPTS="$JAVA_OPTS -  
Ddisplay.classpath.mode.enabled=true"
```

2. Access the URL of a protected resource, using the class name as a query parameter. The following example requests the path of the SystemPropertiesManager package:

```
https://www.example.com/myapp/index.html?  
com.sun.identity.shared.configuration.SystemPropertiesMana  
ger
```

If the agent finds the package, it displays the path. Otherwise, it displays an error.

3. After troubleshooting, remove `Ddisplay.classpath.mode.enabled=true` from the JVM properties. While it is present, the agent returns the class path but performs no other function.

### ▼ [Infinite redirection loops with stateless sessions](#)

#### 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:

```
<timestamp> +0000 ERROR [c53...348]state identifier not  
present in authentication state  
<timestamp> +0000 WARNING [c53...348]unable to verify pre-  
authentication cookie  
<timestamp> +0000 WARNING
```

```
[c53...348]convert_request_after_authn_post(): unable to
retrieve pre-authentication request data
<timestamp> +0000 DEBUG [c53...348] exit status: forbidden
(3), HTTP status: 403, subrequest 0
```

What is happening?

### **Answer**

This redirection loop happens because the client-based (stateless) session cookie is surpassing the maximum supported browser header size. Because 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. Learn more in AM's [Security guide](#).

### ▼ [WebSocket connections](#)

#### **Question**

I am seeing errors such as the following:

```
WARNING: Failed to create new WebSocket connection, backing
off
org.forgerock.openam.agents.notifications.websocket.WebSocket
ConnectionException: Failed to create connection
```

#### **Answer**

Make sure any load balancers or reverse proxies configured in your environment support WebSocket protocols.

Was this helpful?  