Class: MockClient

Extends: `undici.Client`

A mock client class that implements the same api as MockPool.

`new MockClient(origin, [options])`

Arguments:

- **origin** `string` - It should only include the **protocol, hostname, and port**.
- **options** `MockClientOptions` - It extends the `Client` options.

Returns: `MockClient`

Parameter: `MockClientOptions`

Extends: `ClientOptions`

- **agent** `Agent` - the agent to associate this MockClient with.

Example - Basic MockClient instantiation

We can use MockAgent to instantiate a MockClient ready to be used to intercept specified requests. It will not do anything until registered as the agent to use and any mock request are registered.

```
import { MockAgent } from 'undici'

// Connections must be set to 1 to return a MockClient instance
const mockAgent = new MockAgent({ connections: 1 })

const mockClient = mockAgent.get('http://localhost:3000')
```

Instance Methods

`MockClient.intercept(options)`

Implements: `MockPool.intercept(options)`

`MockClient.close()`

Implements: `MockPool.close()`

`MockClient.dispatch(options, handlers)`

Implements `Dispatcher.dispatch(options, handlers)`.

`MockClient.request(options[, callback])`

See `Dispatcher.request(options [, callback])`.

Example - MockClient request

```
import { MockAgent } from 'undici'

const mockAgent = new MockAgent({ connections: 1 })

const mockClient = mockAgent.get('http://localhost:3000')
mockClient.intercept({ path: '/foo' }).reply(200, 'foo')

const {
  statusCode,
  body
} = await mockClient.request({
  origin: 'http://localhost:3000',
  path: '/foo',
  method: 'GET'
})

console.log('response received', statusCode) // response received 200

for await (const data of body) {
  console.log('data', data.toString('utf8')) // data foo
}
```