[2]p()*1/2-^ [1]p1-^ TJ
310000pt

# tcpb_client Documentation

**Martinez Group**

**Oct 09, 2018**

## CONTENTS:

CHAPTER

ONE

# TERACHEM PROTOCOL BUFFER (TCPB) CLIENT

This repository is designed to facilitate the development a Python client for communicating with TeraChem.

This client uses C-style sockets for communication, and Protocol Buffers for a clean, well-defined way to serialize TeraChem input & output.

For more information, see the Wiki.

## 1.1 Contact


- Stefan Seritan <sseritan@stanford.edu>

CHAPTER

TWO

# TCPB

## 2.1 tcpb package

### 2.1.1 Submodules

### 2.1.2 tcpb.tcpb module

Simple Python socket client for communicating with TeraChem Protocol Buffer servers

**class** tcpb.tcpb.**TCProtobufClient**(*host*, *port*, *debug=False*, *trace=False*)
Bases: object

Connect and communicate with a TeraChem instance running in Protocol Buffer server mode (i.e. TeraChem was started with the -s|–server flag)

**__init__**(*host*, *port*, *debug=False*, *trace=False*)
Initialize a TCProtobufClient object.

**Parameters**

- **host** (*str*) – Hostname
- **port** (*int*) – Port number (must be above 1023)
- **debug** (*bool*) – If True, assumes connections work (used for testing with no server)
- **trace** (*bool*) – If True, packets are saved to .bin files (which can then be used for testing)

**check_job_complete**()
Pack and send a Status message to the TeraChem Protobuf server asynchronously. This function expects a Status message back with either working or completed set. Errors out if just busy message returned, implying the job we are checking was not submitted or had some other issue

**Returns** True if job is completed, False otherwise

**Return type** bool

**compute_ci_overlap**(*geom=None*, *geom2=None*, *cvec1file=None*, *cvec2file=None*, *orb1afile=None*, *orb1bfile=None*, *orb2afile=None*, *orb2bfile=None*, *unitType='bohr'*, ***kwargs*)
Compute wavefunction overlap given two different geometries, CI vectors, and orbitals, using the same atom labels/charge/spin multiplicity as the previous calculation.

To run a closed shell calculation, only populate orb1afile/orb2afile, leaving orb1bfile/orb2bfile blank. Currently, open-shell overlap calculations are not supported by TeraChem.

**Parameters**

- **geom** – Cartesian geometry of the first point
- **geom2** – Cartesian geometry of the second point
- **cvec1file** – Binary file of CI vector for first geometry (row-major, double64)
- **cvec2file** – Binary file of CI vector for second geometry (row-major, double64)
- **orb1afile** – Binary file of alpha MO coefficients for first geometry (row-major, double64)
- **orb1bfile** – Binary file of beta MO coefficients for first geometry (row-major, double64)
- **orb2afile** – Binary file of alpha MO coefficients for second geometry (row-major, double64)
- **orb2bfile** – Binary file of beta MO coefficients for second geometry (row-major, double64)
- **unitType** – Unit type key, as defined in the pb.Mol.UnitType enum (defaults to 'bohr')
- ****kwargs** – Additional TeraChem keywords, check _process_kwargs for behaviour

**Returns** CI vector overlaps

**Return type** (num_states, num_states) ndarray

**compute_coupling**(*geom=None*, *unitType='bohr'*, ***kwargs*)

Compute nonadiabatic coupling of a new geometry, but with the same atoms labels/charge/spin multiplicity and wave function format as the previous calculation.

**Parameters**

- **geom** – Cartesian geometry of the new point
- **unitType** – Unit type key, as defined in the pb.Mol.UnitType enum (defaults to 'bohr')
- ****kwargs** – Additional TeraChem keywords, check _process_kwargs for behaviour

**Returns** Nonadiabatic coupling vector

**Return type** (num_atoms, 3) ndarray

**compute_energy**(*geom=None*, *unitType='bohr'*, ***kwargs*)

Compute energy of a new geometry, but with the same atom labels/charge/spin multiplicity and wave function format as the previous calculation.

**Parameters**

- **geom** – Cartesian geometry of the new point
- **unitType** – Unit type key, as defined in the pb.Mol.UnitType enum (defaults to 'bohr')
- ****kwargs** – Additional TeraChem keywords, check _process_kwargs for behaviour

**Returns** Energy

**Return type** float

**compute_forces**(*geom=None*, *unitType='bohr'*, ***kwargs*)

Compute forces of a new geometry, but with the same atoms labels/charge/spin multiplicity and wave function format as the previous calculation.

**Parameters**

- **geom** – Cartesian geometry of the new point

- **unitType** – Unit type key, as defined in the pb.Mol.UnitType enum (defaults to 'bohr')
- ****kwargs** – Additional TeraChem keywords, check _process_kwargs for behaviour

**Returns** Tuple of (energy, forces), which is really (energy, -gradient)

**Return type** tuple

**compute_gradient**(*geom=None*, *unitType='bohr'*, ***kwargs*)

Compute gradient of a new geometry, but with the same atom labels/charge/spin multiplicity and wave function format as the previous calculation.

**Parameters**

- **geom** – Cartesian geometry of the new point
- **unitType** – Unit type key, as defined in the pb.Mol.UnitType enum (defaults to 'bohr')
- ****kwargs** – Additional TeraChem keywords, check _process_kwargs for behaviour

**Returns** Tuple of (energy, gradient)

**Return type** tuple

**compute_job_sync**(*jobType='energy'*, *geom=None*, *unitType='bohr'*, ***kwargs*)

Wrapper for send_job_async() and recv_job_async(), using check_job_complete() to poll the server.

**Parameters**

- **jobType** – Job type key, as defined in the pb.JobInput.RunType enum (defaults to 'energy')
- **geom** – Cartesian geometry of the new point
- **unitType** – Unit type key, as defined in the pb.Mol.UnitType enum (defaults to 'bohr')
- ****kwargs** – Additional TeraChem keywords, check _process_kwargs for behaviour

**Returns** Results mirroring recv_job_async

**Return type** dict

**connect**()

Connect to the TeraChem Protobuf server

**disconnect**()

Disconnect from the TeraChem Protobuf server

**is_available**()

Asks the TeraChem Protobuf server whether it is available or busy through the Status protobuf message. Note that this does not reserve the server, and the status could change after this function is called.

**Returns** True if the TeraChem PB server is currently available (no running job)

**Return type** bool

**recv_job_async**()

Recv and unpack a JobOutput message from the TeraChem Protobuf server asynchronously. This function expects the job to be ready (i.e. check_job_complete() returned true), so will error out on timeout.

Creates a results dictionary that mirrors the JobOutput message, using NumPy arrays when appropriate. Results are also saved in the prev_results class member. An inclusive list of the results members (with types):

- atoms: Flat # of atoms NumPy array of 2-character strings
- geom: # of atoms by 3 NumPy array of doubles

- energy: Either empty, single energy, or flat # of cas_energy_labels of NumPy array of doubles
- charges: Flat # of atoms NumPy array of doubles
- spins: Flat # of atoms NumPy array of doubles
- dipole_moment: Single element
- dipole_vector: Flat 3-element NumPy array of doubles
- job_dir: String
- job_scr_dir: String
- server_job_id: Int
- orbfile: String (if restricted is True, otherwise not included)
- orbfile_a: String (if restricted is False, otherwise not included)
- orbfile_b: String (if restricted is False, otherwise not included)
- orb_energies: Flat # of orbitals NumPy array of doubles (if restricted is True, otherwise not included)
- orb_occupations: Flat # of orbitals NumPy array of doubles (if restricted is True, otherwise not included)
- orb_energies_a: Flat # of orbitals NumPy array of doubles (if restricted is False, otherwise not included)
- orb_occupations_a: Flat # of orbitals NumPy array of doubles (if restricted is False, otherwise not included)
- orb_energies_b: Flat # of orbitals NumPy array of doubles (if restricted is False, otherwise not included)
- orb_occupations_b: Flat # of orbitals NumPy array of doubles (if restricted is False, otherwise not included)

Additional (optional) members of results:

- gradient: # of atoms by 3 NumPy array of doubles (available for 'gradient' job)
- nacme: # of atoms by 3 NumPy array of doubles (available for 'coupling' job)
- transition_dipole: Flat 3-element NumPy array of doubles (available for 'coupling' job)
- cas_energy_labels: List of tuples of (state, multiplicity) corresponding to the energy list
- bond_order: # of atoms by # of atoms NumPy array of doubles
- ci_overlap: ci_overlap_size by ci_overlap_size NumPy array of doubles (available for 'ci_vec_overlap' job)

**Returns** Results as described above

**Return type** dict

**send_job_async**(*jobType='energy'*, *geom=None*, *unitType='bohr'*, ***kwargs*)

Pack and send the current JobInput to the TeraChem Protobuf server asynchronously. This function expects a Status message back that either tells us whether the job was accepted.

**Parameters**

- **jobType** – Job type key, as defined in the pb.JobInput.RunType enum (defaults to "energy")

- **geom** – Cartesian geometry of the new point
- **unitType** – Unit type key, as defined in the pb.Mol.UnitType enum (defaults to “bohr”)
- ****kwargs** – Additional TeraChem keywords, check _process_kwargs for behaviour

**Returns** True on job acceptance, False on server busy, and errors out if communication fails

**Return type** bool

**update_address**(*host*, *port*)

Update the host and port of a TCProtobufClient object. Note that you will have to call disconnect() and connect() before and after this yourself to actually connect to the new server.

**Parameters**

- **host** (*str*) – Hostname
- **port** (*int*) – Port number (must be above 1023)

### 2.1.3 Module contents

CHAPTER

THREE

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## t

tcpb, **??**
tcpb.tcpb, **??**