[1. Introduction 2](#_Toc201766045)

[2. System Overview 2](#_Toc201766046)

[2.1 Technologies Used 2](#_Toc201766047)

[2.2 Architecture 3](#_Toc201766048)

[2.3 Dependencies 3](#_Toc201766049)

[3. System Pre-requisites 4](#_Toc201766050)

[4. Configuration Guide 4](#_Toc201766051)

[4.1 Configuration Parameters 4](#_Toc201766052)

[4.2 Environment Setup 5](#_Toc201766053)

[5. Usage Guide 6](#_Toc201766054)

[5.1 Submitting Order Data 6](#_Toc201766055)

[5.2 Checking System Health 7](#_Toc201766056)

[6. API Documentation 7](#_Toc201766057)

[6.1 POST /order\_data\_producer 8](#_Toc201766058)

[6.2 GET /order\_data\_producer/healthCheck 8](#_Toc201766059)

[6.3 Request/Response Formats 8](#_Toc201766060)

[6.4 Authentication and Authorization 8](#_Toc201766061)

[7. Data Models 8](#_Toc201766062)

[7.1 SourceOrderData 8](#_Toc201766063)

[7.2 TargetOrderModel 9](#_Toc201766064)

[8. Version Table 10](#_Toc201766065)

# 1. Introduction

The scope of this document covers the complete technical details of the Order Data Producer Lambda Service, an AWS-based serverless application designed to process, validate, transform, and publish order data to a webhook endpoint. It is intended for developers, testers, system administrators, and stakeholders who need a comprehensive understanding of the system’s functionality, architecture, installation, and maintenance. The document aligns with the requirements outlined in the "Chat | AWS/NodeJs Rampup - Daily call" document and is based on the provided project files.

# 2. System Overview

The Order Data Producer Lambda Service is a serverless application that serves as an interface between a source system and a webhook endpoint (webhook.site). It performs the following functions:

* **Receives order data** in a specific format via an API Gateway endpoint.
* **Validates the data** against predefined rules to ensure correctness.
* **Transforms the data** into a target model format.
* **Publishes the transformed data** to a webhook endpoint.
* **Provides a health check endpoint** for monitoring system status.

The system ensures robust error handling, centralized logging, and dynamic configuration retrieval from AWS Systems Manager (SSM) Parameter Store.

## 2.1 Technologies Used

The system leverages the following technologies:

* **Runtime Environment**: Node.js with TypeScript
* **Serverless Framework**: AWS Lambda
* **API Integration**: AWS API Gateway
* **Configuration Management**: AWS SSM Parameter Store
* **HTTP Client**: Axios (for publishing to webhook.site)
* **Logging Framework**: Winston (for centralized logging)
* **Validation**: Custom validation logic with UUID package
* **Environment Management**: dotenv (for local development environment variables)

## 2.2 Architecture

The Order Data Producer follows a serverless architecture, integrated with AWS services. The key components and flow are as follows:

* **Client Interaction**: Clients interact with the system via API Gateway endpoints:
  + **POST /order\_data\_producer**: Submits order data for processing.
  + **GET /order\_data\_producer/healthCheck**: Verifies system health.
* **Lambda Processing**: The AWS Lambda function (index.ts) handles requests by:
  + Validating input data using validator.ts.
  + Transforming data into the target format using transformer.ts.
  + Publishing transformed data to a webhook using publisher.ts.
  + Retrieving the webhook URL from AWS SSM Parameter Store.
  + Logging all activities using logger.ts.

**Webhook Integration**: Transformed data is sent to a webhook endpoint (e.g., <https://webhook.site/7ad3f036-dc8b-4e3c-8fa3-252dca653cdd>.

*Architecture Diagram*:  
A computer screen shot of a diagram

AI-generated content may be incorrect.

## 2.3 Dependencies

The system relies on the following external dependencies:

* **AWS SDK**: For interacting with AWS SSM Parameter Store.
* **Axios**: For making HTTP POST requests to the webhook endpoint.
* **Winston**: For centralized logging of requests, transformations, and errors.
* **UUID**: For optional validation of unique identifiers.
* **dotenv**: For managing environment variables during local development.

# 3. System Pre-requisites

Before deploying or running the Order Data Producer Lambda Service, ensure the following components are installed and configured:

* **Node.js (v16 or later)**: Required for running the TypeScript application. Download from https://nodejs.org.
* **AWS CLI**: For deploying and managing AWS resources. Install from https://aws.amazon.com/cli.
* **AWS Account**: With permissions to create and manage Lambda functions, API Gateway, and SSM Parameter Store.
* **TypeScript**: Install globally using npm install -g typescript.
* **Jest (optional)**: For running unit tests. Install using npm install --save-dev jest.

# 4. Configuration Guide

The system requires configuration for development and production environments. Key settings are stored in AWS SSM Parameter Store or environment variables (via .env file for local development).

## 4.1 Configuration Parameters

* **SSM Parameter**:

**/order/producer/webhook-url**: Stores the webhook URL (, <https://webhook.site/7ad3f036-dc8b-4e3c-8fa3-252dca653cdd>). Must be encrypted using SecureString.

* **Environment Variables** (for local development, stored in .env file):
  + **AWS\_REGION**: AWS region (e.g., us-east-1).
  + **AWS\_ACCESS\_KEY\_ID**: AWS access key (optional for local testing with AWS CLI credentials).
  + **AWS\_SECRET\_ACCESS\_KEY**: AWS secret key (optional for local testing).

*Example .env file*:

AWS\_REGION=us-east-1

AWS\_ACCESS\_KEY\_ID=your\_access\_key

AWS\_SECRET\_ACCESS\_KEY=your\_secret\_key

## 4.2 Environment Setup

**Development Environment**

1. **Install Dependencies**:
   * Clone the repository: git clone <repository\_url>.
   * Navigate to the project folder: cd <project-folder>.
   * Install dependencies: npm install.
2. **Configure Environment Variables**:
   * Create a .env file in the project root.
   * Add the required variables (e.g., AWS\_REGION=us-east-1).
3. **Set Up SSM Parameter (for local testing)**:
   * Use AWS CLI to create a parameter:  
     aws ssm put-parameter --name "/order/producer/webhook-url" --value "https://webhook.site/[unique-id]" --type SecureString.
4. **Run Locally**:
   * Compile TypeScript code: tsc.
   * Run unit tests: npx jest.

**Production Environment**

1. **Deploy Lambda Function**:
   * Package the application: zip -r function.zip ..
   * Create or update the Lambda function using AWS CLI:  
     aws lambda create-function --function-name order-data-producer --runtime nodejs16.x --role <lambda-role-arn> --handler index.handler --zip-file fileb://function.zip  
     or  
     aws lambda update-function-code --function-name order-data-producer --zip-file fileb://function.zip.
2. **Configure API Gateway**:
   * Create a REST API in API Gateway.
   * Set up routes:
     + POST /order\_data\_producer: Linked to the Lambda function.
     + GET /order\_data\_producer/healthCheck: Linked to the Lambda function.
   * Deploy the API to a stage (e.g., prod).
3. **Store Webhook URL in SSM**:
   * Create an encrypted parameter:  
     aws ssm put-parameter --name "/order/producer/webhook-url" --value "https://webhook.site/[unique-id]" --type SecureString.
4. **Grant Permissions**:
   * Attach the AmazonSSMReadOnlyAccess IAM policy to the Lambda execution role to allow access to SSM Parameter Store.

# 5. Usage Guide

The Order Data Producer Lambda Service enables users to submit order data for processing and check the system's health. Below are the instructions for interacting with the system.

## 5.1 Submitting Order Data

* **Endpoint**: POST /order\_data\_producer
* **Purpose**: Submits order data for validation, transformation, and publishing to a webhook.
* **Request Body**: JSON payload conforming to the SourceOrderData interface (see types.ts).
* **Example Request**:

{

"orderId": "ORD-12345",

"orderDate": "10/15/2023",

"customerId": "CUST-789",

"storeId": 42,

"items": [

{ "sku": "PROD-001", "quantity": 2, "unitPrice": 29.99, "discountAmount": 5.00 },

{ "sku": "PROD-002", "quantity": 1, "unitPrice": 49.99 }

],

"paymentMethod": "CREDIT\_CARD",

"shippingAddress": {

"street": "123 Main St",

"city": "Columbus",

"state": "OH",

"zipCode": "43215",

"country": "USA"

},

"totalAmount": 104.97,

"status": "NEW",

"notes": "Please deliver after 5pm"

}

* **Success Response** (HTTP 200):

{ "status": true, "orderId": "ORD-12345" }

* **Error Response** (HTTP 400, Validation Error):

{ "errors": ["orderId must start with 'ORD-'"] }

* **Error Response** (HTTP 500, Internal Server Error):

{ "message": "Internal server error", "error": "Error details" }

## 5.2 Checking System Health

* **Endpoint**: GET /order\_data\_producer/healthCheck
* **Purpose**: Verifies that the Lambda function is deployed and operational.
* **Response** (HTTP 200):

{ "status": "healthy" }

# 6. API Documentation

The Order Data Producer exposes two API endpoints via AWS API Gateway to handle order processing and health checks.

## 6.1 POST /order\_data\_producer

* **Description**: Accepts order data, validates it, transforms it into the target format, and publishes it to a webhook.
* **Request**:
  + **Body**: JSON object conforming to SourceOrderData.
* **Responses**:
  + **200 OK**: { "status": true, "orderId": string }
  + **400 Bad Request**: { "errors": string[] }
  + **500 Internal Server Error**: { "message": string, "error": string }

## 6.2 GET /order\_data\_producer/healthCheck

* **Description**: Returns the operational status of the Lambda function.
* **Request**: None.
* **Response**:
  + **200 OK**: { "status": "healthy" }

## 6.3 Request/Response Formats

* **Request Format**: JSON for POST requests to /order\_data\_producer. No body for GET requests to /healthCheck.
* **Response Format**: JSON for all responses, with appropriate HTTP status codes and data or error messages.

## 6.4 Authentication and Authorization

* **Authentication**: The current implementation does not require authentication. Future enhancements may include API key or OAuth 2.0-based authentication for secure access.

# 7. Data Models

The system uses two data models defined in types.ts to handle input and output data.

## 7.1 SourceOrderData

* **Description**: The input data model for order data submitted to the Lambda.
* **Fields**:
  + **orderId**: String, must start with "ORD-".
  + **orderDate**: String, format "MM/DD/YYYY".
  + **customerId**: String, unique identifier for the customer.
  + **storeId**: Number, must be positive.
  + **items**: Array of objects with:
    - **sku**: String, required.
    - **quantity**: Number, must be positive.
    - **unitPrice**: Number, must be non-negative.
    - **discountAmount**: Number, optional, must be non-negative.
  + **paymentMethod**: String, required.
  + **shippingAddress**: Optional object with:
    - **street**: String.
    - **city**: String.
    - **state**: String.
    - **zipCode**: String.
    - **country**: String.
  + **totalAmount**: Number, must be non-negative.
  + **status**: String, one of "NEW", "PROCESSING", "SHIPPED", "DELIVERED", "CANCELLED".
  + **notes**: String, optional.

## 7.2 TargetOrderModel

* **Description**: The output data model sent to the webhook.
* **Fields**:
  + **order**:
    - **id**: String, from orderId.
    - **createdAt**: String, date in "YYYY-MM-DD" format.
    - **customer**: Object with id (from customerId).
    - **location**: Object with storeId (stringified).
    - **status**: String, lowercase status.
    - **payment**: Object with method and total.
    - **shipping**: Object with address containing line1, city, state, postalCode, country (defaults to empty strings if not provided).
  + **items**: Array of objects with:
    - **productId**: String, from sku.
    - **quantity**: Number.
    - **price**: Object with base, discount, and final (calculated as quantity \* unitPrice - discountAmount).
  + **metadata**:
    - **source**: String, set to "order\_producer".
    - **notes**: String, from notes or empty.
    - **processedAt**: String, ISO timestamp of processing.

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Comment** |
| V1.0 | Shubhashree | 06/25/2025 | Initial draft |