API Design Guidelines and Standards - Real World Example

Timothy S. Hilgenberg

Version 0.9.0, August, 2022

Table of Contents

Colophon

This document is a companion document to the API Design Standards and Guides. The purpose of this document is to provide a real-world example of how to apply the API design guidelines to a real-world domain HR Benefit Administration.

This book is a very early draft of this manuscript

This book is being written in AsciiDoc using Visual Studio Code. I'm just learning this tool, so I apologize in advance for any bad formatting issues with early versions of the book.

All comments welcome

Early Working Draft Version 0.9.0 — June 2022

Benefits Administration Design Philosophy

A highly effective API will publish a unified, correlated, simplified enterprise data model to both internal applications and external 3rd parties. The primary focus is on person related inquiries and HR business processes. However, it should ultimately expose all the key business administrative recordkeeping entities in the administrative domain. These key entities in the HR Benefit Domain include:

- Person
  - Employee
  - Family Dependent
  - Covered Dependent
- Plan Participation by Domain (Health Management, Defined Contribution, Defined Benefits)
- Person-related Business Processes
- Plan Provisions
- Business Process Definition or Template
- Client
- Employer
- Carrier/Insurance providers

There are two types of domain API for HR Administration:

Inquiries	Provides consumers with person, employment, and domain related plan participant data attributes. These are the current and historical data attributes related to participation in the various benefits plans.
Business Processes	Provides with consumers with the ability to transact - start, manage and close, HR Administrative business processes or workflows.

When doing the design of domain API, one should be following an approach called Domain Driven Design. In this approach, the goal is to create **bounded contexts** around the key primary independent business entities. The key considerations are:

- Person Inquiry data (Results or outcomes of Business Processes) and Person Business Processes (Transactional Requests from participants) are different entities and bounded contexts. They also have very different primary keys.

- At the macro level, one would organize the API around the high-level business entities of the domain
  - Since the inquiry business entities around person all have the same primary key (prsnId, ssn, etc), it does not fit well with the concept of a **bounded context**. However, there is precedence to organize around an organization structure (Conway's Law)
  - By organizing around domains, it follows the future microservices architecture of giving deployment independence to the domain teams
  - The following are the **bounded contexts** for inquiry business entities:
    - Person
    - Worker/Employee
    - Person Defined Contribution
    - Person Defined Benefits
    - Person Reimbursement (YSA, spending accounts)
    - PersonMessaging for communication and GMC messages
  - The goal is to define our REST resources to parallel the definition of a **bounded context**
- Since the business objects can get very large and deep in structure and have components of interest to third parties, one can provide subset API response structures of the macro business object called 'views'
  - These views will be primarily used for access control by external entities
  - To promote simplicity and standardization, there will be a guideline to only have around 5 'views' per domain bounded context.
  - It is felt that a consumer would only need one of these views and that it is ok to receive more data than need. The consumer should just avoid fields they are not interested in.
  - This follows our 'API first' approach, where the design is driven by the business data model and not the perspective of the consumer application.
- From the person perspective, in addition to the organization view, we also want to organize our design around **Person Roles**
  - Inside the person entity, we plan to keep track of all the roles a person can play.

Person Role

- In the HR domain, a person can play may roles and assume many personas. The definition of a person role is a relationship between a person and another independent entity. The main primary entities in the HR domain are Plan and Employer. Typically, the roles form a many:many relationship between the two entities. This allows us to better support situations where the person can play multiple roles of the same type, where the only difference is a different independent object instance.

- However, these roles can be related to any independent entity outside of employer and plan as long as they have a single or compound primary key. For example, citizenship is a role between a person and a government. A shareholder is a relationship between a person and a company. The table below outlines the key ones in our domain.

Table 1. Person Role Table

Component	**Description**
Person Role	Related Primary Entity
Employee/Employment	Employer
HM Plan Participant	HM Plan/Insurance Election
DC Plan Participant	DC Plan/Insurance Election
DB Plan Participant	DB Plan/Insurance Election
Spending Account Plan Participant	Spending Account Plan/Insurance Election
Family Dependent	Secondary Person
Covered Dependent	Secondary Person and Plan

REST Resource Design for Person Inquiry

For most HR applications, there are 5 major bounded contexts. They are all keyed by the persons primary identifier (PrsnID, SSN, etc). They are:

- /persons
  - This contains a person’s demographic/tombstone and contact/address data
- /personsDefinedBenefits
  - This contains the person defined benefit participation for multiple plans
- /personsDefinedContribution
  - This contains the person defined contribution participation for multiple plans
- /personHealthManagement
  - This contains the person participation in multiple plans related to Health Plans
  - Would include all types of health and welfare plan, including medical, life insurance, dependent insurance, spending account and other types of elected plans
- /personReimbursement
  - This contains the person FSA participation for multiple plans

All inquiry data would be placed into one of these bounded contexts. Although this is a bit different from the general Domain Driven Design approach, it is important for domain deployment independence to break down the bounded contexts by domain.

There is a concern that the REST API responses would be very large grained, deeply structured objects and too big for consumers apps to absorb. To provide a more consumable set of data, we are suggesting a concept of a 'view', which would provide a filtered or mapped views of the bounded context. The next section on Inquiry API discussed 'views' in more detail.

Our reading of the REST principals is the REST resource on the base URL should be used to identify the business object in question. It should only contain the aggregate entity in question and the primary keys needed to find this object. In the case of associative entities, like PersonPlan, the multiple entities could be on the base URL (e.g. ignoring PII issues for PersonPlan) Example: /persons/{id}/plans/{planId}

Any action that could modify or act on the business object should go as a query parameter.

Rest Resource Design for Business Processes/Workflow

Business Process Bounded Context Principals

- Recordkeeping Business Processes objects are a different REST resources/bounded context from the person based static inquiry data
    - Business Process Name: /person<business process>
    - Primary Key: businessProcessReferenceNumber
- Resource Granularity
    - Preference: Actual Domain Business Process Type:
- /enrollment, /fundTransfer + id for a specific instance
- Lowest level Business Process Type- Next level specific from Type; /nonQualifiedFundTransfer, /annualEnrollment - Instance level
    - Generic - /businessProcess/{id} - parameter for instance - Least specific, driven by an id
- There should be no direct PUT/POST/DELETE against static person results data outside of a business process, data should only be changes through business processes with the results of modifying business results side data
- Direct update access for plan provisions, business process and employer configuration would be allowed
- HTTP Verbs:
    - POST will be used to create/start a business process
    - GET will be used to get the data for the person instance of the business process template. GET options include:
        - Retrieve the entire resource
        - Provide a "view" option to retrieve a subset of the resource
        - Example: Enrollment calculations for options and pricing
    - PUT will be used to revise/update user transactional data/elections
        - An Overload PUT will be used to drive additional update operations (revise, confirm, complete, etc) against the business process.
        - The standard set of update directives are:
            - Revise

- Validate
- Confirm
- UndoAll or UndoLast or Cancel
- Complete

General Rest Resources Patterns

In general, business processes follow a consistent interaction pattern. There are three API types that are called in sequential order:

1. Business Process Template Provisional Resource with Person business rules applied
2. Person Business Process Resource Instance - Person specific instance of #1
3. Person Domain Resource - Person level outcomes of #2

Business Process Template Provisional Resource w/Person Business Rules applied

1. Primary Key: businessProcessTemplateId
    a. Example: GET /fundTransfer ?personID=XX?plan=XXX
2. The API response would contain:
    a. Business Process Template Provisional Data for all instances of resource type
        i. If Plan specified, Plan provision data or specific business process id
    b. Person Business Rule Calculated Results
        i. if present, optionally brings back results from business rules related to plan and business process policies for the specific person
    c. Person Pending Business Processes
        i. If there are any pending business processes for the person, brings back any pending activities - Header/Summary data
            A. BusinessProcessId
            B. BusinessProcessReferenceId
            C. Person ID
            D. Process Effective Date
            E. Process Status
            F. Outstanding Edits
3. If business process template {id} is known, GET /fundTransfer/{id} ? personId=XX will

respond with a single instance.

4. If the {id} is not known, then the list of instances for the type will be provided.
5. If the Person is provided, then any business process eligibility will be determined and the list filtered by the eligibility to run the business process.

Person Business Process Resource Instance - Person specific instance of #1

1. Primary Key: businessProcessReferenceNumber
   a. Example: GET /personFundTransfer/{businessProcessReferenceNumber}
2. Returns the data related to the person specific input of the transaction
3. Once the participants start providing input, the updating API (PUT, POST, DELETE) methods are used

Person Domain Resource - Person level outcomes of #2

1. Primary Key: person ID + Domain
   a. Example: /personDefinedContributions
2. Person with businessProcessView
   a. Primary Key: personId
   b. Example: /persons/view/businessProcesses;filter="pending"
   c. Optionally filtered by pending/open business processes
   d. Used by home page to display open business process tiles
   e. Can pass person Id, businessProcessId and businessProcessReferenceNumber to business process flow(i.e start through confirm page)
      i. Contains data to retrieve the above 3 resources

Uses Cases

1 - Business Process started from the consumer app - Immediate process with no pending business process possible

In this use case,there is no pending business process, and * the business process is a straight through transaction with a process completion at the end of the interaction * There is only one type of this event in the client configuration

Example: Person Data Update business process for updating contact information (address, email, etc)

1. Upon entering the business process flow:
    a. GET /contactsChange?personID=XX -→ BusinessProcessProvision Business Object
        i. Returns information related business process provisional policies overlying person business rules filtering → Address ID List for person
    b. GET /persons/view/demographics -→ Person
        i. Returns person tombstone (Name, birth date, About You) and contact information
2. After the user interface collects new information:
    a. POST /personContactsChange
        i. Request Data:
            A. PersonId, BusinessProcessId and Effective Date are known
            B. HTTP Request Body contains update fields values, email
            C. Processing Option: NoSave
                I. In this case, there will only be validation and not a true POST to the data base. This would be used where a confirmation is desired. There will need to be a subsequent POST to post and process the business process
        ii. Response
            A. OK - No edit errors
            B. If the POST throws an edit, error(s) will be returned, and no data posted. The user interface will have the option of changing the data and submitting another POST.
            C. Another option is one that allows the caller to request the activity to be saved even if there are edits. This would then create a pending event. This is not your

typical interactions model, where the save is all-or-nothing.

2 - Business Process started from the consumer app - Immediate process/ Processed at night outside the consumer app

In this use case, * there is no pending event, and * the business process will process, but not completed. Stored in a pending state * There could be multiple types of this business process

Example: Defined Contribution Fund Transfer

1. Upon entering the business process flow:
    a. GET /fundTransfer?personID=XX
        i. Returns related business process provisional data and policies overlying person business rules filtering
            A. Could bring back multiple businessProcessIds, application needs to determine (by rule or by asking) what businessProcessId to use or is filtered by person eligibility
    b. GET /personsDefineContributions/view/fund
        A. Returns the Person Defined Contribution current fund allocations
2. After the user interface enters the new information:
    a. POST /personFundTransfer.
        i. PersonId,BusinessProcessId and Effective Date are known
        ii. RequestBody contains update fields values
        iii. Processing Option: NoSave - In this case, there will only be validation and not a true POST to the data base. This would be used where a confirmation is desired. There will need to be a subsequent POST to post and process the business process
        iv. At this point, the personBusinessProcess is saved and in a pending state.
        v. If the POST throws an edit, error(s) will be returned, and no data posted. the user interface will have the option of changing the data and submitting another POST.
3. The completion of the transaction will occur during the evening batch job. Since this is a batch job, it is highly unlikely that and API would be used in this unit of work.

3 - Business Process started from the consumer app - Pending event

In this use case,

- there is pending event
- the business process will process, but not completed. Stored in a pending state
- There could be only one instance of this business process. Example: Pending FundTransfer
  1. Upon entering the business process flow:
     a. GET /fundTransfer?personID=XX
        i. Returns related business process provisional data and policies overlying person business rules filtering
        ii. Brings back the person’s businessProcessReferenceNumber for the business process
     b. GET /personsDefineContributions/view/funds
        i. Returns Person Defined Contribution current fund allocations
  2. After the user interface collects the new information:
     a. PUT/personFundTransfer/{businessProcessReferenceNumber}
     b. RequestBody contains update fields values
     c. Processing Option: NoSave - In this case, there will only be validation and not a true POST to the data base. This would be used where a confirmation is desired. There will need to be a subsequent POST to post and process the business process
     d. At this point, the personBusinessProcess is saved and in a pending state.
     e. If the POST throws an edit, the error(s) will be returned and no data posted. the user will have the option of changing the data and submitting another POST

4 - Business Process from pending event - Started in batch

In this use case, * there is pending event, which was started in batch * there is a prior page with a "tile" showing the pending event * the business process will process, but not completed. Stored in a pending state

Example: Annual Health Care Enrollment

1. Prior to the user session, there would be a batch program to create the annual enrollment

person instance.

   a. The program would need to be provided with:
      i. businessProcessId
      ii. Effective Date
   b. For each eligible person:
      i. POST /personAnnualEnrollment
         A. Creates a person business process object with a businessProcessReferenceNumber

2. Upon entering the business process flow
   a. GET /annualEnrollment/{businessProcessId}?personID=XX
      i. Returns related business process policies overlying person business rules filtering
      ii. Because the businessProcessId was provided, will only bring back one set of data for the given business process
      iii. If the businessProcessId is not provided, it can bring back a list of the business processes for this type of enrollment.
   b. GET /personAnnualEnrollment/{businessProcessReferenceNumber}
      i. Returns the person's pending event elections data
      ii. Returns eligible plan list, plan attributes, and eligible tool data (GEO, DESTRX, HPCC, MEE, etc.)
   c. GET /personsHealthManagement
      i. Returns the Person Health Management current and historical election data

3. After the user interface collects the new information/elections:
   a. PUT/personAnnualEnrollment/{businessProcessReferenceNumber}
      i. RequestBody contains update fields values
      ii. Processing Option: Revise- In this case, there will only be validation and not a true POST to the data base. This would be used where a confirmation is desired. There will need to be a subsequent PUT for confirmation and business process completion. If the PUT throws an edit, the error will be return and no data posted. the user will have the option of:
      iii. Changing the data and submitting another PUT
   b. Post elections PUT business process actions. If after the revise action, there needs to be:
      i. A confirmation action to commit resources to a data store, a PUT with a Confirmation Action indication needs to be sent

    ii. A complete to close the business process. A PUT with a Complete Action indication needs to be sent.

4. Person Query for pending events - Use from Home page
    a. GET /person/ view/ businessProcessHistory
        i. filter="pending", would only bring pending business processes

Appendix A: Business Process Definition

A business process is series of steps/actions/tasks described it a graph like flow (think flow chart) that accomplishes a specific domain related objective. Examples are the entire enrollment process through carrier notification, the entire dependent verification flow from request to final verification. In each of these cases, there is a clear domain objective the users is attempting to complete. This may be done in a short duration single transaction (i.e., committed unit of work) or a longer running flow over time (days or weeks) with time pauses throughout.

Some of the aspects of a business process are:

- There is some event or trigger that initiates the business process (sometimes is called a workflow)
  - This can be a user action/notification (i.e. Birth of a child) or time based (Starting enrollment on a specific date; opening an enrollment window)
- They contain a series of steps, events that trigger action logic.
  - There is a **business process template**, that defines the starting trigger, steps or the logic involved in the process and the definition of what is consider 'finished'. For examples, the steps of an annual enrollment are defined here. Then:
  - The individual instances are created from that template. Tim’s annual enrollment, John’s annual enrollment
  - They can form a flow graph with sequential steps, if with branches and loops
  - Each step is a discreet piece of logic sticking to the 'Single Responsibility' pattern. Examples are updating coverage, sending an email, etc)
  - Multiple steps can be combined into a committable **unit of work** or executed as a single action
  - Each step needs to define:
- Event that triggers the work if the process is in a paused state
- The logic to perform one the event is identified
- What to do if the event is not received (Timeout Policy)
- What to do if a compensation action, moving backward in the flow, is needed
- A well-defined business process has clear outcome that determines when the business process is completed. If the definition of the process causes it to stay pending for a long period of time, it’s probably not well defined
- The process does not have to be all automated and there could be some manual processes outside the process that move the process along, For EOI, the process might be waiting for

that to come in, but the actual act of process the EOI is outside the process.

Example: New Hire Benefits Enrollment:

When a client enters a new employee into their HR system that employee will be create in the benefits enrollment system in near real time so that the employee can log in and access the new hire enrollment instantly. And once the enrollment is complete the elections will be sent to the carriers in real time as well. At this point, an employee could just go to the drug store across the street and get their prescription filled because Rx provider would have their benefit information available already.

The business process is New Hire Enrollment.

The trigger is benefits administration system getting a Request for New Hire Enrollment via an API call from the employer. The outcome is Enrollment completed at the carrier(s). In this case, CVS for Rx. The steps defined in the template would be (timeouts and compensation policies/actions removed) 1. Email step: Send employee an email with link to enrollment - Pause until the employee logs in (trigger) 2. User Experience Step: Employee completes and confirms elections - the employeeSystem updates enrollment data base record 3. Integration Step(s): Carriers/Vendors notified of new elections 4. Email Step: Send employee email stating coverage available and they can enroll as member with carrier

At this point the employee can go to CVS. From a unit of work standpoint, Step 1 is one unit of work, followed by a pause, then Steps 2 is another with a commit point and then Step 3-4 as another unit of work. If something bad happens with the carriers, the business process will stay in the state of Step 2 completed and can be restarted to complete the remaining steps.

In general, there are two types of business processes:

1. Straight Through, where everything is done in one unit of work and it is all or nothing is there is a rollback
2. State Machine Based, where there are a set of state where the process can pause and then resume. These are used when there is time lags awaiting responses from a user. Event triggers are then used to restart the process

One of the main issues with using HTTP REST based APIs is that the API interaction is not guaranteed. The endpoint might not be available, or it might get lost in the network. This is particularly an issue if bulk or batch work is done via API. An HTTP REST API should complete in less than a second to avoid performance problems. This causes operational issues for both sides:consumer and providers. On the consumer side, this means they must remember where the flow is at and be able to restart at that point when the issue has been resolved. On the provider side, the API must be idempotent