.

Troubleshooting Java Agent Issues

On this page:

Related pages:

- Troubleshoot Controller Issues

This topic discusses techniques for troubleshooting agent installation and operation. In particular, it describes how to find and interpret information in agent log files.

Resolving Java Agent Startup Issues

The first thing the Java agent does upon application startup is register with the Controller. Once registered, the agent should appear in the **Settings** > **AppDynamics Agents** list.

If you do not see the agent in the list within a few minutes, check the following:

1. Make sure you have restarted the application server.
2. Verify that the javaagent argument has been added to the startup script of your JVM.
3. Verify that you configured the agent-controller communication properties and agent identification properties in the controller-info.xml file or as system properties in the startup script of your JVM. See Java Agent Configuration Properties.
4. Check the Agent logs directory located at <agent_home>/logs/<Node_Name> for the agent.log file.
5. Verify that the Agent is compatible with the Controller. For details see Agent and Controller Compatibility.

Locating the Java Agent Log Files

Agent log files are located in the <agent_home>/logs/<node_name> folder.

The agent.log file is the recommended file to help you with troubleshooting. This log can indicate the following:

- Incomplete information in your Agent configuration
- The Controller port is blocked
- Incorrect file permissions

Error messages related to starting the Java Agent use this format:

```
ERROR com.singularity.JavaAgent - Could Not Start Java Agent
```

Resolving Incomplete Agent Configuration Issues

The following table lists the typical error messages for incomplete Agent configuration:

Error Message	Solution
Cannot connect to the Agent - ERROR com.singularity.XMLConfigManager - Incomplete Agent Identity data, Invalid Controller Port Value	This indicates that the value for the Controller port in controller-info.xml is missing. Add the Controller port and host value to resolve: • For on-premises Controller installations: 8090 for HTTP and 8181 for HTTPS. • For Controller SaaS service, use the default HTTPS port 443.
Caused by: com.singularity.ee.agent.configuration.a: Could not resolve agent-controller basic configuration	This is usually caused because of incorrect configuration in the Controller-info.xml file. Ensure that the information for agent communication (Controller host and port) and agent identification (application, tier and node names) is correctly configured. Alternatively, you can also use the system properties (-D options) or environment variables to configure these settings.

Unblocking the Controller Port

The following table lists the typical error message when the Controller port is blocked in your network:

Error Message	Solution
ERROR com.singularity.CONFIG.ConfigurationChannel - Fatal transport error: Connection refused WARN com.singularity.CONFIG.ConfigurationChannel - Could not connect to the controller/invalid response from controller, cannot get initialization information, controller host \x.x.x.x\, port 8090, exception Fatal transport error: Connection refused	Try pinging the Controller from the machine where you have configured the application agent. To check if a port is blocked in the network, use the commands: • `netstat -an` for Windows • `nmap` for Linux. The default ports are: • For on-premises Controller installations: 8090 for HTTP and 8181 for HTTPS. • For Controller SaaS service, use the default HTTPS port 443.

Correcting File Permission Issues

Following table lists the typical error message when the file permissions are not correct:

Error Message	Solution
ERROR com.singularity.JavaAgent - Could Not Start Java Agent com. singularity.ee.agent.appagent.kernel.spi.c: Could not start services"	This is usually caused because of incorrect permissions for log files. To troubleshoot: Confirm whether the user who is running the server has read and write permission on the agent directories. If the user has **chmod a-r** equivalent permission, change the permission to **chmod a+r** "<agent_home>"

Maximum Async Handoff Call Graph Samples Error

The following error indicates that the number of handoffs in an asynchronous has exceeded the limit:

```
"WARN AsyncHandOffIdentificationInterceptor - Reached maximum limit 500 of async handoff call graph
samples. No more samples will be taken" Error
```

This can result from transactions being misidentified as async transactions. In AppDynamics 3.6 and later, all Runnables, Callables and Threads are instrumented by default except those that are excluded by the agent configuration in app-agent-config.xml.

In some environments, this may result in too many classes being instrumented, or cause common classes in a framework that implements the Runnable interface to be mistaken for asynchronous activity when it is not, for example Groovy applications using Clojure.

To debug, check the call graph for asynchronous activities that are misidentified as asynchronous activities. If found, exclude the packages that are not really asynchronous activities.

See Enable Thread Correlation for Java.