CarbonML

<CARML> APP OVERVIEW AND USER PROCESS – CREATING & GENERATING MESSAGE TYPES FOR CO2e SUPPLY CHAIN TRACKING

FMCG Use Case Example

June 2023

The Carbon-ML project is developing an extensible open-source ecosystem to provide declarations of measurements for embodied carbon, or any declared trait, in any product or service at any point along the supply chain, within any form or report. Carbon-ML is incubated by Carbon Finance Labs in partnership with Oxy Low Carbon Ventures with a goal to rapidly evolve into an independently governed project.

Carbon Finance Labs

- We are a finance and technology incubator creating and implementing new climate change solutions. Our impact comes from using a global network of resources and knowledge built over decades spent in the carbon, finance and technology sectors globally.

OXY LOW CARBON VENTURES

- Oxy Low Carbon Ventures, LLC (OLCV), a subsidiary of Occidental, Petroleum

CarbonML.org Ecosystem

- **CarbonML.org**
  - Ecosystem of Industry and Service Consortia developing <CarML> Carbon Messaging Language

- **<CarML>**
  - A Global Language for Carbon Related Information
  - Transforming electronic transactional statements to include CO2e declarations and offsets for any product or service at any point in time along the circular supply chain
  - Open-source, technology agnostic – an ease of use translator between different systems languages or infrastructure

Product and Service decisions demand better CO2e data….
Better CO2e data enables more informed decisions

- CarbonML.org global ecosystem of industry, service, technology and other participants collaborating on the development of a global carbon messaging language format using existing taxonomies and schemas that will be transparent and open-source.
- <CarML> an extensible, open, agnostic, adaptable, secure global standard messaging system that extends to other data on electronic transactional or communication formats, such as CO2e and environmental mitigation instruments.
- <CarML> enables the reporting of CO2e and other traits for every product and service at any point along the supply chain, with ease of global implementation.
- CarbonML.org ecosystem participants develop related usage solutions so that companies, consumers, suppliers, governments, etc. can make more informed decisions.

What exactly is <CarML>?

- <CarML> is an application which is supplied to users through a Docker container
- At a basic level, <CarML> is comprised of a collection of CarbonML.org defined data fields in a JSON format with APIs containing endpoints that can connect to any user defined data point/database
- These individual data fields can be combined into any form/document/message type that the user requires through utilization of:
  - <CarML> Custom message type report builder
  - <CarML> Pre-formatted message types and forms
- <CarML> all information on a message type can be individually maintained, tracked, and traced throughout the supply chain, from one party to the next, through use of blockchain technology
- CarbonML.org does not access any company or user data which further enables privacy and security

<CarML> Usage Examples

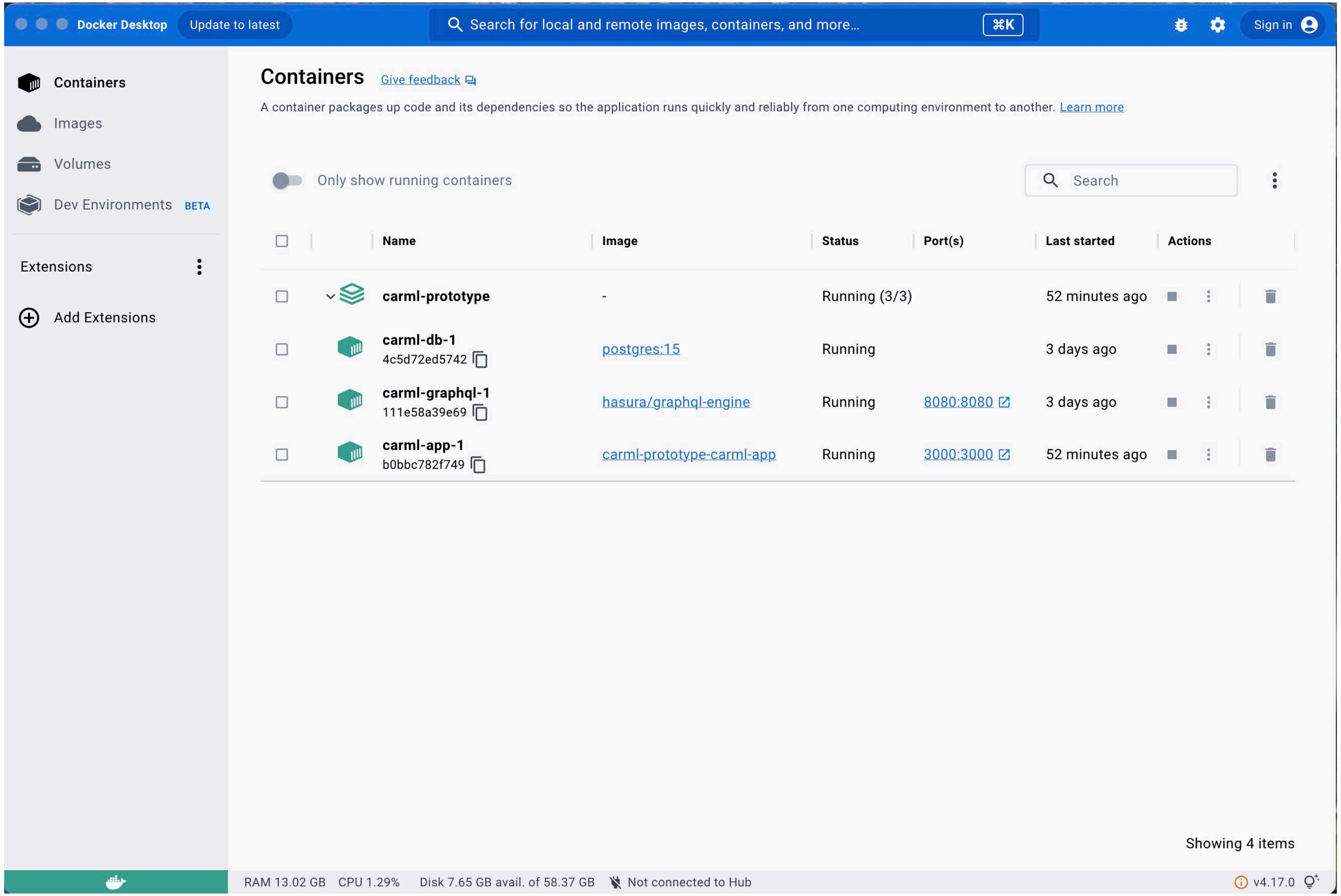
- Standardized and open source, <CarML> provides Product and Service supply chain documents in a standard language format; incorporating CO2e, environmental mitigation instruments, and other traits and parameters of interest to sender/receiver; that can easily be tracked, traced and stored
- Technology agnostic, <CarML> provides ease of extracting data from any systems infrastructure, database, configuration, format and importing to another systems infrastructure, database, configuration, format among others
- Adaptable and extensible, <CarML> provides standard formats for other purposes such as data registries communications, natural resources reporting, just transition measurements, and circularity tracking and traceability among others

<CARML> USER PROCESS – HOW TO CREATE AND DEPLOY A <CARML> MESSAGE TYPE

How the <CarML> Application is Delivered for Installation

- Once a user requests the <CarML> Application, it will be delivered in a Docker Container including among others:
  - README files and user documentation, which can also be accessed on the Carbon-ML-org GitHub site
  - <CarML> Dashboard
  - <CarML> Hasura/GraphQL engine for use with <CarML> data field API to connect endpoints (this can also be provided as a REST client)
  - <CarML> App which provides the Field Selector Message Type Template Builder, Standard Templates, and other components
  - <CarML> PoC components for reference:
    - PoC Prototype
    - PoC Postgres Database

<CarML> Docker Container Contents Example:



Containers

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

□	Name	Image	Status	Port(s)	Last started	Actions
□	carml-prototype	-	Running (3/3)		52 minutes ago	
□	carml-db-1 4c5d72ed5742	postgres:15	Running		3 days ago	
□	carml-graphql-1 111e58a39e69	hasura/graphql-engine	Running	8080:8080	3 days ago	
□	carml-app-1 b0bbc782f749	carml-prototype-carml-app	Running	3000:3000	52 minutes ago	

Showing 4 items

RAM 13.02 GB CPU 1.29% Disk 7.65 GB avail. of 58.37 GB Not connected to Hub v4.17.0

After Installation - <CarML> User Dashboard

- To ensure transparency and ease of use, when a user signs-on to the <CarML> system, through a two-factor authentication process, there will be a user-friendly dashboard display containing the following links and access (including version control information):
    - Previous <CarML> Message Types created and saved by the user
    - Standard <CarML> Message Type Formats developed by CarbonML.org
    - The <CarML> Field Selector Message Type Template Builder which will allow the user to select unique data fields and create and save a customized <CarML> Message Type that is fully compatible with the <CarML> standard.
    - The <CarML> Data Field Dictionary maintained by CarbonML.org
    - Option of uploading user-created <CarML> standard data fields to the “User” section of the Carbon-ML GitHub site. Once user created data fields have been approved by the relevant Carbon-ML Ecosystem Consortium, they will be moved to the official <CarML> approved standard area of the GitHub site and reflected in the <CarML> Data Dictionary.

Definition - What is a <CarML> Message Type?

- A <CarML> Message Type refers to any form, report, or other communication between two parties regarding a product or service at a point in time along the supply chain. Basically, for each step along the supply chain, a new <CarML> Message Type is generated.
  - Each <CarML> Message Type has the ability of displaying information from previous <CarML> message types, along with new information, as the product or service moves along the supply chain, and can incorporate calculated fields to display, for example, the total accumulated CO2e at that point in the supply chain.
  - <CarML> Message Types can incorporate large data fields, links and summary display of other documents such as product specs, EPDs, industry and/or regulatory certificates, and other relevant or required information.
  - <CarML> Message Types are designed to enable ease of communication of all elements within a transaction/event between two parties through incorporation with existing systems and displaying as much or as little data that is required.

Definition - What is a <CarML> Message Type continued?

- Examples of <CarML> Message Types and the internal systems <CarML> can connect to:
  - General Invoice or Purchase Order – purchasing systems
  - Transportation – transport tracking systems
  - Packaging – operations systems
  - Customs or Regulatory Reports – compliance systems
  - Retail or Point of Sale Metrics – corporate analytics
  - Recycling Information – materials components
  - Request for Information on Environmental Offsets – registries and financial markets
  - General Data transfer from one system to another – database infrastructure
- Any <CarML> data, from any Message Type, can be combined or incorporated onto a new <CarML> Message Type at any point along the supply chain, including adding data fields and analytics/calculated fields

Definition - What is a <CarML> Message Type continued?

- <CarML> Message Types are populated using the <CarML> standard format individual data fields
  - <CarML> data fields can include any related product or service information, such as company, product and attributes, link to product specs, CO2e amount, measurement methodology, link to EPD, verification information, location and origin, transaction/event information, date and time, environmental offsets, and any other user and consumer determined information both data and links.
- There are several ways to create a <CarML> Message Type by accessing the <CarML> User Dashboard and selecting from:
  - Pre-built <CarML> Message Type templates from CarbonML.org
  - <CarML> Field Selector Message Type Template Builder
  - Fully customized by selecting a combination of pre-formatted <CarML> data form fields and user created form fields

<CarML> Creating and Generating a Message Type – Oranges Use Case Example

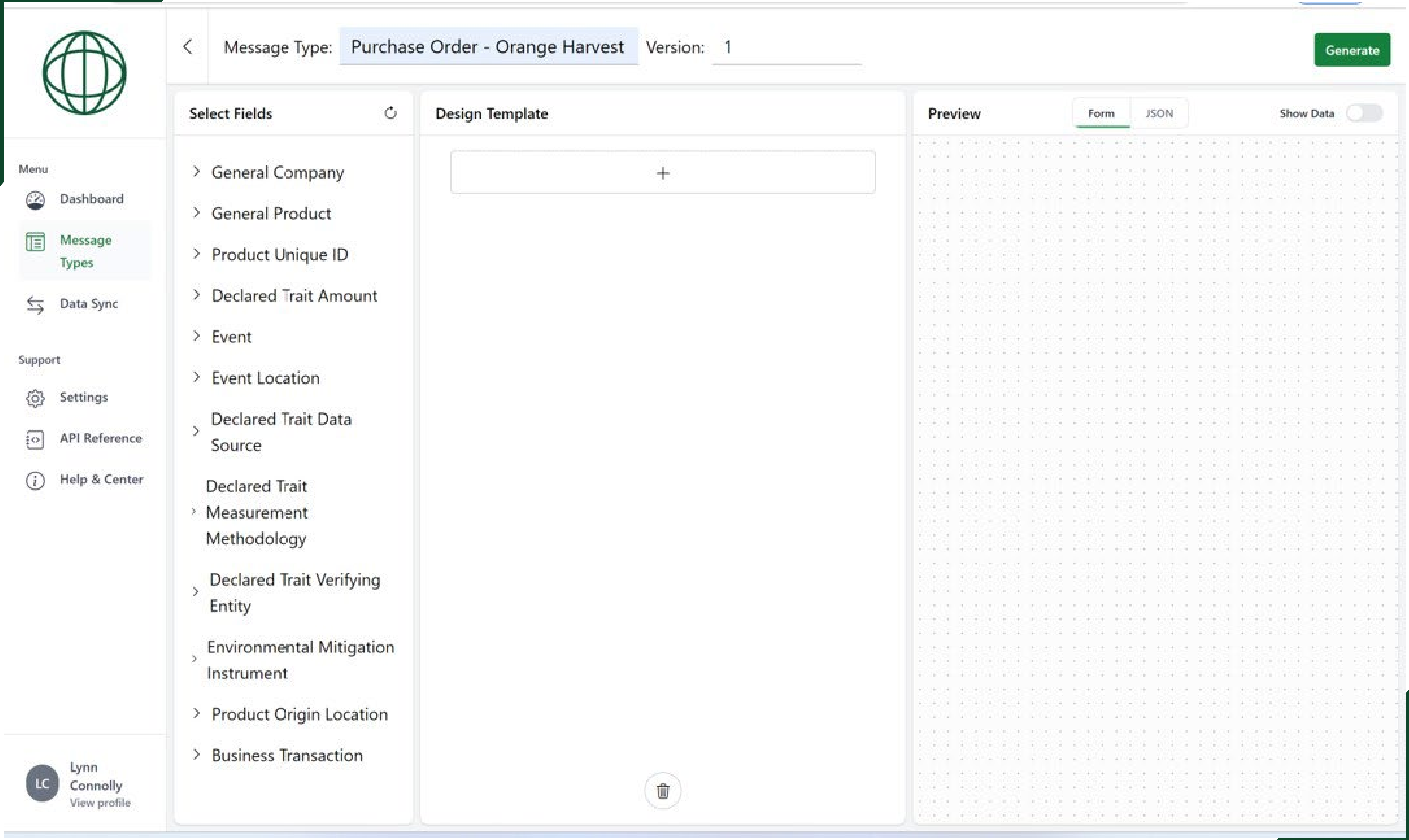
- Category: Fast Moving Consumer Good
- Declared product: Bag of Oranges
- Oranges Transactional Message Types comprised of <CarML> data fields:
  - Purchase Order - Raw Materials Supply – Harvest
  - Packaging and Transport – Packing Shed
  - Finished Goods - Retail Grocer – Point of Sale
- QR Code
- This use case provides a basic overview of how <CarML> Message Types can be constructed and generated. However:
  - This process can be fully automated and incorporated within existing systems which would alleviate many of the manual steps shown in this document
  - The steps shown are to illustrate the process and are not intended to reflect the actual process which would be fully automated

<CarML> Creating and Generating a Message Type – Oranges Use Case Example continued

- This use case provides a basic overview of how <CarML> Message Types can be constructed and generated. However:
  - The <CarML> process is intended to be fully automated and incorporated within existing systems which would alleviate many of the manual steps shown in this document
  - The steps shown are to illustrate the process and are not intended to reflect the actual process which can be a fully automated, efficient, economical process

Step 1: Determine the <CarML> Message Type Required

- The first point in the supply chain for the Oranges use case is the initial purchase order for oranges, from the POS Retail grocer to the Orange Farm – Happy Farms.
- The <CarML> Message Type required is a Purchase Order, and this <CarML> Purchase Order Message Type needs to be created and saved using the <CarML> Field Selector Message Type Template Builder.
- The technology department at Happy Farms has already installed the <CarML> application and attached the API endpoints from the <CarML> data fields to the relevant data residing in their backend systems. A technical example of a <CarML> data field is provided in the Appendix.
- The user at Happy Farms opens the <CarML> Field Selector Message Type Template Builder and begins to construct the <CarML> Purchase Order – Oranges Harvest message type.



Message Type: Purchase Order - Orange Harvest Version: 1

Generate

Menu

- Dashboard
- Message Types
- Data Sync

Support

- Settings
- API Reference
- Help & Center

LC Lynn Connolly
View profile

Select Fields

- General Company
- General Product
- Product Unique ID
- Declared Trait Amount
- Event
- Event Location
- Declared Trait Data Source
- Declared Trait Measurement Methodology
- Declared Trait Verifying Entity
- Environmental Mitigation Instrument
- Product Origin Location
- Business Transaction

Design Template

+

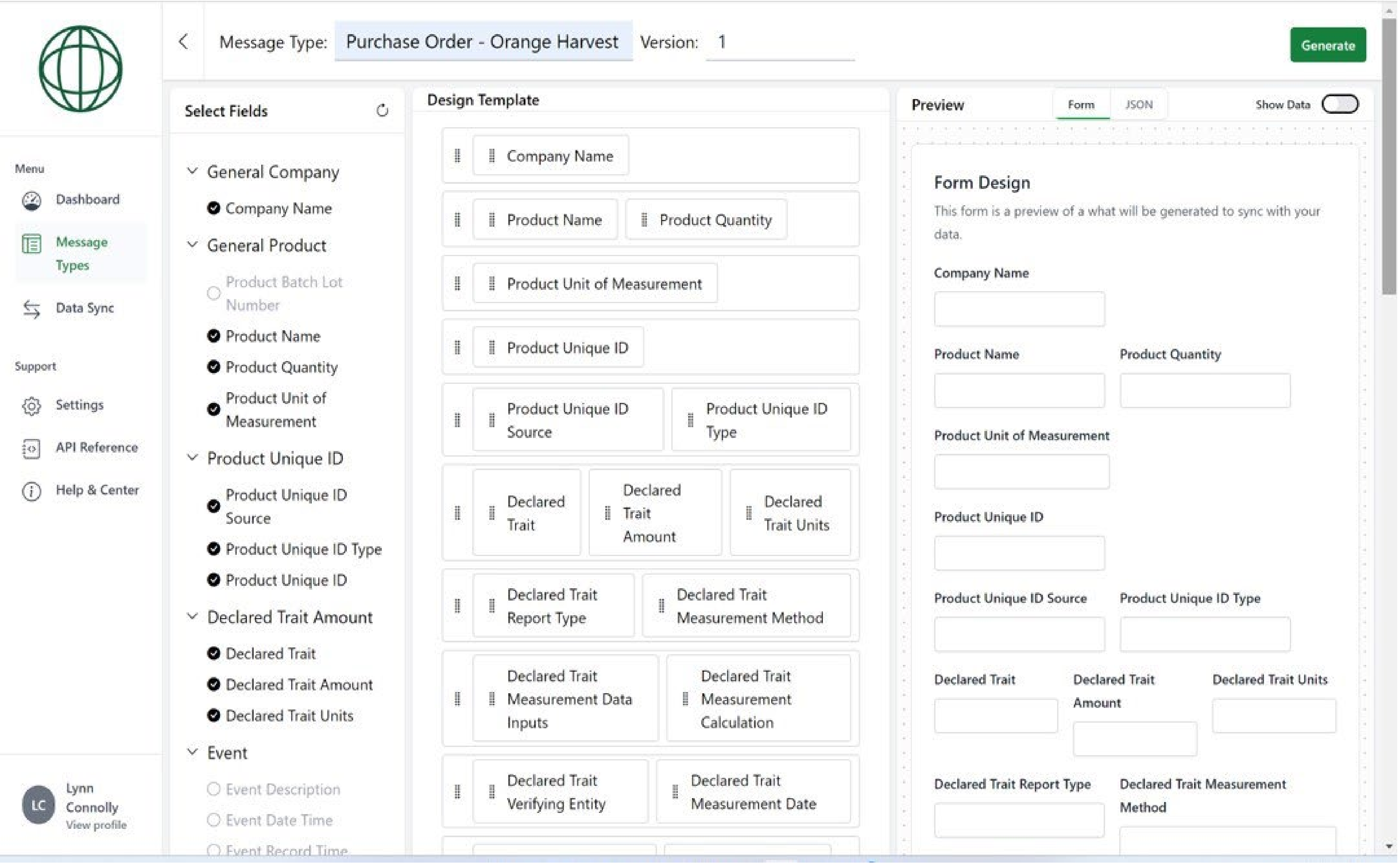
Preview

Form JSON

Show Data

Step 2: Determine the <CarML> Data Fields required for the <CarML> Purchase Order – Orange Harvest Message Type

- The user can then select the <CarML> data fields which are required for the <CarML> Purchase Order – Orange Harvest message type
- For this use case, this message type consolidates information from internal systems for display in the <CarML> standard format so that it can also be easily ingested by the recipient
- The <CarML> data fields can also be moved to create the design of the message type template, as displayed on the following two slides
- Once the <CarML> Purchase Order Template has been designed, it can then be saved for reuse. The technical description of where it is saved can be found in the appendix.



Message Type: Purchase Order - Orange Harvest Version: 1

Select Fields

- General Company
  - ● Company Name
- General Product
  - ○ Product Batch Lot Number
  - ● Product Name
  - ● Product Quantity
  - ● Product Unit of Measurement
- Product Unique ID
  - ● Product Unique ID Source
  - ● Product Unique ID Type
  - ● Product Unique ID
- Declared Trait Amount
  - ● Declared Trait
  - ● Declared Trait Amount
  - ● Declared Trait Units
- Event
  - ○ Event Description
  - ○ Event Date Time
  - ○ Event Record Time

Design Template

- Company Name
- Product Name | Product Quantity
- Product Unit of Measurement
- Product Unique ID
- Product Unique ID Source | Product Unique ID Type
- Declared Trait | Declared Trait Amount | Declared Trait Units
- Declared Trait Report Type | Declared Trait Measurement Method
- Declared Trait Measurement Data Inputs | Declared Trait Measurement Calculation
- Declared Trait Verifying Entity | Declared Trait Measurement Date

Preview

Form Design

This form is a preview of a what will be generated to sync with your data.

Company Name

Product Name

Product Quantity

Product Unit of Measurement

Product Unique ID

Product Unique ID Source

Product Unique ID Type

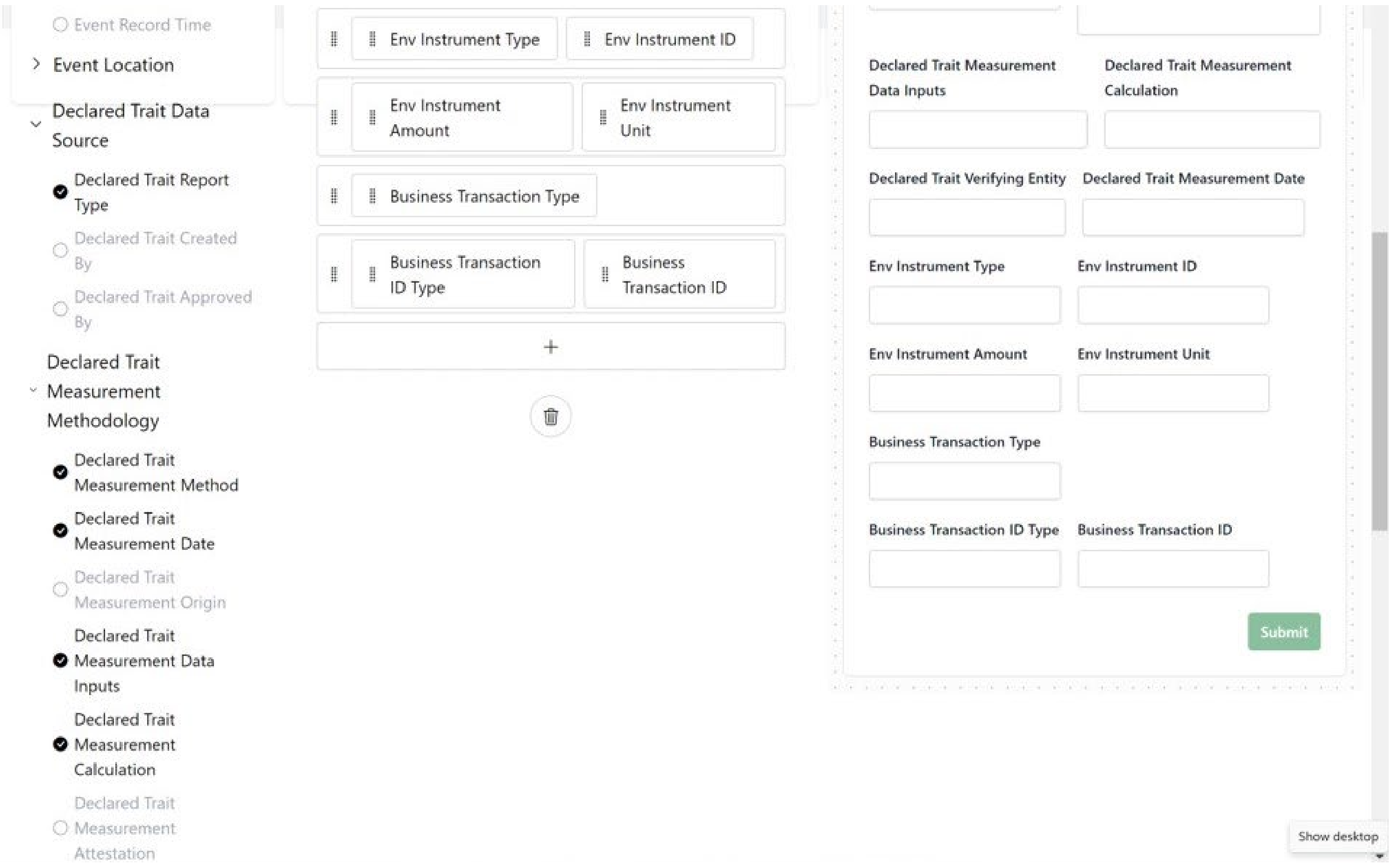
Declared Trait

Declared Trait Amount

Declared Trait Units

Declared Trait Report Type

Declared Trait Measurement Method



○ Event Record Time

› Event Location

⌄ Declared Trait Data Source

● Declared Trait Report Type

○ Declared Trait Created By

○ Declared Trait Approved By

⌄ Declared Trait Measurement Methodology

● Declared Trait Measurement Method

● Declared Trait Measurement Date

○ Declared Trait Measurement Origin

● Declared Trait Measurement Data Inputs

● Declared Trait Measurement Calculation

○ Declared Trait Measurement Attestation

Env Instrument Type | Env Instrument ID

Env Instrument Amount | Env Instrument Unit

Business Transaction Type

Business Transaction ID Type | Business Transaction ID

+

Declared Trait Measurement Data Inputs

Declared Trait Measurement Calculation

Declared Trait Verifying Entity

Declared Trait Measurement Date

Env Instrument Type

Env Instrument ID

Env Instrument Amount

Env Instrument Unit

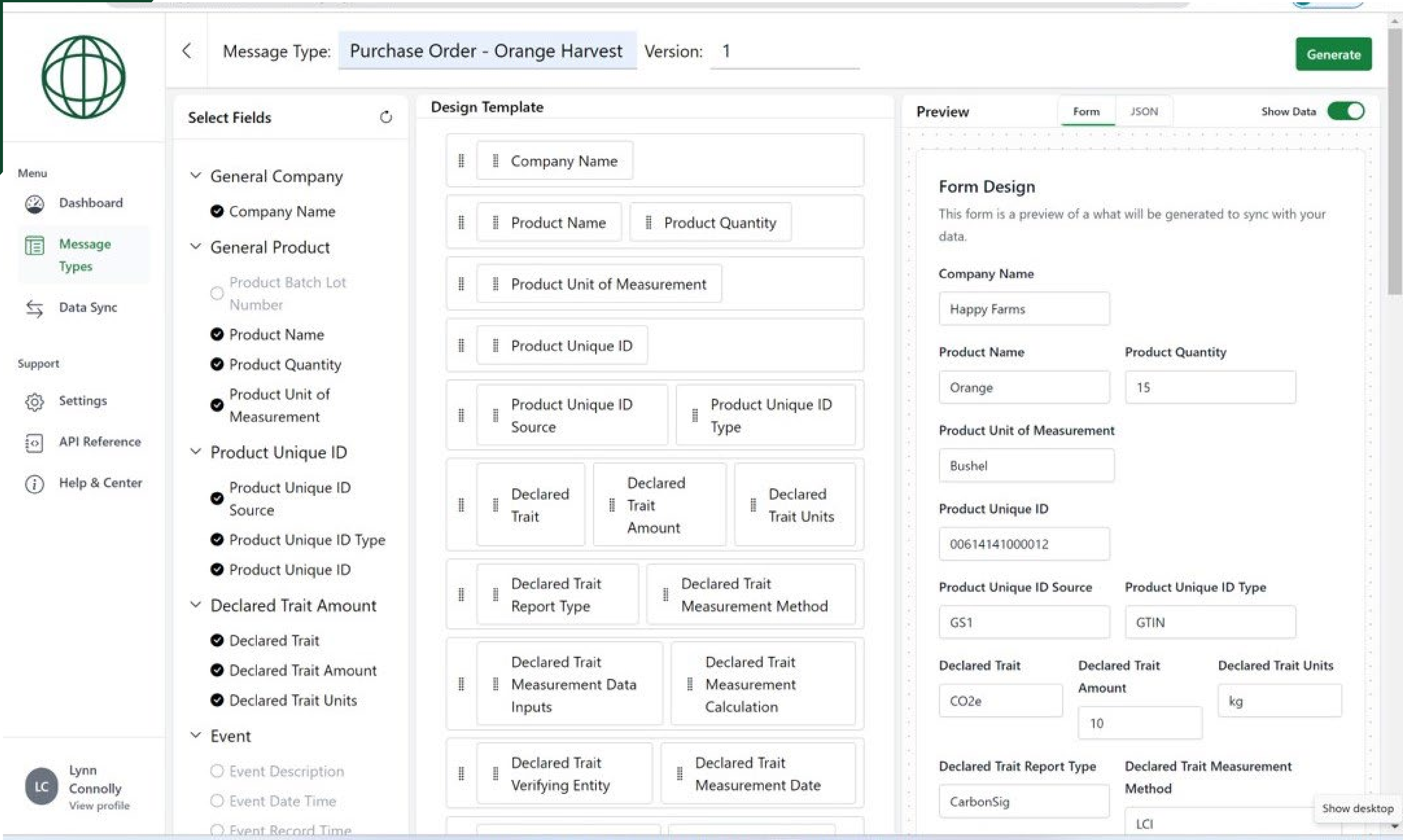
Business Transaction Type

Business Transaction ID Type

Business Transaction ID

Step 3: Select the Product Type and Related Data for the <CarML> Purchase Order – Orange Harvest Message Type

- <CarML> has been designed to be extensible, adaptable and technology agnostic with considerations for current technology infrastructure and the end user built into the design.
- For this use case, the technology department has already incorporated <CarML> within the company's systems infrastructure including the purchase ordering system.
  - For the product selection, <CarML> essentially sits on top of current systems so the creation of a purchase order and product selection would be similar to what currently exists at the company.
- The following slide shows an example of how the data would be displayed in the <CarML> Purchase Order – Orange Harvest message type, by toggling the "Data" button and the subsequent slide illustrates the underlying JSON, by clicking on the "JSON" tab.



Message Type: Purchase Order - Orange Harvest Version: 1

Select Fields

- General Company
  - ● Company Name
- General Product
  - ○ Product Batch Lot Number
  - ● Product Name
  - ● Product Quantity
  - ● Product Unit of Measurement
- Product Unique ID
  - ● Product Unique ID Source
  - ● Product Unique ID Type
  - ● Product Unique ID
- Declared Trait Amount
  - ● Declared Trait
  - ● Declared Trait Amount
  - ● Declared Trait Units
- Event
  - ○ Event Description
  - ○ Event Date Time
  - ○ Event Record Time

Design Template

- Company Name
- Product Name | Product Quantity
- Product Unit of Measurement
- Product Unique ID
- Product Unique ID Source | Product Unique ID Type
- Declared Trait | Declared Trait Amount | Declared Trait Units
- Declared Trait Report Type | Declared Trait Measurement Method
- Declared Trait Measurement Data Inputs | Declared Trait Measurement Calculation
- Declared Trait Verifying Entity | Declared Trait Measurement Date

Preview

Form Design

This form is a preview of a what will be generated to sync with your data.

Company Name: Happy Farms

Product Name: Orange

Product Quantity: 15

Product Unit of Measurement: Bushel

Product Unique ID: 00614141000012

Product Unique ID Source: GS1

Product Unique ID Type: GTIN

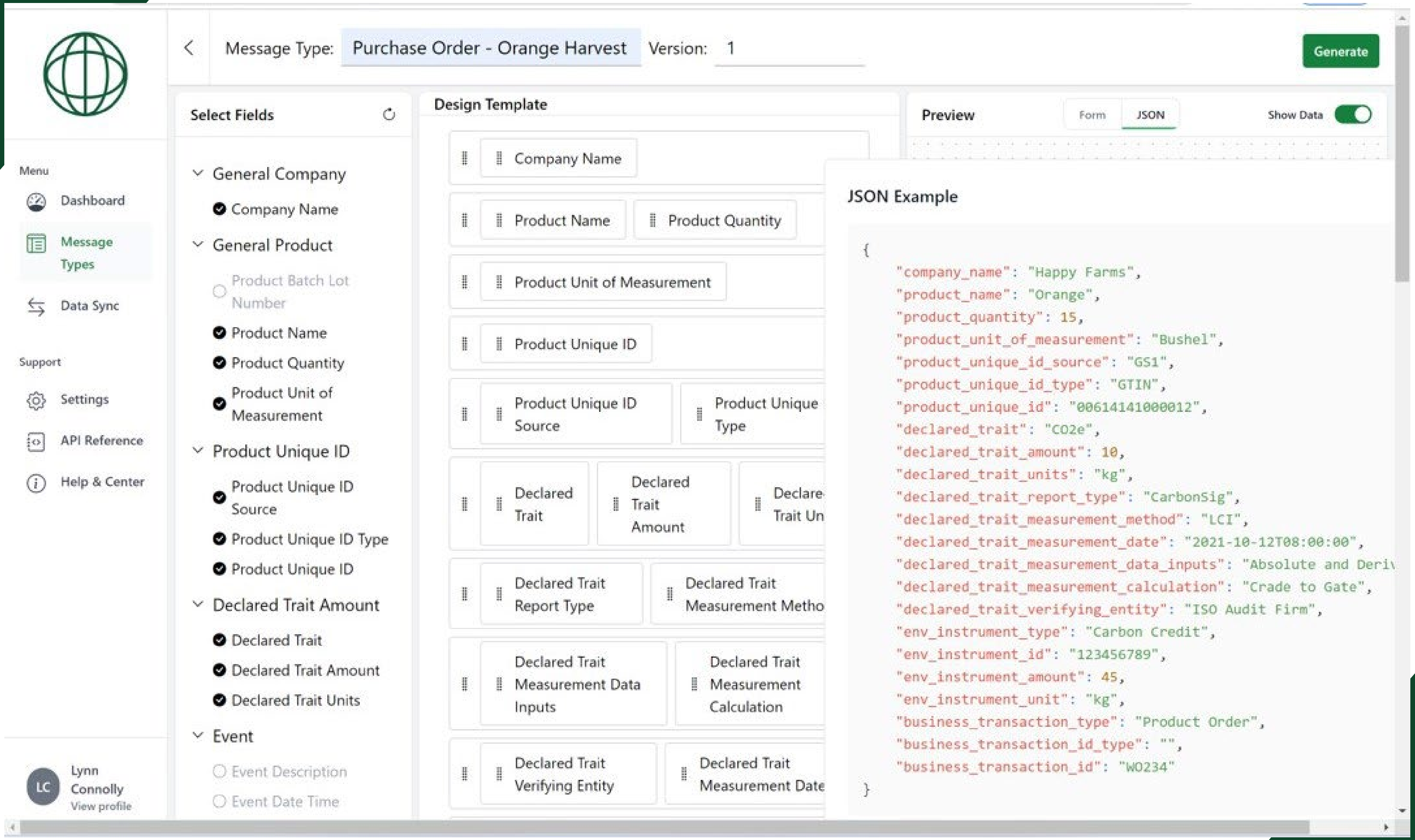
Declared Trait: CO2e

Declared Trait Amount: 10

Declared Trait Units: kg

Declared Trait Report Type: CarbonSig

Declared Trait Measurement Method: LCI



Message Type: Purchase Order - Orange Harvest Version: 1

Select Fields

- General Company
  - ● Company Name
- General Product
  - ○ Product Batch Lot Number
  - ● Product Name
  - ● Product Quantity
  - ● Product Unit of Measurement
- Product Unique ID
  - ● Product Unique ID Source
  - ● Product Unique ID Type
  - ● Product Unique ID
- Declared Trait Amount
  - ● Declared Trait
  - ● Declared Trait Amount
  - ● Declared Trait Units
- Event
  - ○ Event Description
  - ○ Event Date Time

Design Template

- Company Name
- Product Name | Product Quantity
- Product Unit of Measurement
- Product Unique ID
- Product Unique ID Source | Product Unique Type
- Declared Trait | Declared Trait Amount | Declare Trait Un
- Declared Trait Report Type | Declared Trait Measurement Metho
- Declared Trait Measurement Data Inputs | Declared Trait Measurement Calculation
- Declared Trait Verifying Entity | Declared Trait Measurement Date

JSON Example

```
{
    "company_name": "Happy Farms",
    "product_name": "Orange",
    "product_quantity": 15,
    "product_unit_of_measurement": "Bushel",
    "product_unique_id_source": "GS1",
    "product_unique_id_type": "GTIN",
    "product_unique_id": "00614141000012",
    "declared_trait": "CO2e",
    "declared_trait_amount": 10,
    "declared_trait_units": "kg",
    "declared_trait_report_type": "CarbonSig",
    "declared_trait_measurement_method": "LCI",
    "declared_trait_measurement_date": "2021-10-12T08:00:00",
    "declared_trait_measurement_data_inputs": "Absolute and Deri
    "declared_trait_measurement_calculation": "Crade to Gate",
    "declared_trait_verifying_entity": "ISO Audit Firm",
    "env_instrument_type": "Carbon Credit",
    "env_instrument_id": "123456789",
    "env_instrument_amount": 45,
    "env_instrument_unit": "kg",
    "business_transaction_type": "Product Order",
    "business_transaction_id_type": "",
    "business_transaction_id": "WO234"
}
```

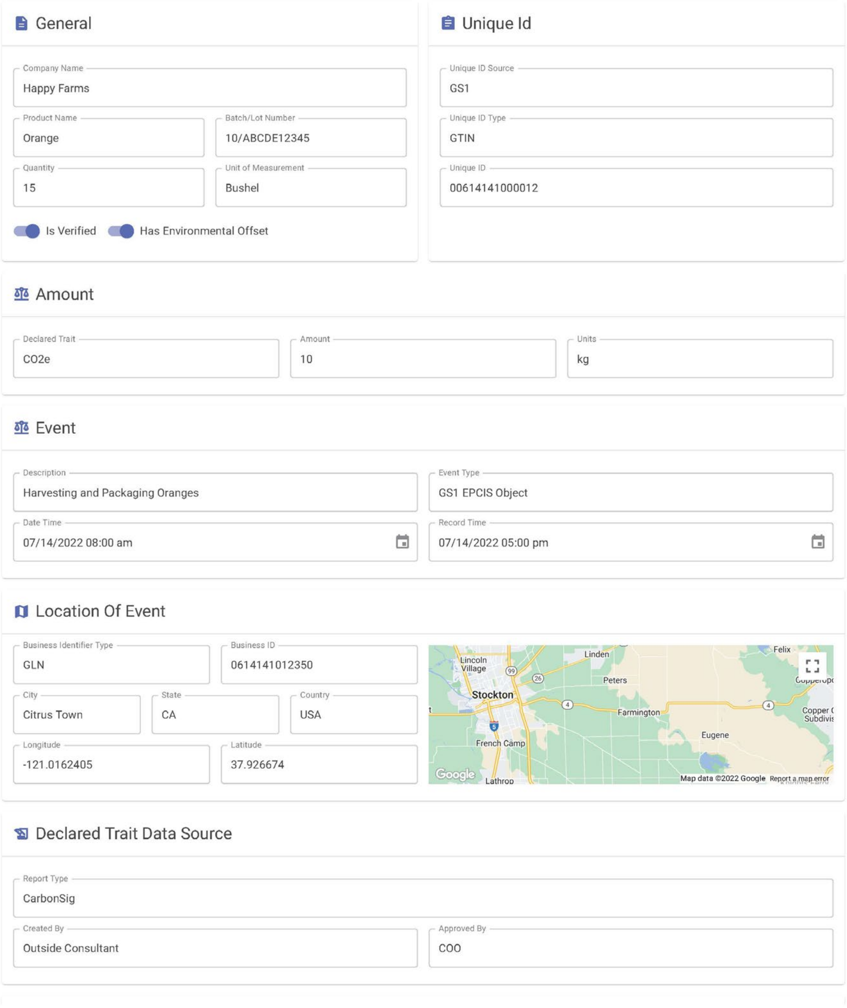
<CarML> Purchase Order – Orange Harvest

- When there is approval for the <CarML> Purchase Order – Orange Harvest message type for the design and data, it can then be generated by clicking the generate button.
- An example of the final <CarML> Purchase Order – Orange Harvest message type is displayed on the following slide.
- All data can be extracted in a JSON or XML format and saved within the recipient’s system, for this use case that is the packaging company which is the next step along the supply chain.
- In addition, all <CarML> message types generated along the supply chain can be enclosed with each new <CarML> message type generated so that, at the final destination all documentation, all <CarML> message types are received.

PoC Message Type - Harvest

<CarML> Standard for CO2e Declaration

Fields can be customized based on message type, product and/or service.



General

Company Name: Happy Farms

Product Name: Orange

Batch/Lot Number: 10/ABCDE12345

Quantity: 15

Unit of Measurement: Bushel

● Is Verified ● Has Environmental Offset

Unique Id

Unique ID Source: GS1

Unique ID Type: GTIN

Unique ID: 00614141000012

Amount

Declared Trait: CO2e

Amount: 10

Units: kg

Event

Description: Harvesting and Packaging Oranges

Event Type: GS1 EPCIS Object

Date Time: 07/14/2022 08:00 am

Record Time: 07/14/2022 05:00 pm

Location Of Event

Business Identifier Type: GLN

Business ID: 0614141012350

City: Citrus Town

State: CA

Country: USA

Longitude: -121.0162405

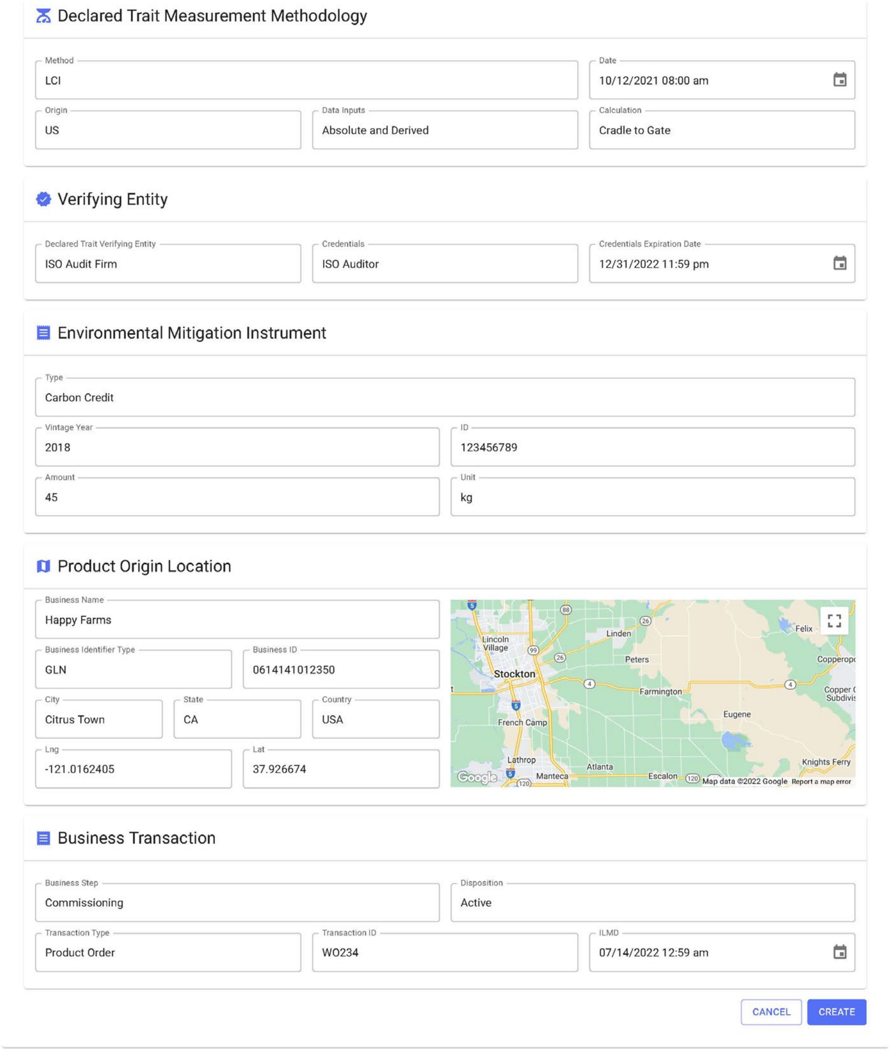
Latitude: 37.926674

Declared Trait Data Source

Report Type: CarbonSig

Created By: Outside Consultant

Approved By: COO



Declared Trait Measurement Methodology

Method: LCI

Date: 10/12/2021 08:00 am

Origin: US

Data Inputs: Absolute and Derived

Calculation: Cradle to Gate

Verifying Entity

Declared Trait Verifying Entity: ISO Audit Firm

Credentials: ISO Auditor

Credentials Expiration Date: 12/31/2022 11:59 pm

Environmental Mitigation Instrument

Type: Carbon Credit

Vintage Year: 2018

ID: 123456789

Amount: 45

Unit: kg

Product Origin Location

Business Name: Happy Farms

Business Identifier Type: GLN

Business ID: 0614141012350

City: Citrus Town

State: CA

Country: USA

Lng: -121.0162405

Lat: 37.926674

Business Transaction

Business Step: Commissioning

Disposition: Active

Transaction Type: Product Order

Transaction ID: WO234

ILMD: 07/14/2022 12:59 am

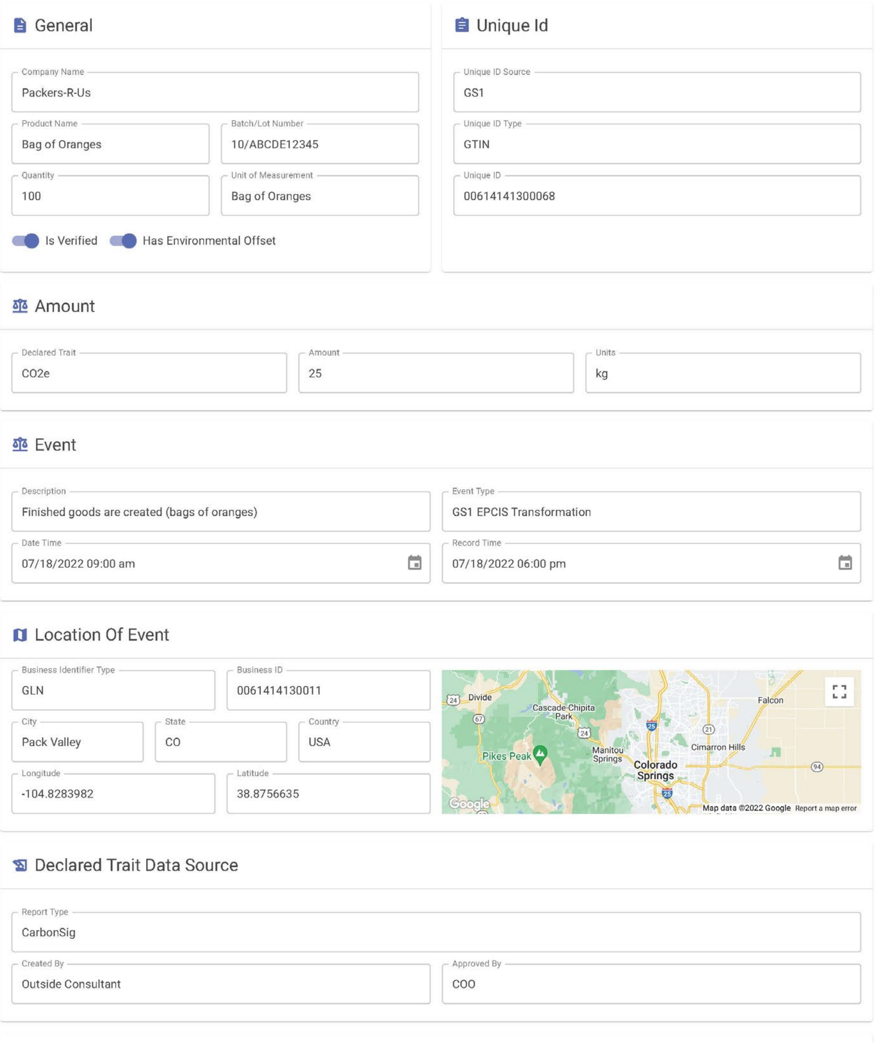
<CarML> Packaging and Shipping Oranges Message Type

- For this use case, the next stop in the supply chain is at the packing shed where the oranges are packed into crates for final transport to the retail grocer.
- If looking at the transaction, this cost is incurred by Happy Farms as part of order fulfillment to the retail grocer.
- Looking at the supply chain and the tracking and traceability for the declared trait of CO2e, this is an important point and process to be declared and included.
  - The CO2e information can also enable both Happy Farms and the Retail Grocer to make more informed service partner and purchasing decisions.
- The following slide shows an example of the <CarML> Packing Shed Message Type and displays the additional CO2e from this step in the process, and additional information regarding environmental offsets utilized.

PoC Message Type - Packing Shed

<CarML> Standard for CO2e Declaration

Fields can be customized based on message type, product and/or service.



General

Company Name	
Packers-R-Us	
Product Name	Batch/Lot Number
Bag of Oranges	10/ABCDE12345
Quantity	Unit of Measurement
100	Bag of Oranges

● Is Verified ● Has Environmental Offset

Unique Id

Unique ID Source
GS1
Unique ID Type
GTIN
Unique ID
00614141300068

Amount

Declared Trait	Amount	Units
CO2e	25	kg

Event

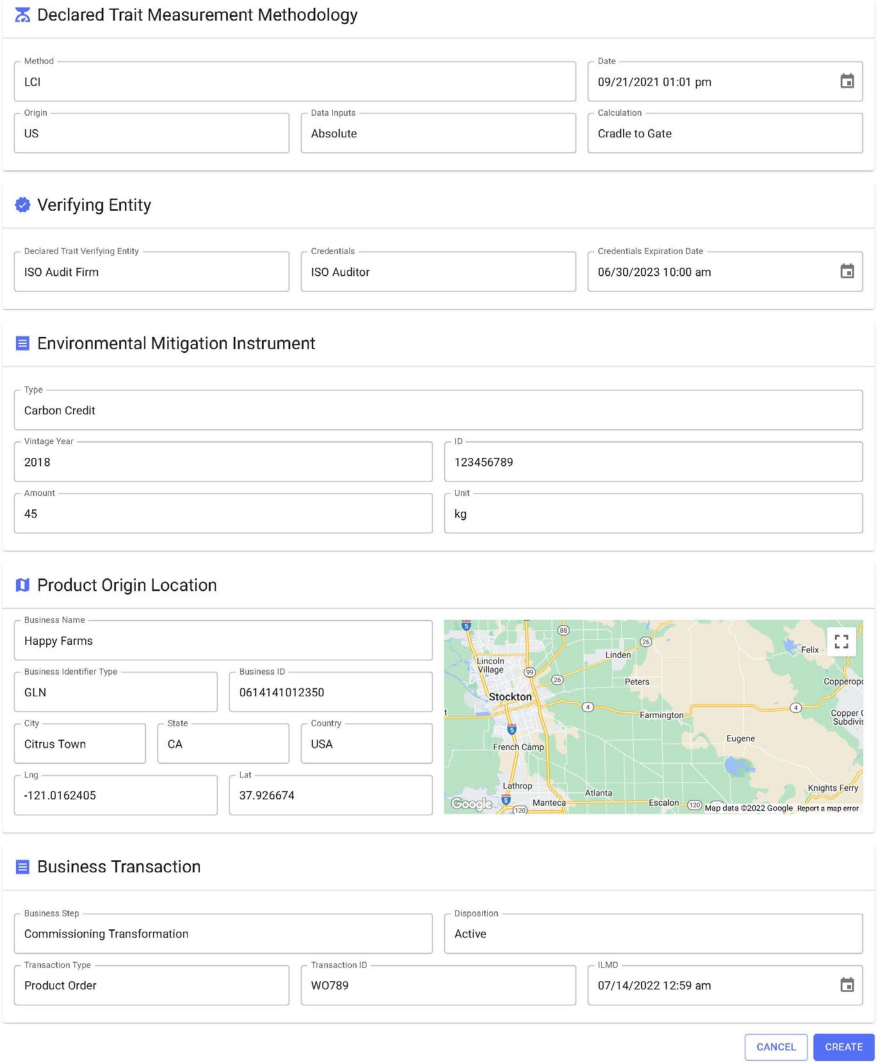
Description	Event Type
Finished goods are created (bags of oranges)	GS1 EPCIS Transformation
Date Time	Record Time
07/18/2022 09:00 am	07/18/2022 06:00 pm

Location Of Event

Business Identifier Type	Business ID	
GLN	0061414130011	
City	State	Country
Pack Valley	CO	USA
Longitude	Latitude	
-104.8283982	38.8756635	

Declared Trait Data Source

Report Type	
CarbonSig	
Created By	Approved By
Outside Consultant	COO



Declared Trait Measurement Methodology

Method		Date
LCI		09/21/2021 01:01 pm
Origin	Data Inputs	Calculation
US	Absolute	Cradle to Gate

Verifying Entity

Declared Trait Verifying Entity	Credentials	Credentials Expiration Date
ISO Audit Firm	ISO Auditor	06/30/2023 10:00 am

Environmental Mitigation Instrument

Type	
Carbon Credit	
Vintage Year	ID
2018	123456789
Amount	Unit
45	kg

Product Origin Location

Business Name		
Happy Farms		
Business Identifier Type	Business ID	
GLN	0614141012350	
City	State	Country
Citrus Town	CA	USA
Lng	Lat	
-121.0162405	37.926674	

Business Transaction

Business Step		Disposition
Commissioning Transformation		Active
Transaction Type	Transaction ID	ILMD
Product Order	WO789	07/14/2022 12:59 am

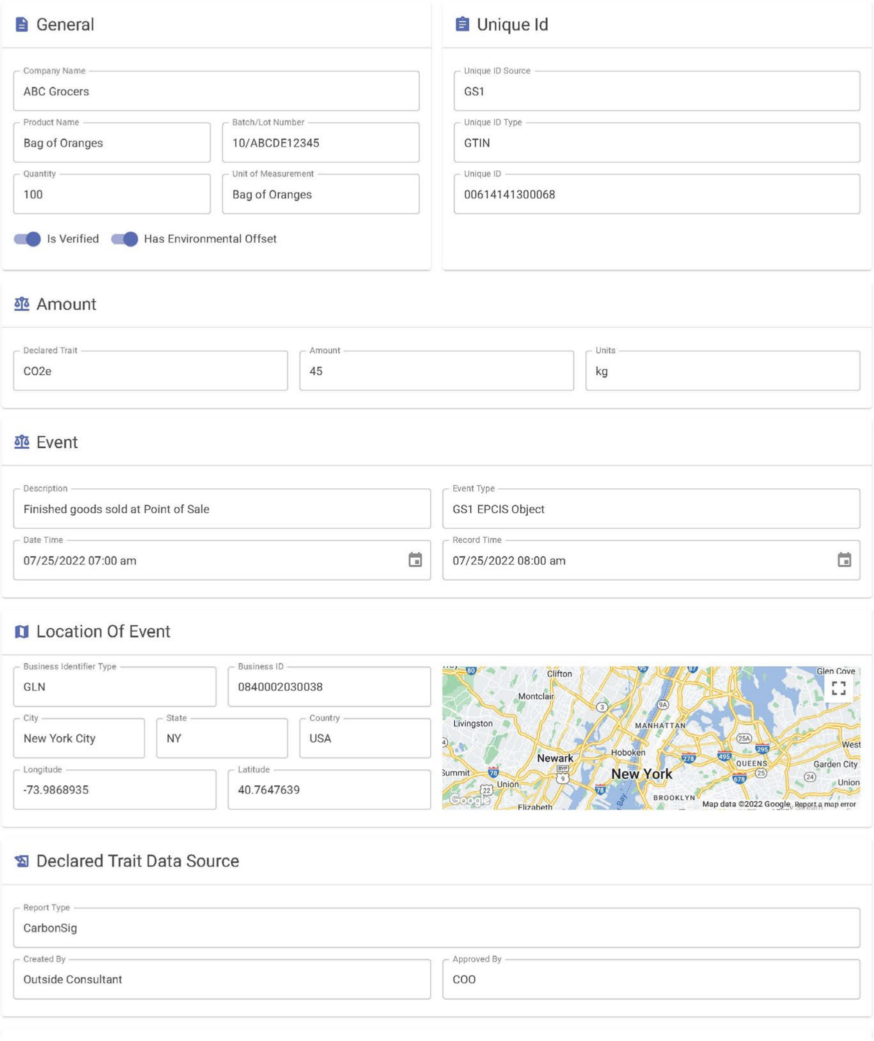
<CarML> Finished Goods Oranges POS Message Type

- For this use case, the final stop in the supply chain is at the retail grocer, for sale of the oranges to the end consumer.
  - However, this use case can be further continued by showing the recycling of any packaging and/or the composting of waste – a circularity use case
- If looking at the transaction, this is the purchase of the oranges by the retail grocer and the fulfillment of that order by Happy Farms.
- Looking at the supply chain and the tracking and traceability for the declared trait of CO2e, this is the final stop along the supply chain for this use case example, and contains additional CO2e and other information.
- The following slide shows an example of the <CarML> Finished Goods POS Message Type and displays the additional CO2e from this step in the process, and additional information regarding environmental offsets utilized.
- And, following that slide is a QR code that has been generated for inclusion on packaging so the consumer can make an informed purchasing decision as well.

PoC Message Type - Finished Goods POS

<CarML> Standard for CO2e Declaration

Fields can be customized based on message type, product and/or service.



General

Company Name	ABC Grocers
Product Name	Bag of Oranges
Batch/Lot Number	10/ABCDE12345
Quantity	100
Unit of Measurement	Bag of Oranges

● Is Verified ● Has Environmental Offset

Unique Id

Unique ID Source	GS1
Unique ID Type	GTIN
Unique ID	00614141300068

Amount

Declared Trait	CO2e
Amount	45
Units	kg

Event

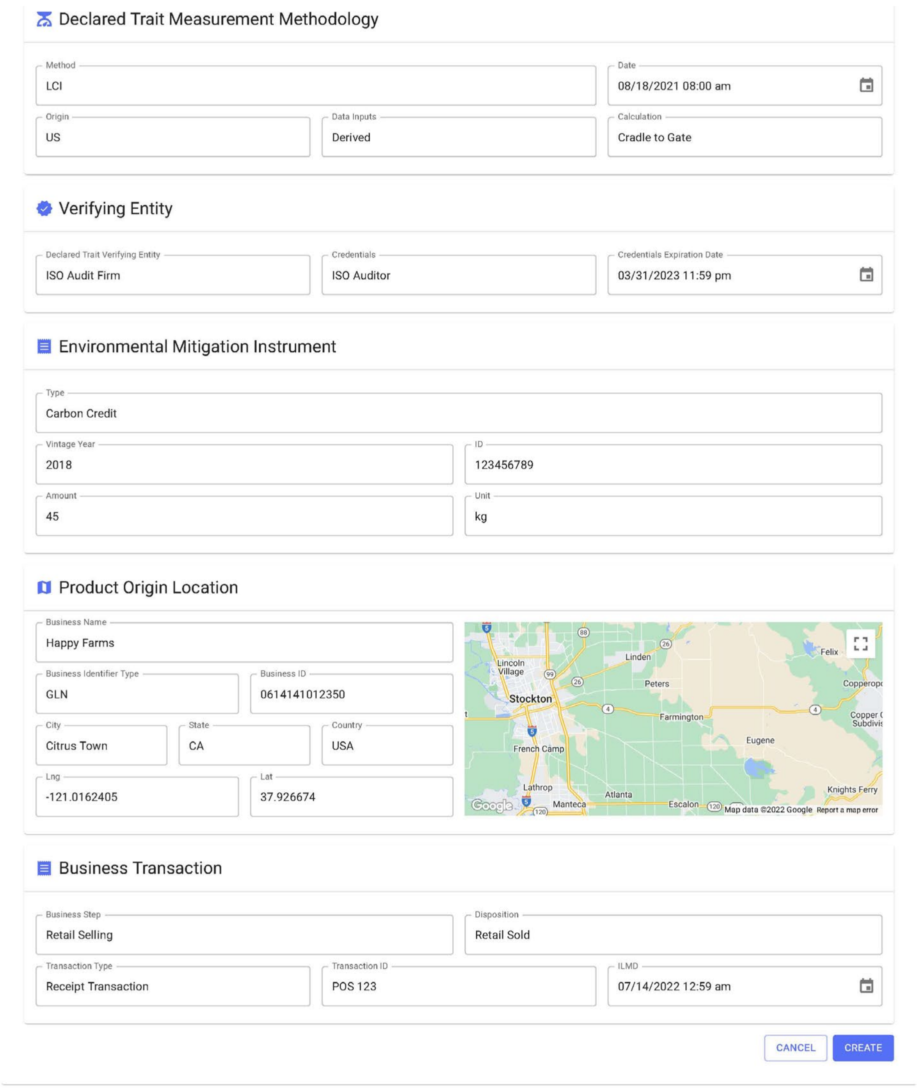
Description	Finished goods sold at Point of Sale
Event Type	GS1 EPCIS Object
Date Time	07/25/2022 07:00 am
Record Time	07/25/2022 08:00 am

Location Of Event

Business Identifier Type	GLN
Business ID	0840002030038
City	New York City
State	NY
Country	USA
Longitude	-73.9868935
Latitude	40.7647639

Declared Trait Data Source

Report Type	CarbonSig
Created By	Outside Consultant
Approved By	COO



Declared Trait Measurement Methodology

Method	LCI
Date	08/18/2021 08:00 am
Origin	US
Data Inputs	Derived
Calculation	Cradle to Gate

Verifying Entity

Declared Trait Verifying Entity	ISO Audit Firm
Credentials	ISO Auditor
Credentials Expiration Date	03/31/2023 11:59 pm

Environmental Mitigation Instrument

Type	Carbon Credit
Vintage Year	2018
ID	123456789
Amount	45
Unit	kg

Product Origin Location

Business Name	Happy Farms
Business Identifier Type	GLN
Business ID	0614141012350
City	Citrus Town
State	CA
Country	USA
Lng	-121.0162405
Lat	37.926674

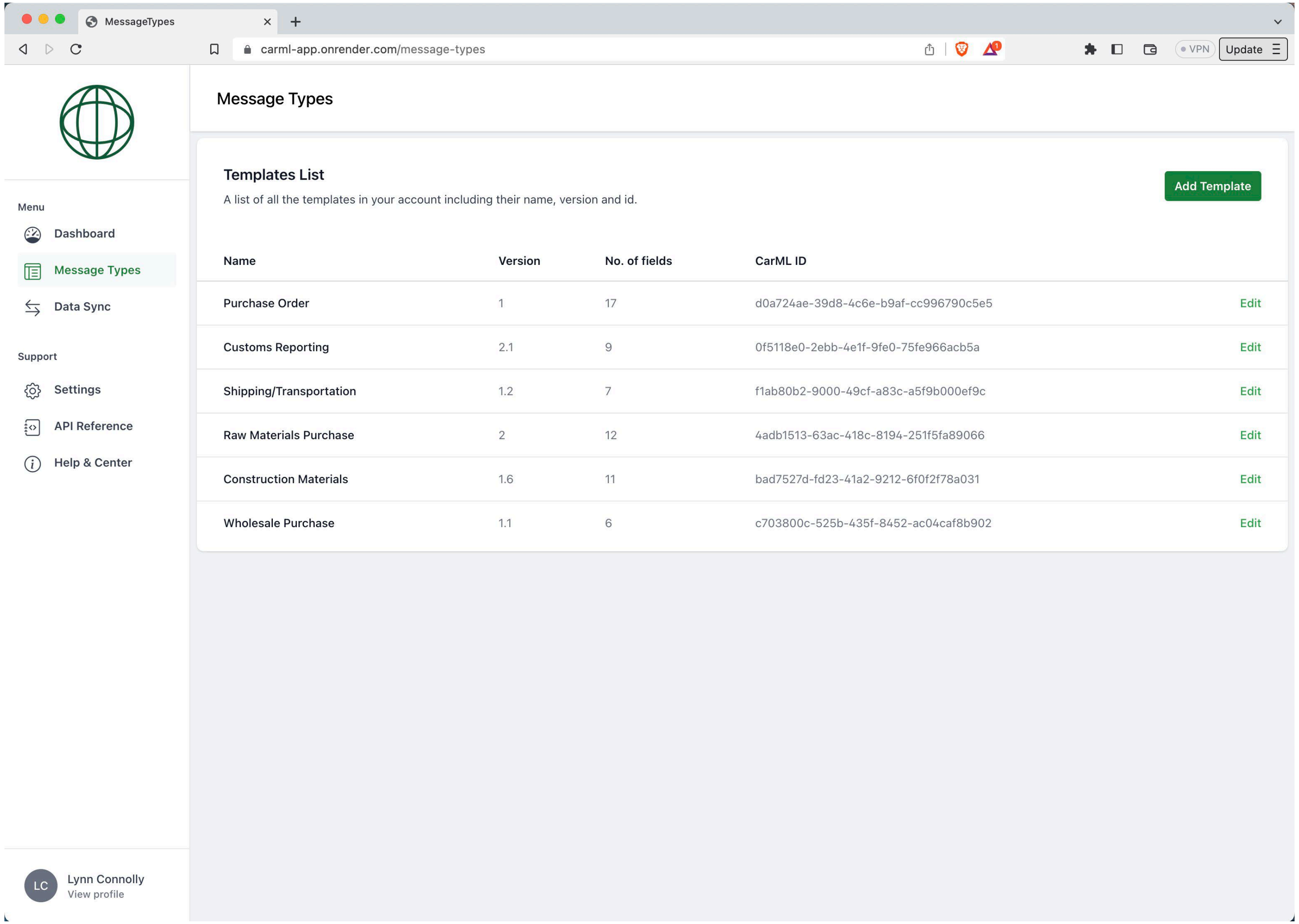
Business Transaction

Business Step	Retail Selling
Disposition	Retail Sold
Transaction Type	Receipt Transaction
Transaction ID	POS 123
ILMD	07/14/2022 12:59 am

<CarML> Oranges Supply Chain Journey QR Code

APPENDIX

Example of Saved <CarML> Message Type Templates including User Created Templates



Message Types

Templates List

A list of all the templates in your account including their name, version and id.

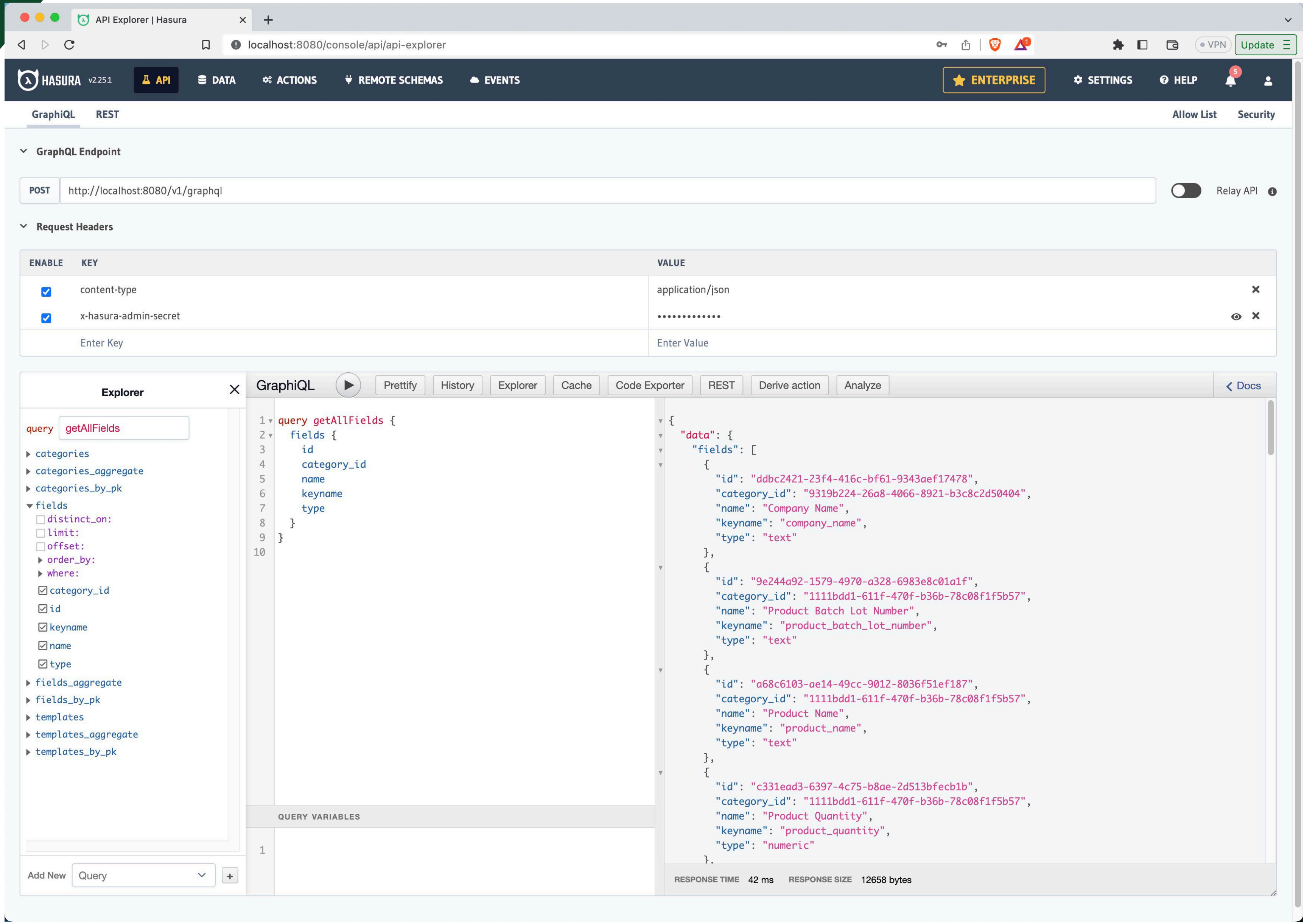
Name	Version	No. of fields	CarML ID	
Purchase Order	1	17	d0a724ae-39d8-4c6e-b9af-cc996790c5e5	Edit
Customs Reporting	2.1	9	0f5118e0-2ebb-4e1f-9fe0-75fe966acb5a	Edit
Shipping/Transportation	1.2	7	f1ab80b2-9000-49cf-a83c-a5f9b000ef9c	Edit
Raw Materials Purchase	2	12	4adb1513-63ac-418c-8194-251f5fa89066	Edit
Construction Materials	1.6	11	bad7527d-fd23-41a2-9212-6f0f2f78a031	Edit
Wholesale Purchase	1.1	6	c703800c-525b-435f-8452-ac04caf8b902	Edit

Carbon-ML-org GitHub Site

- The Carbon-ML-org GitHub site can be accessed through:
  - https://github.com/Carbon-ML-org/
- The site will provide the user with the following:
  - General CarbonML documentation and presentations
  - User Guides for general CarbonML users and for technical users
  - <CarML> Use Cases
  - <CarML> Application and PoC

<CarML> Data Fields and APIs Example

- The <CarML> data fields have been developed utilizing GraphQL APIs. The API supplied with each <CarML> data field has open endpoints to enable the user's technology team to connect each endpoint to the related information within their system's infrastructure.
  - CarbonML.org maintains a <CarML> data field data dictionary which contains the definition and syntax for each <CarML> data field. This data dictionary can be accessed on the CarbonML GitHub site and is also delivered within the Docker container with the <CarML> app.
- For the <CarML> PoC and this use case, we have used GraphQL to access the fields, categories and message types saved within a backend database we created using Postgres.
- The following screenshot of the Hasura GraphQL playground, displays how with the API Explorer one can access all the fields by running a single query. On the left side are the supported ways for running queries, mutations, and subscriptions for live communications through the use of sockets.



POST http://localhost:8080/v1/graphql

Relay API

ENABLE	KEY	VALUE
☑	content-type	application/json
☑	x-hasura-admin-secret	•••••••••••••
	Enter Key	Enter Value

query getAllFields

☑ category_id
☑ id
☑ keyname
☑ name
☑ type

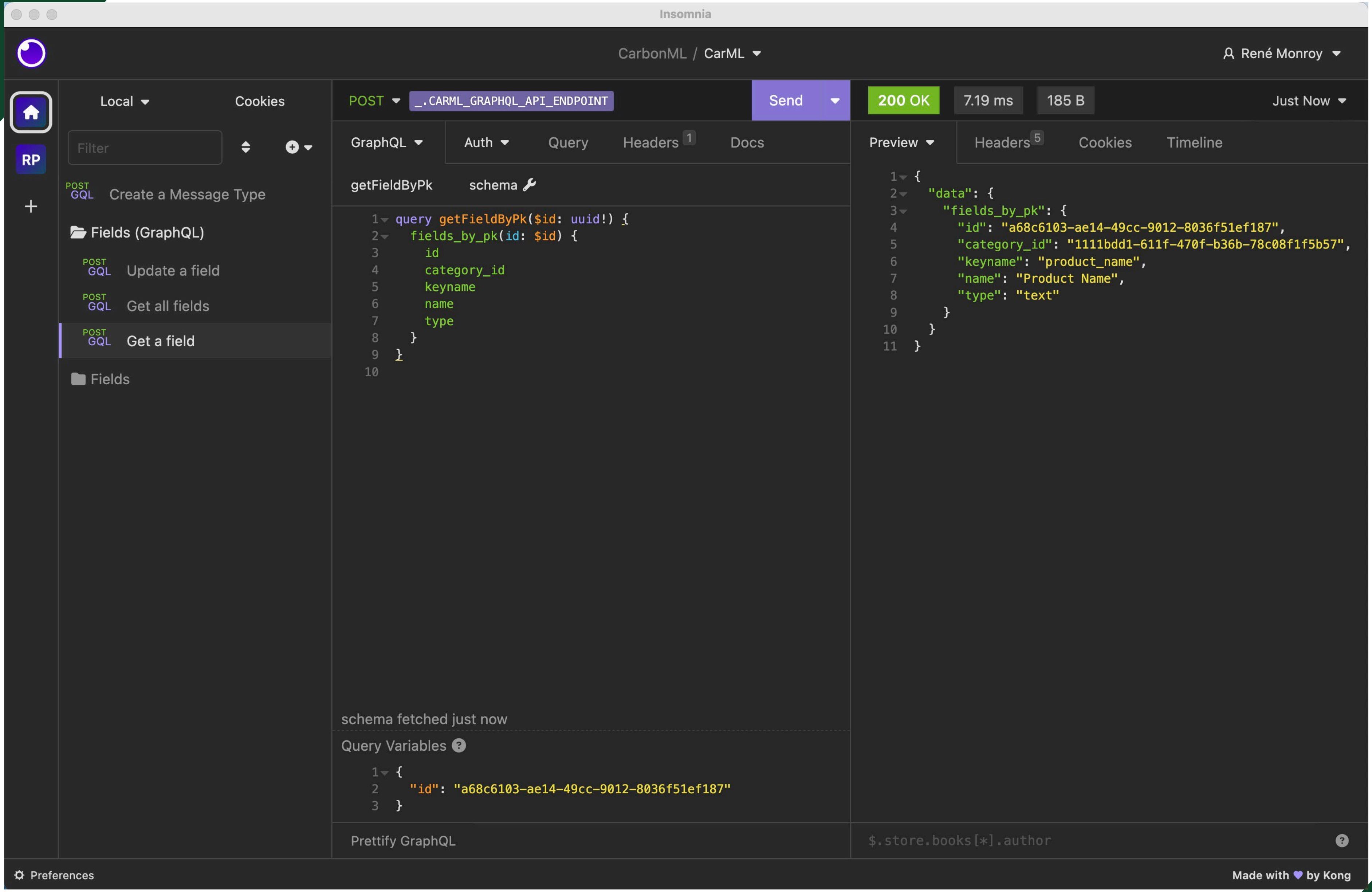
```
query getAllFields {
  fields {
    id
    category_id
    name
    keyname
    type
  }
}
```

```
{
  "data": {
    "fields": [
      {
        "id": "ddbc2421-23f4-416c-bf61-9343aef17478",
        "category_id": "9319b224-26a8-4066-8921-b3c8c2d50404",
        "name": "Company Name",
        "keyname": "company_name",
        "type": "text"
      },
      {
        "id": "9e244a92-1579-4970-a328-6983e8c01a1f",
        "category_id": "1111bdd1-611f-470f-b36b-78c08f1f5b57",
        "name": "Product Batch Lot Number",
        "keyname": "product_batch_lot_number",
        "type": "text"
      },
      {
        "id": "a68c6103-ae14-49cc-9012-8036f51ef187",
        "category_id": "1111bdd1-611f-470f-b36b-78c08f1f5b57",
        "name": "Product Name",
        "keyname": "product_name",
        "type": "text"
      },
      {
        "id": "c331ead3-6397-4c75-b8ae-2d513bfecb1b",
        "category_id": "1111bdd1-611f-470f-b36b-78c08f1f5b57",
        "name": "Product Quantity",
        "keyname": "product_quantity",
        "type": "numeric"
      },
```

RESPONSE TIME 42 ms   RESPONSE SIZE 12658 bytes

<CarML> Data Fields and APIs Example continued

- The following screenshot displays the Insomnia client for use to request a field by a user-determined primary key (the screenshot displays the PoC example) in order to retrieve all of the data for that field.
- In addition, GraphQL can be used to access data through different properties such as text fields, for example keyname.
- <CarML> can also support a REST client by creating custom endpoints to do requests for the same data as shown in the GraphQL example, but in a more direct way.



```
query getFieldByPk($id: uuid!) {
  fields_by_pk(id: $id) {
    id
    category_id
    keyname
    name
    type
  }
}
```

Query Variables

```
{
  "id": "a68c6103-ae14-49cc-9012-8036f51ef187"
}
```

```
{
  "data": {
    "fields_by_pk": {
      "id": "a68c6103-ae14-49cc-9012-8036f51ef187",
      "category_id": "1111bdd1-611f-470f-b36b-78c08f1f5b57",
      "keyname": "product_name",
      "name": "Product Name",
      "type": "text"
    }
  }
}
```

Join Us

www.carbonml.org

Carbon-ML Core Team:

Nick.Gogerty@carbonfinancelab.com

Lynn.Connolly@carbon-ml.org

Rene.Monroy@carbon-ml.org

info@carbon-ml.org

www.carbonml.org