A picture containing font, text, graphics, logo

Description automatically generated

Srik Subramanian, Susheel Jalali

srik@sunnyrewards.co, susheel.j@sunnyrewardsco

Abstract

This document describes the design for the Sunny Rewards Helios Platform Design and related Algorithms.

sunny rewards helios Platform Design

Version 0.76

Contents

[Terminology & Concepts 4](#_Toc137123809)

[Document History 4](#_Toc137123810)

[1 Overview 6](#_Toc137123811)

[2 System Architecture 7](#_Toc137123812)

[2.1 Microservice Layout and Dependency Restrictions 8](#_Toc137123813)

[2.2 Namespace 9](#_Toc137123814)

[2.3 Data Model 9](#_Toc137123815)

[2.4 Microservice Code Folder Structure 9](#_Toc137123816)

[3 Common Lib 11](#_Toc137123817)

[3.1 Data Model Base Structures 11](#_Toc137123818)

[3.1.1 BaseDto 11](#_Toc137123819)

[3.1.2 BaseResponseDto 11](#_Toc137123820)

[3.2 Repository Generic Interfaces 11](#_Toc137123821)

[4 Tenant Microservice 12](#_Toc137123822)

[4.1 Data Model 12](#_Toc137123823)

[4.1.1 Table: Customer 12](#_Toc137123824)

[4.1.2 Table: Sponsor 12](#_Toc137123825)

[4.1.3 Table: Attribute Name 13](#_Toc137123826)

[4.1.4 Table: Sponsor Attribute 13](#_Toc137123827)

[4.1.5 Table: Tenant 13](#_Toc137123828)

[4.1.6 Table: Tenant Attribute 14](#_Toc137123829)

[4.1.7 Table: Customer Attribute 14](#_Toc137123830)

[4.2 API 15](#_Toc137123831)

[5 User Microservice 16](#_Toc137123832)

[5.1 Data Model 16](#_Toc137123833)

[5.1.1 Table: Person 16](#_Toc137123834)

[5.1.2 Table: Role 16](#_Toc137123835)

[5.1.3 Table: Person Role 17](#_Toc137123836)

[5.1.4 Table: Consumer 17](#_Toc137123837)

~~[5.1.5](#_Toc137123838)~~~~[DO NOT CREATE FOR NOW: Table: Family](#_Toc137123838)~~ [18](#_Toc137123838)

[5.2 API 18](#_Toc137123839)

[5.2.1 Person Endpoints 18](#_Toc137123840)

[5.2.2 Consumer Endpoints 19](#_Toc137123841)

[6 Task Microservice 20](#_Toc137123842)

[6.1 Data Model 20](#_Toc137123843)

[6.1.1 Table: Task Type 20](#_Toc137123844)

[6.1.2 Table: Task 20](#_Toc137123845)

[6.1.3 Table: Reward Type 21](#_Toc137123846)

[￼](#_Toc137123847)

[6.1.5 Table: Consumer Task 22](#_Toc137123848)

[6.1.6 Table: Terms Of Service 22](#_Toc137123849)

[6.1.7 Table: Task Detail 23](#_Toc137123850)

[6.2 API 23](#_Toc137123851)

[6.2.1 Task Endpoints 23](#_Toc137123852)

[6.2.2 Task Reward Endpoints 24](#_Toc137123853)

[6.2.3 Consumer Task Endpoints 24](#_Toc137123854)

~~[7](#_Toc137123855)~~~~[(NOT USED NOW) RenderTemplate Microservice](#_Toc137123855)~~ [26](#_Toc137123855)

[7.1 Data Model 26](#_Toc137123856)

[7.1.1 Table: Section Template 26](#_Toc137123857)

[7.1.2 Table: Section Template Element 26](#_Toc137123858)

[7.2 API 27](#_Toc137123859)

[7.2.1 Render Template Info Endpoint 27](#_Toc137123860)

[8 (NOT USED NOW) RenderCMS Microservice 28](#_Toc137123861)

[8.1 Data Model 28](#_Toc137123862)

[8.1.1 Table: Section 28](#_Toc137123863)

[8.1.2 Table: Section Element 28](#_Toc137123864)

[8.1.3 Table: Task Section 29](#_Toc137123865)

[8.2 API 29](#_Toc137123866)

[8.2.1 Render Task Info Endpoint 29](#_Toc137123867)

[9 Wallet Microservice 31](#_Toc137123868)

[9.1 Data Model 31](#_Toc137123869)

[9.1.1 Table: Wallet Type 31](#_Toc137123870)

[9.1.2 Table: Wallet 31](#_Toc137123871)

[9.1.3 Table: Consumer Wallet 32](#_Toc137123872)

[9.1.4 Table: Transaction 32](#_Toc137123873)

[9.1.5 Table: Transaction Detail 33](#_Toc137123874)

[9.2 API 34](#_Toc137123875)

[9.2.1 Wallet Endpoints 34](#_Toc137123876)

[9.2.2 Consumer Wallet Endpoints 35](#_Toc137123877)

[9.2.3 Transaction Endpoints 36](#_Toc137123878)

[10 Cohort Microservice 38](#_Toc137123879)

[10.1 Data Model 38](#_Toc137123880)

[10.1.1 Table: Cohort 38](#_Toc137123881)

[10.1.2 Table: Consumer Cohort 38](#_Toc137123882)

[10.1.3 Table: Cohort Task Reward 38](#_Toc137123883)

[10.2 API 39](#_Toc137123884)

[11 BFF Microservice 40](#_Toc137123885)

[11.1 API 40](#_Toc137123886)

[11.1.1 GET Consumer Summary 40](#_Toc137123887)

[11.1.2 GET Consumer Task List 41](#_Toc137123888)

[11.1.3 GET Consumer Transaction List 41](#_Toc137123893)

[11.1.4 POST Redeem 41](#_Toc137123894)

[11.1.5 POST Consumer Task Enroll 41](#_Toc137123895)

[11.1.6 POST Consumer Task Update 41](#_Toc137123896)

[12 UI 42](#_Toc137123897)

[12.1 App Namespace Repo and Folders 42](#_Toc137123898)

[12.2 Home 42](#_Toc137123899)

[12.2.1 Components 43](#_Toc137123900)

[12.3 Launch Page 45](#_Toc137123901)

[12.4 Task/Activity Popup 46](#_Toc137123902)

[12.5 Task/Activity List Page 46](#_Toc137123903)

[12.6 Transaction List Page 46](#_Toc137123904)

[12.7 Redeem Flow Start Page 47](#_Toc137123905)

[13 Object Naming Conventions 47](#_Toc137123906)

[13.1 Codes in DB 47](#_Toc137123907)

# Terminology & Concepts

|  |  |  |
| --- | --- | --- |
| Term | Synonyms | Comments |
| Helios | Sunny Rewards Application & Platform code name |  |
| API | Application programming interface |  |
| DLL | Dynamically Linked Library | C# .NET Assembly |
| PK | Primary Key |  |
| FK | Foreign Key |  |
| GUID | Globally Unique Identifier | Ensured by the system/ algorithm |
| SAML | Security Assertion Markup Language |  |
| Customer | Enterprise Customer | Enterprise Customer of Sunny Rewards (Health Plan Insurance/Provider) |
| Sponsor |  | Sponsoring client companies of the Enterprise Customer |
| Tenant |  | A division of employees (of sponor) + plan year which is the segmenting field for all other data tables |
| Person |  | An individual |
| Consumer |  | A person+tenant combination |
| Task |  | An activity that when completed by a Consumer, rewards them the configured amount of money into their wallet |
| Wallet |  | A holder of rewarded cash |
| Cohort |  | Arbitrary groupings of people similar characteristics including health conditions |

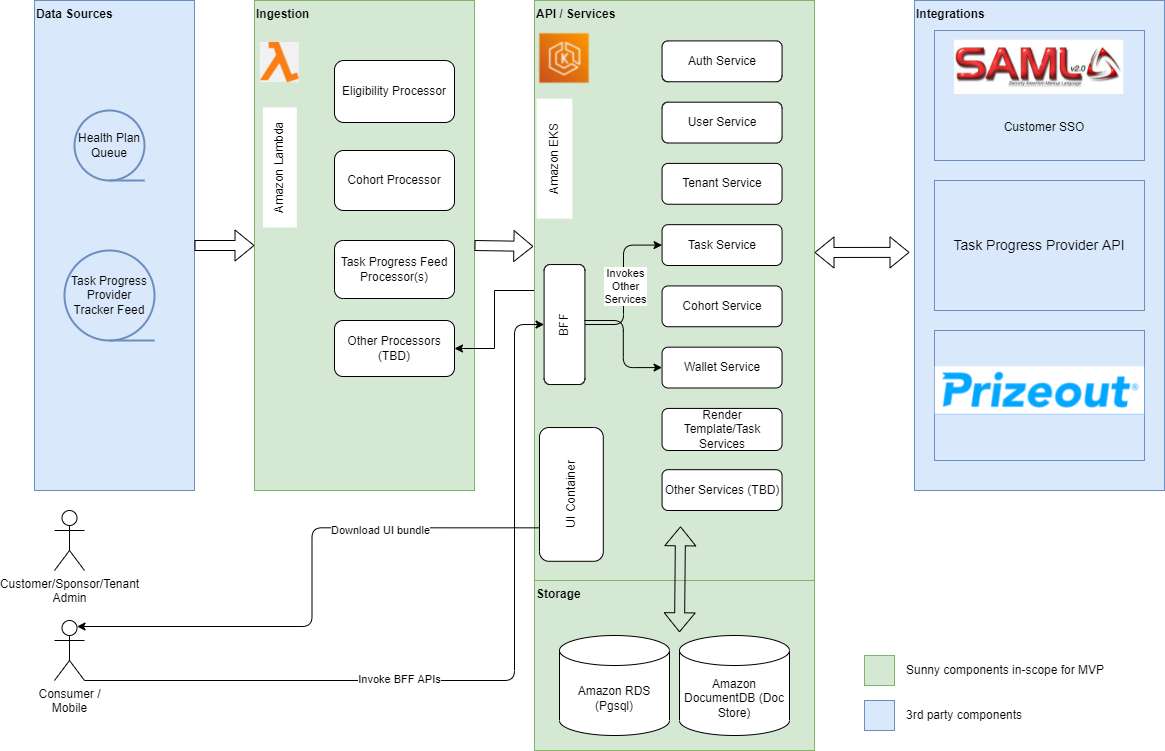
# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Comments |
| 05/16/2023 | 0.1 | Srik Subramanian | Initial revision |
| 05/20/2023 | 0.2 | Susheel Jalali | Added the data model table definitions under all the DB Schemas. |
| 05/21/2023 | 0.3 | Srik Subramanian | Task, Wallet, RenderTemplate, RenderTask, BFF APIs |
| 05/24/2023 | 0.4 | Susheel Jalali | task.task\_reward cohort.cohort\_task\_reward |
| 05/26/2023 | 0.5 | SS | render\_task.task\_section changes,  renamed render\_task to render\_cms,  added required User APIs, modifed user.person and wallet.wallet tables |
| 05/28/2023 | 0.6 | Srik Subramanian | Added 12.2 launch page, 2.4 folder structure for .NET core solutions |
| 05/29/2023 | 0.7 | Srik Subramanian | Modified index requirements in all tables/schemas |
| 05/30/2023 | 0.71 | Srik Subramanian | Updated 12.1, 12.3 |
| 06/01/2023 | 0.72 | Srik Subramanian | Updated \*\_code columns to be varchar(50), and all create, update user columns to be varchar(50), removed update fields from wallet.transaction and transaction\_detail |
| 06/01/2023 | 0.73 | Srik Subramanian | Added code value naming convention sec 13.1 |
| 06/04/2023 | 0.74 | Srik Subramanian | Fixed delete\_nbr to be not nullable in all tables, added more code value naming conv |
| 06/07/2023 | 0.75 | Srik Subramanian | Changes to Task schema (review enabled) |
| 06/08/2023 | 0.76 | Srik Subramanian | Added UI implementation sections under 12.2.1 Components and 12.4 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Overview

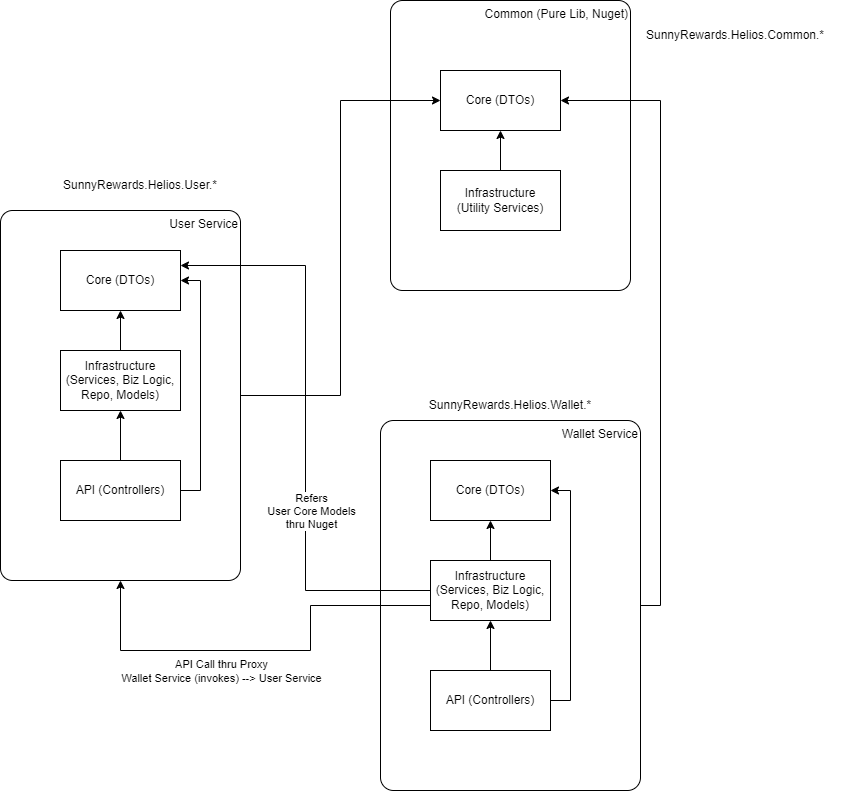
Sunny Rewards Helios is an enterprise class platform that aims to provide a whitelabel environment for Insurance Plan Administration Companies to integrate and provide a redeemable rewards-based incentive to foster better health practices in their subscriber communities. The system is designed to be multi-tenant. This document contains the full system design of the platform including all tiers. This document will be updated as development proceeds in Program Increments.

# System Architecture



The above illustration is a high-level expected architecture – internal components and the design of specific layers may change during implementation.

## Microservice Layout and Dependency Restrictions



The above illustration defines the internal structure of every microservice and allowed dependencies between various microservices.

Every microservice is structured as follows:

* Lives in its own git repo
* Build and deploy is controlled through github workflow files within the repo
* Is a .NET Core Solution
* Except for Common which is a pure library (published as a Nuget), each microservice consists of atleast 3 .NET projects each producing a DLL
  + Core (will be published also a Nuget in a custom Nuget repo) – shall contain all DTOs, externally accessed Interfaces
  + Infrastructure – shall contain implementation of all interfaces, internal business logic implemented as services and Repository DAO layer including Models representing DB tables
  + API – shall contain only Controllers that implement end-points for all the APIs exposed by the microservice
* For example – the User microservice will be structured internally as:
  + SunnyRewards.Helios.User.Core (will be published as Nuget in a custom Nuget repo)
  + SunnyRewards.Helios.User.Infrastructure
  + SunnyRewards.Helios.User.API
  + Deployed URL: <https://user.dev.sunnyrewards.co/api/>...
  + The API project exposes Swagger doc / invoker

Allowable dependencies are as follows:

* Each microservice API project will depend on its own Core and Infrastructure projects
* Each Infrastructure project will depend on its own Core project
* Additionally, each Infrastructure project can depend on other microservice Core Nugets and the Common Nuget

## Namespace

Use namespace prefix in C# for all projects as: SunnyRewards.Helios.<microservice\_name>…

## Data Model

Each microservice shall store data managed by it in its own schema. Microservices shall not access schemas of other microservices. All data retrieval and updates into a schema shall be only possible through the API endpoints provided by the associated microservice.

Apart from the fields listed below, each table shall have the following fields (columns) and behavior:

* create\_ts DateTime NOT NULL
* update\_ts DateTime
* create\_user char(40) NOT NULL
* update\_user char(40)
* delete\_nbr Long (== 0 if not soft deleted, == PK\_value if soft deleted)
* Uniqueness criteria for any U field will be always field+delete\_ nbr

Regarding indexes, possible ones are mentioned, to be selectively implemented, considering judiciously the volumes and access patterns on each table.

## Microservice Code Folder Structure

Following disk folder structure shall be followed for all microservices:

# Common Lib

A common library shall be created to hold definitions and services that are used across all microservices.

Namespace: SunnyRewards.Helios.Common.Core

Repository name: helios-common-lib

## Data Model Base Structures

### BaseDto

BaseDto is the base of all DTOs interchanged across services and BFF/UI.

Members:

* DateTime CreateTs
* DateTime UpdateTs
* String CreateUser
* String UpdateUser
* Long DeleteNbr

### BaseResponseDto

BaseResponseDto is the base of all responses sent by APIs – this allows all API end-points to return error information in a consistent manner across all services.

Members:

* String ErrorCode
* String ErrorMessage
* String ErrorDescriptionType // TEXT, JSON
* String ErrorDescription // more info to be interpreted based on ErrorDescriptionType

## Repository Generic Interfaces

* IRepository<Model>

# Tenant Microservice

## Data Model

Schema: *tenant*

### Table: Customer

Customer table contains the top-level enterprise customer of Sunny Rewards – this is usually a health insurance plan provider.

Table: *customer*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| customer\_id | long | PK | Not null |
| customer\_code | varchar(50) | GUID | Not null, unique |
| customer\_name | varchar(80) |  | Not null |
| customer\_description | varchar(1024) |  |  |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: customer\_code + delete\_nbr
* Unique: customer\_name + delete\_nbr

### Table: Sponsor

Table: *sponsor*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| sponsor\_id | long | PK | Not null |
| customer\_id | long | FK | Not null |
| sponsor\_code | varchar(50) | GUID | Not null, unique |
| sponsor\_name | varchar(80) |  | Not null |
| sponsor\_description | varchar(1024) |  |  |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: sponsor\_code + delete\_nbr
* Unique: sponsor\_name + delete\_nbr

### Table: Attribute Name

Table: *attribute\_name*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Type | Description | Notes | Default |
| attribute\_name\_id | long | PK | Not null |  |
| attribute\_name | char (40) |  | Not null |  |
| create\_ts | DateTime |  | Not null |  |
| update\_ts | DateTime |  |  |  |
| create\_user | varchar(50) |  | Not null |  |
| update\_user | varchar(50) |  |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted |  | Not null |

**Indexes:**

* Unique: attribute\_name + delete\_nbr

### Table: Sponsor Attribute

Table: *sponsor\_attribute*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Type | Description | Notes | Default |
| sponsor\_attribute\_id | long | PK | Not null |  |
| sponsor\_id | long | FK1 | Not null |  |
| attribute\_name\_id | long | FK2 | Not null |  |
| attribute\_ordinal | int | def 0 | Not null | 0 |
| attribute\_value | varchar(1024) |  |  |  |
| create\_ts | DateTime |  | Not null |  |
| update\_ts | DateTime |  |  |  |
| create\_user | varchar(50) |  | Not null |  |
| update\_user | varchar(50) |  |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted |  | Not null |

**Indexes:**

* Unique: sponsor\_id + attribute\_name\_id + delete\_nbr

### Table: Tenant

Table: *tenant*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| tenant\_id | long | PK | Not null |
| sponsor\_id | long | FK | Not null |
| tenant\_code | varchar(50) | GUID | Not null, unique |
| plan\_year | Int |  | Not null |
| period\_start\_ts | DateTime |  | Not null |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: tenant\_code + delete\_nbr

### Table: Tenant Attribute

Table: *tenant\_attribute*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Type | Description | Notes | Default |
| tenant\_attribute\_id | long | PK | Not null |  |
| tenant\_id | long | FK1 | Not null |  |
| attribute\_name\_id | long | FK2 | Not null |  |
| attribute\_ordinal | int | def 0 | Not null | 0 |
| attribute\_value | varchar(1024) |  |  |  |
| create\_ts | DateTime |  | Not null |  |
| update\_ts | DateTime |  |  |  |
| create\_user | varchar(50) |  | Not null |  |
| update\_user | varchar(50) |  |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted |  | Not null |

**Indexes:**

* Unique: tenant\_id + attribute\_name\_id + delete\_nbr

### Table: Customer Attribute

Table: *customer\_attribute*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Type | Description | Notes | Default |
| customer\_attribute\_id | long | PK | Not null |  |
| customer\_id | long | FK1 | Not null |  |
| attribute\_name\_id | long | FK2 | Not null |  |
| attribute\_ordinal | int | def 0 | Not null | 0 |
| attribute\_value | varchar(1024) |  |  |  |
| create\_ts | DateTime |  | Not null |  |
| update\_ts | DateTime |  |  |  |
| create\_user | varchar(50) |  | Not null |  |
| update\_user | varchar(50) |  |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted |  | Not null |

**Indexes:**

* Unique: customer\_id + attribute\_name\_id + delete\_nbr

## API

No APIs required for tenant in the MVP. Data within the tentant schema is referred to in other schemas through application-level “code” fields used as foreign keys.

# User Microservice

## Data Model

Schema: *huser*

### Table: Person

Table: *person*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| person\_id | long | PK | Not null |
| person\_code | varchar(50) | GUID | Not null, unique |
| first\_name | varchar(80) |  | Not null |
| last\_name | varchar(80) |  | Not null |
| language\_code | varchar(5) | en-US only initially | Not null |
| member\_since | DateTime |  | Not null |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |
| email | varchar(100) |  | Not null |

**Indexes:**

* Unique: person\_code + delete\_nbr

### Table: Role

Table: *role*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| role\_id | long | PK | Not null |
| role\_code | varchar(50) | GUID | Not null, unique |
| role\_name | varchar(80) |  | Not null |
| role\_description | varchar(1024) |  |  |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: role\_code + delete\_nbr
* Unique: role\_name + delete\_nbr

### Table: Person Role

Table: person\_role

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| person\_role\_id | long | PK | Not null |
| person\_id | long | FK1 | Not null |
| role\_id | long | FK2 | Not null |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: person\_id + role\_id + delete\_nbr

### Table: Consumer

Table: *consumer*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Type | Description | Notes | Default |
| consumer\_id | long | PK | Not null |  |
| person\_id | long | FK | Not null |  |
| tenant\_code | varchar(50) |  | Not null |  |
| consumer\_code | varchar(50) | GUID | Not null |  |
| registered | bit | def false | Not null | false |
| eligible | bit | def false | Not null | false |
| registration\_ts | DateTime |  |  |  |
| eligible\_start\_ts | DateTime |  |  |  |
| eligible\_end\_ts | DateTime |  |  |  |
| create\_ts | DateTime |  | Not null |  |
| update\_ts | DateTime |  |  |  |
| create\_user | varchar(50) |  | Not null |  |
| update\_user | varchar(50) |  |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted |  | Not null |

**Indexes:**

* Unique: consumer\_code + delete\_nbr

### ~~DO NOT CREATE FOR NOW: Table: Family~~

~~Table:~~ *~~family~~*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ~~Field~~ |  | ~~Type~~ | ~~Description~~ | ~~Notes~~ | ~~Default~~ |
| ~~family\_id~~ |  | ~~long~~ | ~~PK~~ | ~~Not null~~ |  |
| ~~person\_id~~ |  | ~~long~~ | ~~FK~~ | ~~Not null~~ |  |
| ~~family\_code~~ |  | ~~varchar(50)~~ | ~~U~~ | ~~Not null~~ |  |
| ~~head\_of\_household~~ |  | ~~bit~~ |  | ~~Not null~~ |  |
| ~~create\_ts~~ |  | ~~DateTime~~ |  | ~~Not null~~ |  |
| ~~update\_ts~~ |  | ~~DateTime~~ |  |  |  |
| ~~create\_user~~ |  | ~~varchar(50)~~ |  | ~~Not null~~ |  |
| ~~update\_user~~ |  | ~~varchar(50)~~ |  |  |  |
| ~~delete\_nbr~~ |  | ~~long~~ | ~~== 0 if not soft deleted,~~  ~~== PK\_value if soft deleted~~ |  | ~~Not null~~ |

* ~~Unique: family\_code + delete\_nbr~~

## API

Namespace prefix: SunnyRewards.Helios.User.\*

Repository name: helios-user-api

### Person Endpoints

#### CRUD

Only GET required for now

|  |  |
| --- | --- |
| GET | /api/v1/person/{person-id} |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input |  |  |  |
| Output | PersonDto | Response |  |

PersonDto (extends BaseDto) members:

* Long PersonId
* String PersonCode
* String FirstName
* String LastName
* DateTime MemberSince

### Consumer Endpoints

#### GET Consumer

|  |  |
| --- | --- |
| POST (posted GET) | /api/v1/get-consumer |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input | GetConsumerRequestDto | Request |  |
| Output | GetConsumerResponseDto | Response |  |

GetConsumerRequestDto members:

* String ConsumerCode

GetConsumerResponseDto (extends BaseResponseDto):

* ConsumerDto Consumer

ConsumerDto (extends BaseDto):

* Long ConsumerId
* Long PersonId
* String TenantCode
* String ConsumerCode
* Bool Registered
* Bool Eligible
* DateTime RegistrationTs
* DateTime EligibleStartTs
* DatTime EligibleEndTs

# Task Microservice

Task microservice provides the Helios platform with all information storage and tracking for various tasks/activities available and executed by the Consumer.

## Data Model

Schema: *task*

### Table: Task Type

Table: *task\_type*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| task\_type\_id | long | PK | Not null |
| task\_type\_code | varchar(50) | GUID | Not null, unique |
| task\_type\_name | varchar(80) |  | Not null |
| task\_type\_description | varchar(1024) |  |  |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: task\_type\_code + delete\_nbr
* Unique: task\_type\_name + delete\_nbr

### Table: Task

Table: *task*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| task\_id | long | PK | Not null |
| task\_type\_id | long | FK | Not null |
| task\_code | varchar(50) | GUID | Not null, unique |
| task \_name | varchar(80) |  | Not null |
| self\_report | bit | def 0 | Not null |
| confirm\_report | bit | def 0 | Not null |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: task\_code + delete\_nbr

### Table: Reward Type

Table: *reward\_type*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| reward\_type\_id | long | PK | Not null |
| reward\_type\_name | varchar(80) |  | Not null |
| reward\_type\_description | varchar(1024) |  | Not null |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: reward\_type\_name + delete\_nbr

### Table: Task Reward

Table: *task\_reward*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| task\_reward\_id | long | PK | Not null |
| task\_id | long | FK1 | Not null |
| reward\_type\_id | long | FK2 | Not null, For mvp, simple JSON. In future other types also. Or more complex JSON. |
| tenant\_code | varchar(50) |  | Not null |
| task\_reward\_code | varchar(50) | GUID | Not null, unique |
| reward | varchar(1024) |  | Not null, for MVP, reward will be: {  “type”: “money”,  “amount”: <integer $ amount>  } |
| min\_task\_duration | Int |  |  |
| max\_task\_duration | Int |  |  |
| expiry | DateTime |  |  |
| priority | Int |  | Not null,  only for PI-1 - to replace by cohort after PI-1 |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: task\_ reward\_code + delete\_nbr

### Table: Consumer Task

Table: *consumer\_task*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| consumer\_task\_id | long | PK | Not null |
| task\_id | long | FK | Not null |
| tenant\_code | varchar(50) |  | Not null |
| consumer\_code | varchar(50) | GUID | Not null |
| task\_status | varchar(20) | TODO, ENROLLED,  IN\_PROGRESS,  PENDING\_CONFIRMATION,  COMPLETED,  INCOMPLETE,  ABANDONED | Not null |
| progress | Int |  | Not null |
| notes | varchar(1024) |  | Not null |
| task\_start\_ts | DateTime |  |  |
| task\_complete\_ts | DateTime |  |  |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | Long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: task\_id + consumer\_code + delete\_nbr

### Table: Terms Of Service

Table: *terms\_of\_service*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| terms\_of\_service\_id | long | PK | Not null |
| terms\_of\_service\_text | varchar(max) |  | Not null |
| language\_code | varchar(5) |  | Not null, en-US |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

task *task\_detail*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| task\_detail\_id | long | PK | Not null |
| task\_id | long | FK | Not null |
| language\_code | varchar(5) |  | Not null, en-US |
| task \_header | varchar(255) |  | Not null |
| task\_description | varchar(max) |  | Not null |
| terms\_of\_service\_id | long | FK | Nullable FK to terms\_of\_service table entry |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

## API

Namespace prefix: SunnyRewards.Helios.Task.\*

Repository name: helios-task-api

### Task Endpoints

#### GET

GET for given TaskId

|  |  |
| --- | --- |
| GET | /api/v1/task/{task\_id} |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input | Long | TaskId |  |
| Output | TaskDto | Response |  |

TaskDto (extends BaseDto) members:

* Long TaskId
* Long TaskTypeId
* String TaskCode
* String TaskName
* String TaskHeader // populate from task\_detail.task\_header
* String TaskDescription // populate from task\_detail.task\_description
* Bool SelfReport
* Bool ConfirmReport
* String TermsOfService // if task\_detail.terms\_of\_service\_id FK field is not null, load   
   // from terms\_of\_service.terms\_of\_service\_text

### Task Reward Endpoints

#### Find TaskReward

Find all TaskReward for given TenantCode

|  |  |
| --- | --- |
| GET | /api/v1/find-task-rewards |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input | FindTaskRewardRequestDto | Request |  |
| Output | FindTaskRewardResponseDto | Response |  |

FindTaskRewardRequestDto:

* String TenantCode

FindTaskRewardResponseDto (extends BaseResponseDto):

* List<TaskRewardDto> TaskReward

TaskRewardDto (extends BaseDto) members:

* Long TaskRewardId
* Long TaskId
* Long RewardTypeId
* String TenantCode
* String TaskRewardCode
* String Reward
* Int MinTaskDuration
* Int MaxTaskDuration

### Consumer Task Endpoints

#### CRUD

|  |  |
| --- | --- |
| GET | /api/v1/consumer-task/{consumer-task-id} |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input |  |  |  |
| Output | ConsumerTaskDto | Response |  |

ConsumerTaskDto (extends BaseDto) members:

* Long ConsumerTaskId
* Long TaskId
* String TenantCode
* String ConsumerCode
* String TaskStatus
* Int Progress
* String Notes
* DateTime TaskStartTs
* DateTime TaskCompleteTs

|  |  |
| --- | --- |
| POST | /api/v1/consumer-task |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input | ConsumerTaskDto |  |  |
| Output | ConsumerTaskDto |  | With ConsumerTaskId filled |

|  |  |
| --- | --- |
| PUT | /api/v1/consumer-task |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input | ConsumerTaskDto |  |  |
| Output |  |  |  |

##### Algorithm

Special case of business logic required for PUT of ConsumerTask:

* When a ConsumerTask is marked COMPLETED, the appropriate TaskReward needs to be looked up and the Reward needs to be awarded into the appropriate Wallet through an API call from the internal service to Wallet API

# ~~(NOT USED NOW) RenderTemplate Microservice~~

## Data Model

Schema: *render\_template*

### Table: Section Template

Table: *section\_template*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| section\_template\_id | long | PK | Not null |
| section\_template\_code | varchar(50) | GUID | Not null, unique |
| template\_name | varchar(80) |  | Not null |
| template\_ref\_name | varchar(80) |  | Not null |
| template\_attribute | varchar(max) |  |  |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | Long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: section\_template\_code + delete\_nbr

### Table: Section Template Element

Table: *section\_template\_element*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| section\_template\_element\_id | Long | PK | Not null |
| section\_template\_id | Long | FK | Not null |
| section\_template\_element\_code | varchar(50) | GUID | Not null, unique |
| element\_type | varchar(20) | TEXT, HTML, IMAGE, VIDEO, URL | Not null |
| element\_name | varchar(80) |  | Not null |
| element\_attribute | varchar(max) |  |  |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | Long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: section\_template\_element\_code + delete\_nbr

## API

Namespace prefix: SunnyRewards.Helios.RenderTemplate.\*

Repository name: helios-render-template-api

### Render Template Info Endpoint

#### POST

|  |  |
| --- | --- |
| POST (posted GET) | /api/v1/render-template-info |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input | GetRenderTemplateRequestDto | Request |  |
| Output | GetRenderTemplateResponseDto | Response |  |

GetRenderTemplateRequestDto members:

* String SectionTemplateCode

GetRenderTemplateResponseDto (extends BaseResponseDto) members:

* SectionTemplateDto SectionTemplate
* List<SectionTemplateElementDto> SectionTemplateElements // elements within the   
   // template

SectionTemplateDto (extends BaseDto) members:

* Long SectionTemplateId
* String SectionTemplateCode
* String TemplateName
* String TemplateRefName // Component logical name within UI code
* String TemplateAttribute // JSON

SectionTemplateElementDto (extends BaseDto) members:

* Long SectionTemplateElementId
* Long SectionTemplateId
* String SectionTemplateElementCode
* String ElementType
* String ElementName
* String ElementAttribute // JSON

# (NOT USED NOW) RenderCMS Microservice

## Data Model

Schema: *render\_cms*

### Table: Section

Table: *section*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| section\_id | long | PK | Not null |
| section\_template\_code | varchar(50) |  | Not null |
| template\_attribute\_override | varchar(max) | provides a template-level override to rendering attributes |  |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

### Table: Section Element

Table: *section\_element*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| section\_element\_id | long | PK | Not null |
| section\_id | long | FK | Not null |
| section\_template\_element\_code | varchar(50) |  | Not null |
| element\_index | Int | If not null, then used as a scalar index for an array of values for array type elements |  |
| element\_value | varchar(max) | * for elements of type TEXT, HTML, will contain the element data for this task * for elements of type IMAGE, VIDEO (large resources), will point to stored content * for elements type URL, will contain the external URL |  |
| element\_attribute\_override | varchar(max) | provides an element-level override to rendering attributes |  |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

### Table: Task Section

Table: task\_*section*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| task\_section\_id | long | PK | Not null |
| section\_id | long | FK | Not null |
| task\_code | varchar(50) |  | Not null |
| sequence\_no | Int |  | Not null, def 0 |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: section\_id + task\_code + delete\_nbr

## API

Namespace prefix: SunnyRewards.Helios.RenderCms.\*

Repository name: helios-render-cms-api

### Render Task Info Endpoint

#### POST

|  |  |
| --- | --- |
| POST (posted GET) | /api/v1/render-task-info |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input | GetRenderTaskInfoRequestDto | Request |  |
| Output | GetRenderTaskInfoResponseDto | Response |  |

GetRenderTaskInfoRequestDto members:

* String TaskCode

GetRenderTaskInfoResponseDto (extends BaseResponseDto) members:

* Dictionary<Long, RenderTaskInfoEntryDto> RenderTaskInfo // map: SectionId    
   // RenderTaskInfoEntryDto

RenderTaskInfoEntryDto members:

* SectionDto Section
* SectionElementDto SectionElements

SectionDto (extends BaseDto) members:

* Long SectionId
* String SectionTemplateCode
* String TemplateAttributeOverride // JSON

SectionElementDto (extends BaseDto) members:

* Long SectionElementId
* Long SectionId
* String TemplateElementDefinitionCode
* String ElementValue
* String ElementAttributeOverride // JSON

# Wallet Microservice

Wallet microservice provides a double-entry based account keeping function to the Helios platform. This form of account keeping provides easy reporting and reconciliation functions.

## Data Model

Schema: *wallet*

### Table: Wallet Type

Table: *wallet\_type*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| wallet\_type\_id | long | PK | Not null |
| wallet\_type\_code | varchar(50) | GUID | Not null, unique |
| wallet\_type\_name | varchar(80) | For MVP 3 types: REWARD, REDEMPTION, FUND | Not null |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: wallet\_type\_code + delete\_nbr

### Table: Wallet

Table: *wallet*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| wallet\_id | long | PK | Not null |
| wallet\_type\_id | long |  | Not null |
| customer\_code | varchar(50) |  | Not null, if sponsor\_code and tenant\_code are both null, this is a customer level master wallet |
| sponsor\_code | varchar(50) |  | If not null and tenant\_code is null, this is a sponsor level master wallet |
| tenant\_code | varchar(50) |  | If not null, this is a tenant level master wallet |
| wallet\_code | varchar(50) |  | Not null, unique |
| master\_wallet | Bit | def 0 | Not null |
| wallet\_name | varchar(80) |  | Not null |
| active | bit | def 0 | Not null |
| active\_start\_ts | DateTIme |  |  |
| active\_end\_ts | DateTIme |  |  |
| balance | Double |  | Not null |
| total\_earned | Double | Total rewarded earnings so far | Not null |
| earn\_maximum | Double |  | Not null |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | Long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: wallet\_code + delete\_nbr

### Table: Consumer Wallet

Table: *consumer\_wallet*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| consumer\_wallet\_id | long | PK | Not null |
| wallet\_id | long | FK | Not null |
| tenant\_code | varchar(50) |  | Not null |
| consumer\_code | varchar(50) |  | Not null |
| consumer\_role | char(1) |  | Not null |
| earn\_maximum | Double |  | Not null |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | Long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: wallet\_id + consumer\_code + delete\_nbr

### Table: Transaction

Table: *transaction*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| transaction\_id | long | PK | Not null |
| wallet\_id | long | FK | Not null |
| transaction\_code | varchar(50) |  | Not null, this is not unique – there will always be 2 entries in this table with the same transaction\_code due to double entry account keeping |
| transaction\_type | char(1) | A == Add, S == Subtract | Not null |
| previous\_balance | Double |  | Not null |
| transaction\_amount | Double |  | Not null |
| balance | double |  | Not null |
| prev\_wallet\_txn\_code | varchar(100) | GUID | Not null |
| create\_ts | DateTime |  | Not null |
| create\_user | varchar(50) |  | Not null |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: transaction\_code + transaction\_type + delete\_nbr
* Unique: prev\_wallet\_txn\_code + delete\_nbr

### Table: Transaction Detail

Table: *transaction***\_***detail*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| transaction\_detail\_id | long | PK | Not null |
| transaction\_id | long | FK | Not null |
| transaction\_detail\_type | char(20) | REWARD, REDEMPTION, ADJUSTMENT, RETURN, FUND | Not Null |
| consumer\_code | varchar(50) |  |  |
| task\_code | varchar(50) |  |  |
| notes | varchar(1024) |  |  |
| redemption\_ref | varchar(80) |  |  |
| redemption\_item\_description | varchar(1024) |  |  |
| create\_ts | DateTime |  | Not null |
| create\_user | varchar(50) |  | Not null |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: transaction\_id + delete\_nbr

## API

Namespace prefix: SunnyRewards.Helios.Wallet.\*

Repository name: helios-wallet-api

### Wallet Endpoints

#### CRUD

|  |  |
| --- | --- |
| GET | /api/v1/wallet/{wallet-id} |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input |  |  |  |
| Output | WalletDto | Response |  |

WalletDto (extends BaseDto) members:

* Long WalletId
* Long WalletTypeId
* String CustomerCode
* String SponsorCode
* String TenantCode
* String WalletCode
* Bool MasterWallet
* String WalletName
* Bool Active
* DateTime ActiveStartTs
* DateTime ActiveEndTs
* Double Balance
* Double EarnMaximum
* Double TotalEarned
* Double LeftToEarn // this is a computed field (== EarnMaximum – TotalEarned)

|  |  |
| --- | --- |
| POST | /api/v1/wallet |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input | WalletDto |  |  |
| Output | WalletDto |  | With WalletId filled |

##### Algorithm

Special case of business logic required for POST of Wallet:

* POST – initial creation – enforce Balance == 0
* There is no Wallet PUT – must be triggered only through Transaction POST

#### Get Master Wallet

|  |  |
| --- | --- |
| POST (posted GET) | /api/v1/wallet/get-master-wallet |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input | GetMasterWalletRequestDto | Request |  |
| Output | GetMasterWalletResponseDto | Response |  |

GetMasterWalletRequestDto members:

* String WalletTypeName // (REWARD, REDEMPTION, FUND etc)
* String TenantCode
* String SponsorCode // (not used in MVP)
* String CustomerCode // (not used in MVP)

GetMasterWalletResponseDto (extends BaseResponseDto) members:

* WalletDto Wallet

### Consumer Wallet Endpoints

#### CRUD

|  |  |
| --- | --- |
| GET | /api/v1/consumer-wallet/{consumer-wallet-id} |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input |  |  |  |
| Output | ConsumerWalletDto | Response |  |

ConsumerWalletDto (extends BaseDto) members:

* Long ConsumerWalletId
* Long WalletId
* String TenantCode
* String ConsumerCode
* String ConsumerRole
* Double EarnMaximum

|  |  |
| --- | --- |
| POST | /api/v1/consumer-wallet |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input | ConsumerWalletDto |  |  |
| Output | ConsumerWalletDto |  | With ConsumerWalletId filled |

|  |  |
| --- | --- |
| PUT | /api/v1/consumer-wallet |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input | ConsumerWalletDto |  |  |
| Output |  |  |  |

#### Find Consumer Wallet

|  |  |
| --- | --- |
| POST (posted GET) | /api/v1/consumer-wallet/find-consumer-wallet |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input | FindConsumerWalletRequestDto | Request |  |
| Output | FindConsumerWalletResponseDto | Response |  |

FindConsumerWalletRequestDto members:

* String ConsumerCode

Note: for MVP, the List size will be == 1 as each consumer will have only one REWARD type wallet

FindConsumerWalletResponseDto (extends BaseResponseDto) members:

* List<ConsumerWalletDto> ConsumerWallets

### Transaction Endpoints

#### POST Double Entry Transaction

|  |  |
| --- | --- |
| POST | /api/v1/transaction/de-transaction |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input | PostDETransactionRequestDto | Request |  |
| Output | BaseResponseDto | Response |  |

PostDETransactionRequestDto members:

* TransactionEntryDto SubtractEntry
* TransactionEntryDto AddEntry

TransactionEntryDto members:

* TransactionDto Transaction
* TransactionDetailDto TransactionDetail

TransactionDto (extends BaseDto) members:

* Long TransactionId
* Long WalletId
* String TransactionCode
* String TransactionType
* Double PreviousBalance
* Double TransactionAmount
* Double Balance
* String PrevWalletTxnCode

TransactionDetailDto (extends BaseDto) members:

* Long TransactionDetailId
* Long TransactionId
* String TransactionDetailType
* String ConsumerCode
* String TaskCode
* String Notes
* String RedemptionRef
* String RedemptionItemDescription

#### GET Recent Transactions

|  |  |
| --- | --- |
| POST (posted GET) | /api/v1/transaction/recent |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input | GetRecentTransactionRequestDto | Request |  |
| Output | GetRecentTransactionResponseDto List<TransactionEntryDto> | Response |  |

GetRecentTransactionDto members:

* Long WalletId
* Int Count // for most recent, send Count = 1

GetRecentTransactionResponseDto (extends BaseResponseDto) members:

* List<TransactionEntryDto> Transactions

# Cohort Microservice

## Data Model

Schema: *cohort*

### Table: Cohort

Table: *cohort*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| cohort\_id | long | PK | Not null |
| cohort\_code | varchar(50) | GUID | Not null, unique |
| cohort\_name | varchar(80) |  | Not null |
| cohort\_description | varchar(1024) |  |  |
| parent\_cohort\_id | Long |  |  |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | Long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: cohort\_code + delete\_nbr

### Table: Consumer Cohort

Table: *consumer\_cohort*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| consumer\_cohort\_id | long | PK | Not null |
| cohort\_id | long | FK | Not null |
| tenant\_code | Varchar(50) |  | Not null |
| consumer\_code | varchar(50) |  | Not null |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: cohort\_id + consumer\_code + delete\_nbr

### Table: Cohort Task Reward

Table: *cohort\_task\_reward*

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Notes |
| cohort\_task \_id | long | PK | Not null |
| cohort\_id | long | FK | Not null |
| task\_reward\_code | varchar(50) |  | Not null |
| recommended | Bit | def 0 | Not null |
| priority | Int |  | Not null |
| create\_ts | DateTime |  | Not null |
| update\_ts | DateTime |  |  |
| create\_user | varchar(50) |  | Not null |
| update\_user | varchar(50) |  |  |
| delete\_nbr | Long | == 0 if not soft deleted,  == PK\_value if soft deleted | Not null |

**Indexes:**

* Unique: cohort\_id + task\_reward\_code + delete\_nbr

## API

Namespace prefix: SunnyRewards.Helios.Cohort.\*

Repository name: helios-cohort-api

# BFF Microservice

The BFF microservice shall be a stateless API created only to serve the UI. It will be a pure pass through to API calls into various other microservices, packaging the information into more macroscopic structures to avoid multiple roundtrips between the front-end and API tiers. The UI shall be allowed to invoke only the BFF microservice.

## API

Namespace prefix: SunnyRewards.Helios.Bff.\*

Repository name: helios-bff-api

### GET Consumer Summary

|  |  |
| --- | --- |
| POST | /api/v1/bff/consumer-summary |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Name | Comments |
| Input | GetConsumerSummaryRequestDto | Request |  |
| Output | GetConsumerSummaryResponseDto |  |  |

GetConsumerSummaryRequestDto members:

* Long ConsumerId
* String ConsumerCode // one of these need to be filled

GetConsumerSummaryResponseDto (extends BaseResponseDto) members:

* ConsumerPersonDetailDto ConsumerPersonDetail
* ConsumerWalletDetailDto ConsumerWalletDetail
* List<ConsumerTaskDto> PendingTasks
* List<ConsumerTaskDto> CompletedTasks // recent X number of completed tasks
* List<TaskRewardDto> AvailableTasks // tasks ordered by priority

ConsumerPersonDetailDto members:

* String FirstName
* String LastName
* DateTime MemberSince

ConsumerWalletDetailDto members:

* List<WalletDto> Wallets // for MVP, list size == 1
* List<TransactionDto> RecentTransactions // X number of most recent txn (last provides   
   // accurate balance)

#### Service Algorithm

Create an internal service method that implements the following algorithm. BFF controller end-point will invoke the service method.

1. Invoke User API: GET Consumer passing in ConsumerCode
2. Invoke User API: GET Person using the ConsumerDto.PersonId returned from above
3. Invoke Consumer Wallet API: Find Consumer Wallet
4. Invoke Wallet API: GET Wallet with returned Wallets[0].WalletId
5. Fill out only the following in the response GetConsumerSummaryResponseDto for now:
   1. ConsumerPersonDetailDto
   2. ConsumerWalletDetailDto (only Wallets, not RecentTransactions)

### GET Consumer Task List

<TBD>

- Input: selection of Available, Pending, Completed (options to return one or more list)

List<all available TaskRewardsDtos for consumer>

List<all pending TaskRewardsDtos for consumer>

List<all completed TaskRewardsDtos for consumer>

### GET Consumer Transaction List

<TBD>

List<all redemptions by consumer>

### POST Redeem

<TBD>

Redeem flow

### POST Consumer Task Enroll

<TBD>

### POST Consumer Task Update

<TBD>

# UI

## App Namespace Repo and Folders

Namespace prefix (typescript): SunnyRewards.Helios.Ui.\*

Repository name: helios-ui

Sub folders:

* sunnyrewards.helios.app
* sunnyrewards.common.store

## Home

Update Home page style to the following:

A screenshot of a phone

Description automatically generated with medium confidence

All of the data for the Home screen is returned by the BFF API GET Consumer Summary 11.1.1

### Components

#### Home Header

Update Home page header to the following:

A screenshot of a green screen

Description automatically generated with low confidence

Fields:

* Available to spend: ConsumerWalletDetailDto.Wallets[0].Balance
* Earned this year (highlight part of range control): ConsumerWalletDetailDto.Wallets[0].TotalEarned
* Available to earn: ConsumerWalletDetailDto.Wallets[0].LeftToEarn
* “Spend” button is visible, but not functional
* The Actions completed or Actions available section should not be shown

#### Pending Tasks

TBD

#### Actions For You

A picture containing text, screenshot, font, number

Description automatically generated

This section is filled by using GetConsumerSummaryResponseDto.AvailableTasks fields.

Fields:

* Task name: TaskRewardDto[i].TaskDto.TaskHeader
* Task description: Truncated after first few words with “…” suffix: TaskRewardDto[i].TaskDto.TaskDescription
* Reward amount: TaskRewardDto[i].Reward
* Left Icon: (no graphics yet): Take Task Header first letters of first two words and form a circle with those two letters. For example, Enroll in Omada Health will have icon (same background color as the reward):

A green circle with white letters

Description automatically generated with medium confidence

* “See all actions” button will be visible but disabled

#### Recent Activity

A screenshot of a cell phone

Description automatically generated with low confidence

This section is filled by using GetConsumerSummaryResponseDto. ConsumerWalletDetailDto.RecentTransactions fields.

Fields:

* Date: TransactionDto.CreateTs
* Description line: TransactionDto.TransactionDetail.TaskDto.TaskHeader
* Amount with sign: TransactionDto.TransactionAmount
  + “+” if TransactionDto.TransactioType == “A”
  + “-” if TransactionDto.TransactioType == “S”
* “See all transactions” button will be visible but disabled

## Launch Page

This page is only used for demo purposes and will not be in the final product. The launch page will have static content as added in the Jira story.

Add Rewards button next to the “Health & Wellness” link at the top right of the page.

When clicked, open Home page (from section 12.1) of product (server redirect) through POST with the following JSON payload:

{

“consumerCode”: “GUID\_constant”

}

## Task/Activity Popup

A screenshot of a phone

Description automatically generated with low confidence

Implementation:

* Top portion of the popup should be scrollable with fields populated from TaskDto:
  + TaskDto.TaskHeader
  + TaskDto.TaskDescription
  + TaskDto.TermsOfService
* Bottom (dark green) should be sticky and viewable always
* Only the “Earn …” field populated from TaskRewardDto.Reward
* No “Expiry field” required
* Try it now button visible but not functional

## Task/Activity List Page

<TBD>

## Transaction List Page

<TBD>

## Redeem Flow Start Page

<TBD>

# Object Naming Conventions

## Codes in DB

Codes stored in columns of name <xyz>\_code in the database will be named using the following convention so that given just a code value, it will be easy to identify that type of top-level entity it is in the system:

* customer: cus-GUID
* sponsor: spo-GUID
* tenant: ten-GUID
* role: rol-GUID
* person: per-GUID, eg: per-85d4d04f20074b32a91654567b2c6248
* consumer: cmr-GUID
* task: tsk-GUID
* task\_reward: trw-GUID
* wallet\_type: wat-GUID
* wallet: wal-GUID
* transaction: txn-GUID