

Tutorial 3: Express

[Express](#) is a popular web application framework for node. The E2E Transaction Logger provides a middleware for express. This tutorial shows how to use the E2E Transaction Logger with express to achieve the same result as in tutorial 2.

Creating the Server

Create the server using express. The DB and SAP functions are the same as in the tutorial 2 project.

```
var express = require('express');
var fs = require('fs');

function sapGetCustomer(id, cb){
    ...
}

function dbGetCustomerDetail(customer, cb){
    ...
}

var app = express();
app.get('/hello',function(req, res){
    fs.readFile(__dirname + '/Hello.html',function(err, data){
        res.writeHead(200, {'Content-Type': 'text/html'});
        res.end(data);
    });
});
app.get('/goodbye', function(req, res){
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Good Bye\n');
});
app.get('/do-something', function(req, res){
    sapGetCustomer(1, function(err, customer){
        if(err){
            res.writeHead(500, {'Content-Type': 'application/json'});
            res.end(JSON.stringify(err));
            return;
        }
        dbGetCustomerDetail(customer, function(err, customerWithDetail){
            if(err){
                res.writeHead(200, {'Content-Type': 'application/json'});
                res.end(JSON.stringify(customer));
                return;
            }
            res.writeHead(200, {'Content-Type': 'application/json'});
            res.end(JSON.stringify(customerWithDetail));
        });
    });
});
app.listen(1337, function() {
    console.log('Server running at http://127.0.0.1:1337/');
});
```

Now the server is doing the same as in tutorial 2 except for logging.

Adding the Logging

Now, add the logging.

Transactions

To log all transactions made by an express application, you can simply use the E2E Transaction Logger middleware.

On this Page:

- [Creating the Server](#)
- [Adding the Logging](#)
 - [Transactions](#)
 - [IOs](#)
- [Dashboards](#)

Related Pages:

- [Documentation of the E2E Transaction Logger](#)
- [Tutorial 1: Logging the Transactions of a Simple HTTP Server](#)
- [Tutorial 2: Backend Access](#)
- [Tutorial 3: Express](#)
- [Scheer E2E Dashboards](#)

[Download Example Project](#)

```
var e2eLogger = require('e2e-transaction-logger');

var app = express();
app.use(e2eLogger.transactionLoggerMiddleware());
```

Then you will get logs like these:

```
2014-04-30 13:01:39 +0200 9cf8bc91-0b82-4ce9-a592-423bd774bdad
1 GET /hello 0 OK INTERFACE SERVICE_ENTER
2014-04-30 13:01:39 +0200 9cf8bc91-0b82-4ce9-a592-423bd774bdad
1 GET /hello 6 OK INTERFACE SERVICE_EXIT
2014-04-30 13:01:55 +0200 1cfd87b5-aa95-41ea-b15e-560a3290b5a1
2 GET /do-something 0 OK INTERFACE SERVICE_ENTER
2014-04-30 13:01:57 +0200 1cfd87b5-aa95-41ea-b15e-560a3290b5a1
2 GET /do-something 1856 OK INTERFACE SERVICE_EXIT
2014-04-30 13:02:03 +0200 ac650a93-facc-4e5a-846c-6aae9201fd2a
3 GET /goodbye 0 OK INTERFACE SERVICE_ENTER
2014-04-30 13:02:03 +0200 ac650a93-facc-4e5a-846c-6aae9201fd2a
3 GET /goodbye 1 OK INTERFACE SERVICE_EXIT
```

The state of the transaction is defined by the status code of the response: failed if ≥ 400 .

IOs

The transaction logger middleware adds the **trx** attribute in the request object. You can use it to log the IO calls.

```
app.get('/hello',function(req, res){
  var io = req.trx.startIO('Read','FILE','Hello.html');
  fs.readFile(__dirname + '/Hello.html',function(err, data){
    io.end('OK');
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.end(data);
  });
});
```

Do the same in the '/do-something' request and you will see the IO logs:

```

2014-04-30 13:09:05 +0200 7048a612-dd8a-4677-aa32-49d0a5171208
1 GET /hello 0 OK INTERFACE SERVICE_ENTER
2014-04-30 13:09:05 +0200 7048a612-dd8a-4677-aa32-49d0a5171208
1 Read 2 OK FILE IO_ENTER Hello.html
2014-04-30 13:09:05 +0200 7048a612-dd8a-4677-aa32-49d0a5171208
1 Read 4 OK FILE IO_EXIT Hello.html
2014-04-30 13:09:05 +0200 7048a612-dd8a-4677-aa32-49d0a5171208
1 GET /hello 6 OK INTERFACE SERVICE_EXIT
2014-04-30 13:09:08 +0200 bdc2970f-6045-4c9e-b141-0f53cc707b94
2 GET /do-something 0 OK INTERFACE SERVICE_ENTER
2014-04-30 13:09:08 +0200 bdc2970f-6045-4c9e-b141-0f53cc707b94
2 GetCustomer 1 OK SAP IO_ENTER http://sap.e2e.ch:
3000/
2014-04-30 13:09:10 +0200 bdc2970f-6045-4c9e-b141-0f53cc707b94
2 GetCustomer 2386 OK SAP IO_EXIT http://sap.e2e.ch:
3000/
2014-04-30 13:09:10 +0200 bdc2970f-6045-4c9e-b141-0f53cc707b94
2 GetCustomerDetail 2386 OK DB IO_ENTER localhost
/mongodb/
2014-04-30 13:09:11 +0200 bdc2970f-6045-4c9e-b141-0f53cc707b94
2 GetCustomerDetail 2855 ERROR DB IO_EXIT localhost
/mongodb/
2014-04-30 13:09:11 +0200 bdc2970f-6045-4c9e-b141-0f53cc707b94
2 GET /do-something 2857 OK INTERFACE SERVICE_EXIT
2014-04-30 13:09:20 +0200 b71b24a9-cc37-473d-88aa-fb840ab159bb
3 GET /goodbye 0 OK INTERFACE SERVICE_ENTER
2014-04-30 13:09:20 +0200 b71b24a9-cc37-473d-88aa-fb840ab159bb
3 GET /goodbye 1 OK INTERFACE SERVICE_EXIT

```

You can download the complete [example project of tutorial 3](#).

Dashboards

These log files are fully compatible with the [E2E Dashboards](#) services, so they can be loaded into the database and used with the dashboards to analyze the performance of your services.

Find below some examples of the views you would get with this enhanced HTTP server.

Figure: Table View of the E2E Service Dashboard

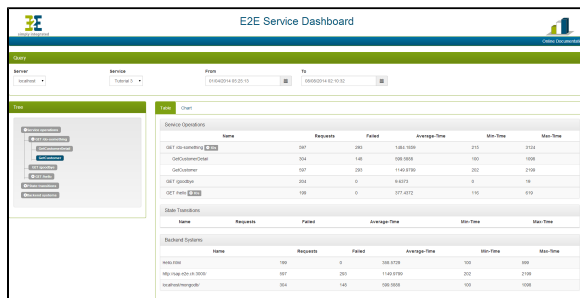


Figure: Graphical View of the E2E Service Dashboard

