IBM Aspera Orchestrator 4.0.1

IBM

Contents

Introduction

IBM Aspera Orchestrator automates the collection, processing, and distribution of a large volume of file-based digital assets.

Orchestrator's run-time engine supports conditional decision-making, manual user inputs, automated high-speed file movement, and third-party system integrations.

Orchestrator uses FASP transport technology for all high-speed file transfers and is fully integrated with all Aspera transfer mechanisms.

Orchestrator includes the following features:

- Job queues that allow reordering of the execution priority of multiple staged jobs.
- A robust library of plugins for functions such as transcoding, quality control, antivirus, ad insertion, digital fingerprinting, and other solutions.
- REST API for management, monitoring, and control from within third-party applications.
- Automatic restart on transmission failures.
- Support for parallel execution of all actions (for example, parallel video encoding into multiple formats).
- A drag-and-drop, browser-based interface that allows the user to graphically compose reusable workflow sequences of inputs, actions, and outputs.

Single Node Installation

This section guides users through all procedures related to installing Orchestrator on one server.

Requirements

These are the minimum requirements for installing Orchestrator.

For information on supported operating systems and browsers, see the release notes for this version of IBM Aspera Orchestrator.

Additional requirements include:

- An Aspera Orchestrator license file.
- For a single-server installation on a production-grade system, a physical server or a virtual machine with the following minimum capacity:
  - 8 CPUs (each with one or more cores).
  - 16 GB of RAM.
  - 100 GB of free disk space.

  For a development/test system, 2 to 4 core CPUs and 4GB of RAM is sufficient.
- Installing these dependencies *before* beginning the installation process for Orchestrator:
  - perl
  - libX11
  - libXext
  - libXft
  - libXi
  - libXtst
  - libaio

Aspera recommends that Orchestrator be installed on its own server. However, Orchestrator can be co-located with other Aspera web applications. Contact Support for instructions on how to install Orchestrator in such an environment.

Aspera *advises against* combining IBM Aspera management systems like Orchestrator with IBM Aspera transfer servers and web application servers (for example, Connect Server or Faspex).

Installing Orchestrator with MySQL on the Same Server

This procedure installs Orchestrator and the MySQL database on the same server. If you need to install Orchestrator on a remote server, use the procedure in "Installing Orchestrator with MySQL on a Remote Server" on page 6.

Overview of the Installation Process:

1. "Before You Begin" on page 2
2. "Running the Installers" on page 2
3. "Completing the Installation in the UI" on page 6

Note: After you install Orchestrator, you can find all installation log information in the directory `/opt/aspera/orchestrator`, in a file with a name similar to the following: `install_3657_20170620104842.log`. The number string in the filename will change each time you reinstall the product.

Before You Begin

Before beginning the installation process, you must be logged in as an admin.

1. Review the system requirements section of the release notes.
2. Confirm that you have the following:
   - Your organization's login credentials (username and password). You will need the credentials to download the installer from the Aspera Downloads page.
   - The license file for Orchestrator. You need this for the final setup procedure in the Orchestrator UI.

     Note: For purchasers of Aspera Enterprise, a license enabling Orchestrator as part of Enterprise can be downloaded from IBM Fix Central.

   If you don't have these, contact your Aspera account manager.
3. Install the required dependencies:
   - perl
   - libX11
   - libXext
   - libXft
   - libXi
   - libXtst
4. On the machine where you will install Orchestrator, download the installation files for the current release.

   Go to IBM Fix Central, click the link for the current release of Orchestrator, then download the installers for `aspera_orchestrator` and `ibm_aspera_common`.

Running the Installers

1. Install the Aspera Common Components.

   Go to the location where the installation files are saved and run the following command as `root`:

```
# yum install aspera-common-version-0.x86_64.rpm
Preparing...                          ########################################### [100%]
```

```
Updating / installing...
   1:aspera-common-version-0    ################################## [100%]
```

Note: If you need to install MySQL on a separate machine, stop here and use the procedure in "Installing Orchestrator with MySQL on a Remote Server" on page 6.

2. Install Orchestrator.

```
 # yum install aspera-orchestrator-version-0x86_64.rpm
Preparing...######################################### [100%]
. . .
To complete the setup of Orchestrator, run "asctl orchestrator:setup"
```

3. Add the instances' IP addresses and hostnames that will access the Orchestrator.

```
What IP addresses and hostnames are allowed to access Orchestrator?
```

Note: You should at least include one accepted host's IP address or hostname during the setup.

You can update the accepted hosts by going to (UI) `Engine > Configuration > accepted_hosts`. If you need to make further changes after the installation, you can do so by modifying the following file `/opt/aspera/var/config/orchestrator/orchestrator.yml`. When including accepted hosts (either through the UI, during setup, or by modifying the YML file) make sure they are separated by a comma (`,`).

4. When installation completes, run the following command:

```
# asctl orchestrator:setup
```

5. When the setup prompt appears, enter "s" for "streamlined" setup or "d" for "detailed" setup.

```
Streamlined or detailed setup (s/d)? (current: s)
```

Selecting detailed setup allows you to customize services and processes in Orchestrator. You may need this setup if you require an advanced configuration; for example, if your system utilizes non-standard ports or not all components are running on a single server.

6. **Streamlined Setup**: If you selected streamlined setup in the previous step, the following prompts appear.

   • Refer to the table for appropriate responses:

Prompt	Response	Default Value
`Enter a new MySQL root password.`	Enter a password. **Note:** Save this for future reference.	
`MySQL will need to start/restart during configuration. Continue (y/n)?`	Enter "y" to proceed with the setup process. If you enter "n", the system will exit setup.	"y"
`What hostname or IP address should Apache use to identify itself (in the SSL Certificate)?`	Enter the hostname or IP address for Apache.	`10.0.174.44, site.com`
`What IP addresses and hostnames are allowed to access Orchestrator?`	Enter the IP address and/or hostname of the instances that are allowed to access the Orchestrator.	`10.0.174.44, site.com` **Note:** These are not default values, but rather examples of the format that the

Prompt	Response	Default Value
		IP address and hostname need to have to function correctly.

7. **Detailed Setup**: If you selected detailed setup in the previous step, the following prompts appear:

Prompt	Details	Default
`What base port should the Mongrel servers start at?`	Enter the base port for the Orchestrator mongrel servers.	`3000`
`How many Mongrel process should be launched?`	Enter the number of mongrel processes; this is related to server capacity.	`3`
`What web root do you want to use?`	Enter the web root.	`/aspera/orchestrator`
`MySQL will run on this machine (y/n)?`	• To run MySQL on this machine, enter "y".	`"y"`
`What port would you like MySQL to listen on?`	If you entered "y" for the prompt above, enter the MySQL port.	`4406`
`Enter a new MySQL root password.`	Enter the MySQL root password. **Note:** Be sure to save this for later use.	none
`MySQL will need to start/restart during configuration. Continue (y/n)?`	Enter "y" to continue with the configuration process. If you enter "n", the system will exit setup.	`"y"`
`Enter IP address of network interface for Apache to listen on.`	Enter the hostname or IP address of your Orchestrator server.	`0.0.0.0`
`What hostname or IP address should Apache use to identify itself (in the SSL Certificate)?`	Enter the hostname or IP address for Apache.	`10.0.174.44, site.com`
`What port would you like to run Apache http on?`	Enter port for Apache HTTP.	`80`
`What port would you like to run Apache https on?`	Enter port for Apache HTTPS.	`443`
`Would you like to generate a self-signed SSL certificate, or install your own ([g]enerate,[c]opy) (current: g)?`	• To generate a self-signed SSL certificate, enter "g" • To install your own, enter "c" **Note:** The key and certificate will be generated in the following directory: `/opt/aspera/common/apache/conf`.	`generate (g)`

Prompt	Details	Default
`What IP addresses and hostnames are allowed to access Orchestrator?`	Enter the IP address and/or hostname of the instances that are allowed to access the Orchestrator.	`10.0.174.44, site.com` **Note:** These are not default values, but rather examples of the format that the IP address and hostname need to have to function correctly.

8. You will get a prompt to confirm your settings. If the settings are correct, enter "y" to accept them and continue; otherwise, enter "n" to re-enter the settings. (Pressing "x" will exit the installation process.
9. At the prompt, enter "y" to restart Apache. Entering "n" will exit setup.
10. At the prompt, enter "y" to restart MySQL. Entering "n" will exit setup.

    When the setup process is complete, a message similar to the following appears:

    ```
    Reminders:
     - Apache needs ports 80 and 443 open in firewalls.
    ```

    Make a note of these port numbers for future reference.
11. To start the Orchestrator processes, run the following command:

    ```
     asctl orchestrator:start
    ```

    Depending on your system, you may see the status `NOT running` in the system response:

    ```
    Orchestrator Status:
      -> Orchestrator Manager running with pid: 3447
      -> Orchestrator Engine NOT running
      -> Mongrel serving orchestrator on port 5000 is NOT running
      -> Mongrel serving orchestrator on port 5001 is NOT running
      -> Mongrel serving orchestrator on port 5002 is NOT running
      -> Mongrel serving orchestrator on port 5003 is NOT running
      -> Mongrel serving orchestrator on port 5004 is NOT running
      -> Orchestrator Monitor NOT running
      -> Asynchronous Worker Process 0 is NOT running
      -> Asynchronous Worker Process 1 is NOT running
      -> Synchronous Worker Process 2 is NOT running
      -> Synchronous Worker Process 3 is NOT running
      -> Synchronous Worker Process 4 is NOT running
      -> Synchronous Worker Process 5 is NOT running
    ```

    This is an indication that the Orchestrator processes need additional time to start. Wait approximately one minute, then check the status by entering the following command:

    ```
    # asctl all:status
    ```

    You should now see confirmation that Apache and MySQL are running, and a status message similar to the following that confirms the processes are now running:

    ```
    Apache:               running
    MySQL:                running

    Orchestrator Status:
      -> Orchestrator Manager running with pid: 3447
      -> Orchestrator Engine NOT running
      -> Mongrel serving orchestrator on port 5000 is running with pid: 3465
      -> Mongrel serving orchestrator on port 5001 is running with pid: 3467
      -> Mongrel serving orchestrator on port 5002 is running with pid: 3469
      -> Mongrel serving orchestrator on port 5003 is running with pid: 3471
      -> Mongrel serving orchestrator on port 5004 is running with pid: 3473
      -> Orchestrator Monitor running with pid: 3463
      -> Asynchronous Worker Process 0 is running with pid: 3480
    ```

```
  -> Asynchronous Worker Process 1 is running with pid: 3482
  -> Synchronous Worker Process 2 is running with pid: 3484
  -> Synchronous Worker Process 3 is running with pid: 3486
  -> Synchronous Worker Process 4 is running with pid: 3488
  -> Synchronous Worker Process 5 is running with pid: 3478
```

Completing the Installation in the UI

Note: If you are performing an upgrade of Orchestrator, skip this section.

1. Open the Orchestrator user interface.

   In a web browser, go to the following URL, where *orchestrator_ip_address* is the IP address or fully qualified name of the machine on which Orchestrator is installed.

   ```
   https://orchestrator_ip_address
   ```

   Note: Depending on your current certificates, you may get a security warning. If so, click **Advanced** and add a security exception for the Orchestrator website URL.
2. In the Login field, enter `admin`; leave the Password field blank.
3. Create a password for the admin user.

   In the Change Password dialog, leave the **Current password** field blank and enter your new password. Enter it again in the **Confirm password** field.
4. Apply the license file.
   a) In the Update License dialog, select a license file to upload, or paste the contents of your license file into the text box.
   b) Click **Update and validate license**.

      The message `New license applied and validated successfully` appears in the dialog.

      Note: The Orchestrator UI displays the currently installed version and license expiration date in the footer of every page.
5. Confirm that all processes are running.

   Click **Engine** > **Processes** and review the Status column to confirm that all processes are running. For any process with a Stopped status, click **Start** to resume.
6. If you need the system user, you must activate that account manually and set a password.

   In the top navigation menu, click **Users**. To the right of the **system** entry, click the "More" (. . .) icon , then click **Activate**.

   Click the "More" icon again, then click **Change password**

   Note: As a security measure, the `system` user is not assigned a password automatically after installation of Orchestrator.

   On the **Change password for system** page, enter your password in the **Current password for admin** field, and enter and confirm a new system user password. Click **Change** to confirm.
7. Now that you've installed Orchestrator, configure the product. See "Product Configuration" on page 55 for details.

Installing Orchestrator with MySQL on a Remote Server

Use this procedure for installing Orchestrator if you host the MySQL database on a remote server.

Overview of the Installation Process:

1. "Before You Begin" on page 7
2. "Configuring MySQL on the Remote Server" on page 7
3. "Running the Installers" on page 9
4. "Completing the Installation in the UI" on page 11

Note: After you install Orchestrator, you can find all installation log information in the directory `/opt/aspera/orchestrator`, in a file with a name similar to the following: `install_3657_20170620104842.log`. The number string in the filename will change each time you reinstall the product.

Note: You can configure two or more Orchestrator instances to share the same MySQL machine; however, both Orchestrator instances must be the same version number.

Before You Begin

Before beginning the installation process:

- You are logged in as an admin.
- You have set up a MySQL database on a server that is separate from the machine where you plan to install Orchestrator.

  Note: You must use the version of MySQL that is bundled with IBM Aspera Common Components (see step 4).

1. Review the system requirements section of the release notes.
2. Confirm that you have the following:
   - Your organization's login credentials (username and password). You will need the credentials to download the installer from the Aspera Downloads page.
   - The license file for Orchestrator. You need this for the final setup procedure in the Orchestrator UI.

     Note: For purchasers of Aspera Enterprise, a license enabling Orchestrator as part of Enterprise can be downloaded from IBM Fix Central.

   If you don't have these, contact your Aspera account manager.
3. Install the required dependencies:
   - perl
   - libX11
   - libXext
   - libXft
   - libXi
   - libXtst
4. Go to IBM Fix Central and click the link for the current release of Orchestrator.
   - On the server where you plan to install Orchestrator, download the installers for `aspera_orchestrator` and `ibm_aspera_common`.
   - On the server where you plan to host MySQL, download the installer for `ibm_aspera_common`.

Configuring MySQL on the Remote Server

1. On the server where you plan to host MySQL, install the Aspera Common Components.

   Run the following command as `root`:

   ```
   # yum install ibm-aspera-common-version-0.x86_64.rpm
   ```
2. Initiate setup of the MySQL database.

   Run the following command to initiate the configuration process:

   ```
   # asctl mysql:setup
   ```
3. Select a setup mode.

   ```
   Streamlined: A simple setup with all components running on this computer.
   Detailed:    An advanced configuration (e.g. non-standard ports or not all components
   running on a single server)
   ```

```
Streamlined or detailed setup (s/d)? (current: s) d
```

- Enter "s" to select *streamlined* setup.
- Enter "d" (*detailed*) if you require an advanced configuration; for example, your system utilizes non-standard ports or not all components are running on a single server.

The following table describes the system prompts for the two setup modes:

System Prompt	**Details**	**Mode (s or d)**
MySQL will run on this machine (y/n)?	Default is "y"	d
What port would you like MySQL to listen on?	Default is 4406	d
Enter a new MySQL root password.	Enter and confirm the MySQL `root` password. **Note:** Save this for future reference.	s, d
MySQL will need to start/restart during configuration. Continue (y/n)?	Default is "y" **Note:** If you enter "n", the system will exit setup.	s, d
`What IP addresses and hostnames are allowed to access Orchestrator?`	Enter the IP address and/or hostname of the instances that are allowed to access the Orchestrator.	`10.0.174.44, site.com` **Note:** These are not default values, but rather examples of the format that the IP address and hostname need to have to function correctly.

4. Enter "y" to confirm the settings.

   For example:

```
==================== Settings ====================
MySQL
  Enabled:     true
  Port:        4406

Are these settings correct? (y/n/x with x for exit) y
```

   MySQL starts.

5. Enter "y" to restart MySQL.

```
MySQL needs to be restarted, restart it now (y/n)? (default:y) y
```

6. Log in to the MySQL database with the `root` password you entered in step 3.

```
/opt/aspera/common/mysql/bin/mysql -h ip_of_mysql_server -P port_number -uroot -p
```

   For example:

```
/opt/aspera/common/mysql/bin/mysql -h 10.0.174.82 -P 4406 -uroot -p
```

7. To create the `orchestrator` database, enter the following:

```
create database orchestrator;
```

8. To allow Orchestrator to access the remote MySQL database, enter the following command.

In the placeholder *host*, enter the IP address or hostname of the server running Orchestrator:

```
asctl mysql:grant_remote_access "host"
```

For example:

```
asctl mysql:grant_remote_access "10.0.115.100"
```

If you are configuring a high availability environment, repeat this command with the *host* for the second Orchestrator server.

Running the Installers

On the server where you need to install Orchestrator:

1. Install the Aspera Common Components.

   Go to the location where the installation files are saved and run the following command as `root`:

   ```
   # yum install aspera-common-version-0.x86_64.rpm
   Preparing...                          ########################################### [100%]
   Updating / installing...
      1:aspera-common-version-0    ########################################### [100%]
   ```

2. Run the Orchestrator installer.

   ```
    # yum install aspera-orchestrator-version-0x86_64.rpm
   Preparing...########################################### [100%]
   . . .
   To complete the setup of Orchestrator, run "asctl orchestrator:setup"
   ```

3. When prompted to complete the Orchestrator setup, run the following command:

   ```
   # asctl orchestrator:setup
   ```

4. Add the instances' IP addresses and hostnames that will access the Orchestrator.

   ```
   What IP addresses and hostnames are allowed to access Orchestrator?
   ```

   Note: You should at least include one accepted host's IP address or hostname during the setup.

   You can update the accepted hosts by going to (UI) `Engine > Configuration > accepted_hosts`. If you need to make further changes after the installation, you can do so by modifying the following file `/opt/aspera/var/config/orchestrator/orchestrator.yml`. When including accepted hosts (either through the UI, during setup, or by modifying the YML file) make sure they are separated by a comma (`,`).

5. When the setup prompt appears, enter "d" for detailed setup.

   ```
   Streamlined or detailed setup (s/d)? (current: s)
   ```

   Selecting detailed setup allows you to customize services and processes in Orchestrator, including MySQL installed on the remote server.

6. Refer to the table for appropriate responses to the system prompts:

Prompt	**Details**	**Default**
`What base port should the Mongrel servers start at?`	Enter the base port for the mongrel servers.	`3000`
`How many Mongrel process should be launched?`	Enter the number of mongrel processes; this is related to server capacity.	3

Prompt	Details	Default
`What web root do you want to use?`	Enter the web root.	`/aspera/orchestrator`
`MySQL will run on this machine (y/n)?`	Because you already configured MySQL to run on a remote machine, enter "n".	"y"
`What is the database server's host name or IP address?`	Enter the IP address of the machine where the remote MySQL database is installed.	
`What port is the remote MySQL listening on?`	Enter the port configured on the remote machine.	
`What user should we use to connect to the database?`	Enter `root`.	`root`
`What password should we use to connect to the database?`	Enter the MySQL `root` password that was created for the remote MySQL database.	
`MySQL will need to start/restart during configuration. Continue (y/n)?`	Enter "y" to continue with the configuration process. If you enter "n", the system will exit setup.	"y"
`Enter IP address of network interface for Apache to listen on.`	Enter the hostname or IP address of your Orchestrator server.	`0.0.0.0`
`What hostname or IP address should Apache use to identify itself (in the SSL Certificate)?`	Enter the hostname or IP address for Apache.	`10.0.174.44, site.com`
`What port would you like to run Apache http on?`	Enter port for Apache HTTP.	`80`
`What port would you like to run Apache https on?`	Enter port for Apache HTTPS.	`443`
`Would you like to generate a self-signed SSL certificate, or install your own ([g]enerate,[c]opy) (current: g)?`	• To generate a self-signed SSL certificate, enter "g" • To install your own, enter "c" **Note:** The key and certificate will be generated in the following directory: `/opt/aspera/common/apache/conf`.	`generate (g)`
`What IP addresses and hostnames are allowed to access Orchestrator?`	Enter the IP address and/or hostname of the instances that are allowed to access the Orchestrator.	`10.0.174.44, site.com` **Note:** These are not default values, but rather examples of the format that the IP address and hostname need to have to function correctly.

The system presents your settings as configured above.

7. If the settings are correct, enter "y" to accept them and continue; otherwise, enter "n" to re-enter the settings. Pressing "x" will exit the installation process.

```
Are these settings correct? (y/n/x with x for exit) y
```

8. Once setup has completed, you are prompted to restart Apache.

   Enter "y". (Entering "n" exits the setup process.)

```
Apache needs to be restarted, restart it now (y/n)? (default:y) y
```

9. At the system request to restart MySQL, enter "y". (Entering "n" exits the setup process.)

```
MySQL needs to be restarted, restart it now (y/n)? (default:y) y
```

   When the setup process is complete, a message about Apache ports appears, for example:

```
Reminders:
 - Apache needs ports 80 and 443 open in firewalls.
```

   Make a note of these port numbers for future reference.

10. To start the Orchestrator processes, run the following command:

```
 asctl orchestrator:start
```

   Depending on your system, you may see the status `NOT running` in the system response:

```
Orchestrator Status:
  -> Orchestrator Manager running with pid: 3447
  -> Orchestrator Engine NOT running
  -> Mongrel serving orchestrator on port 5000 is NOT running
  -> Mongrel serving orchestrator on port 5001 is NOT running
 .  .  .
  -> Orchestrator Monitor NOT running
  -> Asynchronous Worker Process 0 is NOT running
  -> Asynchronous Worker Process 1 is NOT running
  -> Synchronous Worker Process 2 is NOT running
 .  .  .
```

   This is an indication that the Orchestrator processes need additional time to start. Wait approximately one minute, then check the status by entering the following command:

```
# asctl all:status
```

   You should now see confirmation that Apache and MySQL are running, and a status message similar to the following that confirms the processes are now running:

```
Apache:              running
MySQL:               running

Orchestrator Status:
  -> Orchestrator Manager running with pid: 3447
  -> Orchestrator Engine NOT running
  -> Mongrel serving orchestrator on port 5000 is running with pid: 3465
  -> Mongrel serving orchestrator on port 5001 is running with pid: 3467
  .  .  .
  -> Orchestrator Monitor running with pid: 3463
  -> Asynchronous Worker Process 0 is running with pid: 3480
  -> Asynchronous Worker Process 1 is running with pid: 3482
  -> Synchronous Worker Process 2 is running with pid: 3484
  .  .  .
```

Completing the Installation in the UI

Note: If you are performing an upgrade of Orchestrator, skip this section.

1. Open the Orchestrator user interface.

In a web browser, go to the following URL, where *orchestrator_ip_address* is the IP address or fully qualified name of the machine on which Orchestrator is installed.

```
https://orchestrator_ip_address
```

Note: Depending on your current certificates, you may get a security warning. If so, click **Advanced** and add a security exception for the Orchestrator website URL.

2. In the Login field, enter `admin`; leave the Password field blank.
3. Create a password for the admin user.

   In the Change Password dialog, leave the **Current password** field blank and enter your new password. Enter it again in the **Confirm password** field.
4. Apply the license file.

   a) In the Update License dialog, select a license file to upload, or paste the contents of your license file into the text box.

   b) Click **Update and validate license**.

      The message `New license applied and validated successfully` appears in the dialog.

      Note: The Orchestrator UI displays the currently installed version and license expiration date in the footer of every page.
5. Confirm that all processes are running.

   Click **Engine** > **Processes** and review the Status column to confirm that all processes are running. For any process with a Stopped status, click **Start** to resume.
6. If you need the system user, you must activate that account manually and set a password.

   In the top navigation menu, click **Users**. To the right of the **system** entry, click the "More" (. . .) icon , then click **Activate**.

   Click the "More" icon again, then click **Change password**

   Note: As a security measure, the `system` user is not assigned a password automatically after installation of Orchestrator.

   On the **Change password for system** page, enter your password in the **Current password for admin** field, and enter and confirm a new system user password. Click **Change** to confirm.
7. Now that you've installed Orchestrator, configure the product. See "Product Configuration" on page 55 for details.

Installing Custom Ruby Gems

This procedure describes how to install a custom Ruby gem during the installation process for Orchestrator.

Note: You must reinstall custom gems after an upgrade of Orchestrator, because Ruby gems are not preserved during an upgrade.

1. Download the custom gem to the server where Orchestrator is installed.
2. From the command line, set `GEM_HOME` to the path for the installation of Orchestrator.

   For example:

```
C:\Program Files (x86)\Aspera\Orchestrator\vendor\ruby\lib\ruby\gems\2.3.0
```

3. To install the custom gem to `GEM_HOME` defined above, run the following from the command line:

```
# gem install
```

4. Add the name of the custom gem to the Gemfile:

```
# path_to_custom_gem/custom_gem_name install
```

5. Confirm that all dependencies in your Gemfile are available to your application:

```
# bundle install
```

6. Confirm that the gem is installed:

```
gem list -i <gem_name> -v version
```

7. Perform the steps in "Installing Orchestrator with MySQL on the Same Server" on page 2 (see the section, Running the Installers).
8. Start Orchestrator.

```
# asctl orchestrator:start
```

Upgrading Orchestrator

Upgrading: Overview

This section provides guidance in upgrading Orchestrator to the current release for a single node installation. To upgrade a high availability setup, see "Upgrading Orchestrator 2.3.5+ to 4.0.1 in High Availability" on page 48.

The upgrade is done with Orchestrator service interruption on a single node, so customers must plan ahead for downtime. The upgrade automatically uninstalls the previous installation before installing the new version. Therefore, it is essential that you create a backup according to the procedure in "Upgrading Orchestrator: Requirements" on page 13, in case rollback is needed after upgrading the product.

Workflows and Plugins

Note: Because Orchestrator 3.2.0 and later versions only support Ruby 2.3.0, plugins that use Ruby 1.8.7 will no longer be available after an upgrade from versions before v. 3.2.0.

Plugins can be upgraded "live", without restarting Orchestrator.

When Orchestrator is stopped and restarted, the previously running workflow instances pick up at the point where they were left just before they were stopped.

Orchestrator is backward compatible so that the existing workflows can still run with the newer Orchestrator version. If the workflows must be upgraded as well (for example, to take advantage of a new plugin), the upgrade can be done live by manually exporting/importing each workflow.

Upgrading Orchestrator: Requirements

Note: If you have a high availability setup, see "Upgrading Orchestrator 2.3.5+ to 4.0.1 in High Availability" on page 48

The following are requirements which need to be completed before upgrading Orchestrator on a single node. See "Orchestrator Directory Locations" on page 224 for more information about the directory locations referred to in this procedure.

Note: All commands in the following procedure must be performed as `root`, or an equivalent superuser.

1. Confirm that both the Aspera Common Package and Aspera Orchestrator are already installed on the Linux server.
2. Perform a backup of MySQL data with a MySQL dump.

```
$ /opt/aspera/common/mysql/bin/mysqldump -h localhost -u root -p orchestrator > /tmp/
orchestrator_db_backup_timestamp.sql
```

   Note: The generated `.sql` file may take several gigabytes of disk space depending on the database size, so make sure that enough space is available on the targeted partition (`/tmp` in the example above) before running the backup.
3. Run a backup of Orchestrator data with the Snapshot feature.

See “Create Snapshot” on page 80 for details.

Note: Move all backup files onto a safe server after running the backup so they don’t take up space on Orchestrator.

4. Back up any important configuration files which are located in the `config` directory.

   This directory is overwritten with new data during an upgrade. If the database configuration is not standard (if, for example, there is a nonstandard password), back up the following file:

   ```
   /opt/aspera/orchestrator/config/database.yml
   ```

   It will be overwritten by the default released file, and you will lose the data unless it is backed up elsewhere. Copy the files to a safe location, such as `/tmp`:

5. Copy any custom files — such as `aspera_logo_simple.jpg` and `aspera.css` — to a safe location, and restore them after the upgrade.

   During the software installation phase of an upgrade, custom images, branding files and style sheets are copied from `/opt/aspera/var/config/orchestrator/` to `/opt/aspera/orchestrator/public/`. Thus make sure that any custom files (as well as `orchestrator.yml`, if the default Orchestrator configuration has been changed) are placed under `/opt/aspera/var/config/orchestrator/` before doing the upgrade.

   ```
   /opt/aspera/orchestrator/public/images/aspera_logo_simple.jpg
   /opt/aspera/orchestrator/public/stylesheets/aspera.css
   ```

6. Back up the `/var` directory.

   Although this directory is preserved during upgrade, it is important to take the precaution.

   ```
   $ cd /opt/aspera/ ; tar chvfz /tmp/aspera_orig_timestamp.tar.gz var/
   ```

Upgrading a Single Orchestrator Node from v. 2.3.5+ to v. 4.0.1

The procedure in this topic is for users upgrading a single Orchestrator node from v. 2.3.5 + to v. 4.0.1. If you are upgrading in a high availability environment, use the procedure described in “Upgrading Orchestrator 2.3.5+ to 4.0.1 in High Availability” on page 48.

Important: IBM Aspera supports direct upgrades to the current General Availability (GA) version from only two GA versions prior to the current release. To upgrade to the latest version, you must be within two GA versions of the current version. Upgrading from older version requires upgrading in steps. For example, if you are four GA versions behind, upgrade to two GA versions behind (GA - 2), and then upgrade to the current GA version.

Note: If you are upgrading from an earlier version of Orchestrator (before 2.3.5+), contact Support.

Warning: Prior to performing any upgrade customers should:

1. Perform a full environment back up and ensure the back up is successful. In case the upgrade fails, the only reliable, short-term fix is to roll back the environment using the back up.
2. Test the upgrade in a test environment comparable to the production environment.
3. If upgrading the test environment is successful, upgrade the production environment, but do not bring the production environment back online.
4. Prior to bringing the production environment back online, the customer must test the application to determine if an immediate rollback is needed. Otherwise, customers risk losing all data generated between upgrade and rollback.

Before upgrading Orchestrator, review product release notes. IBM Aspera strives to maintain compatibility from one release to the next, but sometimes we must make changes that affect a small number of customers. Review the release notes for all Orchestrator versions that have been released since your current version. In particular, the **Breaking Changes** section highlights changes that may require you to adjust your workflow, configuration, or usage. These issues may impact your new installation.

Before you begin the upgrade: Follow the procedure in "Upgrading Orchestrator: Requirements" on page 13.

Important:

- Make sure you have your username and password, because you will need them to log in after upgrade.
- Make sure you create and save a MySQL database dump, because you will need to restore it after upgrade.

I. Preliminary Tasks: Backing up Orchestrator Data and Stopping the Services

Before upgrading the Orchestrator server, you must back up all Orchestrator data so that you can restore them after upgrade, if needed.

See "Orchestrator Directory Locations" on page 224 for more information about the directory locations referred to in this procedure.

Note: All commands in the following procedure must be performed as `root`, or an equivalent superuser.

1. Create a backup of MySQL data with `mysqldump`.

   ```
   # /opt/aspera/common/mysql/bin/mysqldump -h localhost –u root –p orchestrator > /tmp/
   orchestrator_db_backup_timestamp.sql
   ```

   Troubleshooting tip: You may see an error similar to this when performing MySQL related operations:

   ```
   ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)
   ```

   Re-run the command with `socket=`, for example:

   ```
   # /opt/aspera/common/mysql/bin/mysqldump -h localhost -u root --socket=/opt/aspera/common/
   mysql/var/run/mysql.sock -p orchestrator > /tmp/dump.sql
   ```

   Important: Confirm that the `dump.sql` file is created (and that it is > 0 in size) and save the file for data recovery, as needed.

   Note: The generated `.sql` file may take several gigabytes of disk space depending on the database size, so make sure that enough space is available on the targeted partition (`/tmp` in the example above) before running the backup.
2. Run a backup of Orchestrator data with the Snapshot feature.

   See "Create Snapshot" on page 80 for details.

   Note: Move all backup files onto a safe server after running the backup so they don't take up space on Orchestrator.
3. Back up any important configuration files which are located in the `config` directory, since this directory is overwritten with new data during an upgrade.

   On each node, copy the following files to a safe location, such as `/tmp`:

   ```
   /opt/aspera/orchestrator/config/orchestrator.yml
   /opt/aspera/var/config/orchestrator/licenses/orchestrator-license*
   /opt/aspera/orchestrator/config/database.yml
   ```
4. Copy any custom files — such as `aspera_logo_simple.jpg` and `aspera.css` — to a safe location.

   During the software installation phase of an upgrade, custom images, branding files and style sheets are copied from `/opt/aspera/var/config/orchestrator/` to `/opt/aspera/orchestrator/public/`. Thus make sure that any custom files (as well as `orchestrator.yml`, if the default Orchestrator configuration has been changed) are placed under `/opt/aspera/var/config/orchestrator/` before doing the upgrade.

   ```
   /opt/aspera/orchestrator/public/images/aspera_logo_simple.jpg
   /opt/aspera/orchestrator/public/stylesheets/aspera.css
   ```
5. Back up the `var` directory.

Although this directory is preserved during upgrade, it is important to take the precaution.

```
$ cd /opt/aspera/ ; tar chvfz /tmp/aspera_orig_timestamp.tar.gz var/
```

II. Upgrading the Orchestrator Server

Because of the Orchestrator's upgrade from MySQL 5.1 to 5.7, the upgrade procedure requires uninstalling the current version of the application and directories before installing the new version of Orchestrator.

1. Stop all Orchestrator-related services:

   ```
   # asctl all:stop
   ```

2. Uninstall the existing version of Orchestrator:

   ```
   # rpm -e aspera-orchestrator
   ```

3. Uninstall the existing version of Common:

   ```
   # rpm -e aspera-common
   ```

4. Delete these directories:

   ```
   /opt/aspera/common
   /opt/aspera/orchestrator
   ```

5. Install Orchestrator using the applicable procedure:
   - "Installing Orchestrator with MySQL on the Same Server" on page 2
   - "Installing Orchestrator with MySQL on a Remote Server" on page 6
6. Reload the MySQL database using the database dump you created in step 1:

   ```
   # /opt/aspera/common/mysql/bin/mysql -h 127.0.0.1 -P 4406 -u <user> -p orchestrator < /tmp/
   orchestrator_db_backup_timestamp.sql
   ```

   Troubleshooting tip: You may see an error similar to this when performing MySQL related operations:

   ```
   ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)
   ```

   Re-run the command with `socket=`, for example:

   ```
   # /opt/aspera/common/mysql/bin/mysql -h localhost -u root --socket=/opt/aspera/common/
   mysql/var/run/mysql.sock -p orchestrator < /tmp/dump1.sql
   ```

7. Migrate the MySQL database back to the schema for Orchestrator 4.0.

   Run these commands:

   ```
   # cd /opt/aspera/orchestrator
   # script/cmd.sh
   Loading development environment (Rails 4.2.7)
   >> Database.generate
   ```

8. Restart your browser (recommended) or empty the browser cache.
9. Log in to the Orchestrator UI with the login and password you saved before you began the upgrade.
10. Check the following directory if there are capsules present in the folder.

    ```
    /opt/aspera/var/config/orchestrator/capsules/
    ```

    Remove all capsules that are specific to the previous version of Orchestrator (in other words, the version of Orchestrator that you are upgrading to the current version).
11. Upgrade your plugins and portlets as needed. See "Upgrading Plugins and Portlets After an Upgrade of Orchestrator" on page 19.

Upgrading a Single Orchestrator Node from v. 4.0.0 to v. 4.0.1+

The procedure in this topic is for users upgrading a single Orchestrator node from v. 4.0.0 to v. 4.0.1 and later.

Important: IBM Aspera supports direct upgrades to the current General Availability (GA) version from only two GA versions prior to the current release. To upgrade to the latest version, you must be within two GA versions of the current version. Upgrading from older version requires upgrading in steps. For example, if you are four GA versions behind, upgrade to two GA versions behind (GA - 2), and then upgrade to the current GA version.

Warning: Prior to performing any upgrade customers should:

1. Perform a full environment back up and ensure the back up is successful. In case the upgrade fails, the only reliable, short-term fix is to roll back the environment using the back up.
2. Test the upgrade in a test environment comparable to the production environment.
3. If upgrading the test environment is successful, upgrade the production environment, but do not bring the production environment back online.
4. Prior to bringing the production environment back online, the customer must test the application to determine if an immediate rollback is needed. Otherwise, customers risk losing all data generated between upgrade and rollback.

Before upgrading Orchestrator, review product release notes. IBM Aspera strives to maintain compatibility from one release to the next, but sometimes we must make changes that affect a small number of customers. Review the release notes for all Orchestrator versions that have been released since your current version. In particular, the **Breaking Changes** section highlights changes that may require you to adjust your workflow, configuration, or usage. These issues may impact your new installation.

Before you begin the upgrade: Follow the procedure in "Upgrading Orchestrator: Requirements" on page 13.

Important:

- Make sure you have your username and password, because you will need them to log in after upgrade.
- Make sure you create and save a MySQL database dump, because you will need to restore it after upgrade.

I. Preliminary Tasks: Backing up Orchestrator Data and Stopping the Services

Before upgrading the Orchestrator server, you must back up all Orchestrator data so that you can restore them after upgrade, if needed.

See "Orchestrator Directory Locations" on page 224 for more information about the directory locations referred to in this procedure.

Note: All commands in the following procedure must be performed as `root`, or an equivalent superuser.

1. Create a backup of MySQL data with `mysqldump`.

   ```
   # /opt/aspera/common/mysql/bin/mysqldump -h localhost –u root –p orchestrator > /tmp/
   orchestrator_db_backup_timestamp.sql
   ```

   Troubleshooting tip: You may see an error similar to this when performing MySQL related operations:

   ```
   ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)
   ```

   Re-run the command with `socket=`, for example:

   ```
   # /opt/aspera/common/mysql/bin/mysqldump -h localhost -u root --socket=/opt/aspera/common/
   mysql/var/run/mysql.sock -p orchestrator > /tmp/orch400_db_backup.sql
   ```

Important: Confirm that the `orch400_db_backup.sql` file is created (and that it is > 0 in size) and save the file for data recovery, as needed.

Note: The generated `.sql` file may take several gigabytes of disk space depending on the database size, so make sure that enough space is available on the targeted partition (`/tmp` in the example above) before running the backup.

2. Run a backup of Orchestrator data with the Snapshot feature.

   See "Create Snapshot" on page 80 for details.

   Note: Move all backup files onto a safe server after running the backup so they don't take up space on Orchestrator.

3. Back up any important configuration files which are located in the `config` directory, since this directory is overwritten with new data during an upgrade.

   On each node, copy the following files to a safe location, such as `/tmp`:

```
/opt/aspera/orchestrator/config/orchestrator.yml
/opt/aspera/var/config/orchestrator/licenses/orchestrator-license*
/opt/aspera/orchestrator/config/database.yml
```

4. Copy any custom files — such as `aspera_logo_simple.jpg` and `aspera.css` — to a safe location.

   During the software installation phase of an upgrade, custom images, branding files and style sheets are copied from `/opt/aspera/var/config/orchestrator/` to `/opt/aspera/orchestrator/public/`. Thus make sure that any custom files (as well as `orchestrator.yml`, if the default Orchestrator configuration has been changed) are placed under `/opt/aspera/var/config/orchestrator/` before doing the upgrade.

```
/opt/aspera/orchestrator/public/images/aspera_logo_simple.jpg
/opt/aspera/orchestrator/public/stylesheets/aspera.css
```

5. Back up the `var` directory.

   Although this directory is preserved during upgrade, it is important to take the precaution.

```
$ cd /opt/aspera/ ; tar chvfz /tmp/aspera_orig_timestamp.tar.gz var/
```

II. Upgrading the Orchestrator Server

Orchestrator's upgrade procedure requires uninstalling the current version of the application before installing the new version of Orchestrator. You will also upgrade Common and confirm the new version has been installed.

1. Stop all Orchestrator-related services and check their status:

```
# asctl orchestrator:stop
```

```
# asctl all:status
```

2. Uninstall the existing version of Orchestrator:

```
# rpm -qa | grep aspera
```

```
# rpm -e aspera-orchestrator-4.0.0.XXXXXX.x86_64
```

3. Upgrade the existing version of Common and confirm its version:

```
# rpm -Uvh ibm-aspera-common-1.2.33-XXXXXXX.x86_64.rpm
```

```
# rpm -qa | grep aspera
```

4. Install Orchestrator using the applicable procedure:

   - "Installing Orchestrator with MySQL on the Same Server" on page 2
   - "Installing Orchestrator with MySQL on a Remote Server" on page 6

5. Restart your browser (recommended) or empty the browser cache.
6. Log in to the Orchestrator UI with the login and password you saved before you began the upgrade.
7. Upgrade your plugins and portlets as needed. See "Upgrading Plugins and Portlets After an Upgrade of Orchestrator" on page 19.

Upgrading Plugins and Portlets After an Upgrade of Orchestrator

During an upgrade, plugins are not upgraded automatically and new portlets are not enabled by default; the steps below will allow you to update these as needed.

1. Go to **Engine > Plugins** and click **Available Plugins** to see the newer plugins.

   Each plugin contains a Revision History screen providing details about the newer plugin. After reviewing the revision history, upgrade the plugin and force maintenance. For more details on plugin installation and upgrading, see "Plugins" on page 88.
2. Find out if any new plugins are not already enabled by selecting **Not enabled only**, and enable them if necessary.

   Note: In a production system, upgrade the new plugins individually only when necessary (compare the previous version with the current released version). It is advisable to test them in pre-production first to make sure that the new plugins operate well with the existing workflows.
3. Click **Engine > Plugins > Reload Plugins**, and then click **Engine > Processes > Force Maintenance** to set the new code.
4. If the new portlets are not enabled, and they are needed, click **Engine > Portlets > Enable all**.
5. Test your workflows to make sure they function properly after the upgrade.

Rolling Back an Upgrade

Note: The full procedure below is applicable to an active/active ACM-based cluster. For a single node rollback, skip the ACM-related steps.

1. Disable ACM.

   In the STDOUT below, the `orchestrator1` node has MySQL running as `active`. This is your starting point.

   ```
   $ /opt/aspera/acm/bin/acmctl -i
   Checking current ACM status...
   Aspera Cluster Manager for Orchestrator - status
   ------------------------------------------
   Local hostname:         orchestrator1
   Active node:            orchestrator1 (me)
   Status of this node:    active
   Status file:            current
   Disabled globally:      no
   Disabled on this node:  no
   …
   ```
2. On the node where MySQL is active, follow the steps below:

   a) Disable ACM.

   ```
   $ /opt/aspera/acm/bin/acmctl -d
   ```

   b) Copy the backup packages into a local directory, for example, `/tmp/`.

   The following are examples of backup packages.

   ```
   orchestrator_db_backup_timestamp.sql
   aspera_orig_timestamp.tar
   ```

   c) Stop Aspera Orchestrator:

   ```
   $ asctl orchestrator:stop
   ```

d) Use `mysqldump` to back up the current database.

```
$ /opt/aspera/common/mysql/bin/mysqldump -h <IP address> -u root -p orchestrator > /tmp/
orchestrator_db_backup_current_timestamp.sql
```

e) Archive the existing `/opt/aspera/var`.

```
$ cd /opt/aspera/ ; tar cvfz /tmp/aspera_current_timestamp.tar.gz var/
```

f) Import the backup database.

```
$ /opt/aspera/common/mysql/bin/mysql -u root -p aspera orchestrator < /tmp/
orchestrator_db_backup_orig_timestamp.sql
```

g) Import the backup archive in `/opt/aspera/var`.

```
$ cd /opt/aspera/
$ rm -rf var/
$ tar xvfz /tmp/aspera_orig_<timestamp>.tar.gz
```

h) Check if there are capsules present in the following folder and if there are any, remove them.

```
/opt/aspera/var/config/orchestrator/capsules/
```

i) Reinstall the previous release (in this example, Orchestrator 2.3.0).

```
$ yum install aspera-orchestrator-2.3.0.build_version.x86_64.rpm
```

j) Restart Orchestrator.

```
$ asctl orchestrator:restart
```

k) Re-enable ACM.

```
$ /opt/aspera/acm/bin/acmctl -e
```

3. On the node where MySQL is passive,follow the steps below:

a) Disable ACM.

```
$ /opt/aspera/acm/bin/acmctl -d
```

b) Stop Aspera Orchestrator.

```
$ asctl orchestrator:stop
```

c) Reinstall the previous release (in this example, v2.3.0).

```
$ yum install aspera-orchestrator-2.3.0.build_version-0.x86_64.rpm
```

d) Restart Aspera Orchestrator.

```
$ asctl orchestrator:start
```

Uninstalling Orchestrator

This procedure uninstalls Orchestrator from the server.

1. To obtain the version number of the installed version of Orchestrator, run the following command:

```
# rpm -qa grep aspera
```

This provides a list of all the Aspera products that are currently installed, including Orchestrator.

2. Stop the services.

```
# asctl all:stop
```

3. To uninstall the product, run the following command. The value for *version* is the version number of Orchestrator that you obtained in Step 3.

```
# rpm -e aspera-orchestrator-version-0.x86_64.rpm
```

High Availability (Cluster) Installation

This section describes how to set up and maintain high availability environments with Orchestrator.

Overview of High Availability for Orchestrator

Orchestrator can be deployed in a high availability (HA) environment that leverages the Aspera Cluster Manager (ACM) software. It runs in active/active mode for Apache and Orchestrator services, and in active/passive mode for MySQL.

The Orchestrator HA environment can run on-premise or in the cloud.

The use case described in this guide is a dual Orchestrator instance cluster, where the Orchestrator step execution is balanced between two active nodes, providing a high-availability service with seamless automatic failover in the event that one node fails or becomes unavailable.

It is possible to have more than two nodes in an HA environment. However, these use case scenarios are not addressed in this document. Contract Aspera Professional Services if you need further information about configuring another type of environment.

General Requirements

Aspera Professional Services typically handles the installation of Aspera software and tools for high availability (HA) systems. However, it is the customer's responsibility to provide and configure infrastructure hardware and the required operating systems. In particular, the shared storage required for HA systems must be configured prior to Aspera personnel installing the Aspera products.

The systems requirements for HA installation of Orchestrator on Linux are the following:

- **Hardware:** Normal HA operation requires two servers. Virtual machines can be used as long as enough resources are allocated to each of them.
- **Operating system:**

  Note: Red Hat high-availability packages (such as `ricci`, `luci`, `rgmanager`, and `cman`) are *not* used, and therefore must not be installed or activated in the environment.

  See also, "List of System Commands Used by Aspera Cluster Manager (ACM)" on page 227.
- **Software:**
  - Aspera Orchestrator

    Note: All Orchestrator nodes in the cluster must have the same installed version of Orchestrator.
  - Aspera Cluster Manager (ACM) for Orchestrator (the latest version, provided by Aspera Professional Services)
- **Time synchronization:**The system clocks for all nodes in the HA cluster (including the shared storage) must always be kept in sync in order for ACM to operate correctly. Aspera recommends using the **`ntpd`** daemon (for the NTP (Network Time Protocol)), but any time-synchronization mechanism is acceptable.
- **Database:** Typically, the MySQL database runs on each Aspera Orchestrator node across shared MySQL data. This MySQL database is delivered with the Aspera Common Components and is supported by the Aspera Support team. An external MySQL database with an high availability solution of its own is also a possible solution, although IBM Aspera will not support that option.

Note: The Oracle MySQL cluster solution comes with the NDB engine; this will not interoperate with Orchestrator because only MySQL with the InnoDB engine is supported.

- **Load balancer:** A load balancer that implements a Virtual IP address (VIP) is required. For more information about what is expected from the load balancer, see "Load Balancer Behavior" on page 26.
- **Single point of failure:** Aspera strongly recommends that customers identify single points of failure (SPOF) in the environment and recognize the source of these risks. Often, these are situations where all nodes are plugged into the same power source. Another possible SPOF is shared storage or a load balancer that is not also HA.

Storage Requirements

Shared storage is required for files uploaded to Orchestrator, as well as for the associated MySQL data files and ACM files. The shared storage must meet the following criteria:

- High availability (for example, with RAID arrays) to avoid a single point of failure
- Completely reliable and accessible 100% of the time, so ACM can secure highly available Orchestrator operations and prevent data corruption.

Minimum configuration requirements are the following:

- MySQL data: 20GB
- ACM: 1GB
- Orchestrator `var` data: 20GB

Because ACM is not migrating the mount points over failover, MySQL data, Orchestrator `var` data, and ACM data must be mounted and visible on both nodes simultaneously.

Shared File Systems Compatible with ACM

Aspera recommends dedicating storage for ACM whenever possible in order to avoid creating I/O bottlenecks when large packages are being transferred to shared storage at very high speeds.

ACM supports SAN and NAS external storage. However, be aware of the following limitations:

- A SAN storage requires a cluster file system for the Aspera high availability environment to function properly. Block-level storage access will not work; only file-level storage access is suitable. Files must be visible from both nodes concurrently. If a file is changed on one node, the other node must be able to immediately incorporate the changes.
- A storage solution with NFS Version 3 may cause problems, because NFS3 does not automatically release the locks held by a client. This is critical for MySQL, where only one node at a time may write to the shared disk. With NFS3, if the active node that is running MySQL goes down, the passive node is unlikely to be able to start MySQL when it takes over, because some MySQL files may still be locked on the shared storage. With NFS4 technology, all locks associated with a TCP session that times out will be automatically released, making MySQL automatic failover possible.
- The NFS mount option noac must be used with the `/mnt/shared/orchestrator/acm_data/` mount point for ACM to function properly.

  Important: Using noac on any mount point *other than* `/mnt/shared/orchestrator/acm_data/` may lead to severe performance issues.

  See "Mounting Remote File Systems on Each Orchestrator Server" on page 28 for more information.
- CIFS is not supported.

Folders that Must be Shared

For Orchestrator on-premise high availability installation, the folders and other data sources listed below must be set up with shared external storage and shared between the nodes.

- The `run` folder:

  ```
  /opt/aspera/var/run/orchestrator/
  ```

  Important run-time information is stored in this folder. For example, the file trigger history is stored here, and thus each node has the same perspective of already-triggered files. Alternatively, the history may also be kept in the shared MySQL data. In orchestrator.yml, set the `parameter persistence_mode` to either drive or db (both options are correct values).
- The `config` folder:

  ```
  /opt/aspera/var/config/orchestrator/
  ```

  The plugin directories are present in this folder (in the actions subdirectory). Importing a plugin is done on only one node.
- The `archive` folder:

  ```
  /opt/aspera/var/archive/orchestrator/
  ```

  The plugins are archived in this folder; sharing it allows the nodes to share the same view of archived plugins.
- The `data` folder:

  ```
  /opt/aspera/common/mysql/data
  ```

- The `acm` folder.

  ```
  /opt/aspera/acm
  ```

Note: On servers that have other Aspera products installed, do not share this top-level folder:

```
/opt/aspera/var/
```

Instead, share the these sub-folders:

```
/opt/aspera/var/config/orchestrator
/opt/aspera/var/run/orchestrator
/opt/aspera/var/archive/orchestrator
```

Note: Create the mount on `/mnt` to the shared storage:

```
$ mount -v -o vers=3,proto=tcp,port=2049 shared_storage mount_point
```

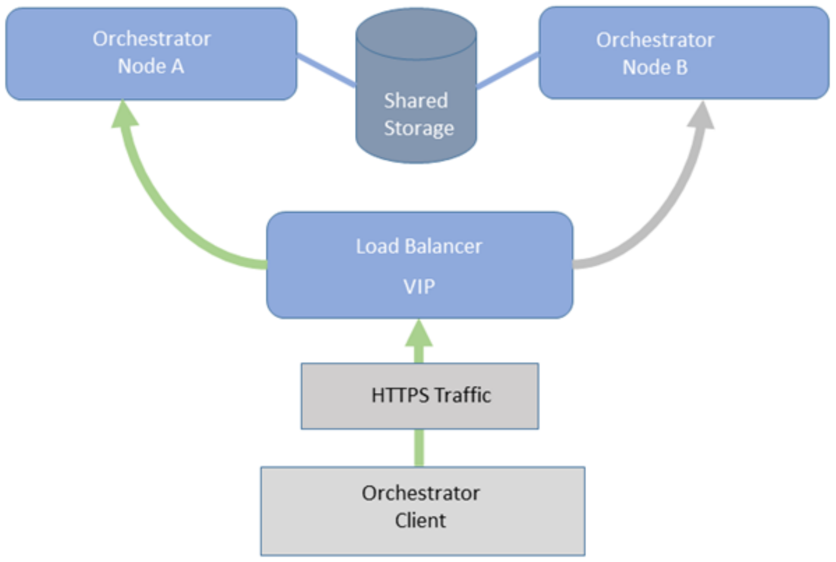
For example:

```
$ mount -v -o vers=3,proto=tcp,port=2049 ha_storage.aspera.us /mnt
```

Architecture of the High Availability Cluster

Overview of High Availability Architecture

The architecture of the active/active high availability system in Orchestrator is comprised of the two nodes, a shared storage, and a load balancer, which monitors the health of each Orchestrator node and redirects traffic accordingly.



When the load balancer detects that an Orchestrator node is unreachable, it automatically stops redirecting traffic to the unavailable node, and redirects all traffic to the remaining healthy nodes, balancing the load among them. Once the faulty node can be reached, the load balancer automatically detects the presence of the new healthy node and includes it in the traffic-sharing function.

The load balancer is provisioned with a virtual IP address (VIP) for user access; which it uses to manage all traffic related to the Orchestrator service. A fully qualified domain name (FQDN) —typically `orchestrator.mydomain.com`—is used to access the Orchestrator service and points to the VIP of the load balancer.

Both Orchestrator nodes can use private IP addresses. Only the VIP needs to be a public IP address, because it will be used by the clients to connect to the Orchestrator service components.

For additional information, see "Storage Requirements" on page 22, "Services Stack for the Orchestrator Nodes" on page 24, and "Load Balancer Behavior" on page 26.

Services Stack for the Orchestrator Nodes

Regardless of which architecture is deployed, both Orchestrator *nodes* (servers) are considered *active* because clients can contact any of them to access the web application portion or the transfer server portion. Nevertheless, not all of the Aspera services run at the same time on both machines.

While some services are considered `active/active` and do run on both nodes, other services are considered `active/passive` and only run on one of the two nodes. The node that runs all the services is called the *active* node, and the node that only runs the `active/passive` services is called the *passive* node.

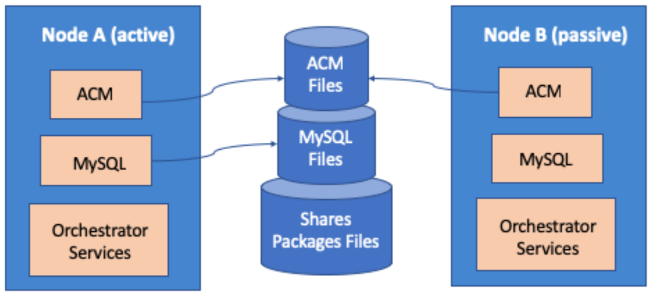


Figure 1. Orchestrator Services Stack Interacting with the Shared Storage

The **`mysql`** service runs only on the active node. While both nodes can access the ACM files and Shares packages simultaneously (read-write mode), the MySQL data files are accessed at a specific time by a single instance of the MySQL service running on the active node.

The following table lists each service and its location:

Service Name	**State (Node A/Node B)**	**Location**
`apache`	`active/active`	Runs on both nodes
`mysqld`	`active/passive`	Runs on active node only
`orchestrator` • `manager` • `engine` • `monitor` • `mongrels` • `workers`	`active/active`	Runs on both nodes

Aspera Cluster Manager (Overview)

Aspera Cluster Manager (ACM) is the software module responsible for starting the right services on an Orchestrator node according to that node's current status (active or passive). It is also in charge of monitoring the active node to determine when to fail-over the `active/passive` services from the active to the passive (when the active node becomes unresponsive).

Note: ACM must run as `root`.

How does it work?

ACM is installed on both Orchestrator nodes. Both instances of ACM first determine the status of the node on which they are running by checking a common status file stored on the shared space dedicated to ACM. In order to avoid a race condition while accessing that common status file, a specific locking mechanism—**`aslockfile`**— is used to synchronize both instances.

Once the status of a node is determined, the ACM instance running on the active node verifies that all of the services are running, and it starts any service that is not running. Once this is done, the instance updates the status file in order to keep its last modification date current.

The ACM instance running on the passive node checks that the status file is *current*, meaning that its last modification date is not older than 2 minutes). If the file is current, ACM checks that the `active/passive` services are up and running; it then starts all the services that are not running currently but should be running. If the common status file is no longer current, then it is a failover scenario, and ACM takes over as the new active node by starting all of the services.

How long does a failover process take?

If the passive node fails, then ACM does nothing. It is up to the load balancer to detect that the passive node is unresponsive and redirect the traffic accordingly. See "Load Balancer Behavior" on page 26 for more information. This process typically takes one minute or less.

If the active node fails, then ACM eventually detects that the status file is no longer current and it triggers a failover. Additionally, the load balancer detects that the active node is down and it redirects all traffic to the healthy node. This process typically takes up to 5 minutes.

Load Balancer Behavior

A load balancer monitors the health of each Orchestrator nodes and redirects the traffic accordingly. To learn how the load balancer fits into the overall architecture of high availability systems, see "Overview of High Availability Architecture" on page 23.

HTTPS Traffic

The load balancer must monitor the health of the HTTPS service running on each node. To do this, it can either use a method based on an HTTPS request, or simply check whether TCP port 443 is responding, that is, whether a SYN ACK packet is received after a SYN packet is sent by the monitoring service. If an RST packet is received instead, or if no packet is received at all, then the monitoring feature must consider the monitored service to be down and discard the related node (take it offline).

The load balancer can redirect any HTTPS request to any of the healthy nodes. Because the orchestrator web application uses a database shared by both nodes, any healthy node can respond to any request.

HTTP Redirection

The orchestrator application uses HTTPS by default, and it sets an automatic redirection from HTTP/TCP/80 to HTTPS/TCP/443 to force users to use a secure connection.

The load balancer can forward HTTP requests to the nodes, which then handle the redirection. Alternatively, the load balancer itself can handle the redirection; this prevents any insecure connections from being established with a node.

Note: A load balancer is optional for the functioning of a high availability environment. The majority of users can access the Orchestrator web UI without this feature.

Alternative High Availability Setups

There are three principal alternative setups available for high-availability systems:

- *Multi-tenant setup: two Orchestrators instances run on the same node*. Each Orchestrator instance has its own access URL and `config` directory, but they share a database.
- *Multiple Orchestrator nodes which operate independently*. Tracking issues may be simpler in this case, because each node can run its own dedicated workflows. **Note:** In a setup with the Federated Workflow plugin, an Orchestrator node can coordinate workflows running on other (multiple, independent) Orchestrator nodes.
- *Distributed architecture (MySQL running on a separate server from Orchestrator)*. For example, you can have three Orchestrator servers (all active and running Orchestrator and Apache services) and two MySQL servers (one running the MSQL service in active mode and the other running in passive mode). Note the following requirements:

- No upward limit on the number of Orchestrator servers
- Maximum of two MySQL servers
- For boot or failure of Orchestrator servers, ACM controls the service restart
- For boot or failure of MySQL servers, ACM controls the MySQL failover and service restart

See "Installing and Configuring MySQL on a Separate (Remote) Machine" on page 58 for more information on setting up MySQL with distributed architecture.

Setting up Orchestrator in the Cloud

The following are the setup options for Orchestrator in the cloud (assuming high availability installation):

- No shared storage, because typically there is no file-based storage available (only object storage).
- An external database with an high availability solution of its own is accessed with a unique virtual IP address.
- The file trigger history is stored in the external database.
- Plugins are not shared by the nodes, so importing and archiving a plugin must be done on both nodes of the cluster.
- Edits to the Ochestrator configuration and Apache configuration must be done on both nodes of the cluster.

Note: ACM (Aspera Cluster Manager) is *not* an option for cloud installation, so the services must be started and controlled by the operating system.

Installing and Configuring Operating Systems

Note: All commands in this section are run as `root`.

Installing a Supported Operating System

The admin user's first task is to install a supported operating system on both nodes that will support the Orchestrator application. See "General Requirements" on page 21 for details on supported operating systems.

Updating Your Environment

If desired, update some or all packages on each system.

```
# yum -y update
```

Checking Network Settings and Names

Confirm that your network settings are correctly configured and that each host has a unique host name that is properly configured within the name resolution mechanism you use (DNS, hosts file, and so on). Each host must be able to resolve its own name, as well as the name of the other node.

Run the following command on both nodes.

```
# hostname
haorchestratornode_id.my_domain.com
```

Configuring Local Firewalls

Do not place a traffic filter between the two nodes. If your nodes are located behind a corporate firewall (and thus appropriately protected), disable the Linux firewall components. The **chkconfig** command prevents the firewall from becoming active when the system is rebooted.

```
# service iptables stop
iptables:    flushing firewall rules:                [ OK ]
iptables:    Setting chaings to policy ACCEPT: filter        [ OK ]
iptables:    Unloading modules:                [ OK ]

# service ip6tables
ip6tables: Flushing firewall rules:                  [ OK ]
ip6tables: Setting chains to policy ACCEPT: filter         [ OK ]
ip6tables: Unloading modules:                  [ OK ]

# chkconfig iptables off
# chkconfig ip6tables off
```

If using CentOS 7 or RHEL 7, you must run two final commands to disable the firewall:

```
# systemctl stop firewalld
# systemctl disable firewalld
```

Note: If you do not disable the firewall, configure it to open the necessary ports for Aspera. See "TCP and UDP Ports Used in Orchestrator High Availability Environments" on page 227 for a list of ports used by the Orchestrator HA environment.

Disabling SELinux on All Servers

SELinux must be disabled or set to `permissive` in the `/etc/selinux/config` file on each Orchestrator server system. You can confirm the SELinux current status by running the **sestatus** command.

```
# sestatus
SELinux status: disabled
```

Creating User Accounts and Groups on Each Orchestrator Server (For Non-Root User)

The installation of the Aspera Common Components automatically creates a `mysql` user.

Note: It is critical to ensure that the UID and GID for the `mysql` user account is consistent across all Orchestrator servers.

Ensure that the permissions defined on an NFS server are appropriate for the shared directories (in other words, consistent with what has already been defined on the shared directories).

Note:

NFS v. 4 uses ID mapping to ensure the enforcement of shared directory ownership; it must be configured on the NFS server and each NFS client in a way that avoids access problems with Orchestrator and ACM.

Mounting Remote File Systems on Each Orchestrator Server

Orchestrator servers in HA environments must be configured with shared storage. There are three shared directories that need to be available to each Orchestrator server.

Note: Rights—such as `mysql:mysql`—may be set only on the NFS server side.

The following shared storage names are used as mount points in this document, but you may use any mount point names you prefer:

Mount Point	Usage	Owner	Permissions
/mnt/shared/orchestrator/*mysql_data* **Note:** Replace /mnt/shared/orchestrator/*mysql_data* by the mount point you chose for mysql data	Stores shared MySQL data files	mysql:mysql	drwx------
/mnt/shared/orchestrator/orchestrator_var_data	Stores Orchestrator upload and download files	nobody.nobody	drwx------
/mnt/shared/orchestrator/acm_data/	Stores shared ACM files	nobody.nobody	drwx------

1. Configure the /etc/fstab file.

   When this file is configured, the shared directories are automatically mounted when the system reboots.

   In the following example /etc/fstab file, the shared directories (/mnt/shared/orchestrator/mysql_data, /mnt/shared/orchestrator/orchestrator_var_data, and /home/mnt/shared/orchestrator/acm_data/) are shared from the NFS server with an IP address of 10.0.75.10 (note that the entries in your file do not need to share a single IP address). The example below contains options typically used when mounting file systems for use in the Orchestrator high availability environment. For further details on how to adjust the file structure according to your storage vendor requirements, consult the man page for the file (man fstab).

```
10.0.75.10:/home/mysql_data       /mnt/shared/orchestrator/mysql_data        nfs4
rw,sync,hard,intr 0 0
10.0.75.10:/home/orchestrator           /mnt/shared/orchestrator/
orchestrator_var_data            nfs4 rw,sync,hard,intr 0 0
10.0.75.10:/home/mnt/shared/acm_data/        /mnt/shared/orchestrator/acm_data/
nfs4 rw,noac,sync,hard,intr 0 0
```

   These shared directories are mounted to their corresponding local directories on each orchestrator server (/mnt/shared/orchestrator/*mysql_data*, /mnt/shared/orchestrator/orchestrator_var_data, and /mnt/shared/orchestrator/acm_data/).

   Explanation of Configuration Options

Option	Details
nfs4	This first entry indicates the type of file system which is being shared (in this case, nfs4).
noac	This setting disables attribute caching; it is crucial for the /mnt/shared/orchestrator/acm_data/ directory because attribute caching can break ACM. (Note that attribute caching may be useful in other directories because it can optimize (speed up) performance.)
hard	This setting permits unlimited retry requests to the NFS server; however, it can cause the system to freeze. The alternative is the soft setting which developers sometime avoid because it can cause data corruption.

   For a description of all available options, see the man page for the Network File System (man nfs).

2. Once you have configured the /etc/fstab file, make sure the mount points have been created on both Orchestrator servers, and confirm each directory's ownership and permissions.

Installing Orchestrator in High Availability Environments

Overview of Installation Process in High Availability

Begin by running the Orchestrator installer on each node according to the instructions in "Installing Orchestrator with MySQL on the Same Server" on page 2.

Once Orchestrator is running properly on each server, the next step is to configure the high availability environment by integrating the nodes with each other. This involves configuring the MySQL services and installing ACM software on each server; see "Setting up Aspera Cluster Manager" on page 31 and "Configuring MySQL and Aspera Services" on page 36.

Note: If watchfolders are used in Orchestrator workflows, the files triggered by Orchestrator must be visible to all Orchestrator instances if the workflow logic relies on reading and writing the files after triggering them. For this reason, you should install the watchfolders on shared storage.

Configuring orchestrator.yml for High Availability

This article describes how to update the Orchestrator configuration for high availability environments.

1. Search for the following file:

   ```
   /opt/aspera/var/config/orchestrator/orchestrator.yml
   ```

   If you discover the file, follow the steps below.

   a) Back up the file in another location, then remove it. This is necessary because `orchestrator.yml` is shared between nodes.

   b) Compare this version of `orchestrator.yml` with the default file, located in the directory below:

      ```
      /opt/aspera/orchestrator/config/orchestrator.yml
      ```

      Look for differences between the two files (for example, in the number of threads).

      If you do observe differences between the two files, update the default `orchestrator.yml` file on each node with the values from the `orchestrator.yml` file located in /opt/aspera/var/config/orchestrator/

2. Edit `orchestrator.yml` on the first node.

   On the first node, open the following file:

   ```
   /opt/aspera/orchestrator/config/orchestrator.yml
   ```

   Edit the file to reflect the following configuration, adding any element that is missing.

   ```
   engine_instance: 1
   use_db_time: true
   active_active: true
   install_config_file: /opt/aspera/orchestrator/config/orchestrator.yml
   aspera_dir: /opt/aspera
   orchestrator_dir: /opt/aspera/orchestrator
   run_dir: /opt/aspera/var/run/orchestrator
   config_dir: /opt/aspera/var/config/orchestrator
   archive_dir: /opt/aspera/var/archive/orchestrator
   install_images_dir: /opt/aspera/var/config/orchestrator/images
   install_pages_dir: /opt/aspera/var/config/orchestrator/pages
   install_css_dir: /opt/aspera/var/config/orchestrator/stylesheets
   log_file: /opt/aspera/orchestrator/log/orchestrator.log
   ```

3. Edit `orchestrator.yml` on the second node.

   a) On the second node, open the following file:

      ```
      /opt/aspera/orchestrator/config/orchestrator.yml
      ```

b) Edit the file to match the configuration for the first node (shown in the previous step), with one exception:

   `engine_instance` should be set to 2.

4. Customize HTML branding on each node. (Optional)

   a) Open the following file and edit it to customize branding:

   ```
   /opt/aspera/orchestrator/config/branding.yml
   ```

   b) Remove the following file, if it exists:

   ```
   /opt/aspera/var/config/orchestrator/branding.yml
   ```

5. Store the log file in a local folder. (Optional)

   Log files can be stored in the following local folder:

   ```
   /opt/aspera/orchestrator/log/
   ```

   They can also point to the log folder on the shared storage:

   ```
   /opt/aspera/var/run/orchestrator/log/
   ```

Setting up Mount Points for Shared Orchestrator /var Data

1. Run the following commands to create the mount point for Orchestrator directories:

   ```
   $ cd /mnt/shared/orchestrator/orchestrator_var_data
   $ mkdir -p var/archive/orchestrator
   $ mkdir -p var/config/orchestrator
   $ mkdir -p var/run/orchestrator
   ```

2. On both nodes, run the following to add symbolic links:

   ```
   $ cd /opt/aspera/var/archive
   $ mv orchestrator orchestrator.bak
   $ ln -s /mnt/shared/orchestrator/orchestrator_var_data/var/archive/orchestrator orchestrator

   $ cd /opt/aspera/var/run
   $ mv orchestrator orchestrator.bak
   $ ln -s /mnt/shared/orchestrator/orchestrator_var_data/var/run/orchestrator orchestrator

   $ cd /opt/aspera/var/config
   $ mv orchestrator orchestrator.bak
   $ ln -s /mnt/shared/orchestrator/orchestrator_var_data/var/config/orchestrator orchestrator
   ```

3. Copy the shared data:

   ```
   $ cp -R /opt/aspera/var/archive/orchestrator.bak/* /opt/aspera/var/archive/orchestrator
   $ cp -R /opt/aspera/var/config/orchestrator.bak/* /opt/aspera/var/config/orchestrator
   $ cp -R /opt/aspera/var/run/orchestrator.bak/* /opt/aspera/var/run/orchestrator
   ```

Setting up Aspera Cluster Manager

You must set up Aspera Cluster Manager (ACM) as part of a high availability installation.

For overview information about ACM, see "Aspera Cluster Manager (Overview)" on page 25.

Installing ACM Software on the Shared Storage

1. Obtain the latest ACM software from IBM Aspera and place it in the shared `/mnt/shared/orchestrator/acm_data/` directory.
2. Extract the ACM software on the dedicated shared volume by running the following command, as `root`:

```
$ cd /mnt/shared/orchestrator/acm_data/
$ tar xzvf acm4orchestrator-0-9.tar.gz
```

Note: You only need to run this step on one node; the `/mnt/shared/orchestrator/acm_data/` directory is shared by both `orchestrator` servers.

3. Create a symbolic link by executing the following commands on both Orchestrator servers:

```
$ cd /opt/aspera
$ ln –s /mnt/shared/orchestrator/acm_data/ ./acm
```

4. Confirm that the file `database.yml` is present in the following directory:

```
/opt/aspera/orchestrator/config/
```

If you are using a distributed architecture, with MySQL running on servers that are separated from the Orchestrator servers, copy the `database.yml` file from the `/opt/aspera/orchestrator/config/` directory on any Orchestrator server into the `/opt/aspera/acm/config/` directory on any MyQL server.

5. Confirm that the MySQL user and password defined in `database.yml` match the `MYSQL_USER` and `MYSQL_PWD` values as defined in the following file:

```
/opt/aspera/acm/bin/acmcommon
```

6. Configure the services to run in the following directory:

```
/opt/aspera/acm/bin/acmcommon
```

By default, all services run together, as in the following example:

```
SERVICES=('APACHE' 'ORCHESTRATOR' 'MYSQL')
```

If you are running Orchestrator in a distributed architecture (MySQL and Orchestrator running on different servers), follow these additional steps.

- To run Orchestrator + Apache services only, set the following:

```
SERVICES=('APACHE' 'ORCHESTRATOR')
```

- To run MySQL services only, set the following:

```
SERVICES=('MYSQL')
```

Note: In a distributed architecture, the ACM mount points on `/opt/aspera/acm/` must point to different folders on the shared systems for the Orchestrator + Apache services and the MySQL systems. This is required because `SERVICES` must be set differently on those systems.

For example, in `/etc/fstab` on the Orchestrator + Apache server, you could have the following:n

```
10.0.75.10:/home/mnt/shared/acm_data_orchestrator/ /mnt/shared/orchestrator/acm_data/ nfs4
rw,noac,sync,hard,intr 0 0
```

On the server for MySQL, you could have the following:

```
10.0.75.10:/home/mnt/shared/acm_data_mysql/ /mnt/shared/orchestrator/acm_data/ nfs4
rw,noac,sync,hard,intr 0 0
```

7. To configure the ACM log file to be `/var/log/aspera.log`, managed by the `rsyslog` service, on each node, modify `/etc/rsyslog.conf` with the following steps.

   a) Add these lines at the end of the file (note that `# Aspera Logging` is text, not a command):

   ```
   # Aspera Logging
   local2.* -/var/log/aspera.log
   ```

   b) Replace all `cron.`none occurrences with the following:

```
cron.none;local2.none
```

c) Replace `/var/log/messages` occurrence with `-/var/log/messages`

d) Restart `rsyslog`:

```
$ /etc/init.d/rsyslog restart
```

8. Set up your system to rotate the logs.

   You may find that your log file, `/var/log/aspera.log` is growing too quickly. In that case, there are several ways to rotate Aspera logs:

   - "Option A": Add `/var/log/aspera.log` to the following directory:

     ```
     /etc/logrotate.d/syslog
     ```

   - "Option B": Create an entry for `aspera.log` in the following file:

     ```
     /etc/logrotate.conf
     ```

   - "Option C": Create a separate configuration file for `aspera.log` in the following directory:

     ```
     /etc/logrotate.d/
     ```

   Option A will rotate your logs with the system logs (usually once a week, compressed, and saving the last 10 logs). However, on some servers, there is so much traffic that the logs need to be rotated more often than once a week; in that case, use Option B or C.

   Option A: Add `/var/log/aspera.log` to the entries in `/etc/logrotate.d/syslog`, as follows:

   ```
   /var/log/messages
   /var/log/secure
   /var/log/maillog
   /var/log/spooler
   /var/log/boot.log
   /var/log/cron
   /var/log/aspera.log {
   sharedscripts
   postrotate
   /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true
   /bin/kill -HUP `cat /var/run/rsyslogd.pid 2> /dev/null` 2> /dev/null || true
   endscript
   }
   ```

   Option B: Edit `/etc/logrotate.conf` by adding the configuration after the line `# system-specific logs may also be configured here.`

   The following example compresses and rotates 10 logs whenever `/var/log/aspera.log` reaches 100MB. After log rotation is complete, it will run whatever scripts are specified by `postrotate ... endscript`.

   ```
   /var/log/aspera.log {
   rotate 10
   size 100M
   create 664 root
   postrotate
   /usr/bin/killall -HUP syslogd
   endscript
   compress
   }
   ```

   The following example compresses and rotates 10 logs once daily. Instead of moving the original log file and creating a new one, the **`copytruncate`** option tells **`logrotate`** to first copy the original log file, then truncate it to zero bytes.

   ```
   /var/log/aspera.log {
   daily
   rotate 10
   copytruncate
   ```

```
compress
}
```

Option C: Create a separate `/etc/logrotate.d/aspera` configuration file containing the same information as Option B.

If you find that logs are being overwritten before long transfers of many files are complete, you can increase the log size. For more information, see the IBM Aspera High-Speed Transfer Server Admin Guide.

9. Now that you have completed this procedure, you can review the steps in "Setting up Aspera Cluster Manager" on page 31.

Turning Off Automatic Restart for the MySQL, Apache, and Orchestrator Services

The `chkconfig` command turns off the automatic restart process for the Aspera services after reboot. Disabling automatic restart is essential in a high availability environment so that ACM can control the restart process as needed to maintain high availability.

1. Turn off the MySQL, Apache, and Orchestrator services by running the following:

```
# chkconfig aspera_mysqld off
# chkconfig aspera_httpd off
# chkconfig AsperaOrchestrator off
```

2. If you are using a Centos 7.x platform, run the following additional steps after turning off the services.
   a) After reboot, turn off (disable) the automatic restart of the `orchestrator` service:

   ```
   # systemctl disable orchestrator_asctl.service
   ```

   b) Remove the symbolic link:

   ```
   # /etc/systemd/system/multi-user.target.wants/orchestrator_asctl.service
   ```

   c) To confirm that it is disabled, run the following:

   ```
   # systemctl list-unit-files | grep orchestrator_asctl.service
   orchestrator_asctl.service                  disabled
   ```

Configuring the crontab Entry to Run ACM

Configure ACM services in **`crontab`** on both nodes so that the `acm4orchestrator` script is launched every minute.

Use the **`crontab -e`** command to configure your entry as follows.

```
$ crontab -e
* * * * * /opt/aspera/acm/bin/acm4orchestrator ip_address > /dev/null 2>&1
```

One parameter must be entered in **`crontab`**: the IP address of the host where the script is running. This parameter is passed to the **`acm4orchestrator`** script.

In the example below, the IP address is `10.0.71.21`.

```
$ crontab -e
* * * * * /opt/aspera/acm/bin/acm4orchestrator 10.0.71.21 device_ID > /dev/null 2>&1
```

Find the device ID:

```
$ stat -c "%d" mount_point
```

For example:

```
$ stat -c "%d" /mnt
```

In the example above, `/mnt` is the shared storage location.

Create the mount to the shared storage:

```
$ mount -v -o vers=3,proto=tcp,port=2049 shared_storage mount_point
```

For example:

```
$ mount -v -o vers=3,proto=tcp,port=2049 ha_storage.aspera.us /mnt
```

Once configured in **`crontab`**, the **`acm4orchestrator`** script runs regularly to determine the `active` node and start the required Orchestrator services on both the `active` and `passive` nodes, depending on their current status (`active` or `passive`).

Obtaining the crontab Parameter Values

To list the IP addresses available on a system, run the following command:

```
$ ip addr | grep "inet"
inet 127.0.0.1/8 scope host lo
inet6 ::1/128 scope host
inet 10.0.75.21/16 brd 10.255.255 scope global eth0
```

Running an ACM Sanity Check

The **`acmctl`** command has an option to check that the necessary configurations have been made that allow the **`acm4orchestrator`** script to run appropriately. You should make sure that each server passes the sanity test.

1. Run the **`acmctl`** command with the **`-s`** option on both nodes to verify the basic ACM prerequisites.

   Note that the status of the `Checking if an entry for ACM seems to exist in crontab` displays as KO, not OK, in the example below, because the user has not yet created the `crontab` entry that will run the ACM software on each server.

   ```
   $ /opt/aspera/acm/bin/acmctl –s

   ACM sanity check
   ----------------
   Checking if an entry for ACM seems to exist in the crontab     OK
   Checking that the orchestrator master service is disabled in chkconfig     OK
   Checking that SE Linux mode is not set to enforcing     OK
   ```

2. Correct any task that does not pass the sanity check (except for tasks with `crontab` status; those tasks are addressed in the next section).
3. Run the following command, where `ip_address` is the IP address of the host where the script is running:

   ```
   /opt/aspera/acm/bin/acm4orchestrator ip_address
   ```

   Examine the output to make sure there are no displayed errors.

Identifying the Status of ACM on Each Orchestrator Server

The following command can be used to identify which Orchestrator server is active and which is passive:

```
/opt/aspera/acm/bin/acmctl -i
```

For more information about using this command, see "Disabling and Re-enabling ACM on One Node" on page 45

Connecting to Orchestrator with the VIP

If a load balancer is providing a virtual IP address (VIP) in front of the Orchestrator servers, and the services are running properly, you can now connect to the Orchestrator application using the VIP assigned to the ACM cluster.

Configuring MySQL and Aspera Services

Setting up Database Connectivity

1. For the active/passive MySQL setup within an active/active cluster, open the following the file, `database.yml`:

   ```
   /opt/aspera/orchestrator/config/database.yml
   ```

2. Update the parameter entries `database.yml` with values that are appropriate for your system.

   Note: For installations with Ruby 2.3.0, you may need to edit the default values for the `primary_host` and `alternate_host` entries.

   File entries for an installation with Ruby 2.3.0:

   ```
   development:
     host: 127.0.0.1
     port: 4406
     primary_host: ip_address_primary_host
     alternate_host: ip_address_alternate_host
     adapter: mysql2
     username: root
     password: aspera
     database: orchestrator
     checkout_timeout: 1
     pool: 10
   test:
     host: 127.0.0.1
     port: 4406
     primary_host: ip_address_primary_host
     alternate_host: ip_address_alternate_host
     adapter: mysql2
     username: root
     password: aspera
     database: orchestrator_test
     checkout_timeout: 1
     pool: 10
   production:
     host: 127.0.0.1
     port: 4406
     primary_host: ip_address_primary_host
     alternate_host: ip_address_alternate_host
     adapter: mysql2
     username: root
     password: aspera
     database: orchestrator
     checkout_timeout: 1
     pool: 10
   ```

   File entries for an installation with Ruby 1.8.7:

   ```
   development:
       host: localhost
       port: 4406
       primary_host: ip_address_primary_host
       alternate_host: ip_address_alternate_host
       adapter: mysql
       username: root
       password: aspera
       database: orchestrator
       wait_timeout: 1
       pool: 10
   production:
       host: localhost
   ```

```
     port: 4406
     primary_host: ip_address_primary_host
     alternate_host: ip_address_alternate_host
     adapter: mysql
     username: root
     password: aspera
     database: orchestrator
     wait_timeout: 1
     pool: 10
```

3. Copy `database.yml` to the following directory:

   ```
   /opt/aspera/var/config/orchestrator/
   ```

   Run the following command:

   ```
   $ cp database.yml /opt/aspera/var/config/orchestrator
   ```

   This step ensures that the data is preserved during an upgrade.

Configuring MySQL and Aspera Services: Overview

When the ACM software is implemented in a high availability environment, the MySQL database is shared by the Aspera servers. Orchestrator accesses the shared MySQL database with the `root` account.

In order to configure the MySQL database to recognize the account, it is necessary to grant access privileges for the account within the database *before the database is shared*. This process involves using one system to configure the database for the `root` account, configuring each of the servers to use the shared database, and configuring each Orchestrator server to use a special `database.yml` file that specifies a primary and secondary MySQL server.

Granting Privileges for the Orchestrator Database Account

You must access the MySQL database to grant access privileges for the user `root`. This requires accessing the database on the node that you selected as the primary node with the MySQL `root` password and then granting privileges for the user `root` from either Orchestrator server.

This section provides two methods for granting privileges. "Method 1" is the simplest; however, it does not also succeed in setting the access. "Method 2" has a higher efficacy because it employs direct configuratinon of the MySQL environment.

Method 1: Run asctl Commands on the Command Line

To allow access with IP addresses or hostnames, run both **`asctl`** commands below on the primary node to add access for both server nodes:

```
$ asctl mysql:grant_remote_access "first_server_ip"
$ asctl mysql:grant_remote_access "second_server_ip"
```

Method 2: Run Commands in the MySQL Environment

1. Retrieve the value for `password` (the MySQL `root` account password) from the file, `/opt/aspera/var/config/orchestrator/database.yml`, on the primary node.
2. Copy the password from the previous step into the `password` field for each section (`development`, `test`, and `production`) in the file, `/opt/aspera/orchestrator/config/database.yml` on the primary server.
3. On the same Orchestrator server system (primary server), log in to `mysql` as `root` using the password value you retrieved from the `.my.cnf` file as shown below:

   ```
   $ $ /opt/aspera/common/mysql/bin/mysql -uroot -hlocalhost -proot_account_password
   orchestrator
   ```

   Note: There is no space between the **`-p`** option and the password value.

If the command is successful, you will be presented with the MySQL environment:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 42
Server version: 5.1.54 Source distribution

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>
```

4. Run the following commands using the default password value for the user `root` and replacing the default IP addresses with the IP addresses of the Aspera Orchestrator servers in your environment.

   .

   ```
   mysql> grant all privileges on *.* to 'root'@'10.0.100.100' identified by 'aspera' ;
   Query OK, 0 rows affected (0.00 sec)
   mysql> grant all privileges on *.* to 'root'@'10.0.20.20' identified by 'aspera' ;
   Query OK, 0 rows affected (0.00 sec)
   ```

   In the example above, the users are represented by `'root'@'10.0.100.100'` and `'root'@'10.0.20.20'`, and the corresponding password for each is represented by `aspera`

   Note: Include the quotation marks exactly as shown (`‘root’@’10.0.100.100’` and `‘root’`) and make sure to include the final semicolon symbol ";" which must be separated from the previous string with a space (`‘aspera' ;`).
5. Enter `exit` to exit the MySQL environment.

   ```
   mysql> exit
   ```
6. To verify that the changes have been implemented, log into MySQL from each node:

   ```
   $ cd /opt/aspera/common/mysql/bin/
   $ ./mysql -uroot -h primary_node_ip_address –proot_account_password orchestrator
   ```

   Note: As shown in the example above, there is no space between `-u` and `root`, `-h` and the IP address, and `-p` and the root account password.

   If you are able to access the MySQL environment, the changes are successfully implemented. If you do not see the wecome message, `Welcome to the MySQL monitor`, re-run the `mysql> grant all privileges` statements (as shown in Step 3) with the `root` account.

Sharing MySQL Data

This procedure lets you configure Orchestrator for the sharing of MySQL.

The steps below give you information about the current ownership on Orchestrator directories and then provide the commands for changing them as needed so that you can share MySQL data.

Note: All directories in Orchestrator (`/opt/aspera/../*/`) have `root:root` ownership *with the exception of* the following directories:

- `/opt/aspera/common/mysql` - `root:mysql` ownership
- `/opt/aspera/common/mysql/data` - `mysql:mysql` ownership

1. Preliminary step: stop the services.
   Stop Orchestrator and MySQL on both nodes with the following commands:

   ```
   $ asctl orchestrator:stop
   $ asctl mysql:stop
   ```
2. Confirm that the following directory has `rwxr-xr-x` permissions:

   ```
   /mnt/shared/orchestrator/
   ```

This directory is the Orchestrator mount point on the shared storage. If it does not have the necessary permissions, change them with the following command:

```
$ chmod 755 /mnt/shared/orchestrator/
```

3. Create a `mysql_data` directory under the mount point for the MySQL shared data files.
4. Open the `/etc/group` and `/etc/passwd` files to confirm that `mysql group-id` and `mysql user-id` are the same on both nodes.
5. On both nodes, run the following to change the ownership of the `mysql` directory to the `mysql` user and group.

```
$ cd /opt/aspera/common/mysql
$ mv ./data ./data.bak
$ ln -s /mnt/shared/orchestrator/mysql_data ./data
$ chown -Rh mysql:mysql ./data
```

6. Confirm that the operation in the previous step was successful by running a list operation on the `mysql` directory.

```
$ ls -lah /opt/aspera/common/mysql
...
lrwxrwxrwx 1 mysql mysql 4 Jun 12 15:25 data -> /mnt/shared/orchestrator/mysql_data
drwxr-x--- 5 mysql mysql 4.0K Jan 18 16:26 data.bak
...
```

7. Create an empty file in the following directory:

```
/opt/aspera/common/mysql/data/
```

Now run the following commands to confirm that this directory comes with `mysql` ownership (rather than nobody ownership):

```
$ cd /opt/aspera/common/mysql/data/
$ sudo -u mysql touch toto.txt
$ ls –l
-rw-r--r-- 1 mysql mysql        0 May 12 08:19 toto.txt
$ rm –f toto.txt
```

8. On one node where the shared storage is accessible, copy the MySQL data into the shared volume.

```
$ cp -R /opt/aspera/common/mysql/data.bak/* /opt/aspera/common/mysql/data
```

9. On the other node, verify that you can see the data files in this directory. It may be necessary to perform a failover to make the shared storage visible.

```
$ ls /opt/aspera/common/mysql/data/
-rw-rw----. 1 mysql mysql 18874368 Apr 23 11:04 ibdata1
-rw-rw----. 1 mysql mysql 19922944 Apr 23 11:04 ib_logfile0
-rw-rw----. 1 mysql mysql 19922944 Apr 22 16:01 ib_logfile1
drwx------. 2 mysql mysql    32768 Apr 22 16:00 mysql
-rw-rw----. 1 mysql mysql     5536 Apr 23 11:04 mysqld.log
drwx------. 2 mysql mysql    32768 Apr 22 17:52 orchestrator
drwx------. 2 mysql mysql    32768 Apr 22 16:00 test
```

Maintenance of the High Availability Environment

The ACM Control Command (acmctl)

Overview of the ACM Control Command (acmctl)

The **acmctl** command controls the ACM. Running it with the **-h** (Help) option displays the available command options:

```
# /opt/aspera/acm/bin/acmctl -h
Aspera Cluster Manager Control Command
Version: 0.2
```

```
Usage: acmctl {option}
List of options:
-i: Display the current state of ACM
-s: Perform a sanity check of ACM
-D: Disable ACM globally
-E: Enable ACM globally
-d: Disable ACM locally
-e: Enable ACM locally
-b: Back up the MySQL database (active node only)
-A: Display information about the version
```

Check the Orchestrator and ACM Status

You can use the **`-i`** option to display the current status of ACM on a node output shown from the `active` node:

```
Checking current ACM status...
Aspera Cluster Manager for Orchestrator - status
------------------------------------------
Local hostname:         orchestrator2
Active node:            orchestrator2 (me)
Status of this node:    active
Status file:            current
Disabled globally:      no
Disabled on this node:  no

Database configuration file
---------------------------
Database host:         localhost

Orchestrator active/active services status
------------------------------------
Apache:                 running

Orchestrator Status:
-> Orchestrator Engine running with pid: 31524
-> Mongrel serving orchestrator on port 3000 is running with pid: 31636
-> Mongrel serving orchestrator on port 3001 is running with pid: 31681
-> Mongrel serving orchestrator on port 3002 is running with pid: 31685
-> Orchestrator Monitor running with pid: 31713
-> Asynchronous Worker Process 0 is running with pid: 31715
-> Asynchronous Worker Process 1 is running with pid: 31717
-> Synchronous Worker Process 2 is running with pid: 31719
-> Synchronous Worker Process 3 is running with pid: 31721
-> Synchronous Worker Process 4 is running with pid: 31723
-> Synchronous Worker Process 5 is running with pid: 31725

Orchestrator active/passive services status
------------------------------------------
MySQL:                 stopped
```

The following is an example of the **`acmctl -i`** output on the `passive` node:

```

Aspera Cluster Manager for Orchestrator - status
------------------------------------------
Local hostname:            orchestrator1
Active node:               orchestrator2
Status of this node:       passive
Status file:               current
Disabled globally:         no
Disabled on this node:     no

Database configuration file
---------------------------
Database host:         localhost

Orchestrator active/active services status
------------------------------------
Apache:                running

Orchestrator Status:
  -> Orchestrator Engine running with pid: 26932
  -> Mongrel serving orchestrator on port 3000 is running with pid: 26967
  -> Mongrel serving orchestrator on port 3001 is running with pid: 26970
  -> Mongrel serving orchestrator on port 3002 is running with pid: 26973
  -> Orchestrator Monitor running with pid: 26979
  -> Asynchronous Worker Process 0 is running with pid: 26981
  -> Asynchronous Worker Process 1 is running with pid: 26983
  -> Synchronous Worker Process 2 is running with pid: 26985
  -> Synchronous Worker Process 3 is running with pid: 26987
  -> Synchronous Worker Process 4 is running with pid: 26989
  -> Synchronous Worker Process 5 is running with pid: 26991

Orchestrator active/passive services status
------------------------------------------
MySQL:                 stopped
```

Data Provided by acmctl -i

Note that on both the active and passive systems, the output of the **`acmctl -i`** command provides useful information about the status of the Orchestrator servers:

Output Element	Definition
Hostname	The name of the local system.
Active node	The name and IP address of the node that is currently the active node.
Status [of] file	Whether the `/opt/aspera/acm/run/acm4orchestrator.status` file is current or has expired. A status of `expired` usually indicates a failover situation. The status file may not be available for a short period during failover, and the `Status file` may be stated as `Unable to find`.
Disabled globally	Answers the question: Is ACM disabled for all Orchestrator servers?
Disabled on this node	Answers the question: Is ACM disabled on this node?
Database host	The system that is currently managing the MySQL database files.
Orchestrator `active/active` service status	The **apache** service should have a status of `running` on both the `active` and `passive` servers. The **`mysqld`**, **`stats-collector`**, **`orchestrator-background-default-0`**, **`orchestrator-background-nodes-0`**, **`orchestrator-background-users-0`**, **`orchestrator-background-users-1`**, and **`orchestrator-background-users-2`** services should all be `running` on the `active` server and `not running` on the `passive` server.

ACM Log Files

By default, ACM creates logging information only in the Linux `syslog` file. However, you may modify this default behavior by editing the `/opt/aspera/acm/bin/acm4orchestrator` file and changing the `LOG_TO_FILE` variable value to “1”. This directs ACM to generate a separate log file – `/opt/aspera/acm/log/acm4orchestrator.log`, which only contains data about ACM functions from either orchestrator server (the `/opt/aspera/acm` directory is shared by both servers).

The following example shows the default values.

```
# cd /opt/aspera/acm/bin
# vi acm4orchestrator
```

```
#! /bin/bash
LOG_TO_FILE=0
LOG_TO_SYSLOG=1
```

In this example, the value of `LOG_TO_FILE` has been changed to “1”; ACM now generates a separate log file.

```
# cd /opt/aspera/acm/bin
# vi acm4orchestrator
#! /bin/bash
LOG_TO_FILE=1
LOG_TO_SYSLOG=1
```

Backing up the Orchestrator Database

Note: Aspera strongly recommends performing a backup of the Orchestrator database on regular basis. If the database is corrupted for any reason, restoring it from a healthy backup is the most (if not only) reliable solution.

Aspera also recommends that backup files be stored on dedicated media, (for example, tape or removable disk) stored at a secure location.

To initiate a backup, add the **`-b`** option to the **`acmctl`** command:

```
# /opt/aspera/acm/bin/acmctl -b
Starting backup
Orchestrator: Backup databases... Database backed up in /opt/aspera/orchestrator/backup/
orchestrator-backup-20151028-163200
done
Compressing SQL files
done
Looking for old backups to remove
Found 0 files(s) modified for the last time more than 15 day(s) ago
Backup procedure complete
```

Aspera recommends that you schedule regular backups of the database. To schedule these, create an entry in **`crontab`** on both Orchestrator servers. For example:

```
# crontab -l
* * * * * /opt/aspera/acm/bin/acm4orchestrator 10.0.71.21 20 > /dev/null 2>&1

30 2 * * 1-5 /opt/aspera/acm/bin/acmctl -b > /dev/null 2>&1
```

The example shown above will run a backup of the orchestrator database at 2:30 AM every weekday of every month. See the **`crontab`** man pages for details about the **`crontab`** file format.

Note: A backup is performed only if executed on the `active` node; running the command on a `passive` node will not create a backup.

When a backup is complete, the utility removes all backup files that are older than the default of 7 days. To modify this default value, edit the `/opt/aspera/acm/bin/acmctl` file and set the BACKUP_MAX_AGE_IN_DAYS= variable to the desired number of days.

The default directory where backup files are placed is `/opt/aspera/orchestrator/backup`, but this may be changed by modifying the BACKUP_DIR variable in the `/opt/aspera/acm/bin/acmctl` file or by replacing the default backup directory with a symbolic link pointing to shared storage.

Working with Capsules in a High Availability Environment

Capsules are Orchestrator patches that are applied to your current installation of Orchestrator to improve functionality, without the need for an upgrade.

Working with Capsules in a High Availability Environment

Click **Engine > Capsule Manager** to open the **Capsule Manager** page, which lists all the capsules on the Orchestrator server.

Note: You will not find capsules that have a file name starting with "_" on the **Capsule Manager** page. These are special "one-time-only" capsules designed to run only once after ingest. You can find them in the list of archived capsules after import. In high availability (HA), these special capsules must be ingested *on every instance of the cluster*.

See the section below, "Working with One-time-only Capsules", for more information.

In an HA environment, multiple engines serve Orchestrator. Each engine can have its own capsules.

Once they are imported, capsules can be viewed, archived to the archive directory, or deleted from the server:

View
Opens up a modal window with the capsule code in read-only mode.

Archive
Prompts the user for confirmation to proceed with the archiving of a capsule.

Note: Archiving a capsule removes it from the capsule directory and moves it to the archive directory. Orchestrator must be restarted to allow the change to take effect.

Delete
Remove the file from the Orchestrator server.

Note: This action is irreversible and Orchestrator must be restarted to allow the change to take effect.

Working with One-time-only Capsules

A capsule whose filename begins with "_" is a "one-time-only" capsule, meaning it gets executed only once, during the ingest phase. Unlike other capsules applied during upgrade that are included in a post-upgrade snapshot, "one-time-only" capsules do not get included in the snapshot. After they are applied, you can find them in `/opt/aspera/var/archive/orchestrator/capsules`, but you will not see on the **Capsule Manager** page (**Engine > Capsules**) or in the active capsules directory, `/opt/aspera/var/config/orchestrator/capsules`.

In general you should re-apply one-time-only capsules after a snapshot import, and there is no risk in doing so. However, if the one-time-only capsule is meant to modify a database table that is part of a snapshot, then there is no need to re-apply it.

Importing a Capsule in the UI

Note: Depending on the type of capsule, you may be directed to import the capsule on just one node, or on both nodes.

1. On one node, click **Engine**, and in the left navigation, click **Capsules**.
2. Click **Import**.
3. In the **Upload Capsule File for Import** dialog, click **Browse**.
4. Select a capsule file (a `.rb` file) and click **Upload**.

Importing a Capsule Manually

Important: In a high availability environment, the directory `/opt/aspera/var/config/orchestrator` is normally shared between all Orchestrator instances in the cluster. If this is not the case, you must —on all Orchestrator nodes— perform the first two steps in this procedure and skip step 3.

Note: There may be a case where you need to import a capsule on one node of a cluster, only—not on the other. In that case, you need to manually copy the capsule to `/opt/aspera/var/config/orchestrator/capsules/engine_`*`id`*, where *id* is the ID of the engine instance where you need to apply the capsule.

1. On one of the nodes, copy the capsule into this folder:

   ```
   /opt/aspera/var/config/orchestrator/capsules
   ```

.

2. Restart Orchestrator:

   ```
   # asctl orchestrator:restart
   ```

3. On all other nodes in the cluster, restart Orchestrator.

Suspending ACM

Disabling ACM on All Nodes

When you perform an upgrade, you must disable ACM on all nodes.

1. To disable ACM on all nodes, run the **acmctl** command with the **-D** option on any of the nodes:

   ```
   $ /opt/aspera/acm/bin/acmctl -D
   ACM is disabled globally.
   ```

   Once ACM is disabled, it will stop performing status file updates and it will transition to the `expired` state. The services are allowed to continue running.

2. Verify the status of ACM with one of the following methods:

   - Run the **acmctl -i** command.

     The `Status file: expired` output below confirms that ACM has been disabled.

     ```
     $ /opt/aspera/acm/bin/acmctl -i
     Checking current ACM status...
     Aspera Cluster Manager for Orchestrator - status
     ------------------------------------------
     Local hostname:         orchestrator1
     Active node:            orchestrator1 (me)
     Status of this node:    active
     Status file:            expired
     Disabled globally:      yes
     Disabled on this node:  no
     Database configuration file
     ---------------------------
     Database host:        localhost

     Orchestrator active/active services status
     ------------------------------------
     Apache:               running

     Orchestrator Status:
       -> Orchestrator Engine running with pid: 29500
       -> Mongrel serving orchestrator on port 3000 is running with pid: 29590
       -> Mongrel serving orchestrator on port 3001 is running with pid: 29593
       -> Mongrel serving orchestrator on port 3002 is running with pid: 29596
       -> Orchestrator Monitor running with pid: 29615
       -> Asynchronous Worker Process 0 is running with pid: 29617
       -> Asynchronous Worker Process 1 is running with pid: 29619
       -> Synchronous Worker Process 2 is running with pid:
     29621
       -> Synchronous Worker Process 3 is running with pid: 29624
       -> Synchronous Worker Process 4 is running with pid: 29626
       -> Synchronous Worker Process 5 is running with pid: 29628

     Orchestrator active/passive services status
     ------------------------------------------
     MySQL:                stopped
     ```

   - If the `acm4orchestrator.log` file has been enabled, look in the file.

     ```
     # tail -f /opt/aspera/acm/log/acm4orchestrator.log
     2016-10-28 16:48:01 (-0800) acm4orchestrator haorchestrator1 (21470): ACM is now disabled
     globally: aborting
     2016-10-28 16:48:01 (-0800) acm4orchestrator haorchestrator1 (14664): ACM is now disabled
     globally: aborting
     ```

3. To re-enable ACM on all nodes, run the **acmctl** command with the **-E** option :

```
$ /opt/aspera/acm/bin/acmctl –E
ACM is enabled globally
```

One of the Orchestrator servers becomes the active node, with all associated services started, and the other node is the passive node.

If the active node calls `acm4orchestrator` before the passive node, it only updates the status file.

If the passive node calls `acm4orchestrator` before the active node, it does not perform an automatic failover, because the status file transitioned into the `current` state when ACM was re-enabled.

Disabling and Re-enabling ACM on One Node

As mentioned in "Disabling ACM on All Nodes" on page 44, disabling ACM does not stop the ACM services which are running when ACM disabled.

Important:

If ACM is disabled locally on the currently active node, the node becomes passive. However, those ACM services still running on the previously active but now passive (locally disabled) node could attempt to access some shared files that are also being accessed by the other Orchestrator nodes that have now become active.

Therefore, you should be careful when locally disabling the active node. After you disable ACM locally, stop all Orchestrator services by running the command **`aspera-orchestrator stop`** on the active node; this avoids the potential problem of conflicting access as described above.

1. To disable ACM for one node only, run **`acmctl`** with the **`-d`** option on that node:

```
# /opt/aspera/acm/bin/acmctl -d
ACM is disabled locally
```

2. To verify ACM's status for a disabled node, run the **`acmctl –i`** command on that node:

```
# /opt/aspera/acm/bin/acmctl -i
Checking current ACM status...
Aspera Cluster Manager for orchestrator - status
------------------------------------------ Local hostname:     haorchestrator1
Active node:     haorchestrator2
Status of this node:     passive
Status file:     current Disabled globally:     no
Disabled on this node:     yes
```

3. To re-enable ACM on a node, run the **`acmctl`** command with the **`-e`** option on the node:

```
# /opt/aspera/acm/bin/acmctl -e
ACM is enabled locally
```

Manual Failover

To force a `passive` node to assume the `active` role, simply disable ACM on the active node and stop all Orchestrator services on that node.

1. On the active node, disable ACM locally.

```
$ /opt/aspera/acm/bin/acmctl –d
ACM is disabled locally
```

2. Stop all services.

```
$ asctl all:stop
Orchestrator is being stopped forcefully:
-> Killing Orchestrator Engine process with pid: 24542
-> Killing Mongrel serving orchestrator on port 3000 with pid: 24638
-> Killing Mongrel serving orchestrator on port 3001 with pid: 24641
-> Killing Mongrel serving orchestrator on port 3002 with pid: 24644
-> Orchestrator Monitor running with pid: 24663
-> Killing Orchestrator Monitor with pid: 24663
-> Killing Asynchronous Worker Process 0 with pid 24665
-> Killing Asynchronous Worker Process 1 with pid 24668
-> Killing Synchronous Worker Process 2 with pid 24670
-> Killing Synchronous Worker Process 3 with pid 24672
-> Killing Synchronous Worker Process 4 with pid 24675
-> Killing Synchronous Worker Process 5 with pid 24677
Apache: Stop... done
MySQL: Stop... done
```

3. Run the **`acmctl -i`** command to verify that all Orchestrator services have been stopped.

   The ACM software checks the `/opt/aspera/acm/run/acm4orchestrator.status` file to determine if the file is older than 2 minutes. If the file is older than 2 minutes, the **`acmctl -i`** command reports the `Status [of] file` as `expired`, which causes ACM to start the failover process to the other Orchestrator server.

   Note: This command can be used to verify that all Orchestrator services have been stopped, even if the actual failover has not yet occurred.

```
$ /opt/aspera/acm/bin/acmctl –i
Checking current ACM status...
Aspera Cluster Manager for Orchestrator - status
------------------------------------------
Local hostname:             orchestrator1
Active node:                orchestrator1 (me)
Status of this node:        active
Status file:                current
Disabled globally:          no
Disabled on this node:      yes

Database configuration file
---------------------------
Database host:        localhost

Orchestrator active/active services status
------------------------------------
Apache:               stopped

Orchestrator Status:
  -> Orchestrator Engine NOT running
  -> Mongrel serving orchestrator on port 3000 is NOT running
  -> Mongrel serving orchestrator on port 3001 is NOT running
  -> Mongrel serving orchestrator on port 3002 is NOT running
  -> Orchestrator Monitor NOT running
  -> Asynchronous Worker Process 0 is NOT running
  -> Asynchronous Worker Process 1 is NOT running
  -> Synchronous Worker Process 2 is NOT running
  -> Synchronous Worker Process 3 is NOT running
  -> Synchronous Worker Process 4 is NOT running
  -> Synchronous Worker Process 5 is NOT running

Orchestrator active/passive services status
------------------------------------------
MySQL:                stopped
```

4. After a minute or so, confirm that the formerly passive node has become active with one of the following methods:

   - Run the **`acmctl –i`** command, as in the following example:

```
$ /opt/aspera/acm/bin/acmctl -i
Checking current ACM status...
Aspera Cluster Manager for Orchestrator - status
------------------------------------------
Local hostname:         orchestrator2
Active node:            orchestrator2 (me)
Status of this node:    active
Status file:            current
Disabled globally:      no
Disabled on this node:  no

Database configuration file
---------------------------
Database host:         localhost

Orchestrator active/active services status
------------------------------------
Apache:                 running
```

```
Orchestrator Status:
-> Orchestrator Engine running with pid: 31524
-> Mongrel serving orchestrator on port 3000 is running with pid: 31636
-> Mongrel serving orchestrator on port 3001 is running with pid: 31681
-> Mongrel serving orchestrator on port 3002 is running with pid: 31685
-> Orchestrator Monitor running with pid: 31713
-> Asynchronous Worker Process 0 is running with pid: 31715
-> Asynchronous Worker Process 1 is running with pid: 31717
-> Synchronous Worker Process 2 is running with pid: 31719
-> Synchronous Worker Process 3 is running with pid: 31721
-> Synchronous Worker Process 4 is running with pid: 31723
-> Synchronous Worker Process 5 is running with pid: 31725

Orchestrator active/passive services status
------------------------------------------
MySQL:                  stopped
```

- If the `acm4orchestrator.log` file has been enabled, review the file.

```
# tail –f /opt/aspera/acm/log/acm4orchestrator.log
2016-10-28 16:48:01 (-0800) acm4orchestrator haorchestrator1 (21470): ACM is now disabled
globally: aborting
2016-10-28 16:48:01 (-0800) acm4orchestrator haorchestrator1 (14664): ACM is now disabled
globally: aborting
```

5. On the newly passive node, re-enable ACM to automatically start the active services.

```
$ /opt/aspera/acm/bin/acmctl –e
ACM is enabled locally
```

This action restarts the `active/active` Orchestrator services in order to provide HA functionality.

6. After a few minutes, verify that the `active/active` services have started on the passive node with the following command:

```
$ /opt/aspera/acm/bin/acmctl -i
Checking current ACM status...

Aspera Cluster Manager for Orchestrator - status
------------------------------------------
Local hostname:            orchestrator1
Active node:               orchestrator2
Status of this node:       passive
Status file:               current
Disabled globally:         no
Disabled on this node:     no

Database configuration file
---------------------------
Database host:         localhost

Orchestrator active/active services status
------------------------------------
Apache:                running

Orchestrator Status:
  -> Orchestrator Engine running with pid: 26932
  -> Mongrel serving orchestrator on port 3000 is running with pid: 26967
  -> Mongrel serving orchestrator on port 3001 is running with pid: 26970
  -> Mongrel serving orchestrator on port 3002 is running with pid: 26973
  -> Orchestrator Monitor running with pid: 26979
  -> Asynchronous Worker Process 0 is running with pid: 26981
  -> Asynchronous Worker Process 1 is running with pid: 26983
  -> Synchronous Worker Process 2 is running with pid: 26985
  -> Synchronous Worker Process 3 is running with pid: 26987
  -> Synchronous Worker Process 4 is running with pid: 26989
  -> Synchronous Worker Process 5 is running with pid: 26991

Orchestrator active/passive services status
------------------------------------------
MySQL:                 stopped
```

Automatic Failover with ACM

The failover mechanism relies on periodic checks of the following shared status file:

```
/opt/aspera/acm/run/acm4orchestrator.status
```

By default, the status file is kept up-to-date by the active node changing its last modification date every minute. The status file expires if there is no update by the active node within the last two minutes.

One of two situations will occur when a status file expires:

- If the active node detects that the status file is expired, it automatically updates it.
- If the passive node detects that the status file is expired, it assumes that the peer node is out of service and performs an automatic failover.

  Note: Automatic failover will only occur if the MySQL database is stopped on the peer node; otherwise, the failover will not be triggered.

It usually takes two to three minutes to detect a failover situation, plus two to three minutes to restart Aspera Orchestrator, depending on the number of workers to run.

Upgrading Orchestrator 2.3.5+ to 4.0.1 in High Availability

This procedure is for upgrading Orchestrator 2.3.5+ to 4.0.1 in a high availability environment.

Important: IBM Aspera supports direct upgrades to the current General Availability (GA) version from only two GA versions prior to the current release. To upgrade to the latest version, you must be within two GA versions of the current version. Upgrading from older version requires upgrading in steps. For example, if you are four GA versions behind, upgrade to two GA versions behind (GA - 2), and then upgrade to the current GA version.

Note: If you are upgrading from an earlier version of Orchestrator (before 2.3.5+), contact Support.

Warning: Prior to performing any upgrade, IBM Aspera strongly recommends customers:

1. Perform a full environment back up and ensure the back up is successful. In case the upgrade fails, the only reliable, short-term fix is to roll back the environment using the back up.
2. Test the upgrade in a test environment comparable to the production environment.
3. If upgrading the test environment is successful, upgrade the production environment, but do not bring the production environment back online.
4. Prior to bringing the production environment back online, the customer must test the application to determine if an immediate rollback is needed. Otherwise, customers risk losing all data generated between upgrade and rollback.

Before upgrading Orchestrator, review product release notes. Aspera strives to maintain compatibility from one release to the next, but sometimes we must make changes that affect a small number of customers. Review the release notes for all Orchestrator versions that have been released since your current version. In particular, the *Breaking Changes* section highlights changes that may require you to adjust your workflow, configuration, or usage. These issues may impact your new installation.

I. Preliminary Steps: Backing up Orchestrator Data and Stopping the Services

Before upgrading the Orchestrator server on each node of the cluster, you must back up all Orchestrator data so that you can restore them after upgrade, if needed.

See "Orchestrator Directory Locations" on page 224 for more information about the directory locations referred to in this procedure.

Before you begin the upgrade, follow these backup steps *for both nodes in the cluster*.

Note: All commands in the following procedure must be performed as `root`, or an equivalent superuser.

1. Create a backup of MySQL data with a MySQL dump.

```
$ /opt/aspera/common/mysql/bin/mysqldump -h localhost –u root –p orchestrator > /tmp/
orchestrator_db_backup_timestamp.sql
```

Important: Confirm that the `dump.mysql` file is created (and that it is > 0 in size) and save the file for data recovery, as needed.

Note: The generated `.sql` file may take several gigabytes of disk space depending on the database size, so make sure that enough space is available on the targeted partition (`/tmp` in the example above) before running the backup.

2. Run a backup of Orchestrator data with the Snapshot feature.

   See "Create Snapshot" on page 80 for details.

   Note: Move all backup files onto a safe server after running the backup so they don't take up space on Orchestrator.

3. Back up any important configuration files which are located in the `config` directory, since this directory is overwritten with new data during an upgrade.

   On each node, copy the following files to a safe location, such as `/tmp`:

```
/opt/aspera/orchestrator/config/orchestrator.yml
/opt/aspera/orchestrator/config/orchestrator-license*
/opt/aspera/orchestrator/config/database.yml
```

4. Copy any custom files — such as `aspera_logo_simple.jpg` and `aspera.css` — to a safe location.

   During the software installation phase of an upgrade, custom images, branding files and style sheets are copied from `/opt/aspera/var/config/orchestrator/` to `/opt/aspera/orchestrator/public/`. Thus make sure that any custom files (as well as `orchestrator.yml`, if the default Orchestrator configuration has been changed) are placed under `/opt/aspera/var/config/orchestrator/` before doing the upgrade.

```
/opt/aspera/orchestrator/public/images/aspera_logo_simple.jpg
/opt/aspera/orchestrator/public/stylesheets/aspera.css
```

5. Back up the `var` directory.

   Although this directory is preserved during upgrade, it is important to take the precaution.

```
$ cd /opt/aspera/ ; tar chvfz /tmp/aspera_orig_timestamp.tar.gz var/
```

6. Check the Aspera Cluster Manager (ACM) status to locate the active node:

```
$ /opt/aspera/acm/bin/acmctl –i
Checking current ACM status...
Aspera Cluster Manager for Orchestrator - status
------------------------------------------
Local hostname:         orchestrator1
Active node:            orchestrator1 (me)
Status of this node:    active
Status file:            current
Disabled globally:      no
Disabled on this node:  no

. . .

Orchestrator active/passive services status
------------------------------------------
MySQL:                running
```

   In the above STDOUT, the `mysql` is running on the active server.

7. On both Orchestrator instances, disable ACM.

```
$ /opt/aspera/acm/bin/acmctl -d
```

8. Copy or archive the shared storage that all the symlinks are pointing to.

   For example, you may might create the symlink with this command:

```
ln -s /mnt/shared/orchestrator/mysql_data/ ./data
```

In that case, you would need to back up this directory:

```
/mnt/shared/orchestrator/
```

Important: You may need this directory location as a reference for restoring your configuration.

II. Upgrading the Orchestrator Servers

1. On the machine where you will install Orchestrator, download the installation files for the new release.

   Go to IBM Fix Central, https://www.ibm.com/support/fixcentral/swg/selectFixes?parent=ibm~Other%20software&product=ibm/Other+software/IBM+Aspera+Orchestrator&release=All&platform=All&function=all.

   Click the link for the current release of Orchestrator, then download the installers for `aspera_orchestrator` and `ibm_aspera_common`.
2. Stop the applicable services and processes.

   First, check the release notes for the current Orchestrator release to confirm the required version of the Aspera Common Components.

   - If you do not need to upgrade the Aspera Common Components, run this command:

     ```
     $ asctl orchestrator:stop
     ```
   - If you need to upgrade the Aspera Common Components, run this command:

     ```
     $ asctl all:stop
     ```
3. Run the following command to confirm that the services and processes are stopped.

   ```
   $ asctl all:status
   ```
4. On both nodes, uninstall Orchestrator and the Aspera Common Components. Check the release notes to confirm the required versions.

   Note: Because of the upgrade of the database version from MySQL 5.1 to 5.7 in Orchestrator 4.0, you need to uninstall the previous versions of Orchestrator and Aspera Common before you upgrade Orchestrator.

   ```
   # rpm -e aspera-common
   ```

   ```
   # rpm -e aspera-orchestrator
   ```
5. On both nodes, clean up directories from the previous installation before the fresh install of Orchestrator.

   Delete these directories:

   ```
   rm -r /opt/aspera/common
   rm -r /opt/aspera/orchestrator
   ```
6. On both nodes, upgrade with a clean installation of the Aspera Common Components, followed by Orchestrator.

   Note: Because of the upgrade of the database version from MySQL 5.1 to 5.7 in Orchestrator 4.0, you need to install the new versions of Orchestrator and the Aspera Common Components.

   ```
   # rpm -i ibm-aspera-common-1.2.29.180105-0.x86_64.rpm
   ```

   ```
   # rpm -i aspera-orchestrator-4.0.version-0.x86_64.rpm
   ```
7. Complete the upgrade by setting up Orchestrator.

   ```
   # asctl orchestrator:setup
   ```

Troubleshooting:

When you run `asctl orchestrator:setup`, the process might stop at `Starting MySQL`. To fix the issue, unlink the `data` directory in `mysql` and recreate the symbolic link after running `asctl orchestrator:setup` by running the commands in the example below.

```
# cd /opt/aspera/common/mysql
# unlink data
# mv data,bak data
# asctl orchestrator:setup
# mv data data.bak
# ln -s /mnt/shared/orchestrator/mysql_data/ ./data
```

8. Restart your browser (recommended) or empty the browser cache.
9. Confirm that the new installation of Orchestrator works on both servers by logging into the Orchestrator UI.
10. On both nodes, reload the MySQL database using the database dump you created in "Preliminary Steps: Backing up Orchestrator Data".

```
/opt/aspera/common/mysql/bin/mysql -h 127.0.0.1 -P 4406 -u <user> -p orchestrator < /tmp/
orchestrator_db_backup_timestamp.sql
```

Troubleshooting tip: You may see an error similar to this:

```
ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/tmp/mysql.sock' (2)
```

Re-run the command with `socket=`, for example:

```
# /opt/aspera/common/mysql/bin/mysql -h localhost -u root --socket=/opt/aspera/common/
mysql/var/run/mysql.sock -p orchestrator < /tmp/dump1.sql
```

11. On both nodes, migrate the MySQL database back to the schema for Orchestrator 4.0.

Run these commands:

```
# cd /opt/aspera/orchestrator
# script/cmd.sh
Loading development environment (Rails 4.2.7)
>> Database.generate
```

12. Check the following directory if there are capsules present in the folder.

```
/opt/aspera/var/config/orchestrator/capsules/
```

Remove all capsules that are specific to the previous version of Orchestrator (in other words, the version of Orchestrator that you are upgrading to the current version). Repeat the process on the other node.

13. Consult these articles for any additional required steps:
    - “Installing and Configuring Operating Systems” on page 27
    - “Installing Orchestrator in High Availability Environments” on page 30.
    - “Setting up Aspera Cluster Manager” on page 31
    - “Configuring MySQL and Aspera Services” on page 36 (See the articles in the section.)

III. Post-Upgrade: Restoring Files and Re-enabling Services

1. After upgrade, copy the following files from the backup location (for example, `/tmp`) to the original locations on both instances.
    - Copy `orchestrator.yml`, `orchestrator-license*`, and `database.yml` to the following directory:

```
/opt/aspera/orchestrator/config/
```

- If found, copy `aspera_logo_simple.jpg` to the following:

  ```
  /opt/aspera/orchestrator/public/images/
  ```
- If found, copy `aspera.css` to the following:

  ```
  /opt/aspera/orchestrator/public/stylesheets/
  ```

2. Ensure that the contents of the following files are identical.

   ```
   /opt/aspera/orchestrator/config/database.yml
   /opt/aspera/var/config/orchestrator/database.yml
   ```

   If the files are not identical, use the values in the following file:

   ```
   /opt/aspera/var/config/orchestrator/database.yml
   ```
3. Perform an ACM sanity check on both `orchestrator` instances:

   ```
   /opt/aspera/acm/bin/acmctl -s
   ACM sanity check
   ----------------
   Checking if an entry for ACM seems to exist in the crontab            OK
   Checking that the Orchestrator master service is disabled in chkconfig
   Note: This output shows SysV services only and does not include native
        systemd services. SysV configuration data might be overridden by native
        systemd configuration.f

        If you want to list systemd services use 'systemctl list-unit-files'.
        To see services enabled on particular target use
        'systemctl list-dependencies [target]'.

   KO
   Checking that SE Linux mode is not set to enforcing                OK
   ```

   If you get any "KO" in the response, as in the example above, you need to address it. In the above example you need to rerun the following:

   ```
   # chkconfig service_name off
   ```

   For example:

   ```
   # chkconfig AsperaOrchestrator off
   # chkconfig aspera_mysqld off
   # chkconfig aspera_httpd off
   ```

   Now rerun the ACM sanity check:

   ```
   # acmctl -s
   ACM sanity check
   ----------------
   Checking if an entry for ACM seems to exist in the crontab            OK
   Checking that the Orchestrator master service is disabled in chkconfig
   Note: This output shows SysV services only and does not include native
        systemd services. SysV configuration data might be overridden by native
        systemd configuration.

        If you want to list systemd services use 'systemctl list-unit-files'.
        To see services enabled on particular target use
        'systemctl list-dependencies [target]'.

   OK
   Checking that SE Linux mode is not set to enforcing                OK
   ```

   When everything is confirmed as "OK", as in the example above, you can proceed to the next step.
4. Enable ACM on both instances.

   Note: Before you enable ACM, check the share device ID (per the example later in this step).

   ```
   $ /opt/aspera/acm/bin/acmctl -e
   ```

ACM will start MySQL, Apache and Aspera Orchestrator on one instance and Apache and Orchestrator on another instance.

Note: The following command checks for the share device ID (`40` in this example).

```
$ * * * * /opt/aspera/acm/bin/acm4orchestrator 10.0.154.74 40 > /dev/null 2>&1
```

After reboot this device ID might change. Find the device ID with a command like this:

```
$ stat -c "%d" /mnt
```

In the example above, the expression `/mnt` is an example of a mount point for the shared storage. location.

Create the mount on `/mnt` to the shared storage:

```
mount -v -o vers=3,proto=tcp,port=2049 shared_storage mount_point
```

For example:

```
$ mount -v -o vers=3,proto=tcp,port=2049 ha_storage.aspera.us /mnt
```

IV. Upgrading Plugins and Portlets After Upgrade of Orchestrator

During an upgrade, plugins are not upgraded automatically and new portlets are not enabled by default; the steps below will allow you to update these as needed.

Note: On one instance only, upgrade the plugins as necessary.

1. Go to **Engine > Plugins** and click **Upgrade Available** to see the newer plugins.

   Each plugin contains a Revision History screen providing details about the newer plugin. After reviewing the revision history, upgrade the plugin and force maintenance. For more details on plugin installation and upgrading, see
2. Find out if any new plugins are not already enabled by selecting **Not Enabled**, and enable them if necessary.

   Note: In a production system, upgrade the new plugins individually only when necessary (compare the previous version with the current released version). It is advisable to test them in pre-production first to make sure that the new plugins operate well with the existing workflows.
3. Click **Engine > Plugins > Reload Plugins**, and then click **Engine > Processes > Force Maintenance** to set the new code.
4. If the new portlets are not enabled, and they are needed, click **Engine > Portlets > Enable all**.
5. Test your workflows to make sure they function properly after the upgrade.

V. Importing Plugins and Portlets After an Upgrade of Orchestrator

When importing plugins or portlets in a high availability setup (at any time after the upgrade), do one of the following with the plugin files and portlet files:

- **Recommended:** Import the files on both Orchestrator servers. This loads the files on both servers and ensures that the UI shows the correct version.
- Import the files on one server and reload the same files on the other.

1. On **Engine > Plugins**, click **Enable All**.
2. On **Engine > Processes** click **Force maintenance**.
3. On **Engine > Plugins**, click **Reload All**.
4. On **Engine > Plugins**, click **Disabled Unused**.

   This step ensures improved performance.

Getting Started with Orchestrator

The following steps outline the process of creating a workflow, configuring its steps, configuring workflow logging and monitoring, and launching a workflow as a work order.

Orchestrator uses workflows, composed of a series of connected steps, to automate your content collection, processing, and distribution.

These steps assume that Orchestrator is already installed and running; for instructions, see the IBM Aspera Orchestrator Admin Guide.

1. Log in to the Orchestrator UI.

   Orchestrator workflows, work flow steps, and logging can only be configured in the Orchestrator web UI.
2. Create a workflow or open a workflow template.

   Workflows are based on an arrangement of steps that control content flow/data processing between computers. The actions performed in a workflow can include ingest, processing, transfer, and output of data. Actions can run automatically in response to triggering events or can be managed through manual oversight. Each workflow can be configured for success, failure, and error outcomes. A workflow can connect to multiple sub-workflows that are triggered by particular conditions.

   - For information about workflows, see "Workflows" on page 103. To see sample workflows, go to "The Orchestrator Help Page" on page 66.
   - Remote nodes (servers) involved in a workflow can use Aspera software for file transfer, or they can be connected to Orchestrator with SSH or an IP address. For information, see "Remote Nodes" on page 85.
3. Configure plugins within the workflow.

   Workflow steps consist of action plugins. To add a plugin to a workflow, you can drag-and-drop it from the plugin list into your workflow editing space. Before you can save the workflow, the plugin must be configured and saved as a new template. The configuration options vary, depending on the action.

   For example, you can use the Ascp Client Action Plugin to transfer files as an Aspera Client between the Orchestrator computer and an Aspera server, and must enter the IP address of the server, transfer user credentials, and transfer specifications. You can trigger this plugin by preceding it with and connecting it to the Aspera Node File Watcher Action Plugin, which detects when a specific file or file pattern is present on the Aspera server. Once the transfer is complete, you can perform local file operations.

   - For more information about using plugins, see "Plugins" on page 88 and "The Orchestrator Help Page" on page 66.
   - For information about plugin templates, see "Plugin Templates for Workflows" on page 101.
4. Configure run-time logs (journaling).

   Journals are run-time logs that track the progress of files (assets) through a workflow. Journal books consolidate journal log data for specific sources or actions; for example, to track files sourced from a particular vendor through the entire workflow, or to track all asset ingest steps across multiple workflows. You can create multiple journal books for a workflow; conversely, a single journal book can be used across multiple workflows.

   To use journals and journal books, you assign individual plugins within a workflow to specific journals or journal books. Journal books must be created before a plugin can be assigned to them. For information about journal books, see "Working with Journal Books" on page 116. For instructions about assigning workflow steps to journals and journal books, see "Working with Journals" on page 116.
5. Configure Orchestrator monitoring.

You can create monitors with which you can monitor the status of workflows and specific areas of the file system. For example, you can configure email alerts in order to know as soon as a workflow or folder enters an "Alerts" or "Warning" status.

You can also create a custom dashboard (portal page), from which you have easy access to Orchestrator system and workflow information, start work orders, or interact with other Aspera products. These views and actions are added as portlets; Orchestrator comes with many portlets, and you can also create your own custom portlets.

- For more information about using monitors, see "Monitors" on page 120.
- For more information about creating dashboards and portlets for your dashboards, see "Portal Pages and Portlets" on page 128.

6. Run the workflow by creating a work order.

   The workflow defines the series of actions to take, and you start a workflow by creating and launching a work order. In the work order, you select the workflow you want to run, enter any required parameters, and click start.

   - For instructions for working with work orders, see "Work Orders" on page 113.
   - You can also manage work orders through the Orchestrator API. For instructions, see "API Endpoints—Work Orders" on page 188.

7. Monitor workflow progress.

   Once a work order is launched, you can monitor the progress of a workflow in several ways:

   - If monitors were configured, you might receive email notifications triggered by workflow events.
   - You can follow along on a customized dashboard.
   - You can view journals or journal books to which workflow steps have been assigned.
   - You can view the event queue and adjust event priority. The queue displays pending workflow actions, and these can be re-prioritized in the **Manage Queues** page. For more information, see "Queues" on page 133
   - You can use the Orchestrator API to check the status; see "API Endpoints—Workflows" on page 206 and "API Endpoints—Work Orders" on page 188.

Product Configuration

This topic describes all the options for configuring Orchestrator.

This topic provides guidance for the following:

- Editing the global configuration parameters
- Enabling plugins
- Enabling portlets
- Sending email notifications within the workflows
- Enabling the journaling feature
- Defining users and permissions
- Enabling remote access capabilities
- Adding remote nodes
- Adding Aspera FASP transfer capability

1. Edit the global configuration parameters.

   You can customize your environment by editing configuration parameters in `orchestrator.yml` or the Orchestrator UI. For more detailed information, see "Editing the Orchestrator Configuration" on page 71.

2. Enable plugins.

All Aspera transfer plugins, as well as some commonly used plugins from third-party providers, are enabled by default. You must enable other plugins you plan to use.

To enable other plugins you plan to use, click **Engine > Plugins > Reload Plugins**, then click **Engine > Processes > Force maintenance**. Complete the process by clicking **Engine > Plugins > Enable all plugins**, then clicking **Yes**.

3. Enable portlets.

   To enable the use of portlets in the Dashboard, click **Engine > Portlets > Enable all**.
4. To send email notifications within the workflows, configure the SMTP servers.

   Click **Engine > Mailer Configuration > New mail server**. To test the configuration, click **Send test email**.
5. In Orchestrator, journals collect run-time data about where a file can be found in the workflow. To enable the journaling feature, click **Workflows > Journals**; under Saved Configurations, click **Activate**.

   For further details about journals, see "Overview: Journals and Journal Books" on page 116.
6. Define users, groups, permissions, and passwords.

   See articles in "Managing Users" on page 67 for details.
7. If you plan to use remote access capability with remote file operation and remote file watcher plugins, install IBM Aspera Point-to-Point on the remote node.

   It is necessary to install Point-to-Point because the Aspera **`ascmd`** executable is required.
8. Add remote nodes, if needed.

   For a detailed procedure, refer to the article on creating a remote node in .
9. If you plan to use the Aspera FASP transfer capability, install Point-to-Point and a standard Point-to-Point license on the controlled node (the one which initiates the file transfers).

   The controlled node may be Orchestrator itself.

   a) On the controlled node, open the configuration file (`aspera.conf`):

   ```
   /opt/aspera/etc/aspera.conf
   ```

   b) Check to see if the file contains `persistent-store` lines such as the following:

   ```
   <central_server>
       <port>40001</port>
       <address>127.0.0.1</address>
       <persistent_store>enable</persistent_store>
       <persistent_store_max_age>259200</persistent_store_max_age>
       <persistent_store_on_error>ignore</persistent_store_on_error>
   </central_server>
   ```

   If they do not already exist, add them.

   c) If `aspera.conf` has been changed in the above steps, restart Aspera Central.

   d) Add a remote node for the controlled node with these elements: `Node type = Aspera FASP`; `ssh port = 33001`; and your `root` login information.

   For a detailed procedure, see the article on creating a remote node in "Creating a Remote Node" on page 86.

Database Configuration and Maintenance

This section describes the optimal configuration and maintenance for databases connected to Orchestrator.

Reclaiming Disk Space Used by Orchestrator

If you find that your IBM Aspera Orchestrator database is filling up too quickly, you can free up space by manually cleaning up database tables and fine-tuning your database configuration.

Cleaning up Database Tables

1. Stop Orchestrator.

```
# asctl orchestrator:stop
```

2. Run the following command to open the MySQL console:

```
# /opt/aspera/common/mysql/bin/mysql --password=mysql_password
```

3. Direct MySQL to use `orchestrator` as the default database:

```
> use orchestrator;
```

4. Remove bkp tables, as needed.

   You can remove existing tables with bkp in the name, because these are created by snapshot imports.

```
> drop table if exists bkp_table;
```

5. Prune tables with the following commands.

   Note: If the table is in a particularly poor condition, each command may take longer than expected.

```
> optimize table variables;
> optimize table prerequisites;
> optimize table work_inputs;
> optimize table work_outputs;
> optimize table work_steps;
> optimize table work_orders;
```

6. Exit MySQL.

```
> exit;
```

7. Restart Orchestrator.

```
# asctl orchestrator:start
```

Configuring the Database

The procedure below describes how to set cutoff values for work orders at both the global and individual level.

Important: Making a drastic reduction to the cutoff values when the database is nearly full—for example, changing the cleanup cutoff from 30 days to 10 days—may put a strain on the system and thus degrade performance. In other words, Orchestrator will slow down because it has too much database deletion to do all at once. IBM Aspera recommends that you decrease these configuration values gradually to avoid an excessive load on Orchestrator, monitoring the system throughout this process.

1. Configure global database purge and cleanup cutoffs.

   You can reduce the amount of data stored in the database by configuring the number of days after which work orders should be purged or cleaned, using the `purge_cutoff` and `cleanup_cutoff` parameters. By default `purge_cutoff` is set to 60 days, and `cleanup_cutoff` is set to 30 days.

   - `purge_cutoff` applies to work orders with the following statuses:
     - STATUS_CANCELED
     - STATUS_COMPLETE
     - STATUS_ROLLEDBACK
     - STATUS_DESTROYED
   - `cleanup_cutoff` applies to work orders with the following statuses:
     - STATUS_ERROR
     - STATUS_ROLLEDBACK

  - STATUS_CANCELED
  - STATUS_COMPLETE
  - STATUS_FAILED
  - STATUS_DESTROYED
  - STATUS_CLEARED

   Edit the values of `purge_cutoff` and `cleanup_cutoff` in the `orchestrator.yml` configuration file:

   ```
   /opt/aspera/orchestrator/config/orchestrator.yml
   ```

   Alternatively, you can edit these values by going to **Engine > Configuration** in the Orchestrator UI.
2. Configure work order-level cutoffs.

   Individual work orders can be removed from the database prior to the global cutoff values defined in step 1.

   - Workflows: Click **Workflows**, then click the **More** icon to the right of a workflow and select **Edit core parameters**. In the dialog, enter the purge cutoff value in the **Purge after (days)** field, and enter the cleanup cutoff in the **Clean up after (days)**.
   - Work Orders: Click **Work Orders** and follow the same process as above for the work orders you need to configure.

Installing and Configuring MySQL on a Separate (Remote) Machine

You can install your MySQL database on a different machine from the one where you install Orchestrator. See "Installing Orchestrator with MySQL on the Same Server" on page 2 for the additional steps you need to run before installing your Orchestrator instance.

Migrating Data from a Current Orchestrator Instance to a New One

You can migrate data from an existing Orchestrator instance to a newly-installed Orchestrator instance.

Current Instance and New Instances that Have the Same Version of Orchestrator

If the (existing) source and (new) target instances have the same installed version of Orchestrator, migrate the database using one of these methods:

- Create a snapshot from the source, then import (restore) that snapshot on the target. Use the procedure in "Create Snapshot" on page 80, followed by the procedure in "Restore Orchestrator from a Snapshot" on page 139.

  Important: This method will not preserve existing work orders.
- Run a backup of the database using a `mysqldump` export on the source, then run a database `mysql` import on the target. This method preserves the work orders but may take a long time if the database is large.

  To run the backup:

  ```
  /opt/aspera/common/mysql/bin/mysqldump -h IP_address -u root -p orchestrator > /tmp/
  orchestrator_db_backup_timestamp.sql
  ```

  To run the import:

  ```
  /opt/aspera/common/mysql/bin/mysql -h IP_address -u root -p orchestrator < /tmp/
  orchestrator_db_backup_timestamp.sql
  ```

  If necessary, manually copy the configuration files from source to target.

Current Instance and New Instances that Have the Different Versions of Orchestrator

If the source and target Orchestrator instances do not have the same Orchestrator versions (for example, the target is on higher version), choose one of these options:

- Follow the same procedure as above, then run `asctl orchestrator:setup` after the database migration.
- Create a new installation on the target instance with the same version as the source instance, then do an upgrade to the higher version of Orchestrator on the target instance.

Troubleshooting Orchestrator Startup Operations

The Orchestrator log can be found in the following default location:

```
/opt/aspera/var/run/orchestrator/log/orchestrator.log
```

The log file is rotated to prevent it from growing too large. For more information, see "Managing Your Orchestrator Logs" on page 60.

MySQL Issues

Orchestrator relies on the proper functioning of the MySQL database. If the MySQL database is not started, Orchestrator won't start. The following are reasons that a user might not be able to start the MySQL database:

- The user doesn't have permissions to run MySQL.
- The disk drive is full.
- The following conditions are true for MySQL or the user who started MySQL:
  - Doesn't have the rights to write into `mysql` logs.
  - Cannot write to the disk with `temp` files (PID files, SOCK files).

License Files Issues

Orchestrator won't start if the license file is absent or expired. Look for messages in the log file that indicate this issue. The footer in the Orchestrator UI also gives an indication of when a license file expires.

Troubleshooting

Troubleshooting Orchestrator Startup Operations

The Orchestrator log can be found in the following default location:

```
/opt/aspera/var/run/orchestrator/log/orchestrator.log
```

The log file is rotated to prevent it from growing too large. For more information, see "Managing Your Orchestrator Logs" on page 60.

MySQL Issues

Orchestrator relies on the proper functioning of the MySQL database. If the MySQL database is not started, Orchestrator won't start. The following are reasons that a user might not be able to start the MySQL database:

- The user doesn't have permissions to run MySQL.

- The disk drive is full.
- The following conditions are true for MySQL or the user who started MySQL:
  - Doesn't have the rights to write into `mysql` logs.
  - Cannot write to the disk with `temp` files (PID files, SOCK files).

License Files Issues

Orchestrator won't start if the license file is absent or expired. Look for messages in the log file that indicate this issue. The footer in the Orchestrator UI also gives an indication of when a license file expires.

Managing Your Orchestrator Logs

Orchestrator activity logs are useful for internal troubleshooting, as well as for sharing with IBM Aspera for technical support. You can set a policy for how Orchestrator retains log files.

You can access the logs:

- from the directory `/opt/aspera/var/run/orchestrator/log/`. This directory also contains logs from previous days.
- in the Orchestrator UI, by clicking **Engine**, then clicking **Log Viewer** in the left navigation.

The file `orchestrator.log` always contains the most recent log data.

For an explanation of how Orchestrator retains activity logs, and how to set a retention policy, see "Log Rotation in Orchestrator".

Log Rotation in Orchestrator

To prevent your log file from growing too large, Orchestrator automatically rotates your log data by date. Alternatively, you can configure Orchestrator to rotate your log data by maximum file size.

The file `orchestrator.log` always contains the most recent log data. By default, the data in this file rotates into a new log file at the beginning of each day. Log files that are rotated by date is in the format `orchestrator.`*`year_month_day`*`.log`; for example, `orchestrator.20200210.log`, `orchestrator.20200211.log`, and so on.

To rotate your log data by a maximum file size (in MB), instead of by date, set a value for `max_log_size_mb` in the Orchestrator configuration. Log files that are rotated by maximum file size are in the format `orchestrator.`*`sequential_log_file_number`*`.log`; for example,`orchestrator.1.log`, `orchestrator.2.log`, and so on.

For either method—rotating logs by date or by maximum file size—Orchestrator controls the total number of log files kept in the system at any given time by deleting them after they have been there for a maximum number of days, or have reached a maximum number of files. You can configure the parameter `log_retention_days` to set a different maximum retention value.

See "Editing the Orchestrator Configuration" on page 71 for more information on setting `max_log_size_mb` and `log_retention_days`.

Troubleshooting for High Availability Setups

Issue: MySQL "not starting" Error

You may see this error;

```
MySQL: Start...not starting
'Error executing '/opt/aspera/common/mysql/share/mysql/mysql.server' start' ().
Starting MySQL.................................................................
```

- Most of the time, it's because of incorrect syntax in the `/opt/aspera/orchestrator/config/database.yml` file. This is an example of a correct `database.yml`:

```
production:
  host: 127.0.0.1
  primary_host: 10.68.62.132
  alternate_host: 10.68.62.133
  port: 4406
  adapter: mysql2
  username: root
  password: aspera
  database: orchestrator
  checkout_timeout: 1
  pool: 10
```

- There may be a permission issue on the `/opt/aspera/common/mysql/data/` folder. The ownership needs to be `mysql:mysql`, not `root:mysql`:

```
# ll /opt/aspera/common/mysql/data/ -d
drwxr-x--- 7 mysql mysql 4096 Mar 10 16:37 /opt/aspera/common/mysql/data/
```

- Confirm that the UID and GID of the `mysql` user account is the same on both Orchestrator instances. Also, confirm that the `mysql` user is part of the `mysql` group:

```
# id mysql
uid=998(mysql) gid=997(mysql) groups=997(mysql)
```

- Make sure you can write in the `/opt/aspera/common/mysql/data/` folder with `mysql` rights:

```
# sudo -u mysql touch /opt/aspera/common/mysql/data/file.txt
```

Issue: "Unable to acquire lock" Error

You may see this error:

```
aslockfile: Unable to acquire lock on /opt/aspera/acm//run/acm4orchestrator-lock (err 36865)
```

- Open the directory `/opt/aspera/acm/run/`:

```
# ll /opt/aspera/acm/run/
-rw-r--r-- 1 root root 76 Mar 16 17:54 acm4orchestrator-lock.lck
-rw-r--r-- 1 root root 34 Mar 16 16:36 acm4orchestrator.status
```

  The lock file, `acm4orchestrator-lock.lck` should not be present if `acm4orchestrator` is not running on `crontab` or on the command line. It is normally released after each call to `acm4orchestrator`, when the parent process kills the dependent `aslockfile` process.
- To test the lock file acquisition and release, comment out the `acm4orchestrator` entry in `crontab`:

```
# crontab -e
# * * * * * /opt/aspera/acm/bin/acm4orchestrator 37.58.112.102 > /dev/null 2>&1
```

- Run `acm4orchestrator` from the command line:

```
# /opt/aspera/acm/bin/acm4orchestrator 37.58.112.102
```

  If you get the same error, check the running processes (on both machines):

```
# ps -ef | grep -i acm
# ps -ef | grep -i lockfile
```

- Kill the existing processes if any with `kill -9` and retry `acm4orchestrator` on the command line.
- Make sure that `/dev/null 2>&1` is present at the end of the `crontab` entry. A regular entry is like:

```
* * * * * /opt/aspera/acm/bin/acm4orchestrator 37.58.112.102 > /dev/null 2>&1
```

Effective Database Maintenance

If you have a high volume of transactions in your workflows, you need to continually monitor and configure you database.

To ensure that your database does not fill up too quickly and cause a negative impact on performance, see "Reclaiming Disk Space Used by Orchestrator" on page 56.

Working with SAML

This section describes configuration and usage for SAML in your Orchestrator setup.

SAML and Orchestrator

IBM Aspera Orchestrator 3.0.4 supports Security Assertion Markup Language (SAML) 2.0, an XML-based standard that allows secure web domains to exchange user authentication and authorization data. With the SAML model, you can configure Orchestrator as a SAML *online service provider (SP)* that contacts a separate online *identity provider (IdP)* to authenticate users. Authenticated users can then use Orchestrator to access secure content.

With SAML enabled, Orchestrator redirects a user to the IdP sign-on URL. The user signs in with the IdP and the IdP sends a SAML assertion back to Orchestrator, which grants the user access to Orchestrator. When a SAML user logs in to Orchestrator for the first time, Orchestrator automatically creates a new user account based on the information provided by the SAML response. Any changes subsequently made to the account on the DS server are not automatically picked up by Orchestrator. For more information about user provisioning for SAML users, see "User Accounts Provisioned by Just-In-Time (JIT) Provisioning" on page 63.

IdP Requirements

To use SAML with Orchestrator, you must already have an identity provider (IdP) that meets the following requirements:

- Supports SAML 2.0
- Able to use an HTTP POST Binding.
- Able to connect to the same directory service that Orchestrator uses.
- Not configured to use pseudonyms.
- Can return assertions to Orchestrator that include the entire contents of the signing certificate.
- If prompted, set to sign the SAML response. (Signing the SAML assertion is optional.)

Configure the SAML IdP

Before configuring SAML in Orchestrator, make sure you configure your IdP to send a correct SAML response to Orchestrator. For more information, see "Configuring Your Identity Provider (IdP)" on page 63.

For instructions on configuring SAML, see "Configuring SAML in Orchestrator" on page 64.

SAML and Directory Services

Orchestrator supports the use of both SAML and directory services. If you configure both services to Orchestrator, ensure the services use different Active Directory domains. Aspera advises against configuring LDAP directly to Orchestrator if the SAML IdP acts as a frontend for the same Active Directory domain.

Bypassing the Default SAML IdP

Orchestrator provides a mechanism for users to bypass the SAML redirect and log in using a local username and password. This feature allows admins to correct server settings, including a mis-configured SAML setup, without logging in through SAML.

To bypass the SAML login, add `logon?local=true` to the end of the login URL. For example:

```
https://10.0.0.10/aspera/orchestrator/logon?local=true
```

User Accounts Provisioned by Just-In-Time (JIT) Provisioning

When a SAML user logs in to IBM Aspera Orchestrator for the first time, Orchestrator automatically creates a new user account based on the information provided by the SAML response. If the SAML response also contains group information, and that group does not yet exist in Orchestrator, Orchestrator automatically creates a new SAML group for each group of which the user is a member.

Group Permissions

A SAML user belonging to multiple groups is given the permissions and settings of all groups it belongs to with permissions overriding restrictions. For example, if Group A disallows sending to external users but Group B does not, users who belong to both groups are allowed to send to external users. Settings that require specific handling are as follows:

- Account expiration is only enabled if all groups to which a user belongs specify account expiration. If account expiration is enabled, the expiration date is set to the latest expiration date from among all groups.
- For any settings that use **Server Default**, **Yes** or **Allow**, and **No** or **Deny**, the setting is set to **Yes** if any group specifies **Yes**, and it is set to **No** if all groups are set to **No**. Otherwise, it is set to use the server default.
- For advanced transfer settings, override is enabled if all groups specify override or if any group specifies any transfer rate that is higher than the server default. If override is enabled, each transfer rate is set to the higher of the highest value from among the groups and the server default. The minimum rate policy is locked only if all groups specify the setting.

Configuring Your Identity Provider (IdP)

IdP Requirements

To use SAML with Orchestrator, you must already have an identity provider (IdP) that meets the following requirements:

- Supports SAML 2.0
- Able to use an HTTP POST Binding.
- Able to connect to the same directory service that Orchestrator uses.
- Not configured to use pseudonyms.
- Can return assertions to Orchestrator that include the entire contents of the signing certificate.
- If prompted, set to sign the SAML response. (Signing the SAML assertion is optional.)

IdP Metadata Formats

You must configure formats to set up your IdP to work with Orchestrator:

Tag	Format
NameID Format	`urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`

If the IdP is capable of reading SAML XML metadata for a service provider, you can upload a saved XML metadata file to configure the IdP. You can retrieve the XML metadata for an existing Orchestrator. Do the following:

1. At the top-right, click the **Admin** dropdown.
2. Click the **Metadata** button. An XML file opens. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
entityID="http://10.0.154.125/aspera/orchestrator/saml/metadata?=1" ID="_ab676d30-
b03b-0135-65e4-0050569fd8f4">
<md:SPSSODescriptor AuthnRequestsSigned="false" WantAssertionsSigned="false"
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</md:NameIDFormat>
    <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
POST" Location="http://10.0.154.125/aspera/orchestrator/saml_response/1" index="0"
isDefault="true"/>
</md:SPSSODescriptor>
</md:EntityDescriptor>
```

3. Right-click the page and click **Save as**; save it as `*filename_metadata*.xml`.

SAML Assertion Requirements

Orchestrator: expects assertion from an IdP to contain the following elements:

Default Attribute	**Orchestrator User Field**	**Required**
`NameID` / `SAML_SUBJECT`	Username	Yes, with the format: `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`
`email`	Email address	Yes
`given_name`	First name	Yes
`surname`	Last name	Yes

Configuring SAML in Orchestrator

SAML configuration is only available for Orchestrator 3.1.0 and above.

Orchestrator supports one active SAML configuration at a time. If more than one SAML configuration is enabled, the first enabled configuration in the list becomes the active one.

Creating an New SAML Configuration

1. Log in to Orchestrator.
2. On the top-right of the page, click the **Admin** dropdown and click **Preferences**.
3. In the left-side menu, click **SAML Configuration**.
4. On the SAML Configurations page, click **New SAML Configuration**.
5. Enter your information in the fields according to the table below.

New SAML Configuration Template

Field	
Name	Enter a name for new configuration. This name is used by Orchestrator to differentiate between multiple SAML configurations.
Name ID format	Defaults to the following:

Field	
	`urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified` Customize as needed. The format must match format used with your IdP.
SSO target URL mapping	This is your IdP Single Sign-On URL.
Certificate	Obtained from IdP server admin.
Fingerprint	Obtained from IdP server admin. You can use this instead of the certificate to authenticate with SAML IdP.
Enable configuration	Select the checkbox to enable configuration and display the SAML option on the local login page.
Publicly visible	If selected, makes the SAML configuration publicly visible (performs the same function as "Enable configuration".
Login text	Enter message (welcome, instructions) that user sees at login
Fingerprint algorithm	Defaults to the following: `http://www.w3.org/2000/09/xmldsig#sha1` Customize as needed
Attribute for email	This value must map to the corresponding attribute in your SAML IdP's SAML response; see "Configuring Your Identity Provider (IdP)" on page 63 for details.
Attribute for first name	This value must map to the corresponding attribute in your SAML IdP's SAML response; see "Configuring Your Identity Provider (IdP)" on page 63 for details.
Attribute for last name	This value must map to the corresponding attributes in your SAML IdP's SAML response; see "Configuring Your Identity Provider (IdP)" on page 63 for details.
Allowable clock drift	Optional. Enter milliseconds allowed for clock drift between Orchestrator and SAML IdP; default is `0`
Roles to be assigned	This assigns a default Orchestrator role to the SAML user. Click the plus sign (+) next to **Group selection** to expand the list. Select the checkbox next to the desired group. If, for example, you select Administrator, the `orchestrator` user created when logging in with SAML is assigned to the Orchestrator Administrator group.

6. When finished, click **Save**.

   The configuration now appears in the list on the SAML Configurations page.

7. Finish the SAML configuration by adding the Orchestrator server metadata to the relying party file on the IdP server.

   The following steps allow you to obtain the required Orchestrator metadata.

   a) Click the **Metadata** button.

      An XML file opens. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
entityID="http://10.0.154.125/aspera/orchestrator/saml/metadata?=1" ID="_ab676d30-
b03b-0135-65e4-0050569fd8f4">
<md:SPSSODescriptor AuthnRequestsSigned="false" WantAssertionsSigned="false"
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
```

```
    <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</
md:NameIDFormat>
    <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
POST" Location="http://10.0.154.125/aspera/orchestrator/saml_response/1" index="0"
isDefault="true"/>
</md:SPSSODescriptor>
</md:EntityDescriptor>
```

b) Right-click the page and click **Save as**; save it as `*filename_metadata*.xml`.

c) Copy the XML file to the IdP server.

Modifying an Existing SAML Configuration

To modify an existing SAML Configuration, click the **More** button (•••) to the right of the desired configuration in the list. The context menu opens.

- To edit the configuration, click **Edit**.\
- To delete the configuration, click **Destroy**.

The Orchestrator Help Page

The Help page provides access to a number of key information resources for Orchestrator like template workflows, descriptions of the plugins, and a test environment for API endpoints.

At the top-right corner of the page, click **Admin > Help Center**.

Plugins and Workflows

Click **Plugins** or **Workflows** to see a complete list with descriptions. You can filter the display be entering a search string in the Search box. This returns a list all plugins or workflows that have a name matching the search string or have tags that match the provided search string.

- **Workflows** displays a list of preconfigured Workflows for various purposes. Each workflow displays search term buttons that link to the associated plugins. To the right of each workflow is a view icon and a download icon. The View icon shows the image of the workflow as it appears in the workflow designer. Clicking the Download icon downloads the workflow to your system so that you can import it (see "Importing a Workflow" on page 107).

  Note: These workflows do not run "as is", because there will be dependencies (for example, a remote node), but they will run after proper configuration.

- **Plugins** displays complete a list of plugin in the system. Click the "Example" icon at the right to view all preconfigured workflows that use that particular plugin.

Admin Guide

Click **Admin Guide** in the far-left navigation pane for guidance in installing, configuring, and upgrading Orchestrator. You can also view the latest Admin Guide on the Aspera website.

API Test Environment

Click **API** in the far-left navigation pane to open a list of commonly used API calls in Orchestrator. In the left pane, enter inputs and outputs, select JSON or XML, and choose various function calls such as **Initiate** or **Check status** to test the response on the local system. The URL and response body displays in the right pane.

Starting, Stopping, and Restarting Orchestrator

This section contains procedures for starting, stopping, and restarting Orchestrator.

Starting Orchestrator

1. Log in to the Linux machine that hosts Orchestrator as `root`.
2. Start Orchestrator using one of the following commands:

```
# asctl orchestrator:start
# /etc/init.d/AsperaOrchestrator start
# service AsperaOrchestrator start
```

Stopping Orchestrator

1. Log in to the Linux machine that hosts Orchestrator as `root`.
2. Stop Orchestrator using one of the following commands:

```
# asctl orchestrator:stop
# /etc/init.d/AsperaOrchestrator stop
# service AsperaOrchestrator stop
```

Restarting Orchestrator

1. Log in to the Linux machine that hosts Orchestrator as a user with privileges to restart Orchestrator.
2. Restart Orchestrator using one of the following commands:

```
# asctl orchestrator:restart
# /etc/init.d/AsperaOrchestrator restart
# service AsperaOrchestrator restart
```

Managing Users

This section contains procedures for managing user accounts.

Creating Users

To create a user, click **Users** in the top menu.

1. Click **+ New User** at the top left of the page.
2. Under **New user information**, enter the details for the new user.

   Email, **Phone** and **Sms** are optional fields.

   After all information is entered, click **Save user information**.

   Note: See "Creating Strong Passwords" on page 68 for detailed information about the characteristics of strong passwords and how to enforce them.
3. To edit a user account, click **Accounts > Users**; in the Listing Users pane click the dropdown arrow next to the user name and select **Edit Profile/Permissions**.

Modifying User Information and Assigning a User to a Group

To modify user information or assign a user to a group, click **Users** in the top menu. To the right of the username for the user, click the "More" icon (**...**), then click **Edit Profile/Permissions**.

Updating User Information

In the **Modify user information** panel, modify the user details as needed, then click **Save user information**.

Note: Email, **Phone** and **Sms** are optional fields.

See "Creating Strong Passwords" on page 68 for detailed information about the characteristics of strong passwords and how to enforce them.

Assigning a User to a Group

The table below lists each group and its associated permission. All groups can view the **Dashboard**, although levels of access vary.

Group	Permission
Developer	**Workflows**
System	**Engine**
Contributor	**Tasks** (view only)
Scheduler	**Queues**
Operator	**Work Orders** and **Monitor**
Admin	Full system access.

In the **Role/Group Memberships** area, click the dropdown arrow for **Select Group**, click the desired group name, and click **Add to group**.

Deleting Users

Admin users and non-admin users can delete any user *except* for the users listed in the "Can't delete" column in the table below.

User Type	Can't delete
Admin users	• Themselves • The `admin` user account • The `system` user account
Non-admin users	• Themselves • The `admin` user account • The `system` user account • Users with admin privileges

Creating Strong Passwords

Orchestrator has a strong password option, which is enabled by default. When enabled, the user must enter a password which satisfies the following criteria:

- Minimum length of eight characters.

- At least one upper-case character.
- At least one lowercase character.
- One or more special characters, such as @#$%^&+=
- New user passwords cannot be the same as any of the previous three passwords.

The above criteria are configured by default, but a number of parameters can be edited in the Orchestrator configuration:

- To change the requirements for a strong password, update the regular expression in `strong_password_regex`.
- To disable the strong password option, change the value of `impose_strong_password` to `false`.
- To deactivate a user account after a predefined number of unsuccessful attempts to log in, change the value of `allowed_logon_attempts`. By default, the value of this parameter is set to 3.

  Note: After Orchestrator deactivates a user account, an admin must manually reactivate the user in the UI.

For more information about the Orchestrator configuration, see "Editing the Orchestrator Configuration" on page 71.

Resetting Passwords

To reset the password for a user, click **Users** in the top menu.

1. To the right of the username for the user, click the "More" icon (**...**), then click **Change Password**.
2. On the **Change Password for (user)** page:
   - In the box for **Current password for admin**, enter your own password. This is a measure for enhanced security in the application.
   - In the remaining two boxes, enter and confirm the new password for the user. See "Creating Strong Passwords" on page 68 for more information.
3. Click **Change**.
4. Log out of all open sessions for the user whose password was changed.

   Note: This is another important measure to protect the security of Orchestrator users.

Managing Permissions

The Manage Permissions page displays lists of permissions for various roles in Orchestrator.

This procedure requires admin privileges.

1. In the top menu, click **Users**.
2. In the left navigation, click **Permissions**.
3. On the **Manage Permissions** page, click **By Type**, **By User**, or **By Group**.
4. To view the permissions available to a particular permission type (or user, or group), click an item in the list.

   A **Permissions** pane appears. This area lists all the permissions available to that selection.
5. To add or remove a permission, click a listed permission to view its details and then click **Grant Permission** or **Revoke** for a user or a group.
6. To make an additional permission available to a type, user, or group, click **New Permission**.
   - From the **Principal** dropdown, select a permission.
   - From the **Group/role or user** dropdown, select a user or group.

   Click **Create**.

Defining a New Permission

To define a new permission for a user or a group, click **Users** in the top menu.

1. In the left panel, click **Permissions**.

   The **Manage Permissions** page appears.
2. To define a new permission, click **+ New Permission**.
3. In the **New Permission** panel, select the principal for the new permission from the dropdown menu or click **Enter new principal** to enter a new principal.

   A *principal* is a unique string to identify permissions. For example:

   - The principal `Aspera:Orchestrator` sets a permission for the entire application.
   - The principal `Aspera:Orchestrator:Workflow` restricts the permission to all **Workflows** in the application.
   - The principal `Aspera:Orchestrator:Workflow:view` allows viewing permission, only, for all **Workflows**.
4. Select the user or group to which the new permission will apply.

Configuring LDAP Access

Orchestrator supports configuration for multiple LDAP servers.

1. In the top menu, click **Users**.
2. In the left navigation, click **Active Directory Setup**.
3. On the **Active Directories** page, click **+ New Active Directory**.
4. On the **Active Directories** page, enter the following information:

   - Enter the **Name** and **Description** for the new Active Directory.
   - To enable the LDAP service, select **Enable Directory Service**.
   - For **Server**, enter the LDAP server fully qualified name or IP address.
   - For **Port**, enter the LDAP service port.
   - For **Treebase**, enter the base distinguished name.
   - For **Filter** field, enter `uid` for LDAP.
   - For **Login Method**, select `Anonymous` (no login/password required) or `Login` (login/password required).
5. Click **Save** to validate the settings against the LDAP server (LDAP bind done with Server, Port and Login credentials).

   If the process is successful, the following message appears:

   ```
   Active directory was successfully created. Successfully connected to Active Directory.
   ```

   If the new active directory fails, an error message appears, citing the specific reasons for failure and giving instructions for retrying the process.

   ```
   Connection to Active Directory failed: Reason for error. Active Directory has been disabled.
   Adjust your settings, click on Enable Directory Service, and save again to retest.
   ```
6. Log in to Orchestrator with an LDAP username and password (without first creating the user in Orchestrator GUI).

   A user with type `Active Directory User` is created automatically by Orchestrator the first time this login is successful.
7. In the top menu, click **Users**.

   In the row for the relevant user, examine the **Type** column to confirm that `Active Directory User` is the displayed value.

After the user logs in with a username and password, Orchestrator performs a combination of LDAP binding and search. First, it connects and binds to the LDAP server via the login credentials as configured in the Active Directory Service Details screen. Then it searches the LDAP server for an entry corresponding to the filter specified in step "4" on page 70(for example, `uid=loginname`). If the entry exists, it rebinds as that user with the password entered by the user.

System Administration

Editing the Orchestrator Configuration

To edit the configuration of Orchestrator, admin users can make updates in the Orchestrator UI or the Orchestrator configuration file.

Editing the Configuration in the UI

To edit the configuration in the UI, click **Engine > Configuration > Configure Orchestrator Parameters** and edit the fields on the page.

Editing the Configuration File

To edit the configuration file, open the file, `orchestrator.yml`:

```
/opt/aspera/var/config/orchestrator/orchestrator.yml
```

If `orchestrator.yml` is missing in the directory, copy it from the following location into the directory above (which is preferred because it cannot be overwritten during upgrade):

```
/opt/aspera/orchestrator/config/orchestrator.yml
```

Important: If you have a large volume of system activity and limitations on the capacity of your database, the default value for some parameters is probably too high for your system and could lead to system failure. Additionally, you might experience slow performance immediately after an upgrade because—potentially—Orchestrator could start purging a high volume of data.

For the following parameters, you should set a lower number (the smallest number compatible with business requirements):

- `cleanup_cutoff`
- `journal_cleanup_cutoff`
- `journal_purge_cutoff`
- `purge_cutoff`

Aspera recommends that you monitor your database and adjust these values as needed for optimal functioning of the system.

The following table describes the available parameters:

Parameter name	Default value	Usage
`active_active`	`false`	Set value to `true` when Orchestrator is clustered as active-active.
`ad_user_groups`	`Contributor`	When an Active Directory user logs in, this group is assigned by default.
`allow_blank_password`	`false`	Users who do not have a password will be forced to create a password.

Parameter name	Default value	Usage
`allowed_logon_attempts`	3	Defines the number of allowed login attempts with an incorrect password before the user account is deactivated.
`archive_dir`	Operating system-specific (refer to "Orchestrator Directory Locations" on page 224).	This directory contains an archive of plugins, capsules and snapshots. The directory contents are generated by actions initiated from the Orchestrator UI.
`aspera_dir`	`/opt/aspera`	The parent folder for all Aspera products.
`asynch_threads`	2	Determines the number of asynchronous workers that are generated when Orchestrator starts.
`authenticate_via_saml`	false	If true, the user is presented with a SAML login prompt.
`auto_activate_portlets`	`false`	When a portlet is assigned to another user or group, and this parameter is set to `true`, the portlet is activated. The default behavior is that users must activate portlets manually.
`build`	not applicable	Represents the Orchestrator product build number. **Important:** This value should not be edited.
`cleanup_cutoff`	`60`	Determines the number of days the data for `failed`, `canceled` and `error` work orders is kept in the database. **Important:** If you have a large volume of system activity and limitations on the capacity of your database, the default value is probably too high for your system and could lead to system failure. Additionally, you might experience slow performance immediately after an upgrade because —potentially—Orchestrator could start purging a high volume of data. *Aspera recommends that you set a lower number* (the smallest number compatible with business requirements). You should continue to monitor your database and adjust this value as needed for optimal functioning of the system.
`config_dir`	Operating system-specific (refer to	This directory contains an actions folder which has run time instances of:

Parameter name	Default value	Usage
	"Orchestrator Directory Locations" on page 224).	• All available plugins • Product configuration file, `orchestrator.yml` • Database configuration file, `database.yml` • Licenses folder • Capsules folder • Folder with configuration data for workflow actions, `actions_config` • File with a list of plugins, `default_plugins_list.yml` • File with a list of portlets, `default_portlet_list.yml`
`custom_tables`	not applicable	Any custom database tables that must be backed up should be entered here as comma-separated values.
`date_format`	`american`	Format for dates displayed in Orchestrator
`db_bin`	`/opt/aspera/common/mysql/bin`	This parameter gives Orchestrator the path to the `bin` directory of MySQL for commands like `mysqldump`.
`default_step_timeout`	`60`	The number of seconds after which a step should fail after a period of inactivity.
`delete_journal_with_workorder`	`false`	When this option is set to `true`, and the user selects a workflow for deletion, both the selected work order and the journal associated with it are deleted.
`disable_autocomplete_login_page`	`false`	Disables autocomplete of username and password on the login page. By default, this behavior is allowed.
`engine_heartbeat`	`1`	The frequency with which the Orchestrator engine is polled for status.
`engine_instance`	`0`	In an active-active or active-passive cluster, this parameter takes a unique value in each of the nodes. **Important:** For a setup with single node installation, preserve the default value.
`external_facing_uri`	not applicable	Hostname or IP address
`external_javascript_lib`	not applicable	Empty by default. Any new Javascript libraries used must be defined in this parameter.
`fast_start`	`false`	[Windows] Changing the value to `true` on Windows prevents Orchestrator from

Parameter name	Default value	Usage
		starting all processes at once, which saves computer resources.
`hide_preferences_for_groups`	not applicable	If a group name is provided here, the **Preferences** option—available from the **admin** menu at the top right of each page—is hidden.
`high_performance_polling`	`true`	Offers higher performance polling for new steps in a work order. If true, the higher performant polling is used. If false, the lesser performant polling is used.
`ifv_skip_start_event`	`false`	If set to `true`, inline file validation skips `start` events.
`impose_strong_password`	`false`	If set to `true`, strong password policies are imposed, such as the following: • Check the strength of a password during user creation and user change-password operations. • Keep track of unsuccessful login attempts. • Don't allow the user to set any of the previous three passwords as the new password.
`install_config_file`	`opt/aspera/var/config/orchestrator/`	The path where the configuration file,`orchestrator.yml`, is stored. This path should point to a folder that will not be overwritten during upgrades.
`install_css_dir`	`opt/aspera/var/config/orchestrator/stylesheets/`	A safe location for storing custom CSS files so they are not overwritten during upgrades
`install_images_dir`	`opt/aspera/var/config/orchestrator/stylesheets/images/`	A safe location for storing custom images so they are not overwritten during upgrades
`install_pages_dir`	`opt/aspera/var/config/orchestrator/stylesheets/pages/`	A safe location for storing custom web pages so they are not overwritten during upgrades
`journal_cleanup_cutoff`	`10`	Defines the number of days for which journals can accumulate before deletion. **Important:** If you have a large volume of system activity and limitations on

Parameter name	Default value	Usage
		the capacity of your database, the default value is probably too high for your system and could lead to system failure. Additionally, you might experience slow performance immediately after an upgrade because —potentially—Orchestrator could start purging a high volume of data. *Aspera recommends that you set a lower number* (the smallest number compatible with business requirements). You should continue to monitor your database and adjust this value as needed for optimal functioning of the system.
`journal_delete_empty_packages`	`true`	If you need the system to retain empty packages, set this option to `false`.
`journal_kept_duration`	`10`	The number of days entries are kept in the Journal view
`journal_on`	`true`	Changing to the value to `false` deactivates the Journal feature on an Orchestrator server.
`journal_purge_cutoff`	`30`	Defines the number of days for which end-state (failed or errored) journals can accumulate before deletion; does not delete in-progress journals. **Important:** If you have a large volume of system activity and limitations on the capacity of your database, the default value is probably too high for your system and could lead to system failure. Additionally, you might experience slow performance immediately after an upgrade because —potentially—Orchestrator could start purging a high volume of data. *Aspera recommends that you set a lower number* (the smallest number compatible with business requirements). You should continue to monitor your database and adjust this value as needed for optimal functioning of the system.
`load_basic_workflow_dir`	Orchestrator Dir + /vendor/ workflows	The directory that has template workflows available for loading when Orchestrator starts.
`load_default_workflows`		Changing to the value to `false` prevents the Orchestrator server from loading the default workflows.
`log_file`	`true`	The path to the Orchestrator log file

Parameter name	Default value	Usage
`log_level`	`debug`	The verbosity level of the Orchestrator log file. The value debug is the highest level available. Other log levels, in descending order of verbosity, are `info`, `warn`, `error`,`fatal`, and unknown.
`log_retention_days`	`7`	Sets the retention policy for log files. If you configure a value for `max_log_size_mb`, the value of this parameter is the maximum number of log files present in the system at any given time. Otherwise, the the value of this parameter is the maximum number of days that Orchestrator retains a log file.
`logon_timeout`	`0`	This value denotes the number of seconds after which a user is logged off, after a period of inactivity. Setting it to `0` allows users to remain logged in until they manually log off.
`maintenance_interval`	`600`	Number of engine "heartbeats" after which an Orchestrator maintenance task is generated. In other words, after creating work orders with a defined frequency of creation, the engine then initiates a maintenance task, such as deleting old work orders.
`maintenance_heartbeat_interval`	`300`	The frequency with which Orchestrator maintenance tasks are generated
`make_view_by_workflow_default`	`false`	If set to `true`, the default view on the **Work Orders** page changes from **View by Work Order** to **View by Workflow**.
`max_tasks_displayed`	`12`	The number of tasks that should be displayed in the **Tasks** list (under the bell icon at the top right of the screen)
`minimal_xml_escaping`	`false`	Specifies how characters in XML responses for the Orchestrator endpoints are escaped. If `false`, all characters that are conventionally escaped in XML—according to best practices—are escaped. If `true`, only "<'"and "&" are escaped.
`mongrel_action_timeout`	`300`	The number of seconds after which a mongrel process is determined to be stuck on an action
`mongrel_max_daemons`	`3`	The number of mongrel instances that should be spawned when Orchestrator is started
`mongrel_starting_port`	`3000`	The port where the first mongrel instance is generated. Other instances will be

Parameter name	Default value	Usage
		assigned a port that is incremented by a value of 1 per generated instance.
orchestrator_dir	/opt/aspera/orchestrator	The directory where Orchestrator is installed
perform_ad_operation	false	If set to true, Orchestrator performs non-reversible Active Directory operations such as adding, modifying or deleting Active Directory users.
persistence_mode	disk	The location where persistence files are stored. Allowable values are disk and db
personalized	true	If set to false, the CSS is reverted to a UI with a blue header (older Orchestrator UI)
publish_default_workflows	true	Changing this value to false prevents the Orchestrator server from publishing the default workflows.
purge_cutoff	30	Determines the number of days that the data for a successful work order is kept in the database. **Important:** If you have a large volume of system activity and limitations on the capacity of your database, the default value is probably too high for your system and could lead to system failure. Additionally, you might experience slow performance immediately after an upgrade because—potentially—Orchestrator could start purging a high volume of data. *Aspera recommends that you set a lower number* (the smallest number compatible with business requirements). You should continue to monitor your database and adjust this value as needed for optimal functioning of the system.
rails_log_level	error	The verbosity level of the Rails log file. This value can be increased to warn, info, and debug (listed in increasing order, with debug being the highest), or it can be decreased to fatal.
recent_workflow_count	10	Returns the specified number of workflows
referer_hosts	not applicable	Used to block a request from an untrusted source or to enter the hostname or IP address of the Orchestrator server

Parameter name	Default value	Usage
`run_dir`	Operating system-specific (refer to "Orchestrator Directory Locations" on page 224).	This directory contains run time information and is populated with files and folders from plugins, work orders, and portlets. It also contains the log file.
`strict_cache_control`	`false`	If set to true, cache control header are placed on all web pages.
`strict_prepost_processing`	`false`	When set to `true`, a failure in the pre-processing or post-processing of a step will cause the step to fail.
`strong_password_regex`	^(?=.{8,})(?=.*[a-z])(?=.*[0-9])(?=.*[A-Z])(?=.*[@#$%^&+=]).*$	The regular expression against which the password is checked when a new user is created and when a current user password is changed. This will take effect only when the `impose_strong_password` parameter is set to `true`. The default regular expression requires that the password has the following characteristics: • A minimum length of eight characters. • At least one uppercase character. • At least one lowercase character. • At least one of these special characters: @#$%^&+=
`synch_threads`	4	Determines the number of synchronous workers that are generated when Orchestrator starts
`task_refresh_frequency`	30	The frequency with which Orchestrator polls for a new task
`use_db_time`	`false`	By default, the system time (local time) is used for the date format. If set to true, displays the database time.
`use_new_designer`	`true`	If set to `false`, the Orchestrator designer screen will revert to a plain background, instead of the default grid display.
`web_root`	`/aspera/orchestrator/`	When this parameter is set to a valid URL that is different from the default, Orchestrator makes a request to Apache to render the Orchestrator UI at that URL.

Synchronous and Asynchronous Workers

A synchronous worker waits until the end of the execution of a step to gather its status and outputs. If the timeout is left as the default setting and no unique message is rendered periodically there is a chance that the synchronous step fails after 60 seconds due to timeout error.

An asynchronous worker gathers the status and outputs of a step periodically. The frequency with which it does this is determined by the polling frequency of the step (usually every 5 seconds); for this reason, if an asynchronous worker is used, there is a smaller likelihood that the step will time out.

Adding Mongrel Processes for a New Apache Port

The number of Orchestrator mongrels can be increased, and they can also have the port listen on a different Apache port. This procedure is applicable to situations where a high volume of API calls to Orchestrator slows down the user interface.

1. Edit `orchestrator.yml` (the Orchestrator configuration file) to increase the number of mongrels.
   a) Open `orchestrator.yml` from the following location:

      ```
      opt/aspera/var/config/orchestrator/
      ```

   b) Locate the option `max_mongrels` and increase it to the desired integer.
   c) Restart Orchestrator, using the procedure in "Restarting Orchestrator" on page 67.
2. Create a virtual host for Orchestrator inside the configuration file for Apache.
   a) Create a file in the following directory:

      ```
      /opt/aspera/common/apache/custom
      ```

      Aspera recommends naming this file `custom_config.conf`.
   b) Add the following configuration content to the file:

      ```
      Listen port_number
      VirtualHost *:port_number
          ServerAdmin webmaster@localhost

      Alias /aspera/orchestrator/images "/opt/aspera/orchestrator/public/images"
      Alias /aspera/orchestrator/stylesheets "/opt/aspera/orchestrator/public/stylesheets"
      Alias /aspera/orchestrator/javascripts "/opt/aspera/orchestrator/public/javascripts"
      <Directory "/opt/aspera/orchestrator/public">
       Options -Indexes -FollowSymLinks
       AllowOverride none
       Order allow,deny
       Allow from all
      </Directory>

      <Proxy balancer://orchestrator_cluster>
       Allow from all
            BalancerMember http://127.0.0.1:3000
            BalancerMember http://127.0.0.1:3001
            BalancerMember http://127.0.0.1:3002
            BalancerMember http://127.0.0.1:3003
            BalancerMember http://127.0.0.1:3004
      </Proxy>

      ProxyPass /aspera/orchestrator/images !
      ProxyPass /aspera/orchestrator/stylesheets !
      ProxyPass /aspera/orchestrator/javascripts !

      # send the proxy request

         # @web_root starts with / if not nil
         ProxyPass /aspera/orchestrator balancer://orchestrator_cluster/aspera/orchestrator

         RequestHeader set X_FORWARDED_PROTO "https"

      </VirtualHost>
      ```

      Note:

The number of mongrels currently running in Orchestrator must match the number of BalancerMembers (load balancers); there are five BalancerMembers in the example above.

c) Open the configuration file for Apache to confirm that the custom directory is loaded. The configuration file is found in the following location:

```
/opt/aspera/common/apache/ashttpd.conf
```

Note: Repeat this step as needed to add more ports to the configuration.

3. Restart Apache with the following command:

```
# asctl apache:restart
```

4. Test the results of the prodedure.

a) Open Orchestrator from the following URL:

```
http://localhost:port_name/aspera/orchestrator/
```

Note: Replace `localhost` with the Orchestrator IP address if accessing Orchestrator from a remote node.

b) Log in with a valid user ID and password.

You are redirected to the Work Orders page.

Create Snapshot

This `asctl` method creates a snapshot of all Orchestrator configuration and design-time data.

Method name	snapshot
Command	`asctl orchestrator:create_snapshot`
Details	Creates snapshot of all Orchestrator configuration and design-time data.
Inputs	None
Outputs	STDOUT to indicate the progress of the snapshot and eventually a success or failure message

```
# asctl orchestrator:create_snapshot
Orchestrator: Create snapshot... Snapshot of Orchestrator configuration was successfully
created at 2016-06-15T14:55:25-07:00.
Snapshot file located at /opt/aspera/var/archive/orchestrator/snapshots/
Backup_at_20160615_145506.snap
done
```

Working with Capsules on a Single Node

Capsules are Orchestrator patches that are applied to your current installation of Orchestrator to improve functionality, without the need for an upgrade. If your Orchestrator installation is in a high availability (clustered) environment, see "Working with Capsules in a High Availability Environment" on page 42

Working with Capsules on a Single Node

Click **Engine > Capsule Manager** to open the **Capsule Manager** page, which lists all the capsules on the Orchestrator server.

Note: You will not find capsules that have a file name starting with "_" on the **Capsule Manager** page. These are special "one-time-only" capsules designed to run only once after ingest. You can find them in the list of archived capsules after import. See the section below, "Working with One-time-only Capsules", for more information.

Once they are imported, capsules can be viewed, archived to the archive directory, or deleted from the server:

View
Opens up a modal window with the capsule code in read-only mode.

Archive
Prompts the user for confirmation to proceed with the archiving of a capsule.

Note: Archiving a capsule removes it from the capsule directory and moves it to the archive directory. Orchestrator must be restarted to allow the change to take effect.

Delete
Remove the file from the Orchestrator server.

Note: This action is irreversible and Orchestrator must be restarted to allow the change to take effect.

Working with One-time-only Capsules

A capsule whose filename begins with "_" is a "one-time-only" capsule, meaning it gets executed only once, during the ingest phase. Unlike other capsules applied during upgrade that are included in a post-upgrade snapshot, "one-time-only" capsules do not get included in the snapshot. After they are applied, you can find them in `/opt/aspera/var/archive/orchestrator/capsules`, but you will not see on the **Capsule Manager** page (**Engine > Capsules**) or in the active capsules directory, `/opt/aspera/var/config/orchestrator/capsules`.

In general you should re-apply one-time-only capsules after a snapshot import, and there is no risk in doing so. However, if the one-time-only capsule is meant to modify a database table that is part of a snapshot, then there is no need to re-apply it.

Downloading a Capsule Before Importing It

Important: Only download (and later import) a capsule that corresponds to the version of Orchestrator currently installed on your system.

To download an updated version of a capsule, go to the link provided by IBM Aspera on Cloud, https://aspera.pub/5nxA_C4.

Note: In order to download capsules, users must install IBM Aspera Connect. See https://www.ibm.com/aspera/downloads/ for more information.

Importing a Capsule in the UI

1. On one node, click **Engine**, and in the left navigation, click **Capsules**.
2. Click **Import**.
3. In the **Upload Capsule File for Import** dialog, click **Browse**.
4. Select a capsule file (a `.rb` file) and click **Upload**.

Importing a Capsule Manually

1. Copy the capsule into `/opt/aspera/var/config/orchestrator/capsules`.
2. Restart Orchestrator:

```
# asctl orchestrator:restart
```

Customizing the Orchestrator Logo

You can replace the default logo for IBM Aspera Orchestrator with a custom logo.

A custom logo is useful for creating a visual distinction between different Orchestrator instances.. For example, you can use different logos for the development server and the production server, or for the two nodes in a high availability environment.

1. Open the directory below and make a backup of `aspera_orchestrator_logo.png`.

   ```
   /opt/aspera/orchestrator/public/images/aspera_orchestrator_logo.png
   ```

   This step preserves the original logo file in case you need to revert your changes.
2. In the same directory as above, overwrite the default file with your custom logo file.
3. To edit the branding configuration file, `branding.yml`, open it in this location:

   ```
   /opt/aspera/var/config/orchestrator/branding.yml
   ```

   Set the value of the `use_custom_logo` field to `true` (default is `false`):

   ```
   use_custom_logo: true
   ```
4. Restart Orchestrator to propagate your changes:

   ```
   # asctl orchestrator:restart
   ```

Migrating from Disk Persistence to Database Persistence

Trigger steps created in Aspera Orchestrator store information in a persistent location to prevent duplicating triggers. Use the procedure below to migrate from a disk (file) to database persistence.

1. Stop active work orders (by cancelling them, pausing them or destroying them), and stop any monitors that might restart those work orders.
2. Back up the configuration with the following command:

   ```
   $ cp /opt/aspera/var/config/orchestrator/orchestrator.yml /opt/aspera/var/config/
   orchestrator/orchestrator.yml_date
   ```
3. Update the configuration by adding the following line to the `.yml` file:

   ```
   persistence_mode: db
   ```
4. Stop Orchestrator according to the procedure in “Stopping Orchestrator” on page 67.
5. Migrate the persistence information from the `.prst` files to the database.

   ```
   $ cd <Orchestrator_Install_Directory>
   $ ruby script/cmd
   Dir.glob('<Orchestrator Run Directory>/*/*.prst').each { |prst|
       s=SharedState.retrieve_from_file(prst)
       SharedState.persist_to_db(s,prst)
       s_db=SharedState.retrieve_from_db(prst)
       if(s_db != s)
           throw "Error migrating SharedState file '#{prst}'"
       end
   }
   ```
6. Start Orchestrator according to the procedure in “Starting Orchestrator” on page 67.
7. Watch the Orchestrator log file (`orchestrator.log`) for errors and confirm that the work orders start successfully in the UI.

   See “Orchestrator Directory Locations” on page 224 for the location of your log file.

Rollback from Database Persistence to Disk Persistence

This article describes a rollback of the procedure in “Migrating from Disk Persistence to Database Persistence” on page 82.

1. Back up the configuration.

```
$ cp opt/aspera/var/config/orchestrator/orchestrator.yml_date opt/aspera/var/config/
orchestrator/orchestrator.yml
```

2. Update the configuration by removing the following line from the `.yml` file:

   ```
   persistence_mode: disk
   ```

3. Stop Orchestrator according to the procedure in "Stopping Orchestrator" on page 67.
4. Start Orchestrator according to the procedure in "Starting Orchestrator" on page 67.
5. Watch the Orchestrator log file (`orchestrator.log`) for errors and confirm that the work orders start successfully in the UI.

   See "Orchestrator Directory Locations" on page 224 for the location of your log file.

Adding Mongrel Processes for a New Apache Port

The number of Orchestrator mongrels can be increased, and they can also have the port listen on a different Apache port. This procedure is applicable to situations where a high volume of API calls to Orchestrator slows down the user interface.

1. Edit `orchestrator.yml` (the Orchestrator configuration file) to increase the number of mongrels.
   a) Open `orchestrator.yml` from the following location:

      ```
      opt/aspera/var/config/orchestrator/
      ```

   b) Locate the option `max_mongrels` and increase it to the desired integer.
   c) Restart Orchestrator, using the procedure in "Restarting Orchestrator" on page 67.
2. Create a virtual host for Orchestrator inside the configuration file for Apache.
   a) Create a file in the following directory:

      ```
      /opt/aspera/common/apache/custom
      ```

      Aspera recommends naming this file `custom_config.conf`.
   b) Add the following configuration content to the file:

      ```
      Listen port_number
      VirtualHost *:port_number
          ServerAdmin webmaster@localhost

      Alias /aspera/orchestrator/images "/opt/aspera/orchestrator/public/images"
      Alias /aspera/orchestrator/stylesheets "/opt/aspera/orchestrator/public/stylesheets"
      Alias /aspera/orchestrator/javascripts "/opt/aspera/orchestrator/public/javascripts"
      <Directory "/opt/aspera/orchestrator/public">
       Options -Indexes -FollowSymLinks
       AllowOverride none
       Order allow,deny
       Allow from all
      </Directory>

      <Proxy balancer://orchestrator_cluster>
       Allow from all
            BalancerMember http://127.0.0.1:3000
            BalancerMember http://127.0.0.1:3001
            BalancerMember http://127.0.0.1:3002
            BalancerMember http://127.0.0.1:3003
            BalancerMember http://127.0.0.1:3004
      </Proxy>

      ProxyPass /aspera/orchestrator/images !
      ProxyPass /aspera/orchestrator/stylesheets !
      ProxyPass /aspera/orchestrator/javascripts !

      # send the proxy request

         # @web_root starts with / if not nil
         ProxyPass /aspera/orchestrator balancer://orchestrator_cluster/aspera/orchestrator

         RequestHeader set X_FORWARDED_PROTO "https"
      ```

```
</VirtualHost>
```

Note:

The number of mongrels currently running in Orchestrator must match the number of BalancerMembers (load balancers); there are five BalancerMembers in the example above.

c) Open the configuration file for Apache to confirm that the custom directory is loaded. The configuration file is found in the following location:

```
/opt/aspera/common/apache/ashttpd.conf
```

Note: Repeat this step as needed to add more ports to the configuration.

3. Restart Apache with the following command:

```
# asctl apache:restart
```

4. Test the results of the prodedure.

a) Open Orchestrator from the following URL:

```
http://localhost:port_name/aspera/orchestrator/
```

Note: Replace `localhost` with the Orchestrator IP address if accessing Orchestrator from a remote node.

b) Log in with a valid user ID and password.

You are redirected to the Work Orders page.

Increasing Apache Proxy Timeout

With use cases that cause mongrels to execute longer queries, users may experience proxy errors because Apache is not receiving a response from the mongrels. The following configuration increases the possible timeout values.

```
# tell Apache where to find static files /aspera/orchestrator
Alias /aspera/orchestrator/images "/opt/aspera/orchestrator/public/images"
Alias /aspera/orchestrator/stylesheets "/opt/aspera/orchestrator/public/stylesheets"
Alias /aspera/orchestrator/javascripts "/opt/aspera/orchestrator/public/javascripts"
Alias /aspera/orchestrator/my/images "/opt/aspera/var/config/orchestrator/web/images"
Alias /aspera/orchestrator/my/stylesheets "/opt/aspera/var/config/orchestrator/web/stylesheets"
Alias /aspera/orchestrator/my/javascripts "/opt/aspera/var/config/orchestrator/web/javascripts"
<Directory "/opt/aspera/orchestrator/public">
  Options -Indexes +FollowSymLinks
  AllowOverride none
  Order allow,deny
  Allow from all
</Directory>
<Directory "/opt/aspera/var/config/orchestrator/web">
  Options -Indexes -FollowSymLinks
  AllowOverride none
  Order allow,deny
  Allow from all
</Directory>

ProxyTimeout 1200

<Proxy balancer://orchestrator_cluster>
       BalancerMember http://127.0.0.1:3000
       BalancerMember http://127.0.0.1:3001
       BalancerMember http://127.0.0.1:3002
       BalancerMember http://127.0.0.1:3003
       BalancerMember http://127.0.0.1:3004
</Proxy>

# don't proxy static files, serve directoy
ProxyPass /aspera/orchestrator/images !
ProxyPass /aspera/orchestrator/stylesheets !
ProxyPass /aspera/orchestrator/javascripts !
ProxyPass /aspera/orchestrator/my/images !
ProxyPass /aspera/orchestrator/my/stylesheets !
ProxyPass /aspera/orchestrator/my/javascripts !

# send the proxy request

  # @web_root starts with / if not nil
ProxyPass /aspera/orchestrator balancer://orchestrator_cluster/aspera/orchestrator timeout=1200
```

Disabling a Mail Server

You can disable a mail server so that the Email Notification plugin does not send emails. This functionality is useful in situations such as testing a workflow in a test/development environment, where you don't want emails to be sent out.

When the mail server referenced by the Email Notification step is disabled, the step still receives and outputs the correct data for the message it would otherwise have sent, and it completes successfully. However, it does not send an email notification.

To disable a mail server, click **Engine**, then click **Mailer Configurations** in the left navigation. To the right of the mail server name, unselect **Active?**.

Note: Even if you disable the mail server, users can still request that Orchestrator reset a forgotten password.

Remote Nodes

This section describes working with remote nodes.

Overview: Remote Nodes

Importance of Creating Remote Nodes

A remote node manages information needed to connect from a FASP Session, ASCMD, SSH, or TCP to a remote server. All plugins provide a dropdown arrow option to select a remote node for their plugin operations.

Without a remote node, you must input an IP address or host names, user ID, and password at multiple places in your workflows. If this information changes, your must change the information in multiple places. If you use a remote node and populate the information there instead, the only place the information changes in that node.

Types of Remote Nodes:

There are four types of remote nodes. The table below describes the circumstances under which each type should be used.

Node type	**Conditions for use**
Aspera FASP	The remote node has one of the following installed: • • Enterprise Server • Connect Server • Server On-Demand
SSH	The remote node accepts SSH connections.

Node type	Conditions for use
	Note: Unix (*nix) machines accept SSH connections by default on Windows machines, usually when software like Open SSH is installed.
IP	The remote node does not have the characteristics of either an Aspera FASP node or an SSH node (for example, a transcoding server or a quality check server).
Aspera Orchestrator	The remote node is also an Orchestrator server.

Creating a Remote Node

This article provides a procedure for creating a remote node. A remote node may be an Aspera FASP node (for example. a remote file watcher, remote file operation, or FASP transfer-controlled node), an SSH node (for example, an `ffmpeg` transcoding or remote execution), or an IP node (for example, Vantage transcoding from a SOAP API).

1. Click **Workflows** in the top menu, then click **Remote Nodes** in the left navigation menu.
2. Click the **New Node** button
3. In the Node Configuration dialog, enter the following information:

Field	Details	Required
Name	Name of node	Must provide node name or node ID
Node ID		Must provide node ID or node name
Comments		No
Address	IP address of node	Yes
Secondary addresses	Secondary IP addresses for node	No
Node type	Allowable values: Aspera Fasp, SSH, IP, Aspera Orchestrator	Yes
Orch login	Login for the `orchestrator` user.	Yes, for Aspera Orchestrator only
Orch API key	Orchestrator user's API key.	Yes, for Aspera Orchestrator only
Orch web root	Web root of Orchestrator node. If no value entered, default is `aspera/orchestrator/`	Yes, for Aspera Orchestrator only
Node OS	Node operating system	No
SSH Port	SSH port of node	Yes, if applicable
TCP Port	TCP port of node	Yes, if applicable
SSH/Fasp login	Login string for SSH/Fasp	No
SSH/Fasp password	Password for SSH/Fasp	No
Re-enter	To confirm, re-enter the password above	No

Field	Details	Required
Certificate location	Directory path for certificate	No
Additional attributes	Enter additional attributes	No

4. Test the remote node settings by following the procedure in "Testing a Remote Node" on page 87.

Editing a Remote Node

1. Click **Workflows > Remote Nodes**.
2. In the far-left pane, click the name of the node you wish to edit.
   The Remote Node Details pane appears to the right.
3. Click **Edit node information**.
   The Node Configuration dialog opens.
4. Edit the information as needed and click **Save**.

Testing a Remote Node

You can test your remote node settings as soon as they are created.

1. In the left pane of the Manage Remote Node screen, click the name of the new node that you want to test.
   The remote node details populate in the right pane.
2. Click **Edit node information**.
3. In the Node Configuration dialog, edit the information fields as needed, then click **Save**.

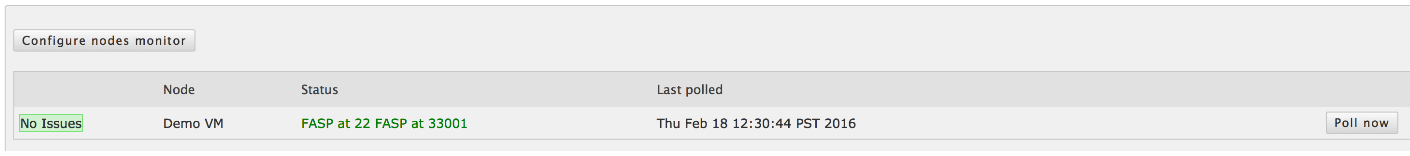
Removing a Remote Node

1. Click **Workflows > Remote Nodes**.
2. In the far-left pane, click the name of the node you need to remove.
   The Remote Node Details pane appears to the right.
3. Click **Remove node**, then click **OK** in the confirmation pop-up.

Remote Node Status Notifications

In the Node list (**Monitor > Nodes**) you can view status alerts at the listing for the new remote node that appears.

If a connectivity test is successful for a particular port, a green `No Issues` label displays, which means that tests on all the ports are responding, and the Status for the port appears in green text. For example, in the notification below, SSH connectivity and FASP connectivity on their respective ports are listening and active.



	Node	Status	Last polled
No Issues	Demo VM	FASP at 22 FASP at 33001	Thu Feb 18 12:30:44 PST 2016

If connectivity for a port fails, a red Alerts label appears, and the Status is displayed with red text.

Warning labels appear in yellow background.

The screen has three buttons, as follows:

Poll now
Performs a force poll of the monitor.

Deactivate

Deactivates the monitor. The node monitor no longer polls for connectivity tests and the status changes to `Not running`.

Edit

Provides options for editing the monitor configuration settings (changing the name of the monitor, updating the notification type from `Alerts` to `Warnings` for a port, changing the test type).

Note: A remote node associated with a particular node monitor cannot be changed once it is added to the remote node monitor. If you want to change the remote node, you have to create a new node monitor and configure the remote node in that new node monitor.

Generating a Certificate for a Node in a Workflow

You can easily generate a certificate for a specific node associated with your workflows.

1. In the top menu, click **Workflows**.
2. In the left navigation menu, click **Remote Nodes**.
3. Click the **Generate Certificate** button.
4. Enter a name for the certificate and then click **Generate**.

Plugins

This section describes working with plugins, and also contains lists of Orchestrator plugins by category.

Overview of Plugins

Plugins allow you to add the functionality of both IBM Aspera and external services into your workflow.

Each plugin that you add in the workflow designer becomes a step in the workflow, so that you can apply various actions to your files—transfer, encode, and so on.

To download an updated version of a plugin, go to the link provided by IBM Aspera on Cloud, https://aspera.pub/ZvFN-Y0. Plugins are normally are not platform- or version-dependent—unless explicitly stated in the help file for the plugin—and can be imported (upgraded) in the Orchestrator UI.

Note: In order to download plugins, users must install IBM Aspera Connect. See https://www.ibm.com/aspera/downloads/ for more information.

The full list of available plugins are on the **Plugin Manager** page. Click **Engine > Plugins**.

Searching for a Plugin

1. Click **Engine > Plugins**.

   The Plugin Manager page opens.
2. In the search box, enter the name of the plugin you are looking for.

   The Action Type list displays the plugins that match the search string. A longer search string will return a more specific list.

Searching for a Archived Plugin

1. To locate the plugin for which you need to view archived versions, go to the **Plugin Manager** page.

   Click **Engine**, then click **Plugins** in the left navigation.
2. On the **Plugin Manager** page, search for the plugin.

   Click the **More** button () to the right of the plugin, then click **Archive List**.
3. A dialog showing all available archived versions of the plugin appears.

Enabling Plugins

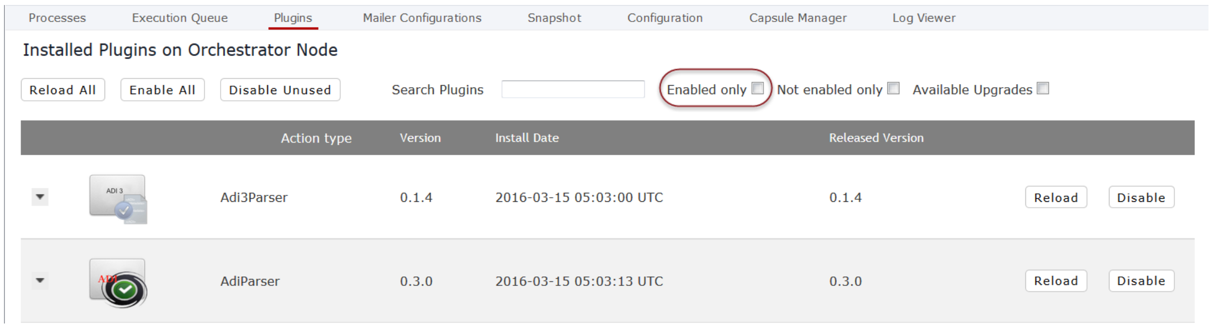
You must enable a plugin before you can use it in a workflow.

1. Click **Engine > Plugins** to open the Plugin Manager page.
2. In the list on the page, locate the plugin you need to enable.
3. To enable one or more plugins, click **Enable** to the right of the plugin name.
4. To enable all plugins at once, click **Enable All**.

Disabling a Plugin

This procedure describes how to disable an Orchestrator plugin.

1. (Preliminary step) To locate the plugin you need to disable, do one of the following:
   - Follow the procedure in "Searching for a Plugin" on page 88.
   - On the Installed Plugins screen (**Engine > Plugins**), scroll through the Action Type list.
   - On the Installed Plugins screen (**Engine > Plugins**), select **Enabled only** to view the list of plugins that are not already enabled.



Installed Plugins on Orchestrator Node

Action type	Version	Install Date	Released Version
Adi3Parser	0.1.4	2016-03-15 05:03:00 UTC	0.1.4
AdiParser	0.3.0	2016-03-15 05:03:13 UTC	0.3.0

2. Click **Disable** (to the right of the plugin name).

Importing a Plugin

Importing a plugin enables that plugin inside Orchestrator's plugin repository.

If that plugin was previously enabled, all components and settings for that plugin (code, version, functionality) will be overwritten by new instance of the plugin being imported. If no prior instance of the plugin is found in the repository, which is usually the case with newly developed plugins, the plugin being imported will create a new entry in the plugin manager.

To download an updated version of a plugin, go to the link provided by IBM Aspera on Cloud, https://aspera.pub/ZvFN-Y0. Plugins are normally are not platform- or version-dependent—unless explicitly stated in the help file for the plugin—and can be imported (upgraded) in the Orchestrator UI.

Note: In order to download plugins, users must install IBM Aspera Connect. See https://www.ibm.com/aspera/downloads/ for more information.

1. On the Plugin Manager page (**Engine > Plugins**), click **Import Plugin**.

   The Upload an Action Plugin File for Import dialog appears.
2. Click **Choose File** and select a plugin file, then click **Open**.

   Note: A valid plugin has a `.plugin` extension, and it has a version number not lower than `0_0_1`.

   The name of the plugin appears next to the **Choose File** button.
3. Click **Import** to load the plugin.

   The Plugin Manager page displays a status message regarding the import process (a success message if the import completes, and an error message if the import is unsuccessful).

Reloading a Plugin

Reloading a plugin forces Orchestrator to incorporate the newest code for that plugin.

1. (Preliminary step) To locate the plugin you need to reload, do one of the following:
   - Follow the procedure in "Searching for a Plugin" on page 88.
   - On the Installed Plugins screen (**Engine > Plugins**), scroll through the Action Type list.
2. Click **Reload** (to the right of the plugin name).

   A Reload Status dialog, displaying the status of all associated mongrels, appears. Click **OK** to dismiss the dialog.
3. Open the Orchestrator Processes screen (**Engine > Processes** and click **Force maintenance** to complete the plugin reload process.

Exporting a Plugin

1. To locate the plugin you need to export, click **Engine > Plugins** and scroll through the Action Type list to find the desired plugin.
2. To the right of the plugin name (at the end of the row), click the **More** icon, then click **Export**.
3. Click the dropdown arrow to the left of the plugin name and click **Export**.
4. Save the plugin file to the desired directory location.

Archiving a Plugin

Archiving a plugin allows you to easily roll back from a plugin upgrade.

This process saves a copy of the plugin in Orchestrator's `actions_archive` directory.

Your archive directory is found in the following location:

```
/opt/aspera/var/archive/orchestrator/
```

1. (Preliminary step) To locate the plugin you want to archive, do one of the following:
   - Follow the procedure in "Searching for a Plugin" on page 88.
   - On the Installed Plugins screen (**Engine > Plugins**), scroll through the Action Type list.
2. Click the dropdown arrow to the left of the plugin name and click **Archive**.

   The Installed Plugins screen reloads; if archiving completes successfully, a message showing the archive directory location is displayed.

Configuring Users for Remote Plugins

The table below clarifies what type of user to define when using remote plugins in Orchestrator.

Requirements for Defining Users in Remote Plugins

Plugin Name	Type of User to Configure	User Shell	User Docroot Applied to Paths	ascmd Running in the Background - Required	Aspera Software (EntSrv, P2P) Installation - Required	Aspera License - Required
Remote File Watcher or Remote	SSH system user and Aspera transfer user	`/bin/bash` or	Yes	Yes	Yes	No

Plugin Name	Type of User to Configure	User Shell	User Docroot Applied to Paths	ascmd Running in the Background - Required	Aspera Software (EntSrv, P2P) Installation - Required	Aspera License - Required
Folder Watcher		`/bin/aspshell`				
Remote File Operation	SSH system user and Aspera transfer user	`/bin/bash` or `/bin/aspshell`	Yes	Yes	Yes	No
Remote Execution	SSH system user	`/bin/bash`	No	No	No	No
FASP Transfer	Node API user	n/a	Yes	No	Yes	Yes

Viewing the Archive List

Orchestrator maintains a list of archived versions for each plugin.

1. (Preliminary step) To locate the plugin for which you need to view the archive list, do one of the following:
   - Follow the procedure in "Searching for a Plugin" on page 88.
   - On the Installed Plugins screen (**Engine > Plugins**), scroll through the Action Type list.
2. Click the dropdown arrow to the left of the plugin name and click **Archive List**.

   A dialog showing all available archived versions of the plugin appears.

Revision History Options for Plugins

For each plugin, users can view all revisions that are made to the template. This applies to both global and configured templates.

To view the revision history for a plugin, do the following:

1. Click **Workflows > Workflow Actions**).
2. To the right of the plugin name, click the **More** icon.
3. Click **View revision history**.

   Users can click one of the following options:

   View

   View the configuration for the template (read-only).

   Delete

   Delete the template from the revision.

   Publish

   Replace the current template with the selected revision.

Asset Management Plugins

Plugin Name	Notes
Dalet Asset Management	Enables the use of the APIs listed in the Dalet Asset Management System
Evertz Mediator	Provides ability to interact with Evertz Mediator system
ShowMgr	Interacts with ShowMgr, using API calls.
The Platform	Creates, reads, and updates The Platform objects.
Xytech MediaPulse	Queries the MediaPulse API and returns MediaOrders and the corresponding source files.

Code Sample Plugins

Plugin Name	Notes
Hello world	Reference implementation for a simple Action plugin illustrating the API

File Operations Plugins

Plugin Name	Notes
Archive manager	Manages a file archive by deleting or compressing aging entries
Aspera Node API	Performs file system operations such as search, rename, and delete. Acquires from the node information needed to set up a transfer, including token, SSH user, and port. Queries the status of a node (`ping`)
Aspera Node API Transfer Watcher	Monitors transfers to an Aspera server managed by ATCM
Aspera Node Search	Performs a file system operations search through the Aspera Node API
Dependency Checker	Confirm the existance and stability of the files that are specified in a list
Diva archive	Archives or restores files to and from the DIVArchive system. It implements the DIVA REST WEB service API 2.1
Exiftool XMP Tagger	Retrieves metadata information about an image, video or audio file using the ExifTool application. It also can insert a metadata tag in an image file (but not in a video file)
FFProbe Information	Submits a file to FFProbe to grab media properties and technical metadata
File archiving operation	Executes file archiving operations on Orchestrator locally-mounted file systems
File generator	Creates a file using dynamic content provided from parameters or the output from previous steps in the workflow.Although it can generate any text file, it is primarily used to generate XML files to be used by other steps or other applications. The user defines the template to be used, the variable content required, and variable names, and maps the variable inputs
File Info	Extracts multiple information about a local file, given its full path
ISM M3U8 Parser	Parses an ISM file and a M3U8 file
Local file operation	Executes file operations on Orchestrator locally-mounted file systems

Plugin Name	**Notes**
Local file permission	Performs the chmod and chown file operations after a file has been downloaded
MD5 checksum	Calculates or verifies against a previously calculated checksum the MD5 checksum for a specified file
MediaInfo	Retrieves media information about a video file or audio file using the MediaInfo application
Mediabin asset management	Performs asset management on a Mediabin server. Supported operations include asset listing and retrieval (with download)
Path Conversion	Converts a file path from one mount point to another
Remote file operation	Executes file operations on a remote node.
SGLFlashnet Archiving	Performs archiving and restoring of files.
Spreadsheet parser	Parses a spreadsheet to extract its value in a structured, usable order.
Symantec PGP Encryption	Programatically encyrpts and decrypts a file using the Symantec PGP command-line tool
XML Parsing	Parses an XML file.

File Transfer Plugins

Plugin Name	**Notes**
Ascp Client	Performs a file transfer upload or download operation to/from a server using Aspera's FASP protocol. In order for it to work, an Aspera client or server product needs to be installed on the Orchestrator server
Aspera Files Package Delivery	Delivers packages to Aspera on Cloud
Aspera Node API Transfer	Transfers to an Aspera server managed by ATCM
Aspera OTFV (Out of Transfer File Validation)	Detects validation requests for both transfer sessions and files and allows you to implement a validation workflow in Orchestrator
Console smart transfer operation	Initiates a Smart Transfer Template in Aspera Console
FASP Controller	Modifies the parameters of a running transfer using Aspera's FASP protocol
FASP Stream	Provides the ability to start, monitor and cancel a stream between two Aspera transfer servers using FASP Stream
FASP Transfer	Initiates a server-to-server transfer using Aspera's FASP protocol
Faspex Delivery	Generates Faspex packages and exposes upload links
FTP transfer	Moves files using the FTP protocol
FTPS transfer	Moves files using the FTPS protocol
FXP Transfer	Moves files using the FXP protocol
HTTP Transfer	Uploads, downloads and deletes files from a remote host using HTTP or HTTPS (Hypertext Transfer Protocol or Hypertext Transfer Protocol over SSL)

Plugin Name	Notes
iTunes transporter	Delivers audio and video content, in a pre-generated iTunes Store package, to the iTunes Store
Robocopy	Provides file copy operations on Windows.
SCP transfer	Can upload or download files to or from a remote host using SCP (secure copy over SSH).
SFTP transfer	Provides file transfer functionality to an SFTP server.
Shares Transfer Setup	Returns the transfer specification for a transfer with a share in Aspera Shares, using a Shares user credentials.
WebDAV Operation	Uploads, downloads, posts commands, makes a directory, gathers properties, copies, moves, deletes, or checks files on a remote host which supports WedDAV protocol (an extended version of the HTTP protocol).

File Transformation Plugins

Plugin Name	Notes
ADI Transformation	Converts ADI 1.1 version compliant XML files to ADI 3.0 version and vice versa
Imagemagick	Retrieves media information about an image or applies transformation to that image using the Imagemagick application
MacCaption	Performs captioning operations using the Telestream MacCaption application
MXF Legalizer	Enables MXF files compliant by implementing the MXF Legalizer v2.5.2 SOAP API
Subler Operation	Performs file transformation operations on a Mac system using the Subler toolset.
XSLT Transformation	Transform an XML file by using XSLT.

FIMS Plugins

Plugin Name	Notes
FIMS Transfer	Submits a transfer request to a service that is compliant with the FIMS transfer specification
FIMS Transformation	Submits a transform request to a service that is compliant with the FIMS transform specification

Integration Plugins

Plugin Name	Notes
Amazon Dynamo DB Operation	Integrates Amazon DynamoDB's key-value and document database on the AWS platform
Amazon Glacier Operations	Integrates with Amazon Glacier services
Amazon MediaConvert	Transcodes video content. The plugin can be used with a custom payload
Amazon S3 Operation	Executes file operations (such as upload) on an Amazon S3 storage
Amazon Simple DB	Integrates Amazon Simple DB's database storage and operations

Plugin Name	Notes
Amazon SNS Operation	Uses managed messaging with the Amazon Simple Notification service
Amazon SQS Plugin	Performs an action in an Amazon SQS queue
Amazon SWF Operation	Executes Amazon Simple Workflow API calls
Amazon Transcribe	Sends audio and video to the AWS Amazon Transcribe service and receives transcribed data in JSON format
AMQP message	Inserts messages into an AMQP queue
Aspera on Cloud Management	Manages functions in Aspera on Cloud, including creating and deleting dropboxes and creating and deleting users
Aspera Shares Management	Uses the Shares command line utility to perform Shares tasks
Async Remote Execution	Enables remote content distribution with Aspera Sync
CatDV	Executes a REST requests to CatDV server
Corba operation	Makes a CORBA call to a CMS/AMS using the JacORB utility
Curl operation	Executes an HTTP query to a web service
Custom progress notifier	Intercepts the progress notification of a step and adds a customized behavior to it; for example, logging to an external log or RSS feed
Database query	Executes an SQL query on a database and formats its result
Facebook Upload	Uploads videos to Facebook
Frame.io Operation	Integrates Frame.io's professional video review and collaboration capabilities
Google Cloud Pub/Sub	Accesses the Google Cloud Pub/Sub messaging service
Google Cloud Storage	Accesses the Google Cloud Storage service
Influx DB Operation	Integrates Influx DB's time series database, which can handle high write and query loads
Jira Issue Creation	Creates Jira tickets
Kafka Stream	Writes workflow execution data to an external monitoring system
Local execution	Invokes a third-party application or a script running locally on the workflow server. The integration mechanism conforms to the CGI-BIN protocol used by web and application servers (for example, input parameters are provided as an environment variable, and output is provided on the `stdio`)
MediaMate Operation	Encodes and insert captions with MediaMate
Mongo DB Operations	Accesses MongoDB functionality
Remote execution	Executes a script or executable on a remote node
REST request	Periodically executes a REST query to a web service. It can be used for a one-off request or as a polling mechanism.
SOAP request	Periodically executes a SOAP query on a web service. It can be used for a one-off request or as a polling mechanism.
Softlayer API	Accesses the SoftLayer Network Storage service, which adds control over IBM cloud storage
Watson Language Translator	Accesses the Watson Language Translator service

Quality Control Plugins

Plugin Name	**Notes**
ADI 3.0 Parser and Validator	Parses or validates the contents of ADI 3.0 XML
ADI Parser and Validator	Parses or validates the contents of ADI XML
Aurora file verification	Enables the submission of a file verification request to a Digimetrics Aurora Server
Baton file correction	Submits a file correction request to an Interra Baton Server
Baton file verification	Submits a file verification request to an Interra Baton Server
Baton xml report	Retrieves an XML report from Baton.
Cerify File Verification	Submits a quality control task to a Cerify Server
Irt mxf analyser	Submits a file verification request to an IRT MXF Analyser server.
Mx Fixer	Analyzes an MXF file for AS11 compatibility using Metaglue MX Fixer
Orion	Provides integration with Orion OTT Monitor
Pulsar content verification	Submits a content verification task to a Pulsar server
QScan File Verification	Test your digital media content with QScan
Quasar Venera QC	Submits a file verification request to a Quasar server
Vidchecker verification	Executes and monitors quality control verifications on a VidChecker server
VQBif Analyzer	Executes `VQBif` commands on a remote node
Watson Speech to Text	Gets speech transcriptions with the IBM Watson Speech to Text service
Watson Text to Speech	Gets speech synthesis with IBM Watson Text to Speech service
Watson Video Enrichment	Gets structured information from video files with the IBM Watson Video Enrichment service
XSD Validation	Validates an XML file against an XSD template

Resource Plugins

Plugin Name	**Notes**
Load manager	Balances a request among a pool of resources that have specific capacities. Pauses a workflow until enough resources become available and routes the request to the least-taxed resource

System Plugins

Plugin Name	**Notes**
Array fan-out	Executes, for each value contained in an array, the step between the fan-out and the funnel-in, in parallel
Exit status	Sets up a special exit status message within the workflow
Federated workflow	Launches workflows on a remote Orchestrator instance and either gets the ID and outputs or just gets the ID and does not wait for the outputs of the remote workflow

Plugin Name	Notes
Filter	At run time, selects the step to execute next based on defined conditions
Funnel-in	Pairs with a fan-out and converges the multiple parallel threads back into one
Merge point	Reconciles the inputs coming from different branches of logical statements. It provides the ability to define pairs of inputs that get merged. Such steps can be used without inputs to clarify branching in a complex workflow
Orchestrator Alert Entry	Enters custom alerts into Orchestrator
Snapshot	Takes a snapshot of the Orchestrator configuration.
Sub-workflow	Wraps an entire workflow—defined separately—into a step that can be used in another workflow, thus facilitating the creation of a complex workflow composed of sub-workflows.It is similar to the Workflow plugin; however, Sub-workflow and Workflow Launcher differ in the following ways: • Workflow Launcher launches another workflow and then the current workflow immediately proceeds; this behavior is asynchronous compared with Sub Workflow, which waits for the launched workflow to return before proceeding with the current workflow. • A Workflow Launcher step is `Complete` when it successfully launches the workflow to which it is configured, whereas a Sub Workflow step is `Complete` when the sub-workflow it launches has completed its execution.
User lookup	Looks up information about Orchestrator users identified by ID, name, login, or email address.
Work Order Metadata	Provides information about work orders and customizes the label display for them.
Workflow Launcher	Launches one workflow from within another. It is similar to the Sub-workflow plugin; however, Workflow Launcher and Sub-workflow differ in the following ways: • Workflow Launcher launches another workflow and then the current workflow immediately proceeds; this behavior is asynchronous compared with Sub Workflow, which waits for the launched workflow to return before proceeding with the current workflow. • A Workflow Launcher step is `Complete` when it successfully launches the workflow to which it is configured, whereas a Sub Workflow step is `Complete` when the sub-workflow it launches has completed its execution.

Transcoding Plugins

Plugin Name	Notes
Akamai transcoding	Starts and monitors a Transcoding job using the Akamai Transcoding platform
Alchemist Snell OD operation	Submits a file conversion job to a Snell OD server
Amazon Elastic Transcoding	Queue to Amazon Elastic Transcoder
Amberfin transcoding	Submits a file transcoding job to a Amberfin server
Ateme Transcoding	Submits file transcoding job to a Akamai server
Bitmovin Encoding	Submits file transcoding jobs to a Akamai server

Plugin Name	Notes
Cambria FTC Transcoder	Transcode video files with Cambria FTC
Carbon/WFS transcoding	Submits a file transcoding job to a Harmonic Carbon Coder/WFS server
Digital Rapids TM Transcoding	Submits a file transcoding job to a DigitalRapids Transcode Manager server
Dolby VM600	Process audio files with Dolby Program Optimizer VM600
Elemental MediaConvert	Process video files and clips with AWS Elemental MediaConvert
Elemental Transcoding	Submit a file transcoding job to an Elemental server
Emotion Operation	Interacts with the Emotion Engine API for audio processing
Encoding.com Transcoding	Submits a file encoding job to a Encoding.com server.
Envivio Transcoding	Submit file encoding jobs to an Envivio VOD Encoder through a 4Balancer manager
Eolementhe Workflow Manager	Call and monitor an Eolementhe workflow
Episode Transcoding	Submits a file transformation job to a Telestream Episode engine
FFMPEG Transcoding	Submits and controls a file transcoding operation using the FFMPEG toolset
FlipFactory Transcoding	Submits a file transformation job to a Telestream FlipFactory server
HandBrake Transcoding	Submits and controls a file transcoding operation using the HandBrake toolset
Hybrik Transcoder	Transcodes your digital content
Marquis MEWS (Medway)	Uses the APIs listed under the Medway Web Service solution of Marquis
Minnetonka audiotoolserver	Processes, converts, encodes, and decodes audio files with the Minnetonka AudioTools Server. It implements the AudioTools SOAP-based WEB service API
Mog MXF Speed Rails	Runs a MOG mxfSPEEDRAIL workflow (for example, mxf transcoding + ingest to Avid)
Timecode manager	Enables timecode manipulations like conversions, sums, and differences.
Unified Packager	Creates a server manifest file with the `mp4split` tool
Vantage Transcoding	Submits a file transcoding job to a Vantage server.
VQMA Analyzer	Performs a Video Analysis operation using the VQMA toolset on a Windows node.
Xf converter	Converts (rewraps) audio and video wrapper files (AVI, GXF, MOV, MP4 and MXF) with OpenCube XFConverter. It implements the XFConverter SOAP-based WEB service API 2.0.
Zencoder transcoding	Submits a file transformation job to the Zencoder cloud service.

Triggers Plugins

Plugin Name	Notes
AMQP trigger	Inserts messages into an AMQP queue
Amazon SQS Trigger	Trigger your workflow when a message is released from a queue in Amazon Simple Queue Service. This plugin is called "Amazon SQS Watcher" when used in a workflow.
Aspera central watcher	Senses the initiation or completion of a transfer on an Aspera server
Aspera Files Package Watcher	Detects when a new package is uploaded to Aspera on Cloud
Aspera Node API Transfer Watcher	Monitors transfers to an Aspera server managed by ATCM
Aspera node file watcher	Detects files and folders matching a certain pattern on a remote Aspera server with the Aspera Node API
Custom trigger	Used to script a customized behavior in Ruby and to keep executing it until a particular condition is met
Database trigger	Periodically executes an SQL query on a database and triggers when a new `strow` is found
Email inbox watcher	Triggers workflows based on the arrival of incoming packages in a IMAP inbox
Faspex inbox watcher	Triggers workflows based on the arrival of incoming packages in a Faspex inbox
Faspex Package Monitor	Monitor a package that has been uploaded to the Faspex server
FTP Folder Trigger	Detects folders on a remote node using the FTP protocol
FTP trigger	Detects files on a remote node with the FTP protocol
Local file watcher	Creates triggers based on the presence—in a specified folder—of files that match a specified pattern
Local folder watcher	Waits for the arrival of a folder matching a certain pattern and (optionally) waits for the stability of all files in that folder
Remote file Watcher	Waits for the arrival of files matching a certain pattern on a remote Aspera server.
Remote Folder Watcher	Waits for the arrival of folders matching a certain pattern on a remote Aspera server.
Schedule trigger	Senses the initiation or completion of a transfer on an Aspera server.
SCOM notification	Submits a log `stentry` to the Windows Event Log. The execution can only occur on a Windows host.
SFTP trigger	Provides hotfolder functionality on a remote server with SFTP.
SOAP request listener	Waits for a SOAP message and extracts its content.

User Interactions Plugins

Plugin Name	Notes
Console notification	Inserts an stentry in the Aspera Console representing Orchestrator events

Plugin Name	Notes
Email Notification	Sends email as part of the execution of a workflow
RSS Feed Reader	Reads an RSS feed; can be used as a trigger.
SCOM Notification	Submits a log `stentry` to the Windows Event Log. The execution can only occur on a Windows host.
Slack Notification	Provides the ability to interact with Slack
User Input	Provides an interactive way to present information and collect data from users through web forms within the Orchestrator control application.

Virus Scan Plugins

Plugin Name	Notes
ClamAV check	Checks a file, folder, or set of files for viruses using the clamAV utility
ICAP virus scan	Communicates with an ICAP-enabled virus scanner engine to quarantine infected files.
McAfee virus check	Checks a file, folder, or set of files for viruses using the McAfee anti-virus application
Symantc Icap DLP Operations	Makes an ICAP call to the Symantec DLP Engine.
Symantec Icap Virus Scan Operations	Makes and ICAP call to the Synamtec Antivirus Engine.

Other Utilities Plugins

Plugin Name	Notes
Collection manager	Creates a name value pair collection (Hash) or adds a field to one
Custom Python	Writes and runs custom Python scripts in Orchestrator, using Python 3.
Custom Ruby	Used to script a customized behavior in Ruby
File journal log stentry	Logs a state for a file in the Orchestrator file journal
Input Manipulation	Manipulates inputs using data operations based on the input type
Load balancer	Manages a round-robin distribution to a pool of resources.
Mass Parameter Setter	Used to set the values of multiple parameters and can be used to store defaults values or a set of predefined values
Password Generator	Generates random strings that can be used as passwords. The length and constraint of the generated random strings can be configured
Queue stager	Stages items into a managed queue and provides a way to pause a workflow until all items have been released from the queue. While staged in the queue, items can be reordered by a user with appropriate privileges
Regex matcher	Parses a regular expression and extracts parts of it
Resource manager	Helps manage a limited pool of resources by providing a way to pause a workflow until a resource becomes available.
SharedState Operations	Creates and retrieves shared-state objects.

Plugin Name	Notes
Simple Parameter Lookup	Provides a framework to associate miscellaneous values to a parameter at design time and retrieve these associated values at run time based on the value of a run-time parameter.
UUID Generator	Generates a unique ID with specific formatting, for example, `39fa4e20-0f32-0132-d394-282066598c75`.

Plugin Templates for Workflows

This section contains procedures for working with action templates, which are the configuration fields for plugins.

Creating Workflow Action Templates and Global Templates

You can configure plugins to use as action templates to be reused in steps in one or more workflows. You can also create *global templates* from plugin configurations that were created for a step in an individual workflow; this procedure adds those configurations to the list of action templates that are available for reuse.

Creating a Workflow Action Template

1. Click **Workflows** in the top menu, then click **Workflow Actions** in the left navigation menu.

   A list of action templates appears, organized by category.
2. Click any plugin in the list.

   Note: If the plugin you need is not listed, click **Engine** in the top menu and then click **Plugins** in the left navigation menu. Review the list to confirm whether the plugin is enabled. If not, click **Enable** next to the plugin name.
3. To open the configuration dialog, click the **New** button.

   Note: If a template already exists for this plugin, it will appear in the Action Template list. You can edit this existing plugin template by clicking the template name.
4. Enter data in the template fields and click **Save**.

   The new template appears in the Action Template list.

Creating a Global Template for Workflows

1. Click **Workflows** in the top menu, then click **Workflow Designs** in the left navigation menu.
2. Open one of the workflows in the list.
3. Right-click a step in the workflow, then click **Change Action Template**.
4. To the right of one of the actions in the list, click the globe icon.

   The template now appears in the Action Template list under **Workflows** > **Workflow Actions**, and can be reused in other workflows.

Editing a Workflow Action Template

1. Click **Workflows** in the top menu, then click **Workflow Actions** in the left navigation menu.

   A list of action templates appears, organized by category.
2. Click any plugin in the list.

If the plugin you need is not listed, click **Engine** in the top menu and then click **Plugins** in the left navigation menu. Review the list to confirm whether the plugin is enabled. If not, click **Enable** next to the plugin name.

3. In the Action Template list, click the name of the template you want to edit.
4. In the dialog, update the template information as needed and do one of the following.
   - Click **Modify template** to update the template.

Duplicating a Template

1. Click **Workflows > Workflow Actions**.

   A list of action templates appears, organized by category.
2. Click the name of the template you want to duplicate.

   A panel listing all action templates for this plugin appears to the right of the plugin list.
3. In the Action Template list, to the right of the template name, click the **More** button and click **Duplicate**.

   A duplicate template with the name **Copy of *template_name*** appears in the Action Template list.

Exporting and Importing Workflow Templates

Exporting a Workflow Template

This section describes how to export a workflow template. *Exporting* a workflow template means saving a copy on a designated server location. You can export to a local server, in which case the copy serves as a backup to be imported later into the local Orchestrator instance, or you can export to a remote server, which allows you to import the copied template into a remote Orchestrator instance.

1. On the Orchestrator instance from which you are exporting the workflow template, click **Workflows** in the top menu, then click **Workflow Actions** in the left navigation menu.

   A list of action templates appears, organized by category.
2. In the list of available plugins, click the plugin you need to export.

   If the plugin you need is not listed, click **Engine** in the top menu and then click **Plugins** in the left navigation menu. Review the list to confirm whether the plugin is enabled. If not, click **Enable** next to the plugin name.
3. In the plugin templates panel, click the **More** icon, then click **Export**.
4. Save the plugin `.yml` file to the desired server location.

Importing a Workflow Template

Importing a workflow template involves importing the local copy directly into the local Orchestrator or remote Orchestrator instance.

1. In the Orchestrator instance where you want to import the workflow, click **Workflows** in the top menu, then click **Workflow Actions** in the left navigation menu.

   A list of action templates appears, organized by category.
2. In the list of available plugins, click the plugin you need to import.

   If the plugin you need is not listed, click **Engine** in the top menu and then click **Plugins** in the left navigation menu. Review the list to confirm whether the plugin is enabled. If not, click **Enable** next to the plugin name.
3. In the plugin templates panel, click the **Import** button.
4. Click **Choose File** and select a file to open, then click **Upload**.

Workflows

This section contains procedures for working with workflows.

Creating a Workflow

1. Click **Workflows > Workflow Designs > New Workflow**.
2. In the Workflow Core Parameters dialog, enter the name, description, comments, and desired parameters for the other fields.
   - The system enters a default workflow ID, which can be edited.
   - The Purge After (Days) and Clean Up After (Days) fields allow users to input a customized number of days for the respective activity which bypasses the value that is already configured system-wide. For example, test workflows and sample workflows might have a low value for Purge After and Clean up After but production workflows might have a higher value for those fields.

   Note: By default, new workflows are created in the root folder. If you would like to change the location of a workflow to a different folder, see "Changing the Folder Location for a New or Existing Workflow" on page 110.
3. When finished, click **Create**.

   The workflow designer screen appears.

Editing a Workflow

Editing the Workflow Core Parameters

Click **Workflows**. To the right of the workflow you need to edit, click the **More** menu (**. . .**), then click **Edit core parameters**.

You can edit parameters such as the number of days before cleanup and the throttle limit for workers.

Undoing and Redoing Actions

To access the workflow designer, click **Workflows** and open a workflow in the list.

UI options: To undo and redo your changes to a workflow, click the undo arrow or redo arrow at the top of the left panel.

Keyboard shortcuts: To undo your changes, press Ctrl+Z (on Windows) and Cmd+Z (on Mac). To redo your changes, press Control+R (Mac). There is no equivalent keyboard shortcut for redoing an action on Windows.

Note: These keyboard shortcuts may conflict with your system's existing keyboard shortcut mappings.

Selecting, Moving, and Deleting Multiple Work Steps

To access the workflow designer, click **Workflows** and open a workflow in the list.

Selecting

To select multiple work steps in order to move or delete them as a group, press Ctrl, Shift, or Cmd, then click the steps one at a time. You can add additional steps to a group of steps you have already selected by holding down Ctrl, Shift, or Cmd and clicking the additional steps one at a time.

Alternatively, you can click the canvas (the background grid pattern in the window) next to a work step that you want to select, then click the left mouse button and drag the mouse to highlight an area that selects that work step and additional work steps (note: this procedure only selects steps that are next to or immediately above or below each other).

Note: The blue highlighting disappears after the steps are selected, but this does not affect their "selected" status.

To remove an individual work step from a group of selected steps, press Ctrl, Shift, or Cmd and click that step.

To select all work steps in the designer, press Ctrl+A (Windows) or Cmd+A (Mac).

Moving

After selecting multiple work steps, you can move them as a group by releasing the selection key (Ctrl, Shift, or Cmd), clicking one of the selected steps, and dragging to a new location—which moves all the selected steps together.

Deleting

After selecting multiple work steps, you can delete them as a group. Right-click one of the selected work steps, then click **Batch Delete**.

Highlighting Workflow Errors

To access the workflow designer, click **Workflows** and open a workflow in the list.

When a work step is improperly configured—for example, it does not have any connections, or it is missing a required input—it is highlighted with a red border that indicates an error when the user tries to save or publish the workflow. Hovering over the work step opens a tooltip that lists the errors which need to be fixed.

Changing the Workflow Description

1. Click **Workflows**.
2. In the Workflow Definitions list, click the dropdown arrow next to the desired workflow and click **Edit descriptions**.
3. In the Edit Workflow Description dialog, edit the information as needed, then click **Update Core Parameters**.

Copying Work Step Inputs and Outputs

1. Click **Work Orders**.

   The **View by Work Order** tab is highlighted by default.
2. Click a work order in the list in the left pane.
3. In the list of work order instances that appears in the right pane, click a work order instance.

   A list of work steps appears.
4. Click a work step to open a display of inputs and outputs.
5. Copy the input or output.

   - Click the Copy icon ( ). The desired text is selected and copied to the clipboard. The confirmation message `Copied!` displays for 2 seconds.
   - To copy a variable with more than 1024 characters, click **More** (which appears next to its listed value). This opens a dialog displaying the entire variable value. Click the Copy icon ( ) to select and copy the variable to the clipboard.

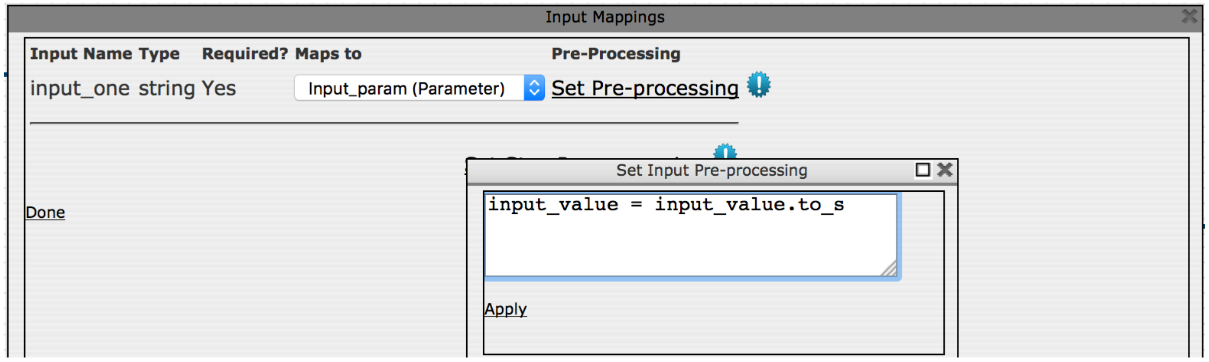
     The confirmation message `Copied!` displays for 2 seconds.
6. Paste the input or output value by following the procedure in "Setting Pre-processing and Post-processing for Work Steps" on page 105.

Setting Pre-processing and Post-processing for Work Steps

In the workflow designer, the user can set pre-processing for each input and post-processing for each output of a work step in a workflow.

Pre-processing

1. Click **Workflows** and open a workflow from the list.
2. Right-click a work step and click **Map Inputs**.
3. In the Input Mappings dialog, click **Set Pre-processing**.
4. In the **Set Input Pre-processing** popup, enter the input value, then click **Apply**.



Input Name	Type	Required?	Maps to	Pre-Processing
input_one	string	Yes	Input_param (Parameter)	Set Pre-processing

```
input_value = input_value.to_s
```

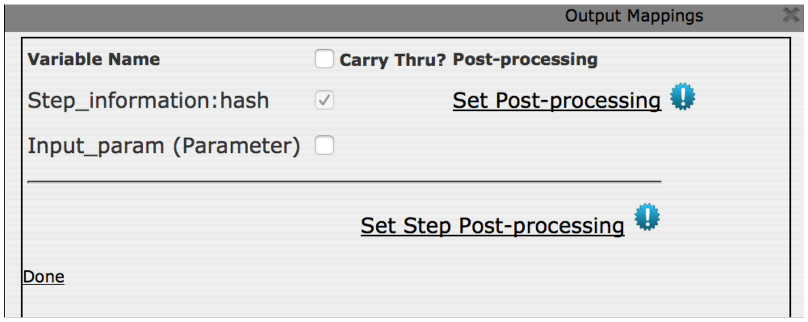
Note: If you copied an input value by following the procedure in "Copying Work Step Inputs and Outputs" on page 104, press Ctrl+V to paste the value.

Post-processing

1. Click **Workflows** and open a workflow from the list.
2. Right-click a work step and click **Map Outputs**.
3. In the Output Mappings dialog, click **Set Post-processing**.
4. In the **Set Ouput Post-processing** popup, enter the output value, then click **Apply**.

Note: If you copied an output value by following the procedure in "Copying Work Step Inputs and Outputs" on page 104, press Ctrl+V to paste the value.

To indicate that pre-processing or post-processing is applied, an icon ( ) is displayed. For example:



Variable Name	Carry Thru?	Post-processing
Step_information:hash	✓	Set Post-processing
Input_param (Parameter)	☐	

Copying and Deleting Work Steps

Orchestrator includes functionality for copying and deleting work steps in the workflow designer.

Copying a Work Step

To copy a step in a workflow, do the following:

1. Right-click the step and click **Copy Step**.
2. Right-click anywhere on the canvas and click **Paste from clipboard**.

Deleting a Work Step

To delete a step in a workflow, right-click **Delete Step**.

Note: To undo your action, press Ctrl+Z.

Launching a Workflow

1. Click **Workflows**.
2. In the Workflow Definitions list, click the dropdown arrow next to the workflow you want to launch and click **Launch**.
3. To launch the workflow, click **OK**.
4. Click **Work Orders** to view the workflow execution.

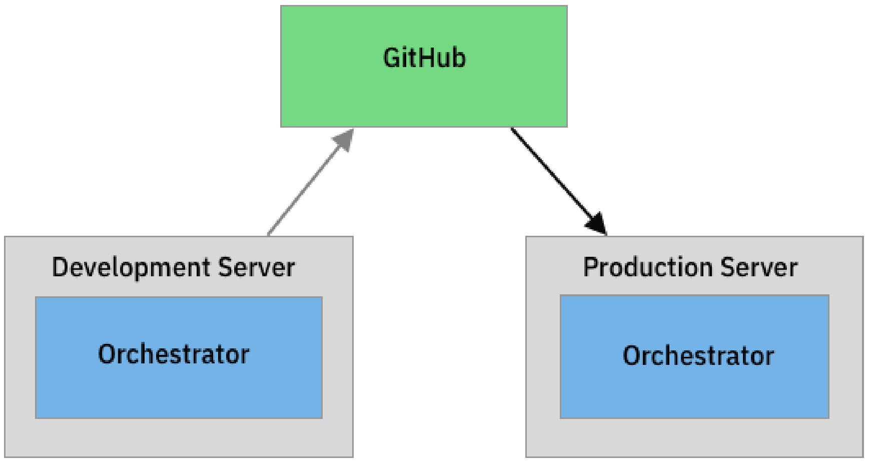
   Note: The same workflow can be started from **Work Orders** instead. See "Creating a Work Order" on page 113 for the alternate procedure.

Publishing Workflow Updates from Github

You can now configure your Orchestrator instance to work with Orchestrator's GitHub integration, so that when you update a workflow, you can push your changes to Github from your development instance or pull updates from Github to your production instance.

The ability to pull the latest workflow updates from GitHub can benefit your use cases, including:

- Syncing multiple Orchestrator instances
- Data recovery



Creating a Configuration for GitHub

Note: You can only have one GitHub configuration at a time.

1. To create your configuration, click **Engine** in the top menu, then click **Git Configurations** in the left navigation menu.
2. If no configuration currently exists, click **New configuration**.
3. Enter values for the fields below:

Field	Definition
Configuration Name	Custom name for the GitHub integration
Git Username	Username for your GitHub account
Git Password	Password for your GitHub account
Git Branch	Branch for pushing data to Github (for a development server) or pulling data from Github (for a production server)
Git Repository	Name of the project repository in GitHub
Is Production?	If true, this integration is pulling data from Github to a production server. If false, this integration is pushing data to a development server.

Synchronizing Your Development Integration with an Existing Workflow

Synchronizing your GitHub integration with an existing workflow allows you to publish that workflow to Github.

Note: The configured git repository must have at least one commit for you to be able to push to GitHub from the development server. If the commit history is empty, you get a `409 Conflict` error.

1. Click **Workflows** in the top menu.
2. To the right of the workflow name, click the **Push to git** icon (⤓).
3. In the **Are you sure you want to push updates for 'CustomPython1' to Git?** dialog, click **OK**.

Synchronizing Your Production Integration with an Existing Workflow

Synchronizing your GitHub integration with an existing workflow allows you to pulling updates to that workflow out of Github.

Note: The configured git repository must have at least one commit for you to be able to pull from GitHub to the development server. If the commit history is empty, you get a `409 Conflict` error.

1. Click **Workflows** in the top menu.
2. To the right of the workflow name, click the **Pull from git** icon (⤓).
3. In the **Are you sure you want to pull updates for 'CustomPython1' from Git?** dialog, click **OK**.

Updating an Existing Configuration

1. To edit your configuration, click **Engine** in the top menu, then click **Git Configurations** in the left navigation menu.
2. Click the name of the configuration, then click **Edit**. Edit the fields as needed, then click **Update**.

Deleting an Existing Configuration

1. To delete your configuration, click **Engine** in the top menu, then click **Git Configurations** in the left navigation menu.
2. Click the name of the configuration, then click **Destroy**.

Importing a Workflow

1. Click **Workflows**.
2. In the Workflow Definitions list, open the folder where the desired workflow is located.

3. Click **Import Workflow**.
4. Upload a workflow file.
   a) Click **Browse** and select a file from your local disk.
   b) Select a workflow from **Add as revision to** if you want the imported workflow to be added as the latest revision of an existing workflow.
   c) (Optional) Click **Link to Existing Sub-Workflows**.

      This option helps avoid the creation of duplicates while importing the workflows. If a sub-workflow that is used by multiple workflows or called multiple times in the same workflow has this option checked, the system will confirm whether the sub-workflow is already present in the repository. If the sub-workflow is already present, the system maps a new workflow to the existing sub-workflow; if the sub-workflow is not already present, it creates a single copy and maps any future instances to it.
   d) Click **Upload**.

   Note: If Journals are set up in the workflow being imported, then the JournalBook will be created and activated automatically (unless the Journal is present in the JournalBook already).

   The Edit Workflow Description screen appears, with a message confirming that the workflow was uploaded successfully.
5. Add a **Description** and **Comments** as needed, then click **Update Core Parameters**.

   The user is returned to the **Workflow Designs** view and a message, "Workflow was successfully updated", appears.

   In the screen view above, the imported workflow has been added as a revision of the `ffmpeg` workflow. Click **Latest** (next to the workflow name) to view the imported workflow.

Exporting a Workflow

This article describes the procedure for exporting a workflow from one Orchestrator node (for example, a development machine) in preparation for importing it to another Orchestrator node (for example, a production machine).

When these steps are completed, the exported workflow can imported with the procedure in "Importing a Workflow" on page 107.

1. Click **Workflows**.
2. In the Workflow Definitions list, locate the workflow you want to export and click the dropdown arrow next to the workflow name. Click **Export**.
3. The Export Workflow dialog opens, with one of the following conditions:
   - For workflows with dependencies (sub-workflows, remote nodes, and global templates), the dialog will list the dependencies. Click the **Export with dependencies** check box to automatically export the dependencies with the workflow.

     Note: This step (exporting the dependencies with the workflow) is recommended to ensure proper functioning of the workflow after it is imported by another Orchestrator machine.

     Click **Export**.
   - For workflows without dependencies (the dialog will state `Dependencies: none`), click **Export**.
4. In the dialog that opens, select the option to save the file to your local machine, then click **OK**.

   Note: A workflow with dependencies is saved as a `.wkf` file; a workflow without dependencies is saved as a `.yml` file.

   You can now follow the procedure in "Importing a Workflow" on page 107 to import the workflow into another machine.

Deploying a Workflow to an External Orchestrator Instance

Development and production machines can directly deploy a workflow from one Orchestrator instance to another, without having to first export from the source machine and then import into the destination machine.

You can deploy a workflow either in the UI or with the Orchestrator API. This section discusses deployment using the UI; for information on the API methods, see Export a Workflow and Import a Workflow.**Requirements:**

- Both Orchestrator instances must be v. 3.1.0 or newer.
- You must have a remote note that is configured with node type `Aspera Orchestrator`.
- The workflow must be in a "Published" state.

1. In the top menu, click **Workflows**.
2. In the list of workflows, click the **More** icon, then click **Deploy**.
3. From the **Target Orchestrator** dropdown, select the target Orchestrator node.
4. If you want the workflow to be published when it deploys, select **Publish workflow on Deploy**. If you do not select this option, it will remain in a "Draft" state.
5. Click **Deploy**.

   A confirmation message, "Successfully deployed workflow" appears.

Copying a Workflow

1. Click **Workflows**.
2. In the Workflow Definitions list, click the **More** icon to the right of the workflow you want to copy and click **Duplicate**.
3. In the dialog, update the information as needed and click **Create**.
   a) Enter a new workflow name.
   b) Select **Deep copy?** to create copies of templates used within the workflow. The templates become configurations in the duplicated workflow.
   c) In **Create in folder**, select the folder where the workflow will be placed.
   d) Select a role in **Run instances as**.
   e) Click **Create**.

Printing a Workflow

This procedure generates a graphical rendering of your workflow in an HTML file.

1. Click **Workflows**.
2. In the Workflow Definitions list, click the dropdown arrow next to the workflow you want to print and click **Generate printable documentation**.

   An HTML page is generated, containing the a graphical representation of the workflow, its input and outputs, configuration, comments, and revision history.
3. Print the HTML page using your browser print utility.

Creating a Workflow Folder

You can organize your workflows in folders.

1. Click **Workflows**.
2. From the current level, which is `root`, do one of the following:
   - Click **New Folder**.

    - Open a preexisting folder in the Workflow Definitions list (in which you want the new folder to be inserted), and click **New Folder**.
3. In the New Workflow Folder dialog, enter the folder name and click **Create**.

   The new folder is listed in alphabetical order under Workflow Definitions.

Changing the Folder Location for a New or Existing Workflow

By default, new workflows are created in the root folder, which automatically displays its contents in the Workflow Defnitions list

To begin, go to **Workflow > Workflow Designs**.

To move a workflow to a location other than the root folder, do one of the following:

- Create a new workflow in a specific folder.
  1. Open a preexisting folder in the Workflow Definitions list.
  2. Click **New Workflow**.
- Move an existing workflow to a specific folder.
  1. Click the dropdown arrow next to the workflow name and click **Move to folder**. The Move Workflow into a Folder dialog appears.
  2. In **Move to**, select a folder name and click **Move to folder**.

If you want to create a new folder to hold the workflow, rather than using a preexisting folder, see “Creating a Workflow Folder” on page 109.

Deleting a Workflow Folder

You can delete a workflow folder only, or the both the workflow folder and its contents.

1. Click **Workflows**.
2. In the Workflow Definitions list, click the dropdown arrow next to the folder you want to delete and click one of the following:
   - **Delete Folder**. This option moves any workflows inside the folder to the parent directory.
   - **Delete Folder with Contents**. This option also deletes any workflows inside the folder.

   A pop-up appears, asking you to confirm deletion.
3. Click **OK**.

Managing Workflow Revisions

1. Click **Workflows**.
2. In the Workflow Definitions list, click the dropdown arrow next to the desired workflow and click **View revision history**.
3. Select a specific workflow revision.
   a) Click the dropdown arrow next to the revision number.
   b) Click one of the following:

      Visualize
      Enables the user to view and edit the workflow revision. After publishing this revision, it will become the *published* (executable) version of the workflow.

      Publish/Unpublish
      Publishing a workflow revision makes that revision active. Unpublishing a workflow revision demotes it to `Draft` status; it does not impact current workflows until the user selects `Publish` again.

Edit XML
Enables the user to view and edit the XML code associated to the workflow revision.

Duplicate
Duplicates the workflow revision.

Delete
Deletes the workflow revision.

Setting Workflow Permissions

You can set workflow permisions for users and groups.

1. Click **Workflows**.
2. On the Workflow Design page, click the **More** menu icon (to the right of the workflow on which you need to set the permission), then click **Set permissions**.
3. Set group level permissions and user level permissions for the workflow.
   - In the Group Level Permissions area, click the following dropdowns menus to make a selection:
     - **Select Group**, to select a group role.
     - **Select Authorized Action**, to select an action (**all**, **view**, **edit**, or **run**. The default is `All`.

       Note: If you need to monitor the workflow in your user dashboard (for example, tracking workflow statistics and step statistics), select `run`.
     - **Allow Group**, to add the permission.
   - In the User Level Permissions area, click the following settings:
     - The **Select User** dropdown arrow, to select an **admin** or **system** user.
     - The **Select Authorized Action** dropdown arrow, to select an action (**all**, **view**, **edit**, or **run**. The default is `All`.

       Note: If you need to monitor the workflow in your user dashboard (for example, tracking workflow statistics and step statistics), select `run`.
     - **Allow User**, to add the permission.
4. To remove a permission for a user or group, click the **revoke** button next to that permission.

Setting Batch Permissions for Workflows

As described in the previous topic, "Setting Workflow Permissions" on page 111, workflow permissions can be set manually, for an individual workflow. However, it is possible to set batch permissions for workflows, at the folder level.

The permission on a folder can apply to entire user groups or individual users, and it applies to all content in that folder.

Note: This feature was introduced in Orchestrator 3.0.1, but the folder permissions can be applied retroactively to upgrades of product implementations prior to v. 3.0.1.

The following steps describe the process of setting permissions on a folder that contains workflows (including sub-folders).

1. Click **Workflows**.
2. On the Workflow Design page, click the **More** menu icon (to the right of the folder on which you need to set permissions) and click **Set permissions**.
3. In the Set Permissions dialog, set the desired permissions on the folder.

   There are two sections where you can edit folder permissions:
   - Group Level Permissions allow you to add a permission to an entire group. Select a group, select the access level, and click **Allow Group**. Click the **revoke** button to remove an existing permission.

- User Level Permissions allow you to add access for an individual user. Select a user, select the access level, and click **Allow User**. Click **revoke** to remove an existing permission.

Below are descriptions of the three permission types: `view`, `edit`, `run`, and `all`.

Folder Permissions

Permission Type	Description / Scope
view	Cam view top level of folder contents only; can't view sub-folders. Can't add or edit content.
edit	Can add and update folder contents and can view all workflows in the folder.
run	Allows the user or group to execute the workflow. This option allows you to monitor workflow statistics in your user dashboard.
all	Can view all levels of content in the folder (including contents of subfolders), and has full edit and run permissions. A user that creates a folder automatically has `all` permssion on that folder; members of the group `Developer` also have this permission.

4. Close the dialog to return to the Workflow Design page.

Generating a Workflow Call URL

This procedure creates a URL you can use to call the workflow within an **ascp** transfer.

1. Click **Workflows**.
2. In the Workflow Definitions list, click the dropdown arrow next to the desired workflow and click **Get Aspera call URL**.

   A pop-up containing a generated URL appears.

   Use the URL to call the workflow within an **ascp** transfer. For example, antivirus workflow logic can be called inline for transfer cancellation during a FASP transfer if a virus is detected.

Creating Test Cases with the Workflow Testing Utility

The workflow testing utility allows you to create multiple testing scenarios for each step in a workflow that simulate inputs and show workflow progress; you can include pre-processing and post-processing.

One use case for this feature is a situation in which a user creates a workflow, and then one or more additional users need to edit the workflow. To ensure that all subsequent users are aware of all possible scenarios for a step, the original user creates one or more tests for each step that can be verified after every change to the workflow; this ensures that the workflow is functioning as expected.

1. Click **Workflows**.
2. On the Workflow Design page, open a workflow by clicking the workflow name.
3. Right-click the step for which you want to create a test, then click **Test Step**.
4. Click **New Test Case**.
5. Edit the fields in the configuration dialog, then click **Save**.

   The new test case displays in the list of test case for that step.
6. Click **Run** to run all test cases in the list.

   The page displays the status and result for each test case.

Creating and Comparing Workflow Snapshots

The workflow snapshot feature allows you to record the state of a workflow before you make additional edits. You can then compare an earlier snapshot to a later one to see what changes you and other users have made over time.

Creating Workflow Snapshots

In this first procedure, you create workflow snapshots.

1. Click **Workflows** in the top menu.
2. Select the checkbox next to one or more workflows.
3. In the upper blue navigation menu, click the **Create Workflow Snapshot** icon.
4. Enter the name and optional comments, then click **Create Revision**.
5. After making changes to the workflow (or workflows, if you selected more than one), select the same one or more workflows, and again click the **Create Workflow Snapshot** icon.

   Repeat this step as needed to create additional snapshots.

Comparing Workflow Snapshots

In this procedure, you compare the workflow snapshots that were created above.

1. Click **Workflows**.
2. In the left navigation menu, click **Workflow Snapshots**.
3. Select two snapshots in the list, and click **Compare**.
4. An image of the workflow appears, with highlights indicating any changes to the workflow from one snapshot to the other. Below the image is a text description of the changes.

   If the snapshot includes more than one workflow, click the dropdown to select a particular workflow and view the comparison of snapshot versions for it.
5. Click **Workflows** in the top menu.

Viewing a Workflow as a Dependency Tree

The dependency tree feature gives you a comprehensive overview of a workflow and its dependencies.

1. In the top menu, click **Workflows**.
2. Locate the desired workflow in the list and click the **More** icon, then click **Dependency tree**.

   A dependency tree display opens. Below the workflow name are expandable lists of steps in the workflow (with their associated plugins), subworkflows, and remote nodes.
3. (Optional) To open the workflow in the designer, click the workflow name.

Work Orders

This section creates procedures for working with work orders.

Creating a Work Order

This procedure creates a work order, which is a run-time instance of a workflow.

1. Click **Work Orders > New Work Order**.
2. In the Create a Work Order dialog, **Search** or **Browse** to select a particular workflow.
3. Enter run-time parameter values, if required.
4. Open **Advanced Work Order Operations** and update the information as needed.

- Enter a description and comments.
- Select an execution priority.
- Add comma-separated tags.

5. To begin execution of the steps of the workflow, click **Start**.

Copying Work Step Inputs and Outputs

1. Click **Work Orders**.
   The **View by Work Order** tab is highlighted by default.
2. Click a work order in the list in the left pane.
3. In the list of work order instances that appears in the right pane, click a work order instance.
   A list of work steps appears.
4. Click a work step to open a display of inputs and outputs.
5. Copy the input or output.
   - Click the Copy icon ( ). The desired text is selected and copied to the clipboard. The confirmation message `Copied!` displays for 2 seconds.
   - To copy a variable with more than 1024 characters, click **More** (which appears next to its listed value). This opens a dialog displaying the entire variable value. Click the Copy icon ( ) to select and copy the variable to the clipboard.

     The confirmation message `Copied!` displays for 2 seconds.
6. Paste the input or output value by following the procedure in "Setting Pre-processing and Post-processing for Work Steps" on page 105.

Configuring Work Order Filters

This section describes the work order filtering options in Orchestrator. To access these options, click **Work Orders > Filter by**.

The following table describes the basic filtering options.

Filtering Option	**Description**
Work Order IDs	Enter a comma-separated list of the work order IDs.
Workflows	Click inside the box to see a list of options and then click your selection. Click additional selections as needed.
Word Order Status	Click inside the box to see a list of options and then click your selection. Click additional selections as needed.
Title	The name given to the workflow execution.
Tags	Tags (`QC workflows`, `Virus_scan`, and so on) are added to a workflow to increase the effectiveness of search. Enter multiple tags as a comma-separated list.
Max Entries	Sets the maximum results displayed on each page.

Click the "+" symbol to open the Advanced options.

Filtering Option	**Description**
Running As	A `system` user has access to the Engine; an `admin` user has full system access.

Filtering Option	Description
Priority	The available options are `low`, `medium`, `high`, and `preemptive`. The execution of a work order with `preemptive` priority will be interrupted when a work order with a higher priority setting is launched.
Key Word	Key word content to facilitate search.

Deleting a Work Order

1. Click **Work Orders**.

   In the left pane, a list of work orders appears.
2. Click the name of the work order you want to delete.

   The work order details appear in the pane to the right.
3. Click **Destroy**; in the confirmation pop-up, click **OK**.

Bulk Restart for Work Orders

The bulk restart option appears when you have two or more workflows. It allows you to restart all work orders simultaneously.

Note:

- Bulk restart will not respect the `Throttle` parameter that is set in the workflow configuration. See "Editing the Workflow Core Parameters" in "Editing a Workflow" on page 103 for more information.

1. Click **Work Orders > View by Workflow**.
2. In the left pane, click the name of a workflow in the list.

   Note: Bulk restart restarts all available work orders. If you intend to use bulk restart on a specific set of work orders, click **filter** and enter values for the relevant fields.
3. In the right pane, click **Bulk Restart**.

   Note: If the **Bulk Restart** button doesn't appear, there may be work orders that are currently running.
4. Select an option for **Work Order Steps** and do one of the following:
   - Click **WorkOrder** and for **Work Order Status**, select a value. This action selects all the work orders with the work order status indicated and restarts the workflow from the work step indicated.
   - Click **WorkStep** and for **Work Step Status**, select a value. This action selects all the work orders with the work step status indicated and restarts the workflow from the work step indicated.
5. Click **Bulk Restart**.

   A message, such as the following, appears:

   ```
   Warning: restarting 5 work order(s). Are you sure you want to continue?
   ```

   Click **OK**.

   A confirmation pop-up appears.
6. Click **OK**.

Journals

This section describes working with journals.

Overview: Journals and Journal Books

In Orchestrator, a *journal* is a log that collect run-time data about where a file can be found in the workflow. A *journal book* clusters all journal entries into categories.

For example, if an Orchestrator system is handling files from two different media companies, the user can create two different journal books, called (for example) `Company ABC` and `Company XYZ`, to track the entries in the journals. A journal book might also capture all the ingest steps across multiple workflows.

Note: During an upgrade of Orchestrator, the system automatically creates a journal book called `Misc Migrated Journals`; all journals created in the previous version of Orchestrator are migrated into this journal book.

Working with Journal Books

This article describes how to create and manage journal books, which cluster all journal entries into categories.

To create and configure journal books for workflows, click **Workflows > Journals**; this opens the Journals page. The **Journal Book** tab should be selected by default (the **Journaler** tab is for a deprecated feature).

See "Overview: Journals and Journal Books" on page 116 for more information about journal books.

Creating Journal Books

1. Under the **Journal Book** tab, click the **New** button.
2. Enter a name and a description (optional) for the journal book; click active if you want the journal book to start collecting data.

The new journal book now appears in the list under the **Journal Book** tab.

Managing Journal Books

The list of journal books under the **Journal Book** tab includes the name and the description (if entered) of the journal book, and whether a journal book is the primary one.

There are various actions you can take in the UI:

- Edit the name or description of a journal book, or change its status to Active = True or Active = False: Click the pencil icon to the right of the journal book name.
- **Delete all journals in a journal book (but not the journal book itself), along with its associated state information**: Click **Clean up Book Entries**.

  Note: If you intend to delete an entire journal book, Aspera recommends performing this action first; if you delete a journal book without first deleting the journal files it contains, those journal files become orphaned files.

  Important: Confirm that you really want to delete all information for this journal book before clicking this option, because the operation cannot be reversed.
- **Delete a journal book**: Click the trash can icon to the right of the journal book name.

  Note: Aspera recommends first deleting the journals in a journal book (see above) before deleting the journal itself.

Working with Journals

This article describes how to create and use journals, which are logs that collect run-time data about where a file can be found in the workflow

Creating and Configuring Journals

There are two methods for configuring a journal in a workflow:

- Configure the journal inside the configuration for any step in the workflow.
- Add the File Journal Log Entry plugin as a step in the workflow.

Configuring a Journal in Any Step in the Workflow

1. Click **Workflows** and open a workflow in the list.
2. Right-click the step where you want to create the journal, then click **Edit Action Template**.
3. Click the **Journal** icon.
4. Select **Journal on?**
5. Update the fields in the configuration dialog. When done, click **Save**.

Journal Configuration Options

Field	Value
Journal on?	Select this option to set a journal on the step.
New file?	When you begin journaling on a file, you must select this option to map it as a new file.
Journal book	(Only available after you select the "New file?" option.) A journal book holds multiple journals—often organized by category or common property.
File	From the dropdown, select the file to be journaled. This file represents the inputs, outputs, or attributes of the step where the journal is set. If you want to enter the value manually or using ruby code, select "Value or Composite Mapping" and enter the value.
New package?	Select this option for a new package.
Package	Groups journals into useful categories. (For example, if you want to view the files under `/tmp/server1` and `/tmp/server2` separately, map the package to these directory paths and then select `View by Package`).
File path	The path of the file being journaled.
File status	The status of the file being tracked by this journal.
Event	This field is used to manually store an event or the current status of the file. (For example: enter "Fileinprogress for Anti-Virus" or "File transferred to Server 1").
Journal Milestone	Enter the milestone value for the step associated with this journal. See the next section for more details about setting milestones.

Adding the File Journal Log Entry Plugin as a Step in the Workflow

1. Click **Workflows**.
2. Open a new or existing workflow.
3. In the Plugin Manager (left pane), enter `Journal` in the search box and drag the File Journal Log Entry plugin to the desired position in the workflow.

Note: If the plugin does not not appear in the Plugin Manager, you may need to enable the plugin. See "Enabling Plugins" on page 89.

4. Right-click the step and update the configuration options.

Journal Configuration Options

Field	Value
Journal book	(Only available after you select the "New file?" option.) A journal book holds multiple journals—often organized by category or common property.
New file?	When you begin journaling on a file, you must select this option to map it as a new file.
File	From the dropdown, select the file to be journaled. This file represents the inputs, outputs, or attributes of the step where the journal is set. If you want to enter the value manually or using ruby code, select "Value or Composite Mapping" and enter the value.
File path	The path of the file being journaled
Event	This field is used to manually store an event or the current status of the file. (For example: enter "Fileinprogress for Anti-Virus" or "File transferred to Server `1`").
New package?	Select this option for a new package.
Package	Groups journals into useful categories. (For example, if you want to view the files under `/tmp/server1` and `/tmp/server2` separately, map the package to these directory paths and then select `View by Package`).
Step name	Optional custom name for this step in the workflow
Journal status	The current status of the journal step
Journal Milestone	Enter the milestone value for the step associated with this journal. See the next section for more details about setting milestones.
Processing timeout for file	The length of the time the file has to be monitored

Setting Milestones in Journals

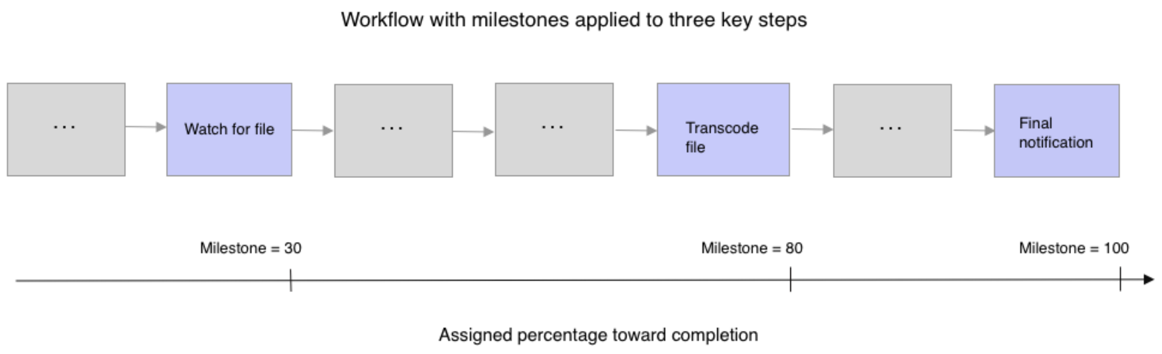
A milestone is a numeric value from `1` to `100` that you assign to a step in a workflow. The value of the milestone helps you estimate the amount of time it takes to complete a particular step or sequence of steps, and track the progress toward completion for the workflow.

You can apply a milestone to just one or two steps, or to every step in the workflow, according to your needs. Each milestone represents the percent completion for the workflow after the step completes. A milestone applied to the final step in the workflow will always be `100`, representing 100% completion of the workflow.

In the example workflow below:

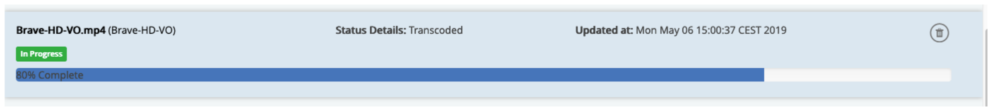
- The second step, "Watch for file"—which triggers on a particular file—has a milestone of `30`. When this step completes, the workflow is 30% complete.

- The fifth step, "Transcode file"—which initiates a transcode process on a file—has a milestone of `80`. When this step completes, the workflow is 80% complete.
- The final step, "Final notification"—which sends an email to a stakeholder—has a milestone of `100`. When this step completes, the workflow is 100% complete.



To set the milestone for a step, open the **Journal** tab in the action template. In the **Journal Milestone** field, enter the milestone value, then click **Save**.

When you publish your workflow and it begins to generate work orders, the Work Orders page applies a milestone to the progress bar (blue progress bar below). For example, this work order is 80% complete, indicating that a step with a milestone of `80` has completed successfully:


Brave-HD-VO.mp4 (Brave-HD-VO) **Status Details:** Transcoded **Updated at:** Mon May 06 15:00:37 CEST 2019

In Progress

80% Complete

Viewing Journal Status

Click **Work Orders > Journals** to open the Journals page. On this page, select a journal book from the dropdown at the top of the page to display the journal entries associated with that journal book. The Journal Filter pane allows you to **View by Files** or **View by Package**, and to set filtering criteria for viewing current journals.

The right pane of the page displays a list of journals as determined by the selections in the Journal Filter Pane, including progress bars for journal status.

Click the file name in the list of journals to view detailed information about the journal entry.

Deleting Journals

There are two methods for deleting journals:

- Delete a work order that contains a journal. If the Work-order that created the journal entry is deleted, the journal entries corresponding to it are also deleted along with it.
- Click **Work Orders > Journals**, then click **View by File** to delete a file individually, or **View by Package** to delete all journal entries corresponding to the selected journal package.

Managing Deletion of Journal Entries

The user permission, `Aspera:Orchestrator:JournalEntries:Edit`—which can be accessed on the **Users** page—allows admins to control which users and groups have access to the **Delete** button for journal entries. For more information on how to add or remove permissions, see "Viewing and Granting Permissions" in the Admin Guide.

Resource Manager

Creating a New Resource Pool

This procedure creates a resource pool, which allows you to track resource usage data across workflows.

1. In the top menu, click **Workflows**.
2. In the left navigation menu, click **Resources**.
3. Click the **New Pool** button.
4. In the Create New Resource Pool dialog, enter information in the data fields.

   For a custom portlet, enter a comma-separated list of resources.
5. Click **Save**.

   The new resource pool appears in the list on the Resource Manager page.
6. To see a dashboard of resources for a particular resource pool, click the icon for that resource pool.

   In the list of resources, you can edit the capacity of a resource, change the resource status from "online" to "offline" (and vice versa), or delete the resource. You can also add a resource to the list.

Viewing Resource History for Workflows

You can review the history of resources usage to help you optimize your workflows.

Prerequisite: You must already have configured at least one resource pool to use this feature. For more information, see "Creating a New Resource Pool" on page 120.

1. Click **Workflows** in the top menu, then click **Resources** in the left navigation menu.

   The Resource Manager page opens.
2. To see a dashboard of resources for a particular resource pool, click the icon for that resource pool.

   In the list of resources, you can edit the capacity of a resource, change the resource status from "online" to "offline" (and vice versa), or delete the resource. You can also add a resource to the list.
3. To view the allocation history for the resource pool, click the **Allocation history** icon at the top-right. **Note:** You must already have used the selected resource pool to view the allocation history; otherwise, there is no available data.

   You can view the allocation history of a resource for the past 30 days; as range of days, divided by day; or hourly, for a particular date.

   Note: You can't open the hourly view for a day with 100% usage, because each hour of that day is at 100%, as well (so there is nothing to be learned from viewing that level of detail).

Monitors

Creating a Workflow Monitor

A workflow monitor allows you to track the successful progress of work orders generated by a workflow.

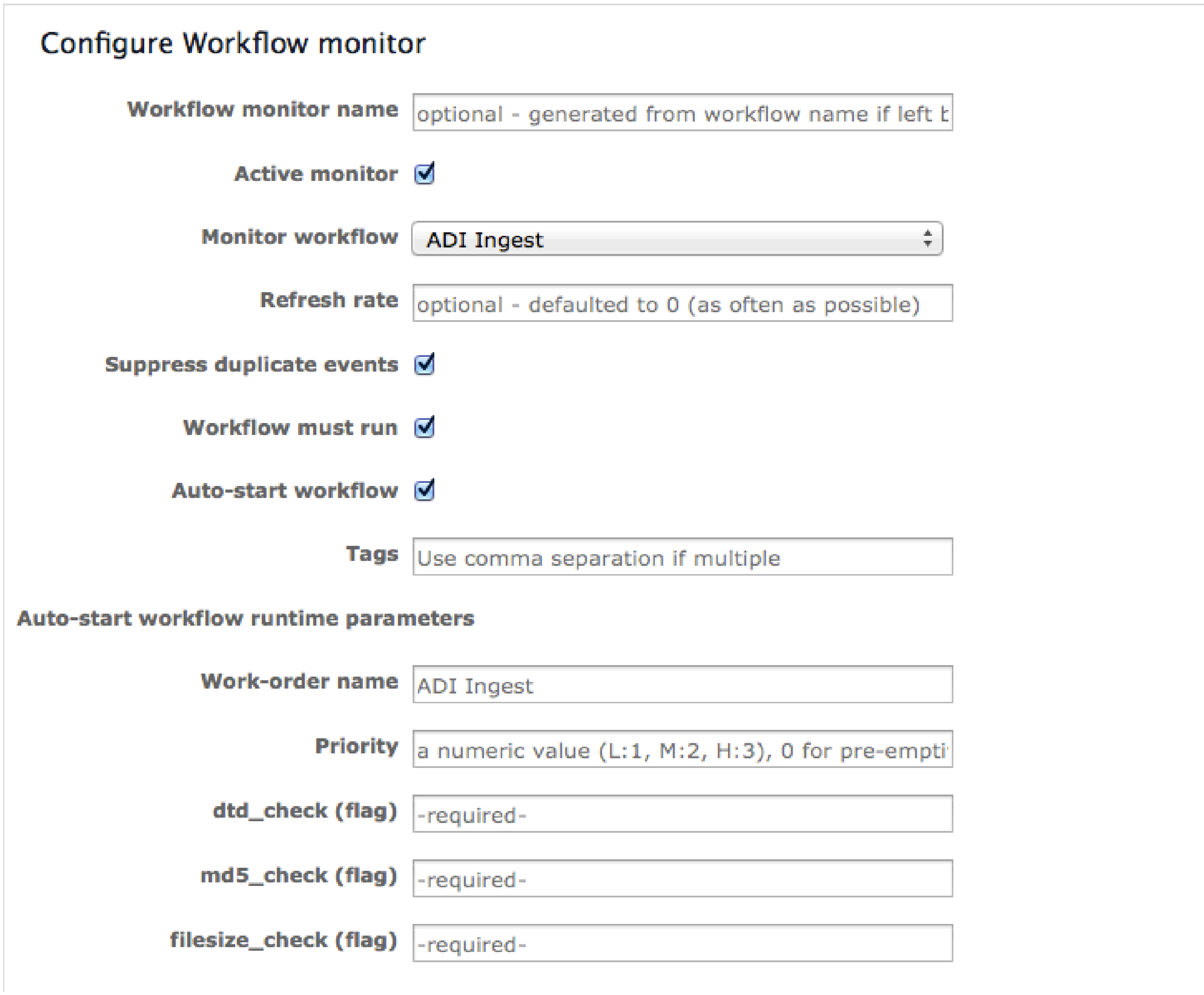
1. Click **Monitor**, then click **Configure workflows monitor**.
2. Click **New workflow monitor**.

   The Configure Workflow Monitor screen opens.
3. Edit the configuration fields as needed. The available options are described in the table below.

Option	Description
Active Monitor	The Monitor process will add the new monitor to its list of workflows to monitor.
Suppress duplicate events	If a monitor state doesn't change for a period of time, this flag stops a notification from being generated at every check.
Workflow must run	When checked, the monitor overview changes to red if no work order is found in a running state.
Auto-start workflow	When checked, the monitor starts a new work order if no work order is found in a running state.
Tags	Comma-separated entries can be added to tags to search and differentiate work orders started by monitors.

Run-time parameters can be provided to the work order via the monitor.

The following example show a monitor that has been created for the `ADI Ingest` workflow. The monitor is active, it must run, and if a work order stops, it will be started automatically.



Configure Workflow monitor

Workflow monitor name	optional - generated from workflow name if left b
Active monitor	✓
Monitor workflow	ADI Ingest
Refresh rate	optional - defaulted to 0 (as often as possible)
Suppress duplicate events	✓
Workflow must run	✓
Auto-start workflow	✓
Tags	Use comma separation if multiple

Auto-start workflow runtime parameters

Work-order name	ADI Ingest
Priority	a numeric value (L:1, M:2, H:3), 0 for pre-empti
dtd_check (flag)	-required-
md5_check (flag)	-required-
filesize_check (flag)	-required-

Bulk Activation and Deactivation for Workflow Monitors

With bulk activation and deactivation for workflow monitors, all workflows can be activated and deactivated simultaneously.

This feature helps the user avoid the inconvenience of deactivating each monitor individually while performing maintenance and taking snapshots.

Activate all activates all the workflow monitors that are not running or inactive. **Deactivate all** deactivates (stoprs) all tworkflow monitors that are running.

These options can be accessed from the monitor dashboard screen (**Monitor > Workflows**) and from the workflow monitor configuration screen (**Monitor > Workflows**, then click **Configure workflows monitor**).

Creating a Folder Monitor

A folder monitor tracks the status of folders associated with a workflow.

1. Click **Monitor**, then click **Configure folders monitor**.

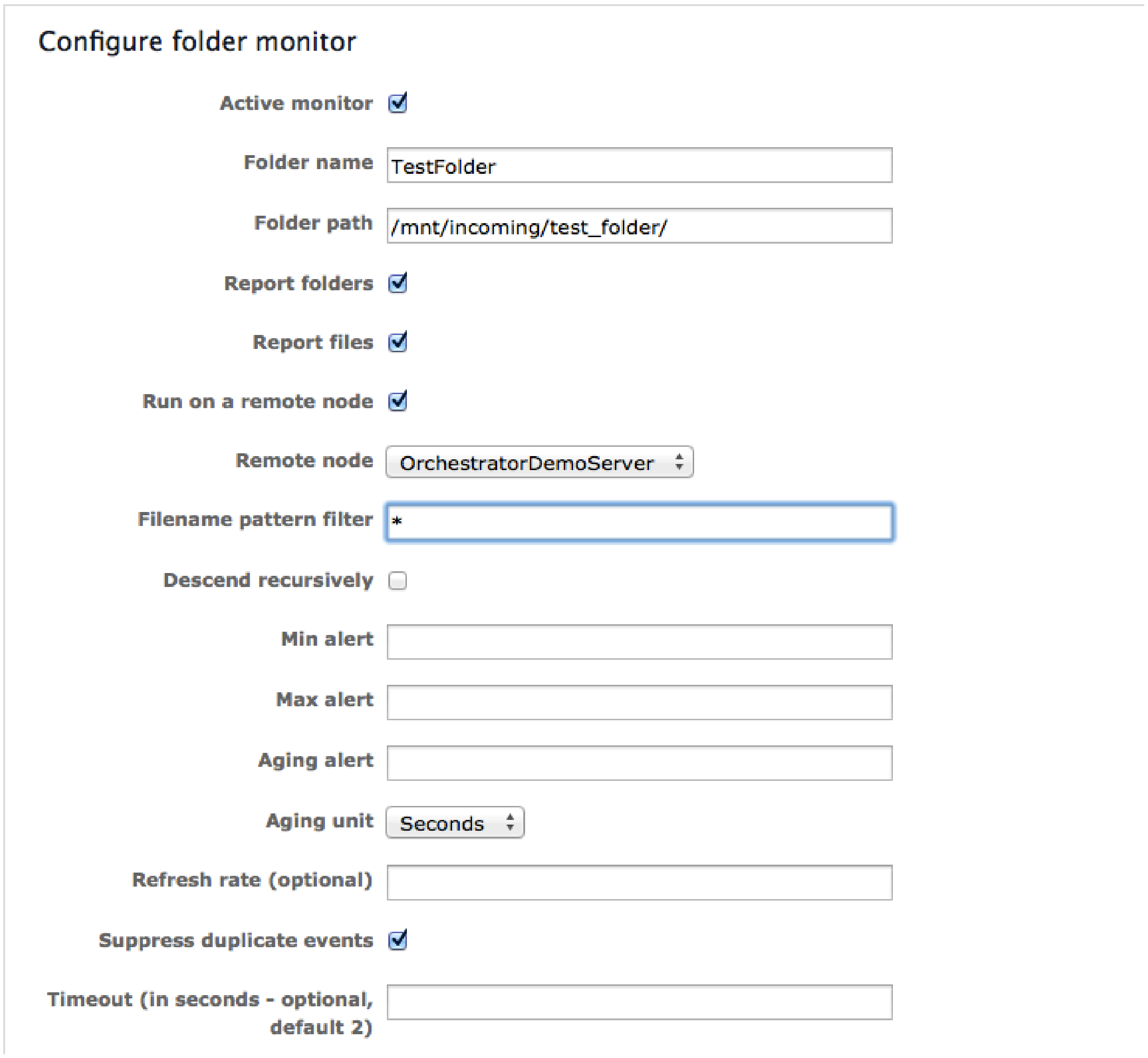
   A new screen opens.
2. Click **New folder monitor**.

   The **Configure folder monitor** screen opens.
3. Edit the configuration fields as needed, then click **Create**.

   Configuration options are described in the following table.

Option	**Description**
Active Monitor	The monitor process will add the new folder monitor to its list of folders to monitor.
Report folders	Stats reported include folders inside the folder.
Report files	Stats reported include files inside the folder.
Run on a remote node	The folder monitor starts an SSH tunnel to the remote node and obtains information about the folder. **Note:** Do not select this option if the folder is local to Orchestrator.
Descend recursively	The stats about the folder will include subfolders and their associated files.
Filename pattern filter	A glob pattern to include stats about certain types of files. For example, `*.mov` will only report files with a `.mov` extension.

The following example shows the configuration of a folder monitor for a folder located at `/mnt/incoming/test_folder/`.


Configure folder monitor

Active monitor	✓
Folder name	TestFolder
Folder path	/mnt/incoming/test_folder/
Report folders	✓
Report files	✓
Run on a remote node	✓
Remote node	OrchestratorDemoServer
Filename pattern filter	*
Descend recursively	☐
Min alert	
Max alert	
Aging alert	
Aging unit	Seconds
Refresh rate (optional)	
Suppress duplicate events	✓
Timeout (in seconds - optional, default 2)	

Creating a Script Monitor

This procedure creates a script monitor, which manages scripts in Orchestrator.

1. Click **Monitor**, then click **Configure scripts monitor**.

   The Scripts screen opens.
2. Click **New script monitor**.

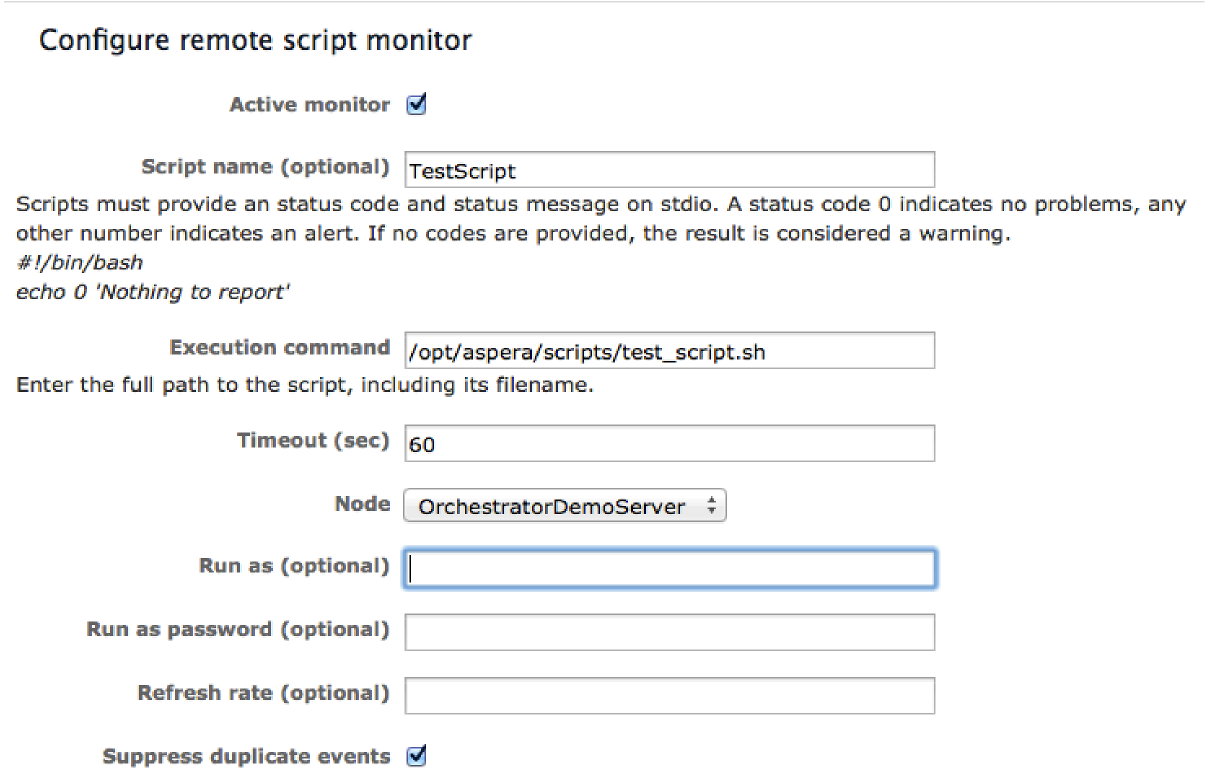
   The **Configure remote script monitor** screen opens.
3. Edit the configuration fields as needed, then click **Create**.

   Configuration options are described in the following table.

Option	Description
Timeout	Time in seconds to abort running the script if it does not respond with a valid exit code.
Run as	Stats reported include folders inside the folder.
Run as password	Password of the above user.

Option	Description
Refresh rate	Frequency with which the script stats should be updated in the Overview screen.
Suppress duplicate events	If a monitor state doesn't change for a period of time, this flag stops a notification from being generated at every check.

The following example shows the configuration of a script monitor for a script located at `/opt/aspera/scripts/test_script.sh` on the node OrchestratorDemoServer. If the script needs to run locally, create a localhost node.


Configure remote script monitor

Active monitor ✓

Script name (optional): TestScript

Scripts must provide an status code and status message on stdio. A status code 0 indicates no problems, any other number indicates an alert. If no codes are provided, the result is considered a warning.

```
#!/bin/bash
echo 0 'Nothing to report'
```

Execution command: /opt/aspera/scripts/test_script.sh

Enter the full path to the script, including its filename.

Timeout (sec): 60

Node: OrchestratorDemoServer

Run as (optional):

Run as password (optional):

Refresh rate (optional):

Suppress duplicate events ✓

Creating a Monitor Group

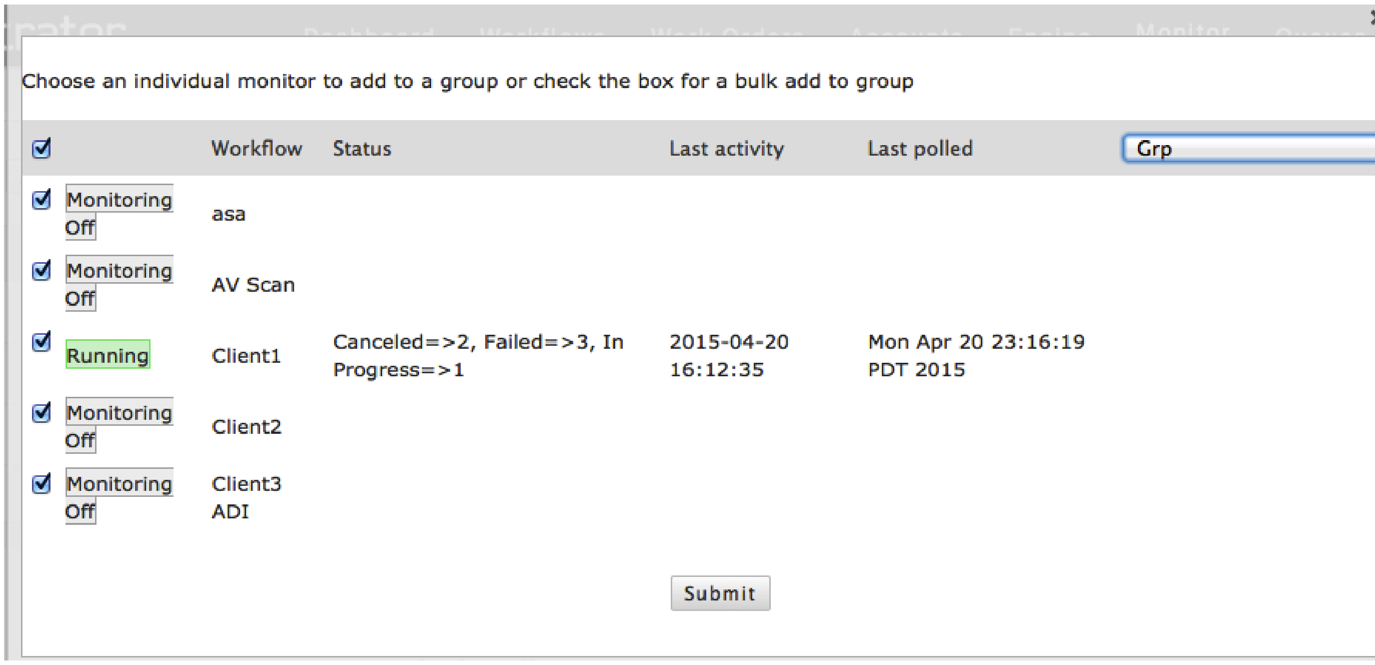
A monitor group consolidates workflow monitors so that bulk operations can be performed on all monitors simultaneously.

1. Open the Monitor Groups page (**Monitor > Monitor Groups**), then click **Define New Monitor Group**.
   The New Monitor Group dialog opens.
2. Enter **Monitor group name** and optional **Comments**, then click **Create**.
   The new monitor group name and details are now displayed in the Monitor Groups list.

Adding Monitors to a Monitor Group

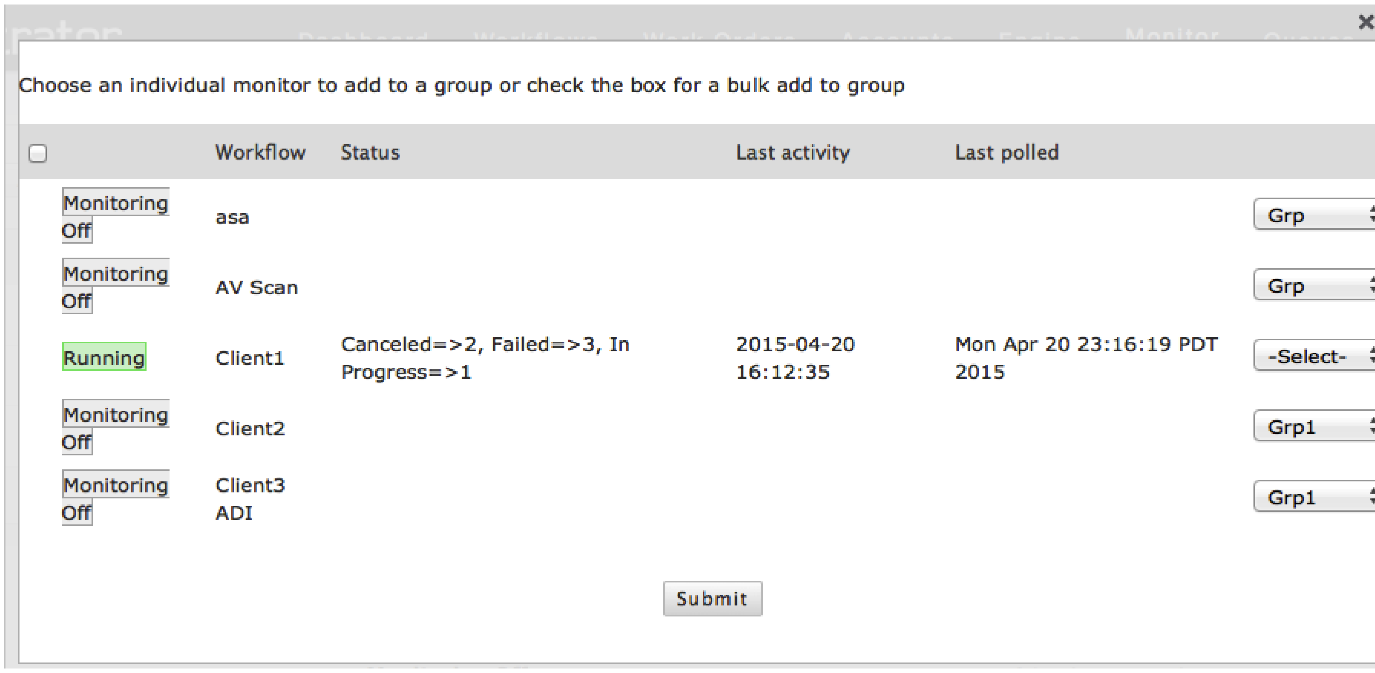
1. Open the Monitor Groups page (**Monitor > Monitor Groups**), then click **Add Monitor Workflow to Monitor Group**.
2. In the dialog, make your selections to assign a monitor to a monitor group.

   All workflow monitors can be added (as a "bulk add") to a single monitor group; this is accomplished by selecting the checkbox in the gray bar. In the dialog example below, all workflow monitors have been added to the Monitor Group `Grp`.


Choose an individual monitor to add to a group or check the box for a bulk add to group

☑	Workflow	Status	Last activity	Last polled	Grp
☑ Monitoring Off	asa				
☑ Monitoring Off	AV Scan				
☑ Running	Client1	Canceled=>2, Failed=>3, In Progress=>1	2015-04-20 16:12:35	Mon Apr 20 23:16:19 PDT 2015	
☑ Monitoring Off	Client2				
☑ Monitoring Off	Client3 ADI				

Alternatively, each workflow monitor can be added to a different monitor group. In the dialog example below, some workflow monitors are added to `Grp` and some are added to `Grp1`.


Choose an individual monitor to add to a group or check the box for a bulk add to group

☐	Workflow	Status	Last activity	Last polled	
Monitoring Off	asa				Grp
Monitoring Off	AV Scan				Grp
Running	Client1	Canceled=>2, Failed=>3, In Progress=>1	2015-04-20 16:12:35	Mon Apr 20 23:16:19 PDT 2015	-Select-
Monitoring Off	Client2				Grp1
Monitoring Off	Client3 ADI				Grp1

3. Click **Submit** to confirm your selections.

Creating a Remote Node Monitor

Using a remote node monitor allows the user to perform connectivity tests (to SSH, FASP, TCP or Aspera Orchestrator) for a particular remote node, on one or more selected ports.

1. Click **Monitor**, then click **Configure nodes monitor**.
2. Click **New node monitor**.
3. In the Configure Node Monitor screen, enter the necessary information.

   The following table describes each of the information fields in the Configure Node Monitor screen.

Option	Description
Active monitor	If this check box is checked, the node monitor is created in active mode; this means that the monitor of the node polls for the status of the nodes when the monitor is active.
Node monitor name	The name of the new monitor node.
Remote node	The remote node on which the monitor is created. The dropdown list of available options is derived from all the remote nodes configured in Orchestrator.
Port	This field displays all the ports that were configured while configuring the remote node. For each port, the user can set the notification type (`Alerts`, `Warnings`, or `Nothing`) and the test type (FASP, SSH, TCP, and `Aspera Orchestrator`). When `Aspera Orchestrator` is selected for the test type, one orchestrator node can know the process statuses (for example, the statuses of engine, mongrels, workers, and so on) of another orchestrator node. **Note:** If no ports are configured in the remote node configuration, the following ports are associated with the new remote node monitor by default: • Port 22 with the SSH, TCP and Orchestrator nodes. • Port 33001 with the FASP node.
Suppress duplicate events	If a monitor state doesn't change for a while, this flag stops a notification from being generated at every check.
Notification workflow	This option lists all the workflows present in Orchestrator. Select the one that will be triggered when the notification has to be sent. For example, an email notification workflow can be configured and assigned to the monitor. For all notifications, that workflow will be triggered with the status of the monitor.
Notification text	The text content indicates some information on the monitor. For example, a monitor name can be provided in the notification text. This text will be embedded in all the notifications that are sent to stakeholders.
Alert processing code	This is an optional input. When there are alerts on a particular node for the selected test, an alert processing code is executed, mostly in lieu of amending the situation that triggered the alert.
Warning processing code	This is an optional input. When there are warnings on a particular node for the selected test, a warning-processing code is executed, mostly in lieu of amending the situation that triggered the warning.

4. Click **Create** to confirm the information.

   The new remote node appears in the Node list, with a status code in green, yellow, or red to the left of the remote node name. See "Remote Node Status Notifications" on page 87 for an explanation of status codes and additional action options.

Exporting a Remote Node Monitor

1. Click **Monitor > Nodes**.
2. Click **Configure nodes monitor**.
3. Click **Export**.
4. In the pop-up, select **Save File**, then click **OK**.

   A remote node monitor file is created with the same name as the original file.

Importing a Remote Node Monitor

1. Click **Monitor > Nodes**.
2. Click **Configure nodes monitor**.
3. Click **Import Monitor Node**.
4. In the pop-up, select a monitor node file, then click **Upload**.
   The imported remote node monitor appears in the Node list.

Creating a Work Order Monitor

Work Order Monitors are used to monitor a workflow instance and to perform an action on its `Completion`, `Failure`, or `Error`.

1. Click **Monitor > WorkOrders**.
2. Click **Configure workorders monitor**.
3. Click **New workorder monitor**.
4. In the Configure Workorder Monitor page, enter the information, then click **Configure**. The following table describes the various options that can be selected.

Option	**Description**
Active Monitor	The monitor process will add the new monitor to its list of workflows to monitor.
Trigger on new work orders	The monitor will only start triggering on new instances of work orders created after the monitor is activated.
Want the last executed step as an output	The work order monitor output is used as a hash parameter called `last_executed_steps` and storing the `step_id` and `step_name` as output.
On complete notification On failure notification On error notification	Specifies that a workflow is to be triggered on completion, failure, or error.

Exporting a Work Order Monitor

1. Click **Monitor > WorkOrders**.
2. Click **Configure workorders monitor**.
3. Click **Export**.
4. In the pop-up, select **Save File**, then click **OK**.
   A work order monitor file is created with the same name as the original file.

Importing a Work Order Monitor

1. Click **Monitor > WorkOrders**.
2. Click **Configure workorders monitor**.
3. Click **Import Monitor Workorder**.
4. In the pop-up, select a monitor node file, then click **Upload**.
   The imported work order monitor appears in the Workflow list.

Managing System Mounts

Mount manager allows the user to mount remote file systems directly from Orchestrator. This feature maintains records of remote file systems in the Orchestrator database so that the user can mount and unmount them.

Viewing System Mounts

To open the Mount Manager view, click **Engine > Mount Manager**.

- To view all mounts created with Mount Manager, look at the **Mounts by Mount Manager** pane.
- To view all mounts in the system, both those created with Mount Manager and those created from the command line, look at the **All Mounts in the System** pane.

Creating New System Mounts

1. To create a new mount, click **Mount Manager > Add mount**.
2. In the **Create New Mount** dialog, enter the parameters for the new mount, then click **Save**.

   The new mount appears in the **Mounts by Mount Manager** pane; the status of the mount is displayed as one of the following:

   - `Unmounted`
   - `Mounted`
   - `Waiting` (in progress)

Mounting and Unmounting a Directory

To mount a directory, click the dropdown arrow to the left of the **Mounted To** column and click **Mount**.

To unmount a directory, click **Unmount**. Once the directory is unmounted, you have the option to edit the parameters of the mount or delete the mount.

Editing and Deleting System Mounts

Note: The edit and delete functions are only available when a directory is unmounted.

To edit the parameters of a mount, click the dropdown arrow to the left of the **Mounted To** column and click **Edit**.

To delete a mount, click the dropdown arrow and click **Delete**.

Portal Pages and Portlets

Overview of Portal Pages and Portlets

Portlets collect data across the Orchestrator application, connect with applications external to Orchestrator, and provide start functionality for work orders. There are default and custom portlets. *Portal pages* provide a seamless way for you to view and interact with the portlet data and functionality.

Installing an Available Portlet

You can install a portlet that is in Other Available Portlets.

1. Click **Engine > Portlets**.

The Manage Portlets page appears. The page is divided into two sections: Installed Portlets and Other Available Portlets.

2. To install a portlet that is in the Other Available Portlets section, locate the desired portlet and click **Enable**.

   The portlet appears in the Installed Portlets section.

Modifying an Existing Portlet

Note: This procedure should be used only by users with knowledge of Ruby and Rails who have taken the Orchestrator Portlet training.

1. Open the following directory:

```
C:\Program Files (x86)\Aspera\Orchestrator\portlets\
/opt/aspera/orchestrator/portlets/
```

2. Find the portlet you need to modify and open that directory.
3. Select the relevant file from the following list:
   - *portlet_name*.rb (Ruby)
   - *portlet_name*.erb (Rails/HTML)

   Open the file in Notepad or another file editing tool.
4. Edit and save the file.
5. Click **Engine > Portlets**.
6. Locate the modified portlet; click **Disable** and then **Enable** to set your changes.

Creating a Custom Portlet

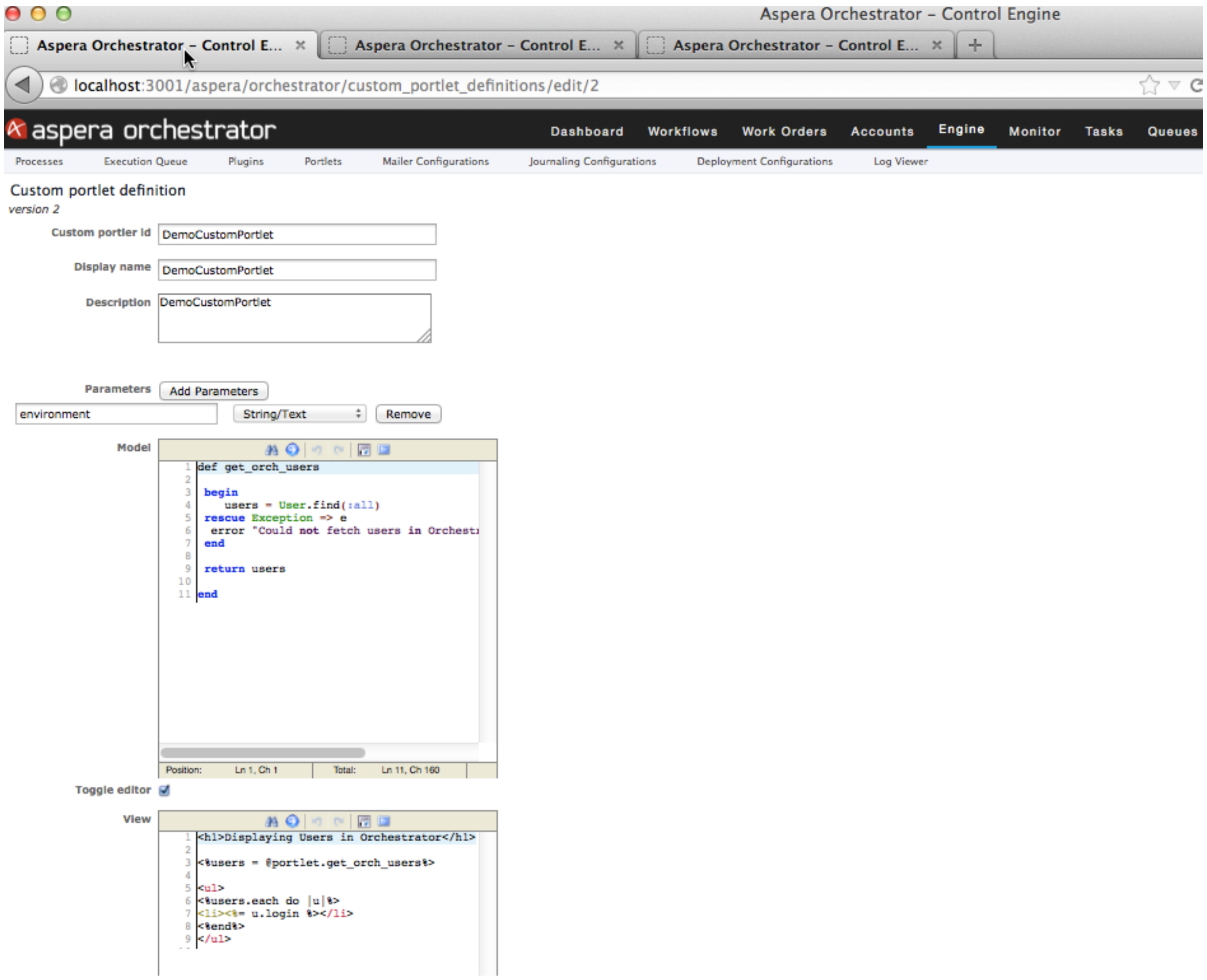
Custom portlets let you tailor the functionality of a portlet.

1. Click **Engine > Portlets**.
2. If the portlet `Custom` does not appear in the Installed Portlets area, go to the Other Available Portlets area, and next to `Custom`, click **Enable**.

   The Manage Portlets page opens.
3. Click **Define Custom Portlet**.
4. In the Custom Portlet Definition page, enter the requested information.

   The **Model** field is used to enter the methods that your **View** field (HTML page or portal page) requires to render correctly.

   These code examples are for the custom portlet definition of a sample portlet used to display Orchestrator users.


Aspera Orchestrator – Control Engine
Aspera Orchestrator – Control E...
Aspera Orchestrator – Control E...
Aspera Orchestrator – Control E...
localhost:3001/aspera/orchestrator/custom_portlet_definitions/edit/2
aspera orchestrator
Dashboard
Workflows
Work Orders
Accounts
Engine
Monitor
Tasks
Queues
Processes
Execution Queue
Plugins
Portlets
Mailer Configurations
Journaling Configurations
Deployment Configurations
Log Viewer
Add Parameters
Remove
Position: Ln 1, Ch 1
Total: Ln 11, Ch 160

Custom portlet definition
version 2

Custom portler Id: DemoCustomPortlet

Display name: DemoCustomPortlet

Description: DemoCustomPortlet

Parameters: environment String/Text

Model

```
def get_orch_users

 begin
    users = User.find(:all)
 rescue Exception => e
  error "Could not fetch users in Orchestı
 end

 return users

end
```

Toggle editor ✓

View

```
<h1>Displaying Users in Orchestrator</h1>

<%users = @portlet.get_orch_users%>

<ul>
<%users.each do |u|%>
<li><%= u.login %></li>
<%end%>
</ul>
```

Model field:

```
def get_orch_users
                begin
    users = User.find(:all , :select => "login")
 rescue Exception => e
    error "Could not fetch users in Orchestrator: #{e.inspect}"
    return []
 end
 return users
end
```

View field:

```
<h1>Displaying Users in Orchestrator</h1>
<%users = @portlet.get_orch_users%>
<ul>
<%users.each do |u|%>
<li><%= u.login %></li>
<%end%>
</ul>
```

5. Click **Save**.
6. Follow the procedure in "Creating a Portal Page" on page 130 to create a portal page with your custom portlet.

Creating a Portal Page

1. At the top-right corner of the Orchestrator UI, click the dropdown arrow next to **admin** and click **Preferences**.

The **Users** page opens.

2. Scroll to the bottom of the page, and in the Dashboards section, click **Define new dashboard**.
3. In the Define New Portal Page dialog, enter the **Name** and **Description**, then click **Create**.

   The new dashboard is displayed in the Dashboards section.

   You can configure your new portal page according to the procedure in "Configuring a Portal Page" on page 131. To reorder the list of portal pages, see "Reordering the List of Portal Pages in the Dashboards List" on page 132.

Configuring a Portal Page

After creating a portal page according to the procedure in "Creating a Portal Page" on page 130, you can configure it by selecting a portlet and customizing the screen allocationt, thus customizing how it will be displayed in the new dashboard.

1. Go to the Users page.

   At the top-right corner of the Orchestrator UI, click the dropdown arrow next to **admin** and click **Preferences**.

   The **Users** page opens.
2. Scroll to the Dashboards section at the bottom of the page.
3. Click the dropdown arrow to the left of the portal page you want to configure, and click **Configure**.

   The Configuration of Portal Page '*portal_page_name*' page opens.
4. Set a horizontal split or vertical split view.

   a) If you see see a dialog titled **Configure Portlet '*Portlet_Name*'**, click **Reset Frame**.

   b) Click **Split Horizontal** or **Split Vertical**.

      To undo, click **Reset horizontal split** or **Reset vertical split**.
5. To confirm your settings, click **Done Configuring**.

Selecting a Custom Portlet for a Portal Page

Note: You should already have followed the procedure in "Creating a Custom Portlet" on page 129.

1. Go to the Users page.

   At the top-right corner of the Orchestrator UI, click the dropdown arrow next to **admin** and click **Preferences**.

   The **Users** page opens.
2. Scroll to the Dashboards section at the bottom of the page.
3. Click the dropdown arrow to the left of the portal page you want to configure, and click **Configure**.

   The Configuration of Portal Page '*portal_page_name*' page opens.
4. In the Choose Portlet dropdown menu, select **Custom Portlet**, then click **Select**.
5. Applicable run-time parameters (in the example, environment) are displayed. Enter a value for the parameter and click **Save**.

   The portal page is now updated.
6. Click **Done Configuring** to return to the Users page.
7. Scroll down to the Dashboards section, click the dropdown next to the portal page, and click **Activate**.

   Click **Preview** in the same menu to view the portal page (Optional).
8. Click **Dashboard** in the top navigation menu to view the updated portal page.

Selecting an Installed Portlet for a Portal Page

An installed portlet allows you to get visibiliity to a selected set of data on your portal page.

Note: You must first follow the procedure in "Creating a Portal Page" on page 130 before following this procedure.

1. Go to the Users page.

   At the top-right corner of the Orchestrator UI, click the dropdown arrow next to **admin** and click **Preferences**.

   The **Users** page opens.
2. Scroll to the Dashboards area at the bottom of the page.
3. Click the dropdown arrow to the left of the portal page you want to configure, and click **Configure**.

   The Configuration of Portal Page '*portal_page_name*' page opens.
4. In the **Choose Portlet** dropdown menu, select an installed portlet, then click **Select**.
5. The workflow steps portlet has certain inputs that allow the portal page to be configurable. Select the necessary inputs, then click **Save**.
6. Click **Done Configuring** to return to the Users page.
7. Scroll down to the Dashboards section, click the dropdown next to the portal page, and click **Activate**.

   Click **Preview** in the same menu to view the portal page (Optional).
8. Click **Dashboard** in the top navigation menu to view the updated portal page.

Assigning a Portal Page to Other Users

After a portal page is created in Orchestrator by a user, it can be assigned to other users.

1. (Preliminary) Follow the procedure in "Configuring a Portal Page" on page 131, then click the dropdown menu next to the desired plugin and select **Activate**.
2. Next to the desired plugin, click the dropdown menu and select **Assign to user or group**.
3. In the Assign Dashboard dialog, select a user and group. click **Add user assignment**. Select a group and click **Add group assignment**.

   See "Modifying User Information and Assigning a User to a Group" on page 68 for an description of group roles.

Reordering the List of Portal Pages in the Dashboards List

In the Dashboards list (**admin > Presferences**) you can use the available control buttons to reorder the list of portal pages.

Click the directional buttons to the right of the portal page name to change the position of a portal page on the dashboard:

'<<'
: Moves the portal page to the top of the list.

">>"
: Moves the portal page to the end of the list.

"↑"
: Moves the portal page position up by one.

"↓"
: Moves the portal page position down by one.

Overview of Portal Pages and Portlets

Portlets collect data across the Orchestrator application, connect with applications external to Orchestrator, and provide start functionality for work orders. There are default and custom portlets. *Portal pages* provide a seamless way for you to view and interact with the portlet data and functionality.

Queues

Viewing and Managing Queues

Note: To be able to view and manage queues, a user must first be added to the Scheduler group. See "Modifying User Information and Assigning a User to a Group" on page 68 for more information.

1. Click **Queues** in the top menu bar to open the Manage Queues page.

   The names of all queues are displayed in the left pane.
2. To view status information, click the name of the desired queue from the list in the left pane.

   50 queued items are displayed at a time. If the count of queued items exceed than 50, access the remaining displayed items by clicking **Next** or **Previous** at the bottom-right corner of the page

Execution Queue Display

The Queued Work Step page displays a summary of how Orchestrator processes are executing in the system.

Click **Engine > Execution Queue** to open the page.

This page provides options for you to **Purge non-active queued items**, **Purge all queued items**, and **Destroy all work orders**.

ASCTL and Orchestrator

Overview

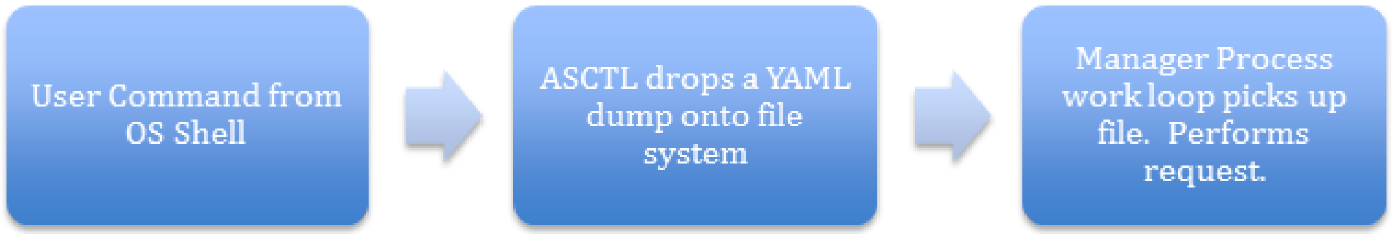
Aspera's **`asctl`** functionality helps users to control many important operations from the command line.

This functionality provides a single interface for all tasks, thus considerably increasing efficiency for Orchestrator processes.

Orchestrator provides a command line utility for certain operations like `start`, `stop`, and `check status` using the `orchestrator` script. Other scripts exist to allow users to set up and install the product, back up data, and migrate data.

The topics in this section provide a high-level description of **`asctl`** methods and the implementation, inputs, and outputs needed to use **`asctl`** with Orchestrator.

Note: Loading Rails will cause slow execution of **`asctl`** commands. Therefore, all of the methods interacting with the Rails code dump a YAML file that contains instructions to assist the Orchestrator process in picking up and understanding the task. The following diagram illustrates the interaction between the operating system shell and Orchestrator in this scenario.


User Command from OS Shell
ASCTL drops a YAML dump onto file system
Manager Process work loop picks up file. Performs request.

Setup

This `asctl` method runs installation and setup of Orchestrator.

Method name	Setup
Command	`asctl orchestrator:setup`
Details	Runs installation and setup of Orchestrator.
Inputs	None (Interactive).
Outputs	Interactively ask questions to setup Orchestrator. Display user inputs and begin install.

```
# rpm -Uvh aspera-orchestrator-version-0.x86_64.rpm
Preparing...                          ########################################### [100%]
Updating / installing...
   1:aspera-orchestrator-version-########################################### [100%]
To complete the upgrade of Orchestrator, run "asctl orchestrator:setup"

Streamlined: A simple setup with all components running on this computer.
Detailed:    An advanced configuration (e.g. non-standard ports or not all components running
on a single server)

Streamlined or detailed setup (s/d)? (current: s) s

===================== Settings =====================
Orchestrator
  Enabled:               true
  Mongrel count:         3
  Mongrel base port:     3000
  Web root:              /aspera/orchestrator
  MySQL is local:        true

Are these settings correct? (y/n/x with x for exit) y
```

Start

This `asctl` method instantiates the Orchestrator processes: manager, engine monitor, workers, and mongrels.

Method name	Start
Command	`asctl orchestrator:start`

Details	Instantiates orchestrator processes.
Inputs	None.
Outputs	STDOUT to show that Orchestrator processes have been stopped.

```
# asctl orchestrator:start
starting...
started...
nohup: redirecting stderr to stdout

Orchestrator Status:
  -> Orchestrator Manager running with pid: 26783
  -> Orchestrator Engine running with pid: 26797
  -> Mongrel serving orchestrator on port 3000 is running with pid: 26802
  -> Mongrel serving orchestrator on port 3001 is running with pid: 26804
  -> Mongrel serving orchestrator on port 3002 is running with pid: 26806
  -> Orchestrator Monitor running with pid: 26800
  -> Asynchronous Worker Process 0 is running with pid: 26813
  -> Asynchronous Worker Process 1 is running with pid: 26815
  -> Synchronous Worker Process 2 is running with pid: 26817
  -> Synchronous Worker Process 3 is running with pid: 26819
  -> Synchronous Worker Process 4 is running with pid: 26821
  -> Synchronous Worker Process 5 is running with pid: 26811
```

Note: When you initially run **asctl orchestrator:start**, you may see, instead of the above Orchestrator Status, `Orchestrator Engine running with pid`, a different status:

```
 Orchestrator Engine NOT running
```

This displays if Orchestrator is still completing the installation process. To confirm that Orchestrator processes are running, wait a minute, and then run the following command:

```
asctl orchestrator:status
```

This will provide an updated status. See "Status" on page 135 for more information.

Status

This `asctl` method returns the current status for Orchestrator.

Method name	Status
Command	`asctl orchestrator:status`
Details	Displays a status: `running`, `not running`, `stopped`, `stop pending`, and so on.
Inputs	None.
Outputs	STDOUT similar to screenshot below; details will vary.

```
# asctl orchestrator:status

Orchestrator Status:
  -> Orchestrator Manager running with pid: 26783
  -> Orchestrator Engine running with pid: 26797
  -> Mongrel serving orchestrator on port 3000 is running with pid: 26802
  -> Mongrel serving orchestrator on port 3001 is running with pid: 26804
  -> Mongrel serving orchestrator on port 3002 is running with pid: 26806
  -> Orchestrator Monitor running with pid: 26800
  -> Asynchronous Worker Process 0 is running with pid: 26813
  -> Asynchronous Worker Process 1 is running with pid: 26815
  -> Synchronous Worker Process 2 is running with pid: 26817
  -> Synchronous Worker Process 3 is running with pid: 26819
  -> Synchronous Worker Process 4 is running with pid: 26821
  -> Synchronous Worker Process 5 is running with pid: 26811
```

Stop

This `asctl` method stops instantiated Orchestrator processes: manager, engine monitor, workers, and mongrels.

Method name	Stop
Command	`asctl orchestrator:stop`
Details	Stops instantiated orchestrator processes — manager, engine monitor, workers, and mongrels.
Inputs	None
Outputs	STDOUT to show that Orchestrator processes have been stopped.

```
# asctl orchestrator:stop
stopping manager, engine and workers..
stopped...

Orchestrator Status:
  -> Orchestrator Manager NOT running
  -> Orchestrator Engine NOT running
  -> Mongrel serving orchestrator on port 3000 is NOT running
  -> Mongrel serving orchestrator on port 3001 is NOT running
  -> Mongrel serving orchestrator on port 3002 is NOT running
  -> Orchestrator Monitor NOT running
  -> Asynchronous Worker Process 0 is NOT running
  -> Asynchronous Worker Process 1 is NOT running
  -> Synchronous Worker Process 2 is NOT running
  -> Synchronous Worker Process 3 is NOT running
  -> Synchronous Worker Process 4 is NOT running
  -> Synchronous Worker Process 5 is NOT running
```

Admin User Configuration

This `asctl` method creates a new admin user, or modifies the existing admin with the same name.

Method name	configure_admin_user + (*login*, *email*, *password*)
Command	`asctl orchestrator:configure_admin_user "login" "email" "password"`
Details	Creates a new admin user, or modifies the existing admin with the same name.
Inputs	User's login, and user's email and password.
Outputs	STDOUT indicates that the new admin has been created or an existing admin account has been modified.

```
# asctl orchestrator:configure_admin_user "administrator" "admin@asperasoft.com" "PASSPhrase"
An admin user has complete control over Orchestrator processes and jobs, continue? (y/n
default:n) y
Orchestrator: Configuration admin user... An install specific config file can be created
at /opt/aspera/var/config/orchestrator/orchestrator.yml to overwrite defaults
Created admin user with login administrator
done
[root@carmen_rhel7_1x64_154 ~]#
```

Asynchronous Worker Count

This `asctl` method displays the number of workers executing asynchronous steps.

Method name	asynchronous_worker_count

Command	`asctl orchestrator:asynchronous_worker_count count`
Details	Displays the number of workers executing asynchronous steps
Inputs	Is_Synchronous (boolean), Worker count (integer)
Outputs	STDOUT to indicate the worker count, or confirm an update if the worker count has been modified.

Synchronous Worker Count

This `asctl` method displays the number of workers executing synchronous steps.

Method name	worker_count(true, count)
Command	`asctl orchestrator:synchronous_worker_count [count]`
Details	Displays the number of workers executing synchronous steps.
Inputs	Is_Synchronous (boolean), Worker count (integer).
Outputs	STDOUT to indicate the worker count, or confirm an update if the worker count has been modified.

```
# asctl orchestrator:synchronous_worker_count
Orchestrator: Synchronous worker count...
4
done
```

Backup Database

This `asctl` method creates a full backup of the Orchestrator database.

If no file name is provided, the output will default to the following file path:

```
/opt/aspera/var/archive/orchestrator/databases/config
```

The file name `conf_`*`timestamp`*`.sql` is assigned.

Method name	backup_database(path)
Command	`asctl orchestrator:backup_database path`
Details	Creates a full backup of the Orchestrator database
Inputs	Path (string)
Outputs	STDOUT to indicate the database has been backed up. All Orchestrator backups are stored under Orchestrator Archive Directory (see section Orchestrator installation directories)

```
# asctl orchestrator:backup_database
Orchestrator: Backup databases... /opt/aspera/var/archive/orchestrator/databases/config/
conf_20160615_145908.sql
done
```

Base Port

This `asctl` method displays the lowest port number for mongrel instances.

Method name	base_port (port)
Command	`asctl orchestrator:base_port`
Details	Displays the lowest port number for mongrel instances.
Inputs	Port number (integer)
Outputs	STDOUT to display the current base port. If a port is provided, it replaces the current base port and displays a success message.

```
# asctl orchestrator:base_port
Orchestrator: Orchestrator Mongrel base port... Mongrel Currently hosted on
port
                                                                   #3000
done
```

Create Setup File

This `asctl` method creates a reusable file that contains responses to the setup prompts during the Orchestrator installation process.

Method name	create_setup_file(file_path)
Command	`asctl orchestrator:create_setup_file file_path`
Details	Creates a reusable file that contains responses to the setup prompts during the Orchestrator installation process.
Inputs	File path (string)
Outputs	STDOUT to indicate the progress of the snapshot and eventually a success or failure message

```
# asctl orchestrator:create_setup_file /tmp/setup.text


Streamlined: A simple setup with all components running on this computer.
Detailed:    An advanced configuration (e.g. non-standard ports or not all components running
on a single server)

Streamlined or detailed setup (s/d)? (current: s)

===================== Settings =====================
Orchestrator
  Enabled:                true
  Mongrel count:          3
  Mongrel base port:      3000
  Web root:               /aspera/orchestrator
  MySQL is local:         true

Saved to file '/tmp/setup.text'
```

Create Snapshot

This `asctl` method creates a snapshot of all Orchestrator configuration and design-time data.

Method name	snapshot
Command	`asctl orchestrator:create_snapshot`
Details	Creates snapshot of all Orchestrator configuration and design-time data.

Inputs	None
Outputs	STDOUT to indicate the progress of the snapshot and eventually a success or failure message

```
# asctl orchestrator:create_snapshot
Orchestrator: Create snapshot... Snapshot of Orchestrator configuration was successfully
created at 2016-06-15T14:55:25-07:00.
Snapshot file located at /opt/aspera/var/archive/orchestrator/snapshots/
Backup_at_20160615_145506.snap
done
```

Restore Orchestrator from a Snapshot

This `asctl` method restores Orchestrator configuration and design time data from a snapshot.

Note: To avoid issues with the restoration process:

- Before you begin, stop Orchestrator with this command:

  ```
  # asctl orchestrator:stop
  ```

  When the process is complete, restart Orchestrator with this command:

  ```
  # asctl orchestrator:start
  ```

- Only use snapshots with the same version of Orchestrator from which they were taken. For example, with an installation of Orchestrator 4.0, only import a snapshot taken from v. 4.0, not a snapshot taken from v. 3.2.

Method name	restore_snapshot(filepath)
Command	`asctl orchestrator:restore_snapshot [filepath]`
Details	Restores Orchestrator configuration and design time data from a snapshot. If a snapshot file is provided, the current configuration will be backed up and the instance will be reverted to the supplied snapshot file
Inputs	Filepath (string)
Outputs	STDOUT the progress of the operation

Disable Orchestrator

This `asctl` method disables Orchestrator.

Method name	disable
Command	`asctl orchestrator:disable`
Details	Disables Orchestrator and removes it from `/etc/init.d/` and `/etc/rc.d/`
Inputs	None
Outputs	STDOUT to indicate Orchestrator is disabled.

```
# asctl orchestrator:disable
Orchestrator: Unregister with Apache... done
Orchestrator: Uninstall service... done
Orchestrator: Disable... done
```

Generate Config

This `asctl` method regenerates the configuration files using the current settings.

Method name	generate_config
Command	`asctl orchestrator:generate_config`
Details	Regenerate the configuration files using the current settings
Inputs	None
Outputs	STDOUT to indicate the configuration file has been generated

Help

This `asctl` method describes the available commands and usage.

Method name	help
Command	`asctl orchestrator:help`
Details	Describes the available commands and usage
Inputs	None
Outputs	STDOUT the text describing the commands and usage

```
# asctl orchestrator:help
Orchestrator: Orchestrator ASCTL help...
Orchestrator Commands (prefix with 'orchestrator:')


asynchronous_worker_count       To view (Or change) the asynchronous thread count
backup_database                 Back up the orchestrator database(full backup) into a dump file
under /opt/aspera/var/archive/orchestrator/databases/config.
base_port                       Display/change lowest port for the Mongrel instances.
change_configuration_parameter  Modify a configuration parameter
configure_admin_user            To Configure the setup of a admin user
create_setup_file               Create a reusable file that contains answers to the setup
questions.
create_snapshot                 Create snapshot of all Orchestrator configuration and design-
time data.
disable                         Disables Orchestrator
export_plugin                   To export an existing plugin from the Orchestrator
export_workflow                 To export an existing workflow from the Orchestrator
generate_config                 Regenerate the configuration files using the current settings
generate_config_file            Generate the config file
help                            Describe the commands and usage
info                            Print configuration info about this component
list_configuration_parameters           List all modifiable configuration parameters
list_workflows                  List all workflows designed in this server
migrate_database                Update database to latest schema.
mongrel_count                   Display the current count of mongrels configured to run.
rake                            Invoke a rake command
rake_command                    Execute a rake command
restart                         Restart Orchestrator
restore_database                Restores mysql from the dump file provided
restore_db_from_file            Restores mysql from the dump file provided
restore_snapshot                Restore Orchestrator configuration and design time data from a
snapshot.
setup                           Configure the Orchestrator component
setup_from_file                 Configure the Orchestrator using the given file settings
start                           To start the Orchestrator engine
start_mongrels                  Start the processes rendering the UI
status                          Shows the current status of the Orchestrator
stop                            To stop the Orchestrator engine
stop_mongrels                   Stop the processes rendering the UI
switch_orchestrator_version     To switch the Orchestrator version
synchronous_worker_count        To view (Or change) the synchronous thread count
```

```
upgrade                         To upgrade the Orchestrator to a new version
uri_namespace                   Display/Change the URI namespace
version                         Display Orchestrator Version
web_root                        Display/Change the web root of Orchestrator
done
```

Info

This `asctl` method prints configuration information about the Orchestrator components.

Method name	info
Command	`asctl orchestrator:info`
Details	Prints configuration information about the Orchestrator components
Inputs	None
Outputs	STDOUT the configuration info about Orchestrator

```
# asctl orchestrator:info
Orchestrator
  status:
Orchestrator Status:
  -> Orchestrator Manager running with pid: 11841
  -> Orchestrator Engine running with pid: 12084
  -> Mongrel serving orchestrator on port 3000 is running with pid: 12089
  -> Mongrel serving orchestrator on port 3001 is running with pid: 12091
  -> Mongrel serving orchestrator on port 3002 is running with pid: 12093
  -> Orchestrator Monitor running with pid: 12087
  -> Asynchronous Worker Process 0 is running with pid: 12100
  -> Asynchronous Worker Process 1 is running with pid: 12102
  -> Synchronous Worker Process 2 is running with pid: 12104
  -> Synchronous Worker Process 3 is running with pid: 12106
  -> Synchronous Worker Process 4 is running with pid: 12108
  -> Synchronous Worker Process 5 is running with pid: 12098
  enabled:          true
  uri_namespace:    /aspera/orchestrator
  mongrel_count:    3
  base_port:        3000
```

List Configuration

This `asctl` method lists all modifiable configuration parameters.

Method name	list_configuration_parameters
Command	`asctl orchestrator:list_configuration_parameters`
Details	Lists all modifiable configuration parameters
Inputs	None
Outputs	STDOUT the modifiable configuration parameters

```
# asctl orchestrator:list_configuration_parameters
Orchestrator: List configuration parameters...

strong_password_regex         :                  ^(?=.{8,})(?=.*[a-z])(?=.*[0-9])(?=.*[A-Z])(?
=.*[@#$%^&+=]).*$
rails_log_level               :                  error
log_level                     :                  debug
journal_on                    :                  true
asynch_threads                :                  2
engine_heartbeat              :                  1
date_format                   :                  american
custom_tables                 :
max_tasks_displayed           :                  12
log_retention_days            :                  7
```

```
install_images_dir              :                   /opt/aspera/var/config/orchestrator/images
install_config_file             :                   /opt/aspera/var/config/orchestrator/
orchestrator.yml
delete_journal_with_workorder :                     false
make_view_by_workflow_default :                     false
fast_start                      :                   false
active_active                   :                   false
install_css_dir                 :                   /opt/aspera/var/config/orchestrator/stylesheets
mongrel_starting_port           :                   3000
journal_kept_duration           :                   10
impose_strong_password          :                   false
auto_activate_portlets          :                   false
referer_hosts                   :
ad_user_groups                  :                   Contributor
load_basic_workflow_dir         :                   /opt/aspera/orchestrator/vendor/workflows
log_file                        :                   /opt/aspera/var/run/orchestrator/log/
orchestrator.log
run_dir                         :                   /opt/aspera/var/run/orchestrator
purge_cutoff                    :                   60
use_db_time                     :                   false
orchestrator_dir                :                   /opt/aspera/orchestrator
allowed_logon_attempts          :                   3
ifv_skip_start_event            :                   false
persistence_mode                :                   disk
mongrel_action_timeout          :                   300
hide_preferences_for_groups    :
personalized                    :                   true
install_pages_dir               :                   /opt/aspera/var/config/orchestrator/pages
archive_dir                     :                   /opt/aspera/var/archive/orchestrator
default_step_timeout            :                   60
synch_threads                   :                   4
build                           :                   1
db_bin                          :                   /opt/aspera/common/mysql/bin
publish_default_workflows       :                   true
use_new_designer                :                   true
web_root                        :                   /aspera/orchestrator
maintenance_interval            :                   600
disable_autocomplete_login_page:                            false
config_dir                      :                   /opt/aspera/var/config/orchestrator
allow_blank_password            :                   false
aspera_dir                      :                   /opt/aspera
cleanup_cutoff                  :                   30
perform_ad_operation            :                   false
external_javascript_lib         :
strict_prepost_processing       :                   false
engine_instance                 :                   0
external_facing_uri             :
load_default_workflows          :                   true
maintenance_heartbeat_interval:                     300
logon_timeout                   :                   0
strict_cache_control            :                   false
mongrel_max_serve               :                   0
mongrel_max_daemons             :                   3
task_refresh_frequency          :                   30
done
```

List Workflows

This method lists all workflows designed on the server where you run the `asctl` command.

Method name	list_workflows(workflow_folder, status_filter, format)
Command	`asctl orchestrator:list_workflows [workflow_folder] [status_filter] [format]`
Details	Lists all workflows designed on the server where you run the `asctl` command.
Inputs	Workflow_folder (String), Status_Filter (String), Format (String)
Outputs	STDOUT the workflows on this server. If a workflow folder is specified, show workflows inside that folder only. Provide options to output an XML file/string or JSON String

```
# asctl orchestrator:list_workflows
Orchestrator: List workflows...
WorkFlow List(Workflow names along with their ids)
"Journal Misc"                                              10
"Sync from GMS Server TEST"                                 14
"Journal"                                                   8
"test"                                                      16
"UserInput"                                                 12
"A_Error_Workflow"                                          4
"A_Failed_workflow"                                         3
"A_weighted_workflow"                                       1
"Baton Failure Prompt"                                      22
"ESPN: Sub - Create package and send via Faspex"            20
"ESPN: Sub - Generate_XML_and_Transfer"                     19
"test  (import_1)"                                          6
"SpreadSheetParser"                                         17
"AAA"                                                       13
"Test"                                                      5
"ESPN: Sub - TAR Media files in a Queue"                    18
"Node with Params"                                          11
"ESPN: Pick up ESPN packages and transfer via Faspex"       21
"QueueStager"                                               15
"Xytech"                                                    7
done
```

Migrate Database

Updates the Orchestrator database to the latest schema.

Method name	migrate_database
Command	`asctl orchestrator:migrate_database`
Details	Updates the Orchestrator database to the latest schema.
Inputs	None
Outputs	STDOUT the result of the operation

```
# asctl orchestrator:migrate_database
Orchestrator: Backup databases...
done
```

Start Mongrels

This `asctl` method starts the processes that render the UI.

Method name	start_mongrels
Command	`asctl orchestrator:start_mongrels`
Details	Starts the processes that render the UI.
Inputs	None
Outputs	STDOUT to show that Orchestrator Mongrel processes have started.

```
# asctl orchestrator:start_mongrels
An install specific config file can be created at /opt/aspera/var/config/orchestrator/
orchestrator.yml to overwrite defaults
Command start_all executed.
```

Stop Mongrels

This `asctl` method stops the processes that render the UI.

Method name	stop_mongrels
Command	`asctl orchestrator:stop_mongrels`
Details	Stops the processes that render the UI.
Inputs	None.
Outputs	STDOUT shows that Orchestrator mongrel processes have been stopped.

```
# asctl orchestrator:stop_mongrels
An install specific config file can be created at /opt/aspera/var/config/orchestrator/
orchestrator.yml to overwrite defaults
Command stop_all executed.
```

Mongrel Count

Displays the current count of mongrels that are configured to run. Mongrels render the Orchestrator UI.

Method name	mongrel_count(number)
Command	`asctl orchestrator:mongrel_count [number]`
Details	Displays the current count of mongrels that are configured to run.
Inputs	None
Outputs	STDOUT the result of the operation

Rake Commands

This `asctl` method invokes a rake command.

Method name	rake_command
Command	`asctl orchestrator:rake [arguments]`
Details	Invoke a rake command. The arguments are determined by the command executed.
Inputs	None
Outputs	STDOUT the result of the operation

Restart

This `asctl` method restarts Orchestrator.

Method name	restart
Command	`asctl orchestrator:restart`
Details	Restarts Orchestrator
Inputs	None
Outputs	STDOUT the stop operation, followed by the start operation

Note: After running `asctl orchestrator:restart`, run `asctl all:status` to confirm the restart.

```
# asctl orchestrator:restart
stopping manager, engine and workers..
stopped...

Orchestrator Status:
  -> Orchestrator Manager NOT running
  -> Orchestrator Engine NOT running
  -> Mongrel serving orchestrator on port 3000 is NOT running
  -> Mongrel serving orchestrator on port 3001 is NOT running
  -> Mongrel serving orchestrator on port 3002 is NOT running
  -> Orchestrator Monitor NOT running
  -> Asynchronous Worker Process 0 is NOT running
  -> Asynchronous Worker Process 1 is NOT running
  -> Synchronous Worker Process 2 is NOT running
  -> Synchronous Worker Process 3 is NOT running
  -> Synchronous Worker Process 4 is NOT running
  -> Synchronous Worker Process 5 is NOT running
starting...
started...
nohup: redirecting stderr to stdout

Orchestrator Status:
  -> Orchestrator Manager running with pid: 29812
  -> Orchestrator Engine NOT running
  -> Mongrel serving orchestrator on port 3000 is NOT running
  -> Mongrel serving orchestrator on port 3001 is NOT running
  -> Mongrel serving orchestrator on port 3002 is NOT running
  -> Orchestrator Monitor NOT running
  -> Asynchronous Worker Process 0 is NOT running
  -> Asynchronous Worker Process 1 is NOT running
  -> Synchronous Worker Process 2 is NOT running
  -> Synchronous Worker Process 3 is NOT running
  -> Synchronous Worker Process 4 is NOT running
  -> Synchronous Worker Process 5 is NOT running

# asctl all:status
Apache:                 running
MySQL:                  running

Orchestrator Status:
  -> Orchestrator Manager running with pid: 29812
  -> Orchestrator Engine running with pid: 29825
  -> Mongrel serving orchestrator on port 3000 is running with pid: 29830
  -> Mongrel serving orchestrator on port 3001 is running with pid: 29832
  -> Mongrel serving orchestrator on port 3002 is running with pid: 29834
  -> Orchestrator Monitor running with pid: 29828
  -> Asynchronous Worker Process 0 is running with pid: 29841
  -> Asynchronous Worker Process 1 is running with pid: 29843
  -> Synchronous Worker Process 2 is running with pid: 29845
  -> Synchronous Worker Process 3 is running with pid: 29847
  -> Synchronous Worker Process 4 is running with pid: 29849
  -> Synchronous Worker Process 5 is running with pid: 29839
```

Restore a Database from a MySQL Dump File

This `asctl` method restores a database from a MySQL dump file so it becomes the primary database.

Method name	restore_database(filepath)
Command	`asctl orchestrator:restore_database [filepath]`
Details	If you provide a MySQL dump file that contains an Orchestrator database, the current database is backed up and the database from the MySQL dump file is used as the primary database.
Inputs	Filepath (string)
Outputs	STDOUT the progress of the operation

Set Up from File

This `asctl` method runs Orchestrator setup, using the configuration found in the file that was created with the asctl method `create_setup_file`.

Method name	setup_from_file (filepath)
Command	`asctl orchestrator:setup_from_file [filepath]`
Details	Runs Orchestrator setup, using the configuration found in the file that was created with the asctl method `create_setup_file`
Inputs	Filepath (string)
Outputs	STDOUT the progress of the operation

```
# asctl orchestrator:setup_from_file /tmp/setup.txt

===================== Settings =====================
Orchestrator
  Enabled:                true
  Mongrel count:          3
  Mongrel base port:      3004
  Web root:               /aspera/orchestrator
  MySQL is local:         true

Are these settings correct? (y/n/x with x for exit) y
Orchestrator
  Enabling orchestrator...

--Generating Orchestrator config file
--Setting up database configuration file
(in /opt/aspera/orchestrator-version)
orchestrator already exists
orchestrator already exists
orchestrator_test already exists
(in /opt/aspera/orchestrator-version)
--Generating Orchestrator executables
--Setting up Orchestrator service
/sbin/chkconfig
--Setting up install specific images
--Setting up install specific stylesheets
--Setting up install specific pages
--Setting up Apache as a proxy
Enable Apache proxying for Apache (from Aspera common)
Updating Apache configuration file /opt/aspera/common/apache/custom/
orchestrator_orchestrator.conf
To start Aspera Orchestrator, run  asctl orchestrator:start
done
  Task Name                                                 Status
==================================================================
Orchestrator
  Enabling orchestrator                                        OK
```

URI Namespace (Webroot)

This `asctl` command configures or changes Orchestrator web root parameter.

Method name	uri_namespace (namespace)
Command	`asctl orchestrator:uri_namespace [namespace]`
Details	Configures or changes the Orchestrator web root parameter
Inputs	Namespace (string)
Outputs	STDOUT the result of the operation

```
# asctl orchestrator:uri_namespace
Orchestrator: /aspera/orchestrator
```

Version

This `asctl` command displays the currently installed version of Orchestrator.

Method name	version
Command	**asctl orchestrator:version**
Details	Displays the currently installed version of Orchestrator.
Inputs	Filepath (string)
Outputs	STDOUT the version of Orchestrator

```
# asctl orchestrator:version
Orchestrator: Orchestrator version... Orchestrator Version is version-00001
done
```

Inline Validation with Orchestrator

Overview of Inline File Validation

If an executable file containing malicious code is uploaded to the server, the malicious code can subsequently be executed by an external product that integrates with an Aspera product.

Inline file validation is a feature that enables file content to be validated while the file is in transit, as well as when the transfer is complete. The validation check is made with a Lua script or with a RESTful call to an external URL. The mode of validation used (URL or Lua) and the timing of the check are set in the Aspera server GUI or`aspera.conf`.

For information about validating with a URL or Lua script, see "Inline File Validation with URI" on page 149 or "Inline File Validation with Lua Script" on page 151.

Specifying the validation source: When validating using a Lua script, you must specify the path to the script in the Aspera server GUI or `aspera.conf`. When validating using a URL, you must specify the URL in `aspera.conf`.

Scheduling validation: You can schedule validation to occur at the following events:

- run at file start
- run at file stop
- run at session start (URL validation is not supported)
- run at session stop (URL validation is not supported)
- run when crossing file threshold (the limit must be set in `aspera.conf`)

Multi-threaded validation: You can increase the number of threads used for validation, allowing validation processes to occur in parallel.

Specifying validation targets: You can include or exclude files with certain characteristics, including the following:

- Accept only files of certain types (extensions)
- Examine the `.zip` / archive files for unapproved file types
- Disallow files greater that a certain size

Requirements for Inline Validation

- Orchestrator is installed with default options.
- One or more Aspera transfer servers (Enterprise Server, Connect Server, Point-to-Point, or Faspex) have been installed with default options.
- An appropriate version of the transfer server has been installed (run **ascp –A** to check the version number).

  You may use any version of the transfer server to perform inline validation with URI. However, to validate with Lua script, v. 3.7.1 or above is required.
- Ports 80 and 443 are open between the Aspera transfer server and Orchestrator.

Performing Inline File Validation with Orchestrator (Validation with URI)

This article describes how to configure the transfer server for inline file validation with URI.

For reference information about the following topics, see the Aspera Enterprise Server Admin Guide:

- Inline file validation with URI or Lua script (two articles); note that validation with Lua script is only available for transfer servers v. 2.7.1 and above
- Inline validation parameters, described in the General Configuration Reference section (file handling and `aspera.conf` transfer articles)

1. Publish a workflow.

   Do one of the following:

   - Design and publish a workflow in Orchestrator.
   - Import a workflow and publish it.

   The workflow must contain a run-time parameter with name `File` and type `Hash`.

   Orchestrator populates the `File` parameter with a hash when **ascp** triggers an API call. For example:

   ```
   {"size"=>"1048576", "session_id"=>"12a66e2b-8c4b-4cf3-b220-737712918071",
   "target_rate_kbps"=>"500", "cipher"=>"aes-128", "file_name_encoding"=>"utf8",
   "min_rate_kbps"=>"0", "startstop"=>"running", "user_id"=>"0", "host"=>"10.0.136.8",
   "rate_policy"=>"fair", "file"=>"/data/root/tmp/test/1m1995", "user"=>"root",
   "direction"=>"recv", "start_byte"=>"0"}
   ```

   In the example above, the `"file"` key is used for processing, and the `"startstop"` key is used to route the workflow.
2. Obtain the Aspera call URL.

   This URL will be entered later in `aspera.conf`, in the `<validation_uri>` parameter.

   Follow the procedure in "Generating a Workflow Call URL" on page 112 to obtain the Aspera call URL of the workflow published in Step 1. The URL should contain a hostname, a route to the Orchestrator controller, and the workflow ID. Make sure to add the port to the URL if it is different from the default port, 443. Below is a sample URL.

   ```
   https://hostname/aspera/orchestrator/external_calls/validate/35?user=admin&password=admin
   ```

   In this example, *hostname* is the IP address or fully qualified domain name of the server on which Orchestrator is installed.
3. To configure the transfer server for validation with URI, edit the configuration file, `aspera.conf`.

   Open the `aspera.conf` (located at `/opt/aspera/etc/aspera.conf`). For all available configuration options, see the IBM Aspera High-Speed Transfer Server Admin Guide.

   Note: If the validation takes more than 30 seconds, add the following line to the configuration to avoid premature timeout:

```
<default>
     <session_timeout_sec>60</session_timeout_sec>
</default>
```

The value between the `<session_timeout_sec>` tags can be increased as needed.

4. Increase the number of mongrels.

   For production servers, Aspera recommends that you increase the number of mongrels; this insures there are enough so that mongrel usage for **ascp** calls can be separated from mongrel usage for Orchestrator. Follow the procedure in "Adding Mongrel Processes for a New Apache Port" on page 79 to increase the number of mongrels.

   Note: Apache must be restarted after this step.

Inline File Validation with URI

For general information about setting up inline file validation, see "Overview of Inline File Validation" on page 147.

Configuration

To set up a validation handler for inline file validation, configure the REST service and set the URL in `aspera.conf`.

The code examples in the steps below are for an admin using a Java servlet deployed on an Apache web server, but this process is generalizable to other programming languages and other servers.

1. Configure the REST service.

   Open `web.xml` and configure the `<servlet>` and `<servlet_mapping>` sections to provide the necessary information for validation.

   Note: The `<servlet-name>` (URL handler) value is reused in both `aspera.conf` (in the next step) and custom code (see **Custom Code for Including and Excluding Files**, below).

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/
javaee/web-app_3_1.xsd"
         version="3.1">

    <servlet>
        <servlet-name>SimpleValidator</servlet-name>
        <servlet-class>aspera.validation.SimpleValidator</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>SimpleValidator</servlet-name>
        <url-pattern>/SimpleValidator/validation/files</url-pattern>
    </servlet-mapping>
</web-app>
```

2. Set the URL in `aspera.conf`.

   Run the following command:

```
# asconfigurator -x "set_user_data;user_name,username;validation_uri,url"
```

   Where *url* is the server's IP address and port, and the servlet name (URL handler) found in `web.xml`. This adds the path to the `<transfer>` section of `aspera.conf`. For example:

```
<transfer>
<validation_uri>http://127.0.0.1:8080/SimpleValidator</validation_uri>
</transfer>
```

Validation Requests and Returned Responses

During the inline validation process, **ascp** automatically generates a JSON-based request. The call is made with the URL defined in `aspera.conf`. For example:

```
POST URL/validation/files HTTP/1.1
Content-type: application/json
```

The system then generates a JSON accepted or error response (`OK` or `Bad Request`). If a file validation fails, it terminates the session with an error message from the URI.

- **Sample JSON accepted response:** The `"file_encryption"` field is only returned if server-side EAR is present.

```
HTTP 200 OK
{
    "id" : "1111-2222-333",
    "file_encryption" : {
        "passphrase" : "supersecret"
    }
    "aspera_response_object_name" : {
        "startstop" : "start"
        "xfer_id" : "AAAA-BBBB",
        .  .  .
        "file_csum" : "a1000abf882",
        "file_csum_type" : "sha2-256"
    }
}
```

- **Sample JSON error response:**

```
HTTP 400 Bad Request
{
  "error" : {
    "code" : "1022",
    "message" : "The file fails validation"
  }
}
```

Custom Code for Including and Excluding Files

Administrators can include or exclude files by enabling whitelisting, blacklisting, or another method of their own design. You can do this by creating custom code in the programming language of your choice, using a web server that runs a REST service. (Connect Server users have the option to use the web server associated with that installation).

The following is an example of custom code that creates a file blacklist, using a Java servlet deployed on an Apache web server. Note that this code uses the servlet name `SimpleValidator`, which was defined in `web.xml` above.

```
package aspera.validation;

import com.google.gson.Gson;
import com.google.gson.JsonObject;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.BufferedReader;
import java.io.IOException;

@WebServlet(name = "SimpleValidator")
public class SimpleValidator extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        StringBuilder fileRequestJSON = new StringBuilder();
        BufferedReader reader = request.getReader();
        String line = "";
        Gson gson = new Gson();
        System.out.println("Got Validation request...");
        while (line != null) {
            line = reader.readLine();
            if (!(line == null)) {
                fileRequestJSON.append(line).append("\n");
            }
        }

        ValidationInput validationInput = gson.fromJson(fileRequestJSON.toString(),
ValidationInput.class);

        System.out.println("FileData JSON: " + fileRequestJSON.toString());

        if (validationInput.file != null && validationInput.file.endsWith(".sh")
           || validationInput.file.endsWith(".exe")) {

            JsonObject innerObject = new JsonObject();
            innerObject.addProperty("message", "Cannot transfer executable file!!");
            innerObject.addProperty("code", 1);

            JsonObject jsonObject = new JsonObject();
            jsonObject.add("error", innerObject);

            response.getOutputStream().println(jsonObject.toString());

            response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
        }
        else {

            JsonObject jsonObject = new JsonObject();
            jsonObject.addProperty("success", true);
            jsonObject.addProperty("data", "File is ok to transfer");
            jsonObject.addProperty("code", 1);
            response.getOutputStream().println(jsonObject.toString());

            response.setStatus(HttpServletResponse.SC_OK);
        }
        return;
    }
}
```

Inline File Validation with Lua Script

For general information about setting up inline file validation, see "Overview of Inline File Validation" on page 147

To use a Lua script for inline file validation, the administrator creates a base-64 encoded Lua action script and sets the path to that script in the GUI or in the `<transfer>` section of `aspera.conf`. During the inline validation, **ascp** automatically generates a request; the parameters for the Lua call are passed to a Lua script defined in `aspera.conf`.

The parameters for Lua calls are passed to Lua scripts by using the array `'env_table'`. The following is an example request body:

```
env_table["startstop"] = "running"
env_table["xfer_id"] = "AAAA-BBBB"
env_table["session_id"] = "1111-2222"
env_table["host"] = "10.0.258.12"
env_table["client_ip"] = "10.0.125.04"
env_table["user"] = "admin"
env_table["userid"] = 24
env_table["direction"] = "send"
env_table["target_rate_kbps"] = 0
env_table["min_rate_kbps"] = 0
env_table["rate_policy"] = "fair"
env_table["cipher"] = "aes-128"
env_table["cookie"] = "xyz"
env_table["manifest_file"] = "/data/manifests/aspera-transfer-1234.txt"
env_table["file"] = "/data/home/luke/test.mpg"
env_table["size"] = 1000000
env_table["start_byte"] = 0
env_table["bytes_written"] = 0
env_table["tags"] = "tags"
env_table["file_name_encoding"] = "utf8"
env_table["file_csum"] = "a1000abf882"
env_table["file_csum_type"] = "sha2-256"
```

Lua Request Body Parameters and Values

Field	Description	Values
`"startstop"`	Sets the type of validation	`start`, `stop`, or `running`
`"xfer_id"`	Value used to identify a transfer session	String
`"session_id"`	Value used to identify a validation session	String
`"host"`	Server hostname or IP address	Hostname or IP address
`"client_ip"`	Client IP address	IP address
`"user"`	SSH account login name	String
`"user_id"`	Value used to identify the user	String
`"direction"`	Direction of transfer (send or receive)	`send` or `recv`
`"target_rate_kbps"`	Target rate (in kbps) for file transfer with **ascp**	Integer
`"min_rate_kbps"`	Minimum rate (in kbps) for file transfer with **ascp**	Integer
`"rate_policy"`	Defines the **ascp** rate policy. This value is taken from the default configuration in the GUI or `aspera.conf`, if not defined here.	`fixed`, `fair`, `high`, or `low`
`"cipher"`	The encryption cipher for file data.	String; `AES128`, `ANY`, or `NONE`
`"cookie"`	The cookie sent to the client system	String
`"manifest_file"`	Path to manifest file, which contains a list of transferred files. The command for this in **ascp** is --file-manifest-path=*file_path*	Filepath
`"file"`	Path to file being validated	Filepath
`"size"`	Allowable file size	Integer (up to 64-bit)
`"start_byte"`		Integer
`"bytes_written"`		Integer
`"tags"`	The JSON request passes the supplied tag values to **ascp**, which in turn passes the tags to the validator.	
`"file_name_encoding"`		String
`"file_csum"`	File checksum	String
`"file_csum_type"`	File checksum type	`md5`, `sha1`, or `any`

The returned values from Lua can be used to indicate validation success, validation failure, the script error, or to change the file destination:

Status	Lua return value
Validation success	No value or `LRET_OK`
Change file destination	`LRET_REDIRECT_DST` followed by a new destination path for the file. This option is only available at file transfer start, returning it at any other state results in an error.

Status	Lua return value
Validation failure	`LRET_INVALID`, optionally followed by a failure description string
Script error	`LRET_ERROR` followed by an error number or error description string

Lua File Interfaces

Three Lua file interfaces allow Lua scripts to reference files: `lua_stat`, `lua_file_delete`, and `lua_rename`.

-
  ```
  lua_stat("file_path")
  ```

  Used to gather metadata for the single file specified by *file_path*, where *file_path* is relative to the docroot, if defined. Metadata output include the following:

  ```
  stat_data["exists"] = "true" | "false"
  stat_data["size"] = file_size
  stat_data["blocks"] = file blocks
  stat_data["blocksize"] = block_size
  stat_data["type"] = "Invalid" | "S_IFDIR" | "S_IFREG"  | "S_IFCHR" |"S_IFBLK" | "S_IFIFO" |
  "S_IFSOCK" |
               "S_IFLNK" | "Block stream" | "Custom" | "Unknown"
  stat_data["mode format"] = "Windows format" | "Linux format"
  stat_data["mode"] = filemode (format based on mode format above)
  stat_data["uid"] = uid
  stat_data["gid"] = gid
  stat_data["ctime"] = ctime
  stat_data["mtime"] = mtime
  stat_data["atime"] = atime
  ```

-
  ```
  lua_file_delete("file_path")
  ```

  Deletes the single file specified by *file_path*, where *file_path* is relative to the docroot, if defined.

-
  ```
  lua_rename("old_file_path","new_file_path")
  ```

  Renames the file specified by *old_file_path* with the new name specified by *new_file_path*, both of which are relative to the docroot, if defined.

Lua Logging Interface

You can output simple text strings (format strings are not supported) to the Aspera logs using the **ascp** log interfaces. For example, to log when the Lua script started, enter the following line in a Lua script:

```
lua_log("Lua script started")
```

This produces the following log entry:

```
xxxxxx LOG lua: Lua script started
```

The following **ascp** logging functions are supported:

- `as_log`
- `as_err`
- `as_dbg1`
- `as_dbg2`
- `as_dbg3`
- `as_dbg4`

To use the **ascp** log functions in your Lua script, replace `as` with `lua`.

Miscellaneous Lua Interfaces

-
  ```
  lua_override_ear_secret("secret")
  ```

  Override the server-side encryption-at-rest (EAR) secret that is set in `aspera.conf` with the specified secret.

Overview of the Orchestrator API

Overview

Orchestrator exposes the endpoints of web services; this allows third-party applications to programmatically control some of its functions, such as launching and monitoring work orders. The Orchestrator API is REST-based.

- **Format:** All API calls have JSON and XML response support. The data format in the responses can be XML or JSON. The response format can be controlled by submitting a parameter format with a value of `xml` or `json` or by adding extensions to the `id` parameter. Detailed examples are provided throughout this guide.
- **Status codes:** The status code in the HTTP responses is `200` for success and `400` and greater for failures. These error codes include `400 (Bad request)`, `401 (Access Denied)`, `403 (Authentication failure)` and `404 (Method not found)`.
- **Methods:** The HTTP GET method is supported for all requests. The HTTP POST method is also supported for the process of initiating a work order with user-defined data in the POST body (for example, XML data).

API Authentication

API calls require proper authentication and have the same authorization requirements as those applied to direct access to Orchestrator through its web-based user interface.

Note: Aspera endorses secure methods for logging in, such as HTTP basic authentication, an API key, or a scrambled password. Aspera *does not* endorse the method of entering a user name and password inline or in the UI.

There are a number of ways to authenticate API calls; these are described in the table below.

Authentication Model	Description
HTTP basic authentication (most secure)	HTTP basic authentication is supported and is recommended to developers as the primary authentication model. The advantage of this method is that the credentials are within each request but they are encoded versus being in plain text. The username and password are combined into a string, `"username:password"`, then they are Base64-encoded and put in the HTTP Authorization header. For example: `Authorization: Basic access_token`
API key or scrambled password	An API key and a scrambled password are generated for each Orchestrator user; either of these values can be used in the REST URLs for authentication. For example: `https://orchestrator3.asperademo.com/aspera/orchestrator/api/initiate/124.json?login=admin&`**`api_key=ghawQesLH2OnnLxpHNAeCwzzm81071841979945`** `https://orchestrator3.asperademo.com/aspera/orchestrator/api/initiate/124.json?`

Authentication Model	Description
	`login=admin&`**`password=_P__Agx825xg1h10p0h7jzn04b03748jhkoi7738505 47`** To obtain the values for the API key or password, do the following: 1. At the top-right of the Orchestrator UI, click the **admin** dropdown arrow and click **Accounts**. 2. In the far-left navigation, click **Users**. 3. In the Users list, click the name for the desired user. The Modify User Information section appears at the topic of the page. 4. For an API key, copy the string that is in the API Key field. For a password, click the **Generate Scrambled Password** button and copy the string that appears.

Application cookie	This can be used if the client is able to use cookies to maintain state information. Once the authentication is made, it will not be necessary to set the credentials again for additional calls.
	
	The user obtains the cookie from the authentication call by supplying a username and password. If this is a particular process that will only ask for a cookie once, then the logoff for that process will also need a cookie to correctly log off the user for which the cookie is set. The cookie should be supplied in the HTTP header.
	
	The host name in the URLs is defined as *Orchestrator_IP_address* in this document to represent the IP address or the fully qualified host name of your Orchestrator node.
	
	Below is an example of a cookie-style request:
	
	`http://`*`Orchestrator_IP_address`*`/aspera/ orchestrator/api/authenticate/admin? password=_P__Agx825xg1h10p0h7jzn04b03748jhkoi773850547`
	
	The following `curl` requests are used to obtain an authentication cookie.
	
	• Get the cookie and write to file:
	
	`curl -v  -b orch.cookies -c orch.cookies -k -X GET 'https://10.0.142.14/ aspera/orchestrator/workflow_reporter/authenticate/admin? password=_P__Agx825xg1h10p0h7jzn04b03748jhkoi773850547'`
	
	• Expose the cookie for API calls:
	
	`curl -v  -b orch.cookies -c orch.cookies -k -X GET 'https://10.0.142.14/aspera/orchestrator/ workflow_reporter/workflows_list/0'`
	
	After a successful authentication, the next URL (for example, one used to initiate a work order) can be called without providing the login and password parameters; this continues until the user is logged off.
	
	Note: To enable CORS on an Apache server, replace the `password=` in the cookie request with `api_key=`.

Sample XML response for success:

```
<?xml version="1.0"?\>
<user time="2011-10-20 17:47:05 UTC" action="authenticate" id="1" login="admin"/>
```

Sample XML response for a bad password case:

```
<?xml version="1.0"?>
<error time="2011-10-20 17:45:39 UTC" action="status" id="0">#<RuntimeError:
Authentication failed></error>
```

Sample XML response for case in which the user does not exist:

```
<?xml version="1.0"?>
<error time="2011-10-20 17:48:09 UTC" action="status" id="0">#<RuntimeError:
Authentication failed></error>
```

To log off, use the following URL:

```
http://Orchestrator_IP_address/aspera/orchestrator/users/logoff/xml
```

Note: You must include the cookie you obtained when you logged in.

Sample XML response for logoff:

```
<?xml version="1.0"?>
<user time="2012-01-25 06:38:22 UTC" action="logoff" id="1" login="admin"/>
```

Available Endpoints in Orchestrator

This page has links to reference information for all API endpoints in Orchestrator.

General

Endpoint	Description
"Persist Custom Data in a Database" on page 164	Submit data to Orchestrator that is persisted in a database table called `shared_states`. For example, a plugin can send a Web notification upon transcode complete or failure.
"Get Alerts for a User" on page 166	Get user alerts based on a particular alert category. You can get user alerts based on a particular alert category, and you can filter for the number of alerts, only.

Monitors

Endpoint	Description
"Activate Monitors" on page 167	Activate monitors for workflows, work orders, scripts, folders, and nodes.
"Deactivate Monitors" on page 168	Dectivatesmonitors for workflows, work orders, scripts, folders, and nodes.
"List Monitors" on page 169	Return a list of all monitors of a particular type—workflow, work order, script, folder, or node—that are available in Orchestrator.

Plugins

Endpoint	Description
"Plugin Version" on page 170	Get the installed version of a single plugin or all plugins.

Endpoint	Description
"Reload One Plugin or All Plugins" on page 173	Reload a plugin, which prompts Orchestrator to update the code for that plugin. If you do not specify a plugin in the request, Orchestrator reloads all plugins.
"Reload Plugin Set" on page 174	Reload a defined set of plugins, which prompts Orchestrator to update the code for those plugins.
"Enable One Plugin or All Plugins" on page 175	If you specify a plugin name in the request, enable the plugin so that it can be used in a workflow. If you do not specify a plugin name, enable all Orchestrator plugins.
"Import a Plugin" on page 175	Import a plugin into your Orchestrator instance.

Portlets

Endpoint	Description
"Get the Dashboard" on page 165	Get the Orchestrator dashboard.
"Portlet Version" on page 171	Get the installed version of a single portlet or all portlets.

Queues

Endpoint	Description
"Queue Item" on page 176	Add an item to a queue.
"Get Queued Items From a Queue" on page 176	Get a list of queued items from a queue.
"Look Up an Item in a Queue" on page 177	Search for an item in one or more queues.
"Get All Queues" on page 179	Get all queues for which the user has permissions. If the user making the request is an administrator, all queues are returned.
"Reorder a Queued Item" on page 179	Reorder a queued item.
"Bulk Reorder a Queued Item" on page 181	Reorder a list of queued items.
"Check Current Status of Execution Statistics" on page 182	Get execution queue statistics.
"Delete an Element from a Managed Queue" on page 183	Delete an element from a managed queue.

Tasks

Endpoint	Description
“List User Tasks” on page 183	List tasks associated with a particular user.
“Get Task Details” on page 184	Get the details of a task.
“Submit a Task” on page 186	Submit a task.
“Search Tasks” on page 187	Search for a specific task.

Work Orders

Endpoint	Description
“Get Work Orders for a Workflow” on page 188	Get work orders associated with a workflow.
“Initiate a Work Order” on page 190	Initiate a new work order for a given workflow.
“Check the Status of a Specific Work Order” on page 192	Poll the status of a specific work order, identified by its work order ID.
“Get the Output of a Work Order” on page 193	Get the output of a work order, for example, step names and step variable names with the associated variable values.
“Restart a Work Order” on page 196	Reset a specific work order, identified by its work order ID. The result includes both the overall status for the work order as well as the individual status of each of its steps.
“Cancel a Work Order” on page 200	Reset a specific work order, identified by its work order ID.
“Get Work Order Statistics” on page 201	Retrieve statistics about work order performance.
“Check the Status of a Specific Workflow Step” on page 202	Poll the status of a specific workflow step in a workflow, identified by its work order ID and step name.
“Restart One or More Work Orders from a Specific Step” on page 198	Restart a work order from a specific step, or restart multiple work orders from a specific step in each work order.
“Cancel a Work Step” on page 204	Cancel a specific step in the workflow.
“Get Work Step Statistics” on page 205	Retrieve statistics about workflow step performance.

Workflows

Endpoint	Description
"Get Available Workflows on the System" on page 206	Get all workflows in Orchestrator and specifies whether each workflow is in a published state or not. Only a published workflow can be initiated as a work order.
"Import a Workflow" on page 209	Import a workflow. You must first export the workflow from the source instance of Orchestrator; see "Export a Workflow" on page 207 for more information.
"Import a Workflow with Constraints" on page 210	Import a workflow with its constraints—sub-workflows and acton (plugin) templates. This process involves 2 requests: 1. Make a GET request to `/find_constraints`. 2. Make a POST request to `/import_with_constraints` with a request body that includes the constraints returned from the first request. Before importing the workflow, you must first export the workflow from the source instance of Orchestrator; see "Export a Workflow" on page 207 for more information.
"Export a Workflow" on page 207	Export a workflow. When you pair this method with "Import a Workflow" on page 209 (for development and production machines), you complete a direct deployment of the workflow from one Orchestrator instance to another.
"Get all Workflow Steps that Use the Same Plugin" on page 212	Get all workflow steps—across all workflows—that use the same plugin.
"Get Inputs Specification for a Workflow" on page 214	Get the run-time parameters for a workflow, along with a flag indicating whether the parameter is required or optional.
"Get the Output Specification of a Workflow" on page 215	Get the output specification of a workflow, for example, step names and step variable names without the variable values.
"Check the Running Status for all Workflows" on page 215	Indicate which workflows have work orders that are running and which workflows do not.
"Check the Running Status for a Specific Workflow" on page 216	Indicate whether or not a specific workflow has work orders that are currently running.
"Check the Detailed Running Status for a Specific Workflow" on page 216	Indicate—for a given workflow—the count of work orders corresponding to each execution status.
"Run Test Cases for Workflow Steps" on page 217	Run the test cases for steps in a workflow.
"Get the Status of Executed Test Cases for Workflow Steps" on page 218	Get the status of test cases for steps in a workflow.
"Publish a Workflow" on page 219	Publish a workflow so it can launch running instances (work orders).

Endpoint	Description
“Get a List of All Workflows that Use a Specific Workflow as a Subworkflow” on page 220	Get a list of all workflows that use the specified workflow as a subworkflow.

Resources

Endpoint	Description
“Add a Resource” on page 220	Add a resource.
“Edit Resource Capacity” on page 221	Edit resource capacity.
“Bring a Resource Online” on page 221	Bring a resource online.
“Take a Resource Offline” on page 222	Take a resource offline.
“Pool Status” on page 222	Get the status of a resource pool—a group of resources, such as IP addresses and mongrels—so you can track resource usage data across workflows.
“Edit Pool Capacity” on page 223	Edit pool capacity.

API Endpoints - Application Management

Orchestrator Background Processes Status

Description:

A method to poll the status of Orchestrator processes. If available, the PID for each process is returned.

Usage:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/processes_status/0?
login=admin&password=aspera
```

Response example (XML):

```
<?xml version="1.0"?>
  <processes time="2012-01-25 08:23:10 UTC" action="status" id="0">
  <process class="engine" pid="9825" status="running"/>
  <process class="mongrel" pid="1337" status="running" port="3000"/>
  <process class="mongrel" pid="1340" status="running" port="3001"/>
  <process class="mongrel" pid="1343" status="running" port="3002"/>
  <process class="monitor" pid="96279" status="running"/>
  <process class="worker" id="0" pid="13752" status="running" type="asynchronous"/>
  <process class="worker" id="1" pid="9863" status="running" type="normal"/>
  <process class="worker" id="2" pid="9865" status="running" type="normal"/>
</processes>
```

Control Process

Description:

This is a method for external applications to control Orchestrator processes such as Engine, Monitor, Mongrel, or Worker.

Request for Engine and Monitor

```
http://Orchestrator_IP_address/aspera/orchestrator/api/control_process/<process_name>.xml?
login=admin&password=pass&command=<command>
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/control_process/engine.xml?
login=admin&password=pass&command='kill'
```

The following are the commands for each process:

Process	Available Commands
Engine	`kill`, `start`, `stop`
Monitor	`kill`, `start`, `stop`
Mongrel	`kill`, `start` `unlock_port`
Worker	`kill`, `start`, `stop`

Note: If a control process command is executed, the API call is successful; however it can take few moments to to take effect.

Usage for mongrel:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/control_process/process_name.xml?
login=admin&password=pass&command=command&port=port
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/control_process/mongrel.json?
login=admin&password=admin&command=kill&port=3002
```

Usage for worker:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/control_process/<process_name>.xml?
login=admin&password=pass&command=<command>&worker_id=<worker_id>
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/control_process/worker.xml?
login=admin&password=pass&command=kill&worker_id=4
```

Response example for engine start (XML):

```
<processes time="2015-12-03 10:29:19 UTC" action="status" engine_id="0">
<process class="engine" pid="83296" status="running"/>
<process class="mongrel" pid="83301" status="running" port="3000"/>
<process class="mongrel" pid="83303" status="running" port="3001"/>
<process class="mongrel" pid="83305" status="running" port="3002"/>
<process class="monitor" pid="83299" status="running"/>
<process class="worker" id="0" pid="83312" status="running" type="asynchronous"/>
<process class="worker" id="1" pid="83314" status="running" type="asynchronous"/>
<process class="worker" id="2" pid="83316" status="running" type="normal"/>
<process class="worker" id="3" pid="83318" status="running" type="normal"/>
<process class="worker" id="4" pid="83320" status="running" type="normal"/>
<process class="worker" id="5" pid="83310" status="running" type="normal"/>
</processes>
```

Response example for mongrel kill on port 3002 (XML):

```
{
   "processes":{
      "process":[
         {
            "pid":25907,
            "class":"engine",
            "status":"running"
         },
         {
            "pid":25912,
            "port":3000,
            "class":"mongrel",
            "status":"running"
         },
         {
            "pid":25914,
            "port":3001,
            "class":"mongrel",
            "status":"running"
         },
         {
            "pid":"-",
            "port":3002,
            "class":"mongrel",
            "status":"not running"
         },
         {
            "pid":25910,
            "class":"monitor",
            "status":"running"
         },
         {
            "id":"0",
            "type":"asynchronous",
            "pid":25923,
            "class":"worker",
            "status":"running"
         },
         {
            "id":"1",
            "type":"asynchronous",
            "pid":25925,
            "class":"worker",
            "status":"running"
         },
         {
            "id":"2",
            "type":"normal",
            "pid":25927,
            "class":"worker",
            "status":"running"
         },
         {
            "id":"3",
            "type":"normal",
            "pid":25929,
            "class":"worker",
            "status":"running"
         },
         {
            "id":"4",
            "type":"normal",
            "pid":25931,
            "class":"worker",
            "status":"running"
         },
         {
            "id":"5",
            "type":"normal",
            "pid":25921,
            "class":"worker",
            "status":"running"
         }
      ],
      "engine_id":0,
      "action":"status",
      "time":"2016-01-12 23:34:03"
   }
}
```

Orchestrator Monitor

Description:

This method polls the status of monitors for workflows, folders and scripts.

Usage:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/monitor_snapshot/all?
logon=admin&password=aspera
```

This returns the status for all the monitors.

A more restrictive view can be obtained with any of the following:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/monitor_snapshot/workflows?
logon=admin&password=aspera
        http://Orchestrator_IP_address/aspera/orchestrator/api/monitor_snapshot/folders?
logon=admin&password=aspera
        http://Orchestrator_IP_address/aspera/orchestrator/api/monitor_snapshot/scripts?
logon=admin&password=aspera
```

Response example (XML):

```
<?xml version="1.0"?>
<monitor time="2012-05-10 05:47:53 UTC" action="snapshot" id="all">
    <monitor_workflows>
        <monitor_workflow id="1" workflow_id="1" workflow_name="ENG Contributions"
active="true" >
          <status>No Issues</status>
            <status_report>In Progress=>1</status_report>
            <last_activity>2012-05-09 23:40:15</last_activity>
            <last_polled>2012-05-10 05:47:51 UTC</last_polled>
        </monitor_workflow>
        <monitor_workflow id="2" workflow_id="5" workflow_name="Non ENG Contributions"
active="false" >
          <status>Monitoring Off</status>
        </monitor_workflow>
        <monitor_workflow id="3" workflow_id="14" workflow_name="Livex OnDemand distribution"
active="false" >
          <status>Monitoring Off</status>
        </monitor_workflow>
    </monitor_workflows>
    <monitor_folders>
        <monitor_folder id="1" folder="temporary folder" path="/tmp" active="false">
          <status>Monitoring Off</status>
        </monitor_folder>
        <monitor_folder id="2" folder="Other tmp" path="/tmp" active="true">
          <status>Alerts</status>
            <status_report>Aging entries (> 2 min)</status_report>
            <content>7 files, 1 folders</content>
            <age_range>3353 - 36065 min.</age_range>
            <last_polled>2012-05-10 05:46:59 UTC</last_polled>
        </monitor_folder>
    </monitor_folders>
    <monitor_scripts>
        <monitor_script id="1" script="warn" node="node_IP_address" path="/opt/aspera/scripts/
warn.sh"  active="false">
          <status>Monitoring Off</status>
        </monitor_script>
        <monitor_script id="2" script="chill" node="node_IP_address" path="/opt/aspera/scripts/
ok.sh"  active="true">
          <status>No Issues</status>
            <status_report>All is good</status_report>
            <last_polled>2012-05-10 05:47:52 UTC</last_polled>
        </monitor_script>
    </monitor_scripts>
</monitor>
```

Ping a Remote Orchestrator Instance

Description:

This is a method to fetch the Orchestrator version, operating system information and licensing information.

This method may be used in a Federated workflow to check a remote Orchestrator status before remote/ delegated requests are sent to it.

Usage:

```
http://<orchestrator-IP-address>/aspera/orchestrator/api/remote_node_ping/
```

Response example (XML):

```
<?xml version="1.0"?>
<remote_orchestrator_info>
    <orchestrator-version>2.1.0.92065-00008</orchestrator-version>
    <platform>x86_64-pc-linux-gnu</platform>
    <engine_id>0</engine_id>
    <license-info>
    <licensee>AsperaColo</licensee>
    <can_execute>true</can_execute>
    <max_workflows>0</max_workflows>
    <comments>Orchestrator Server</comments>
    <max_processes>0</max_processes>
    <can_design>true</can_design>
    <mac-address>any</mac-address>
    </license-info>
</remote_orchestrator_info>
```

Response example (JSON):

```
{
   "remote_orchestrator_info":{
      "orchestrator-version":"2.3.5.xxxxx-00003",
      "license-info":{
         "mac-address":"any, any",
         "licensee":"development, development",
         "max_workflows":0,
         "can_execute":true,
         "can_design":true,
         "max_processes":0,
         "comments":"Permanent dev license - generated by Marc, Permanent dev license -
generated by Marc"
      },
      "engine_id":0,
      "platform":"i686-apple-darwin14.1.0"
   }
}
```

Note: This method does not require any authentication parameters.

API Endpoints—General

Persist Custom Data in a Database

This method allows external applications to submit data to Orchestrator that is persisted in a database table called `shared_states`. For example, a plugin can send a Web notification upon transcode complete or failure.

Orchestrator uses basic authorization in the request.

Note: The default format for the response is XML format. To return a response in JSON format, add `format=json` to the request.

Note: You must use URL encoding for any special characters in parameter values.

Usage

```
http://Orchestrator_IP_address/aspera/orchestrator/api/custom_api_call?
login=user_name&identifier=identifier_valuejob_id=job_id
```

For example:

```
http://10.0.0.10/aspera/orchestrator/api/custom_api_call?
login=my_user_name&identifier=testing&uniq_id=321&job_id=123&job_result=success
```

To simplify searching for the data entered in the table, users can pass two parameters to make each entry uniquely identifiable. The fields are `identifier` which is a string, and `uniq_id`, which is an integer.

All key-value pairs (except for `login`, `password`, `identifier`,`uniq_id`) will be stored in a serialized hash in the `shared_states` table.

The database entries can be searched using a MySQL query or a Ruby query.

MySQL query:

```
select id, entry from shared_states where aggregate_type = “identifier” and aggregate_id =
uniq_id
```

Ruby query:

```
SharedState.find(:all, :conditions => [“aggregate_type = ? and aggregate_id = ?”, identifier,
uniq_id])
```

Query Parameters

identifier=*identifier*
Specifies a custom data identifier (type: string).

uniq_id=*another identifier*
Specifies a custom data unique identifier (type: integer).

data_name_X=*data_1_value*
Specifies some data to pass in, where `data_name_X` is an integer.

Response

```
<?xml version="1.0"?>
<api_call action="custom_api_call">
   <shared_state>
      <id>34</id>
      <creation_time>"2014-03-22 21:11:35 UTC"</creation_time>
   </shared_state>
</api_call>
```

Get the Dashboard

This method allows external applications to access the Orchestrator dashboard.

Any URL used to access the dashboard can be called externally using the inline login and password.

Note: The default format for the response is XML format. To return a response in JSON format, add `format=json` to the request.

Note: You must use URL encoding for any special characters in parameter values.

Orchestrator uses basic authorization in the request.

Usage

This is an example a call to access the first page in the dashboard:

```
http://Orchestrator_IP_address/aspera/orchestrator/frames/home/0?login=admin
```

To enable inline credentials for the dashboard, the following parameter must be set in `orchestrator.yml`:

```
inline_auth_allowed:true
```

The following example applies to direct-start pages (allowing the user to manually start a workflow via a customized HTML page from an external application):

```
http://Orchestrator_IP_address/aspera/orchestrator/work_orders/direct_start/1?login=admin
```

Get Alerts for a User

This API method retrieves user alerts.

Get user alerts based on a particular alert category. You also can filter for the number of alerts, only, by setting value of the parameter `count_only` as true.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

```
https://Orchestrator_IP_address/aspera/orchestrator/api/alerts/?login=admin&count_only=true
https://Orchestrator_IP_address/aspera/orchestrator/api/alerts/?
login=admin&count_only=true&category=user
https://Orchestrator_IP_address/aspera/orchestrator/api/alerts/?login=admin&category=user
```

Example response:

```
<?xml version="1.0" encoding="UTF-8"?>
<alerts type="array">
    <alert>
        <id type="integer">1</id>
        <category-id type="integer">2</category-id>
        <alert-header nil="true"/>
        <alert-description>User 'admin' logged off</alert-description>
        <created-at type="dateTime">2017-04-06T23:09:43Z</created-at>
        <updated-at type="dateTime">2017-04-06T23:09:43Z</updated-at>
    </alert>
    <alert>
        <id type="integer">2</id>
        <category-id type="integer">2</category-id>
        <alert-header nil="true"/>
        <alert-description>User 'admin' successfully logged in from IP: '127.0.0.1' using
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:45.0) Gecko/20100101 Firefox/45.0'</alert-
description>
        <created-at type="dateTime">2017-04-06T23:09:47Z</created-at>
        <updated-at type="dateTime">2017-04-06T23:09:47Z</updated-at>
    </alert>
    <alert>
        <id type="integer">3</id>
        <category-id type="integer">2</category-id>
        <alert-header nil="true"/>
        <alert-description>User 'admin' logged off</alert-description>
        <created-at type="dateTime">2017-04-06T23:09:51Z</created-at>
        <updated-at type="dateTime">2017-04-06T23:09:51Z</updated-at>
    </alert>
    <alert>
        <id type="integer">4</id>
        <category-id type="integer">2</category-id>
        <alert-header nil="true"/>
        <alert-description>User 'admin' successfully logged in from IP: '127.0.0.1' using
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:45.0) Gecko/20100101 Firefox/45.0'</alert-
description>
        <created-at type="dateTime">2017-04-07T17:06:07Z</created-at>
        <updated-at type="dateTime">2017-04-07T17:06:07Z</updated-at>
    </alert>
    <alert>
        <id type="integer">5</id>
        <category-id type="integer">2</category-id>
        <alert-header nil="true"/>
        <alert-description>User 'admin' successfully logged in from IP: '127.0.0.1' using
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:45.0) Gecko/20100101 Firefox/45.0'</alert-
description>
        <created-at type="dateTime">2017-04-10T04:10:06Z</created-at>
        <updated-at type="dateTime">2017-04-10T04:10:06Z</updated-at>
    </alert>
    <alert>
        <id type="integer">6</id>
        <category-id type="integer">2</category-id>
        <alert-header nil="true"/>
        <alert-description>User 'admin' successfully logged in from IP: '127.0.0.1' using
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:45.0) Gecko/20100101 Firefox/45.0'</alert-
description>
        <created-at type="dateTime">2017-04-10T23:14:09Z</created-at>
        <updated-at type="dateTime">2017-04-10T23:14:09Z</updated-at>
    </alert>
    <alert>
        <id type="integer">7</id>
        <category-id type="integer">2</category-id>
        <alert-header nil="true"/>
        <alert-description>User 'admin' successfully logged in from IP: '127.0.0.1' using
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:45.0) Gecko/20100101 Firefox/45.0'</alert-
description>
        <created-at type="dateTime">2017-04-10T23:55:44Z</created-at>
        <updated-at type="dateTime">2017-04-10T23:55:44Z</updated-at>
    </alert>
</alerts>
```

API Endpoints—Monitors

Activate Monitors

Description:

This API method activate monitors—for workflows, work orders, scripts, folders, and nodes.

The response shown on this page is in JSON format. To return the default response (XML format) omit the `&format=json` which is present in the request below.

Usage

```
https://Orchestrator_IP_address/aspera/orchestrator/api/activate_monitor?
format=json&logon=login_name&monitor_type=monitor_type&portable_id=portable_id
```

For example:

```
https://10.0.0.10/aspera/orchestrator/api/activate_monitor?
format=json&logon=my_login_name&monitor_type=workflow&portable_id=1
```

Parameters

You must include either the `id` parameter or `portable_id` parameter in the request.

Note: For calls to an Orchestrator instance on an external server, use the `portable_id` parameter, because this value remains constant. Don't use the `id` parameter, because a new ID is generated from the database with each new call, and this makes it difficult to track the origin of the monitor.

You can obtain either parameter value from the Orchestrator API or UI:

- API: Parse the value you need from the "List Monitors" on page 169 response.
- UI: Click **Monitors** to open the Monitors page, then click the appropriate tab for the type of monitor (Workflow, Work Order, Script, Folder, or Node) to display a list of all available monitors of that type.
  - The `id` parameter value is in the **ID** column, to the left of the monitor name.

- For the `portable_id` parameter, click the edit icon to the right of the parameter name to open a dialog of monitor details. The value is in the **Portable ID** field.

Parameter Name	**Required/Optional**	**Data Type**	**Definition**
`monitor_type`	Required	string	Type of monitor being deactivated. Allowed values: `workflow`, `work order`, `folder`, `script`, node
`id`	Required	string	ID generated by the database. This should only be used for calls within the same Orchestrator instance.
`portable_id`	Optional	string	Secondary ID generated by Orchestrator. Use this option when making a request to an external Orchestrator instance.

Response

```
{
  "action": "activate_monitor",
  "time": "2018-04-24 22:02:32",
  "id": "1",
  "description": "activated"
}
```

Deactivate Monitors

This API method allows you to deactivate monitors (for workflows, work orders, scripts, folders, and nodes).

The response shown on this page is in JSON format. To return the default response (XML format) omit the `&format=json` which is present in the request below.

Usage:

```
https://Orchestrator_IP_address/aspera/orchestrator/api/deactivate_monitor?
format=json&logon=login_name&monitor_type=monitor_type&portable_id=portable_id
```

Parameters

You must include either the `id` parameter or `portable_id` parameter in the request.

Note: For calls to an Orchestrator instance on an external server, use the `portable_id` parameter, because this value remains constant. Don't use the `id` parameter, because a new ID is generated from the database with each new call, and this makes it difficult to track the origin of the monitor.

You can obtain either parameter value from the Orchestrator API or UI:

- API: Parse the value you need from the "List Monitors" on page 169 response.
- UI: Click **Monitors** to open the Monitors page, then click the appropriate tab for the type of monitor (Workflow, Work Order, Script, Folder, or Node) to display a list of all available monitors of that type.

- The `id` parameter value is in the **ID** column, to the left of the monitor name.
- For the `portable_id` parameter, click the edit icon to the right of the parameter name to open a dialog of monitor details. The value is in the **Portable ID** field.

Parameter Name	Required/Optional	Data Type	Definition
`monitor_type`	Required	string	Type of monitor being deactivated. Allowed values: `workflow`, `work order`, `folder`, `script`, node
`id`	Required	string	ID generated by the database. This should only be used for calls within the same Orchestrator instance.
`portable_id`	Optional	string	Secondary ID generated by Orchestrator. Use this option when making a request to an external Orchestrator instance.

Response Example

```
{
  "action": "deactivate_monitor",
  "time": "2018-04-24 22:02:32",
  "id": "1",
  "description": "de-activated"
}
```

List Monitors

This API method returns a list of all monitors of a particular type (workflow, work order, script, folder, or node) that are available in Orchestrator.

Usage

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

```
https://Orchestrator_IP_address/aspera/orchestrator/api/list_monitors?monitor_type=monitor_type
```

For example:

```
http://localhost:3000/aspera/orchestrator/api/list_monitors?monitor_type=workflow
```

Parameters

Parameter Name	Required/Optional	Data Type	Definition
`monitor_type`	Required	boolean	Type of monitor to list. Allowed values: `workflow`, `work_order`,

Parameter Name	Required/Optional	Data Type	Definition
			folder, script, node

Response

Response (XML Format):

```
<?xml version="1.0" encoding="UTF-8"?>
<monitor-workflows type="array">
    <monitor-workflow>
        <id type="integer">1</id>
        <state type="boolean">false</state>
        <workflow-id type="integer">1</workflow-id>
        <workflow-name>Success/Fail/Error</workflow-name>
        <reporting-format nil="true"/>
        <must-run type="boolean">false</must-run>
        <created-at type="dateTime">2019-08-14T19:03:42Z</created-at>
        <updated-at type="dateTime">2019-08-14T20:33:12Z</updated-at>
        <cached-status>--- {}
</cached-status>
        <cached-at type="dateTime">2019-08-14T20:33:11Z</cached-at>
        <refresh-rate type="integer" nil="true"/>
        <auto-start type="boolean">false</auto-start>
        <auto-start-parameters>{"work_order_name"=&gt;"", "priority"=&gt;""}</auto-start-
parameters>
        <alert-processing></alert-processing>
        <warning-processing></warning-processing>
        <ignore-duplicate type="boolean">true</ignore-duplicate>
        <cached-notification nil="true"/>
        <notification-workflow-id type="integer">0</notification-workflow-id>
        <notification-sent-at type="dateTime" nil="true"/>
        <tags type="array"/>
        <monitor-group-id type="integer" nil="true"/>
        <trigger-on-new-files type="boolean">false</trigger-on-new-files>
        <complete-notification-workflow-id type="integer" nil="true"/>
        <failure-notification-workflow-id type="integer" nil="true"/>
        <error-notification-workflow-id type="integer" nil="true"/>
        <is-runtime-config type="boolean" nil="true"/>
        <last-step-needed type="boolean" nil="true"/>
        <cancel-notification-workflow-id type="integer" nil="true"/>
        <max-running type="integer" nil="true"/>
        <all-outputs-in-json type="boolean">false</all-outputs-in-json>
        <all-outputs type="boolean">false</all-outputs>
        <portable-id>204ba3c0-a0f4-0137-f9e7-08e9fe6bebcd</portable-id>
    </monitor-workflow>
</monitor-workflows>
```

API Endpoints—Plugins and Portlets

Plugin Version

This method gets the installed version of a single plugin or all plugins.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

All plugins:

```
https://Orchestrator_IP_address/aspera/orchestrator/api/plugin_version?login=admin
```

Response example:

```
<PluginVersion>
<Plugin>
<version>1.1.6</version>
<name>AscpClient</name>
</Plugin>
<Plugin>
<version>2.6.0</version>
<name>AsperaCentralWatcher</name>
</Plugin>
<Plugin>
<version>0.0.3</version>
<name>AsperaNodeApi</name>
</Plugin>
<Plugin>
<version>0.0.6</version>
<name>AsperaNodeFileWatcher</name>
</Plugin>
<Plugin>
<version>0.1.0</version>
<name>AsperaNodeSearch</name>
</Plugin>
<Plugin>
<version>0.4.1</version>
<name>ConsoleNotification</name>
</Plugin>
<Plugin>
<version>0.2.4</version>
<name>CustomRuby</name>
</Plugin>
<Plugin>
<version>0.3.0</version>
<name>FaspControl</name>
</Plugin>
<Plugin>
<version>2.5.5</version>
<name>FaspTransfer</name>
</Plugin>
<Plugin>
<version>0.5.1</version>
<name>FaspexDelivery</name>
</Plugin>
<Plugin>
<version>1.2.3</version>
<name>FaspexInboxWatcher</name>
</Plugin>
<Plugin>
<version>0.2.7</version>
<name>Filter</name>
</Plugin>
</PluginVersion>
```

Single plugin:

To request the version of a single plugin, add the name of the plugin as a query parameter.

```
https://Orchestrator_IP_Address/aspera/orchestrator/api/plugin_version/AscpClient?login=admin
```

In this example, `AscpClient` is the name of the plugin whose version will be returned.

Response example

```
<PluginVersion>
<Plugin>
<name>AscpClient</name>
<version>1.1.6</version>
</Plugin>
</PluginVersion>
```

Portlet Version

This method gets the installed version of a single portlet or all portlets.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

All Portlets:

```
https://Orchestrator_IP_address/aspera/orchestrator/api/portlet_version?login=admin
```

Response example:

```
<PortletVersion>
<Portlet>
<version>0.0.1</version>
<name>Return records</name>
</Portlet>
<Portlet>
<version>0.0.2</version>
<name>Connect Upload</name>
</Portlet>
<Portlet>
<version>0.0.1</version>
<name>CSV Editor</name>
</Portlet>
<Portlet>
<version>0.0.1</version>
<name>File Browser</name>
</Portlet>
<Portlet>
<version>0.0.1</version>
<name>Filter Work Orders</name>
</Portlet>
<Portlet>
<version>0.0.1</version>
<name>Journal View</name>
</Portlet>
<Portlet>
<version>0.0.2</version>
<name>Managed Load Utilization Monitor</name>
</Portlet>
<Portlet>
<version>0.0.4</version>
<name>Monitors Dashboard</name>
</Portlet>
<Portlet>
<version>0.0.1</version>
<name>Orchestrator Information</name>
</Portlet>
<Portlet>
<version>0.0.3</version>
<name>Orchestrator Processes</name>
</Portlet>
<Portlet>
<version>0.0.5</version>
<name>Queue Manager</name>
</Portlet>
<Portlet>
<version>0.0.5</version>
<name>Managed Resource Utilization Monitor</name>
</Portlet>
<Portlet>
<version>0.0.2</version>
<name>Workflow Step Activity</name>
</Portlet>
<Portlet>
<version>0.0.1</version>
<name>Step Performance Statistics</name>
</Portlet>
<version>0.0.8</version>
<name>Workflow Step Statistics</name>
</Portlet>
<Portlet>
<version>0.0.3</version>
<name>User pending tasks</name>
</Portlet>
<Portlet>
<version>0.0.2</version>
<name>Virtual Catcher Portlet</name>
</Portlet>
<Portlet>
<version>0.0.1</version>
<name>Virtual Catcher Status</name>
```

```
</Portlet>
<Portlet>
<version>0.0.4</version>
<name>Workflow Statistics</name>
</Portlet>
<Portlet>
<version>0.0.2</version>
<name>Workflow Performance Statistics</name>
</Portlet>
<Portlet>
<version>0.0.1</version>
<name>Group assigned tasks</name>
</Portlet>
<Portlet>
<version>0.0.2</version>
<name>Custom Portlet</name>
</Portlet>
<Portlet>
<version>0.0.1</version>
<name>Dalet Import</name>
</Portlet>
<Portlet>
<version>0.0.1</version>
<name>WWE Delivery Status</name>
</Portlet>
<Portlet>
<version>0.0.1</version>
<name>France TV Journals</name>
</Portlet>
</PortletVersion>
```

Single Portlet:

To request the version of a single plugin, add the name of the plugin as a query parameter.

```
https://Orchestrator_IP_address/aspera/orchestrator/api/portlet_version/demoPortlet?login=admin
```

In this example, `demoPortlet` is the name of the portlet for which you are requesting the version.

Response example:

```
<PortletVersion>
<Portlet>
<name>Delivery Status</name>
<version>0.0.1</version>
</Portlet>
</PortletVersion>
```

Reload One Plugin or All Plugins

This endpoint allows you to reload a plugin, which prompts Orchestrator to update the code for that plugin. If you do not specify a plugin in the request, Orchestrator reloads all plugins.

The Orchestrator API uses basic authorization for requests.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Reload all plugins

Note: You must use URL encoding for any special characters in parameter values, including username and password.

Usage (GET method):

```
https://Orchestrator_IP_address/aspera/orchestrator/api/reload_plugin
```

Response example:

```
<ReloadStatus>
  <AllPlugins>
    <Port_3000>true</Port_3000>
    <Port_3001>true</Port_3001>
    <Port_3002>true</Port_3002>
  </AllPlugins>
</ReloadStatus>
```

Reload a Specific Plugin

Usage (GET method):

```
https://Orchestrator_IP_address/aspera/orchestrator/api/reload_plugin?action_type=plugin_name
```

For example:

```
https://Orchestrator_IP_address/aspera/orchestrator/api/reload_plugin?action_type=FaspTransfer
```

Parameters

Parameter Name	Required	Data Type	Definition
action_type	false	string	Name of the plugin. While this parameter is optional for the GET method, you must enter a plugin name in order to reload a specific plugin. **Note:** The plugin name must be entered in camel case.

Response example:

```
<ReloadStatus>
  <FaspTransfer>
    <Port_3000>true</Port_3000>
    <Port_3001>true</Port_3001>
    <Port_3002>true</Port_3002>
  </FaspTransfer>
</ReloadStatus>
```

Reload Plugin Set

This API method reloads a defined set of plugins, which prompts Orchestrator to update the code for those plugins.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage (POST method):

Note: You must use URL encoding for any special characters in parameter values, including username and password.

```
https://Orchestrator_IP_address/aspera/orchestrator/api/reload_plugin_set
```

Submit a request body containing the names of plugins to reload (more than one and less than all).

Note: Each plugin name must be entered in camel case.

Request body:

```
<plugin_set>
  <action_type>PluginName1</action_type>
  <action_type>PluginName2</action_type>
  <action_type>PluginName3</action_type>
</plugin_set>
```

For example:

```
<plugin_set>
  <action_type>AmazonS3Operation</action_type>
  <action_type>AdiTransformation</action_type>
  <action_type>AkamaiTranscoding</action_type>
</plugin_set>
```

Response

```
<ReloadStatus>
    <AmazonS3Operation>
        <Port_3000>true</Port_3000>
        <Port_3001>true</Port_3001>
        <Port_3002>true</Port_3002>
    </AmazonS3Operation>
    <AdiTransformation>
        <Port_3000>true</Port_3000>
        <Port_3001>true</Port_3001>
        <Port_3002>true</Port_3002>
    </AdiTransformation>
    <AkamaiTranscoding>
        <Port_3000>true</Port_3000>
        <Port_3001>true</Port_3001>
        <Port_3002>true</Port_3002>
    </AkamaiTranscoding>
</ReloadStatus>
```

Enable One Plugin or All Plugins

If you specify a plugin name in the request, the endpoint enables the plugin so that it can be used in a workflow. If you do not specify a plugin name, the endpoint enables all Orchestrator plugins.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Request Example - Reload a Specific Plugin

```
GET http://localhost:3000/aspera/orchestrator/api/enable_plugin?action_type=BatonFileCorrection
```

Parameters

Parameter Name	Required/Optional	Data Type	Definition
action_type	optional	string	Name of the plugin you need to enable. **Note:** If you do not specify a plugin name, the request enables all Orchestrator plugins.

Response Example

The response returns the ID of the plugin that is enabled:

```
<EnablePlugin>
    <InstallID>732</InstallID>
</EnablePlugin>
```

Import a Plugin

This endpoint imports a plugin into your Orchestrator instance.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Request Example

```
GET http://localhost:3000/aspera/orchestrator/api/import_plugin?file=Adi3Parser_0.1.9.plugin
```

Parameters

Parameter Name	Required/Optional	Data Type	Definition
file	required	string	The plugin filename for the plugin you need to import.

Response Example

The response confirms the plugin name (action type) that was imported.

```
<ImportPlugin>
   <Action_Type>Adi3Parser</Action_Type>
 </ImportPlugin>
```

API Endpoints—Queues

Queue Item

This API method adds an item to a queue.

Required endpoints are `queue_name` and `queued_item`. Optional parameters are `description`, `priority`, and `weight`.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

Example request:

```
https://Orchestrator_IP_address/aspera/orchestrator/api/queue_item/BulkReOrderTest.xml?
login=admin&queued_item=apitest1111&description=mydescription&priority =120&weight=123
```

Example response:

```
<Queued-item>
    <item>apitest1111</item>
    <weight type="integer">123</weight>
    <rank type="integer">12</rank>
    <id type="integer">34</id>
    <priority type="integer">120</priority>
    <description>mydescription</description>
</Queued-item>
```

Get Queued Items From a Queue

This method gets a list of queued items from a queue.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

The required input parameter is `queue_id=`*`queue_id`*.

Request:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/queued_items/queue_id.format?login=admin
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/queued_items/transcode.xml?login=admin
```

Example response:

```
<?xml version="1.0"?>
<Queue>
  <Queued_items>
    <Queued_item>
      <priority>100</priority>
      <weight>3859</weight>
      <queued_item>/demo/to_transcoder/3_10_to_Yuma.avi</queued_item>
      <id>587</id>
      <updated_at>Tue Jan 06 21:38:51 UTC 2015</updated_at>
      <position>50</position>
      <originator_type></originator_type>
      <originator_id>64623</originator_id>
      <item_description>3_10_to_Yuma</item_description>
      <rank>0</rank>
      <created_at>Tue Jan 06 19:29:01 UTC 2015</created_at>
    </Queued_item>
    <Queued_item>
      <priority>100</priority>
      <weight>12823</weight>
      <queued_item>/demo/to_transcoder/39steps_train.avi</queued_item>
      <id>586</id>
      <updated_at>Tue Jan 06 19:28:54 UTC 2015</updated_at>
      <position>100</position>
      <originator_type></originator_type>
      <originator_id>64621</originator_id>
      <item_description>39steps_train</item_description>
      <rank>1</rank>
      <created_at>Tue Jan 06 19:28:54 UTC 2015</created_at>
    </Queued_item>
  </Queued_items>
  <Queue_metadata>
    <comments>ad_hoc creation for QueueStager #4</comments>
    <administrators>
      <group>1</group>
      <user>3</user>
    </administrators>
    <id>transcode</id>
    <paused>true</paused>
    <name>transcode</name>
 </Queue_metadata>
</Queue>
```

Look Up an Item in a Queue

This API method searches for an item in one or more queues.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage: Search for an Item in a Specific Queue

This method allows users to make an API call to obtain queue items using an item description; it takes `queue_desc` as a search parameter. The `queue_name` that is provided in the URL must be a string match to the queued item in the database.

Request:

```
https://Orchestrator_IP_address/aspera/orchestrator/api/lookup_queued_item?
queue_name=test_queue&queue_desc=testdesc
```

Response example:

```
<Queue>
   <Queued_items>
    <Queued_item>
     <position>250</position>
     <rank>1</rank>
     <updated_at>Mon Jan 09 22:14:52 UTC 2017</updated_at>
     <originator_id>69296</originator_id>
     <id>187</id>
     <created_at>Mon Jan 09 22:14:52 UTC 2017</created_at>
     <priority>100</priority>
     <originator_type/>
     <item_description>this is a test description</item_description>
     <queued_item>test_desc</queued_item>
     <weight>0</weight>
    </Queued_item>
  </Queued_items>
  <Queue_metadata>
   <administrators></administrators>
   <comments/>
   <paused>true</paused>
   <name>new queue 2</name>
  </Queue_metadata>
</Queue>
```

This method also takes the parameters `queue_name` and `queued item`, as in the following example.

Usage: Search by a Queued Item ID

The value of the queued item ID can be obtained from the Fetch Queued Items from a Queue API method.

Request

```
:http://Orchestrator_IP_address/aspera/orchestrator/api/lookup_queued_item?
format=xml&queued_item_id=29&login=admin
```

Response example:

```
<Queue>
    <Queue_metadata>
        <comments>ad_hoc creation for QueueStager #21</comments>
        <administrators>
            <group>1</group>
            <group>2</group>
            <group>4</group>
            <group>5</group>
            <group>6</group>
        </administrators>
        <name>Dev_QC_Action</name>
        <paused>false</paused>
    </Queue_metadata>
    <Queued_items>
        <Queued_item>
            <originator_type/>
            <created_at>Fri Dec 04 08:23:49 UTC 2015</created_at>
            <position>110</position>
            <rank>1</rank>
            <updated_at>Fri Dec 04 08:23:49 UTC 2015</updated_at>
            <originator_id/>
            <priority>100</priority>
            <weight>0</weight>
            <queued_item>test_add1</queued_item>
            <item_description/>
            <id>29</id>
        </Queued_item>
    </Queued_items>
</Queue>
```

Get All Queues

This API call returns all queues in Orchestrator for which the user has permissions. If the user making the API call is an administrator, all queues are displayed.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

Request:

```
https://Orchestrator_IP_address/aspera/orchestrator/api/queues_list?login=admin
```

Response:

```
<Queues_List>
    <Queues>
        <queue>
            <name>a1_2</name>
            <update_at>2016-03-04 19:49:37 UTC</update_at>
            <displayed_name>new queue </displayed_name>
            <id>1</id>
            <created_at>2016-03-04 19:49:37 UTC</created_at>
        </queue>
        <queue>
            <name>queue2</name>
            <update_at>2016-07-21 22:50:28 UTC</update_at>
            <displayed_name>queue 2 </displayed_name>
            <id>2</id>
            <created_at>2016-07-21 22:50:28 UTC</created_at>
        </queue>
        <queue>
            <name>new_queue_2</name>
            <update_at>2016-07-28 21:21:22 UTC</update_at>
            <displayed_name>new queue 2 </displayed_name>
            <id>9</id>
            <created_at>2016-07-28 21:21:22 UTC</created_at>
        </queue>
        <queue>
            <name>lklk</name>
            <update_at>2016-09-20 23:34:06 UTC</update_at>
            <displayed_name>lklk</displayed_name>
            <id>186</id>
            <created_at>2016-09-20 23:34:06 UTC</created_at>
        </queue>
    </Queues>
</Queues_List>
```

Reorder a Queued Item

This endpoint reorders a queued item.

Four reorder operations are supported:

- up - move up by one rank
- down - move down by one rank
- `first` - move to the top of the queue
- `last` - move to the bottom of the queue

In the response, the rank field only changes if an operation was successful. A rank of `0` indicates the queued item is on top of the queue.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage: Search for an Item in a Specific Queue

The queued item provided in the URL must be a string match to the queued item in the database.

Request:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/reorder_queued_item?
queue_name=CP_ENT_FFMPEG&format=xml&queued_item=test&login=admin&operation=up
```

Sample response:

```
<Queue>
    <Queue_metadata>
        <comments>ad_hoc creation for QueueStager #37</comments>
        <administrators></administrators>
        <name>CP_ENT_FFMPEG</name>
        <paused>false</paused>
    </Queue_metadata>
    <Queued_items>
        <Queued_item>
            <updated_at>Wed Apr 29 22:06:50 UTC 2015</updated_at>
            <originator_type/>
            <id>26</id>
            <created_at>Wed Apr 29 22:06:50 UTC 2015</created_at>
            <queued_item>test</queued_item>
            <position>100</position>
            <rank>0</rank>
            <weight>0</weight>
            <originator_id>1217</originator_id>
            <item_description>test</item_description>
            <priority>100</priority>
        </Queued_item>
    </Queued_items>
</Queue>
```

Usage: Search by a Queued Item ID

The value of a queued item ID can be obtained from the "Get Queued Items From a Queue" on page 176 API method or the "Look Up an Item in a Queue" on page 177 API method.

Request

```
http://Orchestrator_IP_address/aspera/orchestrator/api/reorder_queued_item?
format=xml&queued_item_id=29&login=admin&operation=down
```

Response example:

```
<Queue>
    <Queue_metadata>
        <comments>ad_hoc creation for QueueStager #21</comments>
        <administrators>
            <group>1</group>
            <group>2</group>
            <group>4</group>
            <group>5</group>
            <group>6</group>
        </administrators>
        <name>Dev_QC_Action</name>
        <paused>false</paused>
    </Queue_metadata>
    <Queued_items>
        <Queued_item>
            <originator_type/>
            <created_at>Fri Dec 04 08:23:49 UTC 2015</created_at>
            <position>110</position>
            <rank>1</rank>
            <updated_at>Fri Dec 04 08:23:49 UTC 2015</updated_at>
            <originator_id/>
            <priority>100</priority>
            <weight>0</weight>
            <queued_item>test_add1</queued_item>
            <item_description/>
            <id>29</id>
        </Queued_item>
    </Queued_items>
</Queue>
```

Bulk Reorder a Queued Item

This endpoint reorders a list of queued items.

The endpoint supports four reorder operations:

- up (move up by one rank)
- down (move down by one rank)
- `first` (move to the top of the queue)
- `last` (move to the bottom of the queue)

In the API responses, only the rank field changes if an operation was successfully performed. Rank `0` indicates that the queued item is on top of the queue.

The only parameter is a comma-separated list of `queued_item_ids`.

Usage

Request:

```
https://Orchestrator_IP_address/aspera/orchestrator/api/bulk_reorder_queue?
queued_item_ids=18,19,20,21&format=xml&login=admin&operation=first
```

Example response:

```
<Queue>
    <queued_items>
        <queued_item>
            <queued_item>test1</queued_item>
            <updated_at>Thu Aug 18 02:30:43 UTC 2016</updated_at>
            <id>18</id>
            <rank>4</rank>
            <originator_type/>
            <item_description/>
            <position>-395</position>
            <created_at>Fri Jul 29 00:23:09 UTC 2016</created_at>
            <priority>100</priority>
            <weight>0</weight>
            <originator_id>68877</originator_id>
        </queued_item>
        <queue_data>
            <comments/>
            <administrators/>
            <name>new queue 2 </name>
            <paused>true</paused>
        </queue_data>
    </queued_items>
    <queued_items>
        <queued_item>
            <queued_item>test2</queued_item>
            <updated_at>Thu Aug 18 02:30:43 UTC 2016</updated_at>
            <id>19</id>
            <rank>3</rank>
            <originator_type/>
            <item_description/>
            <position>-400</position>
            <created_at>Fri Jul 29 00:23:15 UTC 2016</created_at>
            <priority>100</priority>
            <weight>0</weight>
            <originator_id>68878</originator_id>
        </queued_item>
        <queue_data>
            <comments/>
            <administrators/>
            <name>new queue 2 </name>
            <paused>true</paused>
        </queue_data>
    </queued_items>
    <queued_items>
```

```
        <queued_item>
            <queued_item>test3</queued_item>
            <updated_at>Thu Aug 18 02:30:43 UTC 2016</updated_at>
            <id>20</id>
            <rank>2</rank>
            <originator_type/>
            <item_description/>
            <position>-405</position>
            <created_at>Fri Jul 29 00:23:21 UTC 2016</created_at>
            <priority>100</priority>
            <weight>0</weight>
            <originator_id>68879</originator_id>
        </queued_item>
        <queue_data>
            <comments/>
            <administrators/>
            <name>new queue 2 </name>
            <paused>true</paused>
        </queue_data>
    </queued_items>
    <queued_items>
        <queued_item>
            <queued_item>test4</queued_item>
            <updated_at>Thu Aug 18 02:30:43 UTC 2016</updated_at>
            <id>21</id>
            <rank>1</rank>
            <originator_type/>
            <item_description/>
            <position>-410</position>
            <created_at>Fri Jul 29 00:23:28 UTC 2016</created_at>
            <priority>100</priority>
            <weight>0</weight>
            <originator_id>68880</originator_id>
        </queued_item>
        <queue_data>
            <comments/>
            <administrators/>
            <name>new queue 2 </name>
            <paused>true</paused>
        </queue_data>
    </queued_items>
</Queue>
```

Check Current Status of Execution Statistics

This endpoint gets execution queue statistics.

The response includes data that is currently presented in the Execution Statistics page in the Orchestrator UI.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

Request:

```
https://Orchestrator_IP_address/aspera/orchestrator/api/execution_queue_stats/?login=admin
```

Example response:

```
<?xml version="1.0"?>
<execution_queue_stats>
    <next_polling_due>0</next_polling_due>
    <completed_items>1</completed_items>
    <inprogress_synch_count>0</inprogress_synch_count>
    <inprogress_asynch_count>1</inprogress_asynch_count>
    <created>0</created>
    <ready>0</ready>
    <initializing>0</initializing>
    <initialized>0</initialized>
    <in_progress>0</in_progress>
    <paused>0</paused>
    <time_executed>2017-04-13 14:44:45</time_executed>
</execution_queue_stats>
```

Delete an Element from a Managed Queue

This endpoint deletes an element from a managed queue.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Request Example

```
GET http://localhost/aspera/orchestrator/api/delete_queued_item?
queue_name=test_queue&queued_item_id=21
```

Parameters

Parameter Name	Required/Optional	Data Type	Definition
queue_name	required	string	Name of the queue from which to remove the element
queued_item_id	required	integer	ID of the element to remove from the queue

Response Example

The response confirms the deletion of the element:

```
<DeleteQueuedItem>
    <Success>Item with id 21 was succesfully deleted.</Success>
</DeleteQueuedItem>
```

API Endpoints—Tasks

List User Tasks

This endpoint lists tasks associated with a particular user.

The required input parameter is `user`, which specifies the user whose tasks are to be listed.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

Request :

```
http://Orchestrator_IP_address/aspera/orchestrator/api/list_tasks/user.xml?login=admin
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/list_tasks/admin.xml?login=admin
```

Sample response:

```
<User_tasks>
  <task>
    <event>Create a new Endpoint</event>
    <requested_user_inputs>
      <value_type>hash</value_type>
      <name>Step_information</name>
      <required>false</required>
    </requested_user_inputs>
    <requested_user_inputs>
      <value_type>string</value_type>
      <name>Emails</name>
      <required>true</required>
    </requested_user_inputs>
    <requested_user_inputs>
      <value_type>string</value_type>
      <name>Login</name>
      <required>true</required>
    </requested_user_inputs>
    <requested_user_inputs>
      <value_type>string</value_type>
      <name>Name</name>
      <required>true</required>
    </requested_user_inputs>
    <userInput_id>2</userInput_id>
    <state_id>2626</state_id>
    <created_at>Fri Dec 04 03:38:23 UTC 2015</created_at>
    <updated_at>Fri Dec 04 03:38:23 UTC 2015</updated_at>
    <role_id>1</role_id>
    <status>Assigned</status>
    <completedBy></completedBy>
    <group>Administrator</group>
    <user_id></user_id>
    <id>1</id>
  </task>
  <task>
    <event>Create a new Endpoint</event>
    <requested_user_inputs>
      <value_type>hash</value_type>
      <name>Step_information</name>
      <required>false</required>
    </requested_user_inputs>
    <requested_user_inputs>
      <value_type>string</value_type>
      <name>Emails</name>
      <required>true</required>
    </requested_user_inputs>
    <requested_user_inputs>
      <value_type>string</value_type>
      <name>Login</name>
      <required>true</required>
    </requested_user_inputs>
    <requested_user_inputs>
      <value_type>string</value_type>
      <name>Name</name>
      <required>true</required>
    </requested_user_inputs>
    <userInput_id>2</userInput_id>
    <state_id>2626</state_id>
    <created_at>Fri Dec 04 03:38:23 UTC 2015</created_at>
    <updated_at>Fri Dec 04 03:38:23 UTC 2015</updated_at>
    <role_id></role_id>
    <status>Assigned</status>
    <completedBy></completedBy>
    <user>admin</user>
    <user_id>1</user_id>
    <id>2</id>
  </task>
  <user>admin</user>
</User_tasks>
```

Get Task Details

This endpoint gets the details of a task.

You filter the request with one of two path parameters, `task_id` or `workorder_id`.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Filter by Task ID

Request:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/task_details/task_id.format?login=admin
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/task_details /1.xml?login=admin
```

Sample response:

```
<?xml version="1.0" encoding="UTF-8"?>
<User_task>
   <task_inputs />
   <task>
      <userInput_id>2</userInput_id>
      <state_id>2626</state_id>
      <created_at>Fri Dec 04 03:38:23 UTC 2015</created_at>
      <updated_at>Fri Dec 04 03:38:23 UTC 2015</updated_at>
      <role_id>1</role_id>
      <status>Assigned</status>
      <completedBy />
      <user_id />
      <id>1</id>
   </task>
   <user_inputs>
      <type>hash</type>
      <value />
      <name>Step_information</name>
      <required>false</required>
   </user_inputs>
   <user_inputs>
      <type>string</type>
      <value />
      <name>Emails</name>
      <required>true</required>
   </user_inputs>
   <user_inputs>
      <type>string</type>
      <value />
      <name>Login</name>
      <required>true</required>
   </user_inputs>
   <user_inputs>
      <type>string</type>
      <value />
      <name>Name</name>
      <required>true</required>
   </user_inputs>
</User_task>
```

Identify by Work Order ID

When identifying a work order in a workflow with a single task, use the following:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/task_details/WOworkorder_id.format?
login=admin&password=admin
```

When identifying a work order in a workflow with multiple tasks, use the following:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/task_details/WO<workorder_id.format?
login=admin&password=admin&step_name=<step_name>
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/task_details/WO518.xml?
login=admin&password=admin&step_name= Request+approval
```

Sample response:

```
<User_task>
  <user_inputs>
    <required>false</required>
    <name>Step_information</name>
    <value></value>
    <type>hash</type>
  </user_inputs>
  <task_inputs>
    <Op>TestValue</Op>
  </task_inputs>
  <task>
    <created_at>Fri Jun 17 18:47:39 GMT 2016</created_at>
    <completedBy></completedBy>
    <updated_at>Fri Jun 17 18:47:39 GMT 2016</updated_at>
    <role_id>1</role_id>
    <id>3</id>
    <user_id></user_id>
    <userInput_id>5</userInput_id>
    <status>Assigned</status>
    <state_id>547</state_id>
  </task>
</User_task>
```

Submit a Task

This endpoint submits a task.

You can submit a task as one of these user roles:

- User
- Admin on behalf of a user, with the `input_provider` parameter:

The required path parameter is `task_id`.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Submitting as a User

Request:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/submit_task/task_id.format?
login=admin&outputs[Filename]=output_param
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/submit_task/1.xml?
login=admin&outputs[Filename]=tron.avi
```

Example response (XML):

```
<?xml version="1.0" encoding="UTF-8"?>
<User_task>
   <input_provider>admin</input_provider>
   <task>
      <userInput_id>2</userInput_id>
      <state_id>2626</state_id>
      <created_at>Fri Dec 04 03:38:23 UTC 2015</created_at>
      <updated_at>Fri Dec 04 06:01:55 UTC 2015</updated_at>
      <role_id>1</role_id>
      <status>Complete</status>
      <completedBy>1</completedBy>
      <user_id />
      <id>1</id>
   </task>
   <user_inputs>Filenametron.avi</user_inputs>
</User_task>
```

Submitting as an Admin on Behalf of a User

An admin can submit a task on behalf of a user by passing the `input_provider` parameter.

Request:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/submit_task/task_id.format?
login=admin&input_provider=<user>&outputs[Filename]=<output_param>
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/submit_task/55.xml?
login=admin&input_provider='lola'&outputs[Filename]=tron.avi
```

Example response (JSON):

```
{
   "input_provider":"lola",
   "task":{
      "active_assignment":{
         "userInput_id":2,
         "state_id":2626,
         "created_at":"2015-12-04T03:38:23Z",
         "updated_at":"2015-12-04T06:01:55Z",
         "role_id":1,
         "status":"Complete",
         "completedBy":1,
         "user_id":null,
         "id":1
      }
   },
   "user_inputs":{
      "Filename":"tron.avi"
    }
}
```

Search Tasks

This endpoint searches for a specific task in Orchestrator.

Note: Tasks are created with the User Input plugin.

Usage (GET method)

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Note: You must use URL encoding for any special characters in parameter values, including username and password.

```
https://Orchestrator_IP_address/aspera/orchestrator/api/search_tasks?
value=task_name&login=username
```

For example:

```
https://10.0.71.99/aspera/orchestrator/api/search_tasks?value=te&login=my_username
```

Parameters

Parameter Name	Required	Data Type	Definition
value	true	string	A search string to be matched against any data element entered in the configuration for the User Input plugin; for

Parameter Name	Required	Data Type	Definition
			example, the key for a user input.

Response

Note: If no tasks are available, the response is `nil`.

```
<SearchTasks>
    <anon>
        <id>1</id>
        <task>
            <id>1</id>
            <state_id>19586</state_id>
            <userInput_id>15</userInput_id>
            <role_id/>
            <user_id>1</user_id>
            <status>Assigned</status>
            <completedBy/>
            <created_at>2019-06-05 23:52:17 UTC</created_at>
            <updated_at>2019-06-05 23:52:17 UTC</updated_at>
        </task>
        <task_inputs/>
        <user_inputs>
            <name>Full name</name>
            <type>string</type>
            <value/>
            <required>false</required>
        </user_inputs>
        <user_inputs>
            <name>Orchestrator login</name>
            <type>string</type>
            <value/>
            <required>false</required>
        </user_inputs>
        <user_inputs>
            <name>Email address</name>
            <type>string</type>
            <value/>
            <required>false</required>
        </user_inputs>
    </anon>
</SearchTasks>
```

API Endpoints—Work Orders

Get Work Orders for a Workflow

This endpoint gets work orders associated with a workflow.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

Request:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_orders_list/workflow_id.xml?
login=admin&from_date=from_date&to_date=to_date&max_results=maximum_results
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_orders_list/8738.json?
login=admin&from_date='06-27-2015 02:40:59'&to_date='06-29-2015 02:40:59'&max_results=100
```

Required input parameter:

`workflow_id`
Specifies the workflow from which the work orders are listed.

Optional input parameters:

`from_date=`*`from_date`*
Specifies the start date for filtering data.

`to_date=`*`to_date`*
Specifies the end date for filtering data.

`max_results=`*`maximum_results`*
Specifies the maximum number of results to return.

priority
Filters by priority of the work order.

`status`
Filters by status of the work order.

Response example:

```
<?xml version="1.0" encoding="UTF-8"?>
<work-orders type="array">
  <work-order>
    <branchedFrom type="integer" nil="true"></branchedFrom>
    <branchedFromOrder type="integer" nil="true"></branchedFromOrder>
    <cleanup-after-days type="integer" nil="true"></cleanup-after-days>
    <comments nil="true"></comments>
    <completion-date type="datetime">2015-11-06T19:03:28Z</completion-date>
    <created-at type="datetime">2015-11-06T19:03:27Z</created-at>
    <derivedFrom type="integer" nil="true"></derivedFrom>
    <forkedAt type="integer" nil="true"></forkedAt>
    <higherPriority type="boolean" nil="true"></higherPriority>
    <id type="integer">491</id>
    <initiate-date type="datetime">2015-11-06T19:03:28Z</initiate-date>
    <initiatedBy type="integer">1</initiatedBy>
    <label nil="true"></label>
    <launched-by type="integer">1</launched-by>
    <master-id type="integer">491</master-id>
    <max-running type="integer">0</max-running>
    <monitor-id type="integer" nil="true"></monitor-id>
    <name>test</name>
    <priority type="integer" nil="true"></priority>
    <purge-after-days type="integer" nil="true"></purge-after-days>
    <running-as type="integer">2</running-as>
    <status>Complete</status>
    <statusDetails>WorkOrder ended at Fri Nov 06 13:03:28 CST 2015 with status: In Progress</
statusDetails>
    <tags></tags>
    <updated-at type="datetime">2015-11-06T19:03:28Z</updated-at>
    <workflowName>test</workflowName>
    <workflow-id type="integer">1</workflow-id>
    <workflow-revision-id type="integer">24</workflow-revision-id>
  </work-order>
  <work-order>
    <branchedFrom type="integer" nil="true"></branchedFrom>
    <branchedFromOrder type="integer" nil="true"></branchedFromOrder>
    <cleanup-after-days type="integer" nil="true"></cleanup-after-days>
    <comments nil="true"></comments>
    <completion-date type="datetime">2015-11-06T19:46:12Z</completion-date>
    <created-at type="datetime">2015-11-06T19:46:10Z</created-at>
    <derivedFrom type="integer" nil="true"></derivedFrom>
    <forkedAt type="integer" nil="true"></forkedAt>
    <higherPriority type="boolean" nil="true"></higherPriority>
    <id type="integer">492</id>
    <initiate-date type="datetime">2015-11-06T19:46:11Z</initiate-date>
    <initiatedBy type="integer">1</initiatedBy>
    <label nil="true"></label>
    <launched-by type="integer">1</launched-by>
    <master-id type="integer">492</master-id>
    <max-running type="integer">0</max-running>
    <monitor-id type="integer" nil="true"></monitor-id>
    <name>test</name>
    <priority type="integer" nil="true"></priority>
    <purge-after-days type="integer" nil="true"></purge-after-days>
    <running-as type="integer">2</running-as>
    <status>Complete</status>
    <statusDetails>WorkOrder ended at Fri Nov 06 13:46:12 CST 2015 with status: In Progress</
```

```
statusDetails>
    <tags></tags>
    <updated-at type="datetime">2015-11-06T19:46:12Z</updated-at>
    <workflowName>test</workflowName>
    <workflow-id type="integer">1</workflow-id>
    <workflow-revision-id type="integer">24</workflow-revision-id>
  </work-order>
</work-orders type="array">
```

Initiate a Work Order

This endpoint initiates a new work order for a given workflow.

If the call is successful, the response returns a work order ID, which you can use to track the execution status of that work order. Note that if this workflow has sub-workflows, they can't be monitored using the API, because only the parent work order ID is returned.

The request can be either *asynchronous* or *synchronous*. If *asynchronous*, it will return as soon as the work order is created. If *synchronous*, it will return after a specified step in the workflow is executed. The response can be in either XML or JSON format.

Note: The account specified in the login section of the URL must be part of the Orchestrator Operator group and the account must have run permission on the workflow.

If the workflow has run-time inputs, they can be specified in the URL with the following notation:

```
external_parameters[name_of run-time input-n]=value of run-time input-n
```

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Asynchronous Usage:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/initiate/xml?
login=admin&work_order[workflow_id]=workflow_id&external_parameters[name_of run-time
input1]=value of run-time input1&external_parameters[name_of
run-time input-n]=value of run-time input-n…
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/initiate/xml?
login=admin&work_order[workflow_id]=1
```

Required input parameter:

`work_order[workflow_id]=`*`integer`*
: Indicates that the work order is launched with a dedicated worker process.

Optional input parameters:

`work_order[higherPriority]=on`
: Specifies the workflow ID as can be obtained in Orchestrator workflow editor.

`preemptive_level=`*`integer`*
: Indicates a preemptive priority with the specified integer as the level of preemption (lowest goes first).

`work_order[priority]=`*`integer from 1 → low to 3 → high`*
: Specifies the priority level for that work order.

`work_order[name]=`*`string`*
: Specifies a name for the work order replacing the generated default name that is based on the workflow name.

external_parameters[*param_name*]=*string*

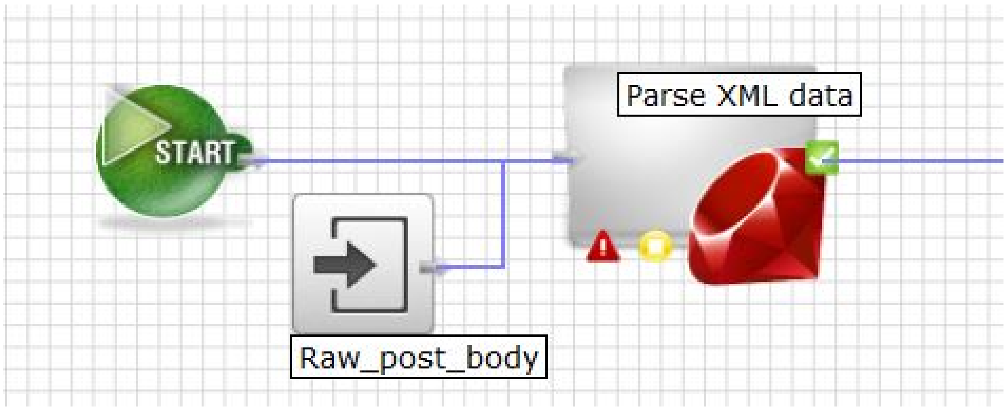
Specifies a name for a parameter to be passed as runtime input to the workflow. The name entered for *param_name* must match the name defined in the workflow as runtime input. Multiple external parameters can be passed, each one separated by the character "&".

Response example:

```
<?xml version="1.0"?>
<work_order time="2014-11-25 13:24:01 UTC" action="initiate" id="8464" workflow_id="1">
   <workorder_id>8464</workorder_id>
   <name>create file</name>
   <workflow id="1" revision_id="14">create file</workflow>
   <status state="Created">WorkOrder created at Tue Nov 25 07:24:01 CST 2014</status>
   <comments/>
   <parameters></parameters>
   <ownership>
      <initiated_by>admin</initiated_by>
      <running_as>system</running_as>
      <launched_by>admin</launched_by>
   </ownership>
   <priority dedicated_worker="false"/>
   <work_steps></work_steps>
</work_order>
```

Note: The error code `400 Bad Request` is returned if a mandatory input parameter is absent from the initiate work order request.

If an `HTTP POST` method is issued, the body can be any user-defined XML data. In this case, the workflow must expect the XML payload with a run-time input defined as `Raw_post_body` and the first step must be the one parsing the data. For example:



Synchronous Usage

This request creates a work order. A variable name can be specified to get an output variable value. Only the variable value is returned in the output, without the long-form XML wrapper around the variable.

```
http://Orchestrator_IP_address/aspera/orchestrator/api/initiate/xml?
login=admin&work_order[workflow_id]=workflow id&external_parameters[name_of
run-time input1]=value of run-time input1&external_parameters[name_of
run-time input-n]=value of run-time input-
n&synchronous=true&explicit_output_step=step_name&explicit_output_variable=variable_name
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/initiate/xml?
login=admin&work_order[workflow_id]=1&external_parameters[foo]=bar&synchronous=true&explicit_out
put_step=create xml content&explicit_output_variable=Generated_file_content
```

Sample value returned in the response:

```
hello
```

Check the Status of a Specific Work Order

This endpoint polls the status of a specific work order, identified by its work order ID.

The result includes both the overall status for the work-order as well as the individual status for all its composing steps. If the work order includes sub-workflows, only the sub-workflow status is displayed, not the status of the steps within the sub-workflow.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

Usage for a request with an XML response:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_order_status/workorder_id?
login=admin
```

Example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_order_status/3109?login=admin
```

The required input parameter is the following:

```
workflow_reporter/work_order_status/workorder_id
```

Below is the usage for a request with JSON response which includes all steps:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_order_status/workorder_id.json?
login=admin
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_order_status/344.json?login=admin
```

Usage for a request with XML response which includes all steps:

```
<?xml version="1.0"?>
<work_order time="2014-11-25 13:11:35 UTC" action="status" id="8461" master_id="8461">
   <name>tf1 demo duplicate</name>
   <workflow id="256" revision_id="5">tf1 demo duplicate</workflow>
   <status state="In Progress">Executing Step: user input (46755)</status>
   <comments/>
   <latest_update/>
   <ownership>
      <initiated_by>admin</initiated_by>
      <running_as>system</running_as>
      <launched_by>admin</launched_by>
   </ownership>
   <priority dedicated_worker="false">2</priority>
   <timestamps>
      <created_at>2014-11-24 16:48:37 UTC</created_at>
      <initiated_at>2014-11-24 16:48:38 UTC</initiated_at>
      <updated_at>2014-11-24 16:48:40 UTC</updated_at>
      <completed_at/>
   </timestamps>
   <work_steps>
      <work_step id="46756">
         <name>Create faspex package</name>
         <status state="In Progress" attempt="1">FASPEX Authorization received</status>
         <action type="FaspexDelivery" id="2"/>
         <created_at>2014-11-24 16:48:37 UTC</created_at>
         <initiated_at>2014-11-24 16:48:39 UTC</initiated_at>
         <updated_at>2014-11-24 16:48:39 UTC</updated_at>
         <completed_at>2014-11-24 16:48:39 UTC</completed_at>
      </work_step>
      …
   </work_steps>
</work_order>
```

Response example (JSON):

```
{
   "work_order":    {
      "workflow_revision_id": 5,
      "name": "tf1 demo duplicate",
      "initiate_date": "2014-11-24T16:48:38Z",
      "forkedAt": null,
      "derivedFrom": null,
      "workflowName": "tf1 demo duplicate",
      "created_at": "2014-11-24T16:48:37Z",
      "cleanup_after_days": null,
      "monitor_id": null,
      "comments": "",
      "updated_at": "2014-11-24T16:48:40Z",
      "priority": 2,
      "max_running": 0,
      "master_id": 8461,
      "launched_by": 1,
      "initiatedBy": 1,
      "branchedFrom": null,
      "statusDetails": "Executing Step: user input (46755)",
      "purge_after_days": null,
      "id": 8461,
      "workflow_id": 256,
      "branchedFromOrder": null,
      "higherPriority": null,
      "completion_date": null,
      "status": "In Progress",
      "running_as": 2
   },
   "work_steps":    [
         {
        "workOrder_id": 8461,
        "preProcessing": null,
        "group": 8461,
        "executionTimeout": 0,
        "activable_date": "2014-11-24T16:48:39Z",
        "stepName": null,
        "created_at": "2014-11-24T16:48:37Z",
        "updated_at": "2014-11-24T16:48:39Z",
        "timeout": null,
        "on_timeout": null,
        "step_type": "FaspexDelivery",
        "attempt": 1,
        "step_id": 3810,
        "statusDetails": "FASPEX Authorization received",
        "rank": 1,
        "id": 46756,
        "activation_date": "2014-11-24T16:48:39Z",
        "workflow_id": 256,
        "synch_filter": 0,
        "completion_date": "2014-11-24T16:48:39Z",
        "action_id": 2,
        "status": "In Progress",
        "running_as": 2,
        "original_activation_date": "2014-11-24T16:48:39Z",
        "workStepName": "Create faspex package",
        "postProcessing": null
      },
         …
   ]
}
```

Get the Output of a Work Order

This endpoint gets the output of a work order, for example, step names and step variable names with the associated variable values.

The inputs to this API method are a work order ID, step name, variable name (with the desired output), and authentication parameters (login and password). The user must have the correct group privileges to view the work order details.

The output of the API call is an XML structure that contains the details such as variable name, type, value, step name, and so on. It is possible that a workflow contains the same variable name in different steps. Use the step name input to drill down to the step you want to fetch the output variable from.

Another URL parameter allows the user to obtain a variable value rather than the long-form XML wrapper around the variable. Spaces used in variable names (such as `step_name`) are accepted in the URL. Single or double quotes (used to surround variable names) are not accepted.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

Request with a long-form response:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_order_output?
workorder_id=workorder_id&step_name=<step_name >&variable_name=variable_name
```

The required input parameters are the following:

- `workorder_id=`*`workorder_id`*
- `step_name=`*`step_name`*
- `variable_name=`*`variable_name`* (This parameter specifies the variable name within a step).

Example 1: Find an output for a given step and a given variable.

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_order_output?
workorder_id=95&step_name=Test2&variable_name=Count_of_ContentFiles&login=admin
```

Response example:

```
<?xml version="1.0"?>
<work_order_output time="2014-03-14 04:50:56 UTC" action="output">
   <variable id="29039" name="Test2:Count_of_ContentFiles">
      <workorder_id>95</workorder_id>
      <step_name>Test2</step_name>
      <variable_name>Count_of_ContentFiles</variable_name>
      <value_type>int</value_type>
      <value>3</value>
      <variable_type>OUTPUT</variable_type>
   </variable>
</work_order_output>
```

Example 2: Find the output for a specific variable.

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_order_output?
workorder_id=1106&variable_name=FileName&login=admin
```

Response example:

```
<?xml version="1.0"?>
<work_order_output time="2014-03-14 02:19:41 UTC" action="output">
   <variable id="28785" name="Update failure:FileName">
      <workorder_id>11068</workorder_id>
      <step_name>Update failure</step_name>
      <variable_name>FileName</variable_name>
      <value_type>string</value_type>
      <value>FileName</value>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="28788" name="Update success:FileName">
      <workorder_id>11068</workorder_id>
      <step_name>Update success</step_name>
      <variable_name>FileName</variable_name>
      <value_type>string</value_type>
      <value/>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="28789" name="Scan for incoming files:FileName">
      <workorder_id>11068</workorder_id>
      <step_name>Scan for incoming files</step_name>
      <variable_name>FileName</variable_name>
      <value_type>string</value_type>
      <value>/demo/comcast/to_disney/wall_e_5.avi</value>
      <variable_type>OUTPUT</variable_type>
   </variable>
</work_order_output>
```

Example 3: Find all of the output for a particular work order.

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_order_output?
workorder_id=1106&login=admin
```

Response example:

```
<?xml version="1.0"?>
<work_order_output time="2014-03-24 21:25:47 UTC" action="output">
   <variable id="29629" name="Successful ingest notification:FullMessage">
      <workorder_id>11109</workorder_id>
      <step_name>Successful ingest notification</step_name>
    <variable_name>FullMessage</variable_name>
      <value_type>string</value_type>
<value>&lt;MessageDefinition&gt;&lt;Destinations&gt;&lt;To&gt;pavan+support@asperasoft.com&lt;/
To&gt;&lt;/Destinations&gt;&lt;Subject&gt;Ingest successful of wall_e_12.avi for Turner&lt;/
Subject&gt;&lt;/MessageDefinition&gt;
</value>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="29630" name="Ingest failure:FullMessage">
      <workorder_id>11109</workorder_id>
      <step_name>Ingest failure</step_name>
      <variable_name>FullMessage</variable_name>
      <value_type>string</value_type>
      <value></value>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="29631" name="Update failure:manifest">
      <workorder_id>11109</workorder_id>
      <step_name>Update failure</step_name>
      <variable_name>manifest</variable_name>
      <value_type>string</value_type>
      <value></value>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="29632" name="Update failure:manifest_id">
      <workorder_id>11109</workorder_id>
      <step_name>Update failure</step_name>
      <variable_name>manifest_id</variable_name>
      <value_type>int</value_type>
      <value></value>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="29637" name="Scan for incoming files:FileName">
      <workorder_id>11109</workorder_id>
      <step_name>Scan for incoming files</step_name>
      <variable_name>FileName</variable_name>
      <value_type>string</value_type>
      <value>/demo/comcast/to_turner/wall_e_12.avi</value>
      <variable_type>OUTPUT</variable_type>
   </variable>
</work_order_output>
```

Example 4: Find all of the output for a single step in a work order.

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_order_output?
workorder_id=1106&login=admin&step_name=test
```

Response example:

```
<?xml version="1.0"?>
<work_order_output time="2014-03-24 19:07:15 UTC" action="output">
   <variable id="29638" name="Forward:Transferred_File_list">
      <workorder_id>11109</workorder_id>
      <step_name>Forward</step_name>
      <variable_name>Transferred_File_list</variable_name>
      <value_type>array</value_type>
      <value>["/demo/comcast/to_turner/wall_e_12.avi"]</value>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="29639" name="Forward:Transfer_Stats">
      <workorder_id>11109</workorder_id>
      <step_name>Forward</step_name>
      <variable_name>Transfer_Stats</variable_name>
      <value_type>string</value_type>

<value>{:FilesFailed=>0, :JobRetryCount=>0, :FilesTransferring=>0, :BytesWritten=>4830720, :Byte
sTransferred=>4830720, :sessionId=>["918b2456-095c-4ce5-9b33-
af8551f4deb9"], :FilesComplete=>1}</value>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="29640" name="Forward:Job_ID">
      <workorder_id>11109</workorder_id>
      <step_name>Forward</step_name>
      <variable_name>Job_ID</variable_name>
      <value_type>string</value_type>
      <value>f698d602-d6e6-45a4-b82a-df5f65621a2e</value>
      <variable_type>OUTPUT</variable_type>
   </variable>
   <variable id="29641" name="Forward:Transferred_Files_md5">
      <workorder_id>11109</workorder_id>
      <step_name>Forward</step_name>
      <variable_name>Transferred_Files_md5</variable_name>
      <value_type>hash</value_type>
      <value/>
      <variable_type>OUTPUT</variable_type>
   </variable>
</work_order_output>
```

Request that returns a short form response (the value for the specified step and variable, only):

```
http://Orchestrator_IP_address/api/work_order_output/workorder_id.string?
login=admin&step_name=step_name&variable_name=variable_name
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_order_output/11081.string?
step_name=Test2&variable_name=Count_of_ContentFiles&login=admin
```

Response example:

```
3
```

Error use case: A step name or variable name is incorrect.

This is returned:

```
<work_order_output time="2014-11-05 08:43:36 UTC" action="output"></work_order_output>
```

Error use case: The step has not yet been executed when the API request is made.

The request returns a response without a value, for example:

```
<?xml version="1.0"?>
<work_order_output time="2014-03-14 04:50:56 UTC" action="output">
   <variable id="29039" name="Test2:Count_of_ContentFiles">
      <workorder_id>11081</workorder_id>
      <step_name>Test2</step_name>
      <variable_name>Count_of_ContentFiles</variable_name>
      <value_type>int</value_type>
      <value/>
      <variable_type>OUTPUT</variable_type>
   </variable>
</work_order_output>
```

Restart a Work Order

This endpoint restarts a specific work order, identified by its work order ID.

The response includes both the overall status for the work order as well as the individual status of each of its steps.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_order_reset/workorder_id?login=admin
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_order_reset/3109?login=admin
```

Required input parameter:

```
workflow_reporter/work_order_reset/workorder_id
```

Response example:

```
<?xml version="1.0"?>
<work_order time="2015-05-18 04:55:18 UTC" action="status" id="1227" master_id="1227">
  <name>Demo - user task</name>
  <workflow id="553" revision_id="3">Demo - user task</workflow>
  <status state="Created">WorkOrder reset at Sun May 17 21:55:18 PDT 2015</status>
  <comments/>
  <latest_update>reset</latest_update>
  <ownership>
    <initiated_by>admin</initiated_by>
    <running_as>system</running_as>
    <launched_by>admin</launched_by>
  </ownership>
  <priority dedicated_worker="false">2</priority>
  <timestamps>
    <created_at>2015-05-13 15:22:38 UTC</created_at>
    <initiated_at>2015-05-13 15:22:39 UTC</initiated_at>
    <updated_at>2015-05-18 04:55:18 UTC</updated_at>
    <completed_at>2015-05-18 04:53:21 UTC</completed_at>
  </timestamps>
    <work_steps>
           <work_step id="8269" category="System">
            <name>Logic prior</name>
            <status state="Inactive" attempt="0">reset</status>
            <action type="MergePoint" id="69"/>
            <created_at>2015-05-13 15:22:38 UTC</created_at>
            <initiated_at>2015-05-13 15:22:40 UTC</initiated_at>
            <updated_at>2015-05-18 04:55:18 UTC</updated_at>
            <completed_at>2015-05-13 15:22:40 UTC</completed_at>
          </work_step>
          <work_step id="8270" category="System">
            <name>Logic prior 1</name>
            <status state="Inactive" attempt="0">reset</status>
            <action type="MergePoint" id="70"/>
            <created_at>2015-05-13 15:22:38 UTC</created_at>
            <initiated_at>2015-05-13 15:22:40 UTC</initiated_at>
            <updated_at>2015-05-18 04:55:18 UTC</updated_at>
            <completed_at>2015-05-13 15:22:40 UTC</completed_at>
          </work_step>
        <work_step id="8271" category="System">
            <name>Post User input</name>
            <status state="Inactive" attempt="0">reset</status>
            <action type="MergePoint" id="71"/>
            <created_at>2015-05-13 15:22:38 UTC</created_at>
            <initiated_at/>
            <updated_at>2015-05-18 04:55:18 UTC</updated_at>
            <completed_at/>
          </work_step>
        <work_step id="8272" category="User Interactions">
            <name>Test input</name>
            <status state="Inactive" attempt="0">reset</status>
            <action type="UserInput" id="20"/>
            <created_at>2015-05-13 15:22:38 UTC</created_at>
            <initiated_at>2015-05-13 15:22:41 UTC</initiated_at>
            <updated_at>2015-05-18 04:55:18 UTC</updated_at>
            <completed_at>2015-05-18 04:53:21 UTC</completed_at>
          </work_step>
        <work_step id="8273">
            <name>Workflow Start</name>
            <status state="False" attempt="">reset</status>
            <action type="WorkOrderEnd" id=""/>
            <created_at>2015-05-13 15:22:38 UTC</created_at>
            <initiated_at/>
            <updated_at>2015-05-18 04:55:18 UTC</updated_at>
            <completed_at/>
          </work_step>
        <work_step id="8274">
            <name>Workflow Failed</name>
            <status state="False" attempt="">reset</status>
            <action type="WorkOrderEnd" id=""/>
            <created_at>2015-05-13 15:22:38 UTC</created_at>
            <initiated_at/>
            <updated_at>2015-05-18 04:55:18 UTC</updated_at>
            <completed_at/>
          </work_step>
          <work_step id="8275">
            <name>Workflow End</name>
            <status state="False" attempt="">reset</status>
            <action type="WorkOrderEnd" id=""/>
            <created_at>2015-05-13 15:22:38 UTC</created_at>
            <initiated_at/>
            <updated_at>2015-05-18 04:55:18 UTC</updated_at>
            <completed_at/>
          </work_step>
        <work_step id="8276">
            <name>Error Encountered</name>
            <status state="False" attempt="">reset</status>
            <action type="WorkOrderEnd" id=""/>
            <created_at>2015-05-13 15:22:38 UTC</created_at>
            <initiated_at/>
            <updated_at>2015-05-18 04:55:18 UTC</updated_at>
            <completed_at/>
          </work_step>
    </work_steps>
</work_order>
```

Restart One or More Work Orders from a Specific Step

With this endpoint, you can restart a work order from a specific step, or restart multiple work orders from a specific step in each work order.

- Restarting a work order from a specific step: Specify the step ID of the step from which to restart the work order.
- Restarting multiple work orders from a specific step in each work order: In a comma-separated list, specify the step IDs of the steps from which to restart the work orders.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

1. Request to restart a work order from a specific step:

```
https://Orchestrator_IP_address/aspera/orchestrator/api/workstep_reset/step_id
```

For example:

```
https://localhost/asperaorchestrator/api/workstep_reset/34
```

Response example:

```
<?xml version="1.0" encoding="UTF-8"?>
<WorkStep_Reset>
  <workstep_reset>true</workstep_reset>

  <work_steps>
    <workOrder_id>69312</workOrder_id>
    <timeout />
    <running_as />
    <preProcessing />
    <journaling />
    <group />

    <completion_date />
    <activation_time />
    <updated_at>Tue Jan 17 21:47:00 UTC 2017</updated_at>
    <id>549780</id>
    <attempt />
    <activable_date />
    <weight />
    <step_type>WorkOrderEnd</step_type>
    <activation_date />
    <workflow_id>59</workflow_id>
    <status>Inactive</status>
    <workStepName></workStepName>
    <percent_complete />
    <action_id />
    <step_id>-34</step_id>
    <stepName />
    <statusDetails>Work-order manually restarted from 'step2' on</statusDetails>
    <rank>1000020</rank>
    <postProcessing />
    <original_activation_date />
    <synch_filter>0</synch_filter>
    <episteps>--- []</episteps>
    <prerequisites>--- []</prerequisites>
    <on_timeout />
    <executionTimeout />
    <created_at>Tue Jan 17 21:46:36 UTC 2017</created_at>
  </work_steps>
</WorkStep_Reset>
```

2. Request to restart multiple work orders from a specific step in each work order:

```
https://Orchestrator_IP_address/aspera/orchestrator/api/workstep_reset/step_id, step_id
```

For example:

```
https://localhost/aspera/orchestrator/api/workstep_reset/34,63
```

Response example:

Two different work orders each restarted from the second step (steps with ID 34 and 64, respectively).

```
<?xml version="1.0" encoding="UTF-8"?>
<WorkStep_Reset>
  <workstep_reset>true</workstep_reset>

  <work_steps>
    <workOrder_id>69312</workOrder_id>
    <timeout />
    <running_as />
    <preProcessing />
    <journaling />
    <group />

    <completion_date />
    <activation_time />
    <updated_at>Tue Jan 17 21:47:00 UTC 2017</updated_at>
    <id>549780</id>
    <attempt />
    <activable_date />
    <weight />
    <step_type>WorkOrderEnd</step_type>
    <activation_date />
    <workflow_id>59</workflow_id>
    <status>Inactive</status>
    <workStepName></workStepName>
    <percent_complete />
    <action_id />
    <step_id>-34</step_id>
    <stepName />
    <statusDetails>Work-order manually restarted from 'step2' on</statusDetails>
    <rank>1000020</rank>
    <postProcessing />
    <original_activation_date />
    <synch_filter>0</synch_filter>
    <episteps>--- []</episteps>
    <prerequisites>--- []</prerequisites>
    <on_timeout />
    <executionTimeout />
    <created_at>Tue Jan 17 21:46:36 UTC 2017</created_at>
    <workOrder_id>69312</workOrder_id>
    <timeout />
```

```
     <running_as />
     <preProcessing />
     <journaling />
     <group />

     <workOrder_id>58432</workOrder_id>
     <timeout />
     <running_as />
     <preProcessing />
     <journaling />
     <group />

     <completion_date />
     <activation_time />
     <updated_at>Tue Jan 17 21:47:04 UTC 2017</updated_at>
     <id>549770</id>
     <attempt />
     <activable_date />
     <weight />
     <step_type>WorkOrderEnd</step_type>
     <activation_date />
     <workflow_id>59</workflow_id>
     <status>Inactive</status>
     <workStepName></workStepName>
     <percent_complete />
     <action_id />
     <step_id>-63</step_id>
     <stepName />
     <statusDetails>Work-order manually restarted from 'step2' on</statusDetails>
     <rank>1000020</rank>
     <postProcessing />
     <original_activation_date />
     <synch_filter>0</synch_filter>
     <episteps>--- []</episteps>
     <prerequisites>--- []</prerequisites>
     <on_timeout />
     <executionTimeout />
     <created_at>Tue Jan 17 21:46:54 UTC 2017</created_at>
   </work_steps>
</WorkStep_Reset>
```

Cancel a Work Order

Description:

An API method to cancel a specific work order, identified by its work order ID. The result includes both the overall status for the work order, as well as the individual status for each of its steps.

Adding the `force` option will ensure that the work order is cancelled even if it is not responding to a cancel request (`work_order_cancel`) to cancel itself. If the `force` option is not provided, the work order status is `Failed`.

Note: If the work order was already complete (or it failed), then the returned status will indicate the actual status.

Usage:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_order_cancel/workorder_id?
login=admin
```

For example, the user could receive one of the following:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_order_cancel/3109?login=admin
        http://Orchestrator_IP_address/aspera/orchestrator/api/work_order_cancel/3109?
login=admin&force=true
```

The required input parameter is the following:

```
workflow_reporter/work_order_cancel/workorder_id
```

Response example (XML):

```
<?xml version="1.0"?>
<work_order time="2012-03-02 03:45:05 UTC" action="cancel" id="3109" master_id="5309">
  <name>test after-end</name>
  <workflow id="22" revision_id="7">test after-end</workflow>
  <status state="Canceled">Canceled at Thu Mar 01 19:45:05 -0800 2012</status>
  <comments></comments>
  <ownership>
    <initiated_by>admin</initiated_by>
    <running_as>admin</running_as>
    <launched_by>admin</launched_by>
  </ownership>
  <priority dedicated_worker="false">2</priority>
  <timestamps>
    <created_at>2012-03-02 03:44:48 UTC</created_at>
    <initiated_at>2012-03-02 03:44:50 UTC</initiated_at>
    <updated_at>2012-03-02 03:45:05 UTC</updated_at>
    <completed_at>2012-03-02 03:45:05 UTC</completed_at>
  </timestamps>
  <work_steps>
        <work_step id="37601">
          <name>never</name>
          <status state="Inactive" attempt="0"></status>
          <action type="MergePoint" id="1"/>
          <created_at>2012-03-02 03:44:48 UTC</created_at>
          <initiated_at></initiated_at>
          <updated_at>2012-03-02 03:44:48 UTC</updated_at>
          <completed_at></completed_at>
        </work_step>
        <work_step id="37602">
          <name>gnarp</name>
          <status state="Obsolete" attempt="1">Work Order Processing is finished</status>
          <action type="CustomRuby" id="12"/>
          <created_at>2012-03-02 03:44:48 UTC</created_at>
          <initiated_at>2012-03-02 03:44:50 UTC</initiated_at>
          <updated_at>2012-03-02 03:45:05 UTC</updated_at>
          <completed_at></completed_at>
        </work_step>
  </work_steps>
</work_order>
```

Get Work Order Statistics

This endpoint retrieves statistics about work order performance.

You can retrieve daily statistics for up to the past 30 days or hourly statistics for a specific day. The primary data filter is the parameter `workflow_id`. The `label` parameter can be added to the request to retrieve a greater level of detail.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

```
https://Orchestrator_IP_address/aspera/orchestrator/api/work_order_stats?
login=admin&workflow_id=1&label=Test
https://Orchestrator_IP_address/aspera/orchestrator/api/work_order_stats?
login=admin&date=2017-04-04
```

Sample response:

```
<?xml version="1.0"?>
<work_step_stats>
    <anon>
        <Date>2017-04-10</Date>
        <Completed_Steps>48289</Completed_Steps>
        <Failed_Steps>0</Failed_Steps>
        <Errored_steps>0</Errored_steps>
        <Obsolete_Steps>0</Obsolete_Steps>
        <Last_hour>Mon Apr 10 12:00:00 UTC 2017</Last_hour>
        <Activating_Steps>3752</Activating_Steps>
        <In_Progress_Steps>0</In_Progress_Steps>
        <Paused_Steps>0</Paused_Steps>
        <Pausing_Steps>0</Pausing_Steps>
        <Canceling_Steps>0</Canceling_Steps>
        <Resuming_Steps>0</Resuming_Steps>
        <Rolling_Back_Steps>0</Rolling_Back_Steps>
        <Undefined_Steps>0</Undefined_Steps>
    </anon>
    <anon>
        <Date>2017-04-09</Date>
        <Completed_Steps>92734</Completed_Steps>
        <Failed_Steps>0</Failed_Steps>
        <Errored_steps>0</Errored_steps>
        <Obsolete_Steps>0</Obsolete_Steps>
        <Last_hour>Sun Apr 09 23:00:00 UTC 2017</Last_hour>
        <Activating_Steps>4087</Activating_Steps>
        <In_Progress_Steps>0</In_Progress_Steps>
        <Paused_Steps>0</Paused_Steps>
        <Pausing_Steps>0</Pausing_Steps>
        <Canceling_Steps>0</Canceling_Steps>
        <Resuming_Steps>0</Resuming_Steps>
        <Rolling_Back_Steps>0</Rolling_Back_Steps>
        <Undefined_Steps>0</Undefined_Steps>
    </anon>
    <anon>
        <Date>2017-04-08</Date>
        <Completed_Steps>99324</Completed_Steps>
        <Failed_Steps>0</Failed_Steps>
        <Errored_steps>0</Errored_steps>
        <Obsolete_Steps>0</Obsolete_Steps>
        <Last_hour>Sat Apr 08 23:00:00 UTC 2017</Last_hour>
        <Activating_Steps>3905</Activating_Steps>
        <In_Progress_Steps>0</In_Progress_Steps>
        <Paused_Steps>0</Paused_Steps>
        <Pausing_Steps>0</Pausing_Steps>
        <Canceling_Steps>0</Canceling_Steps>
        <Resuming_Steps>0</Resuming_Steps>
        <Rolling_Back_Steps>0</Rolling_Back_Steps>
        <Undefined_Steps>0</Undefined_Steps>
    </anon>
    <anon>
        <Date>2017-04-07</Date>
        <Completed_Steps>98554</Completed_Steps>
        <Failed_Steps>0</Failed_Steps>
        <Errored_steps>0</Errored_steps>
        <Obsolete_Steps>0</Obsolete_Steps>
        <Last_hour>Fri Apr 07 23:00:00 UTC 2017</Last_hour>
        <Activating_Steps>4148</Activating_Steps>
        <In_Progress_Steps>0</In_Progress_Steps>
        <Paused_Steps>0</Paused_Steps>
        <Pausing_Steps>0</Pausing_Steps>
        <Canceling_Steps>0</Canceling_Steps>
        <Resuming_Steps>0</Resuming_Steps>
        <Rolling_Back_Steps>0</Rolling_Back_Steps>
        <Undefined_Steps>0</Undefined_Steps>
    </anon>
    <anon>
        <Date>2017-04-06</Date>
        <Completed_Steps>69733</Completed_Steps>
        <Failed_Steps>0</Failed_Steps>
        <Errored_steps>0</Errored_steps>
        <Obsolete_Steps>0</Obsolete_Steps>
        <Last_hour>Thu Apr 06 23:00:00 UTC 2017</Last_hour>
        <Activating_Steps>4047</Activating_Steps>
        <In_Progress_Steps>0</In_Progress_Steps>
        <Paused_Steps>0</Paused_Steps>
        <Pausing_Steps>0</Pausing_Steps>
        <Canceling_Steps>0</Canceling_Steps>
        <Resuming_Steps>0</Resuming_Steps>
        <Rolling_Back_Steps>0</Rolling_Back_Steps>
        <Undefined_Steps>0</Undefined_Steps>
    </anon>
</work_step_stats>
```

Check the Status of a Specific Workflow Step

This endpoint polls the status of a specific step in a workflow, identified by its work order ID and step name.

If the work order includes sub-workflows, only the sub-workflow status displays, not the status for the steps within the sub-workflow.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

Request:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_step_status/work_order.xml?
login=admin&workorder_id=workorder_id&step_name=step_name
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_step_status/work_order.xml?
login=admin&workorder_id=8461&step_name=Create faspex package
```

Response example:

```
<?xml version="1.0"?>
<Work-step-status>
   <inputs type="array"/>
   <carry-thrus type="array">
      <carry-thru>
         <name>myfiles</name>
         <id type="integer">60067</id>
         <type>string</type>
         <value>/root/chris/data_for_demos/shiporder.cmd</value>
      </carry-thru>
   </carry-thrus>
   <work-step>
      <workOrder-id type="integer">8461</workOrder-id>
      <preProcessing nil="true"/>
      <group type="integer">8461</group>
      <executionTimeout type="integer">0</executionTimeout>
      <activable-date type="datetime">2014-11-24T16:48:39Z</activable-date>
      <stepName nil="true"/>
      <created-at type="datetime">2014-11-24T16:48:37Z</created-at>
      <updated-at type="datetime">2014-11-24T16:48:39Z</updated-at>
      <timeout nil="true"/>
      <on-timeout nil="true"/>
      <step-type>FaspexDelivery</step-type>
      <attempt type="integer">1</attempt>
      <step-id type="integer">3810</step-id>
      <statusDetails>FASPEX Authorization received</statusDetails>
      <rank type="integer">1</rank>
      <id type="integer">46756</id>
      <activation-date type="datetime">2014-11-24T16:48:39Z</activation-date>
      <workflow-id type="integer">256</workflow-id>
      <synch-filter type="integer">0</synch-filter>
      <completion-date type="datetime">2014-11-24T16:48:39Z</completion-date>
      <action-id type="integer">2</action-id>
      <status>Complete</status>
      <running-as type="integer">2</running-as>
      <original-activation-date type="datetime">2014-11-24T16:48:39Z</original-activation-date>
      <workStepName>Create faspex package</workStepName>
      <postProcessing nil="true"/>
   </work-step>
   <outputs type="array">
      <output>
         <name>package_id</name>
         <id type="integer">60070</id>
         <type>string</type>
         <value>5e0f9336-5e58-46a5-b944-a8253a956041</value>
      </output>
      <output>
         <name>package_post_url</name>
         <id type="integer">60071</id>
         <type>string</type>
<value><![CDATA[fasp://faspex@testchris2.sl.dev.asperacloud.net:33001/Transcoded -
5e0f9336-5e58-46a5-b944-a8253a956041.aspera-package/PKG - Transcoded?
token=ATM3_ACsvFz3yt1byrRDQ7cEtFKg0wxWS6ubjsWnkA8IAxSlbKoAAEWHQkLFXfeLjDz5Eq8uDH7_3MTA&cookie=as
pera.faspex20:u:90ce20d7]]></value>
      </output>
   </outputs>
</Work-step-status>
```

The following request returns only the step status:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_order.xml?
login=admin&workorder_id=workorder_id&step_name=step_name&status_only=true
```

For example:

```
https://Orchestrator_IP_address/aspera/orchestrator/api/work_step_status/work_order.xml?
login=admin&workorderid=8461&step_name=Create faspex package&status_only=true
```

Response example:

```
Complete
```

Note: An output value can be obtained from the step ID. You find the step ID in the response to the previous API call. For example, the following is an example of a returned output value (XML):

```
<id type="integer">46756</id>
```

Usage for a request with text response:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_step_status/step_id.txt?
login=admin&output_name=output_name
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_step_status/46756.txt?
login=admin&output_name=package_id
```

Response example:

```
5e0f9336-5e58-46a5-b944-a8253a956041
```

Request:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_step_status/252158.xml?
login=admin&output_name=IndexedTable
```

Response example for `<value>`:

```
<?xml version="1.0"?>
<Work_step_output>
   <name>IndexedTable</name>
   <id type="integer">301979</id>
   <value>{'Transcode server' => {"node_id"=>"transcoder", "node"=>"Transcode
server", "capacity_kb"=>"30000"}, 'Mac' => {"node_id"=>"localhost", "node"=>"Mac",
"capacity_kb"=>"20000"}}</value>
   <type>hash</type>
</Work_step_output>
```

Cancel a Work Step

This endpoint cancels a specific step in the workflow.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

The required path parameter is the workflow step ID. The response returns the cancelled workflow step, its current status, and timestamps.

Request:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_step_cancel/workflow_step_id?
login=admin
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/work_step_cancel/3109?login=admin
```

Response example:

```
<?xml version="1.0"?>
  <work_step time="2015-05-18 04:53:20 UTC" action="cancel" id="" >
    <name>Test input</name>
    <workflow id="553"></workflow>
    <status state="Canceling">Awaiting user input at Sun May 17 21:53:17 PDT 2015</status>
    <action type="UserInput" id="20"/>
    <status state="Canceling" attempt="1">Awaiting user input at Sun May 17 21:53:17 PDT 2015</
status>
    <timestamps>
      <created_at>2015-05-13 15:22:38 UTC</created_at>
      <initiated_at>2015-05-13 15:22:41 UTC</initiated_at>
      <updated_at>2015-05-18 04:53:20 UTC</updated_at>
      <completed_at></completed_at>
    </timestamps>
  </work_step>
```

Get Work Step Statistics

This endpoint retrieves statistics about workflow step performance.

You can retrieve daily statistics for up to the past 30 days or hourly statistics for a specific day. The primary data filter is the parameter `workflow_id`. The parameters `step_id` and `step_type` can be added to the request to retrieve a greater level of detail.

Usage

```
https://Orchestrator_IP_address/aspera/orchestrator/api/work_step_stats?
login=admin&workflow_id=1
https://Orchestrator_IP_address/aspera/orchestrator/api/work_step_stats?
login=admin&date=2017-04-04
```

Sample response:

```
<?xml version="1.0"?>
<work_order_stats>
    <anon>
        <Date>2017-03-31</Date>
        <Last_hour>Fri Mar 31 05:56:51 UTC 2017</Last_hour>
        <Complete_Work_Orders>18612</Complete_Work_Orders>
        <Failed_Work_Orders>0</Failed_Work_Orders>
        <Errored_Work_Orders>1</Errored_Work_Orders>
        <Canceled_Work_Orders>6005</Canceled_Work_Orders>
        <Rolled_Back_Work_Orders>0</Rolled_Back_Work_Orders>
        <Destroyed_Work_Orders>0</Destroyed_Work_Orders>
        <Cleared_Work_Orders>0</Cleared_Work_Orders>
        <Flagged_Work_Orders>0</Flagged_Work_Orders>
        <Initializing_Work_Orders>0</Initializing_Work_Orders>
        <Initialized_Work_Orders>0</Initialized_Work_Orders>
        <Created_Work_Orders>0</Created_Work_Orders>
        <Ready_Work_Orders>0</Ready_Work_Orders>
        <In_Progress_Work_Orders>0</In_Progress_Work_Orders>
        <Unknown_Work_Orders>0</Unknown_Work_Orders>
     </anon>
</work_order_stats>
```

API Endpoints—Workflows

Get Available Workflows on the System

This method lists all workflows in Orchestrator and specifies whether each workflow is in a published state or not. Only a published workflow can be initiated as a work order.

Usage:

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Request with an authenticated user:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/workflows_list/workflow_id
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/workflows_list/0
```

Request with inline authentication:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/workflows_list/0?login=admin
```

Response example:

```
<?xml version="1.0"?>
<workflows time="2011-10-20 17:18:58 UTC" action="list" id="0">
    <workflow id="2"
               name="Incoming File Scanning Daemon"
               published_status="published"
               published_revision_id="13"
               latest_revision_id="15"
               last_modification="2011-04-12 06:56:16 UTC" >
    </workflow>
    <workflow id="3"
               name="Incoming Clean File Staging Daemon"
               published_status="published"
               published_revision_id="9"
               latest_revision_id="9"
               last_modification="2011-03-30 19:47:38 UTC" >
        Created from revision #4 from workflow 'Incoming File Scanning Daemon'
    </workflow>
    <workflow id="4"
               name="Asset and Schedule Import Daemon"
               published_status="draft"
               published_revision_id="-"
               latest_revision_id="20"
               last_modification="2011-05-06 19:20:36 UTC" >
    </workflow>
</workflows>
```

Response example (JSON:

```
{
   "workflows":{
      "id":0,
      "action":"list",
      "workflow":[
         {
            "last_modification":"2015-12-15T00:21:08Z",
            "published_status":"published",
            "published_revision_id":4,
            "id":1,
            "latest_revision_id":4,
            "full_name":"TestFol1: fanout test",
            "comments":"",
            "portable_id":"fanout_test",
                "name":"fanout test",
                "folder_id":36
             },
             {
                "last_modification":"2015-12-15T00:21:08Z",
                "published_status":"published",
                "published_revision_id":1,
                "id":2,
                "latest_revision_id":1,
                "full_name":"TestFol1: send email",
                "comments":"",
                "portable_id":"send_email",
                "name":"send email",
                "folder_id":36
             }
          ],
          "time":"2016-01-11 20:33:05"
    }
}
```

Export a Workflow

This API method exports a workflow. When you pair this method with "Import a Workflow" on page 209 (for development and production machines), you complete a direct deployment of the workflow from one Orchestrator instance to another.

Request

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

```
https://Orchestrator_IP_address/aspera/orchestrator/api/export_workflow/workflow_id
```

For example:

```
http://localhost:3000/aspera/orchestrator/api/export_workflow/334
```

Parameters

Parameter Name	Required/Optional	Data Type	Definition
export_file	Required	boolean	Whether to export the workflow file. Values: `true` or `false`
export_file_name	Required	string	Name for the workflow file
folder_id	Optional	string	ID for folder containing the workflow. If no folder ID is given, the workflow is imported into the root ("/") directory.
link_to_subworkflows	Optional	boolean	Whether to link the workflow to subworkflows. Values: `true` or `false`
publish_workflow	Optional	boolean	Whether to publish the workflow. Values: `true` or `false`

Parameter Name	Required/Optional	Data Type	Definition
format	Optional		

Response

The response is a local download of the workflow file.

Sample Returned XML:

```
---
- !ruby/object:Workflow
  attributes:
    portable_id: d2fa6220-9542-0137-14cd-2a0068596101
    generatedFrom: 491
    name: Sub
    id: 334
    comments: Imported from file /Users/anushasyed/Downloads/
Workflow_Master__3615/20190730_151302/Workflow_Sub__3614.wkf
    revision_id: 1
    created_by: 1
    run_as: SYSTEM
    created_at: 2020-01-28 20:42:30.000000000 Z
    updated_at: 2020-01-28 20:42:30.000000000 Z
    folder_id:
    description: ''
    max_running:
    purge_after_days:
    cleanup_after_days:
    tags:
    label:
    source_revision: Sub|1
    templates:
    user_lock:
    locked_by:
- !ruby/object:Whiteboard
  attributes:
    xml_export: |
      <opt id="334">
        <Start x="100" y="50" id="a816916d-e373-172c-a8ed-fa9afae52583">
          <Synch_factor>0</Synch_factor>
          <Prerequisites></Prerequisites>
          <Parameters></Parameters>
        </Start>
        <Error x="1200" y="350" id="e90b608e-ccc5-bf34-85c6-4dd581e52818">
          <Synch_factor>0</Synch_factor>
          <Prerequisites></Prerequisites>
        </Error>
        <End x="1200" y="50" id="e3566700-5c63-ed0f-a471-580d96c06d13">
          <Synch_factor>0</Synch_factor>
          <Prerequisites>
            <Prerequisite id="b4caf448-a957-88be-369e-a8353ac05df0" status="Complete"
router="draw2d.ManhattanConnectionRouter" />
          </Prerequisites>
        </End>
        <Fail x="1200" y="200" id="9a702333-908a-9cb0-e013-fc3645fa9b03">
          <Synch_factor>0</Synch_factor>
          <Prerequisites></Prerequisites>
        </Fail>
        <Steps>
          <Step x="520" y="40" id="b4caf448-a957-88be-369e-a8353ac05df0">
            <Action_id>124</Action_id>
            <Step_name>New CustomRuby Step</Step_name>
            <Step_type>CustomRuby</Step_type>
            <Synch_factor>0</Synch_factor>
            <Prerequisites>
              <Prerequisite id="a816916d-e373-172c-a8ed-fa9afae52583" status="true"
router="draw2d.ManhattanConnectionRouter" />
            </Prerequisites>
            <Parameters></Parameters>
            <Inputs></Inputs>
            <Outputs></Outputs>
            <comments></comments>
            <weight>undefined</weight>
          </Step>
        </Steps>
        <Parameters></Parameters>
        <Notes></Notes>
      </opt>
    id: 491
    workflow_id: 334
    error_free: 1
    created_by: 1
    revision_id: 1
    created_at: 2020-01-28 20:42:30.000000000 Z
    updated_at: 2020-01-28 20:42:30.000000000 Z
    revisions_log:
- CustomRuby__124: !ruby/object:CustomRuby
    attributes:
      id: 124
      name: Log Information  (import_36)
      comments: Logs that this step has completed. Does nothing else.
      execute_code: info "This is coming from CustomRuby"
      validateInputs_code: ''
      outputsSpec_code: ''
      inputsSpec_code: ''
      created_at: 2020-01-28 20:42:30.000000000 Z
      updated_at: 2020-01-28 20:42:30.000000000 Z
      mandatoryInputs: ''
      optionalInputs: ''
      typedOutputs: ''
      use_code_from_github: 0
      github_token: Snicket13!
      github_repo: ''
      github_file_path: ''
      github_branch_name: ''
      github_user_name: admin
      github_hostname: ''
      verify_ssl: 1
- CustomRuby__124: []
- {}
- CustomRuby: 0.5.0
```

Import a Workflow

This endpoint imports a workflow into another Orchestrator instance.

You must first export the workflow from the source machine; see "Export a Workflow" on page 207 for more information.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

Request:

```
https://ip_address/aspera/orchestrator/api/import_workflow?
login=login_name&import_file=file_path
```

For example:

```
https://10.0.0.10/aspera/orchestrator/api/import_workflow?login=admin&import_file=/home/carmen/
Desktop/miscelaneous/ORCHworkflows/
Workflow_Compress_filesinroot_as_TARs_in_a_Queue__8.yml&import_file_name=Workflow_Compress_files
inroot_as_TARs_in_a_Queue__8.yml
```

Parameters

Parameter Name	Required/Optional	Data Type	Definition
import_file	Required	boolean	Whether to export the workflow file. Values: `true` or `false`
import_file_name	Required	string	Name for the workflow file

Parameter Name	Required/Optional	Data Type	Definition
folder_id	Optional	string	ID for folder containing the workflow. If no folder ID is given, the workflow is imported into the root ("/") directory.
link_to_subworkflows	Optional	boolean	Whether to link the workflow to subworkflows. Values: `true` or `false`
publish_workflow	Optional	boolean	Whether to publish the workflow. Values: `true` or `false`
format	Optional		

Import a Workflow with Constraints

To import a workflow in Orchestrator with its constraints—sub-workflows and acton (plugin) templates—make a call to the `/find_constraints` endpoint, followed by a call to the `/import_with_constraints` endpoint.

Summary of Procedure

1. (Preliminary) Export the desired workflow from the source instance of Orchestrator; see "Export a Workflow" on page 207 for more information.
2. Make a GET request to `/find_constraints`.
3. Parse the response for the import constraints, and use those constraints to construct a request body.
4. Make a POST request to `/import_with_constraints`, using the request body that you created in the previous step.

Note: The default format for the response is XML. The examples in this procedure, in JSON format, were returned by adding `&format=json` to the query string.

Usage

1. Make a GET request to `/find_constraints` to get the import constraints for your workflow. For example:

```
https://localhost/aspera/orchestrator/api/find_constraints?file=media/videos/
format_conversion.wkf&wf_id=21
```

The query parameters are:

- `file` - The full path to the workflow to be imported. Required for all requests.
- `wf_id` - The ID of an existing workflow. Include this parameter if you are importing your workflow as a revision to an existing workflow.

The response contains constraint objects, and each constraint object contains multiple workflow elements corresponding to a particular category, such as subworkflows or action templates. Each workflow element has an array of all possible options (the *constraints*).

For example: `subwf constraints` contains `Demo Sub-Workflow` and all the other associated subworkflows of the workflow. `Demo Sub-Workflow` has the constraints `Map to the Existing`

`Sub-Workflow`, `Create a new Sub-Workflow with a unique name`, and `Add a new revision to the existing Sub-Workflow`.

Example response:

```
[
  {
    "filename": "Workflow_Demo_Workflow__3630.wkf"
  },
  {
    "add as revision": 21
  },
  {
    "subwf constraints": {
      "Demo Sub-Workflow": [
        "Map to the Existing Sub-Workflow",
        "Create a new Sub-Workflow with a unique name",
        "Add a new revision to the existing Sub-Workflow"
      ],
      "Demo Sub-Sub-Workflow": [
        "Map to the Existing Sub-Workflow",
        "Create a new Sub-Workflow with a unique name",
        "Add a new revision to the existing Sub-Workflow"
      ]
    }
  },
  {
    "action template constraints": {
      "Search Files": [
        "Map to the existing Global Template",
        "Create a new Global Template with a different name",
        "Override the existing Global Template with this new template"
      ],
      "MD5 Demo_1": [
        "Map to the existing Global Template",
        "Create a new Global Template with a different name",
        "Override the existing Global Template with this new template"
      ],
      "Log Information": [
        "Map to the existing Global Template",
        "Create a new Global Template with a different name",
        "Override the existing Global Template with this new template"
      ]
    }
  },
  {
    "remote node constraints": {
      "Baton": [
        "Map to the existing Remote Node",
        "Create a new Remote Node with a different name",
        "Override the Existing Remote Node with this importing node"
      ]
    }
  },
  {
    "missing plugins": null,
    "Auto-enable missing plugins?": null
  }
]
```

2. Parse the response from the previous step to construct a request body.

   For each element in each constraint object, select *one* constraint from the list of possible options. For example, for the subworkflow `Demo Sub-Workflow` (contained in the `subwf constraints` constraint object), you might select the constraint, `Map to the Existing Sub-Workflow`.

   For example:

```
[
  {
    "filename": "Workflow_Demo_Workflow__3630.wkf"
  },
  {
    "add as revision": 21
  },
  {
    "subwf constraints": {
      "Demo Sub-Workflow": "Map to the Existing Sub-Workflow",
      "Demo Sub-Sub-Workflow": "Add a new revision to the existing Sub-Workflow"
    }
  },
  {
    "action template constraints": {
      "Search Files": "Map to the existing Global Template",
      "MD5 Demo_1": "Map to the existing Global Template",
      "Log Information": "Map to the existing Global Template"
    }
  },
  {
    "remote node constraints": {
      "Baton": "Map to the existing Remote Node"
    }
  },
  {
    "Auto-enable missing plugins?": "true"
  }
]
```

3. Import the workflow by making a POST request to the `/import_with_constraints` endpoint.

   ```
   https://localhost/aspera/orchestrator/api/import_with_constraints
   ```

   Include the request body you created in the previous step.

   Example response:

   Note: The response contains metadata associated with the workflow that you imported.

   ```
   {
     "workflow": {
       "id": 296,
       "name": "Demo Workflow  (import_1)",
       "comments": "Imported from file FULL_PATH_TO_WORKFLOW.wkf",
       "revision_id": null,
       "generatedFrom": null,
       "created_by": 1,
       "run_as": "SYSTEM",
       "created_at": "2019-09-10T23:37:40.748Z",
       "updated_at": "2019-09-10T23:37:40.748Z",
       "folder_id": null,
       "description": "",
       "max_running": null,
       "purge_after_days": null,
       "cleanup_after_days": null,
       "tags": [],
       "label": null,
       "source_revision": "Demo Workflow|3",
       "portable_id": "9d9926b0-9a35-0137-14d1-2a0068596101",
       "templates": {},
       "user_lock": null,
       "locked_by": null
     }
   }
   ```

Get all Workflow Steps that Use the Same Plugin

This endpoint retrieves all workflow steps—across all workflows—that use the same plugin, along with the associated work orders.

The `type` parameter specifies the plugin used by the steps you need to retrieve. For example, you may need to retrieve all steps that use the `RemoteFileWatcher` plugin.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Request

Usage: Retrieve all workflow steps that use the specified plugin:

```
https://10.0.154.60/aspera/orchestrator/api/retrieve_steps?type=plugin_name
```

For example, retrieve all workflow steps that use the RemoteFileWatcher plugin:

```
https://10.0.154.60/aspera/orchestrator/api/retrieve_steps?type=RemoteFileWatcher
```

Request Parameters

Parameter Name	Required/Optional	Data Type	Definition
type	Required	string	The name of the plugin used by the steps you need to retrieve.

Response

The response returns a `RetrieveStepInfo` object that contains a list of all steps with the same plugin.

Response example:

```
HTTP/1.1 200 OK
Date: Fri, 20 Mar 2020 19:11:57 GMT
Server: Apache
Strict-Transport-Security: max-age=63072000; includeSubdomains; preload
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Content-Type: application/xml; charset=utf-8
ETag: W/"4a708341d2a4c91e1bf5673e402dfe4b"
Cache-Control: max-age=0, private, must-revalidate
X-Request-Id: 9c7379a3-209c-465c-b725-47841f0eee1c
X-Runtime: 0.203392
Transfer-Encoding: chunked

<?xml version="1.0"?>
<RetrieveStepInfo>
  <Steps>
    <step_id>adf7efc8-f0e5-f6d7-22af-f3cbc5dfce10</step_id>
    <step_name>New RemoteFileWatcher Step</step_name>
    <step_status>true</step_status>
    <wf_id>4</wf_id>
    <wo_id>4</wo_id>
    <wo_status>Error</wo_status>
  </Steps>
  <Steps>
    <step_id>adf7efc8-f0e5-f6d7-22af-f3cbc5dfce10</step_id>
    <step_name>New RemoteFileWatcher Step</step_name>
    <step_status>true</step_status>
    <wf_id>4</wf_id>
    <wo_id>5</wo_id>
    <wo_status>Complete</wo_status>
  </Steps>
  <Steps>
    <step_id>adf7efc8-f0e5-f6d7-22af-f3cbc5dfce10</step_id>
    <step_name>New RemoteFileWatcher Step</step_name>
    <step_status>true</step_status>
    <wf_id>4</wf_id>
    <wo_id>6</wo_id>
    <wo_status>In Progress</wo_status>
  </Steps>
</RetrieveStepInfo>
```

Response Attributes

Attribute	Required/Optional	Data Type	Definition
step_id	Required	integer	The ID of the returned step
step_name	Optional	string	The name of the returned step

Attribute	Required/Optional	Data Type	Definition
step_status	Optional	boolean	Returns "true" if the workflow prerequisites for this step have been met. Returns "complete" if the workflow step has completed successfully.
wf_id	Required	integer	The ID of the workflow that contains the step.
wo_id	Optional	integer	The ID of the most recent work order for the step.
wo_status	Optional	string	The status of the most recent work order for the step. Allowed values: In Progress, Completed, Failed.

Get Inputs Specification for a Workflow

This endpoint returns the run-time parameters for a workflow, along with a flag indicating whether the parameter is required or optional.

Usage

Request:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/workflow_inputs_spec/workflow_id?
login=admin
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/workflow_inputs_spec/158?login=admin
```

Response example:

```
<?xml version="1.0"?>
<workflow_inputs_spec time="2015-04-30 16:19:21 UTC" action="inputs_spec">
  <workflow_name>ADI Ingest</workflow_name>
  <workflow_id>158</workflow_id>
      <parameter id="288" name="md5_check">
        <workflow_id>158</workflow_id>
        <value_type>flag</value_type>
        <optional>false</optional>
        <value></value>
      </parameter>

      <parameter id="290" name="filesize_check">
        <workflow_id>158</workflow_id>
        <value_type>flag</value_type>
        <optional>false</optional>
        <value></value>
      </parameter>

      <parameter id="289" name="dtd_check">
        <workflow_id>158</workflow_id>
        <value_type>flag</value_type>
        <optional>false</optional>
        <value></value>
      </parameter>
</workflow_inputs_spec>
```

Get the Output Specification of a Workflow

This endpoint gets the output specification of a workflow, for example, step names and step variable names without the variable values.

The inputs to this API method are a workflow ID and authentication parameters (if passing them inline. The user must have privileges to view workflow details.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

The required query parameter is *workflow_id*.

Request:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/workflow_outputs_spec?
login=admin&workflow_id=workflow_id
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/workflow_outputs_spec?
login=admin&workflow_id=878
```

Response example:

```
<?xml version="1.0"?>
<workflow_outputs_spec time="2012-09-14 23:04:50 UTC" action="opspec">
      <context id="5513">
        <workflow_id>878</workflow_id>
        <step_name>Parameter</step_name>
        <variable_name>input</variable_name>
        <value_type>string</value_type>
        <variable_type></variable_type>
        <created_at>2012-09-11 00:31:22 UTC</created_at>
        <updated_at>2012-09-11 00:31:22 UTC</updated_at>
      </context>
      <context id="5512">
        <workflow_id>878</workflow_id>
<step_name>Test FedWF</step_name>
        <variable_name>message</variable_name>
        <value_type>string</value_type>
        <variable_type></variable_type>
        <created_at>2012-09-11 00:31:22 UTC</created_at>
        <updated_at>2012-09-11 00:31:22 UTC</updated_at>
      </context>
</work_order_outputs_spec>
```

Check the Running Status for all Workflows

This endpoint indicates which workflows have work orders that are running and which workflows do not.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

Request:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/workflows_status/0?login=admin
```

Response example:

```
<?xml version="1.0"?>
<workflows time="2012-01-25 07:22:13 UTC" action="status" id="0">
    <workflow id="1" name="AAA - file processing for client" status="not running"
last_activity="-"/>
    <workflow id="3" name="AAA - retrieve file for distribution" status="not running"
last_activity="-"/>
    <workflow id="4" name="AAA - Master Controller processing" status="not running"
last_activity="-"/>
    <workflow id="6" name="AAA - Transcoding" status="not running" last_activity="-"/>
    <workflow id="10" name="test faspex" status="not running" last_activity="-"/>
    <workflow id="13" name="test mapping" status="running" last_activity="2012-01-25 07:21:23
UTC"/>
    <workflow id="14" name="test multi role" status="running" last_activity="2012-01-23
20:11:37 UTC"/>
</workflows>
```

Check the Running Status for a Specific Workflow

This endpoint indicates whether or not a specific workflow has work orders that are currently running.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

The required path parameter is *<workflow_id*.

Request:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/workflows_status/workflow_id?login=admin
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/workflows_status/1?login=admin
```

Response example:

```
<?xml version="1.0"?>
<workflows time="2012-01-25 07:31:17 UTC" action="status" id="1">
    <workflow id="1" name="AAA - file processing for client" status="not running"
last_activity="-"/>
</workflows>
```

Check the Detailed Running Status for a Specific Workflow

This endpoint indicates—for a given workflow—the count of work orders corresponding to each execution status.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

The required path parameter is *workflow_id*.

Request:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/workflow_details/workflow_id?login=admin
```

For example:

```
http://Orchestrator_IP_address/aspera/orchestrator/api/workflow_details/13?login=admin
```

Response example:

Note: Only the statuses that are encountered are reported. For example, if no work orders have failed for that workflow, no lines will be reported for that status.

```
<?xml version="1.0"?>
<workflows time="2012-01-25 07:11:52 UTC" action="details"  id="13">
  <workflow id="13" name="test mapping" >
      <status type="Complete" count="1" last_activity="2012-01-23 19:11:33"/>
      <status type="Failed" count="1764" last_activity="2012-01-25 07:11:15"/>
      <status type="In Progress" count="2" last_activity="2012-01-25 07:11:29"/>
      <status type="Error" count="1" last_activity="2012-01-23 19:05:51"/>
  </workflow>
</workflows>
```

Run Test Cases for Workflow Steps

This endpoint runs the test cases for steps in a workflow.

There are three possible calls to the endpoint, ranging from all-inclusive to most specific:

- Run all the test cases for all the steps in a workflow, given the workflow ID
- Run all the test cases for that step, given a workflow ID and step name
- Run a specific test case, given a workflow ID, step name, and the test template name

Note: The workflow ID mentioned above can be expressed as `*workflow_id*.xml` or `*workflow_id*.json`, as appropriate.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

The `run_id` returned in the response is used in the request for the `/test_step/*workflow_id*` endpoint. See "Get the Status of Executed Test Cases for Workflow Steps" on page 218 for more information.

Usage 1: Run all the test cases for all the steps in a workflow, given the workflow ID

```
https://Orchestrator_IP_address/aspera/orchestrator/api/test_step/workflow_id.xml
```

For example:

```
https://Orchestrator_IP_address/aspera/orchestrator/api/test_step/2383.xml
```

Sample response:

```
<Test_run>
    <Test_run_id>935e0080-97f5-0135-26f8-34363bc6aedc</Test_run_id>
    <Workflow_id>2383</Workflow_id>
    <Test_cases_launched>9</Test_cases_launched>
</Test_run>
```

Usage 2: Run all the test cases for a specific step, given a workflow ID and step name

```
https://Orchestrator_IP_address/aspera/orchestrator/api/test_step/workflow_id.xml?step=step_name
```

For example:

```
https://Orchestrator_IP_address/aspera/orchestrator/api/test_step/2383.json?
step=Get%20file%20size
```

Sample response:

```
{
    "Test_run": {
        "Test_run_id": "935e0080-97f5-0135-26f8-34363bc6aedc",
        "Workflow_id": "2383"
        "Test_cases_launched": 4
    }
}
```

Usage 3: Run a specific test case, given a workflow ID, step name, and the test template name

```
https://Orchestrator_IP_address/aspera/orchestrator/api/test_step/workflow_id.xml?
step=step_name&case=test_template_name
```

For example:

```
https://Orchestrator_IP_address/aspera/orchestrator/api/test_step/2383.json?
step=Get%20file%20size&case=File%20doesnt%exist
```

Example response:

```
{
    "Test_run": {
        "Test_run_id": "935e0080-97f5-0135-26f8-34363bc6aedc",
        "Workflow_id": "2383"
        "Test_cases_launched": 1
    }
}
```

You can add the an additional parameter, `synchronous=`*`true/false`*`.`

For example:

```
https://Orchestrator_IP_address/aspera/orchestrator/api/test_step/3283.json?
step=Get%20file%20size&case=File%20doesnt%exist&synchronous=true
```

Get the Status of Executed Test Cases for Workflow Steps

This endpoint gets the status of test cases for steps in a workflow.

Description:

This endpoint gets the status of test cases for steps in a workflow. Those test cases were previously executed with the `/test_step/`*`workflow_id`* method, which returns a `run_id` that you submit in the method described below. See "Run Test Cases for Workflow Steps" on page 217 for nmore information.

Note: The `run_id` mentioned above can be expressed as *`run_id`*`.xml` or *`run_id`*`.json`, as appropriate.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

Request:

```
https://Orchestrator_IP_address/api/test_step_status/test_run_id.xml
```

For example:

```
https://Orchestrator_IP_address/api/test_step_status/327f01f0-df92-0135-1920-3c15c2d9c4e6.xml
```

Sample response:

```
<?xml version="1.0"?>
<Test_run>
    <Test_run_id>327f01f0-df92-0135-1920-3c15c2d9c4e6</Test_run_id>
    <Workflow_id>2383</Workflow_id>
    <Test_cases_launched>3</Test_cases_launched>
    <Steps>
        <Step_name>New AmazonS3Operation Step</Step_name>
        <Test_cases>
            <Id>65</Id>
            <Name>test11</Name>
            <Status>Complete</Status>
            <Status_details>S3 presigned url generation completed</Status_details>
            <Result_message>No validation set for that test case</Result_message>
            <Last_update>Fri Jan 19 15:59:07 PST 2018</Last_update>
            <Inputs/>
            <Outputs>
                <Pre-signed_Url>https://
s3-us-west-2.amazonaws.com/orchT1/sampleabc?X-Amz-Algorithm=AWS4-HMAC-SHA256&amp;X-Amz-
Credential=AKIAI2LULP7KTP4RTVRQ%2F20180119%2Fus-west-2%2Fs3%2Faws4_request&amp;X-Amz-
Date=20180119T220117Z&amp;X-Amz-Expires=900&amp;X-Amz-SignedHeaders=host&amp;X-Amz-
Signature=65fa19a34e42055635dff8273ccfe2628908e5c089bdb9f35b07b4bb0c293236</Pre-signed_Url>
            </Outputs>
            <Result/>
        </Test_cases>
        <Test_cases>
            <Id>66</Id>
            <Name>t2</Name>
            <Status>Complete</Status>
            <Status_details>S3 presigned url generation completed</Status_details>
            <Result_message>No validation set for that test case</Result_message>
            <Last_update>Fri Jan 19 15:59:07 PST 2018</Last_update>
            <Inputs/>
            <Outputs>
                <Pre-signed_Url>https://
s3-us-west-2.amazonaws.com/orchT1/sampleabc?X-Amz-Algorithm=AWS4-HMAC-SHA256&amp;X-Amz-
Credential=AKIAI2LULP7KTP4RTVRQ%2F20180119%2Fus-west-2%2Fs3%2Faws4_request&amp;X-Amz-
Date=20180119T220117Z&amp;X-Amz-Expires=900&amp;X-Amz-SignedHeaders=host&amp;X-Amz-
Signature=65fa19a34e42055635dff8273ccfe2628908e5c089bdb9f35b07b4bb0c293236</Pre-signed_Url>
            </Outputs>
            <Result/>
        </Test_cases>
        <Test_cases>
            <Id>67</Id>
            <Name>t3</Name>
            <Status>Complete</Status>
            <Status_details>S3 presigned url generation completed</Status_details>
            <Result_message>No validation set for that test case</Result_message>
            <Last_update>Fri Jan 19 15:59:07 PST 2018</Last_update>
            <Inputs/>
            <Outputs>
                <Pre-signed_Url>https://
s3-us-west-2.amazonaws.com/orchT1/sampleabc?X-Amz-Algorithm=AWS4-HMAC-SHA256&amp;X-Amz-
Credential=AKIAI2LULP7KTP4RTVRQ%2F20180119%2Fus-west-2%2Fs3%2Faws4_request&amp;X-Amz-
Date=20180119T220117Z&amp;X-Amz-Expires=900&amp;X-Amz-SignedHeaders=host&amp;X-Amz-
Signature=65fa19a34e42055635dff8273ccfe2628908e5c089bdb9f35b07b4bb0c293236</Pre-signed_Url>
            </Outputs>
            <Result/>
        </Test_cases>
        <Result>true</Result>
    </Steps>
    <Result>true</Result>
</Test_run>
```

Publish a Workflow

This endpoint publishes a workflow so it can launch running instances (work orders).

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Request Example

```
GET http://localhost:3000/aspera/orchestrator/api/publish_workflow?id=627
```

Parameters

Parameter Name	Required/Optional	Data Type	Definition
id	required	integer	ID of the workflow to publish

Response Example

The response confirms that the workflow is published successfully:

```
<WorkflowPublish>
    <Success>true</Success>
</WorkflowPublish>
```

Get a List of All Workflows that Use a Specific Workflow as a Subworkflow

This endpoint gets a list of all workflows that use the specified workflow as a subworkflow.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Request Example

```
GET https://localhost/aspera/orchestrator/api/list_by_subworkflow?sub_wf_id=1
```

Parameters

Parameter Name	Required/Optional	Data Type	Definition
sub_wf_id	required	integer	ID of the workflow that is used by other workflows as a subworkflow

Response Example

The response returns the workflow IDs of workflows that use the specified workflow as a subworkflow.

```
<?xml version="1.0"?>
<ListBySubworkflow>
  <WorkflowId>2</WorkflowId>
  <WorkflowId>4</WorkflowId>
</ListBySubworkflow>
```

API Endpoints—Resources

Add a Resource

This endpoint adds a resource.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

Example request:

```
https://10.0.71.152/aspera/orchestrator/api/add_resource?
resource_name=127.0.0.7&pool_name=load_test&format=json
```

Response example:

```
{
  "ManagedResource": {
    "id": 2,
    "resource_id": "load_test_a->127.0.0.7",
    "resource_name": "load_testa",
    "max": 1,
    "reserved": null,
    "properties": "---\n:name: load_test_a\n:max: 1\n:resource: 127.0.0.7\n",
    "created_at": "2018-04-06T20:58:20.074Z",
    "updated_at": "2018-04-06T20:58:20.074Z",
    "status": null
  }
}
```

Edit Resource Capacity

This endpoint edits resource capacity.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

Request example:

```
https://10.0.71.152/aspera/orchestrator/api/edit_resource_capacity/4?
resource_name=127.0.0.7&pool_name=load_test&format=json
```

Response example:

```
{
  "ManagedResource": {
    "id": 2,
    "resource_id": "load_test_a->127.0.0.7",
    "resource_name": "load_testa",
    "max": 4,
    "reserved": null,
    "properties": "---\n:name: load_test_a\n:max: 4\n:resource: 127.0.0.7\n",
    "created_at": "2018-04-06T20:58:20.000Z",
    "updated_at": "2018-04-06T21:00:11.934Z",
    "status": null
  }
}
```

Bring a Resource Online

This endpoint brings a resource online.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

Example request:

```
https://10.0.71.152/aspera/orchestrator/api/resource_online?
resource_name=127.0.0.1&pool_name=load_test&format=json
```

Response example:

```
{
  "action": "resource_online",
  "time": "2018-04-06 21:12:22",
  "id": "127.0.0.1",
  "description": "Taking 127.0.0.1 online"
}
```

Take a Resource Offline

This endpoint takes a resource offline.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

Example request:

```
https://10.0.71.152/aspera/orchestrator/api/resource_offline?
resource_name=127.0.0.1&pool_name=load_test&format=json
```

Response example:

```
"error":{
  "action": "resource_offline",
  "id": "127.0.0.1",
  "description": "Taking 127.0.0.1 offline"
```

Pool Status

This API method gets the status of a resource pool—a group of resources, such as IP addresses and mongrels—so you can track resource usage data across workflows.

Usage (GET method)

Orchestrator uses basic authorization in the request.

Note: The default format for the response is XML format. To return a response in JSON format, add `format=json` to the request.

Note: You must use URL encoding for any special characters in parameter values, including username and password.

Note: You may have seen one or more pool names represented as `pool_name__a` in the database. This appended part of the pool name is always represented as *underscore underscore a* ("__a"). The request will be successful with or without the "__a". Either request below is acceptable and will return the same results:

```
https://Orchestrator_IP_address/aspera/orchestrator/api/pool_status?
name=pool_name&login=username
```

```
https://Orchestrator_IP_address/aspera/orchestrator/api/pool_status?
name=pool_name__a&login=username
```

For example:

```
https://10.0.71.99/aspera/orchestrator/api/pool_status?name=load_test&login=admin
```

```
https://10.0.71.99/aspera/orchestrator/api/pool_status?name=load_test__a&login=admin
```

Parameters

Parameter Name	Required	Data Type	Definition
name	true	string	Name of the resource pool

Response

XML Response

```
<PoolStatus>
   <ManagedResource>
      <id>127.0.0.7</id>
      <taken>0</taken>
      <capacity>1</capacity>
   </ManagedResource>
</PoolStatus>
```

JSON Response

```
{
  "ManagedResource": [
    {
      "id": "127.0.0.7",
      "taken": 0,
      "capacity": 1,
      "status": "online"
    }
  ]
}
```

Response Attributes

Parameter Name	Data Type	Definition	Example
`id`	string	ID of the resource	127.0.0.7
`taken`	integer	Number of resources currently in use	0
`capacity`	integer	Maximum number of potentially available resources	1

Edit Pool Capacity

This endpoint edits pool capacity.

Note: The default format for the response is XML format. To return a response in JSON format, add `&format=json` to the request.

Usage

Request example:

```
https://10.0.71.152/aspera/orchestrator/api/edit_pool_capacity/4?
pool_name=gen_pool_test&format=json
```

Response example:

```
{
  "ManagedResource": {
    "id": 2,
    "resource_id": "load_test_a->127.0.0.7",
    "resource_name": "load_testa",
    "max": 4,
    "reserved": null,
    "properties": "---\n:name: load_test_a\n:max: 4\n:resource: 127.0.0.7\n",
    "created_at": "2018-04-06T20:58:20.000Z",
    "updated_at": "2018-04-06T21:00:11.934Z",
    "status": null
  }
}
```

Appendix

Orchestrator Directory Locations

The list below provides the location of each directory used by Orchestrator.

Directory Function	Location
Installation	`/opt/aspera/orchestrator/`
Configuration	`opt/aspera/var/config/orchestrator/`
Run time	`/opt/aspera/var/run/orchestrator/`
Archive	`/opt/aspera/var/archive/orchestrator/`
Log	`/opt/aspera/var/run/orchestrator/log/orchestrator.log`
License file (use either of these locations)	`/opt/aspera/orchestrator/config/`*`client_id`*`-orchestrator.aspera-license` `/opt/aspera/var/config/orchestrator/licenses/`*`client_id`*`-orchestrator.aspera-license`

Error Codes in Orchestrator

For each failure in the Orchestrator API, the system returns an numerical error code and a text-based message specific to the type of error. The most common HTTP error code is HTTP `200 Ok`, along with a response body in XML or JSON.

The table below describes the type of error corresponding to each error code.

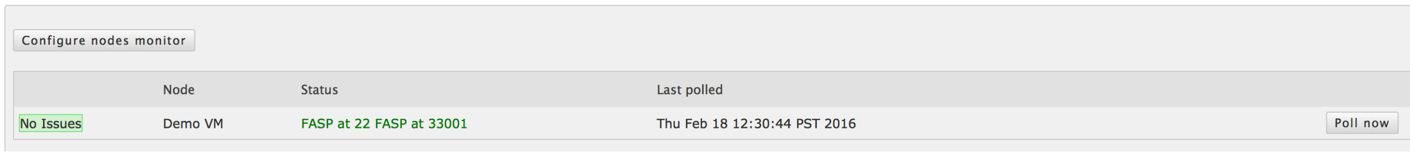
Error Code	Error
`200`	XML Format: `<?xml version="1.0"?>` `<error time="<--Timestamp -->" action="<-- action -->" id="<-- id -->">` `<-- error_message -->` `</error>`
`200`	JSON Format: `{error:` `{"action":"#{action}","time":"#{Timestamp}","id":"#{id}","description":#{error_message` `}}}"`
`400`	A bad request is sent; with invalid data, for example.
`401`	Authentication fails with the provided user/password, user/api_key, user/`basicauth`, or user/scrambled password.
`403`	Authorization fails with the provided user/password, user/api_key, user/`basicauth`, or user/scrambled password.

Error Code	Error
`404`	The user provides an incorrect URL.
`502`	An Orchestrator mongrel (UI rendering daemon) is terminated due to the longevity of the request. Clients see a Proxy error.
`503`	Orchestrator mongrels (UI rendering daemons) are not running.
`connection refused`	The Apache service is stopped.

Remote Node Status Notifications

In the Node list (**Monitor > Nodes**) you can view status alerts at the listing for the new remote node that appears.

If a connectivity test is successful for a particular port, a green `No Issues` label displays, which means that tests on all the ports are responding, and the Status for the port appears in green text. For example, in the notification below, SSH connectivity and FASP connectivity on their respective ports are listening and active.



	Node	Status	Last polled
No Issues	Demo VM	FASP at 22 FASP at 33001	Thu Feb 18 12:30:44 PST 2016

If connectivity for a port fails, a red Alerts label appears, and the Status is displayed with red text.

Warning labels appear in yellow background.

The screen has three buttons, as follows:

Poll now
Performs a force poll of the monitor.

Deactivate
Deactivates the monitor. The node monitor no longer polls for connectivity tests and the status changes to `Not running`.

Edit
Provides options for editing the monitor configuration settings (changing the name of the monitor, updating the notification type from `Alerts` to `Warnings` for a port, changing the test type).

Note: A remote node associated with a particular node monitor cannot be changed once it is added to the remote node monitor. If you want to change the remote node, you have to create a new node monitor and configure the remote node in that new node monitor.

Firewall Rules

Aspera Products

From	To	Port	Protocol	Functionality	Direction
Orchestrator	Enterprise server	40001	TCP	API Requests	Inbound to Enterprise server
Orchestrator	P2P Server	40001	TCP	API Requests	Inbound to P2P server
Orchestrator	Console	4406	TCP	Reporting	Inbound to Console
Orchestrator	Aspera Node API	9091, 9092	TCP	API Requests	Inbound to Aspera Nodes
Orchestrator	Aspera Faspex	443	TCP	API Requests	Inbound to Aspera Nodes

From	To	Port	Protocol	Functionality	Direction
Orchestrator	Aspera Shares	443, 9092	TCP	API Requests	Inbound to Aspera Nodes

3rd party transcoders

From	To	Port	Protocol	Functionality	Direction
Orchestrator	Carbon Transcoder	1120, 1301	TCP	API Requests	Inbound to destination
Orchestrator	Rhozet Transcoder	8731	TCP	API Requests	Inbound to destination
Orchestrator	Elemental Transcoder	80, 443			Inbound to destination
Orchestrator	FlipFactory Transcoder	80, 443			Inbound to destination
Orchestrator	Vantage Transcoder	8676	TCP	API Requests	Inbound to destination
Orchestrator	Digital Rapids Stream	44000	TCP	Reporting	Inbound to destination
Orchestrator	Digital Rapids TM	44000			

3rd party QC tools

From	To	Port	Protocol	Functionality	Direction
Orchestrator	Baton QC	8080	TCP	API Requests	Inbound to destination
Orchestrator	Aurora QC	1001	TCP	API Requests	Inbound to destination
Orchestrator	Cerify QC	80	TCP	Reporting	Inbound to destination

General tools

From	To	Port	Protocol	Functionality	Direction
Orchestrator	Email relay	25	SMTP	Email	Inbound to destination
Orchestrator	Database	4406	TCP	MySQL queries	Inbound to destination
Orchestrator	Database	1433	TCP	MSSQL queries	Inbound to destination
Orchestrator	Database	1521	TCP	Oracle queries	Inbound to destination

Portlets

From	To	Port	Protocol	Functionality	Direction
Aspera connect client	Aspera Transfer server	22 or 33001	TCP	Authentication	Inbound to destination
Aspera connect client	Aspera Transfer server	33001	UDP	Transfer	Inbound to destination

Creating a Basic Token for Transfers to Cloud Storage

The Aspera on Cloud transfer service enables you to run high-speed uploads and downloads to your own cloud storage (on AWS S3, IBM Cloud, Google Cloud, and Azure). Aspera on Cloud does not come with your MQ server; a separate entitlement is required.

Transferring with cloud storage requires a Basic token for authentication. The following instructions describe how to use the Aspera on Cloud UI to create the access key and how to use the command line to transform the access key id and secret into a Basic token.

1. Create the Aspera on Cloud transfer service access key by adding your cloud storage as a node to Aspera on Cloud.

   For instructions, see Creating a New Transfer Service Node in the Aspera on Cloud Help Center.

   Record the access key ID and secret that are created.
2. Create a basic token by encoding the `access_key_id:secret` in base64.

   ```
   $ echo -n diDeuFLcpG9IYdsvxj0SCq4mOohNJTKvp5Q2nRWjDgIA:aspera | base64
   ```

   The basic token looks similar to the following:

   ```
   ZGlEZXVGTGNwRzlJWWRzdnhqMFNDcTRtT29oTkpUS3ZwNVEyblJXakRnSUE6YXNwZXJh
   ```

   If the basic token breaks across lines in the output, rerun the command using the `-w0` option to remove the line break. For example:

   ```
   $ echo -n diDeuFLcpG9IYdsvxj0SCq4mOohNJTKvp5Q2nRWjDgIA:aspera | base64 -w0
   ```

TCP and UDP Ports Used in Orchestrator High Availability Environments

The table below lists the ports that must be open in order for the Orchestrator High Availability environment to operate correctly:

Port	Direction	Service
TCP-80	From web clients to the VIP of the load balancer	load balancer
TCP-80	From the load balancer to the Orchestrator nodes *(if the load balancer does not take care of the HTTP to HTTPS redirection)*	**asperahttpd**
TCP-443	From web clients to the VIP of the load balancer	load balancer
TCP-443	From the load balancer to the Orchestrator nodes	**asperahttpd**
TCP-4406	Between the two Orchestrator nodes	**aspera_mysqld**

List of System Commands Used by Aspera Cluster Manager (ACM)

The following commands must be available on any Linux system running ACM:

```
bash date
 sleep
usleep
sed find
grep
hostname
tee
touch
readlink
crontab
expr
stat let
```

```
lgtogger nc
readlink ip
gzip
chkconfig
which
sestatus
```