# HYBRID WORKFLOW USING PIXAR'S TRACTOR

Jesse Lehrman | Senior Pipeline Developer | Conductor Technologies

CONDUCTOR

## CONTENTS

INTRODUCTION 3

USING A CONDUCTOR SUPER-BLADE 5

Overview 5

Advantages 6

Disadvantages 6

Implementation Details 6

Requirements 6

Submitting to Conductor 6

Polling Conductor 7

Custom Tractor Environment Handler 8

Configuring the Super-Blade 9

Adding a Menu Item 11

Dependency Scanning 11

Next Steps 11

CONCLUSION 13

## INTRODUCTION

Conductor ships with a set of plugins to allow artists to directly submit scenes to be rendered on the Conductor platform. This allows users to get up and running very quickly and satisfies the requirements for individual artists or smaller studios.

When looking at larger studios, there is a desire to have artists focus on their specialized tasks. Decisions about *where* to render need to fall onto render wranglers and production. Artists have very little need to know anything about the render farm at all. Their interest is in submitting the renders and getting the results quickly.

This paper outlines how, on a technical level, to integrate Conductor into Pixar's Tractor, a popular render farm management system. The goal of this paper is to present a method to integrate Conductor into an existing render farm management system so that studios can maintain their current procedures and policies, sheltering artists from the decisions of the *ifs* and *hows* of submitting a job to Conductor.

Given that there is little standardization in the industry, this paper doesn't offer a turnkey solution for integration. It's a guide with concrete code examples on how to accomplish the integration. A lot of the code and configuration can be used as is.

Where the code will likely need to be modified, it's clearly indicated with the following icon:

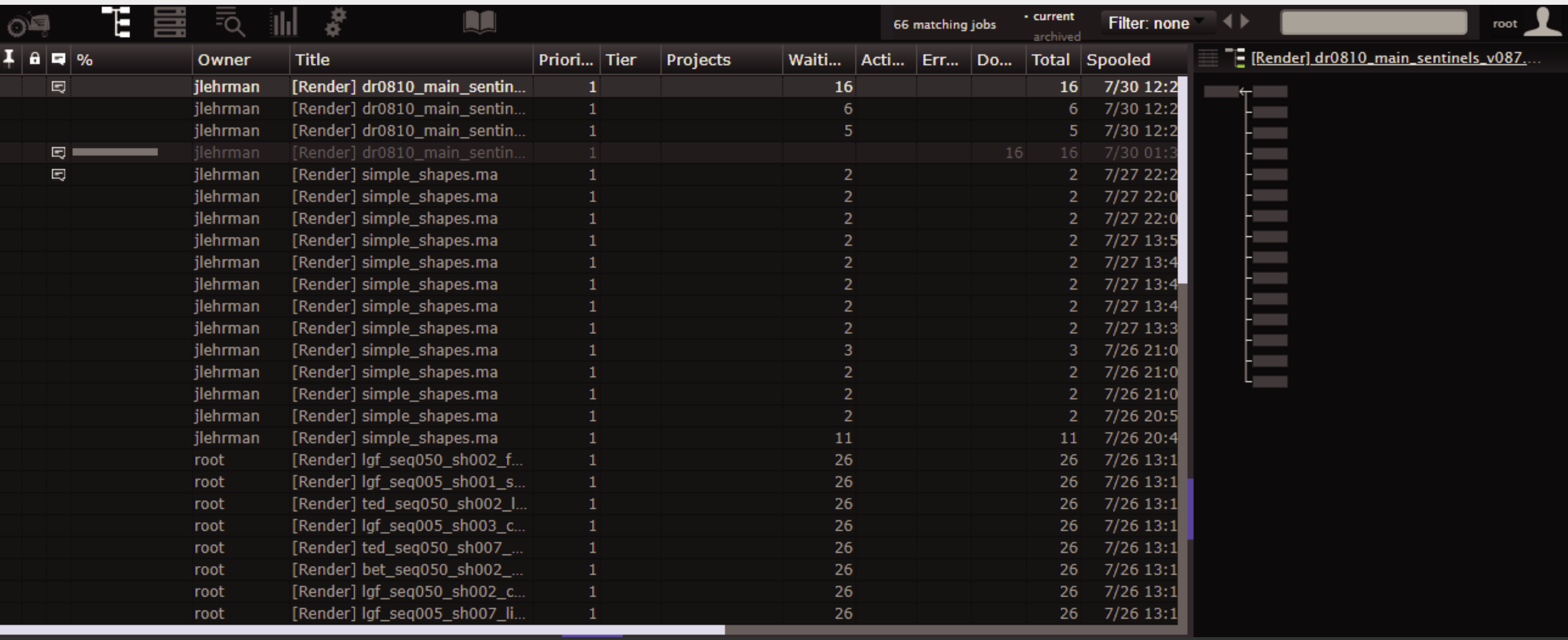
### NOTE

*Can I run Tractor-blade on a Conductor instance?*

We are currently exploring this workflow. For more info, please contact: info@conductortech.com.

Throughout this guide, a Maya render job with a simple hierarchy is used. This is for simplicity purposes; the techniques outlined should work with the most complex jobs.

Additionally, many studios separate the exporting of render scene descriptions (.ass, .rib, .vrscene, *etc.*) from the rendering process. While Conductor doesn't offer this functionality out of the box, it is possible but beyond the scope of this paper.



| | | | % | Owner | Title | Priori... | Tier | Projects | Waiti... | Acti... | Err... | Do... | Total | Spooled |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 🗨 | | jlehrman | [Render] dr0810_main_sentin... | 1 | | | 16 | | | | 16 | 7/30 12:2 |
| | | | | jlehrman | [Render] dr0810_main_sentin... | 1 | | | 6 | | | | 6 | 7/30 12:2 |
| | | | | jlehrman | [Render] dr0810_main_sentin... | 1 | | | 5 | | | | 5 | 7/30 12:2 |
| | | 🗨 | ▬ | jlehrman | [Render] dr0810_main_sentin... | 1 | | | | | | 16 | 16 | 7/30 01:3 |
| | | 🗨 | | jlehrman | [Render] simple_shapes.ma | 1 | | | 2 | | | | 2 | 7/27 22:2 |
| | | | | jlehrman | [Render] simple_shapes.ma | 1 | | | 2 | | | | 2 | 7/27 22:0 |
| | | | | jlehrman | [Render] simple_shapes.ma | 1 | | | 2 | | | | 2 | 7/27 22:0 |
| | | | | jlehrman | [Render] simple_shapes.ma | 1 | | | 2 | | | | 2 | 7/27 13:5 |
| | | | | jlehrman | [Render] simple_shapes.ma | 1 | | | 2 | | | | 2 | 7/27 13:4 |
| | | | | jlehrman | [Render] simple_shapes.ma | 1 | | | 2 | | | | 2 | 7/27 13:4 |
| | | | | jlehrman | [Render] simple_shapes.ma | 1 | | | 2 | | | | 2 | 7/27 13:4 |
| | | | | jlehrman | [Render] simple_shapes.ma | 1 | | | 2 | | | | 2 | 7/27 13:3 |
| | | | | jlehrman | [Render] simple_shapes.ma | 1 | | | 3 | | | | 3 | 7/26 21:0 |
| | | | | jlehrman | [Render] simple_shapes.ma | 1 | | | 2 | | | | 2 | 7/26 21:0 |
| | | | | jlehrman | [Render] simple_shapes.ma | 1 | | | 2 | | | | 2 | 7/26 21:0 |
| | | | | jlehrman | [Render] simple_shapes.ma | 1 | | | 2 | | | | 2 | 7/26 20:5 |
| | | | | jlehrman | [Render] simple_shapes.ma | 1 | | | 11 | | | | 11 | 7/26 20:4 |
| | | | | root | [Render] lgf_seq050_sh002_f... | 1 | | | 26 | | | | 26 | 7/26 13:1 |
| | | | | root | [Render] lgf_seq005_sh001_s... | 1 | | | 26 | | | | 26 | 7/26 13:1 |
| | | | | root | [Render] ted_seq050_sh002_l... | 1 | | | 26 | | | | 26 | 7/26 13:1 |
| | | | | root | [Render] lgf_seq005_sh003_c... | 1 | | | 26 | | | | 26 | 7/26 13:1 |
| | | | | root | [Render] ted_seq050_sh007_... | 1 | | | 26 | | | | 26 | 7/26 13:1 |
| | | | | root | [Render] bet_seq050_sh002_... | 1 | | | 26 | | | | 26 | 7/26 13:1 |
| | | | | root | [Render] lgf_seq050_sh002_c... | 1 | | | 26 | | | | 26 | 7/26 13:1 |
| | | | | root | [Render] lgf_seq005_sh007_li... | 1 | | | 26 | | | | 26 | 7/26 13:1 |

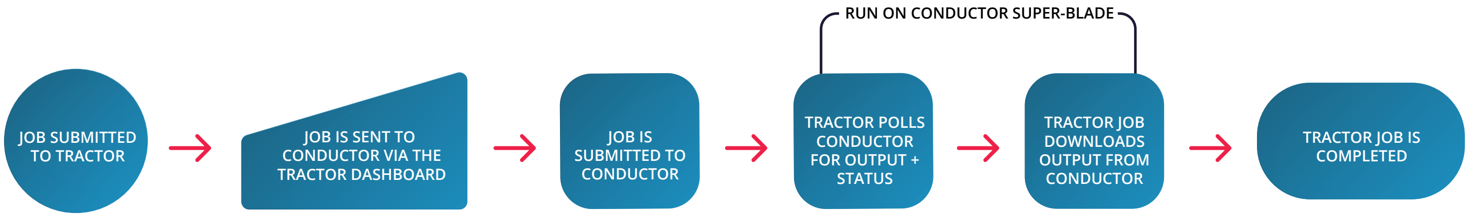
All code examples in this paper are snippets of larger scripts.
These can all be downloaded here.

# USING A CONDUCTOR SUPER-BLADE

## Overview

This method of integrating Conductor into Tractor involves the use of a *Conductor super-blade*. A *Conductor super-blade* has an infinite[1] number of slots. This blade polls Conductor for jobs and pipes the logs directly back to Tractor. A job is submitted to Conductor when a user selects "Send to Conductor" from a context menu in the Tractor dashboard and then the Tractor job is assigned to the *Conductor super-blade*.

With this method, no new Tractor tasks are created. A custom environment handler is used to swap the command being executed.


RUN ON CONDUCTOR SUPER-BLADE
JOB SUBMITTED TO TRACTOR
JOB IS SENT TO CONDUCTOR VIA THE TRACTOR DASHBOARD
JOB IS SUBMITTED TO CONDUCTOR
TRACTOR POLLS CONDUCTOR FOR OUTPUT + STATUS
TRACTOR JOB DOWNLOADS OUTPUT FROM CONDUCTOR
TRACTOR JOB IS COMPLETED

### NOTE

Many studios have unique Tractor job hierarchies. The job structure used in this document is kept minimal for demonstration purposes. See the job matching function on how to accommodate a studio's unique job hierarchy.

[1] In practice, a very high number (ex: 99999) is used

## Advantages

- Achieved solely via Tractor configuration, plugins and external scripts.
- Completely transparent to artists
- No additional Tractor jobs or tasks required

## Disadvantages

- Dependency scanning is performed for non-Conductor jobs
- Requires an on-site resource for the *Conductor super-blade*
- Conductor jobs still need to be managed from the Conductor dashboard

## Implementation Details

### Requirements

- Conductor Client API
- Tractor Query Python API

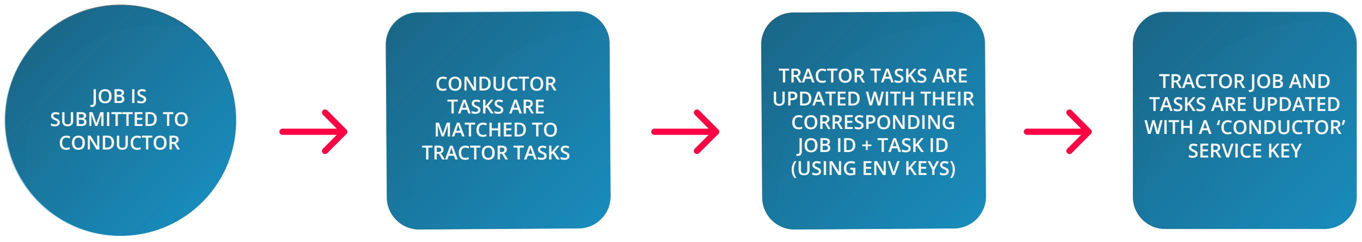
### Submitting to Conductor

This script takes a list of Tractor jobs as input and submits them to Conductor. There are two Python modules used:

- `ConductorJob` - *Wraps the Conductor submission process. Child classes exist for different job types.*
- `JobCommand` - *Parses and obtains details about a command-line string.*

For more details on these modules, please see the code-level documentation.

The basic flow of this script is:


JOB IS SUBMITTED TO CONDUCTOR
CONDUCTOR TASKS ARE MATCHED TO TRACTOR TASKS
TRACTOR TASKS ARE UPDATED WITH THEIR CORRESPONDING JOB ID + TASK ID (USING ENV KEYS)
TRACTOR JOB AND TASKS ARE UPDATED WITH A 'CONDUCTOR' SERVICE KEY

This submission part of this script is not so different than submitting any other job to Conductor. The only thing that is unique is that the Conductor job parameters are constructed from a Tractor job. It also allows for overriding job parameters by using environment variables. This script is intended to be triggered by a new Tractor menu command. Due to this, it's important to add a shebang with the path to the Python executable at the top.

```
#!/usr/bin/python

# Read data that is passed by the Tractor menu command
json_data = sys.stdin.read()

if json_data:
  jobs = json.loads(json_data)

  for job in jobs:

    tractor_job_id = job['jid']

    submitter = conductorjob.ConductorJob.create_from_tractor_job(job_id=tractor_job_id)
    conductor_job_id = submitter.submit_job()

    update_tractor_tasks(tractor_job_id, conductor_job_id)
```

The last method, `update_tractor_tasks()` updates the Tractor job entity so that it has the correct service key that will assign it to the *Conductor super-blade*. Tractor command entities are updated so that they have the environment keys `conductor_jid` and `conductor_tid` which link them to a unique job and task in Conductor. These environment keys are converted into environment variables by the custom `EnvironmentHandler` and then used by the polling script.

The function `match_tractor_commands_to_conductor_tasks()` is critical and matches the Tractor command to the corresponding Conductor task. This function will likely need to be modified to suit the needs of a specific studio. In its current form, it matches based on the start and end frames. If a job contains multiple layers, passes, different types of tasks, *etc*... these will have to be accomodated. Note that the return structure is also quite specific (and simplistic). It's advisable to use a purposefully built class in its place.

## Polling Conductor

This script will poll a specific Conductor task for its status and log. Once the log is available, it will be downloaded and printed so that Tractor can parse as if the job was local. This can help with Tractor's progress and status indicators. Once the job is complete, all the files will be downloaded. An instance of the polling script is run on the *Conductor super-blade* for each Conductor task.

```
while True:

  # Get the task status
  response = json.loads(get_job_status(job_label, task_label))
  task = response['data'][0]
  task_status = task['status']

  if task_status in TASK_LOG_READY_STATUSES:

    # Get any new lines in the log
    json_response = get_task_log( job_id=job_label,
                   task_id=task_label,
                   first_line=first_log_line )
    log_response = json.loads(json_response)

    try:
      first_log_line += int(log_response['new_num_lines'][0])
      print "\n".join(log_response['logs'][0]['log'])

    # Don't fail if there's an issue with the log
    except Exception, errMsg:
      LOG.warning(str(errMsg))
      continue

  if task_status in TERMINATE_STATUSES:
    break

  # If the task failed, there's nothing left to do.
  if task_status in ['failed']:
    raise Exception("Job failed")

  time.sleep(poll_interval)
```

## Custom Tractor Environment Handler

A custom environment handler is used to set certain environment variables and to replace the Tractor command argument with the polling script. For more details on Tractor's custom environment handlers, please see the Tractor documentation.

The custom environment handler class has two methods; `updateEnvironment()` and `remapCmdArgs()`. The Python file needs to be in a location that's added to the `SiteModulesPath` in `blade.config` (see the next section for details).

In `updateEnvironment()`, a loose method is used to map any conductor environment keys to environment variables. This will allow additional environment keys to be added without requiring to modify the handler.

```
def updateEnvironment(self, cmd, env, envkeys):

  for envkey in envkeys:
    if envkey.startswith("conductor_"):
      key, value = envkey.split("=")
      env[key.upper()] = value

  return TrEnvHandler.updateEnvironment(self, cmd, env, envkeys)
```

In `remapCmdArgs()`, the current command is completely replaced with a call to the polling script from the previous step. No arguments are necessary since all the parameters are passed via environment variables.

```
def remapCmdArgs(self, cmdinfo, launchenv, thisHost):

  return ["python", "conductor_poll.py"]
```

## Configuring the Super-Blade

A custom blade profile must be added to `blade.config` in order for the machine to accept Conductor jobs as well as use the custom environment handler created in the previous step. The `ConductorSuperBlade` profile should be placed before any other profiles that might also match that blade. In this case, the blade would also match the `Linux64` profile.

Ensure that the `Hosts` parameter properly matches the hostname of the machine you will be using as the *Conductor super-blade.*

The `Provides` parameter must match the service key set on the Tractor job in the Conductor submit script.

`MaxSlots` should be an arbitrary high number. This won't set a limit on how many jobs can be sent and run on Conductor, it will limit the number of polling tasks running on the *Conductor super-blade*.

```
"BladeProfiles":
[
  {
    "ProfileName": "ConductorSuperBlade",
    "Hosts": {"Name": "conductor*"},
    "Provides": ["Conductor"],
    "EnvKeys": [ "@merge('shared.linux.envkeys')"],
    "Capacity": {
      "MaxSlots": 5000,
      },
  },
  {
    "ProfileName": "Linux64",
    "Hosts": {"Platform": "Linux-*64bit*"},
    "EnvKeys": [ "@merge('shared.linux.envkeys')" ]
  }
]
```

## NOTE

The `blade.config` file also contains an entry for the `SiteModulesPath`. This should include the paths to the custom environment handler as well as any additional Python packages.

In addition, environment keys need to be defined to connect them to the custom environment handler. Here's a sample environment keys file `(shared.linux.envkeys)`:

```
[
  {
    "keys": ["default"],
    "envhandler": "default"
    "environment": {},
  },
  {
    "keys": ["conductor_jid*", "conductor_tid*"],
    "envhandler": "ConductorEnvHandler",
    "environment": {}
  }
]
```

## Adding a Menu Item

A convenient method is required to execute the Conductor submit script. Outlined below is an example of adding a menu item to the Tractor dashboard to do the submission.

In the `menu.config` file, add an entry under the job entity. The `exec` parameter must be an absolute path to the script otherwise it's considered to be relative to the site's configuration directory. Ensure permissions are correct on `submit_to_conductor.py` - it will be executed by the same user running `tractor-engine`.

For more details on adding a menu item, please see the Tractor documentation.

```
"job":
[
 {
  "comment": "Set a job and all its tasks to be rendered on conductor",
  "title": "Send to Conductor",
  "exec": ["submit_to_conductor.py"],
  "suppress": true,
  "values": ["jid"],
  "enabled": true
 }
]
```

## Dependency Scanning

This method does not provide a discrete step for dependency scanning. While there are options to perform the scan at the time of submission, this is not ideal as it would be the `tractor-engine` machine performing the work.

While not ideal, the best option would be to perform the scan when the user submits their scene to get rendered, saving the result in a json sidecar file that can be read at the time of submission.

## Next Steps

Using the *Conductor super-blade* method to integrate Conductor into Tractor provides a robust way to give control of the Conductor submission process to render wranglers and production. There are several features that can be added to augment the experience. Below are a few suggestions.

- Automate which jobs get sent to Conductor by querying a production management system
- Export a render scene description file (.rib, .ass, etc...) locally and render in Conductor
- Add additional menu items to directly control Conductor jobs and tasks

NOTE

In this method, there's no mention of using the uploader/downloader daemons. If the uploader is running, it will offload the file transfer from the submit script – which is desirable. If the downloader daemon is running, it will offload the file transfer from the polling script which is desirable in terms of performance, but undesirable from the point of job transparency (Tractor won't be able to indicate when the rendered files are available to the user).

## CONCLUSION

Properly integrating Conductor into a studio is key to getting the most from the platform. With an established hybrid workflow, studios have a clear path to integrate the platform without the need to alter their workflow or worry about the potential issues of having artists submit jobs directly.

Conductor is an attractive platform to expand your rendering capabilities, and with the included code samples, it can help a studio evaluate the effort required for integration and eventually deploy a seamless cloud enhancement to their existing workflow.

For more information on this, or other related topics:

contact info@conductortech.com
or visit support.conductortech.com

CONDUCTOR