HealtheIntent: Resolving Gaps in Care and Condition Documentation

Jon Fewins

Lead Engineering Manager

November 12th, 2020

Empowering beyond

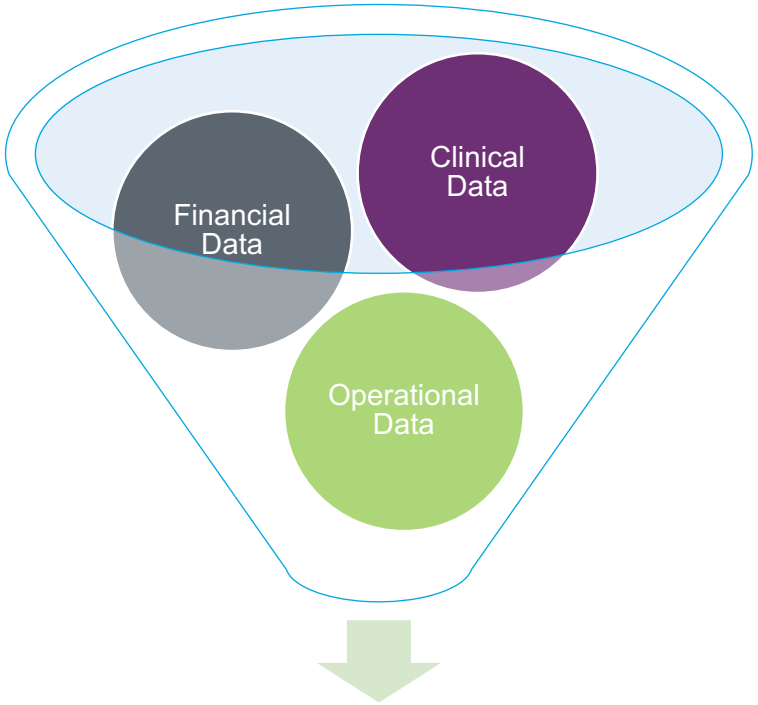
Cerner

Gaps in Care

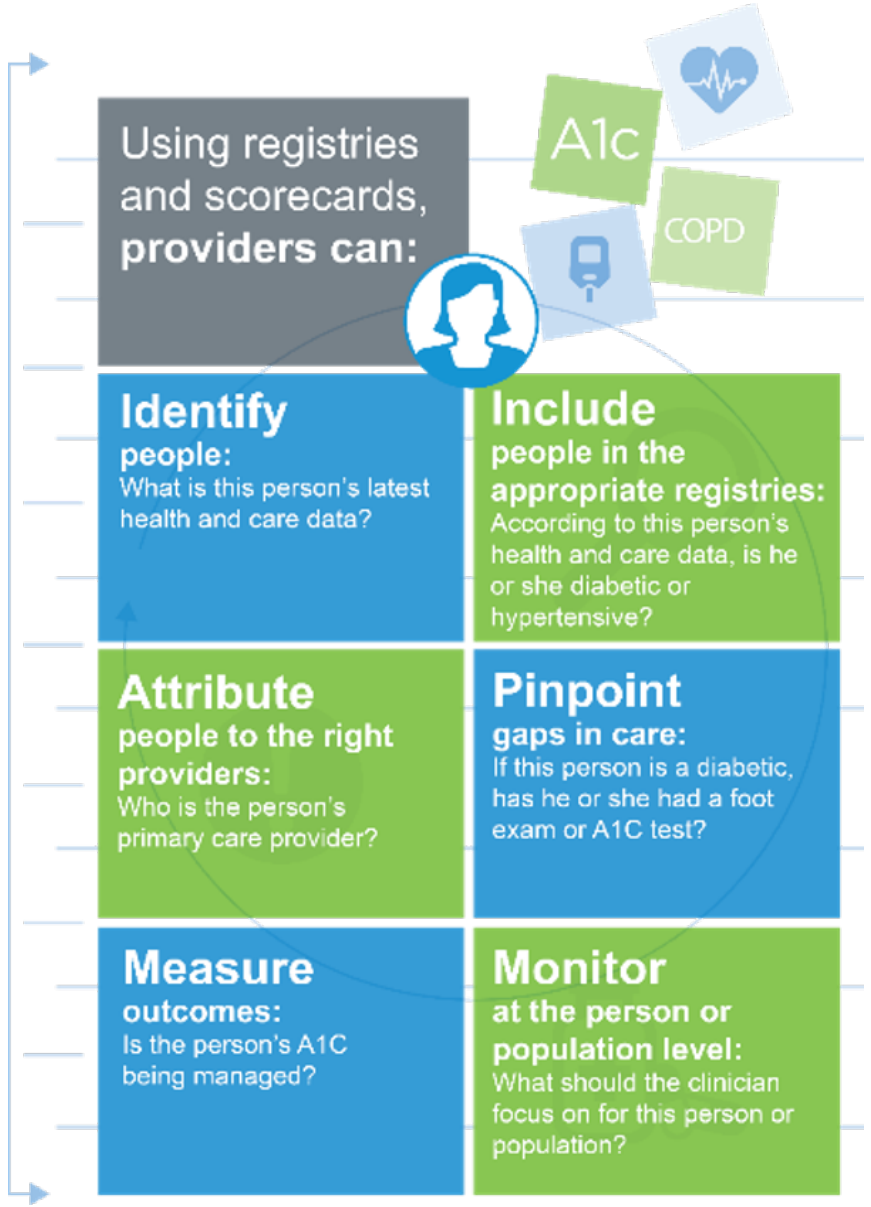
What is a gap in care?

Gaps in Care

Measures



Meaningful Information



Coding Gaps

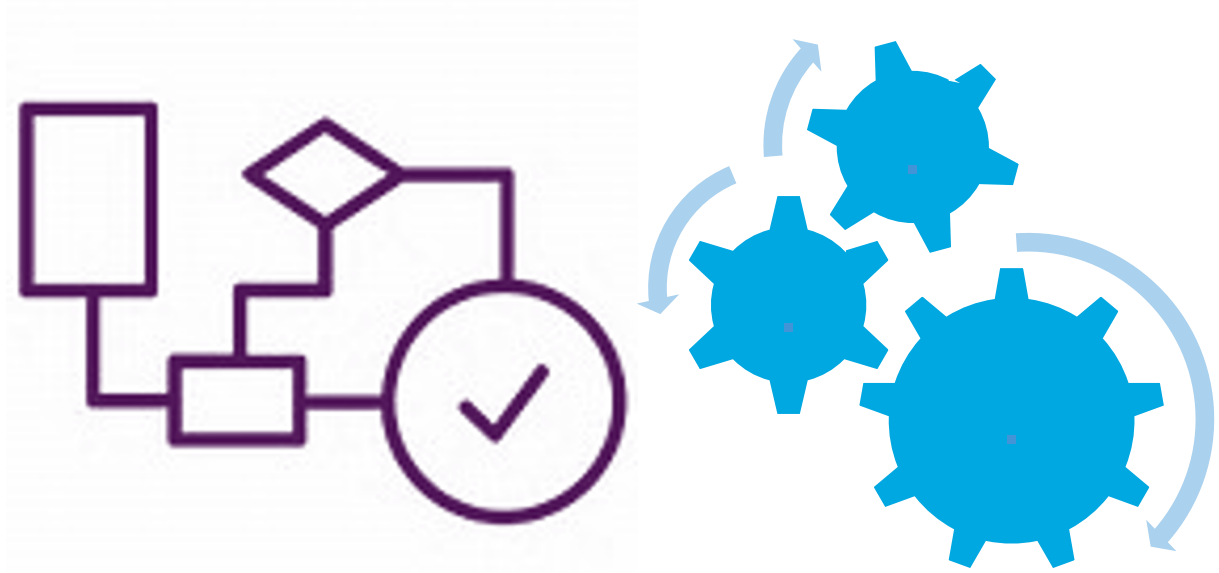
What is a coding gap?

Coding Gaps

HCC / Condition Identification

Hierarchical Condition Category (HCC) - Basics

Hierarchical Condition Category (HCC) - Basics



Diagnosis persistence

Previously documented diagnoses; not documented for current year

Suspected conditions

Potentially undiagnosed or undocumented conditions for current year

How do we **manage HCCs annually** to maximize reimbursement?

Prospective outreach

Identify those previously diagnosed or potentially undiagnosed

Point of care

Flag HCC conditions to diagnosis & document

Post-visit

Enable appropriate follow-up & coder validation

Value Optimization Services

Empowering beyond

Cerner

Value Optimization Services

Find our documentation here 

https://docs.healtheintent.com/#value-optimization

Services related to value-based, quality care

Value Optimization Services

New this year

- HCC Service
  - Hierarchical Condition Category API
- Condition Identification Service
  - Condition Identification API
- Quality Measure Service
  - Quality Measure API

API Domain Structure

API Domain Structure - Definition

- HCC Service
  - Risk Adjustment Models, Hierarchical Condition Categories, Condition Category Codes
- Condition Identification Service
  - Definitions, Confidence Weight Index, Hierarchy Index, Condition Options, Recommendation Policies, etc.
- Quality Measure Service
  - Measures, Registries, Categorizations, Recommendation Policies, etc.

API Domain Structure - Environment

- HCC Service
  - Pipeline
- Condition Identification Service
  - Pipeline
- Quality Measure Service
  - Reporting Project

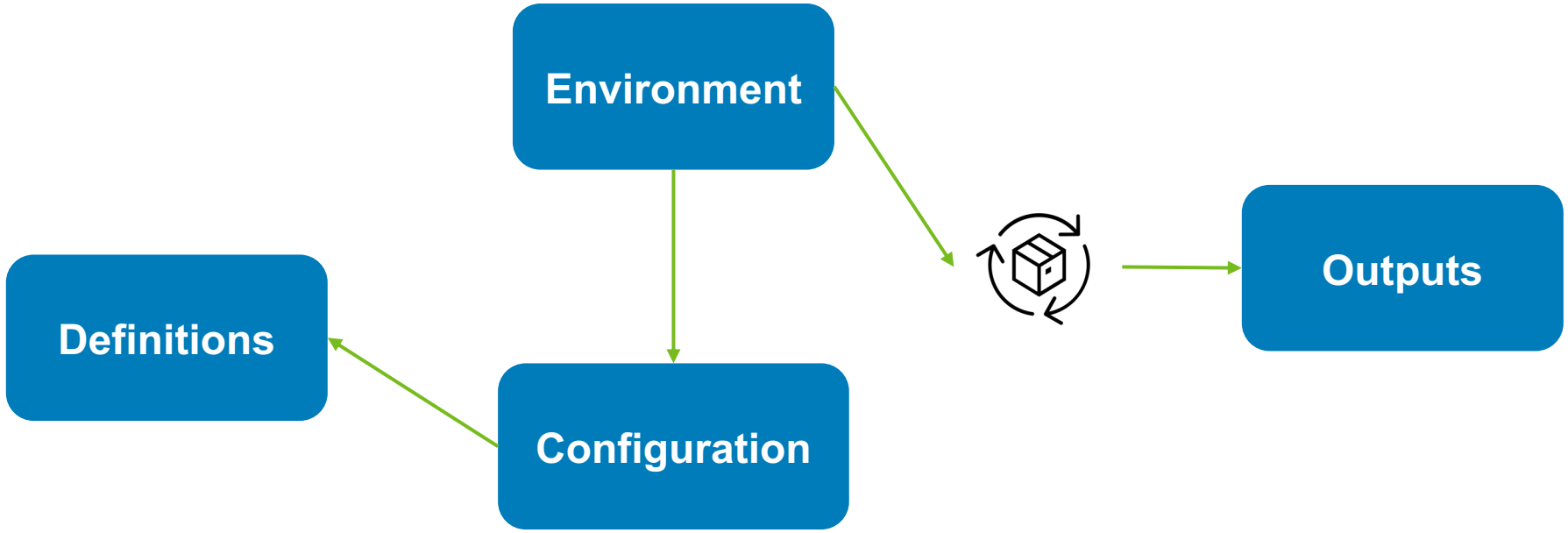
API Domain Structure - Configuration

- HCC Service
  - Pipeline Configuration
- Condition Identification Service
  - Pipeline Configuration
- Quality Measure Service
  - Reporting Project Configuration

API Domain Structure - Outputs

- HCC Service
  - Persistent HCC, Suspected HCC
- Condition Identification Service
  - Prior Diagnosis, Suspected Diagnosis, Diagnosis Clarification
- Quality Measure Service
  - Measure Results, Recommendations

API Domain Structure - Diagram



Use Case

Use Case

- Display coding gaps into a provider workflow
- Allow for the provider to access more information about the coding gaps
- Allow for the provider to provide feedback about the coding gap recommendations

Use Case

- Display coding gaps into a provider workflow
- Allow for the provider to access more information about the coding gaps
- Allow for the provider to provide feedback about the coding gap recommendations

Sounds like a job for CDS Hooks!!

Remote Decision Support – CDS Hooks

Clinical Decision Support (CDS) Hooks is a specification that enables integration of remote decision support within a clinician's EHR workflow through a "hook"-based pattern. See more here.

`hook` - An event triggered by the EHR as part of the clinicians workflow. The EHR will invoke 3rd party CDS services registered to this event.

`CDS Service` - A 3rd party service set up to observe EHR activity. Specifically, this service listens for a specific hook to be invoked by the clinician. When the hook is invoked, the EHR calls upon this service, and the service may decide whether or not to respond with decision support back.

`card` - One type of response the CDS Service can give when called upon by the EHR. It is an entity containing all the information the CDS Service wants to display on the EHR. These cards can contain textual information, suggestions to add/remove/update FHIR resources (mostly pertaining to the patient), or even links to external web pages or SMART applications.

Remote Decision Support – CDS Hooks

Clinical Decision Support (CDS) Hooks is a specification that enables integration of remote decision support within a clinician's EHR workflow through a "hook"-based pattern. See more here.

`hook` - An event triggered by the EHR as part of the clinicians workflow. The EHR will invoke 3rd party CDS services registered to this event.

`CDS Service` - A 3rd party service set up to observe EHR activity. Specifically, this service listens for a specific hook to be invoked by the clinician. When the hook is invoked, the EHR calls upon this service, and the service may decide whether or not to respond with decision support back.

`card` - One type of response the CDS Service can give when called upon by the EHR. It is an entity containing all the information the CDS Service wants to display on the EHR. These cards can contain textual information, suggestions to add/remove/update FHIR resources (mostly pertaining to the patient), or even links to external web pages or SMART applications.

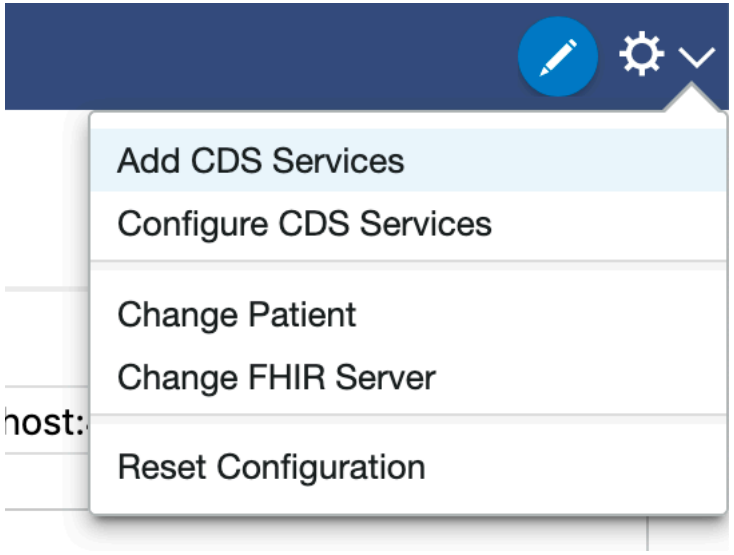
Step 1: Define the CDS Service

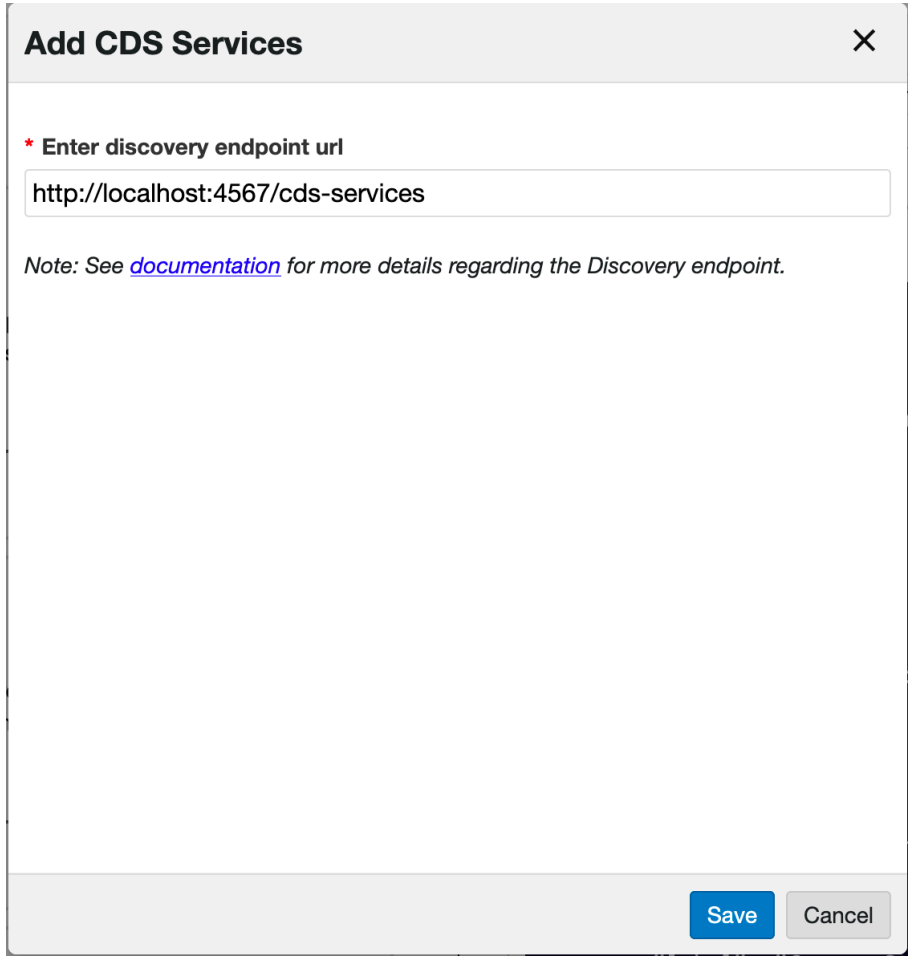
GET http://localhost:4567/cds-services


```
{
    "services": [
        {
            "hook": "patient-view",
            "id": "condition-identification-recommendations-v1",
            "title": "Recommendations from Condition Identification",
            "description": "The CDS service that returns a card with recommendations for closing coding gaps based on evaluation of condition identification rules.",
            "prefetch": {
                "patient": "Patient/{{context.patientId}}"
            }
        }
    ]
}
```

Step 2: Register CDS Service

SMART Health IT Sandbox will be used for testing.





Step 3: Register a Patient

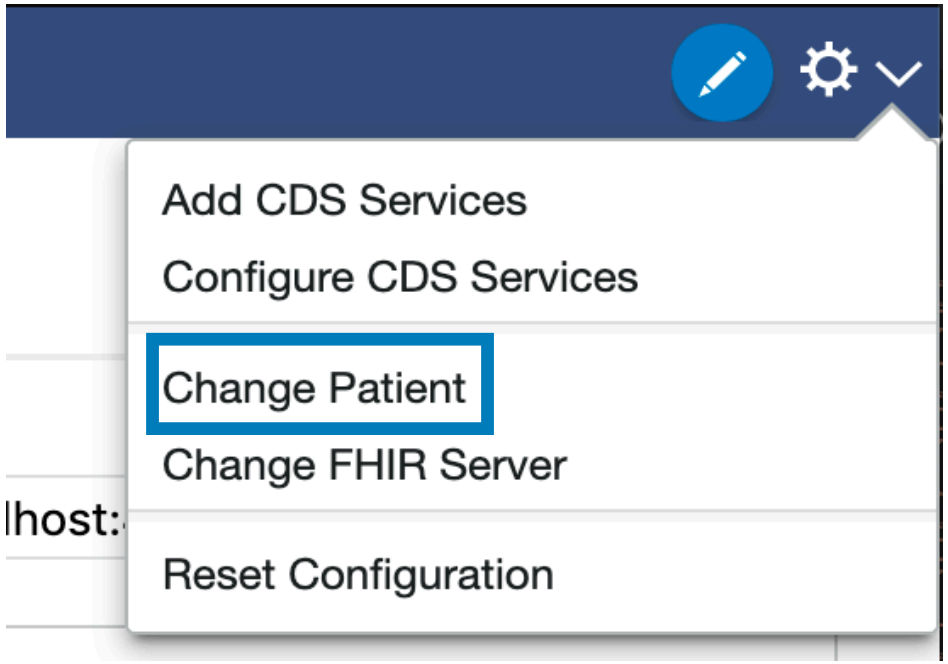
Register a Patient in the SMART Sandbox FHIR server that aligns with a patient in the Cerner Demo HealtheIntent tenant.

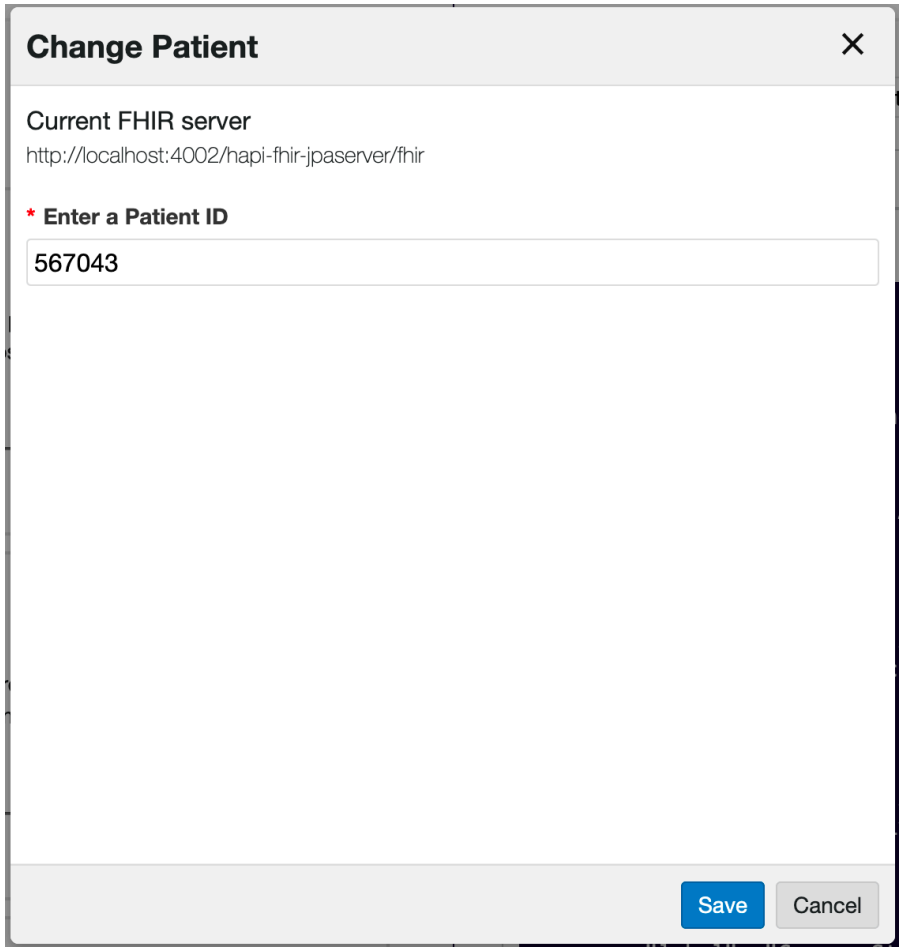
POST https://launch.smarthealthit.org/v/r2/fhir/Patient

Business Identifier for existing patient in Cerner Demo

Using the name of “Learning Code”

Step 4: Change Patient





DEMO

Review Implementation

Source Code

- Ruby based Sinatra application
- No authorization currently built in (just a proof of concept)
- Bearer Token for authorization to HealtheIntent APIs as environment variable

https://github.com/jfewins/cds-services-example

GET CDS Service

```
get '/cds-services' do
  content_type :json
  patient_view_example = {
    hook: 'patient-view',
    id: 'condition-identification-recommendations-v1',
    title: 'Recommendations from Condition Identification',
    description: 'The CDS service that returns a card with recommendations for closing coding gaps ' \
                 'based on evaluation of condition identification rules.',
    prefetch: {
      patient: "Patient/{{context.patientId}}",
    }
  };

  { services: [patient_view_example]}.to_json
end
```

Lookup Patient in HealtheIntent

```ruby
post '/cds-services/condition-identification-recommendations-v1' do
  content_type :json

  request_data = JSON.parse(request.body.read)

  # Look for a business identifier with my system identifier for the patient lookup
  business_id = request_data['prefetch']['patient']['identifier'].first{ |id| id[IntegrationConfig.business_identifier_system] }
  unless business_id
    status 404
    return
  end

  tenant_mnemonic = IntegrationConfig.tenant_mnemonic
  base_api_url = "https://#{tenant_mnemonic}.api.us.healtheintent.com"
  api_client = HealtheIntentApi.new(base_api_url, IntegrationConfig.bearer_token)

  population_id = IntegrationConfig.population_id
  data_partition_id = IntegrationConfig.data_partition_id
  resp = api_client.patient_id_lookup(population_id, data_partition_id, business_id['value'])

  unless resp
    status 404
    return
  end
```

Retrieve Prior Diagnoses

```
  patient_id = resp['items'][0]['patient']['id']
  pipeline_id = IntegrationConfig.pipeline_id
  pds = api_client.get_prior_diagnoses(pipeline_id, patient_id)

  card_adapter = CardAdapter.new(tenant_mnemonic, base_api_url, population_id, pipeline_id, patient_id)
  card_adapter.to_cards(pds).to_json
end
```

Persist Feedback

```ruby
post '/cds-services/condition-identification-recommendations-v1/feedback' do
  content_type :json

  request_data = JSON.parse(request.body.read)

  tenant_mnemonic = IntegrationConfig.tenant_mnemonic
  base_api_url = "https://#{tenant_mnemonic}.api.us.healtheintent.com"
  api_client = HealtheIntentApi.new(base_api_url, IntegrationConfig.bearer_token)

  request_data['feedback'].each do |card|
    if card['outcome'] != 'overridden' || card['overrideReason'] == nil
      next
    end

    tenant_mnemonic = IntegrationConfig.tenant_mnemonic
    population_id = IntegrationConfig.population_id
    pipeline_id = IntegrationConfig.pipeline_id
    system_account_id = IntegrationConfig.system_account_id
    suppression_reason_type = card['overrideReason']['reason']['code']

    card_adapter = CardAdapter.new(tenant_mnemonic, base_api_url, population_id, pipeline_id)
    patient_id = card_adapter.card_uuid_values(card['card'])['patient_id']
    condition_identification_definition_id = card_adapter.card_uuid_values(card['card'])['condition_identification_definition_id']

    api_client.create_suppression(pipeline_id, patient_id, condition_identification_definition_id, suppression_reason_type, system_account_id)
  end
end
```

Thank You

Empowering beyond

Cerner