**Work Related**

URL for CC get API –

**POST**

URL : <http://127.0.0.1:3001/fis-ft/v1/fundstransferservices>

**#Header params:**

Authorization : Bearer Token

originating-reporting-level-cd : "91000"

originating-channel-name : CC

traceabilityid : a372bfaa-913d-4ff8-b20b-bc9ad51bed41

content-Type : application/json

originating-branch-num :8888

**POST BODY:**

{

"fundsTransferCaptureInfo": {

"requestFundsSource": {

"requestSourceDepositTransactionAgreement": {

"bankNum": "91001",

"branchNum": "123456789",

"accountNum": "4318075044"

}

},

"requestFundsDestination": {

"requestDestinationCreditCardAgreement": {

"cardNum": "00006749663191"

}

},

"fundsSourceCccyamt": {

"amount": 120.00,

"currencyCd":"USD"

},

"sourceAccountApplicationId": "IM",

"sourceTransactionCd": "0572",

"fisExtension": {

"sourceBankControl1Num": "94",

"sourceBankControl2Num": "004",

"sourceBankControl3Num": "0000",

"sourceBankControl4Num": "0000",

"destinationBankControl1Num": "94",

"destinationBankControl2Num": "004",

"destinationBankControl3Num": "0000",

"destinationBankControl4Num": "0000"

}

},

"messageDestinationTxt": "ETRANSFER",

"messageSourceTxt": "ETRANSFER",

"reasonCd": "Payment",

"paymentInfo": {

"requestTypeDesc": "Payment",

"actionDesc": "Request Authorization"

},

"actionTypeName": "Immediate"

}

This API is to make Payment for a credit Card.

For a particular Credit card number passed in request body.

Payment will be done from a source account.

So that account no. and amount are also passed in request body.

Logic:

This API is having internal call to another third party API ….creditcardagreements

First of all with following parameters of request param there is a POST call using request module

actionDesc, requestTypeDesc, amount, currencyCd

URL : **POST** : /kdfjs/CreditCardNumber

This operation is used to create an authorization event at TSYS - third party. This will be used to make a payment on the account. On success, this method will return an HTTP status code of 201.

If it error 🡪 display error

If that credit card no. is present in the database then authorization is successful

If It is success 🡪 go to next flow

requestTransformer 🡪 **FISexecution**() 🡪

If source Account is having less amount than the bill amount , transaction can’t be done .

So error 🡪 Insufficient Amount

GET call by request module

URL : **GET** : /kdfjs/CreditCardNumber

If Error in transaction then call delete

URL : **DELETE**: /kdfjs/CreditCardNumber - This is to delete an authorization event created by POST at TSYS - third party(as this transaction cannot happen )

request(

{

headers: {

'content-type': 'application/json',

},

url: baseUrl,

method: "**POST**",

body: inputBody

},

function (error, result, response) {

var data = JSON.parse(response);

var responseDetails = {};

responseDetails.ResponseCd = data.authorizationResponseCd;

if(responseDetails.ResponseCd =="Approved"){

responseDetails.approvalCd = data.approvalCd;

cb(null,responseDetails.approvalCd);

}

else {

console.log('Authorization Error From TYSY ',error);

cb(error);

}

}

);

Project In general

SDLC = **Systems\_development\_life\_cycle**

Get the requirements in the form of

SDS (system design specification),

Mapping sheet - ,Schema

Sequence diagram - equivalent to high level design doc - class flow

Technical design - equivalent to low level design doc - methods, backend list, error conditions

Analysis:

go through all documents , compare all , if any mismatch found , log a query in query tracker,

analysis of dependencies (node modules), estimated time , software’s needed

design :

make a design , how we are going to code , input , output and some pseudo codes

coding - provide a request in postman

apply proper url

get the request using body parser

use async.waterfall as we want step by step execution

map the request parameters to JavaScript object

validate the request

apply if conditions according to the requirements

then the flow comes to request mapper - convert the JSON object to HM elements

to know particular HM element we use cobol copy book

here comes back end connectivity - FIS - Fidelity Information Services

using client specific fis connector - a node module

from this connector FIS connectivity is achieved

after FIS execution get the response in HM elements

Now map the response to JavaScript object in response mapper

do the schema validation

Unit testing - mocha Chai and gulp, generating test report which needs code coverage > 80%

deploy a code on VM for SIT team ,

deliver a code along with VM endpoint and SIT entry criterion docs - UT Doc, Code Coverage, Code Review doc, Error code doc .

and upload the code to git hub

Integration testing - SIT

Client side testing - before production

Production - give production support, monitoring

HM element **HM Dag** **CICSREC Object**

HMIMTRST-ACCT-NO **HMHMTRST** **CICSREC**

.cbl FILE

00100\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

00200\* DATAGROUP: HMAMAFTR GENERATED: 01/26/06, 16:12:52 \*

00300\* HM AM/ALS AFFILIATE ADD STND TRAN \*

00400\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

00500 01 HMAMAFTR-STD-TRAN. HMAMAFTR

00600 03 HMAMAFTR-AF-CTLS PIC X(12). HMAMAF20

00700 03 HMAMAFTR-AF-ACCT PIC X(19). HMAMAF21

00800 03 HMAMAFTR-AF-SYS PIC XX. HMAMAF04

00900 03 HMAMAFTR-AF-AUTO-ABA PIC S9(9) COMP-3. HMAMAF19

01000 03 HMAMAFTR-BD-AUTO-DAYS PIC S9(3) COMP-3. HMAMBD11

**Async.waterfall syntax**

**Async.waterfall ( [ f1,f2,f3 ] ,cb );**

example

async.waterfall ( [

function(next) {

requestValidator.requestValidation(params, function(err, result1) {

if (err) {

next(err, null);

} else {

next(null, result1);

}

});

},

function(result1, next) {

requestMapper.requestTransformer(params, function(err, result2) {

if (err) {

return next(err, null);

} else {

next(null, result2);

}

});

},

function(result2, next) {

fisPagination.execute(params, result2, function(err, result3) {

if (err) {

return next(err, null);

} else {

next(null, result3);

}

});

}

], function(err, result) {

if (err) {

logger.error('Error while executing async waterfall,', err, metadata);

callback(err);

} else {

callback(null, result);

}

});

**TDbank**  project is having 28 webservices ..in java now moving in onto node..

Make separate API of Accout IM , ST test individually and then integrate in As a whole Account , then test and deploy

Also have developed **independent APIs**

BOCP – Back end oracle

CC payment FundsTrasnfer – call ThirdParty API by using request module.

**Sample package.JSON**

{

"name": "api-ust-parties-bocp",

"version": "1.0.0",

"main": "server/server.js",

"scripts": {

"coveralls": "mocha --require blanket --reporter mocha-lcov-reporter | ./node\_modules/coveralls/bin/coveralls.js",

"test": "mocha -R landing test/\*\*/\*-test.js",

"test-cov": "mocha --require blanket -R html-cov > cov/coverage.html"

},

"**dependencies**": {

"sync-request": "2.0.1",

"async": "1.1.0",

"compression": "1.6.2",

"errorhandler": "1.4.3",

"forever-agent": "0.6.1",

"http-status-codes": "1.0.5",

"is-my-json-valid": "2.15.0",

"is-property": "1.0.2",

"lodash": "3.9.3",

"loopback": "2.22.1",

"loopback-boot": "2.8.1",

"loopback-datasource-juggler": "2.38.0",

"request": "2.76.0",

"td-request-context": "0.0.1",

"serve-favicon": "2.0.1",

"cookie-parser": "1.4.1",

"ssl-root-cas": "1.1.10",

"applicationcontext": "0.1.2",

"hashcode": "1.0.3",

"ip": "1.1.2",

"multer": "1.1.0",

"multer-autoreap": "0.1.2",

"xss-filters": "1.2.7",

"sanitizer": "0.1.2",

"node-cache": "3.2.1",

"oracledb" : "1.6.0",

"express-req-metrics": "1.1.1",

"moment": "2.9.0",

"validator": "4.5.0",

"gulp": "3.9.0",

"base64url" :"1.0.5",

"stdout-stream": "1.4.0",

"debug": "2.2.0"

},

"**devDependencies**": {

"chai": "3.0.0",

"chai-spies": "0.6.0",

"gulp": "3.9.0",

"mocha": "2.2.5",

"supertest": "1.1.0",

"td-api-common-gulp": "1.7.1-RC",

"loopback-explorer": "1.7.2"

},

"**repository**": {

"type": "git",

"url": "https://code.td.com/projects/USTSAPI/repos/api-ust-parties-bocp",

"commit": {

"tree": "07529223331ad8c03a9822105377b998168c7cf2",

"parent": "9cdaffda5c4acb1571059f03baaec3df762e0e0b",

"author": "Shashikanth Doke <Shashikant.Doke@td.com>",

"committer": "Shashikanth Doke <Shashikant.Doke@td.com> 1441228270 -0400",

"commit": "eff2c5c600190bd93867b467168444646fe5bb85",

"date": "Wed Sep 2 17:11:10 2015 -0400",

"message": "updated to framework 1.6.1M\tcommon/adapters/lsov10/lsov10.jsM\tcommon/helpers/lsoHelper.jsM\tconfig/schemaConfig.jsonA\tconfig/serverConfig.jsonM\tpackage.jsonM\tserver/config.jsonM\tserver/server.jsM\ttest/loanStatusIntegration-test.jsM\ttest/lsoHelper-test.js"

}

},

"description": "Project code testing API Framework foundation",

"releaseVersion": "1.7.1-RC",

"readme": "README.md",

"engines": {

"node": "0.10.39",

"npm": "1.4.28"

},

"license": "LicenseRef-LICENSE",

"files": [

"package.json",

"README.md",

"client/\*\*/\*",

"common/\*\*/\*",

"lib/\*\*/\*",

"log",

"node\_modules/\*\*/\*",

"server/\*\*/\*"

],

"**bundleDependencies**": [

"applicationcontext",

"sync-request",

"async",

"compression",

"cookie-parser",

"errorhandler",

"forever-agent",

"hashcode",

"http-status-codes",

"ip",

"is-my-json-valid",

"is-property",

"lodash",

"loopback",

"loopback-boot",

"loopback-datasource-juggler",

"multer",

"multer-autoreap",

"request",

"sanitizer",

"serve-favicon",

"ssl-root-cas",

"moment",

"validator",

"td-request-context",

"stdout-stream",

"debug",

"xss-filters",

"base64url"

]

}