



**Intellectual Property Management Plan**

**February 28, 2011**



Turbine Client

User Manual

(Python Scripts)

Version 2014.10.0

October 31, 2014

This Material was produced under the DOE Carbon Capture Simulation Initiative (CCSI), and copyright is held by the software owners: ORISE, LANS, LLNS, LBL, PNNL, CMU, WVU, et al. The software owners and/or the U.S. Government retain ownership of all rights in the CCSI software and the copyright and patents subsisting therein. Any distribution or dissemination is governed under the terms and conditions of the CCSI Test and Evaluation License, CCSI Master Non-Disclosure Agreement, and the CCSI Intellectual Property Management Plan. No rights are granted except as expressly recited in one of the aforementioned agreements.

Revision Log

|  |  |  |
| --- | --- | --- |
| **Version Number** | **Release Date** | **Description** |
| Version 2014.10.0 | 10/31/2014 | 2014 October IAB Release |
|  |  |  |
|  |  |  |

Table of Contents

[1. Introduction 1](#_Toc401809145)

[2. Prerequisites 1](#_Toc401809146)

[2.1. Operating System 1](#_Toc401809147)

[2.2. Required Software 1](#_Toc401809148)

[3. Installation 2](#_Toc401809149)

[3.1. UNIX 2](#_Toc401809150)

[3.2. Windows 2](#_Toc401809151)

[4. Configuration 2](#_Toc401809152)

[5. Basic Operations 3](#_Toc401809153)

[5.1. UNIX 3](#_Toc401809154)

[5.2. Windows 3](#_Toc401809155)

[5.3. Discover Applications 3](#_Toc401809156)

[5.4. Application Description 4](#_Toc401809157)

[5.5. Create Simulation 5](#_Toc401809158)

[6. Advanced Scripting (Bash) 8](#_Toc401809159)

[6.1. Check Status of Session Every Minute 8](#_Toc401809160)

[6.2. Check All Sessions for Submit Jobs 8](#_Toc401809161)

[7. Integration Testing 8](#_Toc401809162)

[8. Support 8](#_Toc401809163)

[9. Python API Reference 9](#_Toc401809164)

[9.1. Application Resource 9](#_Toc401809165)

[9.2. Consumer Resource 9](#_Toc401809166)

[9.3. Job Resource 10](#_Toc401809167)

[9.4. Session Resource 10](#_Toc401809168)

[9.5. Simulation Resource 13](#_Toc401809169)

[10. Python API Configuration File 14](#_Toc401809170)

[10.1. Configuration File 14](#_Toc401809171)

[11. Tests 15](#_Toc401809172)

[11.1. Integration 15](#_Toc401809173)

[12. Logging 16](#_Toc401809174)

To obtain support for this package, please send an email to   
[ccsi-support@acceleratecarboncapture.org](mailto:ccsi-support@acceleratecarboncapture.org).

# Introduction

The Turbine Science Gateway is a web application and execution environment for running and managing scientific applications and storing and archiving the results. Clients interact with the Turbine resource-oriented architecture through a RESTful web interface, either directly over HTTP or using the higher-level Python Client Application Programming Interface (API). The Python API is designed to be easily scriptable, returning structured JSON output that can be easily consumed by other tools. Turbine is a generic solution that can be extended to process modeling and simulation applications. Currently AspenTech’s Aspen Plus®, Aspen Custom Modeler® (ACM), and Microsoft® Excel® applications are supported.

The Turbine Client can be deployed on a single workstation, a cluster, and the Amazon® Web Services EC2® cloud. The server-side software is Windows®-based. The cluster and EC2 deployments allow for parallel application executions, which is shown to scale up to deployments of 400 concurrent instances. This can dramatically increase application throughput and thus decrease the time to solution.

This manual provides detailed usage instructions for using the core Turbine Client scripts with a Turbine Science Gateway. These scripts provide direct access to the Turbine Science Gateway Web API, which is used to upload and change a simulation, execute simulations, and monitor the status of job requests and components.

# Prerequisites

## Operating System

* Windows XP, Windows 2003 Server, Windows 7, Windows 2008 Server
* UNIX® systems (Linux®, Mac® OS)

## Required Software

* Python™ 2.7
  + Other versions may work
* Python dateutil
  + Typically installed automatically

# Installation

The Turbine Python module comes with a setup script that installs everything that a user needs to run Turbine. There are multiple options for installation. The user should select the option that matches their needs. See the README file in the distribution for the full instructions.

## UNIX

If the user is installing UNIX for multiple users and has root privileges, the user should use this command:

% python setup.py install

If the user is only installing UNIX for them self (or does not have root privileges), the user should use this command:

% python setup.py install --user

## Windows

On Windows a user can either modify the environment to include the Python interpreter in the environment path variable or provide the following full path to the interpreter:

> C:\Python2.7\python.exe setup.py install

# Configuration

The Turbine Client requires a configuration file to retrieve data about the Gateway and the simulation. Because the configuration file contains some simulation specific data, it is convenient to keep a version around for each simulation that is being run. The configuration file is in Python ConfigParser format. A very simple sectioned name=value format. The configuration file is referred to as “config.txt” in the examples in the remainder of this manual (refer to Section 10 for details).

# Basic Operations

All of the examples in this section are written in a UNIX convention. For Windows the user needs to invoke the Python interpreter and provide the path to the command utility. All Turbine Client scripts require a configuration file to be passed as an argument (refer to Section 10 for configuration file details).

## UNIX

% CONFIG=config.txt

% turbine\_application\_list $CONFIG

## Windows

> C:\Python2.7\python.exe C:\Python2.7\Scripts\turbine\_application\_list

config.txt

## Discover Applications

Several applications are supported by the Turbine Gateway, but not all may be enabled. Use the script below to discover which applications are available.

% turbine\_application\_list $CONFIG

Total Applications: 3

ACM

AspenPlus

excel

## Application Description

The application description also defines the set of input and output files that are staged in and out for a single execution run. None of the configured applications below stage files out.

% turbine\_application\_list -v $CONFIG

Total Applications: 3

============================================================

ACM

Inputs

1. configuration: {u'Required': True, u'Type': u'text/plain', u'Name': u'configuration'}

2. aspenfile: {u'Required': True, u'Type': u'text/plain', u'Name': u'aspenfile'}

============================================================

AspenPlus

Inputs

1. configuration: {u'Required': True, u'Type': u'text/plain', u'Name': u'configuration'}

2. aspenfile: {u'Required': True, u'Type': u'text/plain', u'Name': u'aspenfile'}

============================================================

excel

Inputs

1. configuration: {u'Required': True, u'Type': u'text/plain', u'Name': u'configuration'}

2. spreadsheet: {u'Required': True, u'Type': u'application/vnd.ms-excel', u'Name': u'spreadsheet'}

============================================================

## Create Simulation

To run an application such as ACM, the user first needs to create a new simulation resource on the Gateway that references this application and then upload the required staged-input files.

To create an ACM simulation the user needs:

* A Turbine Client configuration file (refer to Section 10)
* A Simulation Name, an informative alpha-numeric string (e.g., MEA\_Adsorber\_v1.2)
* The Aspen Custom Modeler file (e.g., acmf)
* A Sinter JSON Configuration file (refer to the *SimSinter User Manual*)

For this example it is assumed that the scripts directory is added to the user’s path as recommended in the Installation Guide. If not, the user has to use the absolute path to the scripts in the example. It is also assumed that the simulation and configuration files have been copied to the current working directory. Below are the steps that are required for creating a simulation using the Turbine Client scripts.

All of the files in this example are available in the Turbine Client distribution, the paths are listed below:

* test/integration/simulations/Hybrid\_split/Hybrid\_v0.51\_rev1.1\_UQ\_0809.acmf
* test/integration/simulations/Hybrid\_split/Hybrid\_v0.51\_rev1.1\_UQ\_0809\_sinter.json
* test/integration/files/Hybrid\_split\_testruns\_noreset.txt

### Create the Simulation Resource

The simulation resource acts as a holder for all of the simulation files which are to be staged into the working directory of each simulation execution.

turbine\_simulation\_create SIMULATION\_NAME APPLICATION\_NAME CONFIG

* **[SIMULATION\_NAME]** 
  + ACMHybridSplit is the simulation name that is used in following steps
* **[APPLICATION\_NAME]** 
  + ACM, this is a reference to the application name
* **[CONFIG]**
  + Turbine Client configuration file

% turbine\_simulation\_create ACMHybridSplit ACM $CONFIG

### Put the aspenfile Simulation File in the ACMHybridSplit Simulation

turbine\_simulation\_update [options] SIMULATION\_NAME FILE\_NAME CONFIG

* **[OPTIONS]**
  + **-r RESOURCE, --resource=RESOURCE**
    - Specifies which staged-in file is specified by the application resource (e.g., aspenfile)
* **[SIMULATION\_NAME]** 
  + ACMHybridSplit is the simulation name, that was created in the previous step
* **[FILE\_NAME]** 
  + Path to file (e.g., acmf)
* **[CONFIG]**
  + Turbine Client configuration file

% turbine\_simulation\_update -r aspenfile ACMHybridSplit Hybrid\_v0.51\_rev1.1\_UQ\_0809.acmf $CONFIG

### Put the SimSinter Configuration File in the ACMHybridSplit Simulation

Add the SimSinter configuration file to the ACMHybridSplit simulation resource. To construct a SimSinter configuration file, refer to the *SimSinter User Manual*. The configuration resource is always the placeholder for the SimSinter configuration file.

% turbine\_simulation\_update -r configuration ACMHybridSplit Hybrid\_v0.51\_rev1.1\_UQ\_0809\_sinter.json $CONFIG

At this point the Gateway has a new simulation that can be run, ACMHybridSplit.

### Launch Simulation Executions “Jobs”

A “session” is used to group a collection of simulation runs or “jobs”. To run a set of jobs the user first needs to create the session. This returns a string that is the session GUID (globally unique identifier). The user should keep track of the GUID as it is used to reference this set of jobs.

% turbine\_session\_create $CONFIG

"e1d318af-87fd-4721-8db7-6b616818c642"

Each simulation execution is referred to as a “job”. To run a set of jobs the user needs to create a JSON file that contains a list of job descriptions (refer to the *SimSinter User Manual*).

% turbine\_session\_append "e1d318af-87fd-4721-8db7-6b616818c642" Hybrid\_split\_testruns\_noreset.txt $CONFIG

Jobs wait in the create state until they are moved to submit by the user. To move the group of simulations onto the job queue:

%turbine\_session\_start "e1d318af-87fd-4721-8db7-6b616818c642" $CONFIG

### Check Status of Simulation Executions “Jobs”

To check the status of all the jobs in the session:

% turbine\_session\_status "e1d318af-87fd-4721-8db7-6b616818c642" $CONFIG

{"create":0,"running":3,"setup":1,"submit":4,"pause":0,"finished":0,"error":0,"cancel":0,"success":2,"terminate":0}

From this the user can see that:

* 4 Runs are waiting on the submit queue
* 2 Runs have completed successfully
* 3 Runs are currently running
* 1 Run is in setup (staging-in files, initializing AspenEngine)

Checking again later, more jobs have completed:

% turbine\_session\_status "e1d318af-87fd-4721-8db7-6b616818c642" $CONFIG

{"create":0,"running":0,"setup":0,"submit":0,"pause":0,"finished":0,"error":0,"cancel":0,"success":10,"terminate":0}

Once the jobs completed, the user can pull the results down from the Gateway. The script turbine\_session\_get\_results prints the resulting JSON file to standard out, to capture the results in a file redirect standard out.

% turbine\_session\_get\_results "e1d318af-87fd-4721-8db7-6b616818c642" $CONFIG > results.json

The results of the run are in the file results.jsonand a user can easily load the results in Python.

% python -c “import json; l = json.load(open(‘results.json’)); print l”

# Advanced Scripting (Bash)

This section contains a few advanced examples of using the Turbine scripts to monitor the Gateway. The variable “SESSION” is set to the identifier of the session the user is interested in. The variable “CONFIG” is set to the absolute path of the Turbine Client configuration file.

## Check Status of Session Every Minute

% I=0; while true; do I=$(($I+1)) ; echo -n "${I}) " ; turbine\_session\_status $SESSION $CONFIG ; sleep 60; done

## Check All Sessions for Submit Jobs

% echo "=== CHECK SESSIONS FOR SUBMIT JOBS ==="; INCR=0; for SESSION in `turbine\_session\_list -j $CONFIG | python -c "import sys,json; print '\n'.join(json.load(sys.stdin))"`; do INCR=$((${INCR}+1)); echo -n "$INCR) $SESSION -- "; (turbine\_session\_status $SESSION $CONFIG | python -c "import sys,json; d = json.load(sys.stdin); x = (d if d['submit']>0 else None); print x") ; done

# Integration Testing

There are several test suites that are available to verify that the Turbine Client and Gateway are working correctly.

* ws\_test\_suite.py
  + Basic Read operations on web resources
* simulation\_test\_suite.py
  + SimulationWriteTest creates a simulation on the Gateway
* acm\_test\_suite.py
  + Executes the ACM simulation on the Gateway
* aspenplus\_test\_suite.py
  + Executes the Aspen Plus simulations on the Gateway
* excel\_test\_suite.py
  + Executes an Excel simulation on the Gateway

To execute a test the user should be in the “test/integration” directory. The user needs to copy over the template “integration\_test.cfg.in” file to “integration\_test.cfg”, and then add the information from their Turbine Client configuration file (refer to Section 10).

% cp integration\_test.cfg.in integration\_test.cfg

% python suites/ws\_test\_suite.py

# Support

To obtain support for this package, send an email to [ccsi-support@acceleratecarboncapture.org](mailto:ccsi-support@acceleratecarboncapture.org).

# Python API Reference

## Application Resource

### turbine\_application\_list

Usage: turbine\_application\_list [options] CONFIG\_FILE

List all application resources, by default print in human readable format.

Options:

-h, --help show this help message and exit

-p PAGE, --page=PAGE page number

-r RPP, --rpp=RPP results per page

-v, --verbose verbose output

-j, --json print results as json to stdout

## Consumer Resource

### turbine\_consumer\_list

Usage: turbine\_consumer\_list [options] CONFIG\_FILE

List all Consumer resources, by default print in human readable format.

Options:

-h, --help show this help message and exit

-v, --verbose verbose output

-s STATUS, --status=STATUS

query on status ['up'|'down'|'error']

-j, --json print results as json to stdout

### turbine\_consumer\_log

Usage: turbine\_consumer\_log [options] CONFIG\_FILE

Retrieves logging messages from compute resource running the specified

Consumer. Log messages are printed to screen in order. This functionality

is not available in all deployments.

Options:

-h, --help show this help message and exit

## Job Resource

### turbine\_job\_script

Usage: turbine\_job\_script [options] CONFIG\_FILE

Queries for job resources based on select criteria, by default prints JSON

array of jobs.

Options:

-h, --help show this help message and exit

-j SUBRESOURCE, --jobid=SUBRESOURCE

JOB ID

-n SIMULATION, --sim=SIMULATION

Simulation Name

-x STATE, --state=STATE

Job Status to query: ['pause', 'success', 'create',

'terminate', 'submit', 'running', 'warning', 'error',

'cancel', 'expired', 'setup']

-c CONSUMER, --consumer=CONSUMER

Consumer GUID to query

-b, --basic Print Basic Information About Job(s)

-p PAGE, --page=PAGE page number

-r RPP, --rpp=RPP results per page

-v, --verbose verbose output

-s SESSION, --session=SESSION

session identifier (guid)

## Session Resource

### turbine\_session\_append

Usage: turbine\_session\_append SESSIONID JOBS\_FILE CONFIG\_FILE

Appends job descriptions to a session, prints the number of jobs added.

Options:

-h, --help show this help message and exit

### turbine\_session\_create

Usage: turbine\_session\_create CONFIG\_FILE

Creates new session resource and prints GUID of created session.

Options:

-h, --help show this help message and exit

### turbine\_session\_delete

Usage: turbine\_session\_delete SESSIONID CONFIG\_FILE

Delete session and all jobs it contains, prints number of jobs deleted as

result of session delete.

Options:

-h, --help show this help message and exit

### turbine\_session\_get\_results

Usage: turbine\_session\_get\_results SESSIONID CONFIG\_FILE

Gets the results of all the completed jobs in this session

Options:

-h, --help show this help message and exit

-p PAGE, --page=PAGE page number

-r RPP, --rpp=RPP results per page

-v, --verbose verbose output

### turbine\_session\_graphs

Usage: turbine\_session\_graphs SESSIONID CONFIG\_FILE

session resource utility, prints basic session statistics

Options:

-h, --help show this help message and exit

-p PAGE, --page=PAGE page number

-r RPP, --rpp=RPP results per page

-v, --verbose verbose output

--plot=PLOT Plot type (see --list). Any unique prefix is OK.

(default=density)

--list List plot types and metrics.

--metric=OPT\_METRIC Metric to show in plot (see --list).

### turbine\_session\_kill

Usage: turbine\_session\_kill SESSIONID CONFIG\_FILE

Terminate jobs in session in state setup, running. Print number of jobs

terminated.

Options:

-h, --help show this help message and exit

### turbine\_session\_list

Usage: turbine\_session\_list [options] CONFIG\_FILE

Prints human readable listing of all session GUIDs.

Options:

-h, --help show this help message and exit

-p PAGE, --page=PAGE page number

-r RPP, --rpp=RPP results per page

-v, --verbose verbose output

-j, --json print results as json to stdout

### turbine\_session\_start

Usage: turbine\_session\_start SESSIONID CONFIG\_FILE

Move all jobs in states pause and create to submit. Prints the number of jobs

moved to state submit.

Options:

-h, --help show this help message and exit

### turbine\_session\_stats

Usage: turbine\_session\_stats SESSIONID CONFIG\_FILE

session resource utility, prints basic session statistics

Options:

-h, --help show this help message and exit

-p PAGE, --page=PAGE page number

-r RPP, --rpp=RPP results per page

-v, --verbose verbose output

### turbine\_session\_status

Usage: turbine\_session\_status [options] SESSIONID CONFIG\_FILE

session resource utility, lists all session resources

Options:

-h, --help show this help message and exit

-p PAGE, --page=PAGE page number

-r RPP, --rpp=RPP results per page

-v, --verbose verbose output

### turbine\_session\_stop

Usage: turbine\_session\_stop SESSIONID CONFIG\_FILE

Move all jobs in state submit to pause. Prints the number of jobs moved to

state pause.

Options:

-h, --help show this help message and exit

## Simulation Resource

### turbine\_simulation\_create

Usage: turbine\_simulation\_create [options] SIMULATION\_NAME APPLICATION\_NAME CONFIG\_FILE

Create an empty Simulation Resource

Options:

-h, --help show this help message and exit

### turbine\_simulation\_get

Usage: turbine\_simulation\_get [options] SIMULATION\_NAME CONFIG\_FILE

Retrieves the Simulation resource, by default prints as JSON.

Options:

-h, --help show this help message and exit

-r RESOURCE, --resource=RESOURCE

return only the specified input subresource

-s, --save Save the resource as a file

### turbine\_simulation\_list

Usage: turbine\_simulation\_list [options] CONFIG\_FILE

Retrieves list of all simulations, by default prints in human readable format.

Options:

-h, --help show this help message and exit

-p PAGE, --page=PAGE page number

-r RPP, --rpp=RPP results per page

-v, --verbose verbose output

-j, --json print results as json to stdout

### turbine\_simulation\_update

Usage: turbine\_simulation\_update [options] SIMULATION\_NAME FILE\_NAME CONFIG\_FILE

Update simulation by essentially doing a PUT with the specified file to the

resource or optionally sub-resource.

Options:

-h, --help show this help message and exit

-r RESOURCE, --resource=RESOURCE

select the staged input file name (matches

Application)

# Python API Configuration File

## Configuration File

The Turbine Python script configuration file is a parameter to every Turbine Python script, it is in “ConfigParser” format. The file contains a separate section for each web resource Application, Consumer, Job, Session, and Simulation. These sections each contain a “url” key, which specifies the URL of the web resource. The “Logging” section is optional. In this section a user can specify a logging configuration file. There is a sample “logging.conf” included in the distribution. The “Authentication” section is for specifying user’s credentials.

[Logging]

fileConfig=logging.conf

[Consumer]

url=https://SERVER:PORT/Turbine/consumer/

[Session]

url=https://localhost:8080/Turbine/session/

[Job]

url=https://localhost:8080/Turbine/job/

[Simulation]

url=https://localhost:8080/Turbine/simulation/

[Application]

url=https://localhost:8080/Turbine/application/

[Authentication]

username=USERNAME

password=PASSWORD

[Security]

TrustedCertificateAuthorities=CACERTS

* **SERVER:PORT** 
  + The IP Address/FQDN and port of the Gateway (for example: *localhost:8080)*
* **USERNAME** 
  + Turbine Gateway account username
* **PASSWORD**
  + Turbine Gateway account password
* **CACERTS**
  + If the user provides the “Security” section the Client validates the server’s certificate against the provided certificate authority list
  + This is a text file containing CA certificates
  + To turn validation off, remove the “Security” section

# Tests

## Integration

A template integration test configuration file is located in the “test/integration” directory, and is named “integration\_test.cfg.in”. The file contains all of the sections contained in the Python API Configuration File (Section 9), and these sections should match. There are additional sections for several of the tests, but these require no modification. Not all of these tests may be relevant for the Turbine Gateway Deployment the user is using, namely some of the applications may not be supported (e.g., ACM, Aspen Plus, Excel, gPROMS).

* WS Tests Suite

% python suites/ws\_test\_suite.py

....

----------------------------------------------------------------------

Ran 4 tests in 0.405s

OK

* Simulation

% python suites/simulation\_test\_suite.py

.

----------------------------------------------------------------------

Ran 1 test in 1.474s

OK

* AspenTech ACM

% python suites/acm\_test\_suite.py

.

----------------------------------------------------------------------

Ran 1 test in 132.458s

OK

* AspenTech Aspen Plus

% python suites/aspenplus\_test\_suite.py

.

----------------------------------------------------------------------

Ran 1 test in 115.863s

OK

# Logging

The Turbine Client uses the standard Python logging module. To use logging for Turbine Client, add a python logging file with the name “logging.conf” into the working directory. Below is an example file.

[loggers]

keys=root,commands,session

[handlers]

keys=consoleHandler

[formatters]

keys=simpleFormatter

[logger\_root]

level=DEBUG

handlers=consoleHandler

[logger\_commands]

level=DEBUG

handlers=consoleHandler

qualname=turbine.commands

propagate=0

[logger\_session]

level=DEBUG

handlers=consoleHandler

qualname=turbine.commands.turbine\_session\_script

propagate=0

[handler\_consoleHandler]

class=StreamHandler

formatter=simpleFormatter

args=(sys.stdout,)

[formatter\_simpleFormatter]

format=%(asctime)s - %(name)s - %(levelname)s - %(message)s

datefmt=