Java Agent Configuration Properties

On this page:

Related pages:

- Administer the Java Agent
- Use System Properties for Java Agent Settings
- Instrument JVMs in a Dynamic Environment

This topic is a reference for the configuration properties for the AppDynamics Java Agent. If you are also installing a Machine Agent on the same machine with the Java Agent, see Standalone Machine Agent Installation Scenarios.

Agent-Controller Communication Properties

Controller Host

The host name or the IP address of the AppDynamics Controller. Example values are 192.168.1.22 or myhost or myhost.example.com. This is the same host that you use to access the AppDynamics browser-based user interface. For an On-premises Controller, use the value for Application Server Host Name that was configured when the Controller was installed. If you are using the AppDynamics SaaS Controller service, see the Welcome email from AppDynamics.

Element in controller-info.xml: `<controller-host>`

System Property: –`Dappdynamics.controller.hostName`

Environment Variable: APPDYNAMICS_CONTROLLER_HOST_NAME

Type: String

Default: None

Required: Yes, if the Enable Orchestration property is false.

If Enable Orchestration is true, and if the app agent is deployed in a compute cloud instance created by an AppDynamics workflow, do not set the Controller host unless you want to override the auto-detected value. See Enable Orchestration Property.

Controller Port

The HTTP(S) port of the AppDynamics Controller. This is the port used to access the AppDynamics browser-based user interface.

If the Controller SSL Enabled property is set to true, specify the HTTPS port of the Controller; otherwise specify the HTTP port. See Co

Copyright © AppDynamics 2012-2018

ntroller SSL Enabled Property.

Element in controller-info.xml: `<controller-port>`

System Property: –`Dappdynamics.controller.port`

Environment Variable: APPDYNAMICS_CONTROLLER_PORT

Type: Positive Integer

Default: For On-premises installations, port 8090 for HTTP and port 8181 for HTTPS are the defaults.
For the SaaS Controller Service, use port 443 for HTTPS connections.

Required: Yes, if the Enable Orchestration property is false.

If Enable Orchestration is true, and if the app agent is deployed in a compute cloud instance created by an AppDynamics workflow, do not set the Controller port unless you want to override the auto-detected value. See Enable Orchestration Property.

SSL Configuration Properties

Controller SSL Enabled

If true, specifies that the agent should use SSL (HTTPS) to connect to the Controller. If SSL Enabled is true, set the Controller Port property to the HTTPS port of the Controller. See Controller Port Property.

Element in controller-info.xml: `<controller-ssl-enabled>`

System Property: –`Dappdynamics.controller.ssl.enabled`

Environment Variable: APPDYNAMICS_CONTROLLER_SSL_ENABLED

Type: Boolean

Default: False

Required: No

Controller Keystore Password

The plain text value of the Controller certificate password. If Use Encrypted Credentials is true, encrypt the password. See Encrypt Agent Credentials.

Element in controller-info.xml: `<controller-keystore-password>`

Type: String

Default: None

Required: No

Controller Keystore Filename

By default, the agent looks for a Java truststore file named cacerts.jks in the <agent_home>/<version>/conf directory in the agent home. Use this property to enable full validation of Controller SSL certificates with a different Java truststore file. See Enable SSL for the Java Agent.

Element in controller-info.xml: `<controller-keystore-filename>`

Type: String

Default: None

Copyright © AppDynamics 2012-2018

Required: No

Force Default SSL Certificate Validation

Used to override the default behavior for SSL validation. The property can have three states:

- **true**: Forces the agent to perform full validation of the certificate sent by the controller, enabling the agent to enforce the SSL trust chain. Use this setting when a public certificate authority(CA) signs your Controller SSL certificate. See Enable SSL On-Premises with a Trusted CA Signed Certificate.
- **false**: Forces the agent to perform minimal validation of the certificate. This property disables full validation of the Controller's SSL certificate. Use this setting when full validation of a SaaS certificate fails.
- **unspecified**: The validation performed by the agent depends on the context:
    - If the agent is connecting to a SaaS controller, full validation is performed.
    - If the agent is connecting to an On-premises controller, and the cacerts.jks file is present, then full validation is performed using the cacerts.jks file.
    - If the agent is connecting to an On-premises controller, and there is no cacerts.jks file, then minimal validation is performed

System Property: –`Dappdynamics.force.default.ssl.certificate.validation`

Type: Boolean

Default: None

Required: No

AppDynamics Agent SSL Protocol

The SSL compatibility table in Agent and Controller Compatibility lists the default security protocol for the different versions of the Java Agent. If the default security protocol for your version of an agent is incompatible with the Controller or it is incompatible with an intervening proxy, pass the `-Dappdynamics.agent.ssl.protocol` system property to configure one of the following security protocols:

- SSL
- TLS
- TLSv1.2
- TLSv1.1

System Property: –Dappdynamics.agent.ssl.protocol

Type: String

Default: See Agent and Controller Compatibility

Required: No

Secure Credential Store Properties

Use Encrypted Credentials

Before you enable Use Encrypted Credentials, see Encrypt Agent Credentials for instructions on how to initialize the Secure Credential Store.

Set Use Encrypted Credentials to True to configure the agent to use credentials encrypted with the Secure Credential Store. When you enable Use Encrypted Credentials, you must supply the Credential Store Filename and the obfuscated Credential Store Password. For more information, see Encrypt Agent Credentials.

Copyright © AppDynamics 2012-2018

When Use Encrypted Credentials is true, encrypt the following:

- Account Access Key
- Controller Keystore Password
- Proxy Password

Element in controller-info.xml: `<use-encrypted-credentials>`

Type: Boolean

Default: False

Required: No

Credential Store Filename

The absolute path to the Secure Credential Store keystore. For more information, see Encrypt Agent Credentials.

Element in controller-info.xml: `<credential-store-filename>`

Type: String

Default: None

Required: If Use Encrypted Credentials is set to True

Credential Store Password

The obfuscated keystore password for the Secure Credential Store. For instructions on how to obfuscate the password, see Encrypt Agent Credentials.

Element in controller-info.xml: `<credential-store-password>`

Type: String

Default: None

Required: If Use Encrypted Credentials is set to True

Agent Identification Properties

Automatic Naming

If enabled and other agent identification properties are not specified in other settings, the tier and application for the agent are automatically named. The default names are in the format MyApp and MyTier.

Element in controller-info.xml: `<auto-naming>`

Type: Boolean

Default: None

Required: No

Application Name

The name of the logical business application that this JVM node belongs to. Note that this is not the deployment name(ear/war/jar) on

Copyright © AppDynamics 2012-2018

the application server.

If a business application of the configured name does not exist, it is created automatically.

Element in controller-info.xml: `<application-name>`

System Property: –`Dappdynamics.agent.applicationName`

Environment Variable: `APPDYNAMICS_AGENT_APPLICATION_NAME`

Type: String

Default: None

Required: Yes

Tier Name

The name of the tier that this JVM node belongs to. Note that this is not the deployment name (ear/war/jar) on the application server.

If the JVM or application server startup script already has a system property that references a tier, such as `-Dserver.tier`, you can use `${server.tier}` as the tier name. For more information, see Use System Properties for Java Agent Settings.

The agent registers the named tier with the Controller, if the tier does not already exist, the first time it connects with the Controller. If a tier with the name already exists in the Controller model, the agent is associated with the existing tier.

Element in controller-info.xml: `<tier-name>`

System Property: –`Dappdynamics.agent.tierName`

Environment Variable: `APPDYNAMICS_AGENT_TIER_NAME`

Type: String

Default: None

Required: Yes

Node Name

The name of the node. Where JVMs are dynamically created, use the system property to set the node name.

If your JVM or application server startup script already has a system property that can be used as a node name, such as `-Dserver.name`, you could use `${server.name}` as the node name. You could also use expressions such as `${server.name}_${host.name}.MyNode` to define the node name. See Use System Properties for Java Agent Settings for more information.

In general, the node name must be unique within the business application and physical host. If you want to use the same node name for multiple nodes on the same physical machine, create multiple virtual hosts using the Unique Host ID property. See Unique Host ID.

Element in controller-info.xml: `<node-name>`

System Property: –`Dappdynamics.agent.nodeName`

Environment Variable: `APPDYNAMICS_AGENT_NODE_NAME`

Type: String

Default: None

Required: Yes

Reuse Node Name

Set this property to true to reuse node names in AppDynamics. When you set the property to true, you don't need to supply a node

Copyright © AppDynamics 2012-2018

name, but you do need to provide a node name prefix using `-Dappdynamics.agent.reuse.nodeName.prefix` .

This property is useful for monitoring environments where there are many JVMs with short life spans. When true, AppDynamics reuses the node names of historical JVMs for new JVMs. This avoids a proliferation of differently named nodes in AppDynamics over time, particularly when the nodes are essentially identical processes that run over different times. An example of this environment is a z/OS Dynamic Workload Manager based-environment where new JVMs are launched and shut down based on actual workload.

AppDynamics generates a node name with App, Tier and Sequence number. The node names are pooled. For example, the sequence numbers are reused when the nodes are purged (based on the node lifetime).

When the Java Agent starts up, it logs output to the console until it registers with the Controller and the Controller generates the node name. To configure the agent to write logs to a file, edit the log4j-unknown.xml at <agent_home>/<version_number>/conf/logging. See I nstrument JVMs in a Dynamic Environment.

The Controller reuses node names based on the node retention period property.

System Property: `–Dappdynamics.agent.reuse.nodeName`

Environment Variable: APPDYNAMICS_JAVA_AGENT_REUSE_NODE_NAME (New in 4.5.8)

Type: Boolean

Default: False

Required: No

Example: With the following configuration, the Controller generates a node name with the prefix "reportGen". Node names will have suffixes –1, –2, and so on, depending on the number of nodes are running in parallel. The name of a node that is shut down and qualifies as a historical node may be reused by a new node.

```
-Dappdynamics.agent.reuse.nodeName=true -Dappdynamics.agent.reuse.nodeName.prefix=reportGen
```

Reuse Node Name Prefix

When you configure the agent to reuse node names, use this property to specify the prefix the Controller uses to generate node names dynamically.

System Property: `–Dappdynamics.agent.reuse.nodeName.prefix`

Environment Variable: APPDYNAMICS_JAVA_AGENT_REUSE_NODE_NAME_PREFIX (New in 4.5.8)

Type: String

Default: None

Required: When `-Dappdynamics.agent.reuse.nodeName=true`

Example: Using the following property specifications, the agent directs the Controller to generate a node name with the prefix "reportGen". Node names will have suffixes --1, --2, and so on, depending on how many nodes are running in parallel.

```
-Dappdynamics.agent.reuse.nodeName=true -Dappdynamics.agent.reuse.nodeName.prefix=reportGen
```

Self Service

Configure the Java Agent to automatically name nodes based upon the platform. For automatic node naming to work, you must specify an application name and a tier name.

System Property: `–Dappdynamics.agent.selfService`

Type: String

Values:

tibco: The Java Agent names nodes for the TIBCO process name. See Configure the Java Agent for TIBCO BusinessWorks for more information.

Copyright © AppDynamics 2012-2018

Default: None

Required: No

Account Properties

If the AppDynamics Controller is running in multi-tenant mode or if you are using the AppDynamics SaaS Controller, specify the account name and key for the agent to authenticate with the Controller.

If you are using the AppDynamics SaaS Controller, the account name is provided in the Welcome email sent by AppDynamics. You can also find this information in the <controller_home>/initial_account_access_info.txt file.

If the Controller is running in single-tenant mode, you only need to configure the account access key. You can find the unique access key for your Controller instance from the License Management page in the UI.

Account Name

The account name used to authenticate with the Controller.

Element in controller-info.xml: `<account-name>`

System Properties: `–Dappdynamics.agent.accountName`

Environment Variable: `APPDYNAMICS_AGENT_ACCOUNT_NAME`

Type: String

Default: None

Required: Yes for AppDynamics SaaS Controller and other multi-tenant users; no for single-tenant users.

Account Access Key

The account access key used to authenticate with the Controller. If Use Encrypted Credentials is true, encrypt the account access key. See Encrypt Agent Credentials.

Element in controller-info.xml: `<account-access-key>`

System Properties: `–Dappdynamics.agent.accountAccessKey`

Environment Variable: `APPDYNAMICS_AGENT_ACCOUNT_ACCESS_KEY`

Type: String

Default: None

Required: Yes

> If you provide application keys through JVM system properties or environment variables, ensure that you use quotes to wrap any shell special characters that may be contained within application keys to prevent the Shell from interpreting them. See the shell documentation for more detail.

Proxy Properties for the Controller

Use the proxy properties to configure the agent to connect to the Controller through a proxy.

Copyright © AppDynamics 2012-2018

Proxy authentication cannot be used in conjunction with agent SSL. To connect the agent through a proxy via SSL, the proxy must be open (not require the agent to authenticate).

Proxy Host

The proxy host name or IP address.

System Property: –`Dappdynamics.http.proxyHost`

Type: String

Default: None

Required: No

Proxy Port

The proxy HTTP(S) port.

System Property: –`Dappdynamics.http.proxyPort`

Type: Positive Integer

Default: None

Required: No

Proxy User Name

The name of the user that is authenticated by the proxy host.

System Property: –`Dappdynamics.http.proxyUser`

Type: String

Default: None

Required: No

Proxy Password

The absolute path to the file containing the password of the user that is authenticated by the proxy host. The password must be the first line of the file.

If Use Encrypted Credentials is false, enter the password in plain text. If Use Encrypted Credentials is true, encrypt the password. See Encrypt Agent Credentials.

System Property: –`Dappdynamics.http.proxyPasswordFile`

Type: String

Default: None

Required: No

Example: –Dappdynamics.http.proxyPasswordFile=/path/to/file-with-password

Copyright © AppDynamics 2012-2018

Other Properties

Enable Orchestration

When set to true, enables auto-detection of the controller host and port when the app server is a compute cloud instance created by an AppDynamics orchestration workflow. See Controller Host Property and Controller Port Property.

In a cloud compute environment, auto-detection is necessary for the Create Machine tasks in the workflow to run correctly.

If the host machine on which this agent resides is not created through AppDynamics workflow orchestration, this property should be set to false.

Element in controller-info.xml: `<enable-orchestration>`

Type: Boolean

Default: False

Required: No

Agent Runtime Directory

Sets the directory under which all files the agent writes at runtime. If this property is specified, all agent logs are written to <Agent-Runtime–Directory>/logs/node-name and transaction configuration is written to the <Agent-Runtime-Directory>/conf/node-name directory. The log folder location can be overridden with the `appdynamics.agent.logs.dir` property.

Element in controller-info.xml: `<agent-runtime-dir>`

System Property: `–Dappdynamics.agent.runtime.dir`

Environment Variable: `APPDYNAMICS_AGENT_BASE_DIR`

Type: String

Default: <agent_home>/nodes

Required: No

Redirect Logfiles

Sets the destination directory to which the logs will be written to.

System Property: `–Dappdynamics.agent.logs.dir`

Type: String

Default: <agent_home>/logs/<Node_Name>

Required: No

Custom path for agent conf directory

Sets a custom path for the agent conf directory. This is where the agent reads its static config files from. If you need to change custom-activity-correlation.xml or app-agent-config.xml and the agent installation is read-only, this instructs the agent to read the static config files from elsewhere.

System Property: `–Dappdynamics.agent.conf.dir`

Type: String

Default: <agent_home>/ver4.5.x.x.x/conf

Copyright © AppDynamics 2012-2018

Required: No

Force Agent Registration

Set to true only under the following conditions:

- The agent has been moved to a new application or tier from the UI, and
- You want to override that move by specifying a new application name or tier name in the agent configuration

Element in controller-info.xml: `<force-agent-registration>`

Type: Boolean

Default: False

Required: No

Auto Node Name Prefix

Set this property if you want the Controller to generate node names automatically using a prefix that you provide.

The Controller generates node names by concatenating the specified prefix with a UUID suffix. For example, if you set the prefix as follows:

`–Dappdynamics.agent.auto.node.prefix=JoannaAutoNode`

The generated node name is

`JoannaAutoNode_d39dbfc1-6f4b-4eb7-a788-c1c0135b6bcb`

This property provides a similar function to the Reuse Node Name Prefix Property property. However, this property is not meant to be used in combination with reusing node names; use Reuse Node Name Prefix Property for those cases instead.

Element in controller-info.xml: Not applicable

System Property: –Dappdynamics.agent.auto.node.prefix=<your_prefix>

Type: String

Default: Serial number maintained by the Controller appended to the tier name

Required: No

Cron/Batch JVM

Set this property to true if the JVM is a batch/cron process. This property can be used to stall the shutdown to allow the agent to send metrics before shutdown.

Element in controller-info.xml: Not applicable

System Property: –Dappdynamics.cron.vm

Type: Boolean

Default: False

Required: No

Unique Host ID

Logically partitions a single physical host or virtual machine such that it appears to the Controller that the application is running on

Copyright © AppDynamics 2012-2018

different machines. Set the value to a string that is unique across the entire managed infrastructure. The string may not contain any spaces. If you have a machine agent associated with the application monitored by the app agent, then this property must be set on the machine agent to the same value. See Standalone Machine Agent Installation Scenarios.

System Property: –Dappdynamics.agent.uniqueHostId

Environment Variable: APPDYNAMICS_AGENT_UNIQUE_HOST_ID

Type: String

Default: None

Required: No

Agent Meta Info

Allows you to associate arbitrary information with a node, which can then be used as a basis for applying health rules or policies by node. For example, you can exclude a health rule from applying to agents tagged as test agents based on a meta-info property. Pass the property in key;value format (for example, "`key1;value1;key2;value2`"). Do not use semicolons as value(s) as it is used as a delimiter.

For information on using the properties in health rules or policies (along with built-in meta-info properties), see Configure Health Rules o r Configure Policies.

System Property: –Dappdynamics.agent.node.metaInfo

Type: String

Default: None

Required: No

Low Entropy

Addresses agent startup issues in systems with low to zero entropy available for seeding the PRNG algorithm. We use the NativePRNGNonBlocking algorithm via SecureRandom if the system property appdynamics.low.entropy is set.

System Property: appdynamics.low.entropy=true

Element in controller-info.xml: No

Environment Variable: No

Type: Boolean

Default: False

Required: No

If `appdynamics.low.entropy=true` then the agent takes measures to ensure it does not block when generating random values, even in the absence of entropy.

Analytics Agent

For use with the transaction analytics feature with a remote (or non-default) Analytics agent. See Enable the App Server Agent for a Remote Analytics Agent for details.

System Property: –Dappdynamics.analytics.agent.url

Element in controller-info.xml: No

Environment Variable: No

Type: String

Copyright © AppDynamics 2012-2018

Default: http://localhost:9090/v2/sinks/bt

Required: No

Copyright © AppDynamics 2012-2018