Micro Focus Pulse

Agent Installation and Configuration Guide

Product version: 19.1

Last updated: April 8, 2019

Table of Contents

Chapter 1 **Introduction 5**
About Agents 6
Agent Architecture 6
Agent Communication 6

Chapter 2 **Installing and Licensing Agents 9**
Installing Agents 10
Updating Agents when a Pulse Server Moves. 10
Approving and Licensing Agents 10
Uninstalling Agents 11

Chapter 3 **Configuring Secure Agent Connections 13**
Introduction 14
Configuring Mutual Authentication 14

Chapter 1
Introduction

About Agents 6
Agent Architecture 6
Agent Communication 6

About Agents

Micro Focus Pulse is a web application that enables development teams to continuously examine the health and quality of their changes. Agents are small Java applications that Pulse uses to run chains. Chains are sets of expert plug-ins that run in a sequence of steps.

- Each agent is associated with a primary Pulse server.
- Each Pulse server can have multiple agents connected to it.
- Agents only include the code required to run chains.
- Agents automatically download the required plug-ins from the Pulse server. You do not need to manually install plug-ins on agents.

For more information about chains, experts, and managing agents, see the Pulse help.

Agent Architecture

Agents are an important part of Pulse scalability. By adding more agents, the throughput and capacity of the system increases and scales to fit your needs. Depending on the number of chains defined, Pulse might require a large number of agents to prevent the queue of chains growing too large.

Agents are unobtrusive and secure. Once an installed agent has been started, the agent opens a socket connection to the Pulse server. Communication between server and agents uses a JMS-based (Java Message Service) protocol and can be secured using SSL, with optional mutual key-based authentication for each end-point. The communication protocol is stateless and will recover after many types of network outages.

- The monitor is a service that manages the worker process, for example: starting, stopping, and managing restarts, upgrades, and security.
- The worker is a process that runs the chains after receiving commands from the server.
- Chains are run in their own process using plug-ins to perform the steps and integrate with third-party tools.

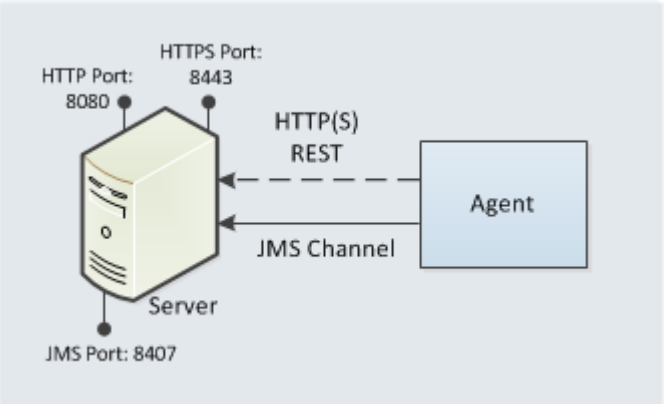
Agent Communication

Agents are triggered using JMS and generally use HTTP to get more detailed information about what they are expected to do or to report detailed results. For example:

- The server uses JMS to send agent commands.
- The agent monitor service uses JMS for all server communications and for sending commands, such as `run step` to the worker process.
- The worker process uses JMS for system communications and HTTP REST services when performing plug-in steps or retrieving information from the server.

- Agent activities such as posting logs, transmitting chain results, and posting files to Pulse use the web tier through HTTP or HTTPS.
- Most clients use browsers to communicate with the web server through HTTP or HTTPS.

Stateless server-agent communication provides significant benefits to performance, security, availability, and recovery after downtime. Because each agent request is self-contained, a transaction consists of an independent message that can be synchronized to secondary storage as it occurs. The server or agent can be taken down and brought back up with minimal impact, other than lost time. If communications fail mid transaction, no messages are lost. Once reconnected, the server and agent automatically determine which messages got through and what work was successfully completed. After an outage, the system synchronizes the endpoints and recovers affected processes. The results of any work performed by an agent during the outage are communicated to the server. In the diagram below, the arrow represents the direction in which the stateless communication was established, but the flow is in both directions with JMS.



For added security, agents do not listen on ports. Agents send requests when they are ready to make the transition to a new state. Because JMS connections are persistent and not based on a request-response protocol, Pulse does not have to continually open and close ports, which enables the server to communicate with agents at any time while remaining secure and scalable. REST-style services achieve statelessness by ensuring that requests include all the data needed by the server to make a coherent response.

Chapter 2

Installing and Licensing Agents

Installing Agents	10
Updating Agents when a Pulse Server Moves	10
Approving and Licensing Agents	10
Uninstalling Agents	11

Installing Agents

1. Download the Pulse Agent installer from Micro Focus SupportLine. There are installers for Windows and Linux.
2. Run the installer and follow the instructions.

NOTE If you are using registry passwords in chains that run on agents, you must create and configure a password registry file on the agent's host machine. Add the path to the registry in the `config.properties` file, for example:

```
registered.password.file=C:\\registry.dat
```

Updating Agents when a Pulse Server Moves

If a Pulse server moves to another port or machine you must update all the associated agents.

1. On each agent open the `config.properties` file.
2. Modify the property `pulse.base.url`. For example:

   ```
   pulse.base.url=http://newhost.company.com:8080/pulse
   ```

Approving and Licensing Agents

When an agent is associated with an instance of Pulse it is automatically displayed in the Agents administration page. To license an agent, a Pulse administrator can approve it. You can see if an agent is licensed, or if there are licensing issues, on the Agents page.

- When an agent is added to Pulse, by default it is unapproved and cannot be used to run chains. If an agent can be trusted, you can approve it.
- Unapproving or deleting an agent releases the license that it holds.
- Pulse includes three free agent licenses.

1. Log into Pulse as a user with administrator rights.
2. On the top navigation click **Administration** and select **Agents**.
3. Select one or more agents.
4. From the **Mark As** list select **Approved**.
5. To unapprove agents and release licenses, from the **Mark As** list select **Unapproved**.

Uninstalling Agents

1. On the remote machine, shut down the agent.
2. Log into Pulse as a user with administrator rights.
3. On the top navigation click **Administration** and select **Agents**.
4. Select one or more agents and click Delete.

NOTE If you do not shut down the agent, or it restarts, it will reappear in Pulse.

Chapter 3

Configuring Secure Agent Connections

Introduction 14

Configuring Mutual Authentication 14

Introduction

Secure Socket Layer (SSL) technology enables clients and servers to communicate securely by encrypting all communications. Data are encrypted before being sent and decrypted by the recipient so that communications cannot be deciphered or modified by third-parties. Pulse enables a server to optionally communicate with its agents using SSL in mutual authentication mode.

In mutual authentication mode communications are encrypted but users are also required to authenticate themselves by providing digital certificates. A digital certificate is a cryptographically signed document intended to guarantee the identity of the certificate's owner. Pulse certificates are self-signed.

When mutual authentication mode is active, Pulse uses it for JMS-based server communication (via SSL from Pulse to agents) and agent communication (via HTTPS from agents to Pulse). In this mode, the Pulse server provides a digital certificate to each agent and each agent provides one to the server. This mode can be implemented during server/agent installation or activated afterwards.

NOTE When using mutual authentication mode you must turn it on for both server and agents, otherwise they will not be able to connect. If one party uses mutual authentication mode, they must all use it.

IMPORTANT! The server and agent properties must be set prior to configuring mutual authentication and exchanging keys.

Configuring Mutual Authentication

Follow these instructions to configure mutual authentication for Pulse servers and agents.

1. Check that your Pulse server and agents are not running.
2. Run this command on the server to create a new key store and a private key for the server (if they do not exist):

```
${tomcatDir}/conf> keytool -genkey -alias [your-alias] -keyalg RSA
-keystore [your-key-store-file-name]
```

Example with the host name of a Pulse server as the CN distinguished name:

```
keytool -genkey -noprompt -alias pulse -keyalg RSA -keystore
server-ssl.jks - keypass 123456 -storepass 123456 -dname
"CN=[pulse-hostname], OU=pulse, O=Company, L=Location, ST=PU,
C=PU"
```

3 To create a new Connector entry, add the text below to the following file on the server:

`${tomcatDir}/conf/server.xml`

```
<Connector port="8443" scheme="https" secure="true"
SSLEnabled="true" clientAuth="false"
sslEnabledProtocols="TLSv1.2,TLSv1.1,TLSv1" sslProtocol="TLS"
keyAlias="pulse" keystoreFile="C:\Tools\apache-
tomcat\apachetomcat- 8.5.32\conf\server-ssl.jks"
keystorePass="123456" truststoreFile="C:\Tools\apache-
tomcat\apachetomcat- 8.5.32\conf\server-ssl.jks"
truststorePass="123456" ...
</Connector>
<!-- In this example keystore and truststore is the same file, but
in
general they could be separated -->
<!-- if keystore has several aliases, a proper should be chosen. In
this
example "pulse" -->
```

Do not use the same port number that is used in an existing Connector entry.

4 Enable SSL/TLS Configuration in the following file on the server:

`${tomcatDir}/conf/server-ssl.jks`

Add the properties below:

`${pulseDataDir}/startup.properties`

```
pulse.activemq.protocol=ssl
pulse.activemq.port=8447
pulse.activemq.key.store.type=jks
pulse.activemq.key.store.filename=C\:\\Tools\\apache-
tomcat\\apache-tomcat-8.5.32\\conf\\server-ssl.jks
pulse.activemq.key.store.password=123456
pulse.activemq.trust.store.type=jks
pulse.activemq.trust.store.filename=C\:\\Tools\\apache-
tomcat\\apache-tomcat-8.5.32\\conf\\server-ssl.jks
pulse.activemq.trust.store.password=123456
```

5 Run this command on the server to create a server certificate:

```
${tomcatDir}/conf> keytool -export -keystore [your-key-store-file-
name] -alias [your-alias] -file [your-alias].crt
```

Example:

```
${tomcatDir}/conf> keytool -export -keystore server-ssl.jks -alias
pulse -file pulse.crt
```

The following message is displayed:

```
**Certificate stored in file pulse.crt**
```

6 Copy the exported `pulse.crt` certificate file to this directory on the agent:

`${pulseAgentDataDir}/conf`

7 Run this command to create a new key store and a private key for the agent (if they do not exist):

```
${pulseAgentDataDir}/conf> keytool -genkey -alias [agent_alias]
-keyalg RSA -keystore [agent-store-file].jks
```

Example:

```
${pulseAgentDataDir}/conf> keytool -genkey -alias agent -keyalg
RSA -keystore agent.jks -noprompt -keypass 123456 -storepass
123456 -dname "CN=[pulse-hostname], OU=agent, O=Company,
L=Location, ST=PU, C=PU"`
```

8 Open this file on the agent:

`${pulseAgentDataDir}/config.properties`

For the embedded agent in a Pulse server, open:

`${pulseDataDir}/startup.properties`

Add these properties:

```
agent.auth.mode=MUTUAL
# For mutual authentication between pulse and agents, key store and
trust store should be configured:
agent.auth.key.store.type=jks
agent.auth.key.store.filename=C\:\\ProgramData\\Micro
Focus\\Dimensions
CM\\PulseAgent\\conf\\agent.jks
agent.auth.key.store.password=123456
agent.auth.trust.store.type=jks
agent.auth.trust.store.filename=C\:\\ProgramData\\Micro
Focus\\Dimensions
CM\\PulseAgent\\conf\\agent.jks
agent.auth.trust.store.password=123456
# Location of the Pulse application.
pulse.base.url=https://[pulse-hostname]:8443/pulse
```

9 Run this command in the agent's `conf` directory to import `pulse.crt`:

```
${pulseAgentDataDir}/conf> keytool -import -keystore agent.jks
-alias pulse -file pulse.crt
```

The following message is displayed:

`**Certificate was added to keystore**`

10 Run this command in the agent's `conf` directory to export the agent key as a certificate:

```
${pulseAgentDataDir}/conf> keytool -export -keystore agent.jks
-alias [agent_alias] -file [agent_alias].crt
```

Example:

```
${pulseAgentDataDir}/conf> keytool -export -keystore agent.jks
-alias agent -file agent.crt
```

The certificate is stored in this file: `[agent_alias].crt`

Before you export an agent key, you must first generate it.

11 Copy the exported certificate file `[agent_alias].crt` to this directory on the server:

`${tomcatDir}/conf`

12 Run this command on the server in `${tomcatDir}/conf` to import the file `[agent_alias].crt`:

```
keytool -import -keystore server-ssl.jks -alias [agent_alias]
-file [agent_alias].crt
```

The following message is displayed:

`**Certificate was added to keystore**`

13 For additional agents, repeat from step 6.

14 Start the Pulse server and agents.