HTCondor Clients in 2012

**Command Line Clients**
- Fully Featured!
- Requires fork/exec and process handling
- Outputs in multiple formats

**Soap Clients**
- Features! (Some)
- **Language agnostic** (everyone hates XML equally?)
- Caveats with respect to scalability, security.

Something Missing In The Middle
Design Philosophy

- **ClassAds**: Everything based on ClassAds; make these the “core” of the bindings.

- **pythonic**: Semantics and APIs should feel natural to a python programmer.
  - Use iterators, exceptions, guards. ClassAds behave as much like a dict as reasonable.

- **Backward compatible**: APIs are here to stay for as long as possible.
  - When we absolutely must, use standard python `DeprecationWarning` techniques.

- **Native code**: Call same HTCondor library code as CLI; identical in performance.

- **Complete**: If you can do it with the command line tools, you should be able to do it with python.
Pythonic!

• Since *pythonic* is in our design philosophy, I decided the education should use the tools favored by the python community:

  • **Sphinx**-based documentation. Hosted on ReadTheDocs; looks / feels / smells like python documentation. Updated, more complete, and contains *tutorials*: not just a reference!

  • **JupyterHub**-based tutorials. Login with a university credential; spawns a Docker container with a private HTCondor instance. Interact via your browser.
Sphinx Docs

https://htcondor-python.readthedocs.io

HTCondor Python module documentation

The HTCondor python modules aim to expose a high-quality, Pythonic interface to the HTCondor client libraries. They utilize the same C++ libraries as HTCondor itself, meaning they have nearly the same behavior as the command line tools. As the python modules are shipped with HTCondor itself, this documentation focuses on references and tutorials on how to use the modules, rather than an install how-to.

Note

This work should be considered experimental (its aim is to be better-formatted and more intuitive); the upstream documentation should be considered authoritative.
The HTCondor python bindings provide a powerful mechanism to interact with HTCondor from a python program. They utilize the same C++ libraries as HTCondor itself, meaning they have nearly the same behavior as the command line tools.

Here, you will learn the basics of the python bindings and how to use them. This tutorial is broken down into two major sections.
Interacting With Daemons

In this module, we'll look at how the HTCondor python bindings can be used to interact with running daemons.

Let's start by importing the correct modules:

```
In [2]: import htcondor
```

Configuration

The HTCondor configuration is exposed to Python in two ways:

- The local process's configuration is available in the module-level param object.
- A remote daemon's configuration may be queried using a RemoteParam.
Terminal View

[jovyan@331dd87f439f work]$ condor_q

-- Schedd: jovyan@331dd87f439f : <172.17.0.2:9618>?... @ 04/29/17 19:42:45
OWNER BATCH_NAME SUBMITTED DONE RUN IDLE HOLD TOTAL JOB_IDS
0 jobs; 0 completed, 0 removed, 0 idle, 0 running, 0 held, 0 suspended
[jovyan@331dd87f439f work]$ condor_run echo "hello world"

hello world

[jovyan@331dd87f439f work]$ condor_status echo "hello world"

Name OpSys Arch State Activity LoadAv Mem ActvtyTime
331dd87f439f LINUX X86_64 Unclaimed Idle 0.050 3539 0+00:00:02

Machines Owner Claimed Unclaimed Matched Preempting Drain
X86_64/LINUX 1 0 0 1 0 0 0
Total 1 0 0 1 0 0 0

[jovyan@331dd87f439f work]$
You can help!

• The contents of the tutorials and documentation are kept on GitHub:
  
  • [https://github.com/bbockelm/htcondor-python/](https://github.com/bbockelm/htcondor-python/)

  • I am interested in partnering with someone to help make the Jupyter service Docker-ized also!

• Find a bug? Spot some missing content?

  • Simply send a pull request; Travis-CI will test and update the static content once merged.
Let’s Proceed!

https://hcc-htcondor-python.unl.edu