envipyengine Documentation

Release

Exelis VIS

Nov 08, 2017

Contents

1 Usage **3**

2 API Documentation **5**
- 2.1 ENVI Py Engine 5
- 2.2 ENVI Py Engine Task 6
- 2.3 ENVI Py Engine Config 7

Python Module Index **11**

ENVI Py Engine provides a client Python package, named envipyengine, to run ENVI analytics provided by ENVI Desktop. The Python package provides the ability to query for available tasks, retrieve task information, and execute tasks on the desktop.

There is an additional client Python package available, named envipyarc, to provide the ability to run ENVI analytics through ArcMap and ArcGIS Pro.

See http://www.harrisgeospatial.com/ for more details on product offerings.

CHAPTER 1

Usage

Before using ENVI Py Engine, you must first configure the package so it can find your Engine executable within your ENVI/IDL installation. To do this make sure the "engine" config option is set:

```
>>> import envipyengine
>>> engine = envipyengine.config.get('engine')
```

If the above command throws an error, then you will need to set the 'engine' property to the the full path of the 'taskengine' executable in your ENVI installation:

```
>>> envipyengine.config.set('engine', <path-to-executable>)
```

When specifying paths in Python strings on Windows, be sure to use two backslashes as your directory separator.

To connect to the ENVI Task Engine and list the available tasks, create a new instance of the Engine class with the engine name from the Python command line:

```
>>> from envipyengine import Engine
>>> envi_engine = Engine('ENVI')
>>> envi_engine.tasks()
```

You must have write permissions for Python's current working directory in order to run the examples.

To get an ENVI task object, use the `task()` method on the Engine object:

```
>>> task = envi_engine.task('SpectralIndex')
```

To get a list of task parameter information, use the `parameters` property on the Task object:

```
>>> task.parameters
```

To execute a task, use the `execute()` method on the Task object. A GSF Job object is returned after the job has been submitted:

```
>>> input_raster = dict(url='<path_to_input_raster>', factory='URLRaster')
>>> parameters = dict(INPUT_RASTER=input_raster,
                      INDEX='Normalized Difference Vegetation Index')
>>> task.execute(parameters)
```

CHAPTER 2

API Documentation

ENVI Py Engine

The ENVI Py Engine object selects a task engine to use with ENVI Py

class envipyengine.engine.**Engine**(*engine_name*, *cwd=None*)

The ENVI Py Engine Class.

Example

Import the module for the example

```
>>> from envipyengine import Engine
>>> from pprint import pprint
```

Create an ENVI Engine and print a list of available tasks.

```
>>> envi_engine = Engine('ENVI')
>>> tasks = envi_engine.tasks()
>>> pprint(tasks)
['AdditiveLeeAdaptiveFilter',
 'AdditiveMultiplicativeLeeAdaptiveFilter',
 'ApplyGainOffset',
 ...
```

name

Returns the name of the task engine associated with the Engine.

Returns The task engine name (i.e. ENVI, IDL, etc.)

task(*task_name*)

Returns an ENVI Py Engine Task object. See ENVI Py Engine Task for examples.

Parameters **task_name** – The name of the task to retrieve.

Returns An ENVI Py Engine Task object.

tasks()

Returns a list of all tasks known to the engine.

Returns A list of task names.

ENVI Py Engine Task

The ENVI Py Engine task object provides task information and can submit a job to the Task Engine with input.

class envipyengine.task.**Task**(*uri=None*, *cwd=None*)

The ENVI Py Engine Task object represents a Task Engine task and its parameters.

Example

Import the modules for the example.

```
>>> from envipyengine import Engine
>>> from pprint import pprint
```

Create an Engine object for ENVI and get a task

```
>>> envi_engine = Engine('ENVI')
>>> task = envi_engine.task('SpectralIndex')
```

Investigate task information.

```
>>> print(task.name)
'SpectralIndex'
>>> print(task.description, type(task.description))
('This task creates a spectral index raster from one pre-defined spectral
index. Spectral indices are combinations of surface reflectance at two
or more wavelengths that indicate relative abundance of features of
interest.', <type 'str'>)
>>> print(task.display_name, type(task.display_name))
('Spectral Index', <type 'str'>)
>>> task_parameters = task.parameters
>>> pprint(task_parameters)
```

Execute a job using the Task Engine

```
>>> input_raster = dict(url='C:\Program Files\Harris\ENVI54\data\qb_boulder_msi',
                        factory='URLRaster')
>>> parameters = dict(INPUT_RASTER=input_raster,
                      INDEX='Normalized Difference Vegetation Index')
>>> result = task.execute(parameters)
```

description

The task description

Returns a string

display_name

The display name of the task

Returns a string

execute(*parameters*, *cwd=None*)

Executes a synchronous task using the Task Engine

Parameters

- **parameters** – A dictionary of key-value pairs of parameter names and values. The dictionary serves as input to the job.
- **cwd** – Set to the current working directory the engine will run in. Defaults to the python current working directory if none specified.

Returns A dictionary containing the Task Engine output.

name
The name of the task

Returns a string

parameters
A list of the task parameter definitions. Each task parameter is a dictionary containing, but not limited to, the following keys:

Key	Data Type	Type	Description
name	string	Required	The name of the parameter
display_name	string	Required	The display name of the parameter
type	string	Required	The parameter data type
direction	string	Required	Can be *input* or *output*
description	string	Required	The parameter description
required	bool	Required	Indicates if the parameter is required on input when submitting a job
dimensions	string	Optional	Indicates if the parameter is an array if set. Dimesions is of the format *[dim1,dim2,...]*
choice_list	list	Optional	A list of available choices for the parameter input
min	type	Optional	The minimum value allowed for the parameter
max	type	Optional	The maximum value allowed for the parameter

Returns a list of parameter dictionaries

uri
The task unique identifier

ENVI Py Engine Config

The config module is used to configure settings for ENVI Py Engine.

The following properties are currently supported:

Property Name	Data Type	Description
engine	string	The full path to the engine executable. Example: "C:\Program Files\Harris\ENVI54\IDL86\bin\bin.x86_64\taskengine.exe"
engine-args	string	Any additional command line arguments that will be passed to the taskengine executable.
Environment Variable	string	Any valid environment variable and value pairs.
Names		All name/value pairs specified in this section will be interpreted as environment variables for use when running the Engine.

Please refer to the following examples of setting configuration values.

Set the Engine Executable path for the current user:

```
>>> import envipyengine
>>> envipyengine.config.set('engine', <executable-path>)
```

Set the ENVI Engine Executable path for all users.

```
>>> import envipyengine
>>> envipyengine.config.set('engine', <install-dir>, system=True)
```

Specify additional arguments for the Task Engine:

```
>>> import envipyengine
>>> envipyengine.config.set('engine-args', '--compile')
```

Specify an environment variable to be used when running the task engine:

```
>>> import envipyengine
>>> envipyengine.config.set_environment(dict('IDL_PATH'=<path-to-idl-code>)
```

The locations of the configuration files are:

OS and Configuration Type	Configuration File
Windows User Configuration	C:\Users\<user>\AppData\Local\envipyengine\settings.cfg
Windows System Configuration	C:\ProgramData\envipyengine\settings.cfg
Mac OS X User Configuration	/Users/<user>/Library/Preferences/envipyengine/settings.cfg
Max OS X System Configuration	/Library/Preferences/envipyengine/settings.cfg
Linux User Configuration	/home/<user>/.envipyengine/settings.cfg
Linux System Configuration	/var/lib/envipyengine/settings.cfg

envipyengine.config.**get**(*property_name*)

Returns the value of the specified configuration property. Property values stored in the user configuration file take precedence over values stored in the system configuration file.

Parameters **property_name** – The name of the property to retrieve.

Returns The value of the property.

envipyengine.config.**get_environment**()

Return all environment values from the config files. Values stored in the user configuration file will take precedence over values stored in the system configuration file.

Returns A dictionary containing the name/value pairs of all environment settings in the config file.

envipyengine.config.**remove**(*property_name*, *system=False*)

Remove a configuration property/value setting from the config file.

Parameters

- **property_name** – The name of the property to remove.
- **system** – Set to True to modify the system configuration file. If not set, the user config file will be modified.

envipyengine.config.**remove_environment**(*environment_var_name*, *system=False*)

Remove the specified environment setting from the appropriate config file.

Parameters

- **environment_var_name** – The name of the environment setting to remove.
- **system** – Set to True to modify the system configuration file. If not set, the user config file will be modified.

envipyengine.config.**set**(*property_name*, *value*, *system=False*)
Sets the configuration property to the specified value.

Parameters

- **property_name** – The name of the property to set.
- **value** – The value for the property.
- **system** – Set to True to modify the system configuration file. If not set, the user config file will be modified.

envipyengine.config.**set_environment**(*environment*, *system=False*)
Set engine environment values in the config file.

Parameters

- **environment** – A dictionary containing the environment variable settings as key/value pairs.
- **system** – Set to True to modify the system configuration file. If not set, the user config file will be modified.

e

envipyengine.config, 7
envipyengine.engine, 5
envipyengine.task, 6

D

description (envipyengine.task.Task attribute), 6
display_name (envipyengine.task.Task attribute), 6

E

Engine (class in envipyengine.engine), 5
envipyengine.config (module), 7
envipyengine.engine (module), 5
envipyengine.task (module), 6
execute() (envipyengine.task.Task method), 6

G

get() (in module envipyengine.config), 8
get_environment() (in module envipyengine.config), 8

N

name (envipyengine.engine.Engine attribute), 5
name (envipyengine.task.Task attribute), 7

P

parameters (envipyengine.task.Task attribute), 7

R

remove() (in module envipyengine.config), 8
remove_environment() (in module envipyengine.config), 8

S

set() (in module envipyengine.config), 8
set_environment() (in module envipyengine.config), 9

T

Task (class in envipyengine.task), 6
task() (envipyengine.engine.Engine method), 5
tasks() (envipyengine.engine.Engine method), 5

U

uri (envipyengine.task.Task attribute), 7