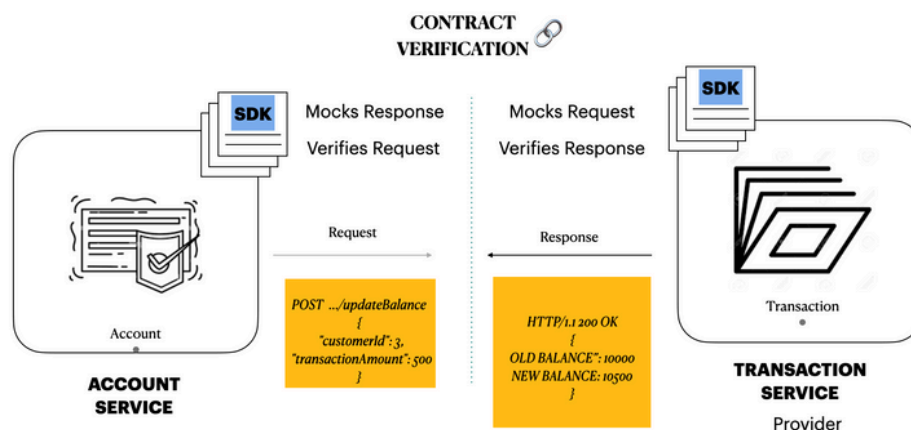


A Quick Guide on HyperTest's Smart Mocking Approach



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

Mocking External Services: Downstream calls, 3rd party APIs



- 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**

The screenshot displays a comparison between the 'Expected Downstream Call' and the 'Real Downstream Call' for an HTTP client request. Both panels show identical details: host (user-service.uat.svc.cluster.local), path (/api/getUserById), query (empty), method (POST), headers (5), body type (JSON), and a JSON body with customerid (3) and transactionAmount (500).

Expected Downstream Call	Real Downstream Call
Host (string): user-service.uat.svc.cluster.local	Host (string): user-service.uat.svc.cluster.local
Path (string): /api/getUserById	Path (string): /api/getUserById
Query (Object) {1}	Query (Object) {1}
Method (string): POST	Method (string): POST
Headers (Object) {7}	Headers (Object) {5}
Body Type (string): JSON	Body Type (string): JSON
JSON Body (Object) {2}	JSON Body (Object) {2}
customerid (number): 3	customerid (number): 3
transactionAmount (number): 500	transactionAmount (number): 500

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

The screenshot compares the 'Expected Response' and the 'Real Response' from a TransactionService call. Both show a 200 status code and application/json content type. The expected response has a newBalance of 10500, while the real response has a newBalance of 9500, indicating a discrepancy.

Expected Response	Real Response
Status Code (number): 200	Status Code (number): 200
Content Type (string): application/json; charset=utf-8	Content Type (string): application/json; charset=utf-8
Headers (Object) {3}	Headers (Object) {3}
Body (Object) {3}	Body (Object) {3}
status (string): Transaction successful	status (string): Transaction successful
newBalance (number): 10500	newBalance (number): 9500
oldBalance (number): 10000	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.



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

Follow us on

