EMI-SOLUTION

**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Revision**# | **Date** | **Author** | **Approver** | **Revision Details** |
| 1.0 | 13/04/2018 | Karim |  |  |
|  |  |  |  |  |

Table of Contents

1.Introduction……………………………………………4

2.Scope…………………………………………………..4

3.Design System overview……………………………....4

4.EMI-Solution API….…………………………..………5

4.1.Access Token Authorization Server…...…………….5

4.2. Refresh token from Authorization Server…...……...6

5.EMI Solution Data Model……………………………..7

**1.Introduction**

This document defines the design of EMI Solution,Post successful approved transaction cashier will Login to innovit Tab (Android) Application and enter all the Card Holder,EMI Details once submitting all the data will stored in the innoviti

Server.

**2.Scope**

EMI Solution will architecture will having the following scope

* Client authentication with the token for user login.
* Fetch the Products list.
* Fetch the Models list based on the product.
* Fetch the Bank ,Tenure Details Based on Product and Model.
* Capture the EMI,Card transaction Details.

**3.Design System Overview**

EMI-Solution in the present implementation relies on the Tab based Andriod App.

All Service offers are based on the EMI details.

A typical use case for the EMI Solution is :

* Post successful of the approved transaction,Cashier will login to

Innoviti Application ,once click on Login button.

* Client App will communicate with emi solution Server.
* An HTTP Post request sent to emi solution server with the client

And user login information to server .

* Server will send the authorization details to the client with token details and redirect to EMI Inquiry page.
* Cashier will select the product,model,bank,tenure details based on the product details purchased by the customer,All the details are fetched from the innoviti emi server.
* Cashier will enter the transaction and emi details will be submitted.
* Server will store the all the detail in to db.

**4. EMI-Solution API**

Following is the proposed API for EMI Solution to be used.API Pass at the minimum as part of Http Header

**4.1.Accessing token from Authorization Server**

**Description :** Responsible for the verification of user identify and providing the

tokens.

**Verb**  : Http Post.

**Headers** : Authorization :Client id & Client Password.

**Body** : grant\_type :password ,

username:demo, password:password

**API Endpoint :** /oauth/token

**Snippet of Request** :

POST /auth-api/oauth/token HTTP/1.1  
 Accept: application/json;charset=UTF-8  
 Authorization: Basic dHJ1c3RlZC1hcHA6cGFzc3dvcmQ=  
 Content-Type: application/x-www-form-urlencoded  
 grant\_type=password&username=demo&password=password

[http://l92.168.0.32:9171/oauth/token](about:blank)

**Snippet of Response** :

{

   "access\_token": "fe0a38cf-3114-4bb6-9a1c-3bfec4a05511",

"token\_type": "bearer",

"refresh\_token": "0816dd60-1287-4a2a-ae94-121898ad58cb",

"expires\_in": 44999,

"scope": "read write trust",

"name": "Demo"

}

**4.2.Refresh token from Authorization Server**

**Description :** when access token expire responsible for the verification of user

identify and providing the tokens.

**Verb**  : Http Post.

**Headers** : Authorization :Client id & Client Password.

**Body** : grant\_type :refresh\_token ,

refresh\_token :0816dd60-1287-4a2a-ae94-121898ad58cb

**API Endpoint :** /oauth/token

**Snippet of Request** :

POST /auth-api/oauth/token HTTP/1.1

Accept: application/json;charset=UTF-8  
 Authorization: Basic dHJ1c3RlZC1hcHA6cGFzc3dvcmQ=  
 Content-Type: application/x-www-form-urlencoded g

grant\_type=refresh\_token&refresh\_token=0816dd60-1287-4a2a-ae94-121898ad58cb

[http://l92.168.0.32:9171/oauth/token](about:blank)

**Snippet of Response** :

{

   "access\_token": "fe0a38cf-3114-4bb6-9a1c-3bfec4a05511",

"token\_type": "bearer",

"refresh\_token": "0816dd60-1287-4a2a-ae94-121898ad58cb",

"expires\_in": 44999,

"scope": "read write trust",

"name": "Demo"

}

**5. EMI-Solution Data Model**

Following the database schema :

Database name : emi\_solution

CREATE TABLE oauth\_client\_details (

client\_id VARCHAR(255) PRIMARY KEY,

resource\_ids VARCHAR(255),

client\_secret VARCHAR(255),

scope VARCHAR(255),

authorized\_grant\_types VARCHAR(255),

web\_server\_redirect\_uri VARCHAR(255),

authorities VARCHAR(255),

access\_token\_validity INTEGER,

refresh\_token\_validity INTEGER,

additional\_information VARCHAR(255),

autoapprove VARCHAR(255)

);

CREATE TABLE oauth\_client\_token (

token\_id VARCHAR(255),

token BLOB,

authentication\_id VARCHAR(255),

user\_name VARCHAR(255),

client\_id VARCHAR(255)

);

CREATE TABLE oauth\_access\_token (

token\_id VARCHAR(255),

token BLOB,

authentication\_id VARCHAR(255),

user\_name VARCHAR(255),

client\_id VARCHAR(255),

authentication BLOB,

refresh\_token VARCHAR(255)

);

CREATE TABLE oauth\_refresh\_token (

token\_id VARCHAR(255),

token BLOB,

authentication BLOB

);

CREATE TABLE oauth\_code (

CODE VARCHAR(255), authentication BLOB

);

CREATE TABLE `users` (

`user\_id` INT(10) UNSIGNED NOT NULL,

`email` VARCHAR(45) DEFAULT NULL,

`first\_name` VARCHAR(45) DEFAULT NULL,

`last\_name` VARCHAR(45) DEFAULT NULL,

`mobile` VARCHAR(15) DEFAULT NULL,

`password` VARCHAR(255) NOT NULL,

`user\_name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`user\_id`),

UNIQUE KEY `UK\_k8d0f2n7n88w1a16yhua64onx` (`user\_name`),

UNIQUE KEY `UK\_6dotkott2kjsp8vw4d0m25fb7` (`email`),

UNIQUE KEY `UK\_63cf888pmqtt5tipcne79xsbm` (`mobile`)

);

CREATE TABLE `roles` (

`role\_id` SMALLINT(5) UNSIGNED NOT NULL,

`role\_name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`role\_id`),

UNIQUE KEY `UK3cyq9kgtpbol1tlouij82oufa` (`role\_id`,`role\_name`),

UNIQUE KEY `UK\_716hgxp60ym1lifrdgp67xt5k` (`role\_name`)

);

CREATE TABLE `user\_roles` (

`user\_id` INT(10) UNSIGNED NOT NULL,

`role\_id` SMALLINT(5) UNSIGNED NOT NULL,

PRIMARY KEY (`user\_id`,`role\_id`),

CONSTRAINT `FKh8ciramu9cc9q3qcqiv4ue8a6` FOREIGN KEY (`role\_id`) REFERENCES `roles` (`role\_id`),

CONSTRAINT `FKhfh9dx7w3ubf1co1vdev94g3f` FOREIGN KEY (`user\_id`) REFERENCES `users` (`user\_id`)

);