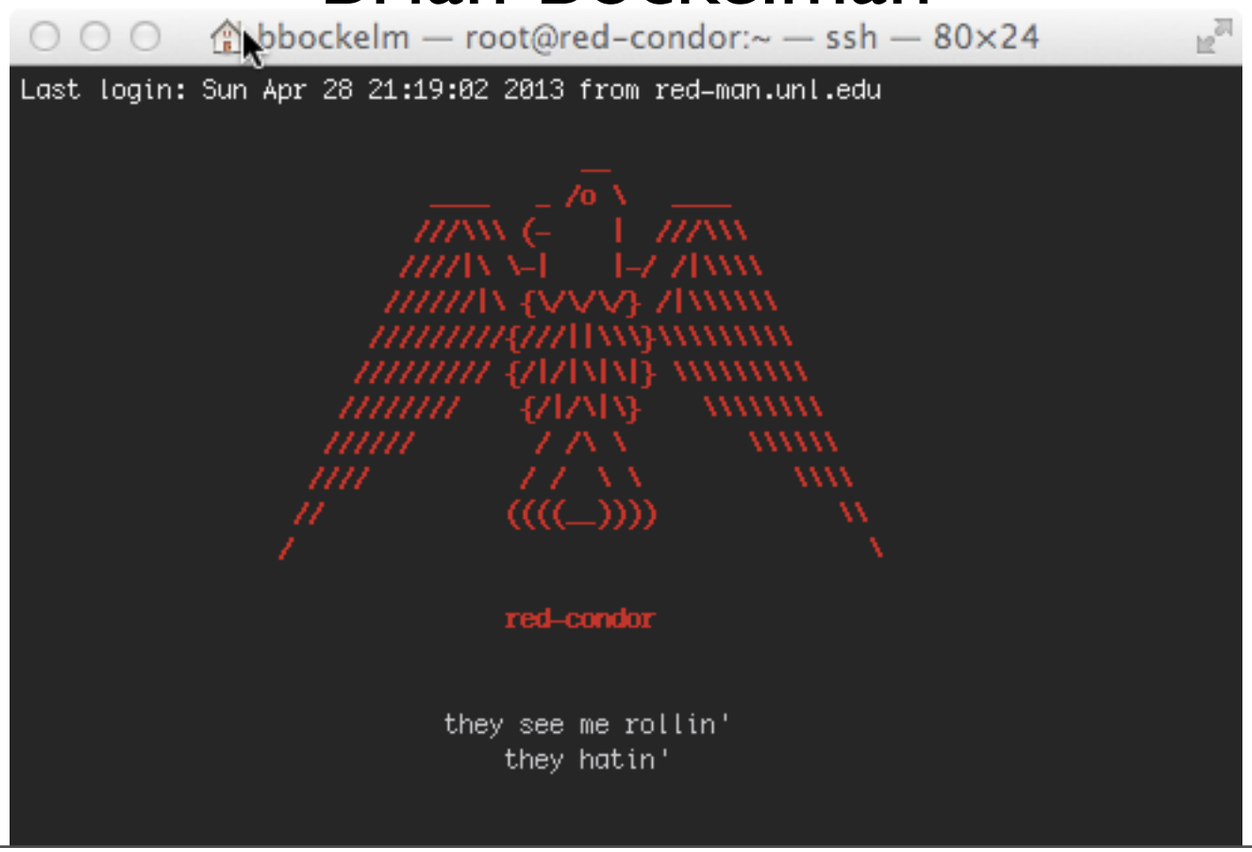
HTCondor Python Tutorial

Brian Bockelman



Welcome, Python!

- HTCondor has long provided two APIs into its ecosystem:
  - **Traditional POSIX API**: Input via stdin / argv, Output via stdout, error via exit codes.
  - **SOAP**: RPC-oriented API. Language-agnostic.
- Both have drawbacks. Fork overhead, parsing overhead, no reuse of security sessions.
- Python is a special case - widely utilized by projects in which build on top of HTCondor.
  - ... and Boost.Python makes it straightforward to write/maintain bindings...
  - HTCondor includes python bindings for most client-side activities since 7.9.4.

Audience!

- The audience for the python bindings are integrators/developers - we consciously expose lower-level interfaces than the CLI.
  - If you want a more straightforward way to interact with HTCondor via python, this tutorial is for you!
  - Some are decently refined; some are pretty raw wrappers around C++.
- These are one of the most powerful ways of programmatically interacting with the system.
  - Not the simplest. (Yet?)
- I assume basic python and intermediate/advanced HTCondor knowledge.

Tutorial Time

- In this tutorial, I plan on covering the *basics* of using the python bindings.
- You'll need your Linux-based laptop out with a fresh install of HTCondor >= 7.9.5.
  - Startup a personal HTCondor instance. Verify you can run basic commands (condor_submit, _status, _q).
- For the most part, this will be "follow along Brian's terminal", but slides are here for later students.
  - (And in case the network connection explodes.)

Login yourself

- Hostname: ec2-54-224-238-91.compute-1.amazonaws.com
- User: demo
- Pass: theHTissilent

Hello, (HTCondor) World

bbockelm — Test Terminal — Python — 80×24

```
Last login: Tue Apr 23 21:10:11 on ttys001
Brians-MacBook-Air:~ bbockelm$ python
Python 2.7.2 (default, Jun 20 2012, 16:23:33)
[GCC 4.2.1 Compatible Apple Clang 4.0 (tags/Apple/clang-418.0.60)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import htcondor
>>> import classad
>>> a = classad.ClassAd({"Hello": "World"})
>>> a
[ Hello = "World" ]
>>>
```

Python Basics

- import htcondor; import classad
- Use dir() to list object names in a module; use help() to get the per-method or class help.
- print classad.version(), htcondor.version()
- htcondor.param['COLLECTOR_HOST'] to access parameter value of COLLECTOR_HOST.

In the beginning, there were ClassAds.

- ClassAds are the *lingua franca* of HTCondor-land.
- Condor CLI often converts ClassAds into human readable form or XML.
- The python bindings use the internal ClassAd objects throughout.
- ClassAds may look like bastardized JSON, but there are important evaluation semantics we can take care of.
- We try to make it pleasant to convert between ClassAds and native Python objects.

ClassAds

bbockelm — Test Terminal — Python — 80×24

```
Last login: Wed Apr 24 07:14:51 on ttys007
Brians-MacBook-Air:~ bbockelm$ python
Python 2.7.2 (default, Jun 20 2012, 16:23:33)
[GCC 4.2.1 Compatible Apple Clang 4.0 (tags/Apple/clang-418.0.60)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import htcondor
>>> import classad
>>> a = classad.ClassAd({"Hello": "World"})
>>> a
[ Hello = "World" ]
>>> a["foo"] = "bar"
>>> a
[ foo = "bar"; Hello = "World" ]
>>> a["bar"] = [1,2,3]
>>> a
[ bar = { 1,2,3 }; foo = "bar"; Hello = "World" ]
>>> a["baz"] = classad.ExprTree("bar")
>>> a
[ baz = bar; bar = { 1,2,3 }; foo = "bar"; Hello = "World" ]
>>> a["baz"]
bar
>>> a.eval("baz")
[1, 2, 3]
>>>
```

bbockelm — Test Terminal — Python — 80×24

```
Last login: Wed Apr 24 12:29:11 on ttys009
Brians-MacBook-Air:~ bbockelm$ python
Python 2.7.2 (default, Jun 20 2012, 16:23:33)
[GCC 4.2.1 Compatible Apple Clang 4.0 (tags/Apple/clang-418.0.60)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import classad
>>> a = classad.ClassAd()
>>> a["foo"] = classad.ExprTree("bar")
>>> a["foo"]
bar
>>> a.eval("foo")
classad.Value.Undefined
>>> a.eval("bar")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'bar'
>>>
```

Sub-ClassAds are supported too!

```
>>> classad.ClassAd({"foo": {"bar": True}})
[ foo = [ bar = 1 ] ]
>>>
```

HTCondor Module

- The "htcondor" Python module allows you to interact with most HTCondor daemons.
- There are two very important objects:
  - Collector: read and write
  - Schedd: submit and manipulate
- And a few other helpers - enums, security manager, interaction with the config system, and sending daemons direct commands.

Collector Basics

- The Collector object allows one to locate daemons, query slot status, and advertise new ClassAds.
- The object takes the network location of the collector daemon for the constructor:
  - coll = htcondor.Collector("red-condor.unl.edu")

Collector Basics

bbockelm — Test Terminal — Python — 80×24

```
Last login: Sat Apr 27 11:26:34 on ttys004
Brians-MacBook-Air:~ bbockelm$ python
Python 2.7.2 (default, Jun 20 2012, 16:23:33)
[GCC 4.2.1 Compatible Apple Clang 4.0 (tags/Apple/clang-418.0.60)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import htcondor
>>> coll = htcondor.Collector("red-condor.unl.edu")
>>> ad = coll.locate(htcondor.DaemonTypes.Schedd, "red.unl.edu")
>>> ad["MyAddress"]
'<129.93.239.129:41562>'
>>> ads = coll.locateAll(htcondor.DaemonTypes.Schedd)
>>> for ad in ads: print ad["Name"]
...
red-gw1.unl.edu
red-gw2.unl.edu
red.unl.edu
flocking@t3.unl.edu
t3.unl.edu
>>>
```

bbockelm — bbockelm@brian-test:~ — ssh — 80×24

```
Connection to red-man.unl.edu closed.
[bbockelm@brian-test ~]$ python
Python 2.4.3 (#1, Jan  8 2013, 21:12:22)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-52)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import classad
>>> import htcondor
>>> ad = classad.ClassAd({"MyType": "Demo", "foo": 1})
>>> coll = htcondor.Collector()
>>> coll.advertise([ad])
>>> coll.query(htcondor.AdTypes.Any, 'MyType =?= "Demo"')
[]
>>> coll.advertise([ad])
>>> coll.query(htcondor.AdTypes.Any, 'MyType =?= "Demo"')
[]
>>> ad = classad.ClassAd({"MyType": "Demo", "foo": 1, "Name": "DemoAd"})
>>> coll.advertise([ad])
>>> coll.query(htcondor.AdTypes.Any, 'MyType =?= "Demo"')
[[ LastHeardFrom = 1367202284; Name = "DemoAd"; MyType = "Demo"; foo = 1; Authen
ticatedIdentity = "unauthenticated@unmapped"; CurrentTime = time() ]]
>>>
```

Collector Advanced

- For many queries, pulling all attributes from the collector is *expensive*.
- You can specify a *projection list* of attributes. HTCondor will return the minimum number of attributes containing the ones you specify.
  - It will always pad in a few extra.

Collector - Advanced

bbockelm — Test Terminal — Python — 80×24

```
Last login: Mon Apr 29 08:47:21 on ttys004
Brians-MacBook-Air:~ bbockelm$ python
Python 2.7.2 (default, Jun 20 2012, 16:23:33)
[GCC 4.2.1 Compatible Apple Clang 4.0 (tags/Apple/clang-418.0.60)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import classad
>>> import htcondor
>>> coll = htcondor.Collector("red-condor.unl.edu")
>>> ads = coll.query(htcondor.AdTypes.Startd, 'true', ['MyAddress', 'Name', 'Cpu
s', 'Memory'])
>>> len(ads)
4649
>>> ads[0]
[ CurrentTime = time(); Memory = 119; MyAddress = "<172.16.1.70:38406>"; Cpus =
0; Name = "slot1@node070.red.hcc.unl.edu"; MyType = "Machine"; TargetType = "Job
" ]
>>>
```

Schedd Basics

bbockelm — Test Terminal — Python — 80×24

```
Last login: Mon Apr 29 14:25:48 on ttys004
Brians-MacBook-Air:~ bbockelm$ grid-proxy-
Brians-MacBook-Air:~ bbockelm$ python
Python 2.7.2 (default, Jun 20 2012, 16:23:33)
[GCC 4.2.1 Compatible Apple Clang 4.0 (tags/Apple/clang-418.0.60)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import htcondor
>>> coll = htcondor.Collector("red-condor.unl.edu")
>>> schedd_ad = coll.locate(htcondor.DaemonTypes.Schedd, "red.unl.edu")
>>> schedd = htcondor.Schedd(schedd_ad)
>>> jobs = schedd.query()
>>> print jobs[0],

    [
        CurrentTime = time();
        BufferSize = 524288;
        JobNotification = 0;
        BufferBlockSize = 32768;
        Err = "/var/lib/globus/job_home/uscmsPool2295/.globus/job/red/1629003077
2317236126.1905433861141216178/stderr";
        CumulativeSlotTime = 0;
        CoreSize = -1;
        NiceUser = false;
        x509UserProxyExpiration = 1367424183
```

bbockelm — demo@ip-10-62-61-234:~ — ssh — 80×24

```
>>> cluster = schedd.submit(classad.ClassAd({"Cmd": "/bin/sh", "Arguments": "-c
'echo Hello world && sleep 1m'", "Out": "test.out", "Err": "test.err"}))
>>> cluster
4
>>> schedd.query('ClusterID =?= 4')
[[ NumCkpts_RAW = 0; BufferSize = 524288; NumJobMatches = 1; LastMatchTime = 136
7273629; LastJobLeaseRenewal = 1367273629; NiceUser = false; CoreSize = -1; Cumu
lativeSlotTime = 0; OnExitHold = false; GlobalJobId = "ip-10-62-61-234.ec2.inter
nal#4.0#1367273629"; RequestCpus = 1; Err = "test.err"; BufferBlockSize = 32768;
 TransferringInput = false; ImageSize = 100; CurrentTime = time(); WantCheckpoin
t = false; CommittedTime = 0; TargetType = "Machine"; WhenToTransferOutput = "ON
_EXIT"; ServerTime = 1367273642; Cmd = "/bin/sh"; JobUniverse = 5; BytesRecvd =
9.386720000000000E+05; ExitBySignal = false; StartdIpAddr = "<10.62.61.234:50475
>"; PublicClaimId = "<10.62.61.234:50475>#1367273386#7#..."; Iwd = "/home/demo";
 NumRestarts = 0; RemoteHost = "ip-10-62-61-234.ec2.internal"; CommittedSuspensi
onTime = 0; OrigMaxHosts = 1; Owner = "demo"; NumSystemHolds = 0; CumulativeSusp
ensionTime = 0; ShadowBday = 1367273629; RequestDisk = DiskUsage; Requirements =
 true && TARGET.OPSYS == "LINUX" && TARGET.ARCH == "X86_64" && TARGET.HasFileTra
nsfer && TARGET.Disk >= RequestDisk && TARGET.Memory >= RequestMemory; MinHosts
= 1; JobNotification = 0; NumCkpts = 0; LastSuspensionTime = 0; NumJobStarts = 0
```

bbockelm — demo@ip-10-62-61-234:~ — ssh — 80×24

```
 "Job"; CondorVersion = "$CondorVersion: 7.9.5 Apr 04 2013 BuildID: 114739 $"; J
obRunCount = 1; StreamErr = false; DiskUsage_RAW = 1; PeriodicHold = false; Proc
Id = 0; User = "demo@ip-10-62-61-234.ec2.internal"; TransferQueued = false; Last
JobStatus = 1; Arguments = "-c 'echo Hello world && sleep 1m'"; Out = "test.out"
; JobCurrentStartDate = 1367273629; JobStatus = 2; PeriodicRelease = false; Auto
ClusterAttrs = "JobUniverse,LastCheckpointPlatform,NumCkpts,RemoteGroup,Submitte
rGroup,SubmitterUserPrio,DiskUsage,ImageSize,RequestDisk,RequestMemory,Requireme
nts,NiceUser,ConcurrencyLimits"; RequestMemory = ifthenelse(MemoryUsage isnt und
efined,MemoryUsage,( ImageSize + 1023 ) / 1024); Args = ""; MaxHosts = 1; TotalS
uspensions = 0; CommittedSlotTime = 0; StartdPrincipal = "unauthenticated@unmapp
ed/10.62.61.234"; CondorPlatform = "$CondorPlatform: x86_64_RedHat6 $"; AutoClus
terId = 7; ShouldTransferFiles = "YES"; ExitStatus = 0; NumShadowStarts = 1; Mac
hineAttrCpus0 = 1; QDate = 1367273629; EnteredCurrentStatus = 1367273629 ]]
>>>
>>>
>>> open("test.out", "r").read()
'Hello world\n'
>>>
```

Submit ClassAds

- We normally submit using the submit file format, not using ClassAds.
  - Switching to ClassAds for submission requires a rewiring a few neurons.
  - Realizing the differences between the macro and ClassAd language costs a few more neurons.
- A few submit file / ClassAds translations:
  - error / Err
  - output / Out
  - executable / Cmd
  - should_transfer_files / ShouldTransferFiles
  - transfer_input_files / TransferIn
  - transfer_output_files / TransferOut
- The second argument to Schedd.submit determines how many processes to submit.
- From macros to ClassAds:
  - Instead of: error = “test.err.$(Process)”
  - Write: Err = strcat(“test.err”, ProcID)

Schedd Advanced

- A few useful methods:
  - **act**: Perform some action on one or more jobs (hold, release, remove, removeX, suspend, continue).
  - **edit**: Edit one or more job ClassAds
  - **reschedule**: Have Schedd request a new negotiation cycle.

Schedd Advanced

bbockelm — demo@ip-10-62-61-234:~ — ssh — 80×24

```
[demo@ip-10-62-61-234 ~]$ python
Python 2.6.6 (r266:84292, Dec  7 2011, 20:48:22)
[GCC 4.4.6 20110731 (Red Hat 4.4.6-3)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import htcondor
>>> schedd = htcondor.Schedd()
>>> jobs = schedd.query('true', ['ClusterID', 'foo'])
>>> jobs
[[ MyType = "Job"; TargetType = "Machine"; ServerTime = 1367284751; ClusterID =
7; CurrentTime = time() ]]
>>> schedd.edit('ClusterID =?= 7', "foo", '"bar"')
>>> schedd.query('true', ['ClusterID', 'foo'])
[[ MyType = "Job"; foo = "bar"; TargetType = "Machine"; ServerTime = 1367284782;
 ClusterID = 7; CurrentTime = time() ]]
>>> schedd.edit('ClusterID =?= 7', "foo", '42')
>>> schedd.query('true', ['ClusterID', 'foo'])
[[ MyType = "Job"; foo = 42; TargetType = "Machine"; ServerTime = 1367284792; Cl
usterID = 7; CurrentTime = time() ]]
>>> schedd.act(htcondor.JobAction.Hold, ['7.0'])
[ TotalNotFound = 0; TotalPermissionDenied = 0; TotalAlreadyDone = 1; TotalJobAd
s = 1; TotalSuccess = 0; TotalChangedAds = 0; TotalBadStatus = 0; TotalError = 0
 ]
>>>
```

Schedd Advanced - File Transfer

bbockelm — demo@ip-10-62-61-234:~ — ssh — 80×24

```
[demo@ip-10-62-61-234 ~]$ python
Python 2.6.6 (r266:84292, Dec  7 2011, 20:48:22)
[GCC 4.4.6 20110731 (Red Hat 4.4.6-3)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import htcondor
>>> import classad
>>> schedd = htcondor.Schedd()
>>> ad_results = []
>>> cluster = schedd.submit(classad.ClassAd({"Cmd": "/bin/sh", "Arguments": "-c
'echo Hello world && sleep 1m'"}), 1, True, ad_results)
>>> cluster
5
>>> ad_results[0]
[ BufferSize = 524288; NiceUser = false; CoreSize = -1; CumulativeSlotTime = 0;
OnExitHold = false; RequestCpus = 1; Err = "/dev/null"; BufferBlockSize = 32768;
 ImageSize = 100; CurrentTime = time(); WantCheckpoint = false; CommittedTime =
0; TargetType = "Machine"; WhenToTransferOutput = "ON_EXIT"; Cmd = "/bin/sh"; Jo
bUniverse = 5; ExitBySignal = false; HoldReasonCode = 16; Iwd = "/home/demo"; Nu
mRestarts = 0; CommittedSuspensionTime = 0; Owner = undefined; NumSystemHolds =
0; CumulativeSuspensionTime = 0; RequestDisk = DiskUsage; Requirements = true &&
 TARGET.OPSYS == "LINUX" && TARGET.ARCH == "X86_64" && TARGET.HasFileTransfer &&
 TARGET.Disk >= RequestDisk && TARGET.Memory >= RequestMemory; MinHosts = 1; Job
Notification = 0; NumCkpts = 0; LastSuspensionTime = 0; NumJobStarts = 0; WantRe
moteSyscalls = false; JobPrio = 0; RootDir = "/"; CurrentHosts = 0; StreamOut =
```

bbockelm — demo@ip-10-62-61-234:~ — ssh — 80×24

```
>>> schedd.query('ClusterID =?= 5', ["ClusterID", "ProcID"])
[[ MyType = "Job"; TargetType = "Machine"; ServerTime = 1367275365; ClusterID =
5; ProcID = 0; CurrentTime = time() ]]
>>> schedd.query('ClusterID =?= 5', ["ClusterID", "ProcID", "JobStatus"])
[[ MyType = "Job"; JobStatus = 5; TargetType = "Machine"; ServerTime = 136727539
3; ClusterID = 5; ProcID = 0; CurrentTime = time() ]]
>>> schedd.spool(ad_results)
>>> schedd.query('ClusterID =?= 5', ["ClusterID", "ProcID", "JobStatus"])
[[ MyType = "Job"; JobStatus = 2; TargetType = "Machine"; ServerTime = 136727541
1; ClusterID = 5; ProcID = 0; CurrentTime = time() ]]
>>> schedd.query('ClusterID =?= 5', ["ClusterID", "ProcID", "JobStatus"])
[[ MyType = "Job"; JobStatus = 4; TargetType = "Machine"; ServerTime = 136727547
1; ClusterID = 5; ProcID = 0; CurrentTime = time() ]]
>>> schedd.retrieve('Cluster =?= 5')
>>> schedd.query('ClusterID =?= 5', ["ClusterID", "ProcID", "JobStatus"])
[[ MyType = "Job"; JobStatus = 4; TargetType = "Machine"; ServerTime = 136727550
4; ClusterID = 5; ProcID = 0; CurrentTime = time() ]]
>>> schedd.act(htcondor.JobAction.Remove, ['5.0'])
[ TotalNotFound = 0; TotalPermissionDenied = 0; TotalAlreadyDone = 0; TotalJobAd
s = 1; TotalSuccess = 1; TotalChangedAds = 1; TotalBadStatus = 0; TotalError = 0
 ]
>>>
>>>
>>>
```

Daemon Commands

- An administrator can send commands to arbitrary HTCondor daemons via python.
  - Uses the same internal protocol as CLI such as condor_off and condor_on.
  - A blessing and a curse: HTCondor doesn't document what the protocol commands do. They are a bit similar to Unix signals in that you receive no indication the command did anything.
- Do *you* know the difference between DaemonOff, DaemonOffFast, DaemonOffPeaceful, DaemonsOff, DaemonsOffFast, DaemonsOffPeaceful, OffFast, OffForce, OffGraceful, and OffPeaceful?
- A new developer best keep to Reconfig, Restart, and DaemonsOff. Send the command to the master.
  - Some commands will take an extra argument - such as the subsystem to restart for "DaemonOff".

Daemon Commands

bbockelm — demo@ip-10-62-61-234:~ — ssh — 80×24

```
[demo@ip-10-62-61-234 ~]$ python
Python 2.6.6 (r266:84292, Dec  7 2011, 20:48:22)
[GCC 4.4.6 20110731 (Red Hat 4.4.6-3)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import htcondor
>>> coll = htcondor.Collector()
>>> ad = coll.locate(htcondor.DaemonTypes.Master, 'ip-10-62-61-234.ec2.internal'
)
>>> print ad['MyAddress'], ad['Name']
<10.62.61.234:34494> ip-10-62-61-234.ec2.internal
>>> htcondor.send_command(ad, htcondor.DaemonCommands.Restart)
>>>
[demo@ip-10-62-61-234 ~]$ tail -f /var/log/condor/MasterLog
04/29/13 23:00:59    /etc/condor/condor_config.local
04/29/13 23:00:59 DaemonCore: command socket at <10.62.61.234:43275>
04/29/13 23:00:59 DaemonCore: private command socket at <10.62.61.234:43275>
04/29/13 23:00:59 Setting maximum accepts per cycle 8.
04/29/13 23:01:00 Started DaemonCore process "/usr/sbin/condor_collector", pid a
nd pgroup = 20604
04/29/13 23:01:00 Waiting for /var/log/condor/.collector_address to appear.
04/29/13 23:01:01 Found /var/log/condor/.collector_address.
04/29/13 23:01:01 Started DaemonCore process "/usr/sbin/condor_negotiator", pid
and pgroup = 20605
04/29/13 23:01:01 Started DaemonCore process "/usr/sbin/condor_schedd", pid and
```

Daemon Commands

- I hope this will really improve the "scriptability" of a HTCondor pool.
  - For example, one could implement a rolling restart cron job that ensures no more than 10% of nodes are draining at once.

Etc

- To invalidate an existing in-process security session:
  - htcondor.SecMan().invalidateAllSessions()
- To access the param subsystem:
  - htcondor.param
  - *Treat like a python dictionary.*
- To reload the client configuration from disk:
  - htcondor.reload_config()

Python Bindings Futures

- Python bindings will continue to receive periodic updates to keep parity with client-side tools. Current plans for 8.0:
  - Improve Schedd.edit method.
  - Release bindings for Mac OS X.
  - Better implement keyword parameters throughout.
  - Config errors should not exit the interpreter.
- Wishlist:
  - Add bindings for condor_tail.
  - Add bindings for condor_ping.
  - Expose more advanced ClassAd functionality (matching).
  - Cleanup the send_command function.
  - DaemonCore?
- I'm looking to broaden the set of maintainers. If you want seriously better bindings, plan to contribute!
  - In particular, I have no knowledge of Windows development! I believe we are a few small patches away from enabling Python bindings for Windows.

Q?

http://research.cs.wisc.edu/htcondor/manual/v7.9/8_6Python_Bindings.html