# Design Document: condor\_peek

Brian Bockelman

## Motivation

Currently there are limited mechanisms to monitor a job while it is running in HTCondor:

* **condor\_ssh\_to\_job**: This allows users to SSH to the worker node where the job is running. However, this is often unreliable when running on the grid as it may violate the site security policy. It also allows unrestricted access, making it risky to integrate with monitoring systems.
* **chirp**: Jobs can push monitoring data into the schedd classad or the schedd filesystem. This requires a-priori planning by the user, changes to the job, and is likely not very scalable if many jobs automatically push frequent file updates.
* **Classad updates**: These provide periodic updates about basic job statistics. These are scalable but provide a limited depth of information..

We propose a new command-line tool, *condor\_peek*. This will allow users to fetch files from a running job back to the local host. It is similar in spirit to the *bpeek* tool from LSF.

The advantages of the proposed *condor\_peek* over existing tools are:

* **Restricted scope**. Should be more reliable on the grid glideins (no dependency on the worker node environment) and less likely to cause issues for security teams.
* **On-demand / small-footprint**: This is done on user request. The load added to the system is proportional to the number of user requrests, not the number of jobs.
* **Low-latency**: This tool will contact the starter directly and invoke a DaemonCore command. It has minimal involvement with the schedd (in order to establish a security context).

## Architecture

We will add a new DaemonCore command, STARTER\_PEEK, to the condor\_starter. This command will accept a ClassAd containing the files requested for transfer and respond with an ad containing the ordered list of files it will transfer. The starter will then transfer the specified files.

The starter will not transfer files if:

* The starter is not in a fully initialized state (re-using the same logic as condor\_ssh\_to\_job).
* The requested files are not the job’s stderr, stdout, or in the transfer\_output\_files.
* The administrator or job owner did not make the request.

The starter *will* transfer files for suspended jobs.

The wire protocol will be primarily ClassAd based and based on CEDAR. Care will be taken to make sure the protocol can be made forward-compatible.

### Client tool, version 1

The condor\_peek tool will be able to specify a single file (one from the transfer output list, stderr, or stdout), a single job, and a maximum number of bytes. It will print the last N bytes of the file contents to stdout.

Python bindings for the new STARTER\_PEEK command will be added.

## Client tool, version 2

The client will be able to specify the maximum number of bytes to transfer, and whether to start off at the beginning or end of the file. The maximum number of bytes specified by the client’s configuration will be respected by *condor\_peek* and the starter.

A user can specify the job constraint in a manner similar to *condor\_q* (job cluster, job cluster and process, owner, or ClassAd expression). Additional flags will allow one to specify which files to transfer. By default, the transferred files will be written to the appropriate on-disk locations (no file transfer URLs will be supported).

A future extension would be to allow the client to request offsets, allowing the implementation of “*condor\_peek –f*” for continuously tailing the file.

## Effort Estimates

These estimates specify estimated “CPU time” for Brian Bockelman. These numbers are prepared after spending 2 hours reading through the relevant *condor\_ssh\_to\_job* and *condor\_fetchlog* code.

* Basic starter changes – 1 day.
* *condor\_peek* (v1) command line tool – 1 day.
* Python bindings – 4 hours.
* Test and integration tests – 2 days.
* Code Review and follow-up – 1 day.

Note I propose leaving v2 of the *condor\_peek* tool for later.