Pricing Engine-Design Document

Introduction

1.1 Purpose

The purpose of this document is to outline the technical design and provide the data flow of pricing engine and its dependent APIs.

This document gives the information on:

- Event will getting trigger from Onboarding and create master record in pricing engine.
- It also make record disable to default records in pricing engine.
- How to create/update/sync Standard-charges
- How to create deal from Amigo/Qatar-amigo.

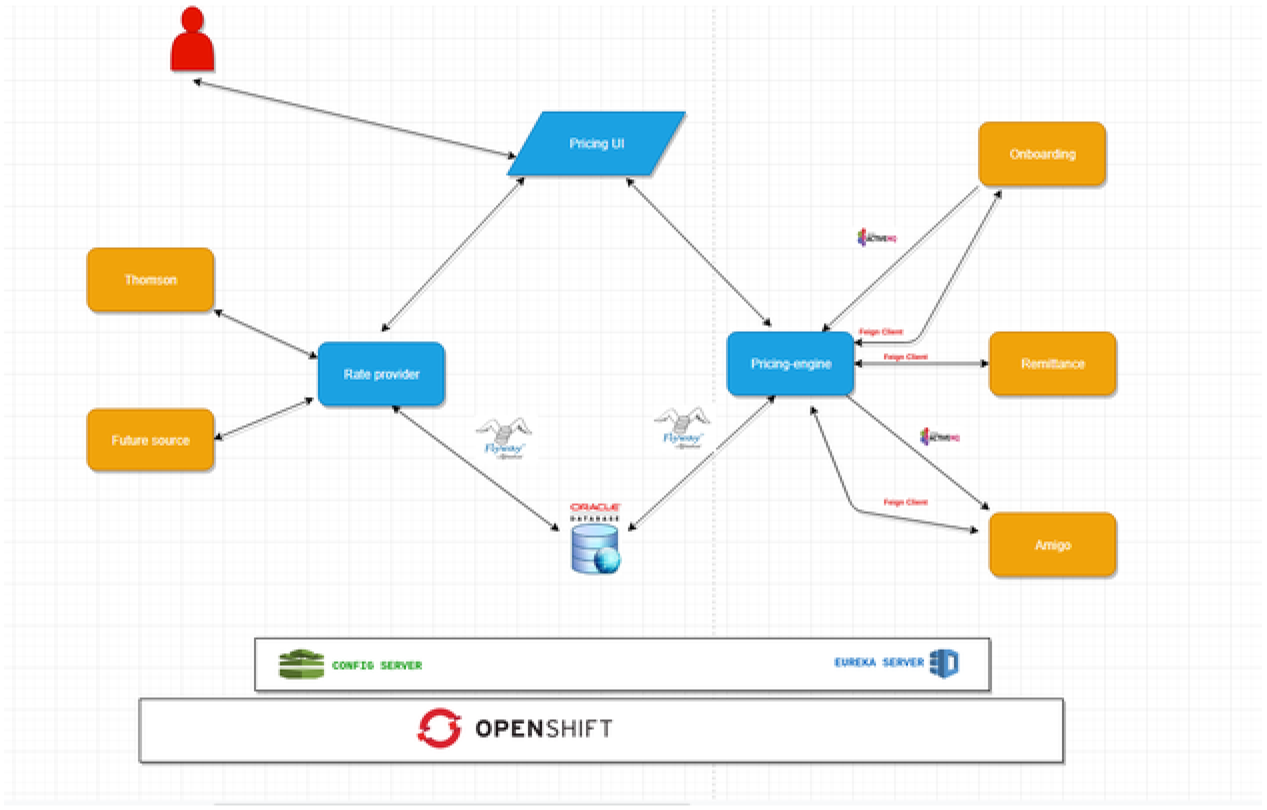
1.2 Scope

This document can be used by developer to understand the data flow of pricing engine. Functional team and tester can use this document for their reference.

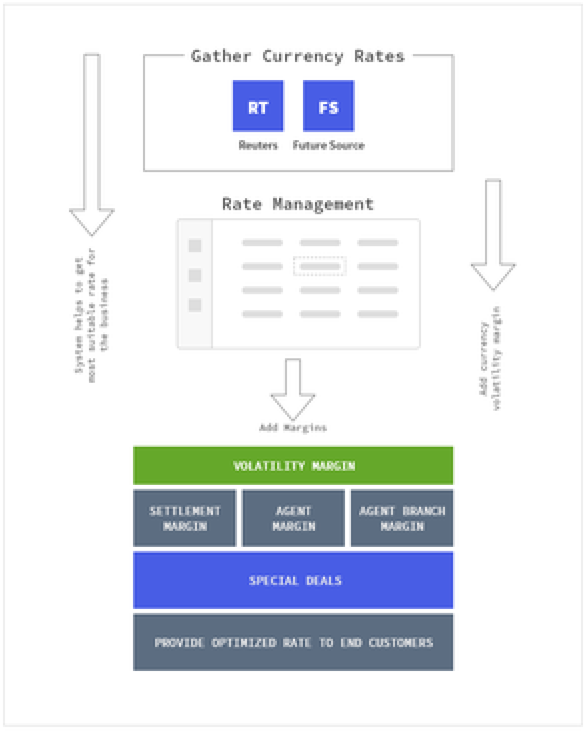
1.4 Audience

The intended audiences for this document are Unimoni Stakeholders, the project development teams, testers and technical architects.

Data flow-Pricing Engine



Solution Architecture-Pricing Engine



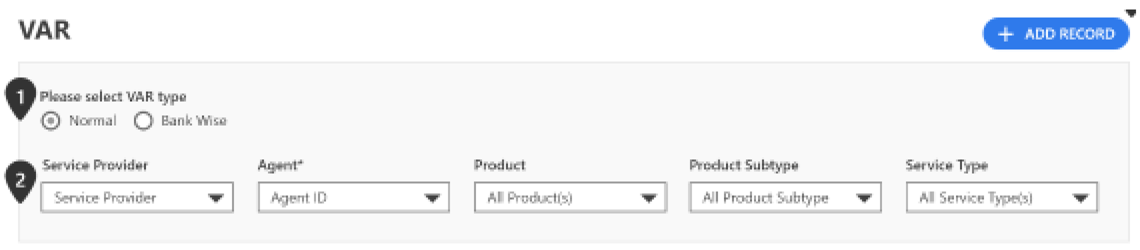
Master API

Brief Description

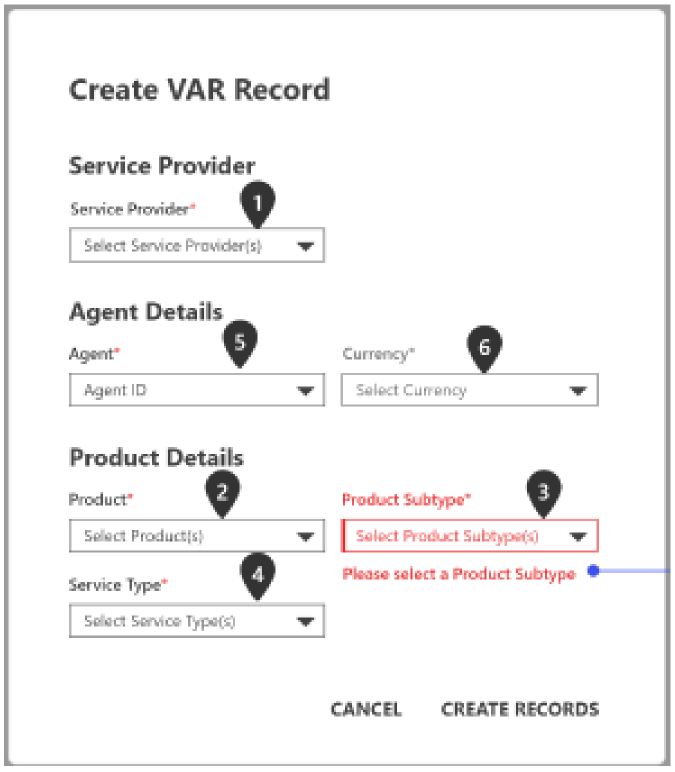
Pricing Engine application need some pre-defined data from Onboarding. All data has been fetched at the time of start-up the pricing-engine using event-listner. This master data contains Service Provider, Agents, Agent Branches, Banks, Currencies, Products/Sub Products/Service types and their relationship with each other.

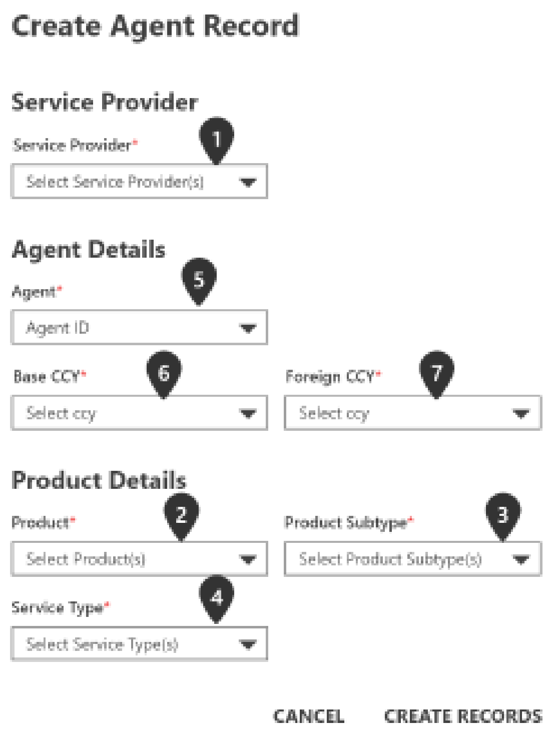
APIs are required by Pricing.

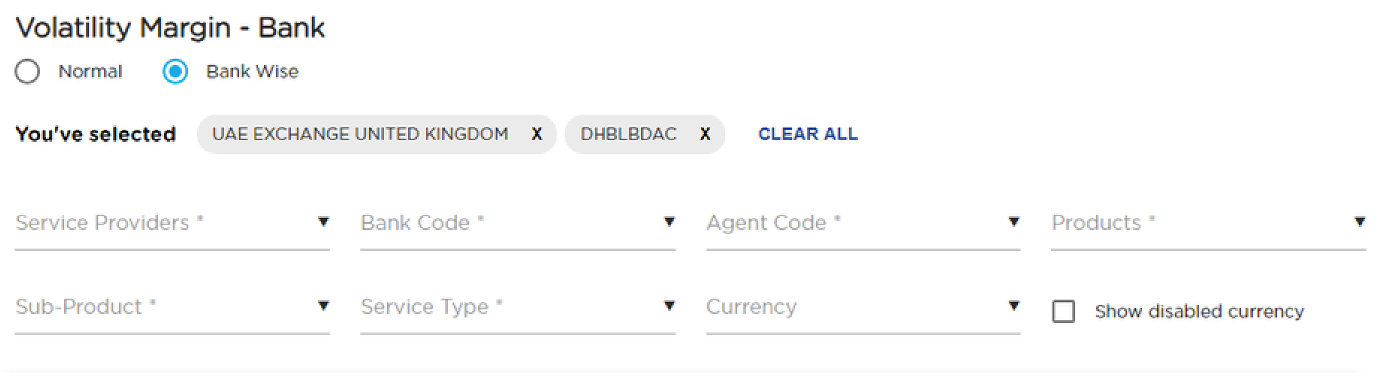
Please refer to following screen fragments where APIs are required at onboarding end.



Please select VAR type: [✓ Normal] Bank Wise

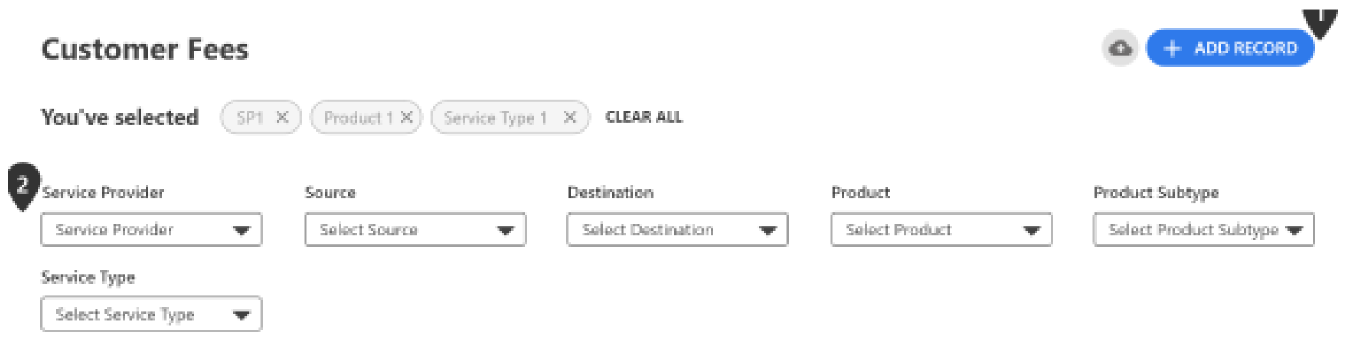






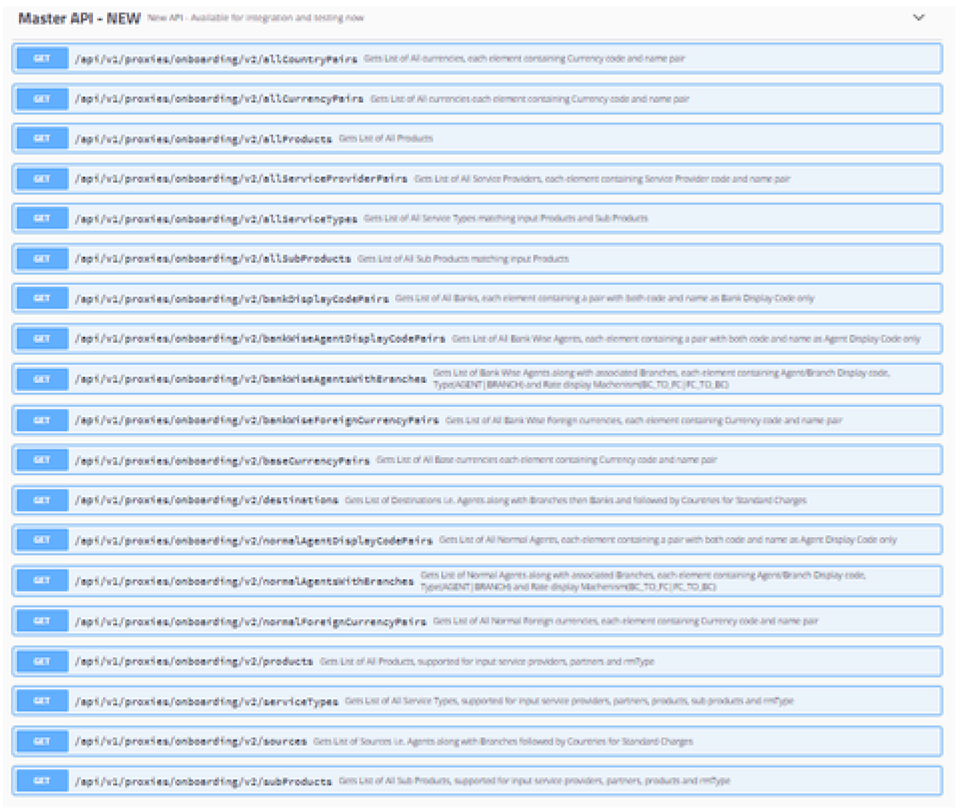
Normal [✓ Bank Wise]

You've selected UAE EXCHANGE UNITED KINGDOM X DHBLBDAC X



A brief snapshot of all the APIs developed at pricing engine level, backed by master data adapter is as follows. Complete details of all these APIs such as Description of each API, request and response is documented in the attachment **Pricing Master Data APIs.pdf**

- For Normal screens user select the service provider first, then list of Agents is populated by hitting the API with selected service providers as query parameters. Then the list of products would be populated by selected service providers and Agents. Similarly sub products and service types would be populated by required selected values from previous dropdowns. On some of screens we display all supported currencies but for other screens we need to display base currencies which are associated with agents or foreign currencies which are associated with banks (either specific pricing true|false for the given bank)
- Similarly for bankwise screens user select the service provider first, then list of Banks is populated by hitting the API with selected service providers as query parameters. Then rest of the dropdown lists are populated by selected service providers, banks, products, sub products etc. whichever is applicable.
- In Agent Margin and Agent Branch margin screens the agents dropdown contains agent along with its branches, each element containing rate display mechanism. The list should contain one agent and its branches together then the next agent and its branches and so on.
- Pricing Engine works on Agent Display code, Agent Branch Display code and Bank Display codes as unique identifiers of Agent, Agent Branch and Bank respectively.
- Below snapshot is swagger documentations for all master api in pricing engine.



Master API - NEW New API - Available for integration and testing now

GET /api/v1/proxies/onboarding/v2/allCountryPairs Gets List of All currencies, each element containing Currency code and name pair

GET /api/v1/proxies/onboarding/v2/allCurrencyPairs Gets List of All currencies each element containing Currency code and name pair

GET /api/v1/proxies/onboarding/v2/allProducts Gets List of All Products

GET /api/v1/proxies/onboarding/v2/allServiceProviderPairs Gets List of All Service Providers, each element containing Service Provider code and name pair

GET /api/v1/proxies/onboarding/v2/allServiceTypes Gets List of All Service Types matching input Products and Sub Products

GET /api/v1/proxies/onboarding/v2/allSubProducts Gets List of All Sub Products matching input Products

GET /api/v1/proxies/onboarding/v2/bankDisplayCodePairs Gets List of All Banks, each element containing a pair with both code and name as Bank Display Code only

GET /api/v1/proxies/onboarding/v2/bankWiseAgentDisplayCodePairs Gets List of All Bank Wise Agents, each element containing a pair with both code and name as Agent Display Code only

GET /api/v1/proxies/onboarding/v2/bankWiseAgentsWithBranches Gets List of Bank Wise Agents along with associated Branches, each element containing Agent/Branch Display code, Type(AGENT | BRANCH) and Rate display Machenism(BC_TO_FC | FC_TO_BC)

GET /api/v1/proxies/onboarding/v2/bankWiseForeignCurrencyPairs Gets List of All Bank Wise Foreign currencies, each element containing Currency code and name pair

GET /api/v1/proxies/onboarding/v2/baseCurrencyPairs Gets List of All Base currencies each element containing Currency code and name pair

GET /api/v1/proxies/onboarding/v2/destinations Gets List of Destinations i.e. Agents along with Branches then Banks and followed by Countries for Standard Charges

GET /api/v1/proxies/onboarding/v2/normalAgentDisplayCodePairs Gets List of All Normal Agents, each element containing a pair with both code and name as Agent Display Code only

GET /api/v1/proxies/onboarding/v2/normalAgentsWithBranches Gets List of Normal Agents along with associated Branches, each element containing Agent/Branch Display code, Type(AGENT | BRANCH) and Rate display Machenism(BC_TO_FC | FC_TO_BC)

GET /api/v1/proxies/onboarding/v2/normalForeignCurrencyPairs Gets List of All Normal Foreign currencies, each element containing Currency code and name pair

GET /api/v1/proxies/onboarding/v2/products Gets List of All Products, supported for input service providers, partners and rmType

GET /api/v1/proxies/onboarding/v2/serviceTypes Gets List of All Service Types, supported for input service providers, partners, products, sub products and rmType

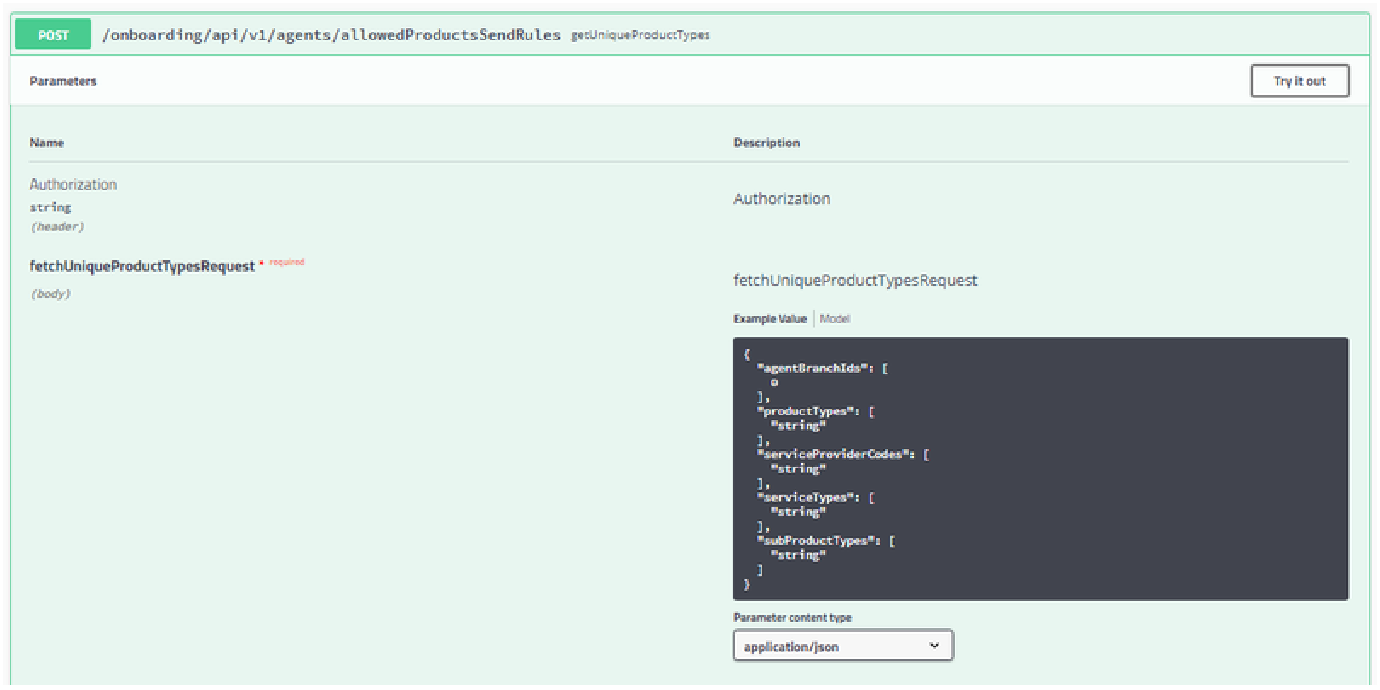
GET /api/v1/proxies/onboarding/v2/sources Gets List of Sources i.e. Agents along with Branches followed by Countries for Standard Charges

GET /api/v1/proxies/onboarding/v2/subProducts Gets List of All Sub Products, supported for input service providers, partners, products and rmType

APIs are given by Onboarding

Normal (Agent wise) Data API :

This API basically gives us all required data associated with all agents and branches along with their product, sub product and service types to be used on normal screens.



POST /onboarding/api/v1/agents/allowedProductsSendRules getUniqueProductTypes

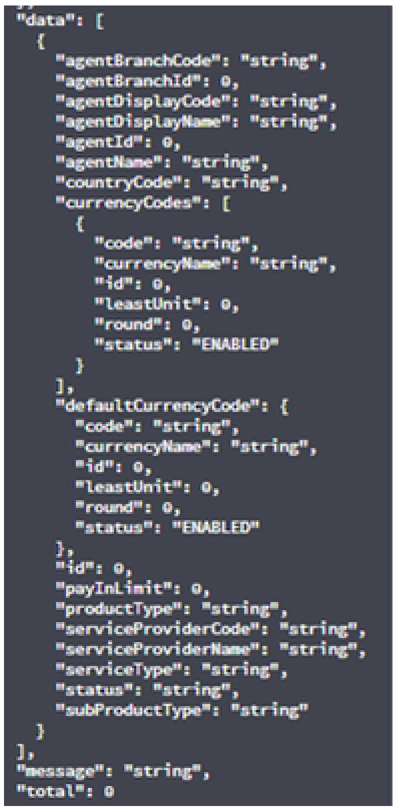
Parameters | Try it out

Name	Description
Authorization string *(header)*	Authorization
fetchUniqueProductTypesRequest * required *(body)*	fetchUniqueProductTypesRequest

Example Value | Model

```
{
  "agentBranchIds": [
    0
  ],
  "productTypes": [
    "string"
  ],
  "serviceProviderCodes": [
    "string"
  ],
  "serviceTypes": [
    "string"
  ],
  "subProductTypes": [
    "string"
  ]
}
```

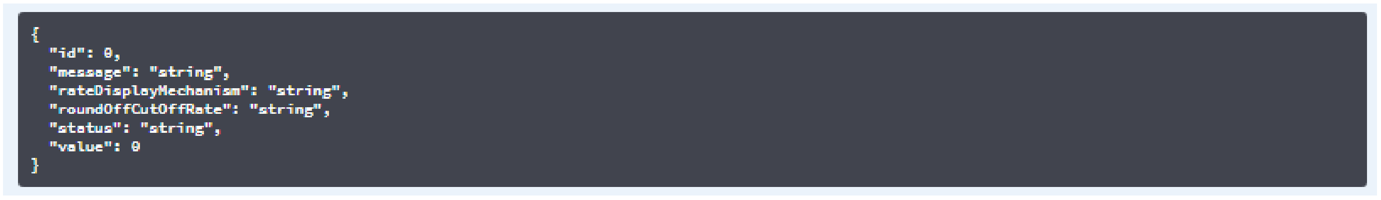
Parameter content type: application/json



```
  "data": [
    {
      "agentBranchCode": "string",
      "agentBranchId": 0,
      "agentDisplayCode": "string",
      "agentDisplayName": "string",
      "agentId": 0,
      "agentName": "string",
      "countryCode": "string",
      "currencyCodes": [
        {
          "code": "string",
          "currencyName": "string",
          "id": 0,
          "leastUnit": 0,
          "round": 0,
          "status": "ENABLED"
        }
      ],
      "defaultCurrencyCode": {
        "code": "string",
        "currencyName": "string",
        "id": 0,
        "leastUnit": 0,
        "round": 0,
        "status": "ENABLED"
      },
      "id": 0,
      "payInLimit": 0,
      "productType": "string",
      "serviceProviderCode": "string",
      "serviceProviderName": "string",
      "serviceType": "string",
      "status": "string",
      "subProductType": "string"
    }
  ],
  "message": "string",
  "total": 0
```

For each agent and branch we need rate display mechanism which is used in all creation and update login on Agent Margin and agent Branch Margin. So we need to call below api.

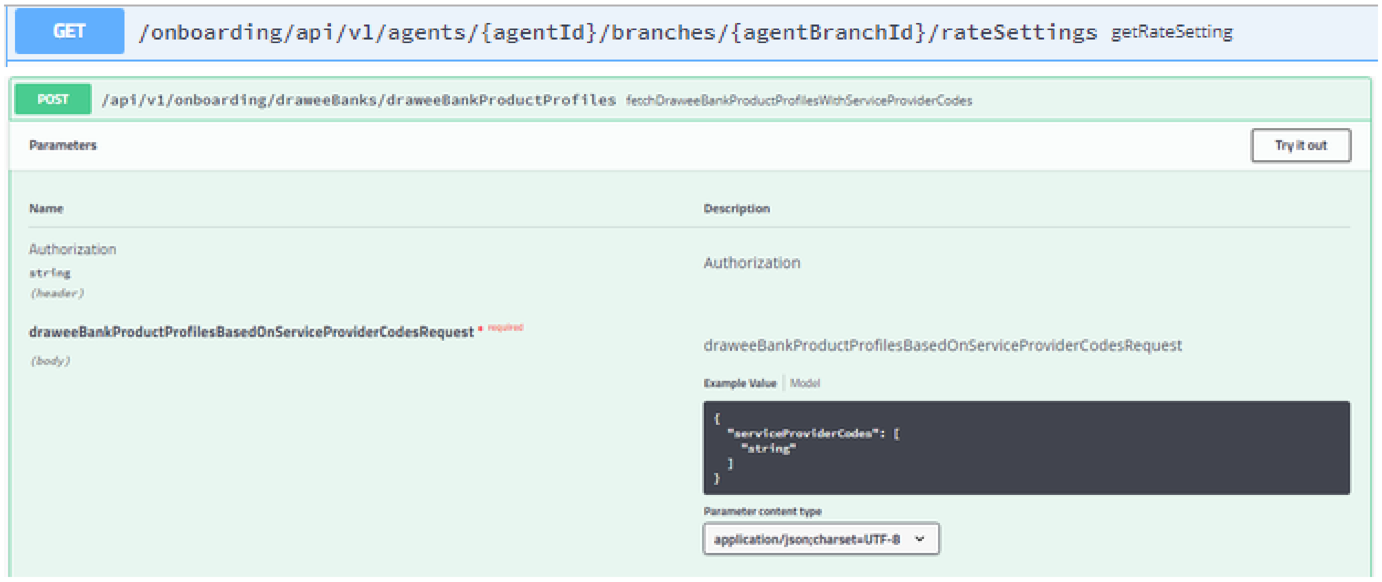
http://agent-onboarding-service-snap.dev.apps.ocp.uaeexchange.com/onboarding/api/v1/agents/453/branches/45/rateSettings



```
{
  "id": 0,
  "message": "string",
  "rateDisplayMechanism": "string",
  "roundOffCutOffRate": "string",
  "status": "string",
  "value": 0
}
```

Bankwise Data API:

Following API gives all the data associated with a bank for selected service provider.



GET /onboarding/api/v1/agents/{agentId}/branches/{agentBranchId}/rateSettings getRateSetting

POST /api/v1/onboarding/draweeBanks/draweeBankProductProfiles fetchDraweeBankProductProfilesWithServiceProviderCodes

Parameters

Try it out

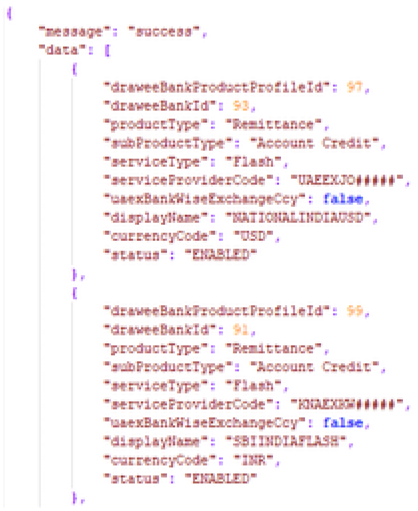
Name	Description
Authorization string *(header)*	Authorization
draweeBankProductProfilesBasedOnServiceProviderCodesRequest * required *(body)*	draweeBankProductProfilesBasedOnServiceProviderCodesRequest Example Value | Model

```
{
  "serviceProviderCodes": [
    "string"
  ]
}
```

Parameter content type

application/json;charset=UTF-8

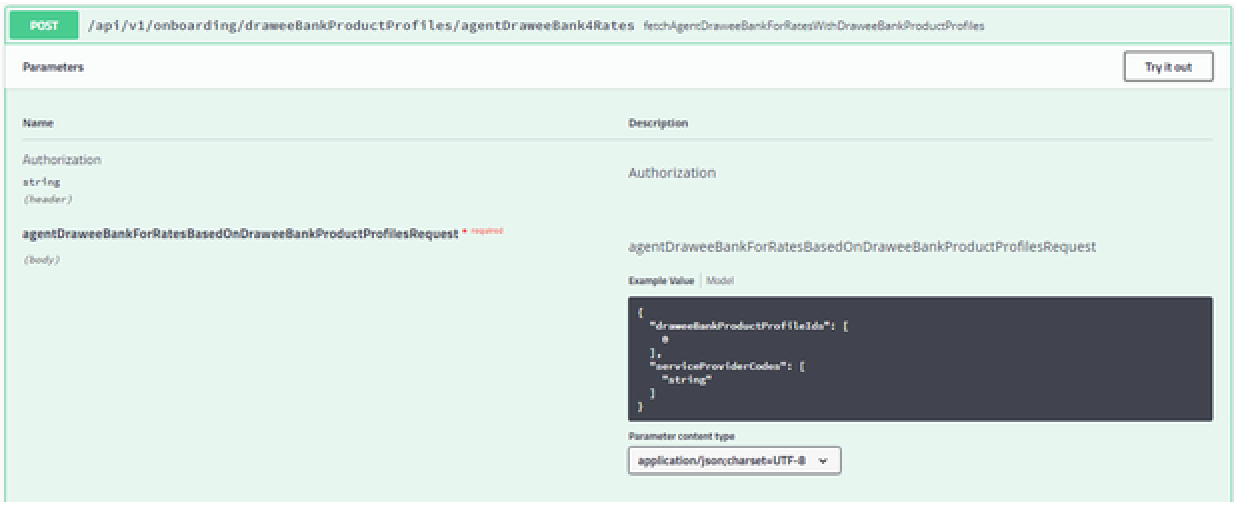
Response:



```
{
    "message": "success",
    "data": [
        {
            "draweeBankProductProfileId": 97,
            "draweeBankId": 93,
            "productType": "Remittance",
            "subProductType": "Account Credit",
            "serviceType": "Flash",
            "serviceProviderCode": "UAEEXJO#####",
            "uaexBankWiseExchangeCcy": false,
            "displayName": "NATIONALINDIAUSD",
            "currencyCode": "USD",
            "status": "ENABLED"
        },
        {
            "draweeBankProductProfileId": 99,
            "draweeBankId": 91,
            "productType": "Remittance",
            "subProductType": "Account Credit",
            "serviceType": "Flash",
            "serviceProviderCode": "KNAEXKW#####",
            "uaexBankWiseExchangeCcy": false,
            "displayName": "SBIINDIAFLASH",
            "currencyCode": "INR",
            "status": "ENABLED"
        },
```

Bankwise Agents API:

Following API give the agent branches associated with a Bank.



POST /api/v1/onboarding/draweeBankProductProfiles/agentDraweeBank4Rates fetchAgentDraweeBankForRatesWithDraweeBankProductProfiles

Parameters

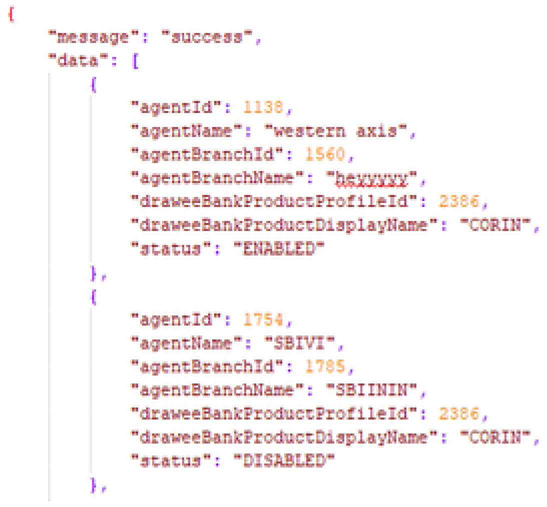
Name	Description
Authorization string *(header)*	Authorization
agentDraweeBankForRatesBasedOnDraweeBankProductProfilesRequest * required *(body)*	agentDraweeBankForRatesBasedOnDraweeBankProductProfilesRequest

Example Value | Model

```
{
  "draweeBankProductProfileIds": [
    0
  ],
  "serviceProviderCodes": [
    "string"
  ]
}
```

Parameter content type: application/json;charset=UTF-8

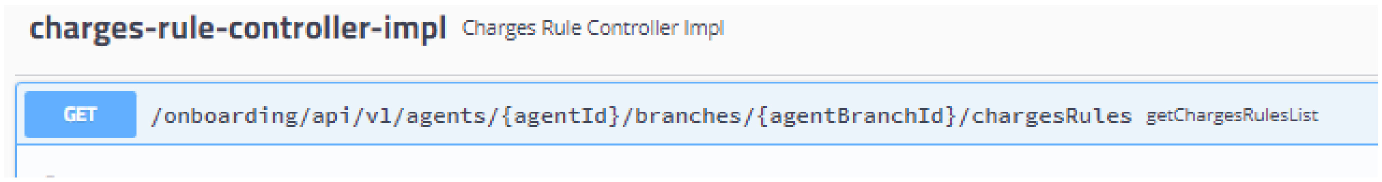
Response:



```
{
    "message": "success",
    "data": [
        {
            "agentId": 1138,
            "agentName": "western axis",
            "agentBranchId": 1560,
            "agentBranchName": "heyyyyy",
            "draweeBankProductProfileId": 2386,
            "draweeBankProductDisplayName": "CORIN",
            "status": "ENABLED"
        },
        {
            "agentId": 1754,
            "agentName": "SBIVI",
            "agentBranchId": 1785,
            "agentBranchName": "SBIININ",
            "draweeBankProductProfileId": 2386,
            "draweeBankProductDisplayName": "CORIN",
            "status": "DISABLED"
        },
```

Charge Rules API:

To process remittance transaction API we need Charge rules for a particular agent branch, product, sub product and service types.



charges-rule-controller-impl Charges Rule Controller Impl

GET /onboarding/api/v1/agents/{agentId}/branches/{agentBranchId}/chargesRules getChargesRulesList

Response:

{

"message": "success",

"data": [

{

"id": 5867,

```
"productType": "Remittance",
"subProductType": "Account Credit",
"serviceType": "Normal",
"reasonCode": "CUSTOMER_WISH",
"chargesRuleSettings": {
"otherCharges": true,
"commission": false,
"tax": false,
"cardCharges": false,
"additionalCharges": false
},
"status": "ENABLED"
}
],
"total": 1
}
```

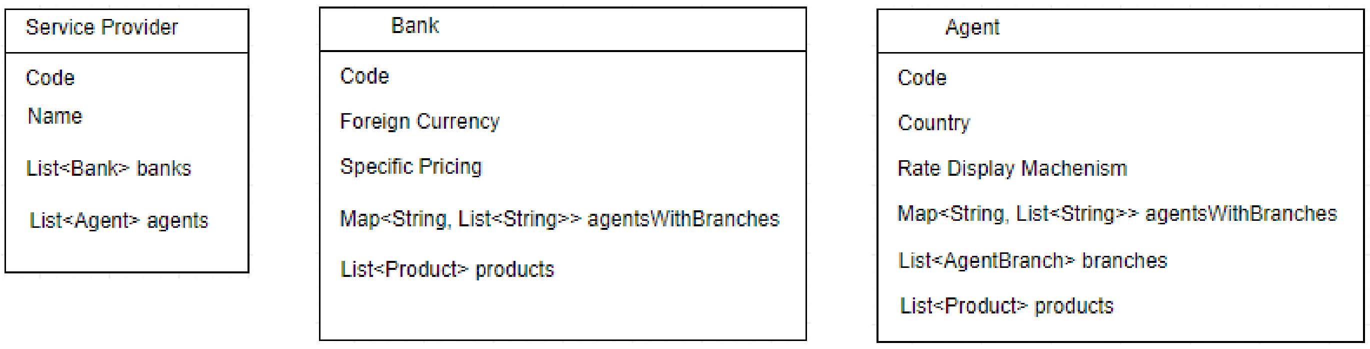
Designing of Master Data adapter

The idea is to develop an adapter over Onboarding API which can handle all complexity associated with onboarding API consumption and gives clearly defined interfaces meeting the exact requirement of Pricing Engine. It’s an Anti-Corruption layer between Pricing and Onboarding.

When pricing application starts it collects all the agent wise and bankwise data from Onboarding by calling above mentioned APIs. Parses and filters all the data and put in a data structure (Logically a tree structure) with the aligned relationship among different entities.

Once we have the data localized at Pricing Service end in required tree structure. Then we can query over this data to get required data set with all required filters.

Following are the main Objects we are caching

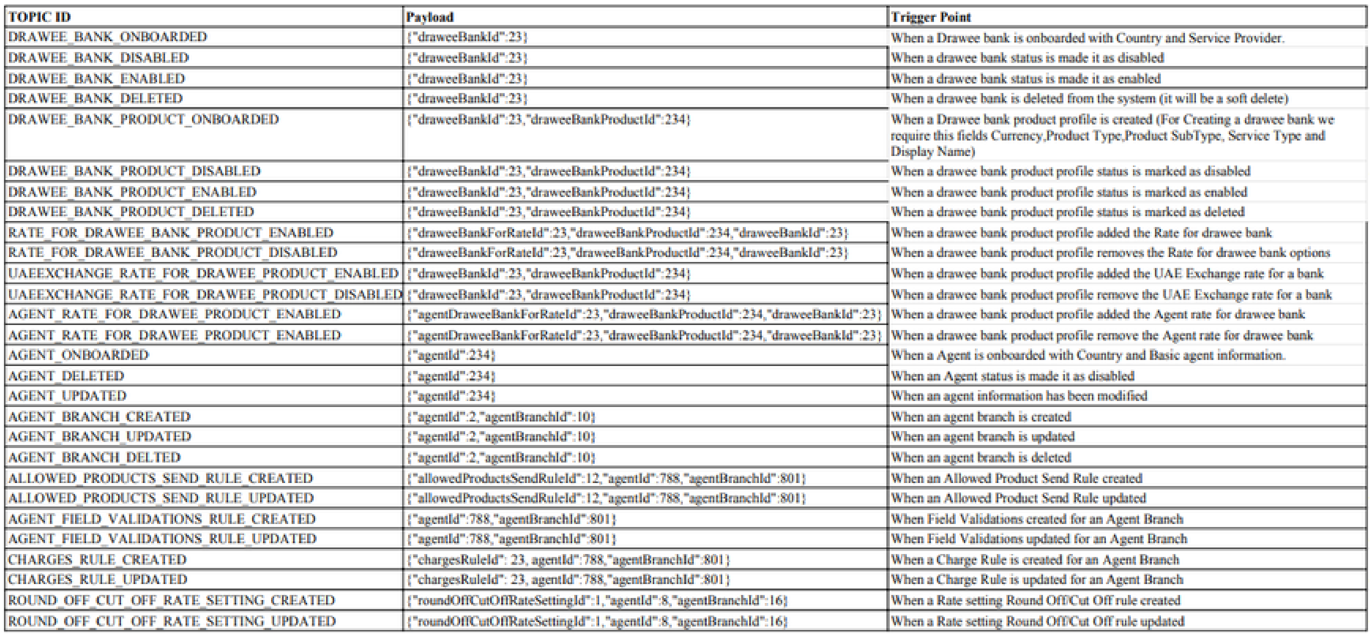


- There would be some application startup time for Pricing engine as it needs to collect all data from Onboarding at startup time. With current data set on Integration the master data initialization time is 10-20 seconds. As to get the rate display mechanism for each agent and branch there is a separate rest call, otherwise the startup time would reduce to few seconds only.
- The startup time would increase as the data size increases. Having mentioned that we are caching very minimal attributes of an object. As confirmed with business the maximum number of banks would be 200-300 and number of agents/branches would be around 2000, which do not qualify to be a big data set or of any concern from memory perspective/load time perspective. There should not be any issue keeping this into memory.0
- Once Pricing starts and gather all the data, further data changes at Onboarding are transported to Pricing by JMS events.

Events Onboarding Design

1. Event will trigger from onboarding application and Pricing will consume the topic details .
2. Pricing will consume the topic .Then it will call multiple API to create default record in Pricing.

Event details



TOPIC ID	Payload	Trigger Point
DRAWEE_BANK_ONBOARDED	{"draweeBankId":23}	When a Drawee bank is onboarded with Country and Service Provider.
DRAWEE_BANK_DISABLED	{"draweeBankId":23}	When a drawee bank status is made it as disabled
DRAWEE_BANK_ENABLED	{"draweeBankId":23}	When a drawee bank status is made it as enabled
DRAWEE_BANK_DELETED	{"draweeBankId":23}	When a drawee bank is deleted from the system (it will be a soft delete)
DRAWEE_BANK_PRODUCT_ONBOARDED	{"draweeBankId":23,"draweeBankProductId":234}	When a Drawee bank product profile is created (For Creating a drawee bank we require this fields Currency,Product Type,Product SubType, Service Type and Display Name)
DRAWEE_BANK_PRODUCT_DISABLED	{"draweeBankId":23,"draweeBankProductId":234}	When a drawee bank product profile status is marked as disabled
DRAWEE_BANK_PRODUCT_ENABLED	{"draweeBankId":23,"draweeBankProductId":234}	When a drawee bank product profile status is marked as enabled
DRAWEE_BANK_PRODUCT_DELETED	{"draweeBankId":23,"draweeBankProductId":234}	When a drawee bank product profile status is marked as deleted
RATE_FOR_DRAWEE_BANK_PRODUCT_ENABLED	{"draweeBankForRateId":23,"draweeBankProductId":234,"draweeBankId":23}	When a drawee bank product profile added the Rate for drawee bank
RATE_FOR_DRAWEE_BANK_PRODUCT_DISABLED	{"draweeBankForRateId":23,"draweeBankProductId":234,"draweeBankId":23}	When a drawee bank product profile removes the Rate for drawee bank options
UAEEXCHANGE_RATE_FOR_DRAWEE_PRODUCT_ENABLED	{"draweeBankId":23,"draweeBankProductId":234}	When a drawee bank product profile added the UAE Exchange rate for a bank
UAEEXCHANGE_RATE_FOR_DRAWEE_PRODUCT_DISABLED	{"draweeBankId":23,"draweeBankProductId":234}	When a drawee bank product profile remove the UAE Exchange rate for a bank
AGENT_RATE_FOR_DRAWEE_PRODUCT_ENABLED	{"agentDraweeBankForRateId":23,"draweeBankProductId":234,"draweeBankId":23}	When a drawee bank product profile added the Agent rate for drawee bank
AGENT_RATE_FOR_DRAWEE_PRODUCT_ENABLED	{"agentDraweeBankForRateId":23,"draweeBankProductId":234,"draweeBankId":23}	When a drawee bank product profile remove the Agent rate for drawee bank
AGENT_ONBOARDED	{"agentId":234}	When a Agent is onboarded with Country and Basic agent information.
AGENT_DELETED	{"agentId":234}	When an Agent status is made it as disabled
AGENT_UPDATED	{"agentId":234}	When an agent information has been modified
AGENT_BRANCH_CREATED	{"agentId":2,"agentBranchId":10}	When an agent branch is created
AGENT_BRANCH_UPDATED	{"agentId":2,"agentBranchId":10}	When an agent branch is updated
AGENT_BRANCH_DELTED	{"agentId":2,"agentBranchId":10}	When an agent branch is deleted
ALLOWED_PRODUCTS_SEND_RULE_CREATED	{"allowedProductsSendRuleId":12,"agentId":788,"agentBranchId":801}	When an Allowed Product Send Rule created
ALLOWED_PRODUCTS_SEND_RULE_UPDATED	{"allowedProductsSendRuleId":12,"agentId":788,"agentBranchId":801}	When an Allowed Product Send Rule updated
AGENT_FIELD_VALIDATIONS_RULE_CREATED	{"agentId":788,"agentBranchId":801}	When Field Validations created for an Agent Branch
AGENT_FIELD_VALIDATIONS_RULE_UPDATED	{"agentId":788,"agentBranchId":801}	When Field Validations updated for an Agent Branch
CHARGES_RULE_CREATED	{"chargesRuleId": 23, agentId":788,"agentBranchId":801}	When a Charge Rule is created for an Agent Branch
CHARGES_RULE_UPDATED	{"chargesRuleId": 23, agentId":788,"agentBranchId":801}	When a Charge Rule is updated for an Agent Branch
ROUND_OFF_CUT_OFF_RATE_SETTING_CREATED	{"roundOffCutOffRateSettingId":1,"agentId":8,"agentBranchId":16}	When a Rate setting Round Off/Cut Off rule created
ROUND_OFF_CUT_OFF_RATE_SETTING_UPDATED	{"roundOffCutOffRateSettingId":1,"agentId":8,"agentBranchId":16}	When a Rate setting Round Off/Cut Off rule updated

1. **AGENT_DISABLED -** Payload**: {"agentId":234}**

- To get agent display code we need to call below api .

    http://agent-onboarding-service-snap.dev.apps.ocp.uaeexchange.com/onboarding/api/v1/agents/agent/{agentId}

2. DRAWEE_BANK_PRODUCT_DISABLED - Payload**: {"draweeBankId":23,"draweeBankProductId":234}**

- We need draweeBankProduct **display code**, **serviceProviderCode**, **products**, **currencies**, **productSubtype**, **serviceType** also in payload
- To get above details we need to call below API.

    http://bank-onboarding-service-snap.integration.apps.ocp.uaeexchange.com/api/v1/onboarding/draweeBanks/{draweeBankId}/draweeBankProductProfiles/{draweeBankProductProfileId}

3. DRAWEE_BANK_PRODUCT_DELETED - Payload**: {"draweeBankId":23,"draweeBankProductId":234}**

- We need draweeBankProduct **display code**, **serviceProviderCode**, **products**, **currencies**, **product**, **productSubtype**, **serviceType**
- To get above details we need to call below API

    http://bank-onboarding-service-snap.integration.apps.ocp.uaeexchange.com/api/v1/onboarding/draweeBanks/{draweeBankId}/draweeBankProductProfiles/{draweeBankProductProfileId}

4. ALLOWED_PRODUCTS_SEND_RULE_CREATED - Payload**: {"allowedProductsSendRuleId":12,"agentId":788,"agentBranchId":801}**

- We need **service provider code**, **product**, **product sub type**, **service type**, **base currencies**, **foreign currencies**

- To get service provider code, product sub type, service type. We need to call below API.

  http://agent-onboarding-service-snap.dev.apps.ocp.uaeexchange.com/onboarding/api/v1/agents/{agentId}/branches/{agentBranchId}/allowedProductsSendRules/{allowedProductsSendRuleId}

- To get base currencies we need to call below API

  http://agent-onboarding-service-snap.dev.apps.ocp.uaeexchange.com/onboarding/api/v1/agents/{agentId}/branches/{branchId}/allowedProductsSendRules

- To get foreign currencies we need to call below API

  http://pricing-engine-service.apps.uat.unimoni.com/api/v1/proxies/onboarding/v2/normalForeignCurrencyPairs?serviceProviders=UAEXMUK%23%23%23%23%23&nameAsLabel=false

5. ROUND_OFF_CUT_OFF_RATE_SETTING_CREATED - Payload**: {"roundOffCutOffRateSettingId":1,"agentId":8,"agentBranchId":16}**

- We need **agent display code, service provider code**, **product sub type**, **service type**, **base currencies**, **foreign currencies**, **rate display mechanism**.
- To get agent display code we need to call below API.

  http://agent-onboarding-service-snap.dev.apps.ocp.uaeexchange.com/onboarding/api/v1/agents/agent/{agentId}

- To get service provider, product, product sub type, service type. We need to call below API.

  http://agent-onboarding-service-snap.dev.apps.ocp.uaeexchange.com/onboarding/api/v1/agents/{agentId}/branches/{agentBranchId}/allowedProductsSendRules/{allowedProductsSendRuleId}

- To get base currencies we need to call below API.

  http://agent-onboarding-service-snap.dev.apps.ocp.uaeexchange.com/onboarding/api/v1/agents/{agentId}/branches/{branchId}/allowedProductsSendRules

- To get foreign currencies we need to call below API.

  http://pricing-engine-service.apps.uat.unimoni.com/api/v1/proxies/onboarding/v2/normalForeignCurrencyPairs?serviceProviders=UAEXMUK%23%23%23%23%23&nameAsLabel=false

- To get rate display mechanism we need to call below API.

  http://agent-onboarding-service-snap.dev.apps.ocp.uaeexchange.com/onboarding/api/v1/agents/{agentId}/branches/{agentBranchId}/rateSettings

6. AGENT_RATE_FOR_DRAWEE_PRODUCT_ENABLED - Payload**: {"agentDraweeBankForRateId":23,"draweeBankProductId":234,"draweeBankId":23}**

- We need **agent display code**, **drawee Bank display code**, **service provider code**, **product sub type**, **service type**, **base currencies**, **foreign currencies**, **rate display mechanism**.
- To get agent display code we need to call below API.

  http://agent-onboarding-service-snap.dev.apps.ocp.uaeexchange.com/onboarding/api/v1/agents/agent/{agentId}

- To get service provider code, product, Drawee Bank id, status, bank display name we need to call below API.

  http://bank-onboarding-service-snap.dev.apps.ocp.uaeexchange.com/onboarding/api/v1/draweeBanks/{draweeBankId}/draweeBankProductProfiles/{draweeBankProductProfileId}

- To get agent id and agent branch id we need to call below API.

http://bank-onboarding-service-snap.dev.apps.ocp.uaeexchange.com/onboarding/api/v1/draweeBanks/{draweeBankId}/draweeBankProductProfiles/{draweeBankProductProfileId}

- To get rate display mechanism we need to call below API.

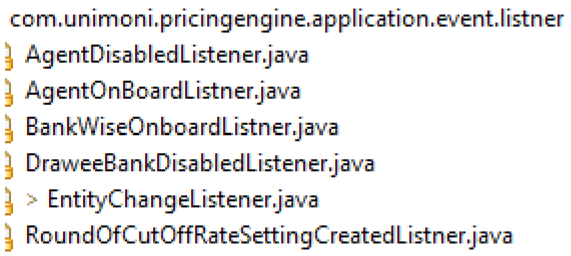
  http://agent-onboarding-service-snap.dev.apps.ocp.uaeexchange.com/onboarding/api/v1/agents/{agentId}/branches/{agentBranchId}/rateSettings

3. Applications Framework

The Application Framework components provide utility classes and other components that are used across the application.

3.1 Event Handler

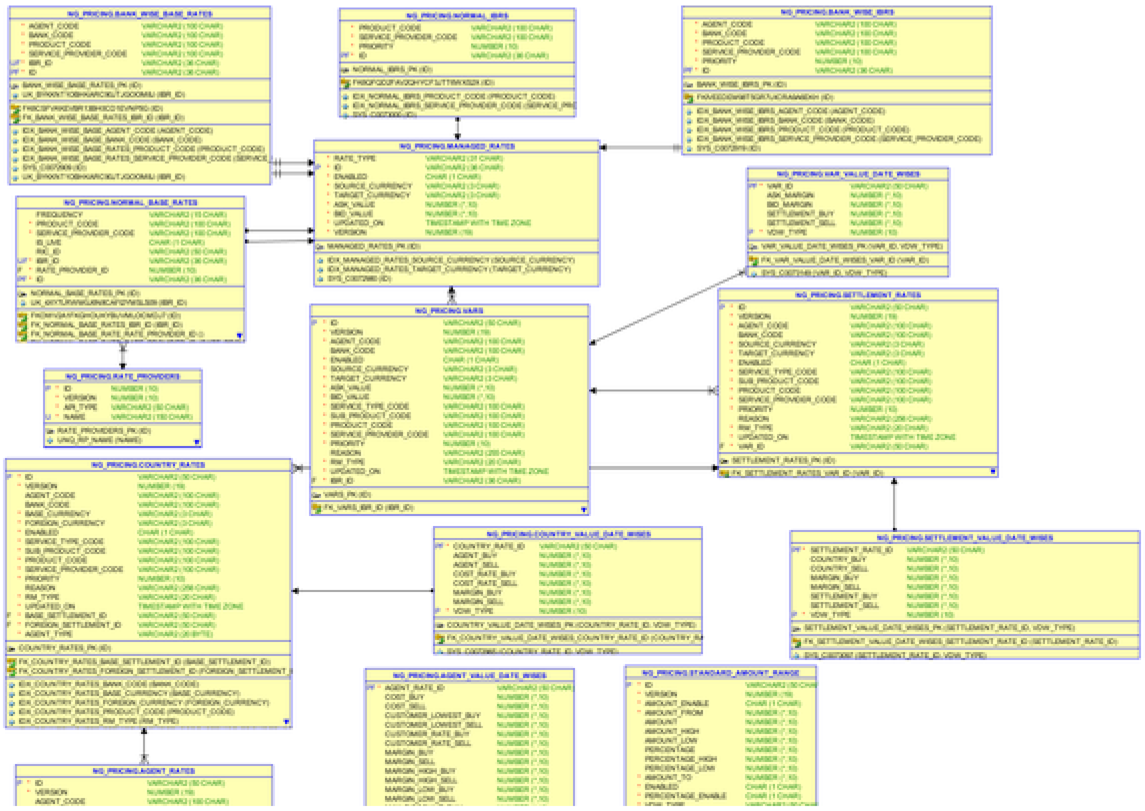
All Onboarding events are listen in “**com.unimoni.pricingengine.application.event.listner**” package.

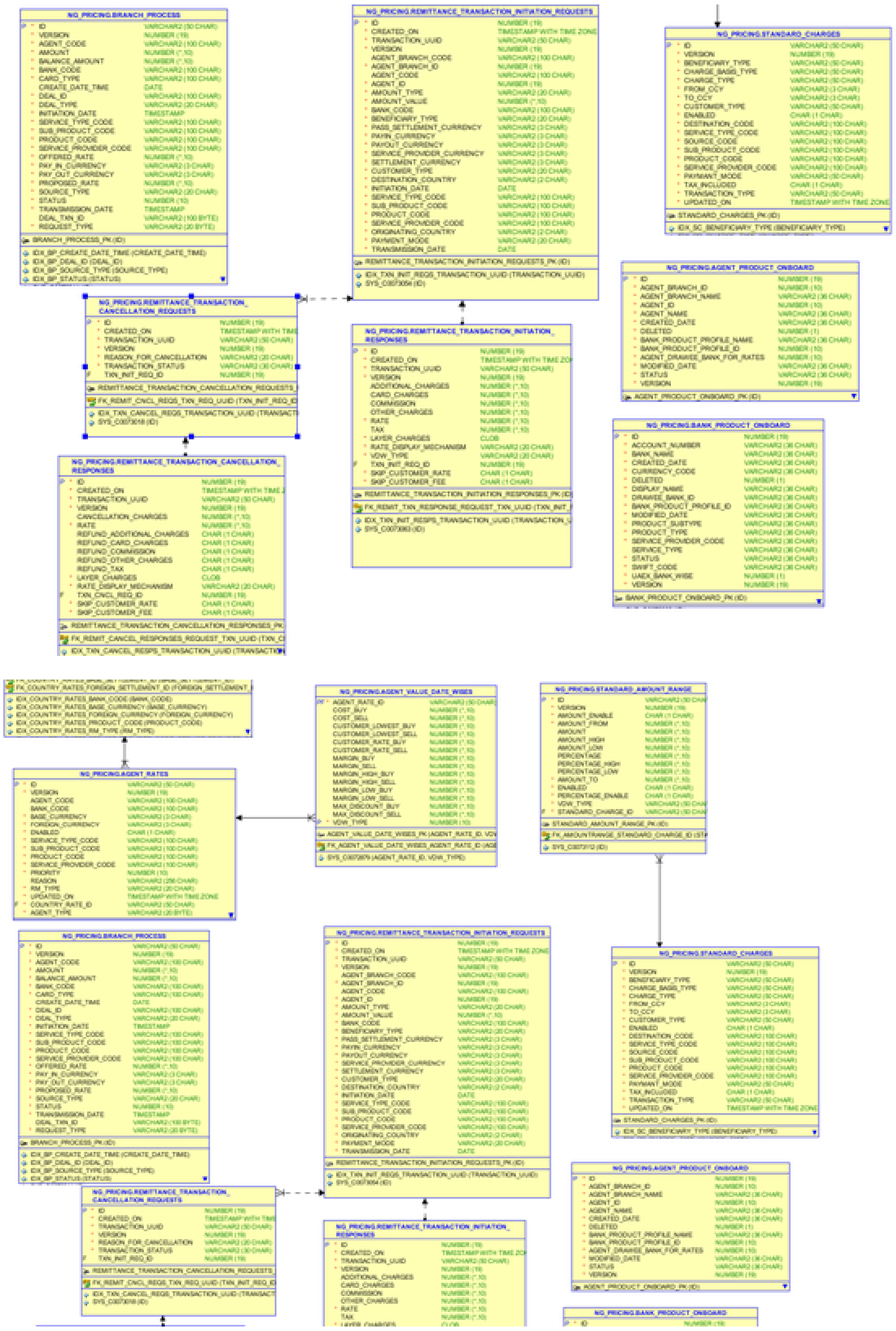


com.unimoni.pricingengine.application.event.listner
AgentDisabledListener.java
AgentOnBoardListner.java
BankWiseOnboardListner.java
DraweeBankDisabledListener.java
> EntityChangeListener.java
RoundOfCutOffRateSettingCreatedListner.java

Database Schema

ER-DIAGRAM-Onboarding events


NG_PRICING_BANK_WISE_BASE_RATES
NG_PRICING_NORMAL_IBRS
NG_PRICING_BANK_WISE_IBRS
NG_PRICING_MANAGED_RATES
NG_PRICING_NORMAL_BASE_RATES
NG_PRICING_VAR_VALUE_DATE_WISES
NG_PRICING_VARS
NG_PRICING_SETTLEMENT_RATES
NG_PRICING_RATE_PROVIDERS
NG_PRICING_COUNTRY_RATES
NG_PRICING_COUNTRY_VALUE_DATE_WISES
NG_PRICING_SETTLEMENT_VALUE_DATE_WISES
NG_PRICING_AGENT_VALUE_DATE_WISES
NG_PRICING_STANDARD_AMOUNT_RANGE
NG_PRICING_AGENT_RATES


NG_PRICING.BRANCH_PROCESS
NG_PRICING.REMITTANCE_TRANSACTION_INITIATION_REQUESTS
NG_PRICING.STANDARD_CHARGES
NG_PRICING.REMITTANCE_TRANSACTION_CANCELLATION_REQUESTS
NG_PRICING.REMITTANCE_TRANSACTION_INITIATION_RESPONSES
NG_PRICING.AGENT_PRODUCT_ONBOARD
NG_PRICING.BANK_PRODUCT_ONBOARD
NG_PRICING.REMITTANCE_TRANSACTION_CANCELLATION_RESPONSES
NG_PRICING.AGENT_VALUE_DATE_WISES
NG_PRICING.STANDARD_AMOUNT_RANGE
NG_PRICING.AGENT_RATES

Remittance process

Remittance Initiation

Parameters in Remittance Input request for initiation.

Parameter	Mandatory	Possible Values
Agent Code	Y	Specific
Bank Code	Y	Specific
Originating Currency	Y	Specific
Destination Currency	Y	Specific
Service Provider	Y	Specific
Product	Y	Specific
Sub Product	Y	Specific
Service Type	Y	Specific
Beneficiary Type	Y	Specific
Customer Type	Y	Specific
Payment Mode	Y	Specific
Destination Country	Y	Specific
Originating Country	Y	Specific
Payin/Payout Amount	Y	Specific
InitiationDate	Y	Specific
TransmissionDate	Y	Specific
PayinCurrency	Y	Specific
PayoutCurrency	Y	Specific
settlementCurrency	Y	Specific
ServiceProviderCurrency	Y	Specific
PassSettlementCurrency	Y	Specific

Parameters in Customer Fees request

Parameter	Mandatory	Possible Values
Source (Agent ID, Country Code)	Y	Specific, All
Destination (Agent ID, Bank Code, Country Code)	Y	Specific, All
Service Provider	Y	Specific, All
Product	Y	Specific, All
Sub Product	Y	Specific, All
Service Type	Y	Specific, All

Beneficiary Type	Y	Specific, All
Customer Type	Y	Specific, All
Transaction Type	Y	Specific
Payment Mode	Y	Specific, All
Charge Type	Y	Specific
Amount Range	Y	Specific
From Currency	Y	Specific, All
To Currency	Y	Specific, All
ValueDatewise	Y	Specific
Date Range	O	Specific, Default

Priority for standard charge is calculated by below logic in next section.

After calculating the priority, system will calculate the rate as below formula:

- **If Payout Amount is available and Charge basis is Payout > Payin**

    a. Calculation Logic for Payin
        i. FC – LC

            Payin = (Payout * Rate)

        i. LC – FC

            Payin = (Payout/Rate)

    a. Check Payin amount against the ranges defined for the records filtered based on the above priority logic.
    b. Check Initiation Date against the Date ranges defined for the records filtered in above step. If it does not fall under any date range, then use the record without the date range.
    c. Based on the Charge Basis, Customer Fees (Additional Charges, Backend Charges, Card Charges, Commission, Other Charges and Tax) will be calculated. Customer Fees can be set in Absolute amount, Percentage or Both.
        i. Calculation Logic

    a. Absolute Amount – Absolute amount will be returned

Customer Fees = Absolute Amount

Percentage - Will be calculated based on the Payout amount.

Customer Fees = (Payin Amount * Percentage)/ 100

Both – Absolute amount alongwith the Percentage value calculated based on the Payout amount.

Customer Fees = Absolute Value + (Payin Amount * Percentage)/ 100

- **If Payout Amount is available and Charge basis is Payin > Payout**
    - Calculation Logic for Payin
        - FC – LC

            Payin = (Payout * Rate)

        - LC – FC

            Payin = (Payout/Rate)

    - Check Payin amount against the ranges defined for the records filtered based on the above priority logic.
    - Check Initiation Date against the Date ranges defined for the records filtered in above step. If it does not fall under any date range, then use the record without the date range.
    - Based on the Charge Basis, Customer Fees (Additional Charges, Backend Charges, Card Charges, Commission, Other Charges and Tax) will be calculated. Customer Fees can be set in Absolute amount, Percentage or Both.
        - Calculation Logic

Absolute Amount – Absolute amount will be returned

Customer Fees = Absolute Amount

Percentage - Will be calculated based on the Payin amount.

Customer Fees = (Payin Amount * Percentage)/ 100

Both – Absolute amount alongwith the Percentage value calculated based on the Payin amount.

Customer Fees = Absolute Value + (Payin Amount * Percentage)/ 100

Remittance Cancellation

Input request for Remittance Cancellation are:

Parameter	**Mandatory**	**Possible Values**
Transaction UUID	Y	Remittance Initiation ID
TransactionStatus	Y	CANCEL_BEFORE_PROVIDER, CANCEL_AFTER_PROVIDER
ReasonForCancellation	Y	CUSTOMER_WISH, AGENT_FAULT, SP_FAULT

Priority for standard charge is calculated by below logic in next section.

During cancellation, system fetch the RuleSettings from onboarding API for particular AgentCode.

API: ***/onboarding/api/v1/agents/{agentId}/branches/{agentBranchId}/chargesRules***

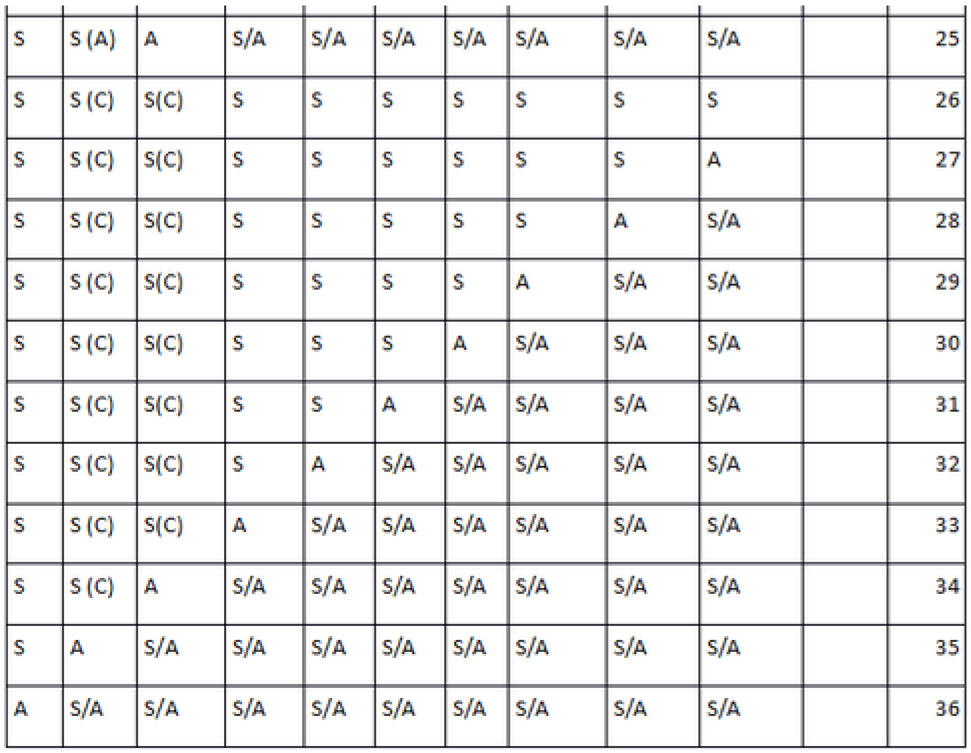
Based on input request (Product, SubProduct, ServiceType, ReasonForCancellation), it would get find the enabled ruleSetting, and return in response.

Priority Calculation

(for more details see: Pricing Rules v_7.4.docx)

During remittance process, standard charge will be calculated based on below priority table.

SP	Source	Destination	Product	Sub Product	Service Type	From/Paying Currency	To/Payout Currency	Customer Type	Beneficiary Type	Payment Mode	Priority
S	S (A)	S(B)	S	S	S	S	S	S	S	A	1
S	S (A)	S(B)	S	S	S	S	S	S	A	S	2
S	S (A)	S(B)	S	S	S	S	S	A	S/A		3
S	S (A)	S(B)	S	S	S	S	A	S/A	S/A		4
S	S (A)	S(B)	S	S	S	A	S/A	S/A	S/A		5
S	S (A)	S(B)	S	S	A	S/A	S/A	S/A	S/A		6
S	S (A)	S(B)	S	A	S/A	S/A	S/A	S/A	S/A		7
S	S (A)	S(B)	A	S/A	S/A	S/A	S/A	S/A	S/A		8
S	S (C)	S(B)	S	S	S	S	S	S	S		9
S	S (C)	S(B)	S	S	S	S	S	S	A		10
S	S (C)	S(B)	S	S	S	S	S	A	S/A		11
S	S (C)	S(B)	S	S	S	S	A	S/A	S/A		12
S	S (C)	S(B)	S	S	S	A	S/A	S/A	S/A		13
S	S (C)	S(B)	S	S	A	S/A	S/A	S/A	S/A		14
S	S (C)	S(B)	S	A	S/A	S/A	S/A	S/A	S/A		15
S	S (C)	S(B)	A	S/A	S/A	S/A	S/A	S/A	S/A		16
S	S (A)	S(C)	S	S	S	S	S	S	S		17
S	S (A)	S(C)	S	S	S	S	S	S	A		18
S	S (A)	S(C)	S	S	S	S	S	A	S/A		19
S	S (A)	S(C)	S	S	S	S	A	S/A	S/A		20
S	S (A)	S(C)	S	S	S	A	S/A	S/A	S/A		21
S	S (A)	S(C)	S	S	A	S/A	S/A	S/A	S/A		22
S	S (A)	S(C)	S	A	S/A	S/A	S/A	S/A	S/A		23
S	S (A)	S(C)	A	S/A	S/A	S/A	S/A	S/A	S/A		24



S	S (A)	A	S/A	S/A	S/A	S/A	S/A	S/A	S/A		25
S	S (C)	S(C)	S	S	S	S	S	S	S		26
S	S (C)	S(C)	S	S	S	S	S	S	A		27
S	S (C)	S(C)	S	S	S	S	S	A	S/A		28
S	S (C)	S(C)	S	S	S	S	A	S/A	S/A		29
S	S (C)	S(C)	S	S	S	A	S/A	S/A	S/A		30
S	S (C)	S(C)	S	S	A	S/A	S/A	S/A	S/A		31
S	S (C)	S(C)	S	A	S/A	S/A	S/A	S/A	S/A		32
S	S (C)	S(C)	A	S/A	S/A	S/A	S/A	S/A	S/A		33
S	S (C)	A	S/A	S/A	S/A	S/A	S/A	S/A	S/A		34
S	A	S/A	S/A	S/A	S/A	S/A	S/A	S/A	S/A		35
A	S/A	S/A	S/A	S/A	S/A	S/A	S/A	S/A	S/A		36

(A) = Agent Code

(B) = Bank Code

(C) = Country Code

S = Specific

A = All

Priority logic calculation (For technical user):

During remittance process, standard charge is picked on above priority logic. The logic is implemented in two steps to calculate the priority for better performance.

Step1: Find priority based on above table at DB level.

Except two fields (Source and Destination), all other fields are considered for priority calculation, and query is implemented to run on DB layer. It also includes enabled Amount Range records and discard all disabled records from priority calculation.

Step2: Output of step-1 is used to calculate the priority between Source and Destination. From Query, we would maximum 9 highest priority charges for each Charge-Type. Based on below table, system will find the single top most priority charge.

Source: Agent Code or Originating Country Code

Destination: Bank Code or Destination country code

Source	Destination	Priority
Agent Code	Bank Code	1
Country Code	Bank Code	2
Agent Code	Country Code	3
Country Code	Country Code	4
Agent Code	Specific/All	5
Specific/All	Specific/All	6

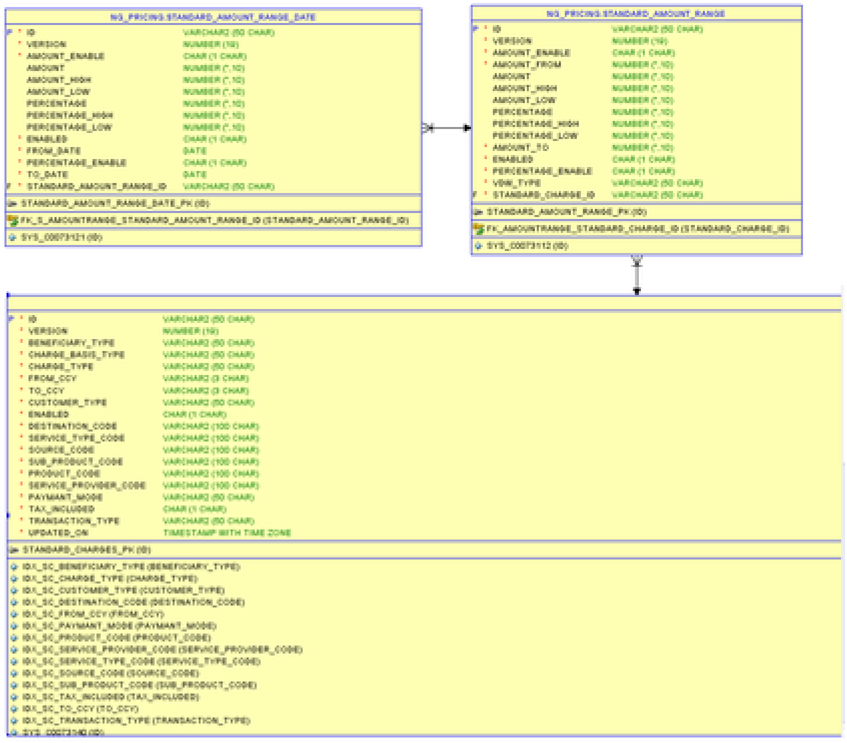
Customer Fee

Requirement Summary

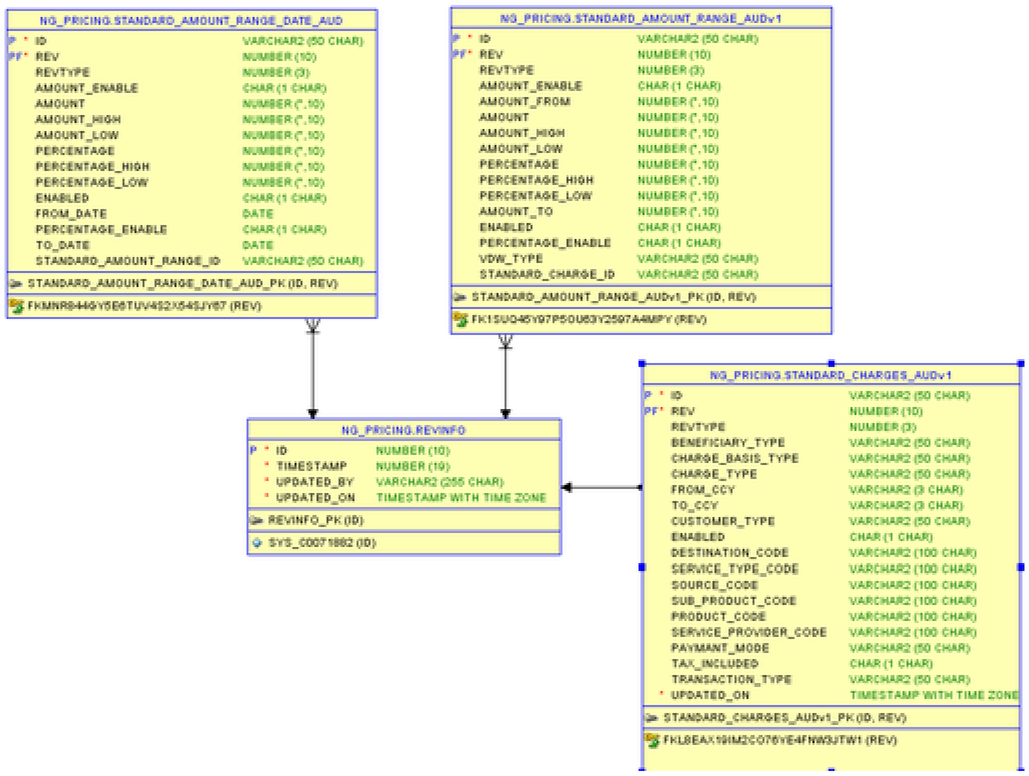
1. User is able to create/update standard charges from front end in PAAS.
2. User is able to create/update standard charges from amigo into PAAS.
3. Standard charge will be used during Remittance process during initiation and cancellation.
4. User is able to download the standard-charges from UI.
5. User is able to upload new standard-charges for given xls format from UI.

Database-Schema

ER-Diagram



NG_PRICING.STANDARD_AMOUNT_RANGE_DATE
ID VARCHAR2 (50 CHAR)
VERSION NUMBER (19)
AMOUNT_ENABLE CHAR (1 CHAR)
AMOUNT NUMBER (*,10)
AMOUNT_HIGH NUMBER (*,10)
AMOUNT_LOW NUMBER (*,10)
PERCENTAGE NUMBER (*,10)
PERCENTAGE_HIGH NUMBER (*,10)
PERCENTAGE_LOW NUMBER (*,10)
ENABLED CHAR (1 CHAR)
FROM_DATE DATE
PERCENTAGE_ENABLE CHAR (1 CHAR)
TO_DATE DATE
STANDARD_AMOUNT_RANGE_ID VARCHAR2 (50 CHAR)
STANDARD_AMOUNT_RANGE_DATE_PK (ID)
FK_S_AMOUNTRANGE_STANDARD_AMOUNT_RANGE_ID (STANDARD_AMOUNT_RANGE_ID)
SYS_C0073121 (ID)
NG_PRICING.STANDARD_AMOUNT_RANGE
ID VARCHAR2 (50 CHAR)
VERSION NUMBER (19)
AMOUNT_ENABLE CHAR (1 CHAR)
AMOUNT_FROM NUMBER (*,10)
AMOUNT NUMBER (*,10)
AMOUNT_HIGH NUMBER (*,10)
AMOUNT_LOW NUMBER (*,10)
PERCENTAGE NUMBER (*,10)
PERCENTAGE_HIGH NUMBER (*,10)
PERCENTAGE_LOW NUMBER (*,10)
AMOUNT_TO NUMBER (*,10)
ENABLED CHAR (1 CHAR)
PERCENTAGE_ENABLE CHAR (1 CHAR)
VDW_TYPE VARCHAR2 (50 CHAR)
STANDARD_CHARGE_ID VARCHAR2 (50 CHAR)
STANDARD_AMOUNT_RANGE_PK (ID)
FK_AMOUNTRANGE_STANDARD_CHARGE_ID (STANDARD_CHARGE_ID)
SYS_C0073112 (ID)
ID VARCHAR2 (50 CHAR)
VERSION NUMBER (19)
BENEFICIARY_TYPE VARCHAR2 (50 CHAR)
CHARGE_BASIS_TYPE VARCHAR2 (50 CHAR)
CHARGE_TYPE VARCHAR2 (50 CHAR)
FROM_CCY VARCHAR2 (3 CHAR)
TO_CCY VARCHAR2 (3 CHAR)
CUSTOMER_TYPE VARCHAR2 (50 CHAR)
ENABLED CHAR (1 CHAR)
DESTINATION_CODE VARCHAR2 (100 CHAR)
SERVICE_TYPE_CODE VARCHAR2 (100 CHAR)
SOURCE_CODE VARCHAR2 (100 CHAR)
SUB_PRODUCT_CODE VARCHAR2 (100 CHAR)
PRODUCT_CODE VARCHAR2 (100 CHAR)
SERVICE_PROVIDER_CODE VARCHAR2 (100 CHAR)
PAYMANT_MODE VARCHAR2 (50 CHAR)
TAX_INCLUDED CHAR (1 CHAR)
TRANSACTION_TYPE VARCHAR2 (50 CHAR)
UPDATED_ON TIMESTAMP WITH TIME ZONE
STANDARD_CHARGES_PK (ID)
IDX_SC_BENEFICIARY_TYPE (BENEFICIARY_TYPE)
IDX_SC_CHARGE_TYPE (CHARGE_TYPE)
IDX_SC_CUSTOMER_TYPE (CUSTOMER_TYPE)
IDX_SC_DESTINATION_CODE (DESTINATION_CODE)
IDX_SC_FROM_CCY (FROM_CCY)
IDX_SC_PAYMANT_MODE (PAYMANT_MODE)
IDX_SC_PRODUCT_CODE (PRODUCT_CODE)
IDX_SC_SERVICE_PROVIDER_CODE (SERVICE_PROVIDER_CODE)
IDX_SC_SERVICE_TYPE_CODE (SERVICE_TYPE_CODE)
IDX_SC_SOURCE_CODE (SOURCE_CODE)
IDX_SC_SUB_PRODUCT_CODE (SUB_PRODUCT_CODE)
IDX_SC_TAX_INCLUDED (TAX_INCLUDED)
IDX_SC_TO_CCY (TO_CCY)
IDX_SC_TRANSACTION_TYPE (TRANSACTION_TYPE)
SYS_C0073140 (ID)


NG_PRICING.STANDARD_AMOUNT_RANGE_DATE_AUD
STANDARD_AMOUNT_RANGE_DATE_AUD_PK (ID, REV)
NG_PRICING.STANDARD_AMOUNT_RANGE_AUDv1
STANDARD_AMOUNT_RANGE_AUDv1_PK (ID, REV)
NG_PRICING.REVINFO
REVINFO_PK (ID)
NG_PRICING.STANDARD_CHARGES_AUDv1
STANDARD_CHARGES_AUDv1_PK (ID, REV)

Tables Name:

STANDARD_CHARGES

STANDARD_CHARGES_AUD

STANDARD_AMOUNT_RANGE

STANDARD_AMOUNT_RANGE_AUD

STANDARD_AMOUNT_RANGE_DATE

STANDARD_AMOUNT_RANGE_DATE_AUD

Table: STANDARD_CHARGES

Unique composite key (SOURCE_CODE + DESTINATION_CODE+ SERVICE_PROVIDER_CODE + PRODUCT_CODE + SUB_PRODUCT_CODE + SERVICE_TYPE_CODE + TRANSACTION_TYPE + BENEFICIARY_TYPE + CUSTOMER_TYPE + CHARGE_TYPE + CHARGE_BASIS_TYPE + PAYMANT_MODE + FROM_CCY + TO_CCY**)**

Sr. No.	Field Name	Type	Source of data	Comment
1	SOURCE_CODE	Drop-down	MasterData	Agent-code, and Originating-country
2	DESTINATION_CODE	Drop-down	MasterData	Bank-code and destination-country
3	SERVICE_PROVIDER_CODE	Drop-down	MasterData	e.g.- UAEXMUK#####
4	PRODUCT_CODE	Drop-down	MasterData	
5	SUB_PRODUCT_CODE	Drop-down	MasterData	
6	SERVICE_TYPE_CODE	Drop-down	MasterData	
7	TRANSACTION_TYPE	Drop-down	ENUM	

8	BENEFICIARY_TYPE	Drop-down	ENUM	
9	CUSTOMER_TYPE	Drop-down	ENUM	
10	CHARGE_TYPE	Drop-down	ENUM	
11	CHARGE_BASIS_TYPE	Drop-down	ENUM	
12	PAYMANT_MODE	Drop-down	ENUM	
13	FROM_CCY	Drop-down	MasterData	
14	TO_CCY	Drop-down	MasterData	
15	TAX_INCLUDED	Yes/No		
16	UPDATED_ON	DATE		

Table: STANDARD_AMOUNT_RANGE

Sr. No.	Field Name	Type	Source of data	Comment
1	AMOUNT	Text		
2	AMOUNT_LOW	Text		
3	AMOUNT_HIGH	Text		
4	AMOUNT_ENABLE	YES/NO		
5	PERCENTAGE	Text		
6	PERCENTAGE_LOW	Text		
7	PERCENTAGE_HIGH	Text		
8	PERCENTAGE_ENABLE	YES/NO		
9	AMOUNT_FROM	Text		
10	AMOUNT_TO	Text		
11	VDW_TYPE	Text		CASE/ FUTURE/ SPOT/ TOM
12	ENABLED	Yes/No		
13	STANDARD_CHARGE_ID	Text		Foreign key with STANDARD_CHARGES

Table: STANDARD_AMOUNT_RANGE_DATE

Sr. No.	Field Name	Type	Source of data	Comment
1	AMOUNT	Text		
2	AMOUNT_LOW	Text		
3	AMOUNT_HIGH	Text		
4	PERCENTAGE	Text		
5	PERCENTAGE_LOW	Text		
6	PERCENTAGE_HIGH	Text		
7	FROM_DATE	Date		
8	TO_DATE	Date		
9	ENABLED	Yes/No		
10	STANDARD_AMOUNT_RANGE_ID	Text		Foreign key with STANDARD_AMOUNT_RANGE

Available Endpoints

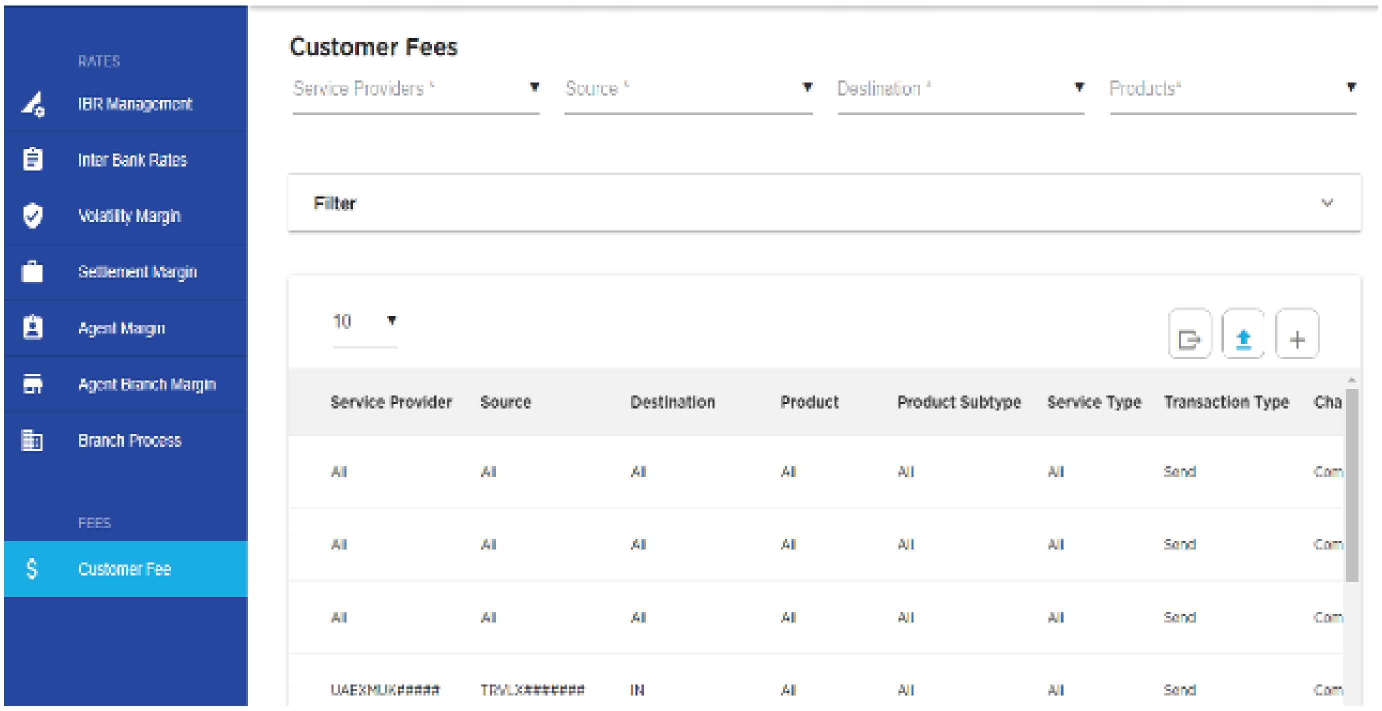
Type	Endpoint	Description
GET	/api/v1/standard-charges	Gets a page of standard charges records matching the selection filters and sort criteria
POST	/api/v1/standard-charges	Creates one or multiple new Standard charges records
PATCH	/api/v1/standard-charges/{scId}/amount-range	Create amount range for given Standard charges
PATCH	/api/v1/standard-charges/{scId}/amount-range/update	Updates one or multiple amount range for particular standard charges
POST	/api/v1/standard-charges/{scId}/amount-range/validate	validate one or multiple amount range for particular standard charges
PATCH	/api/v1/standard-charges/amount-range/status	Enable or disable records at amountRange or amountRange date wise
GET	/api/v1/standard-charges/download	Download all Standard Charges matching the selection filters and sort criteria
POST	/api/v1/standard-charges/upload	upload standard-charge from excel sheet
POST	/api/v1/standard-charges/validate	Validate unique Standard charge
POST	/api/v1/amigo/standardCharges	Create or Update Standard charge record in amigo system

Constraints/Dependencies

1. User must be login to create/update standard charges
2. Master Data from onboarding API must be available. (for Source, Destination, Service-Provider, Product, Sub-Product, Service-Type)
3. There is not depedency on previous layer data. It can be created standalone.

User Interface

UI - Search

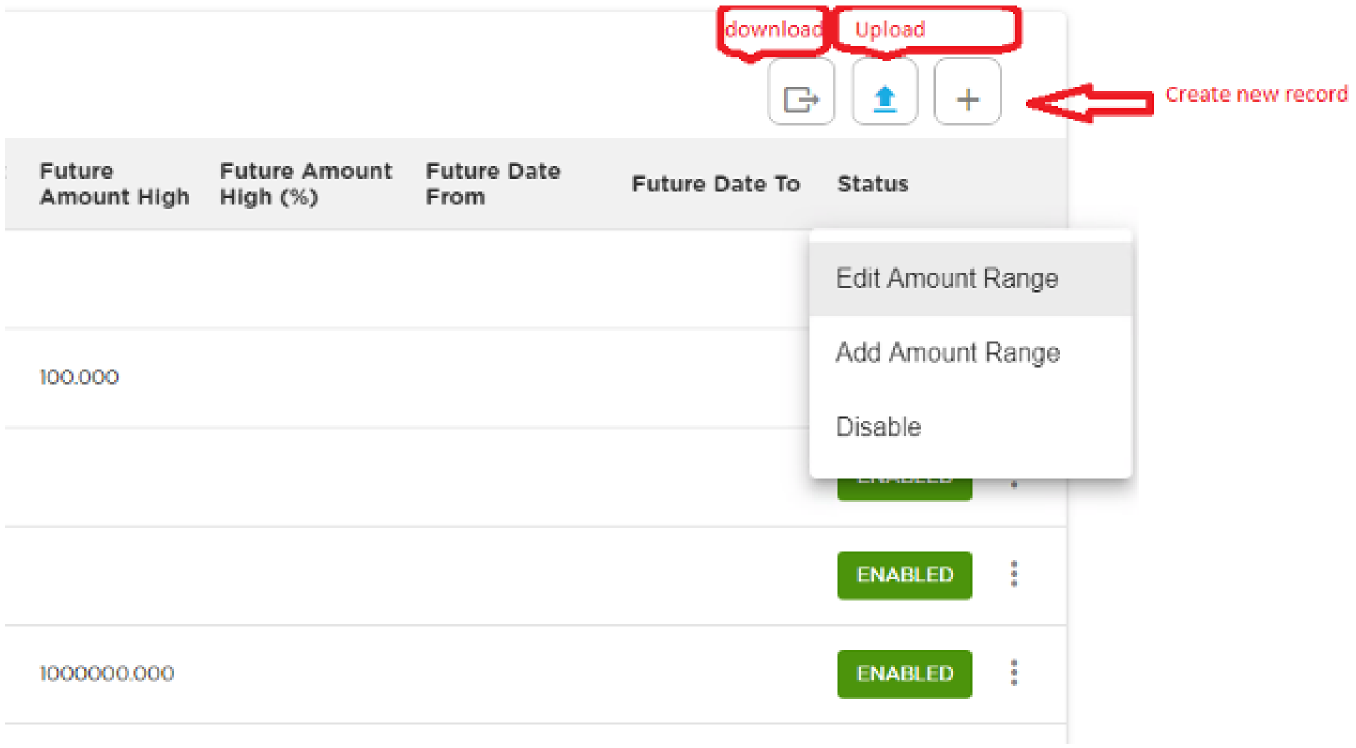


Service Provider	Source	Destination	Product	Product Subtype	Service Type	Transaction Type	Cha
All	All	All	All	All	All	Send	Com
All	All	All	All	All	All	Send	Com
All	All	All	All	All	All	Send	Com
UAEXMUK#####	TRVLX########	IN	All	All	All	Send	Com

At search UI, a user can perform following operations:

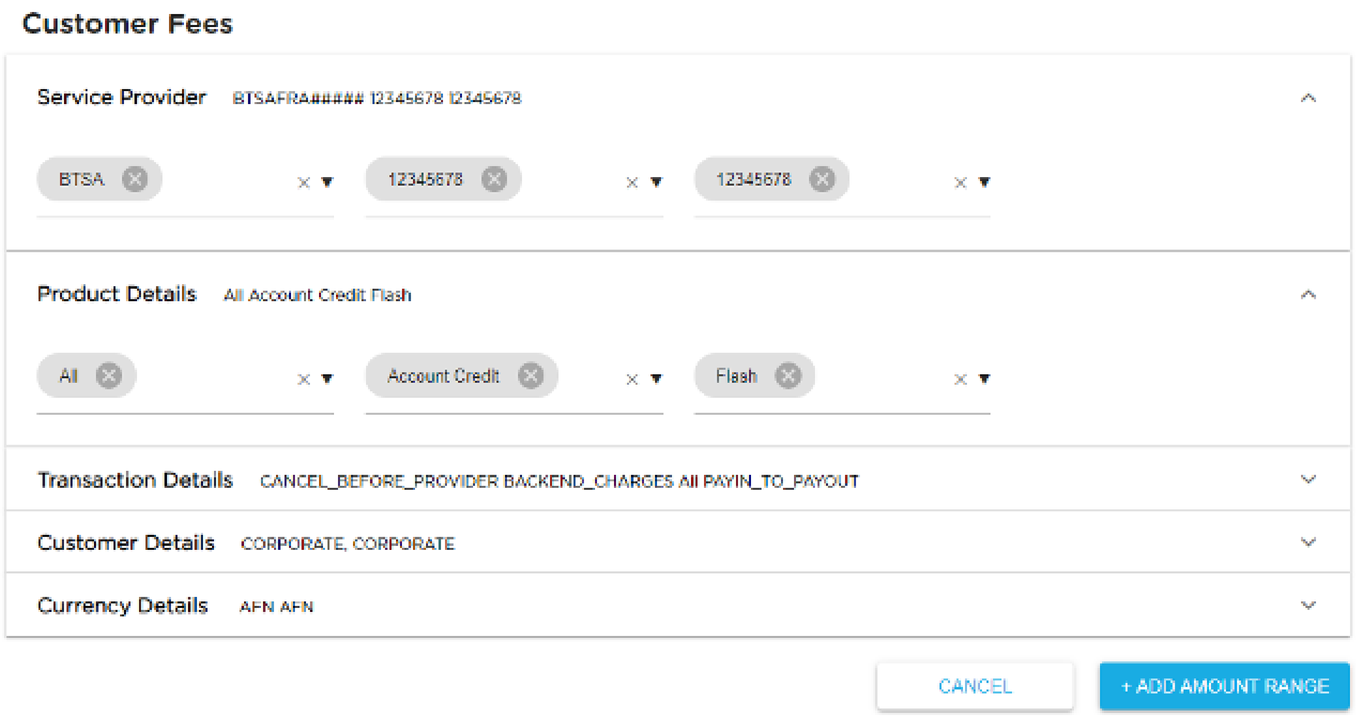
- Search by using search filter

- Upload new standard-charges
- Download standard-charges
- Create new standard-charge
- Update/Edit standard-charge
- Enable/disable record at amount-range or data-range level.



Future Amount High	Future Amount High (%)	Future Date From	Future Date To	Status
100.000				
				ENABLED
				ENABLED
1000000.000				ENABLED

UI-Create



Customer Fees

Service Provider BTSAFRA##### 12345678 12345678

BTSA | 12345678 | 12345678

Product Details All Account Credit Flash

All | Account Credit | Flash

Transaction Details CANCEL_BEFORE_PROVIDER BACKEND_CHARGES All PAYIN_TO_PAYOUT

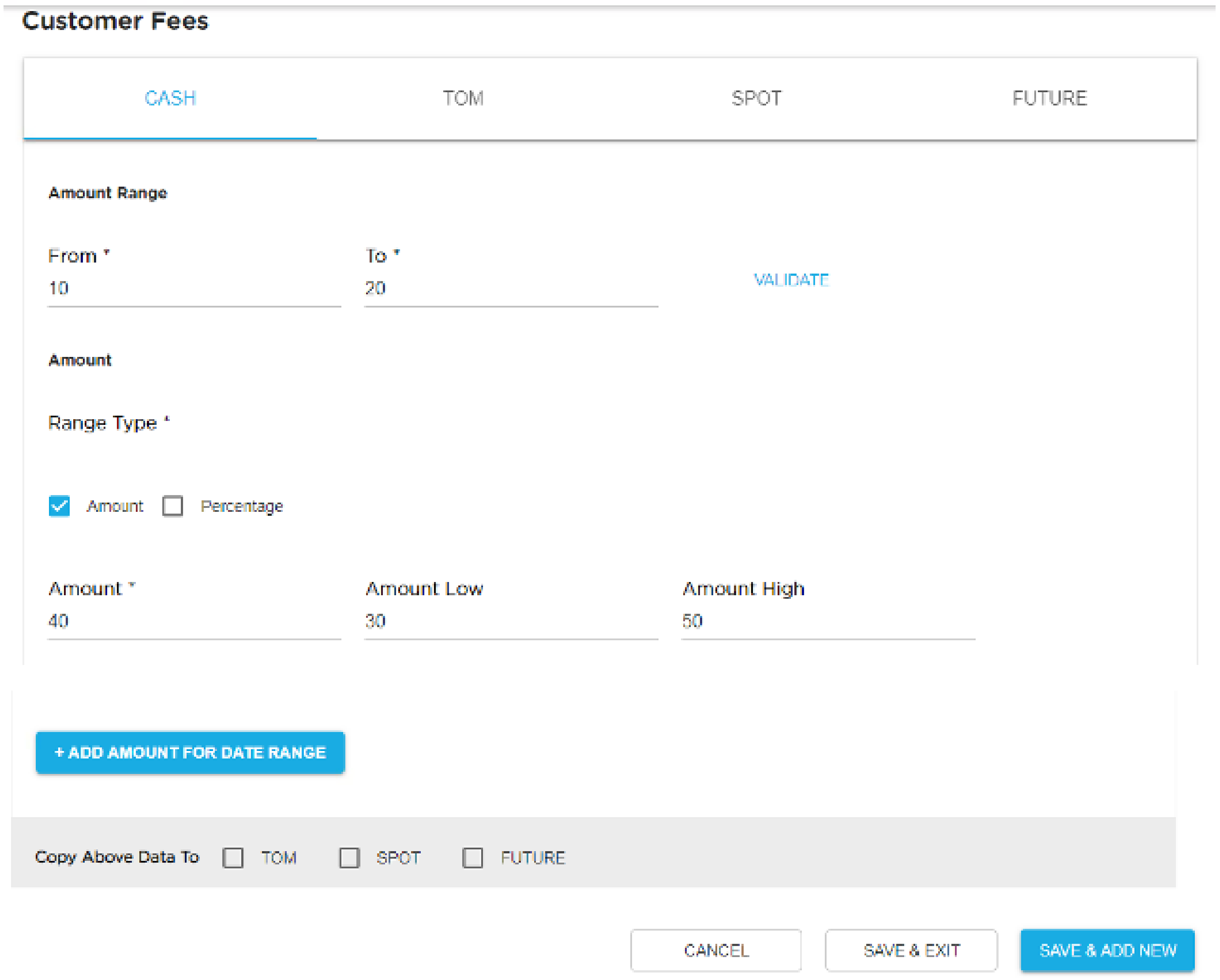
Customer Details CORPORATE, CORPORATE

Currency Details AFN AFN

All fields are mandatory except Tax_Included.

User is able to click on “ADD AMOUNT RANGE”, when all fields are selected.

User must select multiple values for Service Provider, Source_code, Destination_code, Product, Sub_product, Service_type, while single values for other drop-down.



Customer Fees

CASH | TOM | SPOT | FUTURE

Amount Range

From * 10

To * 20

VALIDATE

Amount

Range Type *

[✓ Amount] ☐ Percentage

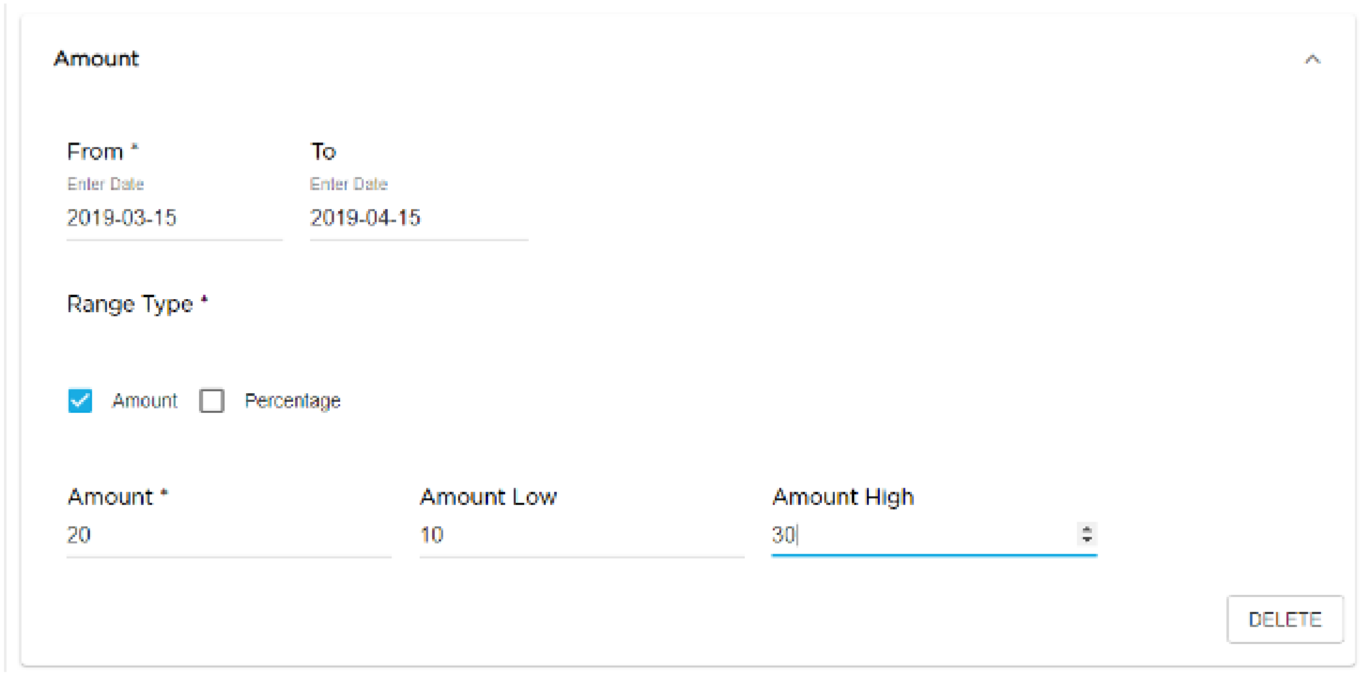
Amount * 40

Amount Low 30

Amount High 50

+ ADD AMOUNT FOR DATE RANGE
Copy Above Data To ☐ TOM ☐ SPOT ☐ FUTURE
CANCEL
SAVE & EXIT
SAVE & ADD NEW

User can add Date Range by clicking “ADD AMOUNT FOR DATE RANGE” button.



Amount

From *
Enter Date
2019-03-15

To
Enter Date
2019-04-15

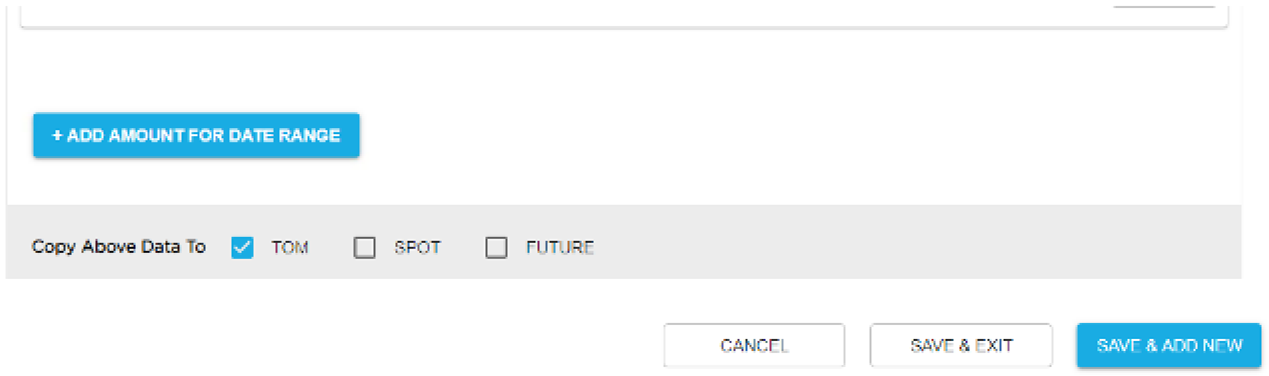
Range Type *

[✓ Amount] Percentage

Amount *
20

Amount Low
10

Amount High
30



Copy Above Data To [✓ TOM] SPOT FUTURE

User can copy the same data for other VDW-Type (TOM, SPOT, FUTURE), or enter new values on corresponding tab. And click on “SAVE & ADD NEW”.

Data will be saved, if it doesn't fail with Unique Key constraint.

On Click of Save- Same standard charge will be pushed to Amigo by Active MQ messaging service.

Create/Update from Amigo(syncing)

Amigo and Pricing engine must be in sync for available standard-charges. If there is any change in Amigo system for particular standard-charge, same data will be pushed to Pricing engine by calling rest API.

From Pricing-engine to Amigo:

Any change in standard-charges in pricing-engine, system push the data to Amigo using Active-MQ.

Queue Name: PAASRateChanged

Type: STANDARD_CHARGES

From Amigo to Pricing-engine

To push new or update of standard-charges from amigo into PAAS, amigo use Rest API as below:

Rest API: api/v1/amigo/standardCharges

Branch Process

Requirement Summary

1. User is able to create/update Special deal (MULTIPLE or SINGLE Type) from Amigo and Global Amigo in PAAS.
2. User is able to offer rate from PAAS.
3. Offered rate must be pushed to concerned partners (Amigo/Global Amigo).
4. User must be able to Accept or Reject the deals created by Amigo.
5. Paas User can see the active deals on UI (created on today).
6. Amigo team must able to consumed amount partially (for multiple deal), or fully.
7. For Qatar exchange Service provider, if deal is created from Qatar, deal must be pushed to Global Amigo. Where it can be accepted or rejected based on the business requirement. And the same update must be pushed to Paas at every transaction.

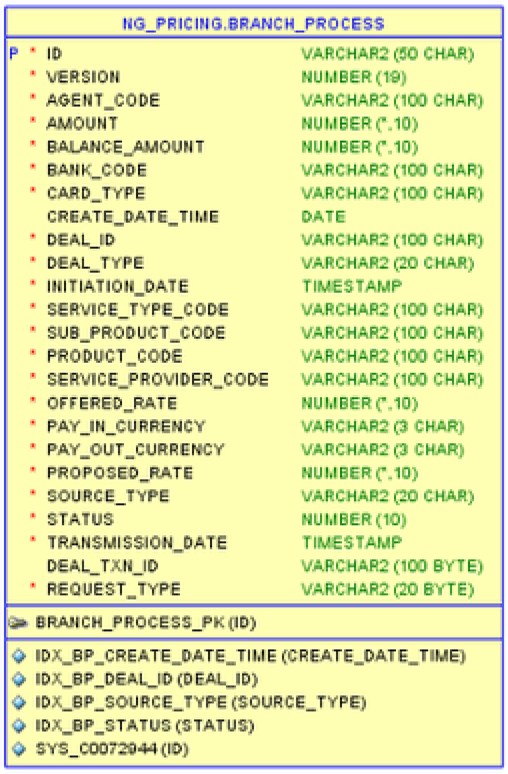
Database-Schema

Tables Name:

BRANCH_PROCESS

Sr. No.	Field Name	Type	Source of data	Comment
1	ID	Text		Generated Unique ID
2	VERSION	Number		
3	AGENT_CODE	Text	Amigo i/p request	
4	AMOUNT	BigDecimal	Amigo i/p request	
5	BALANCE_AMOUNT	BigDecimal	Amigo i/p request	
6	BANK_CODE	Text	Amigo i/p request	
7	CARD_TYPE	Text	Amigo i/p request	
8	CREATE_DATE_TIME	Date	System today date	
9	DEAL_ID	Text	Amigo i/p request	
10	DEAL_TYPE	Text	Amigo i/p request	
11	INITIATION_DATE	Date	Amigo i/p request	
12	SERVICE_TYPE_CODE	Text	Amigo i/p request	
13	PRODUCT_CODE	Text	Amigo i/p request	
14	SUB_PRODUCT_CODE	Text	Amigo i/p request	
15	SERVICE_PROVIDER_CODE	Text	Amigo i/p request	
16	OFFERED_RATE	BigDecimal	Input from Paas	
17	PAYIN_CURRENCY	Text	Amigo i/p request	
18	PAYOUT_CURRENCY	Text	Amigo i/p request	
19	PROPOSED_RATE	BigDecimal	Amigo i/p request	
20	SOURCE_TYPE	Text	Amigo i/p request	
21	STATUS	Yes/No	Amigo i/p request	
22	TRANSACTION_DATE	Date	Amigo i/p request	
23	DEAL_TXN_ID	Text		Generated only for partially implemented , multiple type deal internally.
24	REQUEST_TYPE	Text	Amigo i/p request	

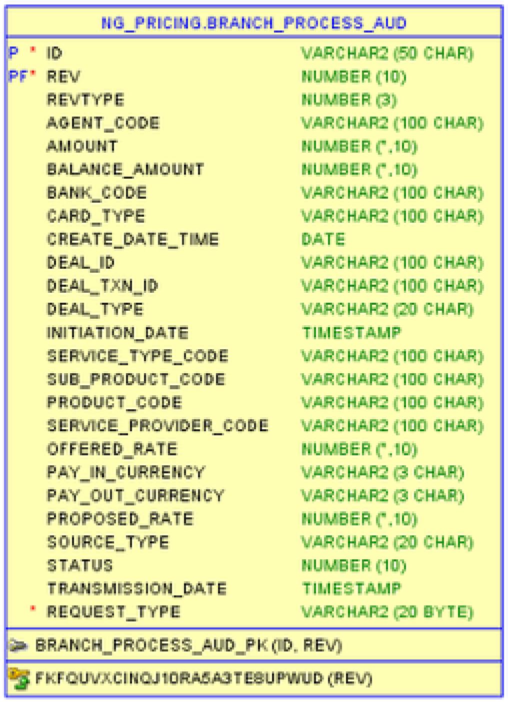
ER-Diagram: Branch-Process



NG_PRICING.BRANCH_PROCESS			
P	*	ID	VARCHAR2 (50 CHAR)
	*	VERSION	NUMBER (19)
	*	AGENT_CODE	VARCHAR2 (100 CHAR)
	*	AMOUNT	NUMBER (*,10)
	*	BALANCE_AMOUNT	NUMBER (*,10)
	*	BANK_CODE	VARCHAR2 (100 CHAR)
	*	CARD_TYPE	VARCHAR2 (100 CHAR)
		CREATE_DATE_TIME	DATE
	*	DEAL_ID	VARCHAR2 (100 CHAR)
	*	DEAL_TYPE	VARCHAR2 (20 CHAR)
	*	INITIATION_DATE	TIMESTAMP
	*	SERVICE_TYPE_CODE	VARCHAR2 (100 CHAR)
	*	SUB_PRODUCT_CODE	VARCHAR2 (100 CHAR)
	*	PRODUCT_CODE	VARCHAR2 (100 CHAR)
	*	SERVICE_PROVIDER_CODE	VARCHAR2 (100 CHAR)
	*	OFFERED_RATE	NUMBER (*,10)
	*	PAY_IN_CURRENCY	VARCHAR2 (3 CHAR)
	*	PAY_OUT_CURRENCY	VARCHAR2 (3 CHAR)
	*	PROPOSED_RATE	NUMBER (*,10)
	*	SOURCE_TYPE	VARCHAR2 (20 CHAR)
	*	STATUS	NUMBER (10)
	*	TRANSMISSION_DATE	TIMESTAMP
		DEAL_TXN_ID	VARCHAR2 (100 BYTE)
	*	REQUEST_TYPE	VARCHAR2 (20 BYTE)

BRANCH_PROCESS_PK (ID)

IDX_BP_CREATE_DATE_TIME (CREATE_DATE_TIME)
IDX_BP_DEAL_ID (DEAL_ID)
IDX_BP_SOURCE_TYPE (SOURCE_TYPE)
IDX_BP_STATUS (STATUS)
SYS_C0072944 (ID)



NG_PRICING.BRANCH_PROCESS_AUD			
P	*	ID	VARCHAR2 (50 CHAR)
PF	*	REV	NUMBER (10)
		REVTYPE	NUMBER (3)
		AGENT_CODE	VARCHAR2 (100 CHAR)
		AMOUNT	NUMBER (*,10)
		BALANCE_AMOUNT	NUMBER (*,10)
		BANK_CODE	VARCHAR2 (100 CHAR)
		CARD_TYPE	VARCHAR2 (100 CHAR)
		CREATE_DATE_TIME	DATE
		DEAL_ID	VARCHAR2 (100 CHAR)
		DEAL_TXN_ID	VARCHAR2 (100 CHAR)
		DEAL_TYPE	VARCHAR2 (20 CHAR)
		INITIATION_DATE	TIMESTAMP
		SERVICE_TYPE_CODE	VARCHAR2 (100 CHAR)
		SUB_PRODUCT_CODE	VARCHAR2 (100 CHAR)
		PRODUCT_CODE	VARCHAR2 (100 CHAR)
		SERVICE_PROVIDER_CODE	VARCHAR2 (100 CHAR)
		OFFERED_RATE	NUMBER (*,10)
		PAY_IN_CURRENCY	VARCHAR2 (3 CHAR)
		PAY_OUT_CURRENCY	VARCHAR2 (3 CHAR)
		PROPOSED_RATE	NUMBER (*,10)
		SOURCE_TYPE	VARCHAR2 (20 CHAR)
		STATUS	NUMBER (10)
		TRANSMISSION_DATE	TIMESTAMP
	*	REQUEST_TYPE	VARCHAR2 (20 BYTE)

BRANCH_PROCESS_AUD_PK (ID, REV)

FKFQUVXCINQJ10RA5A3TE8UPWUD (REV)

Available Endpoints

Type	Endpoint	Description
GET	/api/v1/branchProcess	Get all active branch process records
PATCH	/api/v1/branchProcess/{bpId}/status	Update deal record status-Call from PAAS UI
POST	/api/v1/branchProcess/compareRates	Update deal offered rate
POST	/api/v1/branchProcess/create	Creates deal record from amigo
GET	/api/v1/branchProcess/getDealDetail	Gets deal data and returns current status and offered rate
GET	/api/v1/branchProcess/getRates	Get rates for selected deal
PATCH	/api/v1/branchProcess/update	Update deal record
PATCH	/api/v1/branchProcess/update-status	Update deal record status-Request from Amigo
GET	/api/v1/branchProcessdownload	Download all listed deal records from UI

Type	Endpoint	Description
POST	/Paas_Integration /specialDeal/updateDealOfferedRate	Update deal offered rate at Global Amigo
POST	/Paas_Integration/specialDeal/insertSpecialDealTxnData	Push deal to Global amigo servier for UAE service provider
POST	/Paas_Integration/ specialDeal/updateDealStatus	Update deal status at Global Amigo
POST	/Paas_Integration /specialDeal/ updateDealOfferedRate	Update deal offered rate at Qatar Amigo

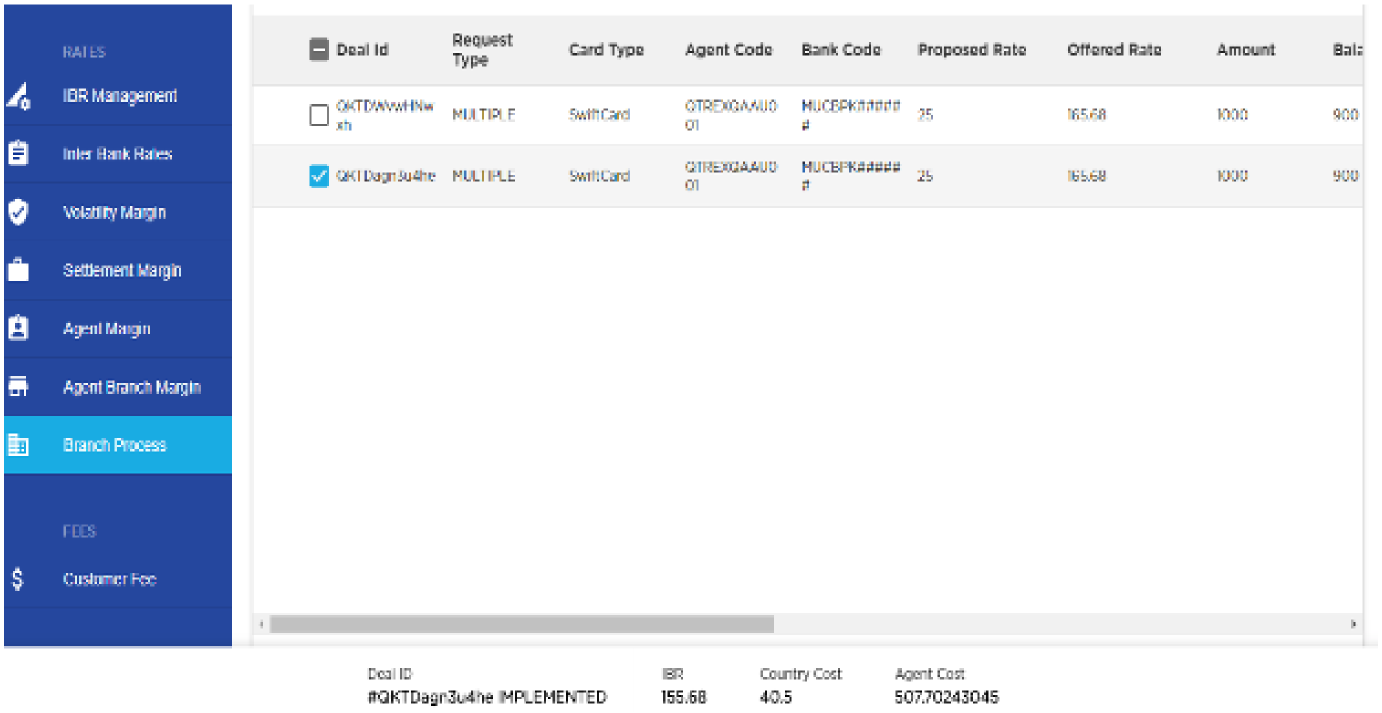
POST	/Paas_Integration/ specialDeal/updateDealStatus	Update deal status at Qatar Amigo

Constraints/Dependencies

1. Deal is landed from either Global Amigo or Qatar Amigo.
2. Paas will never create the deal form UI.
3. In Paas, user can only give offered rate for active deal, and set the status.
4. Amigo APIs must be availale to pull and push the deal.
5. If deal is partially consumed or not consumed, deal must be set to expired at the midnight for remaining amount.
6. Agent Margin, Country Margin rate must be available in corresponding layer for given Bank_code, PAYIN_CURRENCY, PAYOUT_CURRENCY and Agent code. Else Pass user will not able to make any offer and will get exception on UI.
7. Search UI will be refreshed every 30 seconds to get all updated deals.

User Interface-Branch Process

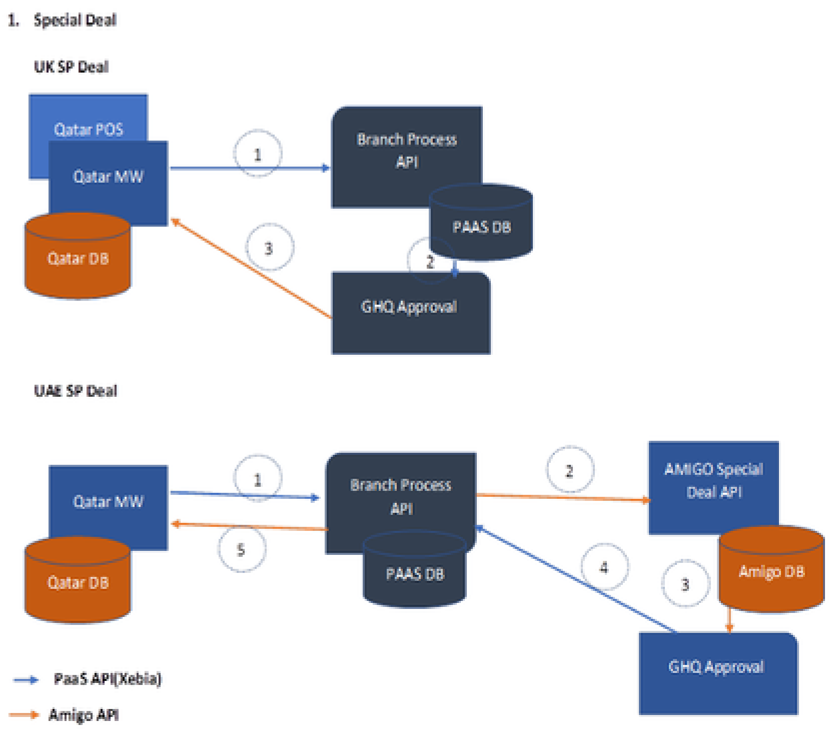
Search UI



Deal Id	Request Type	Card Type	Agent Code	Bank Code	Proposed Rate	Offered Rate	Amount	Bal
QKTDWvwHNwxh	MULTIPLE	SwiftCard	QTREXQAAU001	MUCBPK######	25	165.68	1000	900
✓ QKTDagn3u4he	MULTIPLE	SwiftCard	QTREXQAAU001	MUCBPK######	25	165.68	1000	900

Deal ID	IBR	Country Cost	Agent Cost
#QKTDagn3u4he IMPLEMENTED	155.68	40.5	507.70243045

1. On serach User, can select particular deal to make an offer-rate at bottom.
2. For selected deal, system will fetch the IBR rate, Country cost and agent cost from previous layer.
3. Any update on UI, will be pushed to partner with appropriate Status.



Deals can be raised from Qatar AMIGO (as source) and Global Amigo (as source) for UK service provider that will land to nGAP PaaS and the process would be as follows:

1. Deal initiated from Qatar Amigo or Global Amigo for UK service provider
2. Deals landed in nGAP PaaS; Branch process screen
3. Deals approved/rejected by GHQ user in branch process in nGAP PaaS
4. Deal push back to Qatar or Global Amigo based on source received with the updated status.

Deals can be raised from Qatar AMIGO (as source) for UAE service provider that will land to nGAP PaaS and the process would be as follows.

1. Deal initiated from Qatar Amigo for UAE service provider
2. Deals landed in nGAP PaaS; Branch process screen
3. Deal pushed to Global Amigo
4. Deals approved/rejected by GHQ user in branch process in Global Amigo system
5. Deal push back to nGAP PaaS and from there to Qatar Amigo with the updated status.

Deal creation from Amigo/Qatar

Input request JSON format

```
{
  "agentCode": "PXTVXUKWR001 ",
  "amount": 100,
  "balanceAmount": 100,
  "bankCode": "ASDFG",
  "cardType": "Credit",
  "costRate": "0.00",
  "dealId": "D1000",
  "dealStatus": "REQUEST_RECEIVED",
  "dealType": "GHQ Treasury",
  "ghqofferedRate": "0.00",
  "indicativeRate": "0.00",
  "inititaionDate": "03/29/2019 12:08:24
PM",
  "instrumentType": "XR",
  "isSpecialStatus": "0",
  "payInCcy": "USD",
  "payOutCcy": "INR",
  "requestType": "SINGLE",
  "rate": "30",
  "ratePublishInfo": "0",
  "serviceProviderCode": "UAEEXAE#####",
  "status": "0",
  "transmissionDate": "03/29/2019 12:00:00
AM",
  "sourceType": "AMIGOG"
}
```