Presto 2.5

Print Provider Guide

Table of Contents

Overview

The Presto system allows administrators to plugin custom print providers to augment Presto’s built-in support for discovering and printing to the system's print queues.

This facility allows the system administrator to advertise virtual print queues to Presto users.

Supported Platforms

Windows:	Vista 7, 8, 10; Server 2008 R2; Server 2012
Mac:	OS X 10.11 +
Linux:	Ubuntu 14.04 +

Introduction

Presto Print Provider plugins run as scripts or programs that output their results to `stdout` as JSON formatted text that Presto Server components read.

Presto Print Providers are managed by the Presto Agent component. As such, any custom should be added to the `agent.conf` file located at:

Windows

```
C:\ProgramData\Collobos\Presto\agent.conf
```

Mac

```
/Library/Application Support/Collobos/Presto/agent.conf
```

This file is formatted as JSON text. It describes additional configuration to the agent which it then advertises to the Presto system.

Usage

Let's walk through the steps to integrate an additional print provider into the Presto system.

First, we will edit the agent.conf file and add our custom print provider. After we're done editing, the file should look like the one in Figure 1.

Figure 1: Example Declaration for Custom Directory Provider

After editing the `agent.conf` file and saving the changes to it, Presto Agent will automatically re-read the file and add the new print provider to the Presto system.

The Presto system will add all printers to the system that are reported by the `lookup_printers` command.

If a user prints to a printer that has been discovered via a print plugin, the plugin will be invoked both to start a print job via the `start_job` command, and possibly to cancel a print job via the `cancel_job` command.

Output

The output of the authentication command is also JSON formatted text. In this example, our plugin is asked to authenticate the user *jsmith@example.com*.

If the authentication succeeded, the plugin will send the following output to stdout as in Figure 2, and return a 0 exit code.

Figure 2: A properly formatted user

If the authentication fails, do not print to `stdout` and return a non-zero exit code from your plugin.

Please note that your plugin can use the method it needs to use to do the query. However, it must return the result as JSON formatted text.

Example

Let's work through an actual implementation of the `authenticate_presto_user.bash` script to see how it might be implemented.

Our script will check the username and password. It will only accept the user *`johndoe`* with password *`x`*.

bash

agent.conf
script.bash

This script will only authenticate a user if the name is *`johndoe`* and the password is *`x`*.

Presto can invoke any executable as the authentication script. In the preceding example, we implemented our directory provider as a bash script, but we could have implemented it as a DOS or Powershell script (if running on Windows), as a node.js script, or as a C/C++/Java program. Anything that can be executed from the command line can be used as a directory provider script.

The following examples show different versions of the bash script above for node.js, DOS, and Powershell.

node

agent.conf
script.js

dos

agent.conf
script.bat

powershell

agent.conf
script.ps1

Groups

You are encouraged to add groups to the users emitted by your directory provider plugin. It is quite easy to add to the JSON output. Groups are added as strings in the tags field.
This example associates the groups "**teachers**", "**nice_guys**", and "**managers**" to the user "John Smith".

When Presto receives this, it will turn these into tags that can be used when writing rules. Presto will insert the string "**group:**" to the strings that are output from directory providers.

So in this example, after this user has been added to the system, you would see tags like this in the rule editor:

group:teachers
group:nice_guys
group:managers