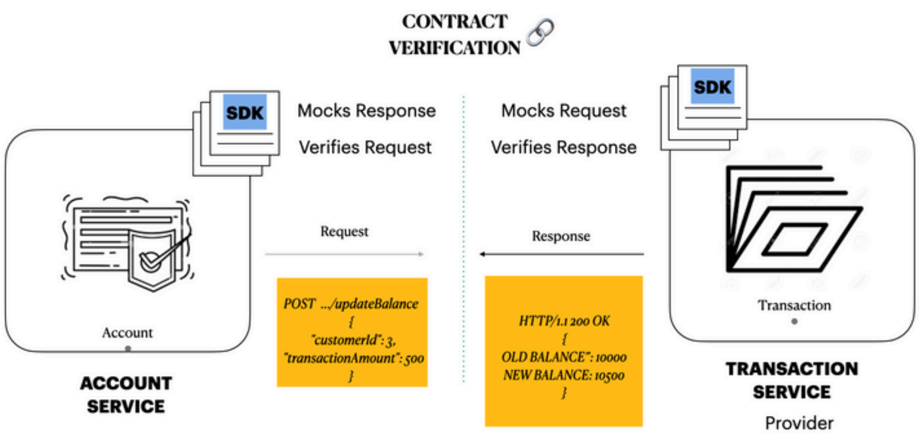
HYPERTEST

A Quick Guide on HyperTest's Smart Mocking Approach

Service to service interactions called contracts are automatically built by HyperTest by monitoring actual interactions.

Mocking External Services: Downstream calls, 3rd party APIs



```
POST .../updateBalance
{
"customerId": 3,
"transactionAmount": 500
}
```

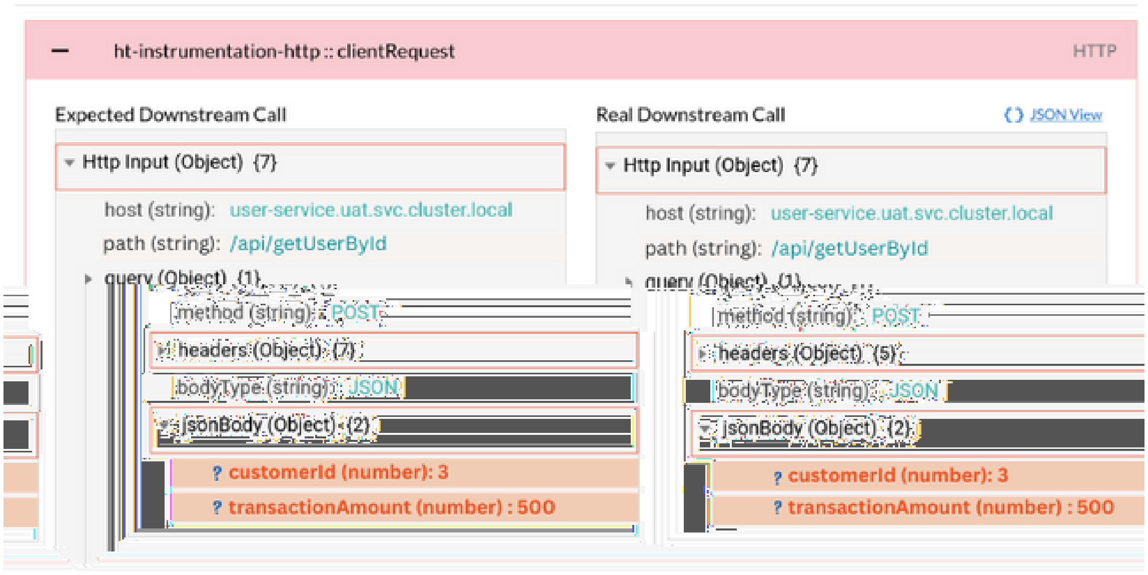
```
HTTP/1.1 200 OK
{
OLD BALANCE": 10000
NEW BALANCE: 10500
}
```

- The HyperTest SDK is set-up on the **AccountService** and **TransactionService**
- It monitors all the incoming and outgoing calls for both **AccountService** and **TransactionService**.
- In this case, the request - response pair i.e. the contract between **AccountService** - **TransactionService** is captured by HyperTest. This contract is used as to mock the **TransactionService** when testing **AccountService** and vice versa.

Now when the developer wants to test his **AccountService** class in Accounts Service, HyperTest CLI builds **AccountService** app locally or at the CI server and calls this request, and supplies the mocked response from **TransactionService**.

HyperTest SDK that tests TransactionService and AccountService separately, automatically asserts 2 things:

- The **TransactionAPI** was called with the correct parameters by the **AccountService**



ht-instrumentation-http :: clientRequest HTTP

Expected Downstream Call

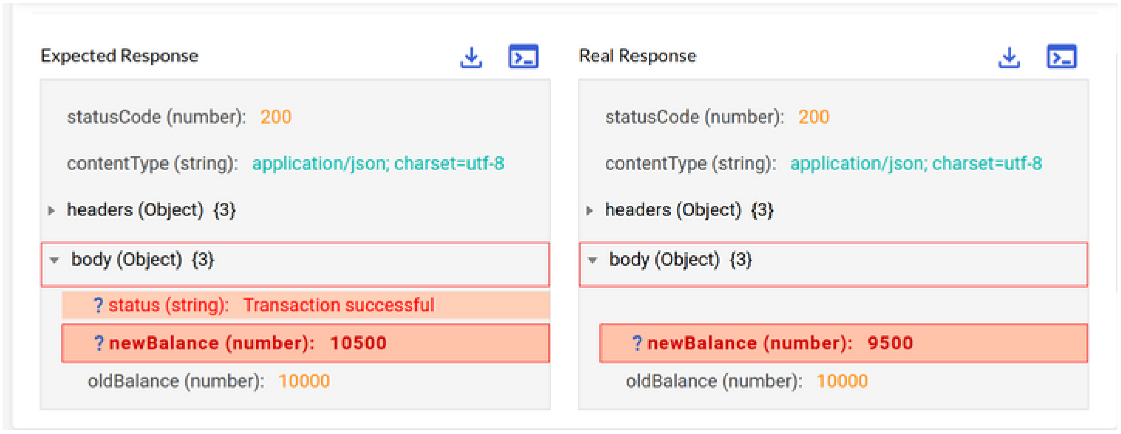
Http Input (Object) {7}
host (string): user-service.uat.svc.cluster.local
path (string): /api/getUserById
query (Object) {1}
method (string): POST
headers (Object) {7}
bodyType (string): JSON
jsonBody (Object) {2}
? customerId (number): 3
? transactionAmount (number) : 500

Real Downstream Call

JSON View

Http Input (Object) {7}
host (string): user-service.uat.svc.cluster.local
path (string): /api/getUserById
query (Object) {1}
method (string): POST
headers (Object) {5}
bodyType (string): JSON
jsonBody (Object) {2}
? customerId (number): 3
? transactionAmount (number) : 500

- Response of the **TransactionService**, i.e. new balance is same as 10500. If not it reports error like this.



Expected Response

statusCode (number): 200
contentType (string): application/json; charset=utf-8
headers (Object) {3}
body (Object) {3}
? status (string): Transaction successful
? newBalance (number): 10500
oldBalance (number): 10000

Real Response

statusCode (number): 200
contentType (string): application/json; charset=utf-8
headers (Object) {3}
body (Object) {3}
? newBalance (number): 9500
oldBalance (number): 10000

HyperTest mocks upstream and downstream calls automatically, somethings that 20 lines of mockito code will be able to do. Best thing, it refreshes these mocks as the behavior of the AccountService (requests) or TransactionService (response) change.

HYPERTEST

95 Third Street
2nd Floor, 94103 San Francisco,
California, USA

Follow us on