



# Micro Focus PulseUno Agent Guide

Copyright © 2014–2021 Micro Focus or one of its affiliates.

The only warranties for products and services of Micro Focus and its affiliates and licensors ("Micro Focus") are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Micro Focus shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Contains Confidential Information. Except as specifically indicated otherwise, a valid license is required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Product version: 21.1

Last updated: July 26, 2021

# Table of Contents

---

<i>Chapter 1</i>	<b>Introduction</b> . . . . .	<b>5</b>
	About Agents . . . . .	6
	Agent Architecture . . . . .	6
	Agent Communication . . . . .	6
<i>Chapter 2</i>	<b>Installing and Licensing Agents</b> . . . . .	<b>9</b>
	Installing Agents . . . . .	10
	Updating Agents if a PulseUno Server Moves. . . . .	10
	Approving Agents . . . . .	10
	Licensing Agents . . . . .	11
	Configuring Dimensions CM Licensing . . . . .	11
	Configuring StarTeam Licensing . . . . .	11
	Configuring AutoPass Licensing . . . . .	11
	Uninstalling Agents . . . . .	12
<i>Chapter 3</i>	<b>Configuring Secure Agent Connections</b> . . . . .	<b>13</b>
	Introduction . . . . .	14
	Configuring Mutual Authentication . . . . .	14



# Chapter 1

---

## Introduction

About Agents	6
Agent Architecture	6
Agent Communication	6

---

## About Agents

Agents are small Java applications that PulseUno uses to run chains. Chains are sets of expert plugins that run in a sequence of steps.

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

For more information about chains, experts, and managing agents, see the [PulseUno help](#).

## Agent Architecture

Agents are an important part of PulseUno 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, PulseUno might require a large number of agents to prevent the queue of chains growing too large.

Agents are unobtrusive and secure. After an installed agent has been started, the agent opens a socket connection to the PulseUno 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 plugins to perform the steps and integrate with third-party tools.

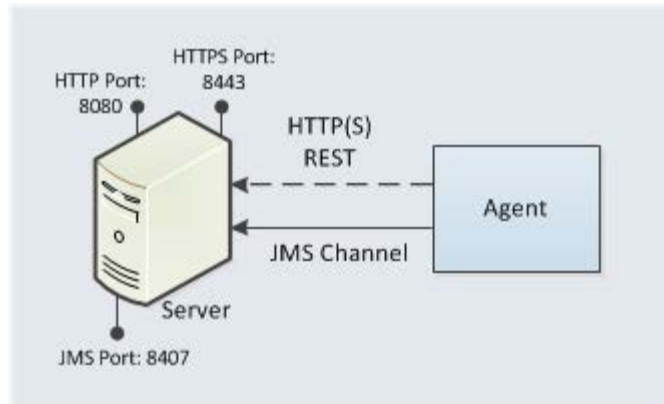
## Agent Communication

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 plugin steps or retrieving information from the server.
- Agent activities such as posting logs, transmitting chain results, and posting files to PulseUno 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.



**IMPORTANT!** To ensure that servers and agents communicate without issues, the clocks on servers and agents must be synchronized.

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, PulseUno 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 if a PulseUno Server Moves	10
Approving Agents	10
Licensing Agents	11
Uninstalling Agents	12

---

## Installing Agents

- 1 Download the PulseUno Agent installer. For details, contact [Micro Focus Support](#). There are installers for Windows and Linux.

- 2 Run the installer and follow the instructions.

**NOTE** (Dimensions CM only) 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 if a PulseUno Server Moves

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

**To update an agent:**

- 1 On an agent, open the `config.properties` file.

- 2 Modify the property `pulse.base.url`. For example:

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

- 3 If you installed the agent as a service, restart the service. Otherwise, shut down the agent using the agent control script and the `shutdown` parameter, and then start the agent.

## Approving Agents

When an agent is associated with an instance of PulseUno it is automatically displayed in the Agents administration page. To license an agent, a PulseUno 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 PulseUno, it is unapproved by default 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.

**To approve or unapprove agents:**

- 1 Log in to PulseUno as a user with administrator rights.

- 2 Navigate to **Administration > 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**.

---

# Licensing Agents

PulseUno includes three free agents. To add agents, you need:

- A configured license server.
- A license file installed on the license server with sufficient capacity for the agents you have approved (in addition to free agents).

The license server configuration is in this PulseUno properties file:

```
$PULSE_DATA/conf/startup.properties
```

where \$PULSE\_DATA is the directory where PulseUno stores its configuration data.

## Configuring Dimensions CM Licensing

The following properties specify the location of a Dimensions CM server:

```
license.type=dcm
```

```
license.dcm.server=<hostname:port>
```

- The Dimensions CM server must be configured with a license server.
- PulseUno asks the Dimensions CM server to check out and check in licenses.
- This configuration is performed by the installer. You only need to change it if you modify the network topology of your installation.

## Configuring StarTeam Licensing

The following properties specify the location of an AutoPass License Server (APLS):

```
license.type=autopass
```

```
license.autopass.server=<APLS URL>
```

- The APLS can be the same one used for your StarTeam server.
- You only need to add this configuration if you have a PulseUno agent license file deployed in the APLS and you require more licenses (in addition to the free licenses).

## Configuring AutoPass Licensing

For details, see the [PulseUno online help](#).

---

# Uninstalling Agents

## To uninstall agents:

- 1 On the remote machine, shut down the agents that you want to uninstall.
- 2 Log in to PulseUno as a user with administrator rights.
- 3 Navigate to **Administration > Agents**.
- 4 Select one or more agents and click **Delete**.

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

## Chapter 3

---

# Configuring Secure Agent Connections

Introduction	14
Configuring Mutual Authentication	14

---

---

# Introduction

**NOTE** This feature is supported only by Dimensions CM.

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. PulseUno 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. PulseUno certificates are self-signed.

When mutual authentication mode is active, PulseUno uses it for JMS-based server communication (via SSL from PulseUno to agents) and agent communication (via HTTPS from agents to PulseUno). In this mode, the PulseUno 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 PulseUno servers and agents.

- 1 Check that your PulseUno 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 PulseUno 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 PulseUno 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 PulseUno 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 PulseUno server and agents.

If you installed the agent as a service, restart the service. Otherwise, shut down the agent using the agent control script and the shutdown parameter, and then start the agent.

