Tax Identification Number Validation API

Interview Rules

- **Ensure that there is at least one commit per section**. Please include the section name in that final commit, for example “Completed: ‘PT1: Basic Format’”.
- Treat this interview as a series of tickets that you regularly execute on during your day-to-day. This should include any level of unit/integration tests that you feel covers the implementation appropriately.
- **LLM powered coding tools should be disabled and are forbidden for use**. That includes tools like Copilot, ChatGPT, etc.

Getting Set Up

- You should have received the archive “recurly_interview.tar.gz” which contains the project skeleton and tools you will need to complete the interview. The README.md file contains more detailed information that you should read through after finishing this section.
- There is a setup script included to initialize the git repository for the interview. Run the script using your shortname to set up your working branch:
  `bin/setup-repo.sh <shortname>`
  For instance, if your name is John Smith, your shortname would be `jsmith`.
- Get started with “PT1: Basic Format” and complete fully through “PT3: External Validation”, in order. Have fun!
- Once you have completed the interview, please run “bin/finish-interview.sh” and submit the tarball named in the script output back to Recurly. Thank you for your time completing this interview!

PT1: Basic Format

A Tax Identification Number (TIN) is a unique identifier issued by governments to taxpayers and businesses for identification and tax filing purposes. These numbers are crucial for sellers because they need to comply with regulations requiring TIN display on invoices when transacting with other businesses. Additionally, in certain jurisdictions, a valid TIN can exempt the seller from charging tax, transferring the tax liability to the buyer who then remits it directly to the government. If a TIN is found to be invalid, the seller may be held responsible for the unpaid taxes. There are also other regulatory requirements tied to the proper use of TINs.

Countries have different formats, algorithms and online validation services to check validate numbers against.

Objective

The goal of this project is to develop a TIN validation service that offers a JSON API capable of consolidating validation processes for various countries. This will enable sellers to ensure the TINs they collect are valid.

Scope

The initial phase of this project involves implementing basic validation checks, specifically checking the length and format of the TINs. Below is a table outlining the required TIN formats for implementation:

Country	ISO-3166	TIN Type	TIN Name	Format (Length)	Example
Australia	AU	`au_abn`	Australian Business Number	`NN NNN NNN NNN` (11)	`10 120 000 004`
Australia	AU	`au_acn`	Australian Company Number	`NNN NNN NNN` (9)	`101 200 000`
Canada	CA	`ca_gst`	Canada GST Number	`NNNNNNNNNRT0001` (9)	`123456789RT0001`
India	IN	`in_gst`	Indian GST Number	`NNXXXXXXXXXXNAN` (15)	`22BCDEF1G2FH1Z5`

*The Format notation is **N** for numeric digit, **A** for alpha digit, **X** for alphanumeric digit. Anything else is a literal value that may be omitted on input.*

API Specification

The API should provide an endpoint that accepts an unformatted TIN and its corresponding country code for validation. The response should indicate whether the TIN is valid. If valid, the response should include the TIN formatted according to the country-specific requirements. If invalid, the response should specify the reason for invalidation.

Here is a sample response schema to get started:

```
Unset
properties:
  valid:
    type: boolean
  tin_type:
    type: string
    enum: [au_abn, au_acn, ca_gst, in_gst]
```

```
  formatted_tin:
    type: string
  errors:
    type: array
    items:
      type: string
```

Here are some valid inputs and outputs:

- Country: `CA`, Number: `123456789`; responds with `123456789RT0001`
- Country: `CA`, Number: `123456789RT0001`; responds with `123456789RT0001`
- Country: `AU`, Number: `10120000004`; responds with `10 120 000 004`
- Country: `AU`, Number: `10 120 000 004`; responds with `10 120 000 004`

PT2: Enhanced Format

Beyond basic format and length checks, some countries utilize specific algorithms to validate TINs. A common example is the use of a variation of the Luhn algorithm, similar to those used for credit card verification.

Objective

This milestone focuses on implementing an algorithmic validation for the Australian Business Number (ABN) based on the specifications outlined on the government website.

API Enhancement

The API should be able to detail whether an ABN is invalid due to failure in passing the algorithmic check.

Here are some additional ABN to test against:

- `10000000000` (valid)
- `10120000004` (valid)
- `10120000005` (invalid)

PT3: External Validation

While local format checks provide initial validation of a Tax Identification Number (TIN), the most definitive method of verification involves querying official government APIs. These services confirm the registration and accuracy of business information associated with a TIN.

Objective

This milestone focuses on integrating with the Australian Business Number (ABN) government web services API to retrieve and validate business entity information. This process not only confirms the validity of an ABN but also ensures that the business details are current and registered appropriately.

Server Setup

A simplified server implementation representing the ABN WS (Web Service) API is included in the repository. This mock server can be run using the command:

`bin/abn_query_server.rb`

Once running, the server is accessible at `localhost:8080`. It responds to queries for an ABN and returns detailed XML data about the business, including its registration status with respect to the Goods and Services Tax (GST).

Sample Request and Response

You can query the server using the following curl command:

`curl localhost:8080/queryABN?abn=10000000000`

This request would yield the following XML response:

```
Unset
<?xml version="1.0" encoding="UTF-8"?>
<abn_response>
       <request>
```

```
            <identifiersearchrequest>
                  <abn>10000000000</abn>
            </identifiersearchrequest>
      </request>
      <response>
            <dateRegisterLastUpdated>2024-04-01</dateRegisterLastUpdated>
            <dateTimeRetrieved>2024-04-26T10:21:16-07:00</dateTimeRetrieved>
            <businessEntity>
                  <abn>10000000000</abn>
                  <status>Active</status>
                  <entityType>Company</entityType>
                  <organisationName>Example Company Pty</organisationName>
                  <goodsAndServicesTax>false</goodsAndServicesTax>
                  <effectiveTo>2024-09-01</effectiveTo>
                  <address>
                        <stateCode>NSW</stateCode>
                        <postcode>2001</postcode>
                  </address>
            </businessEntity>
      </response>
</abn_response>
```

An ABN should be considered valid only if the associated business is registered to remit GST. This key validation step ensures that the ABN is not only correctly formatted but also actively linked to a compliant business entity.

API Enhancement

Enhance the API to determine if an ABN is invalid due to discrepancies found during the government API call. The response should include detailed information about the business entity such as the name and address, alongside the formatted ABN and its validation state.

Here are additional properties to add to the previous response:

```
Unset
```

```
properties:
  business_registration:
    type: object
    properties:
      name:
        type: string
      address:
        type: string
```

Here are some additional ABN's to test against:

- `10000000000` (invalid, business is not GST registered)
- `10120000004` (valid)
- `53004085616` (invalid, ABN server returns 500, external message is "registration API could not be reached")
- `51824753556` (invalid, ABN server returns 404, external message is "business is not registered")