Solution Architecture

# Objectives

This project aims to develop the “formal sector” features requested by institutions in charge of developing health financing schemes in NEPAL, in particular the management of enrollment of the insuree as part of an insurance group

* Manage  group insurance
  + UC1 - The employer wants to enroll an employee with the start dates and the calculation parameters for the policy values.
  + UC2 - The employer wants to extend an employee’s policy entitlement (contract extension).
  + UC3 - The employer wants to stop an employee’s policy entitlement (end of contract or change of calculation parameters for the policy value).
  + UC4 - The scheme administrator wants the employer to pay the contributions related to their employees covered for a given period of time
  + UC5 - the scheme administrator wants to update the status of the contributions due by an employer.
  + UC6 - The scheme administrator wants the policy to be active if the employer paid all contributions before the grace period
  + UC7 - The scheme administrator wants to add a new employer.
* Contribution Plan and Contribution Plans Bundle to manage the Policy value
  + UC8 - The scheme administrator wants to add a Contribution plan (pricing rules linked to an insurance product)
  + UC9 - the scheme administrator wants to be able to configure the calculation parameters in the Contribution plan

# Concepts

## Calculations

The rules for the formal sector can be complex and evolving therefore the “formal sector” solution should bring flexibility to avoid complete redesign and databases change when a parameter to define the contribution value is changed.

To bring that flexibility, the solution described in that document use a “calculation module”. This module will create a framework to design calculations and to manage their activation and version on openIMIS. The calculation rule themselves will be other module dependent of the “Calculation” module. This solution will deliver a calculation module that takes the income (on the level of the contribution collection details) and a rate (on the level of the contribution plan) as parameters.

The value of the parameters will be saved as a JSON string in the json\_ext field.

## Contribution plan and contribution plan bundle

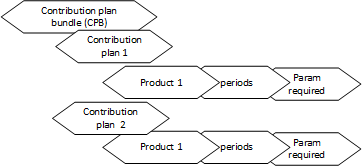
The pricing for the same benefit plan may differ from a sector to another; the «Contribution plan» are to define different cost/pricing rules and the payment periodicity (in month) for a given Benefit plan.

A benefit package (product in IMIS) can have multiple Contribution plans attached meaning the insuree may have different contributions level (different calculation) but contribute to the same pool. The Contribution plans will have parameters such as maximum payment delay, grace periods.

A given company may have to subscribe to multiple benefit plans for each employee (basic health insurance, advanced health insurance, accident\*, retirement\*); The “Contribution plans bundle” groups “Contribution plans” to ease the assignation of multiple Contribution plans to an Insuree/employee. The Contribution plans part of the same bundle must have the same periodicity.

The start date of the Contract will be on the 1st day of the month.

The Contribution plans and Contribution plans bundle might also be usable by the direct insurance/informal sector, but some code change is likely and not covered by this solution



## Policy holders\* and policy holder insurees

Each policy holder (employer) will be registered in openIMIS.

Each policy holder will have access to some schemes

The policy holder would have the possibility to add insuree with a contract period, default pricing parameters (e.g. income) and the contribution plan bundle to use.

* + PolicyHolderContributionPlan >> is it ok with

#### 

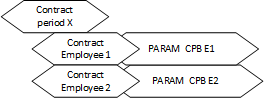
## Contract

Contribution collection is the contract that binds the employer to pay the contributions and binds the insurance to provide coverage for the agreed benefit packages.

Before every period, a contribution collection (list of contributions) will be generated based on the policy holder insuree information provided by the policy holder (default param, contract dates, CPB).

(Optional) Once generated the policy holder could update the contribution collection (refresh list of employees and change insuree parameters).

Once submitted by the policy holder and approved by scheme admin the contribution will be created and be eligible for payment, a trusted user could create directly an approved contribution collection through the API.



## Payment and policy update

Policy holder will make a single payment per contribution collection (list of contributions of his insuree)

The payment will be linked to all the contributions which are part of the contribution collection.

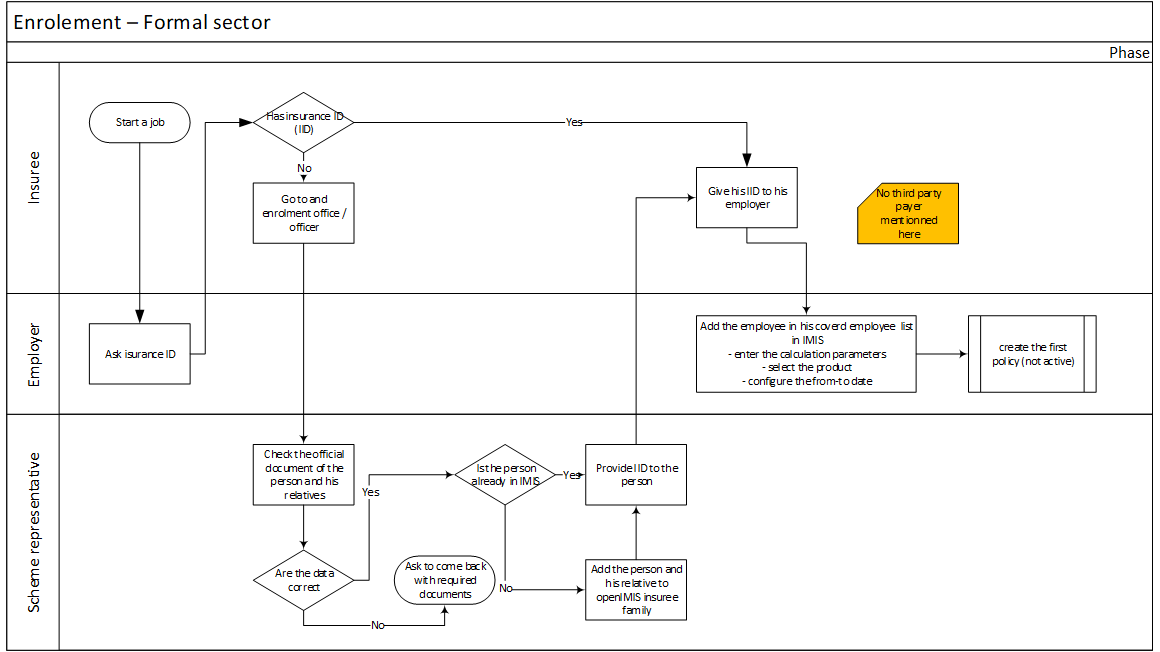
Depending on the configuration of openIMIS, the policy validity will be adapted based on the regime and benefit plan rules (i.e. the insureepolicy will be created) based on the creation of the contribution or reception of the payment.

## Business Processes

The different process bellow will all respect the location scope of the users, meaning that to add a policyholder for a region the scheme clerk must have access to that very region.

### Enrolment

The enrollment process that will be supported will be coming from Nepal requirements, even if we tried to broader, it might not support all the enrollment process of all group insurances



Policyholder (employer)

The first step before enrolling insure under a policy holder, one needs to make sure that the policy holder is defined in the tool; Read right is required to have access to the search/list screen, Managing Policy holders will be the role of the SchemeClerk

Steps

* Search PolicyHolder List > use the search box > search, the default list will show only the active policy holder
* Add: PolicyHolder List > “add” PolicyHolder > fill the details on the Policy holder page > save
* Update PolicyHolder List > “Select” PolicyHolder > update the details on the Policy holder page > save
* Delete PolicyHolder List > “Select” PolicyHolder > Delete (this role should be only available to admin, others should only archive by setting the DateValidTo)
* Add users

Portal

* Add/create user

Policy holder insuree

Managing policy holder insure will be possible from the policy holder page, this should be possible for the PolicyHolderClerk and the SchemeClerk

Steps

* Search PolicyHolder List > “Select” PolicyHolder > Insuree tab > use the search box > search, the default list will show only the active policy holder insuree
* Add: PolicyHolder List > “Select” PolicyHolder > Insuree tab > ‘add’ insure > fill the details on the Policy holder insuree page > save
* Update (error) PolicyHolder List > “Select” PolicyHolder > Insuree tab > update the details on the Policy holder page > save
* Replace (new version) PolicyHolder List > “Select” PolicyHolder > Insuree tab > update the details on the Policy holder page > create new version 🡪 add ValidityTo
* Delete PolicyHolder List > “Select” PolicyHolder > Insuree tab > “Select” an insuree> Delete (this role should be only available to admin, other should only archive by setting the DateValidTo)

**Portal access:**

* Search > My Insuree tab > use the search box > search, the default list will show only the active policy holder insuree
* Add: My Insuree tab > ‘add’ insure > fill the details on the Policy holder insuree page > save
* Update (error) My Insuree tab > update the details on the Policy holder page > save
* Replace (new version) My Insuree tab > update the details on the Policy holder page > create new version 🡪 add ValidityTo
* Delete My Insuree tab > “Select” an insuree> Delete (this role should be only available to admin, other should only archive by setting the DateValidTo)

PolicyHolder contribution plan bundle

Managing Policy holder contribution plan bundle, this should be possible for the SchemeClerk or SchemeAdmin

* Search PolicyHolder List > “Select” PolicyHolder > contribution plan tab > use the search box > search, the default list will show only the active contribution plan
* Add: PolicyHolder List > “Select” PolicyHolder > contribution plan tab > ‘add’ contribution plan bundle > fill the details on the Policy holder insuree page > save
* Replace (new version) > “Select” PolicyHolder > contribution plan tab > “Select” a contribution plan bundle >> update the contribution plan > create a new version > add ValidityTo
* Delete PolicyHolder List > “Select” PolicyHolder > contribution plan tab > “Select” a contribution plan bundle > Delete

contribution plan bundle

Managing a contribution plan bundle should be possible for the SchemeAdmin

* Search contribution plan bundle List > use the search box > search, the default list will show only the active contribution plan bundle
* Add: contribution plan bundle List > ‘add’ contribution plan bundle > fill the details on the contribution plan bundle page > save
* Update contribution plan bundle List > “Select” contribution plan bundle > update the details on the contribution plan bundle > save
* Delete contribution plan bundle List > “Select” contribution plan bundle > “Delete (this roles should be only available to admin, other should only archive by setting the DateValidTo)
* Replace contribution plan bundle-> List contribution plan bundle List > “Select” contribution plan bundle > update the details on the contribution plan bundle, update the contribution plan > replace version > add ValidityTo (DateValidTo of the to-be-replace CPB will be set are the Validity from the new CPB )

contribution plan

Managing a contribution plan bundle should be possible for the SchemeAdmin

* Search contribution plan List > use the search box > search, the default list will show only the active contribution plan
* Add: contribution plan List > ‘add’ contribution plan > fill the details on the contribution plan page > save
* Update contribution plan List > “Select” contribution plan > update the details on the contribution plan > save
* Delete contribution plan List > “Select” contribution plan > “Delete (this roles should be only available to admin, other should only archive by setting the DateValidTo)
* Replace contribution plan bundle List > List contribution plan > “Select” contribution plan > update the details on the contribution plan > create a new version > add ValidityTo (DateValidTo of the to-be-replace CPB will be set are the Validity from the new CPB )

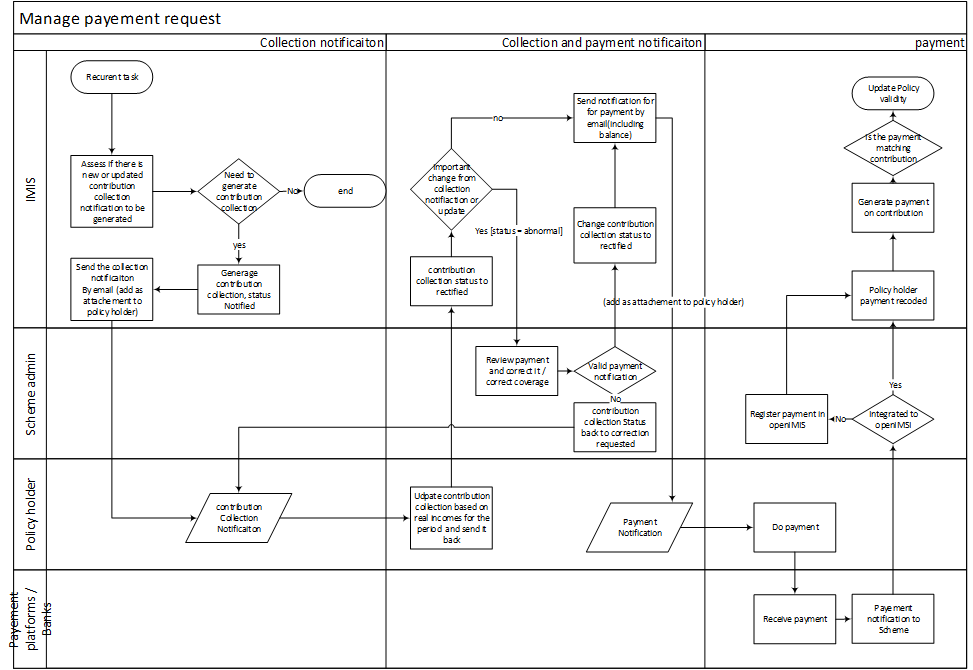
Contract

* Add bulk “contract”> Policy holder List > select policy holder(s) > Contract list > generate bulk contract > enter the date on which the contract must be created if not already covered by previous contract.
* Add “contract” > Policy holder List > Select a Policy Holder >> Contract list> Generate contract > enter the date on which the contract must be created if not already covered by previous contract.> conf with the contract dates
* Update details > Policy holder List > Select a Policy Holder > Contract list > Select contract > update one/ several lines > save
* Submit/sign “contract” > Policy holder List > Select a Policy Holder > Contract list > Select contract(s) > Submit/sign contract
* Review “Contract” > Policy holder List > Select a Policy Holder > Contract list > Select contract(s) > contract Reviewed (only for submitted contract)
* Ask updates “Contract” > Policy holder List > Select a Policy Holder > Contract list > Select contract(s) > ask update (only for submitted contract)
* Amend “contract” > Policy holder List > Select a Policy Holder > Contract list > Select contract >Generate Amendment > conf with Amendment number > add details (if it is about new employee, they should be added before on the employee list)

Portal access:

* Update details > My Contract list > Select contract > update one/several line > save
* Submit/sign “contract” > My Contract list > Select contract(s) > Submit/Sign contract

### Payment



* Add payment: Policy holder List > Select a Policy Holder > Contract list > create payment > enter value > save > conf giving the contract status
* update payment: Policy holder List > Select a Policy Holder > Contract list > Select payment > update value > save > conf giving the contract status
* delete payment: Policy holder List > Select a Policy Holder > Contract list > Select payment > delete payment > conf giving the contract status
* add credit note from contract or amendment

## policy update

The new policy status “contracted” indicate that the insuree policy doesn’t have the same dates as the policy because it is renewable contract payment that defines the validity of the insureepolicy

on payment, a script will:

* Check if there is a valid policy for the insure covered by the contract (+ grace period)
  + Create such policy if it doesn’t exist with the normal duration with a status “contracted.”
* Create or update the policy insure up to the end of the grace period, two insure policies could be created if two policies are required to cover the contract period.

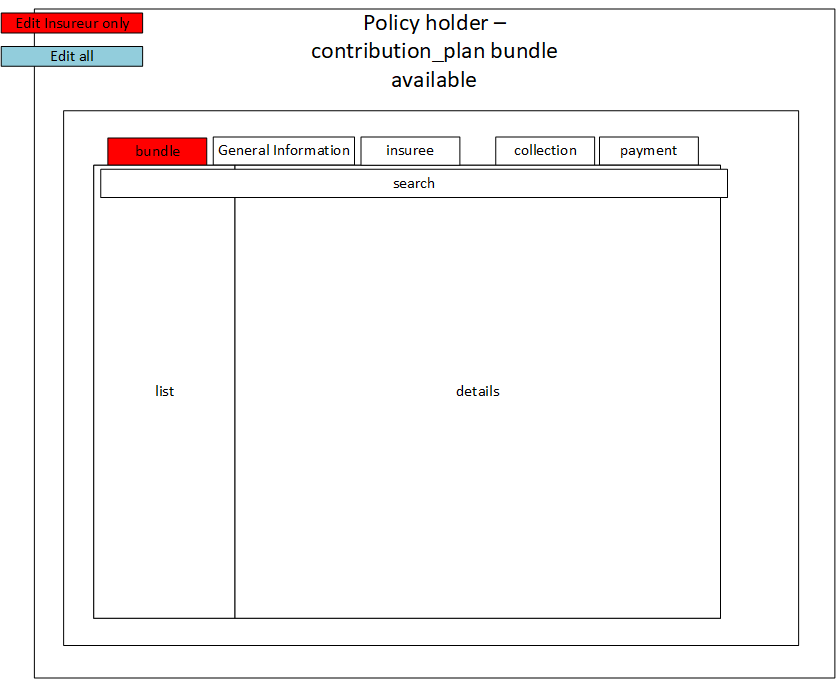
## Pages layout principle

For most of the entities, a list page and a read/edit page will be created;

Tabs will be used to when the entities depend of a parent such as the contribution collections are in a tab of the policyholder card. The tabs are shown only if the user has the “R” rights for the entity.

Two kinds of list/details will cohabit, the first is aligned with other openIMIS list and details as for the claim; this approach will be used for the policyholder and contribution plan bundle.

The second kind is a combined view used for rapid edit or relation list such as the contribution collection, contribution plan bundle available for policyholder, contribution plan member of a bundle.



## Roles and Authorities

Authorities

To manage the group insurance several authorities will be added, the existing authorities have a “\*” :

* PolicyHolder
  + C/R/U/D (4 authorities)
  + Portal R
* PolicyHolderInsuree
  + C/R/U/D (4 authorities)
* Approve (TBC)
  + Portal C/R/U/D (4 authorities)
* PolicyHolderContract
  + C/R/U/D (4 authorities)
  + Submit
  + Approve/ask for change
  + Amend
  + Portal submit
  + Portal Amend
* PolicyHolderContractDetails
  + C/R/U/D (4 authorities)
  + Approve
  + Portal C/R/U/D (4 authorities)
* PolicyHolderUser
  + C/R/U/D (4 authorities)
  + Portal C/R/U/D (4 authorities)
* Payment
  + C\*/R\*/U\*/D\*(4 authorities)
  + Submit (TBC)
  + Validate
  + CreditNote
* ContributionPlanBundle
  + C/R/U/D(4 authorities)
  + Portal R
* ContributionPlan
  + C/R/U/D(4 authorities)
  + Portal R

### Roles

To manage the group insurance several roles will be added, the existing roles have a “\*” :

* SchemeClerk(\*TBC)
  + PolicyHolder
    - C/R/U
  + PolicyHolderInsuree
    - C/R/U/D
    - Approve (TBC)
  + PolicyHolderContract
    - C/R/U
    - Submit
    - Approve
  + ContributionPlanBundle
    - R
  + ContributionPlan
    - R
  + PolicyHolderContributionPlan
    - R
* SchemeAdmin\*
  + PolicyHolder
    - C/R/U/D
  + PolicyHolderInsuree
    - C/R/U/D
    - Approve (TBC)
  + PolicyHolderContract
    - C/R/U/D
    - Submit
    - Approve
  + ContributionPlanBundle
    - C/R/U/D
  + ContributionPlan
    - C/R/U/D
  + PolicyHolderContributionPlan
    - C/R/U/D
* PolicyHolderClerk
  + PolicyHolderInsuree
    - C/R/U
  + PolicyHolderContract
    - R/U
    - Submit
  + ContributionPlanBundle
    - R
  + ContributionPlan
    - R
  + payment
    - submit
* accountant\*
  + Payment
    - Validate

## Entities

The entities will have different versioning in place:

Entity related to a specific date (contract …) no versioning will be in place only logging/audition, the code or reference will be used to differentiate them on a business level.

Configuration item (Product, plans, bundle, calculation, policy holder, policyholders insuree): those could be replaced by another version at a given time therefore no CI could have the same code and overlapping validity dates; they will have

* DateValidFrom (date)
* DateValidTo (date)
* Active
* Code (this stays the same across version)

Sub tables won’t have any versioning because a new version of the parent table will create new row in the sub tables.

### PolicyHolder

* UUID (char(24) )
* Code (varchar)
* Version ( integer - to be used for optimistic locking)
* TradeName (varchar)
* LocationsUUID (fk tblLocations)
  + Not limited to one level
* Address (varchar)
* Phone (varchar)
* Fax (varchar)
* Email (varchar)
* ContactName (varchar)
* LegalFormUUID (fk dictionnary:legalForm )
* ActivityCodeUUID (fk dictionnary:activity)
* AccountancyAccount (varchar)
* PaymentReference(varchar)
* DateValidFrom (date)
* DateValidTo (date)
* Active(bool)
* Json\_ext (json)
* DateCreated(date)
* DateUpdate(date)
* UserUpdate UUID (fk users)
* UserCreate UUID (fk users)

### PolicyHolderInsuree

* UUID (char(24) )
* Version ( integer - to be used for optimistic locking)
* PolicyHolderUUID (fk policyHolder)
* InsureeUUID (fk Insuree)
* ContributionPlanBundleUUID (fk\_contributionPlanBundle)
* LastPolicyUUID (fk Policy)
* Json\_ext (json)
* DateCreated(date)
* DateUpdate(date)
* UserUpdateUUID(fk users)
* UserCreateUUID (fk users)

### PolicyHolderContributionPlan

* UUID (char(24) )
* Version ( integer - to be used for optimistic locking)
* ContributionPlanBundleUUID (fk\_contributionPlanBundle)
* PolicyHolderUUID (fk policyHolder)
* Json\_ext (json)
* DateCreated(date)
* DateUpdate(date)
* UserUpdateUUID(fk users)
* UserCreateUUID (fk users)
* Active (bool)
* DateValidFrom (date)
* DateValidTo (date)

### PolicyHolderUser

* UUID (char(24) )
* UserUUID (fk\_contributionPlanBundle)
* PolicyHolderUUID (fk policyHolder)
* Json\_ext (json)
* DateCreated(date)
* DateUpdate(date)
* UserUpdateUUID(fk users)
* UserCreateUUID (fk users)
* Active (bool)
* DateValidFrom (date)
* DateValidTo (date)

### Contract (Contribution Collection)

* UUID (char(24) )
* m locking)
* PolicyHolderUUID (fk policyHolder)
* AmountNotified (float)
  + Upon Automatic creation
* AmountRectified (float)
  + Manual update from the policy holder
* AmountDue (float)
  + Value approved by the Scheme admins
* PaymentDateDue (date)
* Status (fk:dictionary:ContractStatus)
* PayementReference(string)
* Json\_ext (json)
* DateCreated(date)
* DateUpdate(date)
* UserUpdateUUID(fk users)
* UserCreateUUID (fk users)
* ContractFrom (date)
* ContractTo (date)

it has a sub entity ContractDetails

* UUID (char(24) )
* ContractUUID (fk Contract)
* PolicyHolderInsureeUUID (fk PolicyHolderInsuree)
* ContributionPlanBundleUUID:Version(fk ContributionPlanBundle)
* Json\_ext (json)
* Json\_param (json)
  + PolicyHolderInsuree parameters
* Json\_param\_history (json)
  + Same as Json\_ext but for audits
* DateCreated(date)
* DateUpdate(date)
* UserUpdateUUID(fk users)
* UserCreateUUID (fk users)

it has a sub entity ContractContributionPlanDetails

* UUID (char(24) )
* Version ( integer - to be used for optimistic locking)
* ContributionPlanUUID:Version (fk\_contributionPlan)
* PolicyUUID (fk\_policy)
* ContributionUUID:Version (fk premium)
* Json\_ext (json)
  + ContributionPlan parameters
* DateCreated(date)
* DateUpdate(date)
* UserUpdateUUID(fk users)
* UserCreateUUID (fk users)

### ContributionPlanBundle

* UUID (char(24) )
* Version ( integer - to be used for optimistic locking)
* Active (bool)
* Code (varchar)
* Name (varchar)
* Json\_ext (json)
* DateCreated(date)
* DateUpdate(date)
* UserUpdateUUID(fk users)
* UserCreateUUID (fk users)
* DateValidFrom (date)
* DateValidTo (date)

it has a sub entity ContributionPlanBundleDetails

* UUID (char(24) )
* Version ( integer - to be used for optimistic locking)
* ContributionPlanBundleUUID: (fk ContributionPlanBundle)
* ContributionPlanUUID: (fk\_contributionPlan)
* Json\_ext (json)
* DateCreated(date)
* DateUpdate(date)
* UserUpdateUUID(fk users)
* UserCreateUUID (fk users)
* DateValidFrom (date)
* DateValidTo (date)
* Active (bool)

### ContributionPlan

* UUID (char(24) )
* Version ( integer - to be used for optimistic locking)
* Code (varchar)
* Active (bool)
* CalculationUUID
* BenefitPlanUUID (fk\_product )
* Periodicity (int)
  + Number of month between 2 payment
* Json\_ext (json)
  + GracePeriod
  + Rate
  + Threshold
  + …
* DateCreated(date)
* Amendment (0 for the contract)
* DateUpdate(date)
* UserUpdateUUID(fk users)
* UserCreateUUID (fk users)
* DateValidFrom (date)
* DateValidTo (date)

### CalculationRules

* UUID (char(24) )
* CalculationClassName(varchar)
* Status (int)
* Decription (varchar)
* Version (int)
* Json\_ext (json)
* DateCreated(date)
* DateUpdate(date)
* UserUpdateUUID(fk users)
* UserCreateUUID (fk users)
* Active (bool)
* DateValidFrom (date)
* DateValidTo (date)

It has a sub table CalculationRulesDetails

* UUID (char(24) )
* CalculationRulesUUID (varchar)
* Status (int)
* ClassName
* Main (bool)
* Params (Json)
* ClassParams(json)
  + Type
  + rights
  + Relevance
  + conditions
* Json\_ext (json)
* DateCreated(date)
* DateUpdate(date)
* UserUpdateUUID(fk users)
* UserCreateUUID (fk users)

Dictionary (module conf level)

* Type
* UUID
* code
* label (JSON)