A close up of a sign

Description generated with high confidence

**Build an Order Processing Systems Using Azure Integration Services**

*Matthew Farmer*

Contents

[Objective 3](#_Toc4528487)

[Let’s build something Real! 3](#_Toc4528488)

[The brief, if you choose to accept it 3](#_Toc4528489)

[The Design 4](#_Toc4528490)

[Hints 5](#_Toc4528491)

[General Hints 5](#_Toc4528492)

[Logic App 1 Hints 5](#_Toc4528493)

[Logic Apps 2 Hints 5](#_Toc4528494)

[More details on the solution 6](#_Toc4528495)

[Logic Apps 01 6](#_Toc4528496)

[Sample Schema 6](#_Toc4528497)

[AddProperty Details 7](#_Toc4528498)

[Logic Apps 02 8](#_Toc4528499)

# Objective

## Let’s build something Real!

This exercise will take you through building a ‘real world’ scenario showing how Azure Integration Services components fit together.

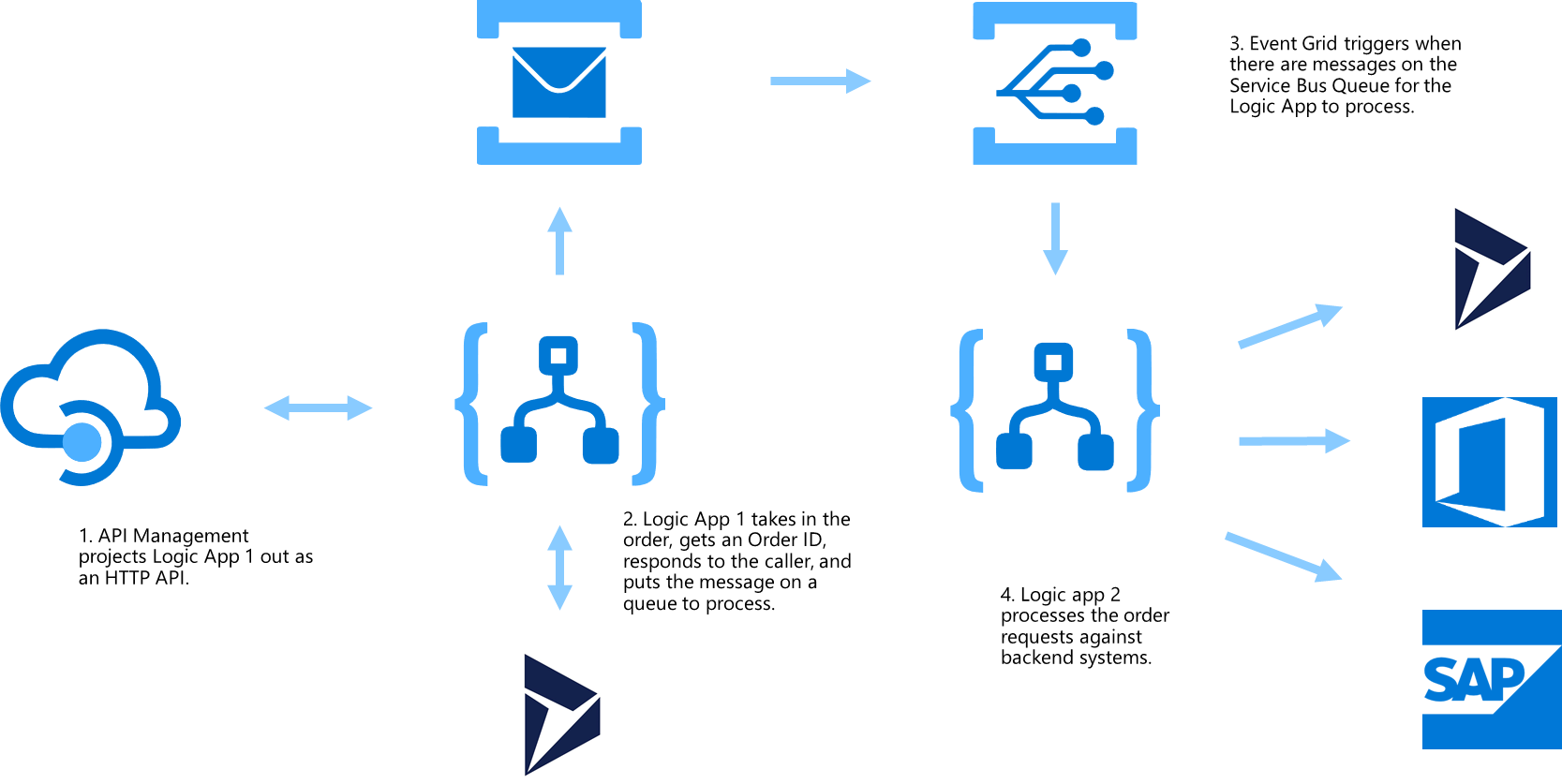
We are going to build a system that will receive orders, and process them with a number of backend systems. We won’t be guiding you step-by-step, but we will give you hints on what you need to achieve.

## The brief, if you choose to accept it

* Your system needs to receive orders, and process them with a number of backend systems
* The system is called by a number of clients over HTTP. The payload they will provide to you is already defined (JSON format).
* You need to reply to the client with an order id so they know their order has been received safely. You should do this quickly so as not to delay them.
* Backend systems can take a while to process orders.
* Patterns of orders coming in is highly variable across the week. Make sure your system is efficient.

Think about how you can meet this requirement, before looking at the proposed design

# The Design



This is not your typical Global Integration Bootcamp lab. In this lab, we are giving full reigns of the solution, so you have a chance to create it in a way that the requirements are met.

But if you need some hints, take a look at the next page for some help.

# Hints

## General Hints

* If you don’t already have an API Management instance to use, start provisioning one first (this can take 30 minutes)
* You will need to provision Service Bus premium tier to enable Event Grid support.
* The Event Grid event, and the Logic App that receive it (Logic App 2), can be provisioned from the Service Bus Events menu.

## Logic App 1 Hints

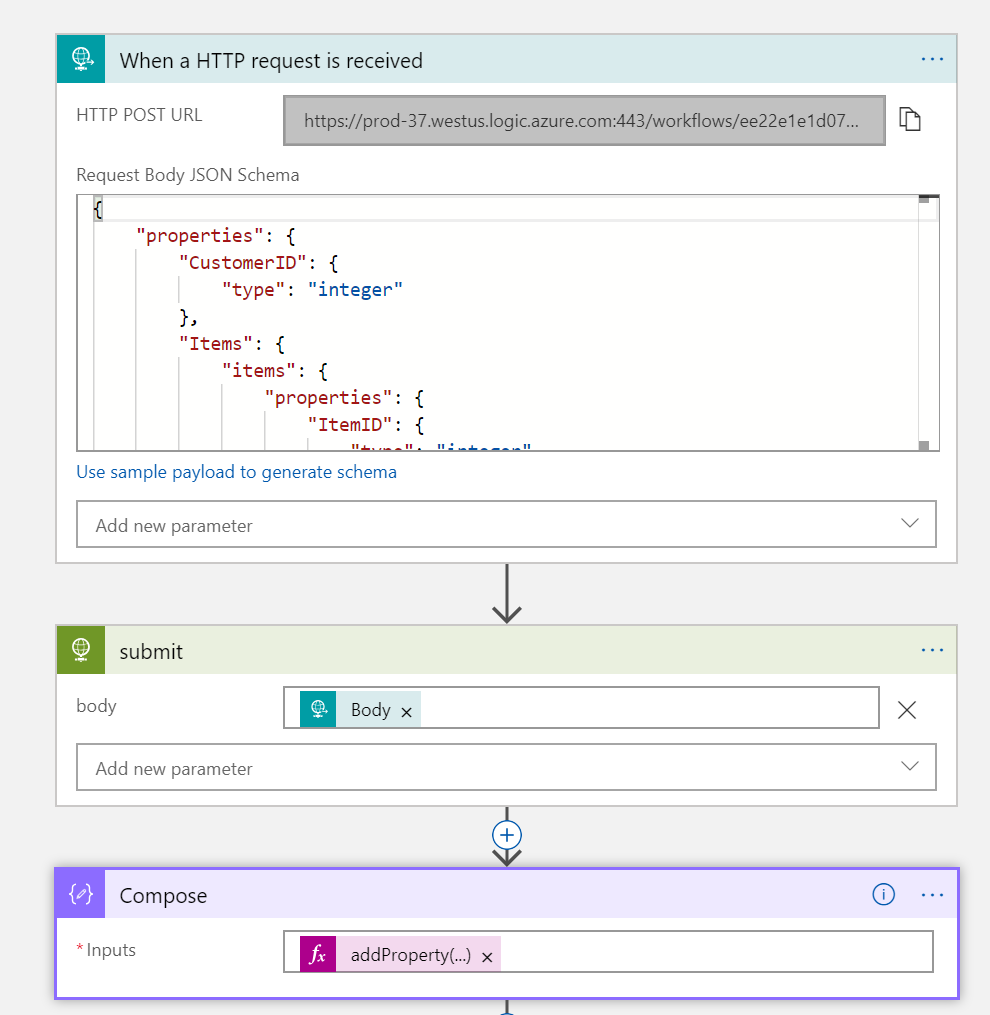
* Ensure you validate your input! Use a schema to validate the payload, and generate properties for use in the Logic App.
* The example shows us getting an Order ID from a backend system. We’re going to use a mock service to do this (in place of a CRM system, for example). Use an HTTP + Swagger action, pointing at the Open API definition at
* https://fabrikamapim.portal.azure-api.net/docs/services/order/export?DocumentFormat=Swagger
* You will need to add the OrderID to the incoming order before passing it to Service Bus. Use a compose action with the addProperty() function to do this.

## Logic Apps 2 Hints

* Use simple connectors to prove the use case (e.g. post to Teams/Slack, send email).
* You should use the peek-lock pattern to lock messages on the queue, and complete them when you are done.
* You will get messages from Service Bus in groups. You will need to loop through them.

# More details on the solution

## Logic Apps 01



### Sample Schema

**{**

**"properties": {**

**"CustomerID": {**

**"type": "integer"**

**},**

**"Items": {**

**"items": {**

**"properties": {**

**"ItemID": {**

**"type": "integer"**

**},**

**"Quantity": {**

**"type": "integer"**

**}**

**},**

**"required": [**

**"ItemID",**

**"Quantity"**

**],**

**"type": "object"**

**},**

**"type": "array"**

**},**

**"PromoCode": {**

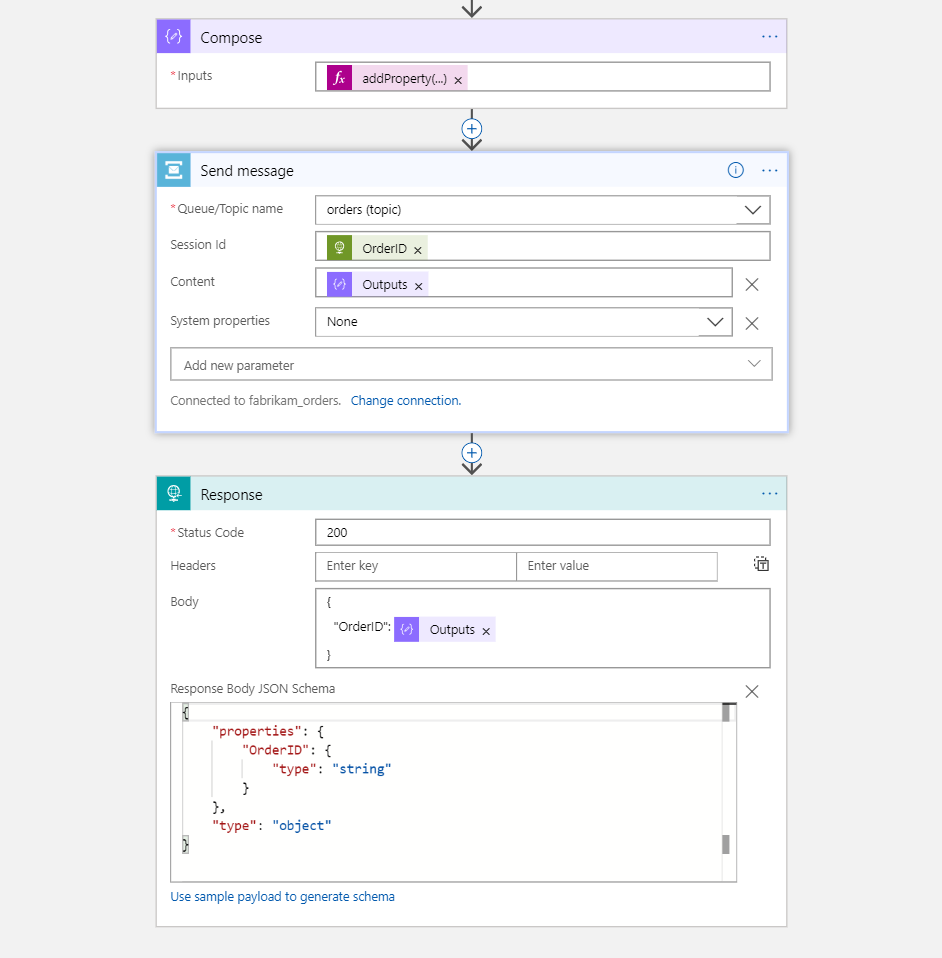
**"type": "string"**

**}**

**},**

**"type": "object"**

**}**



### AddProperty Details

addProperty**(**triggerBody**(),** 'OrderID'**,** body**(**'submit'**)?[**'OrderID'**])**

## Logic Apps 02

