
JSONRPC Documentation

Release 1.0

Edward Langley

March 29, 2016

1	Getting Started	3
2	JSON-RPC Server	5
3	JSON-RPC Proxy	7
4	Indices and tables	9
	Python Module Index	11

Contents:

Getting Started

This code will create a server that logs all requests, and provides two methods to clients: add and subtract:

```
1 #
2 # Copyright (c) 2011 Edward Langley
3
4 from twisted.internet import reactor, ssl
5 from twisted.web import server
6 import traceback
7
8 from jsonrpc.server import ServerEvents, JSON_RPC
9
10 class ExampleServer(ServerEvents):
11     # inherited hooks
12     def log(self, responses, txrequest, error):
13         print(txrequest.code, end=' ')
14         if isinstance(responses, list):
15             for response in responses:
16                 msg = self._get_msg(response)
17                 print(txrequest, msg)
18         else:
19             msg = self._get_msg(responses)
20             print(txrequest, msg)
21
22     def findmethod(self, method):
23         if method in self.methods:
24             return getattr(self, method)
25         else:
26             return None
27
28     # helper methods
29     methods = set(['add', 'subtract'])
30     def _get_msg(self, response):
31         return ' '.join(str(x) for x in [response.id, response.result or response.error])
32
33     def subtract(self, a, b):
34         return a-b
35
36     def add(self, a, b):
37         return a+b
38
39 root = JSON_RPC().customize(ExampleServer)
40 site = server.Site(root)
41
```

```
42
43 # 8007 is the port you want to run under. Choose something >1024
44 PORT = 8007
45 print('Listening on port %d...' % PORT)
46 reactor.listenTCP(PORT, site)
47 reactor.run()
```

To use this server (which is included as `jsonrpc.example_server`), start it and the client in this way:

```
% python -m jsonrpc.example_server &
% python -i -m jsonrpc.__main__ http://localhost:8007
```

```
>>> server.add(1,2)
3
>>> server.subtract(3,2)
1
>>> server.batch_call(dict(
...     add = ((3, 2), {}),
...     subtract = (((), {'a': 3, 'b': 2}))
... ))
[(5, None), (1, None)]
```

JSON-RPC Server

JSON-RPC Proxy

class `jsonrpc.proxy.JSONRPCProxy` (*host*, *path*='jsonrpc', *serviceName*=None, **args*, ***kwargs*)
 A class implementing a JSON-RPC Proxy.

Parameters

- **host** (*str*) – The HTTP server hosting the JSON-RPC server
- **path** (*str*) – The path where the JSON-RPC server can be found

There are two ways of instantiating this class: - `JSONRPCProxy.from_url(url)` – give the absolute url to the JSON-RPC server - `JSONRPC(host, path)` – break up the url into smaller parts

batch_call (*methods*)

call several methods at once, return a list of (result, error) pairs

Parameters **names** – a dictionary { method: (args, kwargs) }

Returns a list of pairs (result, error) where only one is not None

call (*method*, **args*, ***kwargs*)

call a JSON-RPC method

It's better to use `instance.<methodname>(*args, **kwargs)`, but this version might be useful occasionally

classmethod from_url (*url*, *ctxid*=None, *serviceName*=None)

Create a `JSONRPCProxy` from a URL

class `jsonrpc.proxy.ProxyEvents` (*proxy*)

An event handler for `JSONRPCProxy`

Allow a subclass to do its own initialization, gets any arguments leftover from `__init__`

IDGen = '0dd6a330692b02a2f8543e6ffbacc1c41d455e3c'

an instance of a class which defines a `__get__` method, used to generate a request id

get_params (*args*, *kwargs*)

allow a subclass to modify the method's arguments

e.g. if an authentication token is necessary, the subclass can automatically insert it into every call

proc_response (*data*)

allow a subclass to access the response data before it is returned to the user

Indices and tables

- `genindex`
- `modindex`
- `search`

j

`jsonrpc.proxy`, [7](#)

B

`batch_call()` (`jsonrpc.proxy.JSONRPCProxy` method), [7](#)

C

`call()` (`jsonrpc.proxy.JSONRPCProxy` method), [7](#)

F

`from_url()` (`jsonrpc.proxy.JSONRPCProxy` class method), [7](#)

G

`get_params()` (`jsonrpc.proxy.ProxyEvents` method), [7](#)

I

`IDGen` (`jsonrpc.proxy.ProxyEvents` attribute), [7](#)

J

`jsonrpc.proxy` (module), [7](#)

`JSONRPCProxy` (class in `jsonrpc.proxy`), [7](#)

P

`proc_response()` (`jsonrpc.proxy.ProxyEvents` method), [7](#)

`ProxyEvents` (class in `jsonrpc.proxy`), [7](#)