

socks examples

Example for SOCKS 'connect' command

The connect command is the most common use-case for a SOCKS proxy. This establishes a direct connection to a destination host through a proxy server. The destination host only has knowledge of the proxy server connecting to it and does not know about the origin client (you).

Origin Client (you) <-> Proxy Server <-> Destination Server

In this example, we are connecting to a web server on port 80, and sending a very basic HTTP request to receive a response. It's worth noting that there are many socks-http-agents that can be used with the node http module (and libraries such as request.js) to make this easier. This HTTP request is used as a simple example.

The 'connect' command can be used via the SocksClient.createConnection() factory function as well as by creating a SocksClient instance and using event handlers.

Using createConnection with async/await

Since SocksClient.createConnection returns a Promise, we can easily use async/await for flow control.

```
import { SocksClient, SocksClientOptions } from 'socks';

const options: SocksClientOptions = {
  proxy: {
    host: '104.131.124.203',
    port: 1081,
    type: 5
  },

  destination: {
    host: 'ip-api.com', // host names are supported with SOCKS v4a and SOCKS v5.
    port: 80
  },

  command: 'connect'
};

async function start() {
  try {
    const info = await SocksClient.createConnection(options);

    console.log(info.socket);
    // <Socket ...> (this is a raw net.Socket that is established to the
    destination host through the given proxy servers)

    info.socket.write('GET /json HTTP/1.1\nHost: ip-api.com\n\n');
    info.socket.on('data', (data) => {
      console.log(data.toString()); // ip-api.com sees that the last proxy
      (104.131.124.203) is connected to it and not the origin client (you).
      /*
      HTTP/1.1 200 OK
```

```

Access-Control-Allow-Origin: *
Content-Type: application/json; charset=utf-8
Date: Sun, 24 Dec 2017 03:47:51 GMT
Content-Length: 300

{
  "as": "AS14061 Digital Ocean, Inc.",
  "city": "Clifton",
  "country": "United States",
  "countryCode": "US",
  "isp": "Digital Ocean",
  "lat": 40.8326,
  "lon": -74.1307,
  "org": "Digital Ocean",
  "query": "104.131.124.203",
  "region": "NJ",
  "regionName": "New Jersey",
  "status": "success",
  "timezone": "America/New_York",
  "zip": "07014"
}
*/
});
} catch (err) {
  // Handle errors
}
}

start();

```

Using createConnection with Promises

```

import { SocksClient, SocksClientOptions } from 'socks';

const options: SocksClientOptions = {
  proxy: {
    ipaddress: '104.131.124.203',
    port: 1081,
    type: 5
  },

  destination: {
    host: 'ip-api.com', // host names are supported with SOCKS v4a and SOCKS v5.
    port: 80
  },

  command: 'connect'
};

SocksClient.createConnection(options)
  .then(info => {

```

```

    console.log(info.socket);
    // <Socket ...> (this is a raw net.Socket that is established to the destination
    host through the given proxy servers)

    info.socket.write('GET /json HTTP/1.1\nHost: ip-api.com\n\n');
    info.socket.on('data', (data) => {
        console.log(data.toString()); // ip-api.com sees that the last proxy
        (104.131.124.203) is connected to it and not the origin client (you).
        /*
        HTTP/1.1 200 OK
        Access-Control-Allow-Origin: *
        Content-Type: application/json; charset=utf-8
        Date: Sun, 24 Dec 2017 03:47:51 GMT
        Content-Length: 300

        {
          "as": "AS14061 Digital Ocean, Inc.",
          "city": "Clifton",
          "country": "United States",
          "countryCode": "US",
          "isp": "Digital Ocean",
          "lat": 40.8326,
          "lon": -74.1307,
          "org": "Digital Ocean",
          "query": "104.131.124.203",
          "region": "NJ",
          "regionName": "New Jersey",
          "status": "success",
          "timezone": "America/New_York",
          "zip": "07014"
        }
        */
    });
  })
  .catch(err => {
    // handle errors
  });
}

```

Using createConnection with callbacks

SocksClient.createConnection() optionally accepts a callback function as a second parameter.

Note: If a callback function is provided, a Promise is still returned from the function, but the promise will always resolve regardless of if there was an error. (tldr: Do not mix callbacks and Promises).

```

import { SocksClient, SocksClientOptions } from 'socks';

const options: SocksClientOptions = {
  proxy: {
    ipAddress: '104.131.124.203',
    port: 1081,
    type: 5
  }
}

```

```

    },

    destination: {
      host: 'ip-api.com', // host names are supported with SOCKS v4a and SOCKS v5.
      port: 80
    },

    command: 'connect'
  };

  SocksClient.createConnection(options, (err, info) => {
    if (err) {
      // handle errors
    } else {
      console.log(info.socket);
      // <Socket ...> (this is a raw net.Socket that is established to the
      destination host through the given proxy servers)

      info.socket.write('GET /json HTTP/1.1\nHost: ip-api.com\n\n');
      info.socket.on('data', (data) => {
        console.log(data.toString()); // ip-api.com sees that the last proxy
        (104.131.124.203) is connected to it and not the origin client (you).
        /*
        HTTP/1.1 200 OK
        Access-Control-Allow-Origin: *
        Content-Type: application/json; charset=utf-8
        Date: Sun, 24 Dec 2017 03:47:51 GMT
        Content-Length: 300

        {
          "as": "AS14061 Digital Ocean, Inc.",
          "city": "Clifton",
          "country": "United States",
          "countryCode": "US",
          "isp": "Digital Ocean",
          "lat": 40.8326,
          "lon": -74.1307,
          "org": "Digital Ocean",
          "query": "104.131.124.203",
          "region": "NJ",
          "regionName": "New Jersey",
          "status": "success",
          "timezone": "America/New_York",
          "zip": "07014"
        }
        */
      });
    }
  })
}

```

Using event handlers

SocksClient also supports instance creation of a SocksClient. This allows for event based flow control.

```
import { SocksClient, SocksClientOptions } from 'socks';

const options: SocksClientOptions = {
  proxy: {
    ipaddress: '104.131.124.203',
    port: 1081,
    type: 5
  },

  destination: {
    host: 'ip-api.com', // host names are supported with SOCKS v4a and SOCKS v5.
    port: 80
  },

  command: 'connect'
};

const client = new SocksClient(options);

client.on('established', (info) => {
  console.log(info.socket);
  // <Socket ...> (this is a raw net.Socket that is established to the destination
  // host through the given proxy servers)

  info.socket.write('GET /json HTTP/1.1\nHost: ip-api.com\n\n');
  info.socket.on('data', (data) => {
    console.log(data.toString()); // ip-api.com sees that the last proxy
    // (104.131.124.203) is connected to it and not the origin client (you).
    /*
    HTTP/1.1 200 OK
    Access-Control-Allow-Origin: *
    Content-Type: application/json; charset=utf-8
    Date: Sun, 24 Dec 2017 03:47:51 GMT
    Content-Length: 300

    {
      "as": "AS14061 Digital Ocean, Inc.",
      "city": "Clifton",
      "country": "United States",
      "countryCode": "US",
      "isp": "Digital Ocean",
      "lat": 40.8326,
      "lon": -74.1307,
      "org": "Digital Ocean",
      "query": "104.131.124.203",
      "region": "NJ",
      "regionName": "New Jersey",
      "status": "success",
      "timezone": "America/New_York",
    }
    */
  });
});
```

```
        "zip": "07014"
    }
    */
    });
});

// Failed to establish proxy connection to destination.
client.on('error', () => {
    // Handle errors
});

// Start connection
client.connect();
```