Class: MockAgent

Extends: `undici.Dispatcher`

A mocked Agent class that implements the Agent API. It allows one to intercept HTTP requests made through undici and return mocked responses instead.

`new MockAgent([options])`

Arguments:

- **options** `MockAgentOptions` (optional) - It extends the `Agent` options.

Returns: `MockAgent`

Parameter: `MockAgentOptions`

Extends: `AgentOptions`

- **agent** `Agent` (optional) - Default: `new Agent([options])` - a custom agent encapsulated by the MockAgent.

Example - Basic MockAgent instantiation

This will instantiate the MockAgent. It will not do anything until registered as the agent to use with requests and mock interceptions are added.

```
import { MockAgent } from 'undici'

const mockAgent = new MockAgent()
```

Example - Basic MockAgent instantiation with custom agent

```
import { Agent, MockAgent } from 'undici'

const agent = new Agent()

const mockAgent = new MockAgent({ agent })
```

Instance Methods

`MockAgent.get(origin)`

This method creates and retrieves MockPool or MockClient instances which can then be used to intercept HTTP requests. If the number of connections on the mock agent is set to 1, a MockClient instance is returned. Otherwise a MockPool instance is returned.

For subsequent `MockAgent.get` calls on the same origin, the same mock instance will be returned.

Arguments:

- **origin** `string | RegExp | (value) => boolean` - a matcher for the pool origin to be retrieved from the MockAgent.

Matcher type	Condition to pass
`string`	Exact match against string
`RegExp`	Regex must pass
`Function`	Function must return true

Returns: `MockClient | MockPool`.

`MockAgentOptions`	Mock instance returned
`connections === 1`	`MockClient`
`connections > 1`	`MockPool`

Example - Basic Mocked Request

```js
import { MockAgent, setGlobalDispatcher, request } from 'undici'

const mockAgent = new MockAgent()
setGlobalDispatcher(mockAgent)

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

const { statusCode, body } = await request('http://localhost:3000/foo')

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

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

Example - Basic Mocked Request with local mock agent dispatcher

```js
import { MockAgent, request } from 'undici'

const mockAgent = new MockAgent()

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

const {
  statusCode,
  body
} = await request('http://localhost:3000/foo', { dispatcher: mockAgent })

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

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

Example - Basic Mocked Request with local mock pool dispatcher

```js
import { MockAgent, request } from 'undici'

const mockAgent = new MockAgent()

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

const {
  statusCode,
  body
} = await request('http://localhost:3000/foo', { dispatcher: mockPool })

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

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

Example - Basic Mocked Request with local mock client dispatcher

```js
import { MockAgent, request } 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 request('http://localhost:3000/foo', { dispatcher: mockClient })

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

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

Example - Basic Mocked requests with multiple intercepts

```js
import { MockAgent, setGlobalDispatcher, request } from 'undici'
```

```
const mockAgent = new MockAgent()
setGlobalDispatcher(mockAgent)

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

const result1 = await request('http://localhost:3000/foo')

console.log('response received', result1.statusCode) // response received 200

for await (const data of result1.body) {
  console.log('data', data.toString('utf8')) // data foo
}

const result2 = await request('http://localhost:3000/hello')

console.log('response received', result2.statusCode) // response received 200

for await (const data of result2.body) {
  console.log('data', data.toString('utf8')) // data hello
}
```

Example - Mocked request with query body, headers and trailers

```
import { MockAgent, setGlobalDispatcher, request } from 'undici'

const mockAgent = new MockAgent()
setGlobalDispatcher(mockAgent)

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

mockPool.intercept({
  path: '/foo?hello=there&see=ya',
  method: 'POST',
  body: 'form1=data1&form2=data2'
}).reply(200, { foo: 'bar' }, {
  headers: { 'content-type': 'application/json' },
  trailers: { 'Content-MD5': 'test' }
})

const {
  statusCode,
  headers,
  trailers,
  body
} = await request('http://localhost:3000/foo?hello=there&see=ya', {
  method: 'POST',
  body: 'form1=data1&form2=data2'
})
```

```js
console.log('response received', statusCode) // response received 200
console.log('headers', headers) // { 'content-type': 'application/json' }

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

console.log('trailers', trailers) // { 'content-md5': 'test' }
```

Example - Mocked request with origin regex

```js
import { MockAgent, setGlobalDispatcher, request } from 'undici'

const mockAgent = new MockAgent()
setGlobalDispatcher(mockAgent)

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

const {
  statusCode,
  body
} = await request('http://localhost:3000/foo')

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

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

Example - Mocked request with origin function

```js
import { MockAgent, setGlobalDispatcher, request } from 'undici'

const mockAgent = new MockAgent()
setGlobalDispatcher(mockAgent)

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

const {
  statusCode,
  body
} = await request('http://localhost:3000/foo')

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

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

`MockAgent.close()`

Closes the mock agent and waits for registered mock pools and clients to also close before resolving.

Returns: `Promise<void>`

Example - clean up after tests are complete

```
import { MockAgent, setGlobalDispatcher } from 'undici'

const mockAgent = new MockAgent()
setGlobalDispatcher(mockAgent)

await mockAgent.close()
```

`MockAgent.dispatch(options, handlers)`

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

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

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

Example - MockAgent request

```
import { MockAgent } from 'undici'

const mockAgent = new MockAgent()

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

const {
  statusCode,
  body
} = await mockAgent.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
}
```

`MockAgent.deactivate()`

This method disables mocking in MockAgent.

Returns: `void`

Example - Deactivate Mocking

```js
import { MockAgent, setGlobalDispatcher } from 'undici'

const mockAgent = new MockAgent()
setGlobalDispatcher(mockAgent)

mockAgent.deactivate()
```

`MockAgent.activate()`

This method enables mocking in a MockAgent instance. When instantiated, a MockAgent is automatically activated. Therefore, this method is only effective after `MockAgent.deactivate` has been called.

Returns: `void`

Example - Activate Mocking

```js
import { MockAgent, setGlobalDispatcher } from 'undici'

const mockAgent = new MockAgent()
setGlobalDispatcher(mockAgent)

mockAgent.deactivate()
// No mocking will occur

// Later
mockAgent.activate()
```

`MockAgent.enableNetConnect([host])`

When requests are not matched in a MockAgent intercept, a real HTTP request is attempted. We can control this further through the use of `enableNetConnect`. This is achieved by defining host matchers so only matching requests will be attempted.

When using a string, it should only include the **hostname and optionally, the port**. In addition, calling this method multiple times with a string will allow all HTTP requests that match these values.

Arguments:

- **host** `string | RegExp | (value) => boolean` - (optional)

Returns: `void`

Example - Allow all non-matching urls to be dispatched in a real HTTP request

```js
import { MockAgent, setGlobalDispatcher, request } from 'undici'

const mockAgent = new MockAgent()
```

```
setGlobalDispatcher(mockAgent)

mockAgent.enableNetConnect()

await request('http://example.com')
// A real request is made
```

Example - Allow requests matching a host string to make real requests

```
import { MockAgent, setGlobalDispatcher, request } from 'undici'

const mockAgent = new MockAgent()
setGlobalDispatcher(mockAgent)

mockAgent.enableNetConnect('example-1.com')
mockAgent.enableNetConnect('example-2.com:8080')

await request('http://example-1.com')
// A real request is made

await request('http://example-2.com:8080')
// A real request is made

await request('http://example-3.com')
// Will throw
```

Example - Allow requests matching a host regex to make real requests

```
import { MockAgent, setGlobalDispatcher, request } from 'undici'

const mockAgent = new MockAgent()
setGlobalDispatcher(mockAgent)

mockAgent.enableNetConnect(new RegExp('example.com'))

await request('http://example.com')
// A real request is made
```

Example - Allow requests matching a host function to make real requests

```
import { MockAgent, setGlobalDispatcher, request } from 'undici'

const mockAgent = new MockAgent()
setGlobalDispatcher(mockAgent)

mockAgent.enableNetConnect((value) => value === 'example.com')

await request('http://example.com')
// A real request is made
```

`MockAgent.disableNetConnect()`

This method causes all requests to throw when requests are not matched in a MockAgent intercept.

Returns: `void`

Example - Disable all non-matching requests by throwing an error for each

```js
import { MockAgent, request } from 'undici'

const mockAgent = new MockAgent()

mockAgent.disableNetConnect()

await request('http://example.com')
// Will throw
```