Universal Agent Rulebooks

On this page:

The Universal Agent uses rulebooks to manage the deployment and maintenance of runtime agents. This topic introduces rulebooks and strategies for applying them.

Working with Rulebooks

A rulebook is a JSON-formatted configuration file that contains the rules that define runtime agent instances. Entries in the rulebook direct the Universal Agent to install, stop, or start agents.

When you start the Universal Agent, the Universal Agent applies a rulebook. The rulebook contains general settings, subject to condition evaluation logic, for the Universal Agent, along with rules for individual types of runtime agents. When you add a runtime agent rule to the rulebook, the Universal Agent subject to the rule retrieves the runtime agent from the repository and installs it in the `monitor` directory in the Universal Agent home. The `monitor` directory contains the base install directories for each runtime agent.

Operation Mode

The Universal Agent can run in two modes:

- Controller mode: This is the default mode where the Universal Agent retrieves and applies the rulebook stored and served by the Controller, `controller-book.json`. This rulebook is thus shared by all instances of the Universal Agent that are running in *controller* mode.
- Local mode: Used for specific use cases, the Universal Agent applies a locally stored rulebook from the Universal Agent home directory, that is specific to this instance of the agent and is not shared with other instances of the Universal Agent.

The rulebook directory, <*universal_agent_home*>/rulebook, resides on the machine where the Universal Agent is installed.

The modes operate as follows:

- **Controller mode:** The Universal Agent first tries to get the rulebook from the Controller. If successful, the Universal Agent writes the contents of the controller rulebook to its rulebook directory using the filename `controller-book.json`. If the Universal Agent can not get the rulebook from the Controller, it looks at the rulebook directory to see if an older copy of `controller-book.json` exists there. If a copy exists locally, it is used as the rulebook. Otherwise, the Universal Agent uses the `default.json` rulebook file from the rulebook directory; this is also the rulebook used if the Universal Agent can't connect to the Controller.
- **Local mode:** The Universal Agent reads a file named `local.json` from the rulebook directory and uses that file as the rulebook. Any rulebook from the Controller is ignored.

  Local rulebooks are also located in the rulebook directory:

  ```
  <universal_agent_home>/rulebook
  ```

  You can put the Universal Agent into local mode using the Universal Agent CLI. You can tell if the Universal Agent is operating in local mode by the presence of a `local.json` file in the rulebook directory. If present, the Universal Agent is in local mode; otherwise, it is in Controller mode.

You create and manage controller rulebooks using the Universal Agent REST APIs. See the section "*Create or Update a Rulebook*".

Polling Interval

The Universal Agent reads the rulebook at regular intervals and applies changes in the rulebook as they occur, reporting the event to the Controller. By default, the Universal Agent checks the rulebook every 300 seconds (5 minutes).

For testing and initial investigatory work, you may want to reduce this interval to induce more frequent polling. Use the **interval** property to change the polling frequency.

Rulebook Structure

The basic parts of a rulebook are shown in the following example:

```
{ "name": "default",
 "comments": "Installs and starts AppDynamics agents",
 "config": {
  "version": "4.3.0.0",
  "controller_host": "192.168.99.100",
  "controller_port": "8080",
  "account_name": "customer1",
  "account_access_key": "c61f32-nnnn"
 },
 "rules": [{
      "name": "Machine Agent Rule",
      "comments": "Install and start the machine agent",
      "monitor": "machine",
      "condition": "True",
      "config": {
    "state": "started"
      }
  },
  {
      "name": "Java Agent Install Rule 1",
      "comments": "Inject the Java Agent as JVMs are started",
      "monitor": "java",
      "config": {
    "state": "installed",
    "application_name": "MyApp",
    "tier_name": "MyTier",
    "node_name": "MyNode"
      }
  },
  {
      "name": "Java Agent Install Rule 2",
      "comments": "Attach the Java Agent to running JVMs",
      "monitor": "java",
      "config": {
    "state": "attached",
    "application_name": "MyApp2",
    "tier_name": "MyTier2",
    "node_name": "MyNode2"
      }
  },
{
      "name": "Universal Agent rule",
      "comments": "Universal Agent Default Rule",
      "monitor": "universal",
      "config": {
    "version": "4.3.0.0",
    "state": "started"
      },
      "condition": "True"
  }
 ]
}
```

The example rulebook contains three rules that define three runtime agents, two Java app agents and one Machine Agent. The example shows the following fields:

- **name**: Name for this rulebook.
- **comments**: An optional description for this rulebook.
- **config**: Runtime agent settings that apply to all runtime agents in the rulebook. The config notably contains connection settings for the Controller. See Controller Connection Fields for more information.
- **condition**: An expression that must evaluate to true on the Universal Agent host for the rule to apply.
- **rules**: Rules contain the specific rules for controlling one or more agents. There are common fields in the agent rules and fields that are specific to the agent type. See Runtime Agent Rule Fields for more information.

The following sections provide more information on the **config** and **rules** fields.

Controller Connection Fields

The rulebook example above describes settings that runtime agents use to connect to the Controller and how the agent instance is identified in the Controller UI. These settings correspond to values typically configurable for the runtime agents, particularly for the Java app agent configuration file, `controller-info.xml`.

With the Universal Agent, you can set any of the usual properties found in the runtime agent configuration file. You refer to the properties in the Universal Agent configuration files with the identical names contained in `controller-info.xml`, *except* that the hyphen in the name is replaced by an underscore.

Specifically, the rulebook supports the following properties:

- account_access_key
- account_name
- agent_runtime_dir
- application_name
- controller_host
- controller_port
- controller_ssl_enabled
- credential_store_filename
- credential_store_password
- enable_orchestration
- force_agent_registration
- machine_path
- node_name
- tier_name
- use_encrypted_credentials
- use_simple_hostname

For more information on their use, see the `controller-info.xml` inline comments and Administer App Server Agents.

When you set the connection values in the rulebook, the Universal Agent updates the controller-info.xml on disk for the runtime agent. The following values are taken from universalagent.yaml if they are not specified in the rulebook:

- controller_host
- controller_port
- controller_ssl_enabled,
- account_name,
- and account_access_key

Specifying the values in the rulebook, however, enables you to configure the Universal Agent and its configured runtime agents to talk to different Controllers, if needed.

Runtime Agent Rule Fields

The rules section in the Universal Agent rulebook contains rules governing the presence and status of a runtime agent on the Universal Agent host. The default rulebook, `default.json`, is installed with the following predefined machine agent rule:

```
"rules": [
    {
        "name": "Universal Agent rule",
        "comments": "Universal Agent rule",
        "monitor": "universal",
        "config": {
            "version": "4.3.0.0",
            "state": "started"
        },
        "condition": "True"
   }
]
```

The fields for each rule are:

- **name**: Name for this rule.
- **comments**: An optional description for this rule.
- **monitor**: The runtime agent type. Valid values are:
    - `java` for the Java App Agent
    - `machine` for the Machine Agent

- **condition**: A statement that must be true to be applied by a particular Universal Agent. Typically a condition tests an environment variable or system property on the host on which the Universal Agent runs.
- **config**: Agent-level configuration settings such as operating state, node name, and so on. These can be specified per runtime agent in the rule (as illustrated by the state and application_name fields in the sample) or globally (as illustrated by the version field). Global fields enable you to avoid repeating the field in each rule. Any value specified in the rule overrides the global value. The config fields include:
    - **version**: The AppDynamics version of the app agent to use
    - **state**: The action to be applied to the agent, such as installed, which simply installs the agent, or started, which both installs and starts it. For agent specific state information, see Java Agent Rules and Standalone Machine Agent Rules.
    - **application_name**, **tier_name**, and **node_name**: These are equivalent settings to the application-name, tier-name and node-name settings in controller-info.xml for the traditional agent configuration. They specify the business application, tier, and node by which the current monitored process is identified in the Controller UI. For additional connection related fields, see Controller Connection Fields.

Rulebook Strategies

Using Groups

Universal Agent groups are a way to manage multiple Universal Agents as a logical group. By default, Universal Agents are part of the default group and run the default rulebook, `default-controller`. As additional Universal Agents start and register with the controller, they join the default group.

You can create your own groups and add Universal Agents to them using the Universal Agent REST API.

A Universal Agent can be part of multiple groups. When the Universal Agent is added to the groups and those groups have different rulebooks, the controller sends multiple rulebooks to the Universal Agent and the rulebooks are logically merged into a single rulebook.

The resulting merged rulebook is written to controller-book.json. You should not attempt to modify this rulebook directly.

Conditional Rules

You can use conditions to specify criteria for applying rules. In the simplest (and default) case, the value can simply be set to "true" to enable the rule. You can also create test conditions that enable or disable the rule based on the environment in the Universal Agent host.

Condition field syntax in Universal Agent rulebooks is similar to conditions in Python—the condition should evaluate to a Boolean expression with identification data keys as the operands (which should match the regular expression "[a-zA-Z0-9_]+").

To install only on Linux machines and given the "platform_system" environment variable, you could specify the following condition:

```
"condition": "platform_system == '\''Linux'\''"
```

Notice that single quotes in the value are escaped.

As another example, the following condition checks for Linux and a version of the Universal Agent higher than 4.3:

```
"condition": "platform_system == '\''Linux'\'' and universalagent_version > '\''4.3.1'\''"
```

Operators you can use are:

- Logical operators: and, or, not
- Comparison operators: ==, !=, <, <=, >, >=
- Special operators:
    - Regular expression compare, where regular expression comes after operator
    - ~ (AWK-like regular expression match)
    - !~ (AWK-like regular expression inverse match)
- Membership test: in

Instead of using dynamically evaluated conditions, you can put the static values of true or false in the condition field. When used in this way, the condition field gives you a convenient way to individually enable or disable rules.

Uninstalling a Runtime Agent

To uninstall a runtime agent, remove the rule for it in the rulebook and save the file.

The Universal Agent first stops the runtime agent, if it is running, and then uninstalls the runtime agent from the application or machine and removes the directory for it.