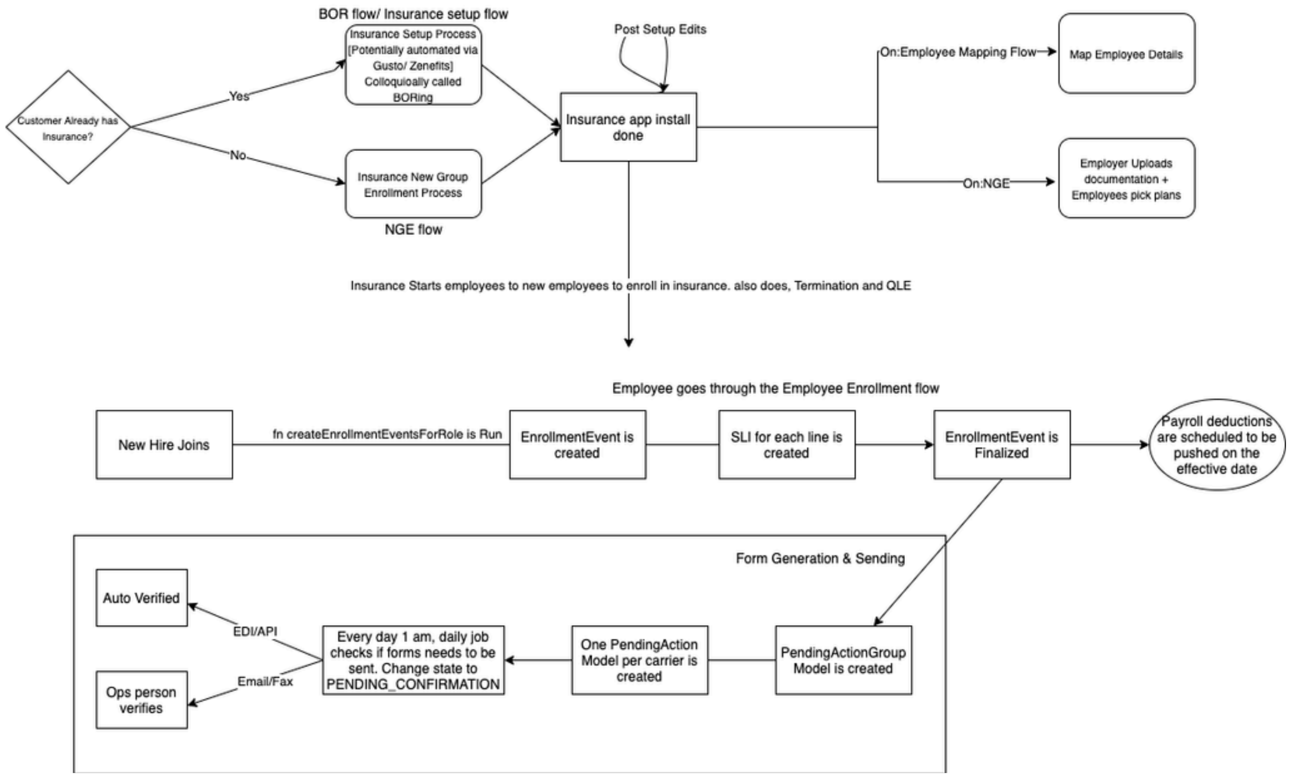
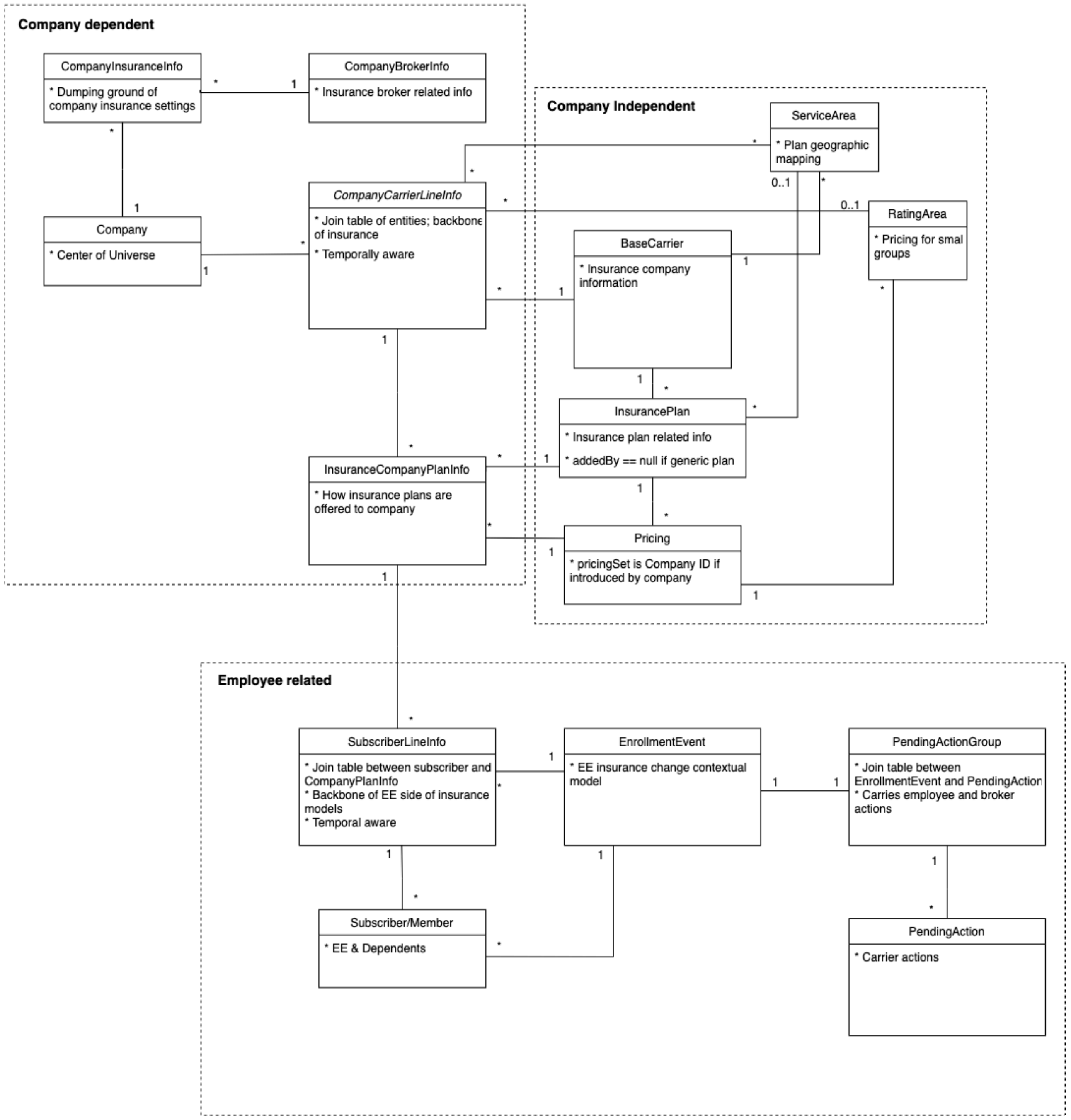
Insurance Models

Flow Diagram


BOR flow/ Insurance setup flow
Insurance Setup Process [Potentially automated via Gusto/ Zenefits] Colloquioally called BORing
Customer Already has Insurance?
Yes
No
Insurance New Group Enrollment Process
NGE flow
Post Setup Edits
Insurance app install done
On:Employee Mapping Flow
Map Employee Details
On:NGE
Employer Uploads documentation + Employees pick plans
Insurance Starts employees to new employees to enroll in insurance. also does, Termination and QLE
Employee goes through the Employee Enrollment flow
New Hire Joins
fn createEnrollmentEventsForRole is Run
EnrollmentEvent is created
SLI for each line is created
EnrollmentEvent is Finalized
Payroll deductions are scheduled to be pushed on the effective date
Form Generation & Sending
PendingActionGroup Model is created
One PendingAction Model per carrier is created
Every day 1 am, daily job checks if forms needs to be sent. Change state to PENDING_CONFIRMATION
EDI/API
Auto Verified
Email/Fax
Ops person verifies

Database Schema

This diagram is incomplete and does not completely match model descriptions that follow.



Company dependent
CompanyInsuranceInfo
* Dumping ground of company insurance settings
CompanyBrokerInfo
* Insurance broker related info
Company
* Center of Universe
CompanyCarrierLineInfo
* Join table of entities; backbone of insurance
* Temporally aware
InsuranceCompanyPlanInfo
* How insurance plans are offered to company
Company Independent
ServiceArea
* Plan geographic mapping
RatingArea
* Pricing for smal groups
BaseCarrier
* Insurance company information
InsurancePlan
* Insurance plan related info
* addedBy == null if generic plan
Pricing
* pricingSet is Company ID if introduced by company
Employee related
SubscriberLineInfo
* Join table between subscriber and CompanyPlanInfo
* Backbone of EE side of insurance models
* Temporal aware
EnrollmentEvent
* EE insurance change contextual model
PendingActionGroup
* Join table between EnrollmentEvent and PendingAction
* Carries employee and broker actions
Subscriber/Member
* EE & Dependents
PendingAction
* Carrier actions
0..1

Source diagram file

Effectivity is used in this design. Learn more here

- Effectivity
- Temporal Property
- https://robinhood.engineering/tracking-temporal-data-at-robinhood-b62291644a31

Database Models

Company Independent Core

Insurance Carrier BaseCarrier in diagram

Represents a Carrier that Insurance works with. Main Attributes: StateCode. CarrierIds.

InsurancePlan Design Debt

- Not directly dependent on company.
- Mainly Provided by external Vendors. Vericred. Noyo. Metlife. Beam etc.
  - NGE team mainly owns this input of external pricing Data into our code.
- If pricing entered by company, a pricing model is created and the company ID is stored in "addedBy"

Pricing Design Debt

- Not directly dependent on company.
  - Mainly Provided by external Vendors. Vericred. Noyo. Metlife. Beam etc.
    - NGE team mainly owns this input of external pricing Data into our code.
  - If pricing entered by company, a pricing model is created and the company ID is stored in "`pricingSet`"
  - PricingChart: Store the actual pricing. Could be a lot of structures. (doc)

RatingArea

- ACA standardizes the factors that insurance companies can use to calculate premiums for small group rates.
  - Each state is required to divide up the areas of the state into locations called **Rating Areas**. When insurance companies price their premiums, all households within a rating area will have the same adjustment factor applied.
  - Pricing related
  - More information provided

ServiceArea

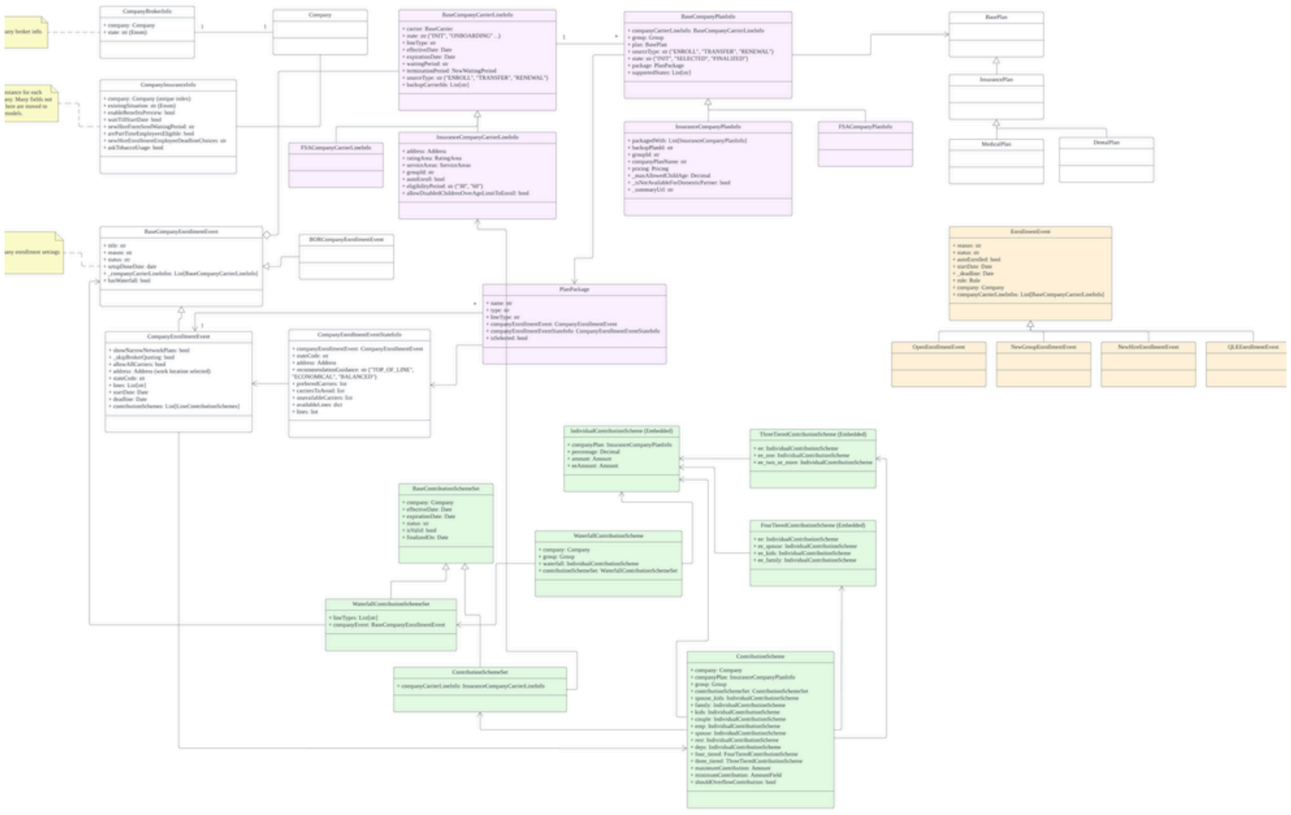
- Refers to where a plan network is available. Determines whether an employee is able to enroll in a certain plan based on where he/she is located.
  - Mostly used for HMO carriers/plans which are limited based on zipcode.
  - Enrollment eligibility related

Relationships

- InsuranceCarrier has One to Many Relationship with InsurancePlan[An Insurance carrier can offer many plans]
- InsurancePlan has One to Many Relationship with InsurancePricing[An Insurance plan can have many pricing depending on Quarter/ zip code etc]

Company Dependent

Following is the UML of company enrollment related models.


CompanyBrokerInfo
Company
BaseCompanyCarrierLineInfo
BaseCompanyPlanInfo
BasePlan
CompanyInsuranceInfo
InsurancePlan
FSACompanyCarrierLineInfo
InsuranceCompanyCarrierLineInfo
InsuranceCompanyPlanInfo
FSACompanyPlanInfo
MedicalPlan
DentalPlan
BaseCompanyEnrollmentEvent
BORCompanyEnrollmentEvent
EnrollmentEvent
PlanPackage
CompanyEnrollmentEvent
CompanyEnrollmentEventStateInfo
OpenEnrollmentEvent
NewGroupEnrollmentEvent
NewHireEnrollmentEvent
QLEEnrollmentEvent
IndividualContributionScheme (Embedded)
ThreeTieredContributionScheme (Embedded)
BaseContributionSchemeSet
WaterfallContributionScheme
FourTieredContributionScheme (Embedded)
WaterfallContributionSchemeSet
ContributionSchemeSet
ContributionScheme

Purple: company plan related classes

Green: contribution scheme related classes

White: company insurance setting and company enrollment events

Yellow: employee enrollment related classes

CompanyBrokerInfo models missing

Company

Need Info

CompanyBrokerInfo

Need Info

CompanyInsuranceInfo

A generic ‘catch-all’ model where we dump a bunch of company insurance settings such as who is the person to sign in BOR letters, do they want part time employees to be eligible etc.

InsuranceCompanyCarrierLineInfo (CCLI)

diagram shows CompanyCarrierLineInfo

A complicated model. One of the backbones of Insurance.

- Temporally aware.
  - This is a basically a unique key on (Carrier, effectiveDate, expirationDate, lineType, Company).
  - I think of it as representing a contract between a company and carrier for a lineType for a particular duration(effectiveDate & expirationDate)

    - Stores things like waiting period/ contribution scheme/ effective date etc.
    - Relationship: A join between Carrier/company per (lineType, Duration)

InsuranceCompanyPlanInfo

CCLI + InsurancePlan

- Mostly contains how the company is offering an InsurancePlan to the company.
    - These are the customizations that a company could do on a plan.
        - e.g. which set of employees they can offer to.
    - CCLI has a one to many relationship to InsuranceCompanyPlanInfo
        - There will be *at least one* entry per CCLI

CompanyEnrollmentEvent

not in diagram

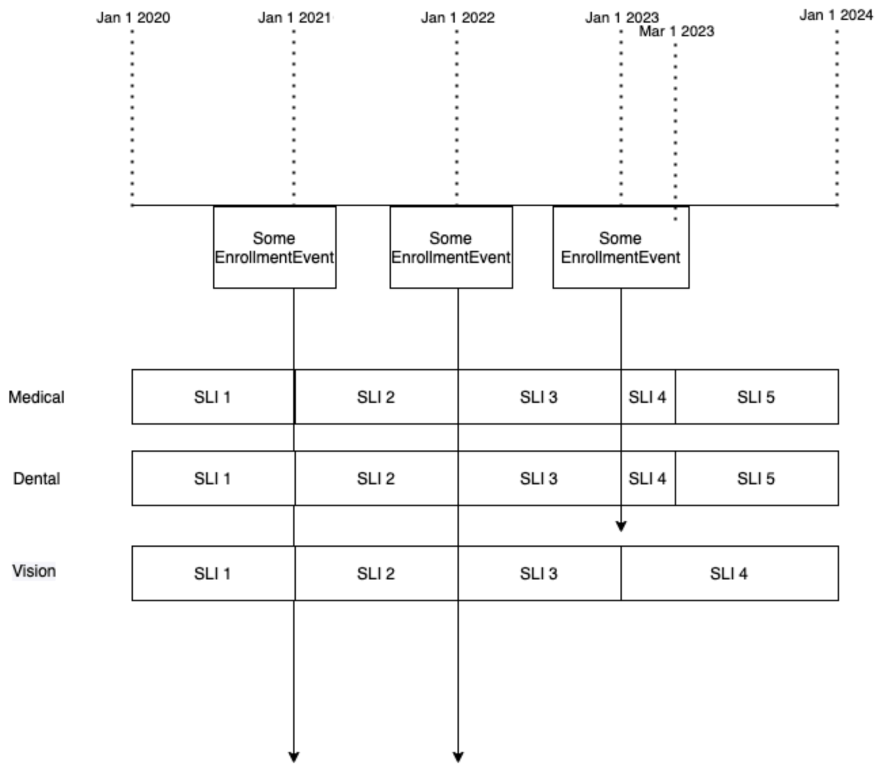
- There are many types of events here. Whenever a major company wide insurance change happens, we create this event.
- Inherit from RPDocumentWithoutRole, this event is a company level event
- Eg: When a company is transferring insurance to Rippling we create a BORCompanyEnrollmentEvent
    - When a company is shopping Insurance[NGE flow] we create a CompanyEnrollmentEvent, with reason=”NEW_GROUP_ENROLLMENT”.
    - When a company is renewing its contract yearly or changing their contracts, we create CompanyEnrollmentEvent, with reason=”OPEN_ENROLLMENT”
    - TODO Sankar to add more details here or link to different page

CompanyEnrollmentEventStateInfo

- One CompanyEnrollmentEvent have many CompanyEnrollmentEventStateInfo
- One CompanyEnrollmentEventStateInfo have many PlanPackage

Employee Related

Subscriber/Member model missing



SubscriberLineInfo

For each line of Insurance, the employee can make selections about their Insurance. Together, these would form a continuous timeline. For each line, we store this in a model called SubscriberLineInfo.

Commonly abbreviated as sli.

SLI is temporally aware model.

Main Attributes:

- EffectiveDate + ExpirationDate
- ChosenPlan → Optional[InsuranceCompanyPlan]
- chosenDependents → List[Members]
- lineType → the Linetype
- EnrollmentEvent → A reference to the Event which caused the creation of this SLI

A new SLI is created only when there is an occasion to do so(an event). This is called an EnrollmentEvent.

Usually, an employee's first SLI is created due to a NewHireEnrollmentEvent. Every year during open enrollment a new SLI is created due to an OpenEnrollmentEvent(which in turn is created due to an CompanyEnrollmentEvent). New slis might also be created due to, QLE. or Termination or Cobra.

Events might affect a change in insurance coverage(and creation of SLIs) for only *some* lines of coverage. Eg: An QLE might result only new selections available for Medical/Dental and Vision.

Note: Even if an employee doesn't change their elections. A new sli means, there was an option for changing the Insurance coverage of an employee

SubscriberCompanyEventInfo

not in diagram

Info per subscriber per company event, i.e. if this subscriber `hasBeenProcessed` , which will be used to identify if the subscriber is eligible to import for insurance enrollment or not.

By leveraging this object, we can support multiple instances of same company event type.

EnrollmentEvent

Enrollment creation is governed by complicated logic explained <link here>.

- Each different kind of enrollment has implement the following functions
  - **getEffectiveDate(self, companyCarrierLineInfo)**
    This gives the effective date of the insurance for this ccli. THIS IS TIME INDEPENDENT.
  - **getDeadline()**
    Deadline to enroll in Insurance. might want to do it earlier than the absolute last minute. Especially for OE/NGE.
    - getEnrollmentLimitDate(self, companyCarrierLineInfo) → This give the last date carrier would accept insurance enrollment. Usually, effective-date + 30 days. ALSO TIME INDEPENDENT.
  - **getReferenceDate()**
    Reference date
  - **isAvailableForEmployee(ccli, referenceDate)**
    Is the plan offered by CCLI available to the employee on the given date.
- Common Enrollment Event Functions
  - getAvailableCCLIs()
  - getAvailablePlans()
  - manageNotificationLifeCycle()
- `OpenEnrollmentEvent` Inherit from RPDocumentWithRole, which is a role level event

PendingActionGroup Employee/Ops Related

Whenever, an EnrollmentEvent is finalized. AKA, when an employee makes all the insurance selections available to them, we create a PendingActionGroup (through a `on_change` hooked function `EnrollmentEvent.onFinalized()`). This is co-ordinator model which coordinates all the communication to the carriers about the employee insurance selections.

The co-ordination function is process().

Common abbreviation: pag.

PendingAction Employee/Ops Related

For each carrier we *might* have to communicate the employee details, we create a pendingAction, the initial status is “INIT”

PendingActionGroup has one-to-many relationship with PendingAction.

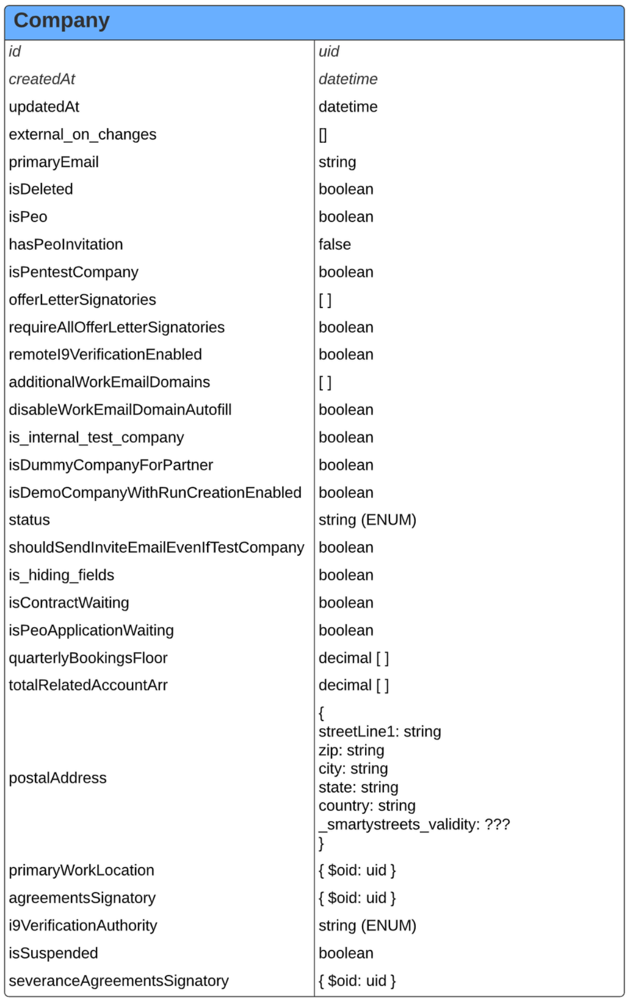
On PendingActionGroup.process(), we try to fill the employee selections in forms. If we are successful, we set the status of PA to “PROCESSED”.

PendingAction has two extremely important functions.

a) shouldPerformAction: Decides if we need to communicate the enrollment information to the insurance carrier.

b) performAction: Actually performs the action and communicates information to the carrier.

- Actual communication logic is more complicated and changes too much to put in writing. Best is to look at tests or the code itself.



Company	
id	*uid*
createdAt	*datetime*
updatedAt	datetime
external_on_changes	[]
primaryEmail	string
isDeleted	boolean
isPeo	boolean
hasPeoInvitation	false
isPentestCompany	boolean
offerLetterSignatories	[ ]
requireAllOfferLetterSignatories	boolean
remoteI9VerificationEnabled	boolean
additionalWorkEmailDomains	[ ]
disableWorkEmailDomainAutofill	boolean
is_internal_test_company	boolean
isDummyCompanyForPartner	boolean
isDemoCompanyWithRunCreationEnabled	boolean
status	string (ENUM)
shouldSendInviteEmailEvenIfTestCompany	boolean
is_hiding_fields	boolean
isContractWaiting	boolean
isPeoApplicationWaiting	boolean
quarterlyBookingsFloor	decimal [ ]
totalRelatedAccountArr	decimal [ ]

postalAddress	{
	streetLine1: string
	zip: string
	city: string
	state: string
	country: string
	_smartystreets_validity: ???
	}

primaryWorkLocation	{ $oid: uid }
agreementsSignatory	{ $oid: uid }
i9VerificationAuthority	string (ENUM)
isSuspended	boolean
severanceAgreementsSignatory	{ $oid: uid }

Important Tech Talk Sessions

Core tech talk 1

Enrollment event talk

Insurance plan and pricing talk

Permissions