Pacifica Python Uploader Documentation

David Brown

Jan 25, 2019

Contents:

Pacifica Python Uploader, *pacifica-uploader*, is a Python programming language library for managing, serializing and transporting (over a network) archives of files (referred to as "bundles"), managing both the data and the metadata of the bundle, and interacting with Pacifica Ingest and Pacifica Policy servers.

CHAPTER 1

Installation

The Pacifica software is available through PyPi so creating a virtual environment to install is what is shown below. Please keep in mind compatibility with the Pacifica Core services.

1.1 Installation in Virtual Environment

These installation instructions are intended to work on both Windows, Linux, and Mac platforms. Please keep that in mind when following the instructions.

Please install the appropriate tested version of Python for maximum chance of success.

1.1.1 Linux and Mac Installation

```
mkdir ~/.virtualenvs
python -m virtualenv ~/.virtualenvs/pacifica
. ~/.virtualenvs/pacifica/bin/activate
pip install pacifica-uploader
```

1.1.2 Windows Installation

This is done using PowerShell. Please do not use Batch Command.

```
mkdir "$Env:LOCALAPPDATA\virtualenvs"
python.exe -m virtualenv "$Env:LOCALAPPDATA\virtualenvs\pacifica"
& "$Env:LOCALAPPDATA\virtualenvs\pacifica\Scripts\activate.ps1"
pip install pacifica-uploader
```

CHAPTER 2

Uploader Metadata Configuration

The uploader configuration begins with an array of metadata objects. The attributes of each object and how an uploader should manipulate them are documented below. Much of the attributes define how a query should be given to the policy server and how the user should be presented with the results so they can make a choice.

Example Metadata Configuration Snippet:

```
{
  "destinationTable": "Transactions.submitter",
  "displayFormat": "{_id} - {first_name} {last_name}",
  "displayTitle": "Currently Logged On",
  "displayType": "logged_on",
  "metaID": "logon",
  "queryDependency": {},
  "queryFields": [
    "first_name",
    "last_name",
    "_id"
  ],
  "sourceTable": "users",
  "value": "",
  "valueField": "_id"
}
```

- Destination Table - `destinationTable`

The destination table and column for the value to be put into. The value is a string of the format `TABLE.COLUMN`.

- Display Format - `displayFormat`

The formatted string to show the user an entry for data from the `sourceTable`. This is uploader independent and uses string formatting specific to Python (in this implementation) for rendering the string.

- Display Title - `displayTitle`

The title for the resulting data returned from the query.

- Display Type - `displayType`

This is for the uploader to choose the values for. This may represent a select drop down list, a radio button options or whatever the uploader would like to present to the user.

- Metadata ID - `metaID`

This is the unique ID for the metadata in the system. This should be a unique string for all metadata objects for the entire configuration.

- Query Dependencies - `queryDependency`

This is a hash containing the dependencies for the query and where to find the values in the current metadata configuration. The hash is a `column` to `metaID` mapping. These dependencies are passed as `where` arguments to the policy query.

- Query Fields - `queryFields`

This is a list of columns from the source field to pull in as part of the query. These will be given to the `displayFormat` string to render the entry for users to pick.

- Source Table - `sourceTable`

The source table from which the query will be requesting data from.

- Result Value - `value`

The value of the `valueField` column from the `sourceTable` to be put into the `destinationTable` for the upload.

- Value Field - `valueField`

The value field to be put into the table and column defined by `destinationTable`.

CHAPTER 3

Uploader Expectations and Application Flows

This section describes how an end-user of Pacifica Python Uploader is expected to interact with the modules, classes and methods above, and, by extension, Pacifica Ingest and Pacifica Policy servers.

Keywords for the API

```
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD",
"SHOULD NOT", "RECOMMENDED",  "MAY", and "OPTIONAL" in this document are to be
interpreted as described in [RFC 2119](https://www.ietf.org/rfc/rfc2119.txt).
```

3.1 Uploader Program Flow

1. The uploader program MUST construct a new instance of the `pacifica.uploader.metadata.MetaUpdate` class. The new instance of the `pacifica.uploader.metadata.MetaUpdate` class MAY be associated with zero or more of instances of the `pacifica.uploader.metadata.MetaObj` class. The `pacifica.uploader.metadata.MetaObj.value` field MAY be `None`. The new instance of the `pacifica.uploader.metadata.MetaUpdate` class MUST NOT be associated with any instances of the `pacifica.uploader.metadata.FileObj` class.
2. To determine completeness, the new instance of the `pacifica.uploader.metadata.MetaUpdate` class SHOULD be validated using the `pacifica.uploader.metadata.MetaData.is_valid()` method (inherited by the `pacifica.uploader.metadata.MetaUpdate` sub-class). Then, the uploader program MUST call the `pacifica.uploader.metadata.PolicyQuery.PolicyQuery.valid_metadata()` method. The new instance of the `pacifica.uploader.metadata.MetaUpdate` class MUST be valid prior to bundling.
3. The uploader program MUST dereference the `pacifica.uploader.metadata.MetaObj.displayType` field to determine the mode of selection for the `pacifica.uploader.metadata.MetaObj.value` field. The value of the `pacifica.uploader.metadata.MetaObj.displayType` field is uploader-program-specific, i.e., the value MUST be defined by the uploader program.
4. The uploader program MUST assign a non-`None` value to each `pacifica.uploader.metadata.MetaData.query_results` field by calling the `pacifica.uploader.metadata.MetaUpdate.`

query_results() method. The pacifica.uploader.metadata.MetaData.query_results field is a list.

5. The value of the pacifica.uploader.metadata.MetaData.query_results field MUST be rendered according to the uploader-program-specific definition that is interpreted from the value of the pacifica.uploader.metadata.MetaObj.displayFormat field, e.g., in the Python programming language, by calling the str.format method or by leveraging a template engine, such as Cheetah or Jinja2.

6. The uploader program MAY call the pacifica.uploader.metadata.MetaUpdate.query_results() method for instances of the pacifica.uploader.metadata.MetaObj class whose value field is non-None.

7. The uploader program MUST handle all instances pacifica.uploader.metadata.MetaUpdate class, regardless of validity, i.e., the uploader program MUST NOT reject an instance of the pacifica.uploader.metadata.MetaUpdate class under any circumstances, e.g., if there are unsatisfied dependencies between instances of the pacifica.uploader.metadata.MetaData class.

8. When the uploader program is ready for a given pacifica.uploader.metadata.MetaObj.value field to be selected, the uploader program MUST assign to the pacifica.uploader.metadata.MetaObj.value field the value of the pacifica.uploader.metadata.MetaObj.valueField field, and then call the pacifica.uploader.metadata.MetaObj.update_parents() method. The effect of this operation is to update the pacifica.uploader.metadata.MetaObj.value fields of associated and dependent instances of the pacifica.uploader.metadata.MetaObj class. After modification, the new state of the instance of the pacifica.uploader.metadata.MetaUpdate.MetaUpdate class SHOULD be displayed to the end-user, as previously discussed.

9. The uploader program MUST verify that pacifica.uploader.metadata.MetaUpdate.MetaUpdate.is_valid() == True. If the instance of the pacifica.uploader.metadata.MetaUpdate.MetaUpdate class is not valid, then the uploader program MUST repeat the instructions in the paragraph 8.

10. The uploader program MUST call the pacifica.uploader.metadata.PolicyQuery.PolicyQueryData.valid_metadata() method to validate the instance of the pacifica.uploader.metadata.MetaUpdate.MetaUpdate class prior to upload. This prevents the uploader program from uploading metadata that is invalid with respect to the policy of the Pacifica Ingest server.

11. When the uploader program is ready to bundle the data, the uploader program MUST construct a list of objects, representing the fields of the corresponding instance of the tar.TarInfo class. Each object MUST export a fileobj field whose value implements the file protocol, i.e., exports a read() method.

12. The uploader program MUST construct a new instance of the pacifica.uploader.bundler.Bundler class using the instances of the pacifica.uploader.metadata.MetaUpdate.MetaUpdate and tar.TarInfo classes, as previously stated in paragraph 11. Then, the uploader program MUST construct a file-like object that can be written to in binary mode, and then call the pacifica.uploader.bundler.Bundler.stream() method.

13. The uploader program MUST construct a new instance of the pacifica.uploader.Uploader.Uploader class. Then, the uploader program MUST construct a file-like object that can be read in binary mode, and then call the pacifica.uploader.bundler.Bundler.upload() method.

14. Finally, the uploader program MUST verify the result of the ingest by calling the pacifica.uploader.Uploader.Uploader.getstate() method. If an ingest-related error occurs, then the uploader program MAY repeat the ingest operation.

CHAPTER 4

Uploader Python Module

4.1 Bundler Python Module

4.1.1 Bundler Python Module

Main Bundler module containing classes and methods to handle bundling.

class pacifica.uploader.bundler.bundler.**Bundler**(*md_obj*, *file_data*, ***kwargs*)
Class to handle bundling of files to stream a tarfile.

__init__(*md_obj*, *file_data*, ***kwargs*)
Constructor of the bundler class.

Add the MetaData object *md_obj* and file *file_data* to create. The *file_data* object should be a list of hashes. That are fed to TarInfo objects except for fileobj which is passed to addfile method.

Note: The `arcname` keyword argument MUST be provided when calling the `tarfile.TarFile.gettarinfo()` method.

Example MetaData Obj:

```
[
  {
    'name': 'archive file path',
    'fileobj': 'open file object for read',
    'size': 'size of the file',
    'mtime': 'modify time of the file'
  }
]
```

__weakref__
list of weak references to the object (if defined)

_build_file_info(*file_data*, *hashsum*)
Build the FileObj to and return it.

_save_total_size()
Build the total size from the files and save the total.

_setup_notify_thread(*callback*, *sleeptime=5*)
Setup a notification thread calling callback with percent complete.

static _strip_subdir(*fname*)
Remove the data subdir from the file path.

_tarinfo_from_file_data(*file_data*)
Return a tarinfo object from file_data.

stream(*fileobj*, *callback=None*, *sleeptime=5*)
Stream the bundle to the fileobj.

This method is a blocking I/O operation. The `fileobj` should be an open file like object with 'wb' options. An asynchronous callback method MAY be provided via the optional `callback` keyword argument. Periodically, the callback method is provided with the current percentage of completion.

class pacifica.uploader.bundler.bundler.**HashFileObj**(*filedesc*, *hashval*, *upref*)
File like object used for reading and hashing files.

__init__(*filedesc*, *hashval*, *upref*)
Create the hash file object.

__weakref__
list of weak references to the object (if defined)

hashdigest()
Return the hash digest for the file.

read(*size=-1*)
Read wrapper function.

This is the bundler library.

This module exports classes and methods for constructing and streaming bundles of files to a designated file descriptor. The file descriptor is opened once, and the stream is generated by a single pass over the specified files.

class pacifica.uploader.bundler.**Bundler**(*md_obj*, *file_data*, ***kwargs*)
Class to handle bundling of files to stream a tarfile.

__init__(*md_obj*, *file_data*, ***kwargs*)
Constructor of the bundler class.

Add the MetaData object *md_obj* and file *file_data* to create. The *file_data* object should be a list of hashes. That are fed to TarInfo objects except for fileobj which is passed to addfile method.

Note: The `arcname` keyword argument MUST be provided when calling the `tarfile.TarFile.gettarinfo()` method.

Example MetaData Obj:

```
[
  {
    'name': 'archive file path',
    'fileobj': 'open file object for read',
    'size': 'size of the file',
    'mtime': 'modify time of the file'
  }
]
```

__weakref__
list of weak references to the object (if defined)

_build_file_info(*file_data*, *hashsum*)
Build the FileObj to and return it.

_save_total_size()
Build the total size from the files and save the total.

_setup_notify_thread(*callback*, *sleeptime=5*)
Setup a notification thread calling callback with percent complete.

static _strip_subdir(*fname*)
Remove the data subdir from the file path.

_tarinfo_from_file_data(*file_data*)
Return a tarinfo object from file_data.

stream(*fileobj*, *callback=None*, *sleeptime=5*)
Stream the bundle to the fileobj.

This method is a blocking I/O operation. The `fileobj` should be an open file like object with 'wb' options. An asynchronous callback method MAY be provided via the optional `callback` keyword argument. Periodically, the callback method is provided with the current percentage of completion.

4.2 Common Python Module

Common uploader functionality.

class pacifica.uploader.common.**CommonBase**
Contains methods to implement common functionality.

__weakref__
list of weak references to the object (if defined)

_server_url(*parts*, *env_prefix*, *kwargs*)
Server URL parsing for init class method.

_setup_requests_session()
Setup a requests retry session so we can talk to http services.

4.3 Metadata Python Module

4.3.1 Metadata Python Module

MetaData class to handle input and output of metadata format.

class pacifica.uploader.metadata.metadata.**FileObj**
FileObj class for holding file metadata.

Instances of this class represent individual files, including both the data and metadata for the file. During a file upload, instances of this class are automatically associated with new instances of the `pacifica.uploader.metadata.MetaData` class.

The above named fields are identical to those of the `pacifica.metadata.orm.Files` class, provided by the Pacifica Metadata library.

class pacifica.uploader.metadata.metadata.**MetaData**(**args*, ***kwargs*)

Class to hold a list of MetaObj and FileObj objects.

This class is a sub-class of `list` that implements the index protocol (`__getitem__`, `__setitem__` and `__delitem__`) as a proxy to the indices of the value of the `metaID` field of the associated instance of the `pacifica.uploader.metadata.MetaObj` class.

Instances of this class are upper-level objects that provide the metadata for interacting with the designated Pacifica Ingest server.

__delitem__(*key*)

Delete the item from the array and hash.

__getitem__(*key*)

Get the node based on metaID.

__init__(**args*, ***kwargs*)

Call the super constructor and add a metaID index to it as well.

__setitem__(*key*, *value*)

Set the item and if metaID exists save the index into a map.

__weakref__

list of weak references to the object (if defined)

append(*value*)

Append the value to the list.

extend(*iterable*)

Extend the array from the values in iterable.

insert(*key*, *value*)

Insert the value to the list.

is_valid()

Return true if all the values of MetaObjs are something.

pop(*key=-1*)

Remove the key from the list and return it.

remove(*value*)

Remove the value from the list.

class pacifica.uploader.metadata.metadata.**MetaDataDecoder**(**, object_hook=None, parse_float=None, parse_int=None, parse_constant=None, strict=True, object_pairs_hook=None*)

Class to decode a json string into a MetaData object.

decode(*s*)

Decode the string into a MetaData object.

class pacifica.uploader.metadata.metadata.**MetaDataEncoder**(**, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, sort_keys=False, indent=None, separators=None, default=None*)

Class to encode a MetaData object into json.

encode(*o*)
Encode the MetaData object into a json list.

class pacifica.uploader.metadata.metadata.**MetaObj**
MetaObj class holding a specific metadata element.

Instances of this class represent units of metadata whose representation is disjoint to a file, i.e., units of metadata that are describe but are not stored as part of a file.

pacifica.uploader.metadata.metadata.**_FileObj**
alias of *pacifica.uploader.metadata.metadata.FileObj*

pacifica.uploader.metadata.metadata.**_MetaObj**
alias of *pacifica.uploader.metadata.metadata.MetaObj*

pacifica.uploader.metadata.metadata.**file_or_meta_obj**(***json_data*)
Determine if this is a File or Meta object and return result.

pacifica.uploader.metadata.metadata.**metadata_decode**(*json_str*)
Decode the json string into MetaData object.

This method deserializes the given JSON source, `json_str`, and then returns a new instance of the `pacifica.uploader.metadata.MetaData` class.

The new instance is automatically associated with new instances of the `pacifica.uploader.metadata.MetaObj` and `pacifica.uploader.metadata.FileObj` classes.

pacifica.uploader.metadata.metadata.**metadata_encode**(*md_obj*)
Encode the MetaData object into a json string.

This method encodes the given instance of the `pacifica.uploader.metadata.MetaData` class, `md_obj`, as a JSON object, and then returns its JSON serialization.

Associated instances of the `pacifica.uploader.metadata.MetaObj` and `pacifica.uploader.metadata.FileObj` classes are automatically included in the JSON object and the resulting JSON serialization.

4.3.2 Meta Update Python Module

Module used to update MetaData objects.

This module exports classes and methods for constructing and executing the strategy for modifying the values, including the parents and children, of instances of the *pacifica.uploader.metadata.MetaData* class.

class pacifica.uploader.metadata.metaupdate.**MetaUpdate**(*user*, **args*, ***kwargs*)
Class to update the MetaData object.

This class is a sub-class of the `pacifica.uploader.metadata.MetaData` class that is specialized to issue and handle queries to Pacifica Policy servers.

__init__(*user*, **args*, ***kwargs*)
Pull the user from the arguments so we can use that for policy queries.

dependent_meta_id(*meta_id*)
Get the dependent meta ID.

directory_prefix()
Return the directory prefix of the MetaObjs which have directoryOrder.

get_auth()
Return the auth object to be used by other instances.

query_results(*meta_id*)
Build a PolicyQuery out of the meta_id.

This method creates a `pacifica.uploader.metadata.PolicyQuery` object that queries the policy server and returns the results.

update_parents(*meta_id*)
Update the parents of the meta_id.

4.3.3 MJSON Python Module

Encode and decode objects into json.

This module exports generators for encoding and decoding instances of the *collections.namedtuple* class using the JSON data format.

`pacifica.uploader.metadata.mjson.`**generate_namedtuple_decoder**(*cls*)
Return a namedtuple decoder for the class cls.

Generate a sub-class of `json.JSONDecoder`, which decodes a JSON object into an instance of `cls`.

`pacifica.uploader.metadata.mjson.`**generate_namedtuple_encoder**(*cls*, *mangle=<function strip_obj>*)
Return a namedtuple encoder for class cls.

Generate a sub-class of `json.JSONEncoder`, which encodes the instances of `cls` as a JSON object.

`pacifica.uploader.metadata.mjson.`**strip_obj**(*obj*)
Remove all keys who's values are False.

4.3.4 Policy Query Python Module

This is the module for quering the Policy service.

This module exports classes and methods for interacting with the designated Pacifica Policy server.

class `pacifica.uploader.metadata.policyquery.`**PolicyQuery**(*user*, **args*, ***kwargs*)
Handle quering the policy server.

Instances of this class represent queries to the designated Pacifica Policy server.

__init__(*user*, **args*, ***kwargs*)
Set the policy server url and define any data for the query.

The HTTP end-point for the policy server is automatically pulled either from the system environment or from the keyword arguments, ***kwargs*.

_set_url_from_parts()
Set the url from the parts in self.

static fromjson(*json_str*)
Import json string to self.

get_results()
Get results from the Policy server for the query.

This method returns a JSON object that is the result set for a query to the Pacifica Policy server, i.e., the entities that match the criteria that is represented by the associated instance of the `pacifica.uploader.metadata.PolicyQuery.PolicyQueryData` class.

get_user()
Get the user id.

set_user(*user*)
Set the user for the current PolicyQuery.

tojson()
Export self to json.

valid_metadata(*md_obj*)
Check the metadata object against the ingest API.

This method validates the given instance of `pacifica.uploader.metadata.MetaData`, `md_obj`, against the Pacifica Policy server endpoint.

class pacifica.uploader.metadata.policyquery.**PolicyQueryData**
Policy query data elements for policy query requests.

This class is a sub-class of the *collections.namedtuple* class. This class is used directly against the Pacifica Uploader Policy endpoint.

pacifica.uploader.metadata.policyquery.**_PolicyQueryData**
alias of *pacifica.uploader.metadata.policyquery.PolicyQueryData*

pacifica.uploader.metadata.policyquery.**_mangle_decode**(***json_data*)
Mangle the decode of the policy query object.

pacifica.uploader.metadata.policyquery.**_mangle_encode**(*obj*)
Move the from_table to just from.

This is the metadata library.

The *pacifica.uploader.metadata* module exports classes and methods for manipulating and serializing the metadata for bundles of files.

Encoding and decoding to the JSON data format is supported for compatible objects (see *pacifica.uploader.metadata.Json* module for more information).

class pacifica.uploader.metadata.**MetaData**(**args*, ***kwargs*)
Class to hold a list of MetaObj and FileObj objects.

This class is a sub-class of `list` that implements the index protocol (`__getitem__`, `__setitem__` and `__delitem__`) as a proxy to the indices of the value of the `metaID` field of the associated instance of the `pacifica.uploader.metadata.MetaObj` class.

Instances of this class are upper-level objects that provide the metadata for interacting with the designated Pacifica Ingest server.

__delitem__(*key*)
Delete the item from the array and hash.

__getitem__(*key*)
Get the node based on metaID.

__init__(**args*, ***kwargs*)
Call the super constructor and add a metaID index to it as well.

__setitem__(*key*, *value*)
Set the item and if metaID exists save the index into a map.

__weakref__
list of weak references to the object (if defined)

append(*value*)
Append the value to the list.

extend(*iterable*)
Extend the array from the values in iterable.

insert(*key*, *value*)
Insert the value to the list.

is_valid()
Return true if all the values of MetaObjs are something.

pop(*key=-1*)
Remove the key from the list and return it.

remove(*value*)
Remove the value from the list.

class pacifica.uploader.metadata.**MetaObj**
MetaObj class holding a specific metadata element.

Instances of this class represent units of metadata whose representation is disjoint to a file, i.e., units of metadata that are describe but are not stored as part of a file.

class pacifica.uploader.metadata.**FileObj**
FileObj class for holding file metadata.

Instances of this class represent individual files, including both the data and metadata for the file. During a file upload, instances of this class are automatically associated with new instances of the `pacifica.uploader.metadata.MetaData` class.

The above named fields are identical to those of the `pacifica.metadata.orm.Files` class, provided by the Pacifica Metadata library.

class pacifica.uploader.metadata.**MetaUpdate**(*user*, **args*, ***kwargs*)
Class to update the MetaData object.

This class is a sub-class of the `pacifica.uploader.metadata.MetaData` class that is specialized to issue and handle queries to Pacifica Policy servers.

__init__(*user*, **args*, ***kwargs*)
Pull the user from the arguments so we can use that for policy queries.

dependent_meta_id(*meta_id*)
Get the dependent meta ID.

directory_prefix()
Return the directory prefix of the MetaObjs which have directoryOrder.

get_auth()
Return the auth object to be used by other instances.

query_results(*meta_id*)
Build a PolicyQuery out of the meta_id.

This method creates a `pacifica.uploader.metadata.PolicyQuery` object that queries the policy server and returns the results.

update_parents(*meta_id*)
Update the parents of the meta_id.

pacifica.uploader.metadata.**metadata_encode**(*md_obj*)
Encode the MetaData object into a json string.

This method encodes the given instance of the `pacifica.uploader.metadata.MetaData` class, `md_obj`, as a JSON object, and then returns its JSON serialization.

Associated instances of the `pacifica.uploader.metadata.MetaObj` and `pacifica.uploader.metadata.FileObj` classes are automatically included in the JSON object and the resulting JSON serialization.

`pacifica.uploader.metadata.`**`metadata_decode`**(*json_str*)
Decode the json string into MetaData object.

This method deserializes the given JSON source, `json_str`, and then returns a new instance of the `pacifica.uploader.metadata.MetaData` class.

The new instance is automatically associated with new instances of the `pacifica.uploader.metadata.MetaObj` and `pacifica.uploader.metadata.FileObj` classes.

4.4 Uploader Python Module

Uploader module send the data to the ingest service.

This module exports classes and methods for interacting with Pacifica Ingest servers.

class `pacifica.uploader.uploader.`**`Uploader`**(***kwargs*)
Uploader class to upload the bundle to an ingest server.

This class exports methods that provide an API for connecting to and handling connections to Pacifica Ingest servers.

`__init__`(***kwargs*)
Set the ingest endpoint url.

`getstate`(*job_id*)
Get the ingest state for a job.

This method takes a `job_id` as input, and returns a JSON object, as defined by the Pacifica Ingest API for obtaining the status of the current job.

`upload`(*read_fd*, *content_length=None*)
Upload the data from a file like object.

This method takes a file-like object as input that has been opened for reading in binary mode, and returns a `job_id` for the upload.

This is the uploader library.

This section gives an overview of the modules, classes and methods that are exported by the Pacifica Python Uploader library: *PacificaUploader*.

class `pacifica.uploader.`**`Uploader`**(***kwargs*)
Uploader class to upload the bundle to an ingest server.

This class exports methods that provide an API for connecting to and handling connections to Pacifica Ingest servers.

`__init__`(***kwargs*)
Set the ingest endpoint url.

`getstate`(*job_id*)
Get the ingest state for a job.

This method takes a `job_id` as input, and returns a JSON object, as defined by the Pacifica Ingest API for obtaining the status of the current job.

upload(*read_fd*, *content_length=None*)
Upload the data from a file like object.

This method takes a file-like object as input that has been opened for reading in binary mode, and returns a `job_id` for the upload.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

Symbols