valtech_

Getting Started with Optimizely B2B Commerce

Alain Marsolais
Optimizely Practice Director

Agenda

Section 01

About Me

About Me

- I have been working with Valtech_(formerly Absolunet) for 7 years, starting has a senior developer, team-lead, practice-lead and now as the Optimizely Practice Director
- Gold OMVP, Platinum SME, B2BCommerce & Content Cloud Certified Developer
- I really enjoy anything that has to do with software architecture and systems integration.
- I am also lead guitar in a Radiohead Tribute Band

Section 02

Introduction

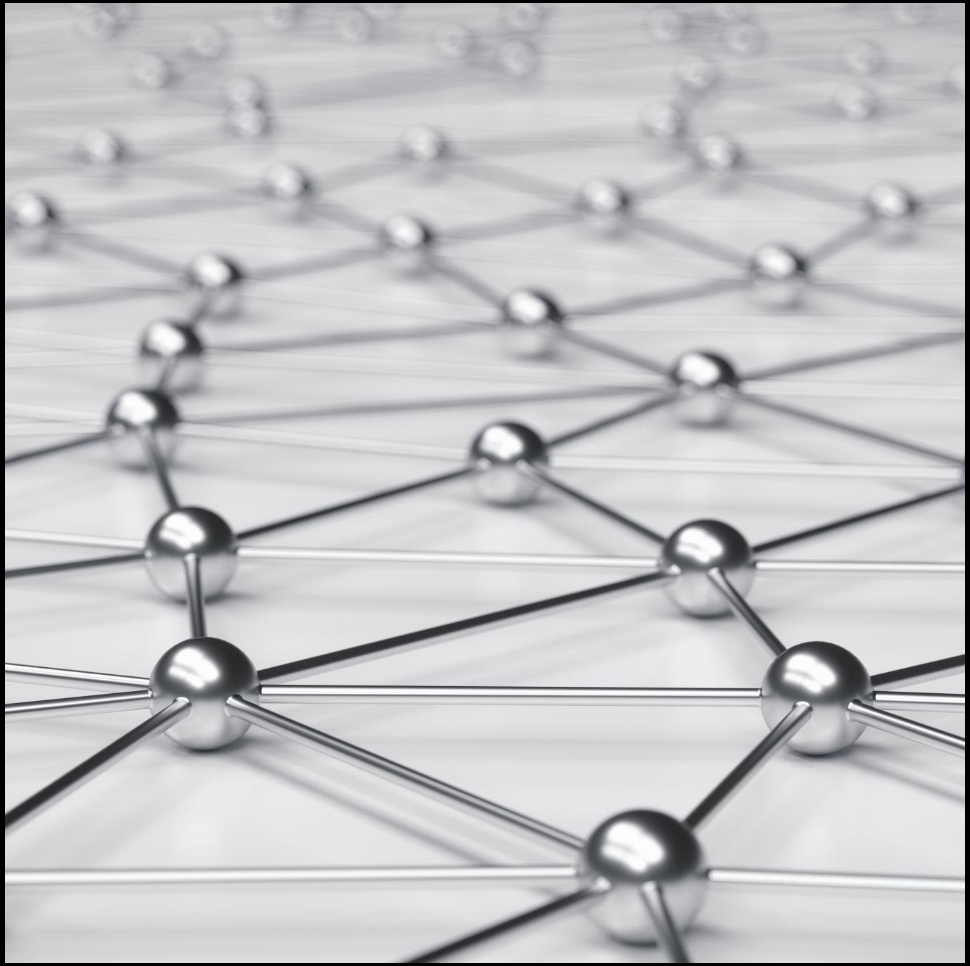
What it is

- Formally known as Insite Commerce, the solution was renamed Optimizely B2B Commerce after Insite Software was acquired by Optimizely in 2019.
- Designed specifically for the complexities of B2B commerce, the platform offers more out-of-the-box B2B functionality that any of its competitors
- Fit most of the mid-market to small-enterprise business needs, especially in the distribution & manufacturers sectors
- Built with Headless Commerce APIs, a strong Presentation Layer (CMS), and a robust integration layer to connect to any external system (ERP, PIM, CRM, …).
- Available in a SAAS cloud offering with monthly updates.

The Platform

/ B2B Commerce Cloud differs from other B2B solutions in that it is natively deep and rich with features and functionality, while remaining fully extensible and integrates smoothly with other external systems (ERPs, Tax services, Payment Gateways, etc.)

/ It is not a toolkit, but a platform that is highly configurable and requires an in-depth knowledge of our Administration Console & Service API layer to execute many business and/or technical requirements.



Feature List

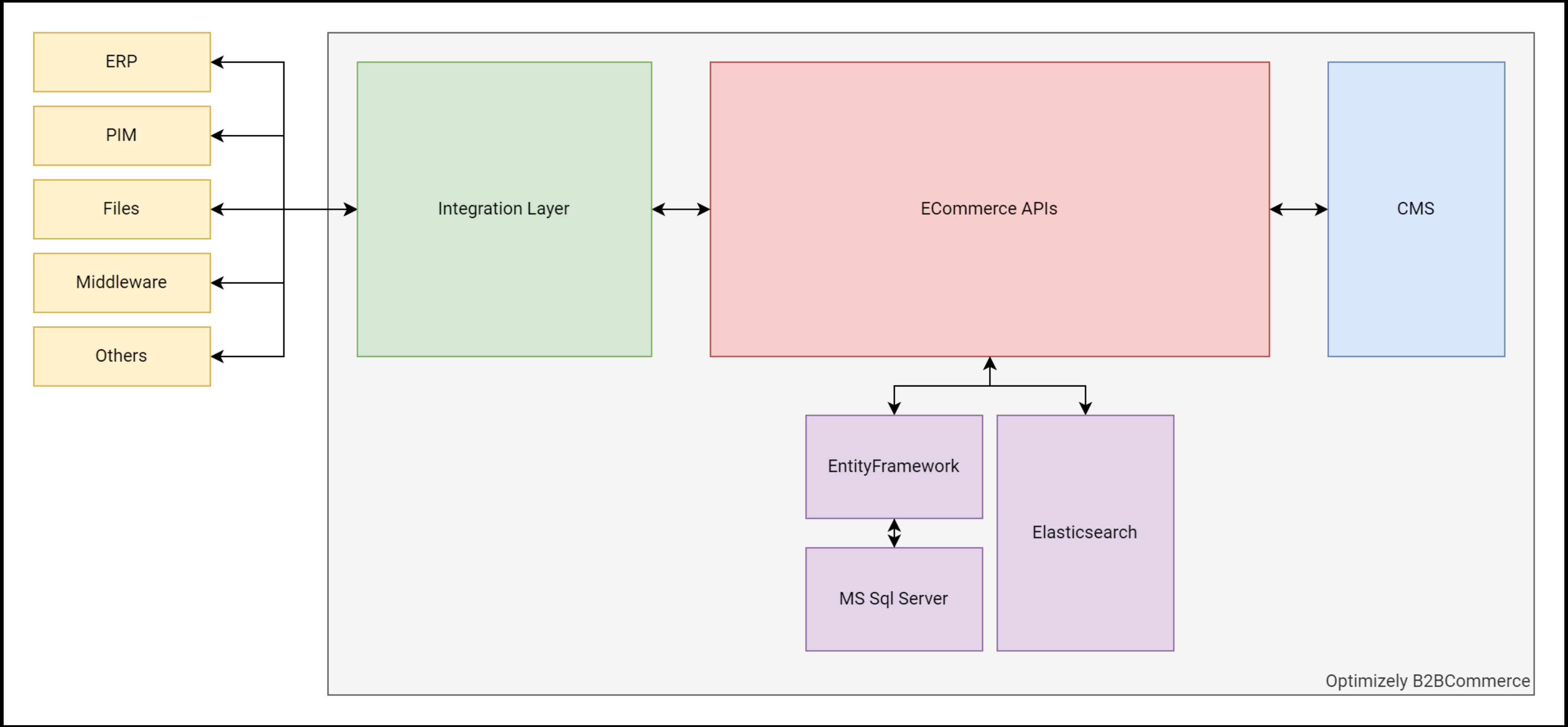
- Full Product Catalog (PDP, PLP, CDP)
- Bill To / Ship To Customers
- Product Restrictions
- Brand Merchandising
- Search & Faceting
- Cart & Checkout
- Order & Invoice History
- Quick Order & Reorder
- Shipping & Pickup in Store
- Multiple Payment Methods including CC & E-Check
- Quotes, Order Approval & Budget Management
- Promotions
- List & Saved Orders
- User Management
- Punchout Ordering
- Multiple website configurations
- Personalization & Customer Segmentation
- Return Request (RMA) & Order Cancelation
- Vendor Managed Inventory (VMI)
- And many more

Demo

Section 03

Global Architecture

Global Architecture



Section 04

CMS

Spire CMS

- Newer version of the content management system (CMS) offered with your Optimizely B2B Commerce license.
  - Released in April 2020
  - Met parity with Classic (previous CMS) in October 2020
- Built using the latest front-end technologies like React, Redux, styled-components and more
- Lighter & Faster page delivery, and Faster API Endpoints (Product V2, …)

- SEO / Server-Side Rendering (SSR)
- Better support for Accessibility
- Improved upgradability
- Catalog & commerce pages template
- Simplified CMS Data Structure
- Improved page editing experience

Classic CMS

- Previous version of the content management system (CMS) offered in your Optimizely B2B Commerce license

2022 2023 2024 2025

January 2022
End Of Support of AngularJS, the Optimizely B2B Commerce Classic CMS frontend library

January 2023 to December 2024
Optimizely B2B Commerce announced that *“new features and capabilities will only be added to Spire, but we will address base code bugs & critical security issues in Classic”* through December 2024.

January 2025
Optimizely B2B Commerce announced that *“this will be the official End of Life for Classic. There will be no base code changes made to Classic after this date”*

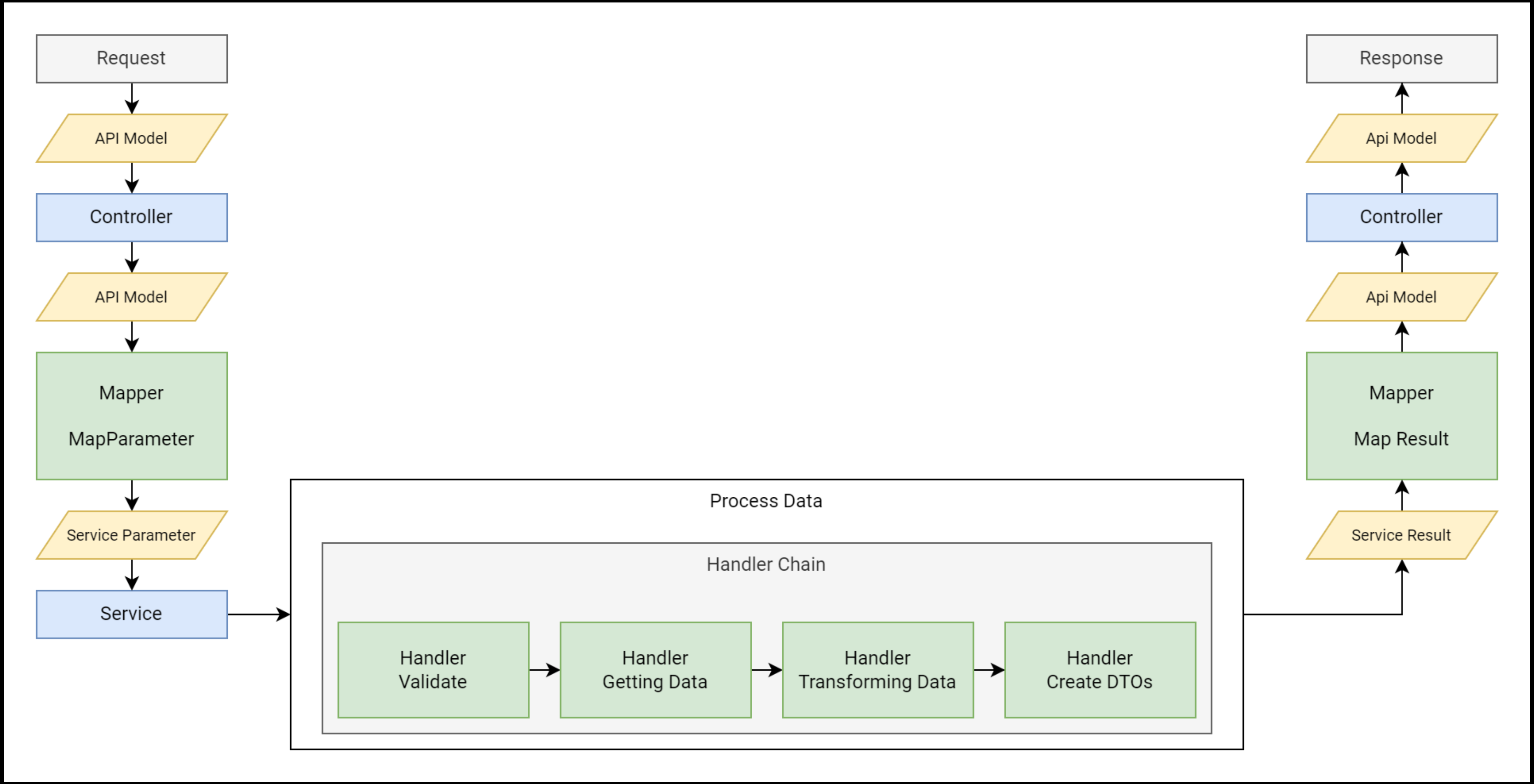
Section 05

Storefront Commerce API

Storefront API

- REST API built with ASP.NET MVC Web API
- Contains all the logic for the standard B2B Commerce features provided in the base code
- Easy to customize and adapt to your customers' needs using extension points.
- A lot of effort has been put into breaking down the code into smaller pieces with a single responsibility.
- All APIs definitions are available via an Swagger integration

Storefront API architecture



Section 04-02

Handlers

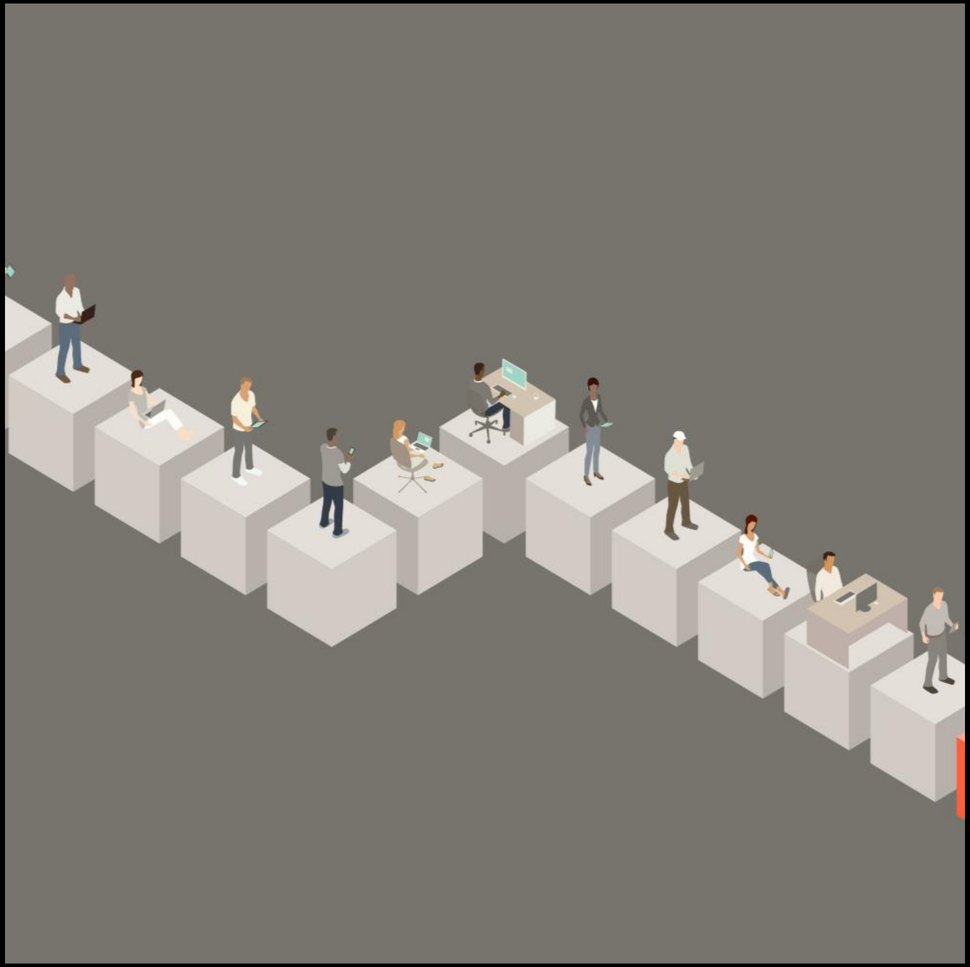
Definition

/ A handler chain is an extension point in the Optimizely B2B platform. It allows you to add custom logic during a request for information from the platform.

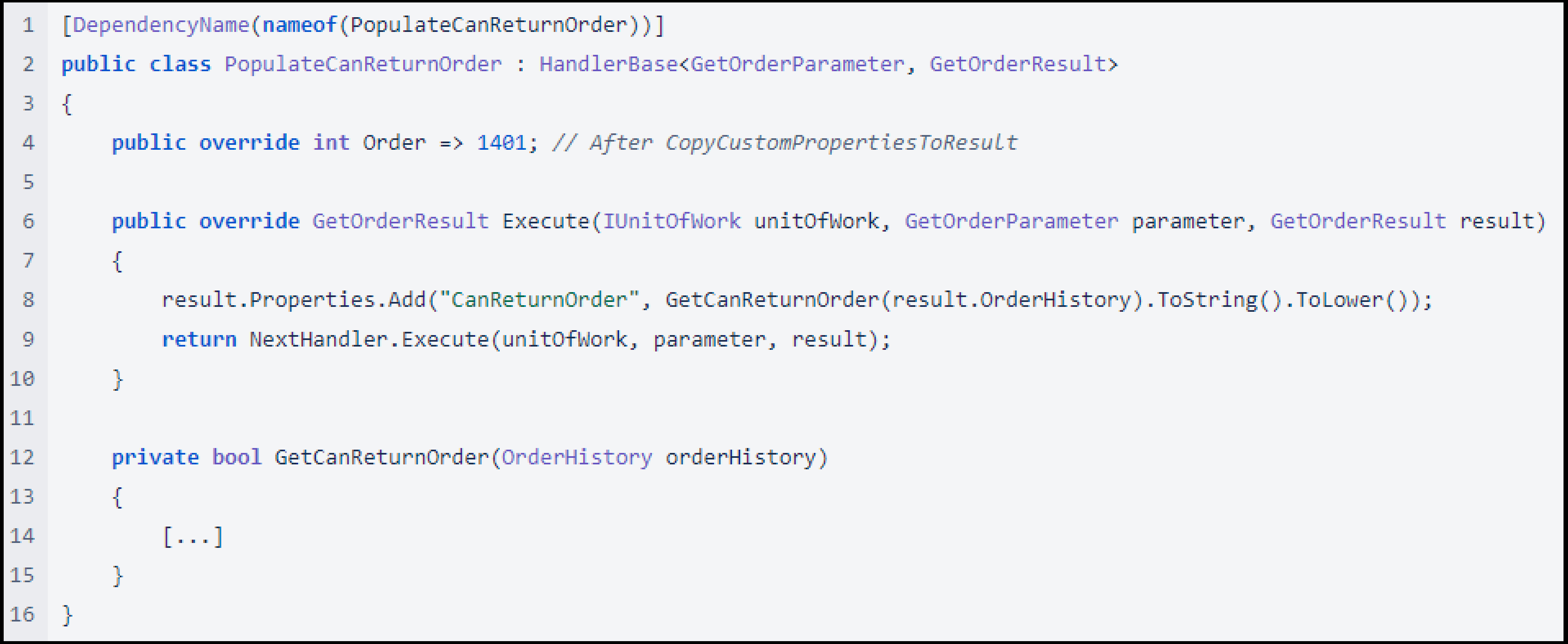
/ The handler chain follows the chain of responsibility design pattern. This chain is made of links, which are called "Handlers". A handler class is a portion of the handler chain execution.

/ Each handler is responsible for calling the next handler in the chain. It can also decide to terminate the chain immediately.

/ It is possible to add and/or replace handlers in an existing handler chain in order to update the behavior of the platform.



Example #1



```
[DependencyName(nameof(PopulateCanReturnOrder))]
public class PopulateCanReturnOrder : HandlerBase<GetOrderParameter, GetOrderResult>
{
    public override int Order => 1401; // After CopyCustomPropertiesToResult

    public override GetOrderResult Execute(IUnitOfWork unitOfWork, GetOrderParameter parameter, GetOrderResult result)
    {
        result.Properties.Add("CanReturnOrder", GetCanReturnOrder(result.OrderHistory).ToString().ToLower());
        return NextHandler.Execute(unitOfWork, parameter, result);
    }

    private bool GetCanReturnOrder(OrderHistory orderHistory)
    {
        [...]
    }
}
```

Section 04-03

Pipelines

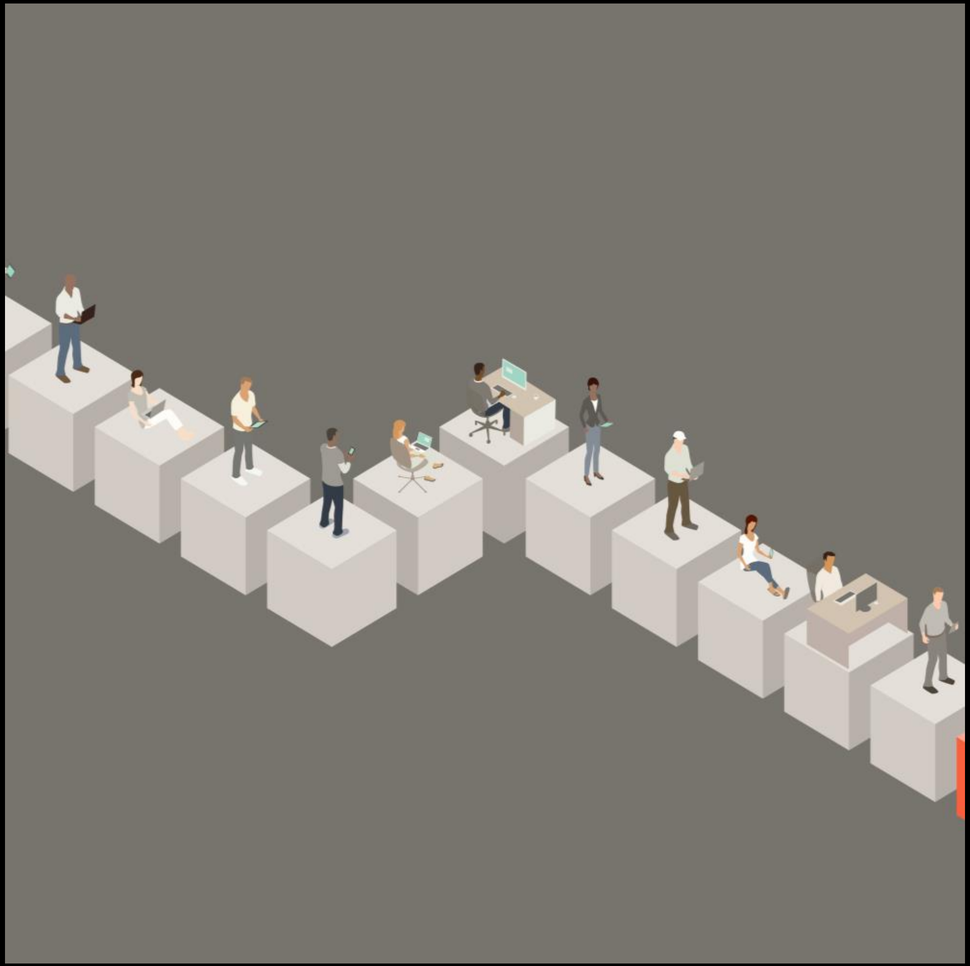
Definition

/ A pipeline is an extension point in the Optimizely B2B platform. It allows you to add custom logic during a request for information from the platform.

/ Very similar to handler chains, it follows the principles of the chain of responsibility design pattern. A pipeline is made of classes, which are called "pipes".

/ A pipe class is a portion of the pipeline execution. Each pipe is responsible for returning the result of its operations. The result is then provided to the next pipe class in the pipeline.

/ It is also possible to add and/or replace pipe in an existing pipeline in order to update the behavior of the platform.

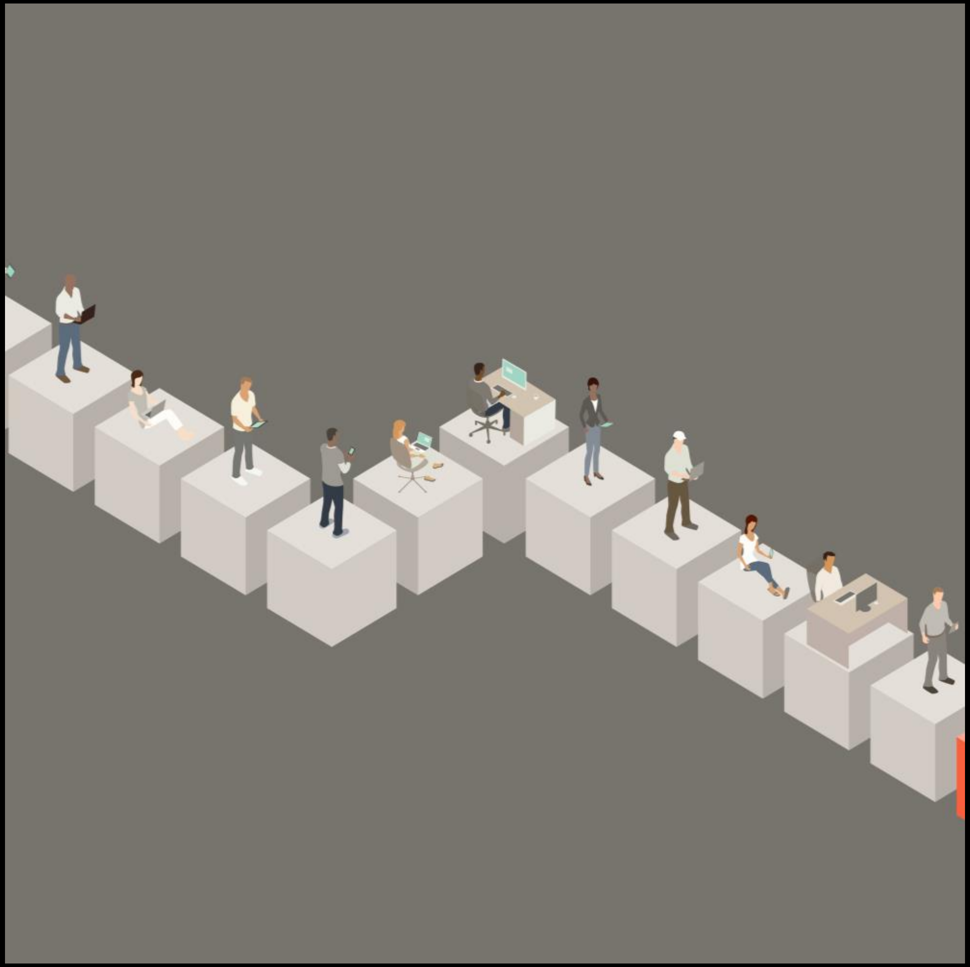


Differences from Handlers

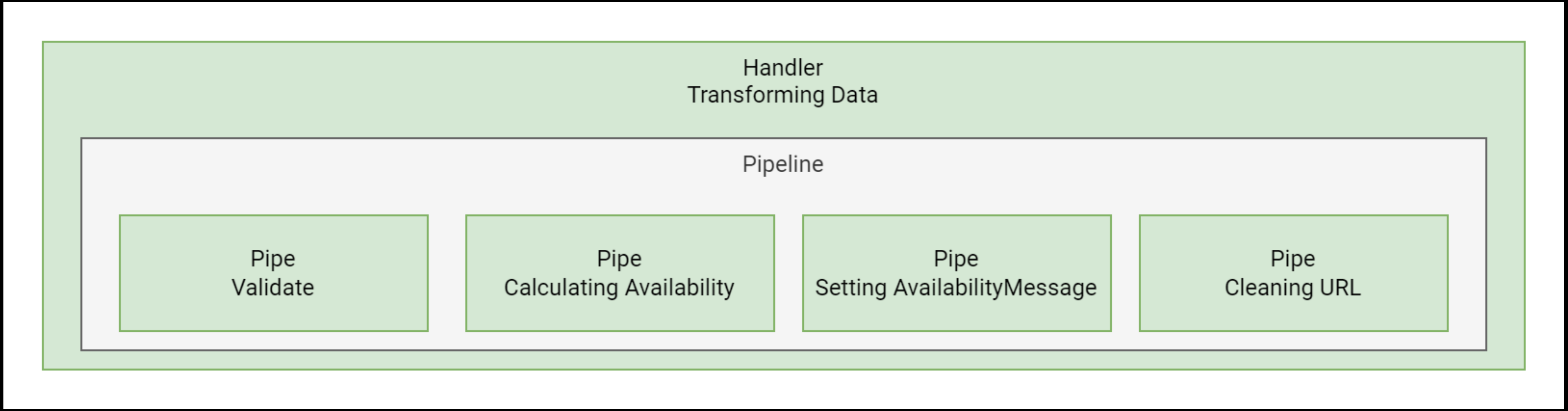
/ Basically, a pipeline has a reusable business logic. The idea is that several handler chains might potentially want to execute the same pipeline versus a handler chain that will be more related to a direct call to the storefront API.

/ No need to specify the DependencyName attribute on a pipe class

/ The Execute method of a pipe class simply returns the result. No need to call the next pipe class.

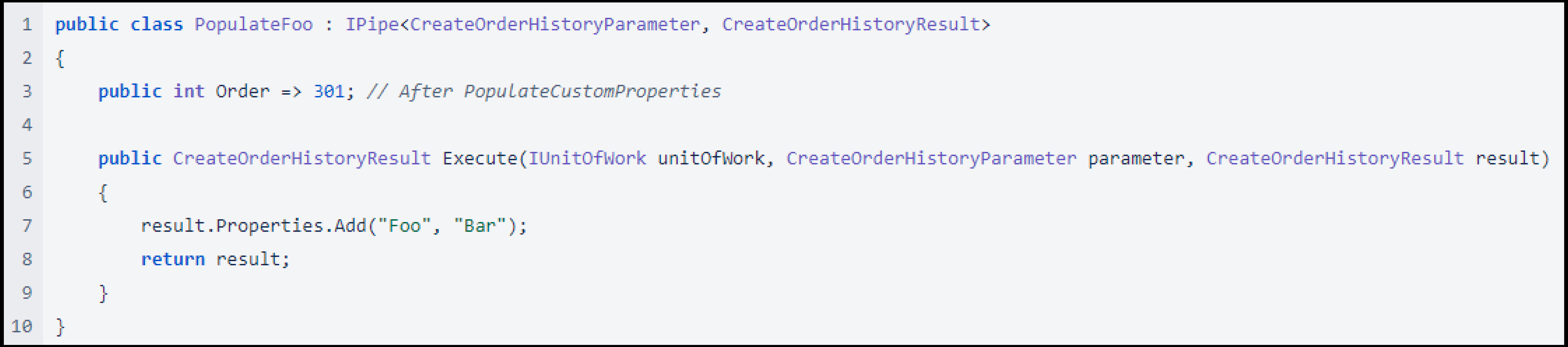


Storefront API architecture



Additional information

- It is best to always name our extension pipe classes with meaningful names indicating the operation performed by the pipeline pipe.
- It is also recommended to indicate in the comment the time of execution of the pipe, i.e., the name of the pipe class executing before or after our extension pipe.



```
public class PopulateFoo : IPipe<CreateOrderHistoryParameter, CreateOrderHistoryResult>
{
    public int Order => 301; // After PopulateCustomProperties

    public CreateOrderHistoryResult Execute(IUnitOfWork unitOfWork, CreateOrderHistoryParameter parameter, CreateOrderHistoryResult result)
    {
        result.Properties.Add("Foo", "Bar");
        return result;
    }
}
```

Section 04-01

Mappers

Introduction

- Mappers are extension points of the Optimizely B2B Commerce platform.
- These objects are used to bridge the gap between the API presentation layer and the business logic layer (services) for both input and output.
- For the input, we refer to the term parameters, and for the output, we refer to the term results.
- Optimizely provides a set of mapper classes to support all its API input and output points. It is possible to inherit from these classes in order to customize or modify the default behavior.
- However, it is important to understand the responsibility of a mapper class before customizing it in order to follow the best development practices.
- When you derive from a mapper class, you must first define the constructor in order to respect one of the inherited class. Then, two methods are at your disposal: MapParameter and MapResult.

MapParameter Example

```
public class StoreGetCountryMapper : GetCountryMapper
{
    public StoreGetCountryMapper(IGetStateMapper getStateMapper, IObjectToObjectMapper objectToObjectMapper, IUrlHelper urlHelper) : base(getStateMapper, objectToObjectMapper, urlHelper)
    {
    }

    public override GetCountryParameter MapParameter(string countryId, HttpRequestMessage request)
    {
        var parameter = base.MapParameter(countryId, request);

        // A mapper is an appropriate place to provide additionnal parameters coming from the HTTP
        // request for the upcoming handler chain that will read this information and perform
        // business logic around it.
        parameter.Properties.Add("AppendRandomStringToCountryName", request.GetQueryString("AppendRandomStringToCountryName"));

        return parameter;
    }
}
```

MapResult Example

```
public class StoreGetCountryMapper : GetCountryMapper
{
    public StoreGetCountryMapper(IGetStateMapper getStateMapper, IObjectToObjectMapper objectToObjectMapper, IUrlHelper urlHelper) : base(getStateMapper, objectToObjectMapper, urlHelper)
    {
    }

    public override CountryModel MapResult(GetCountryResult serviceResult, HttpRequestMessage request)
    {
        var countryModel = base.MapResult(serviceResult, request);

        // A mapper is an appropriate place to add properties to the API model,
        // for any entity field that you would like to have available on the frontend
        countryModel.Properties.Add("BlockAnonymousTraffic", serviceResult.Country.BlockAnonymousTraffic.ToString());

        // A mapper is an appropriate place to add "display" properties
        // that can become handy on the frontend
        countryModel.Properties.Add("StatesCountDisplay", $"This country contains {countryModel.States.Count} states.");

        return countryModel;
    }
}
```

Section 06

Admin Commerce API

Admin API

- / OData API built with ASP.NET MVC web API.
- / Exposes the application’s data on a database level.
- / Allows querying and manipulation of the application’s data.
- / Can also be useful for debugging purposes
- / No customization is allowed in this API
- / All APIs definitions are available via an Swagger integration

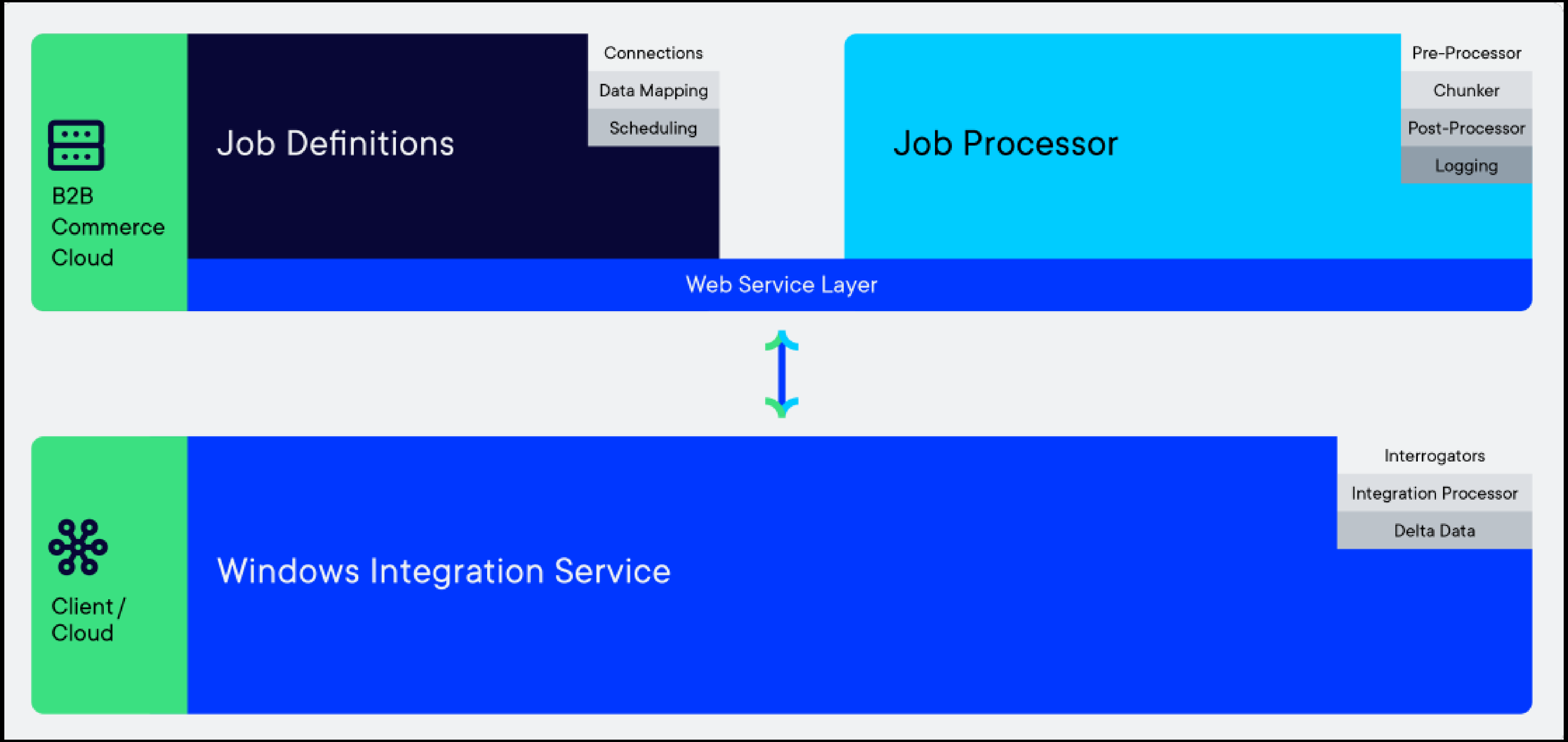
Section 07

System Integration

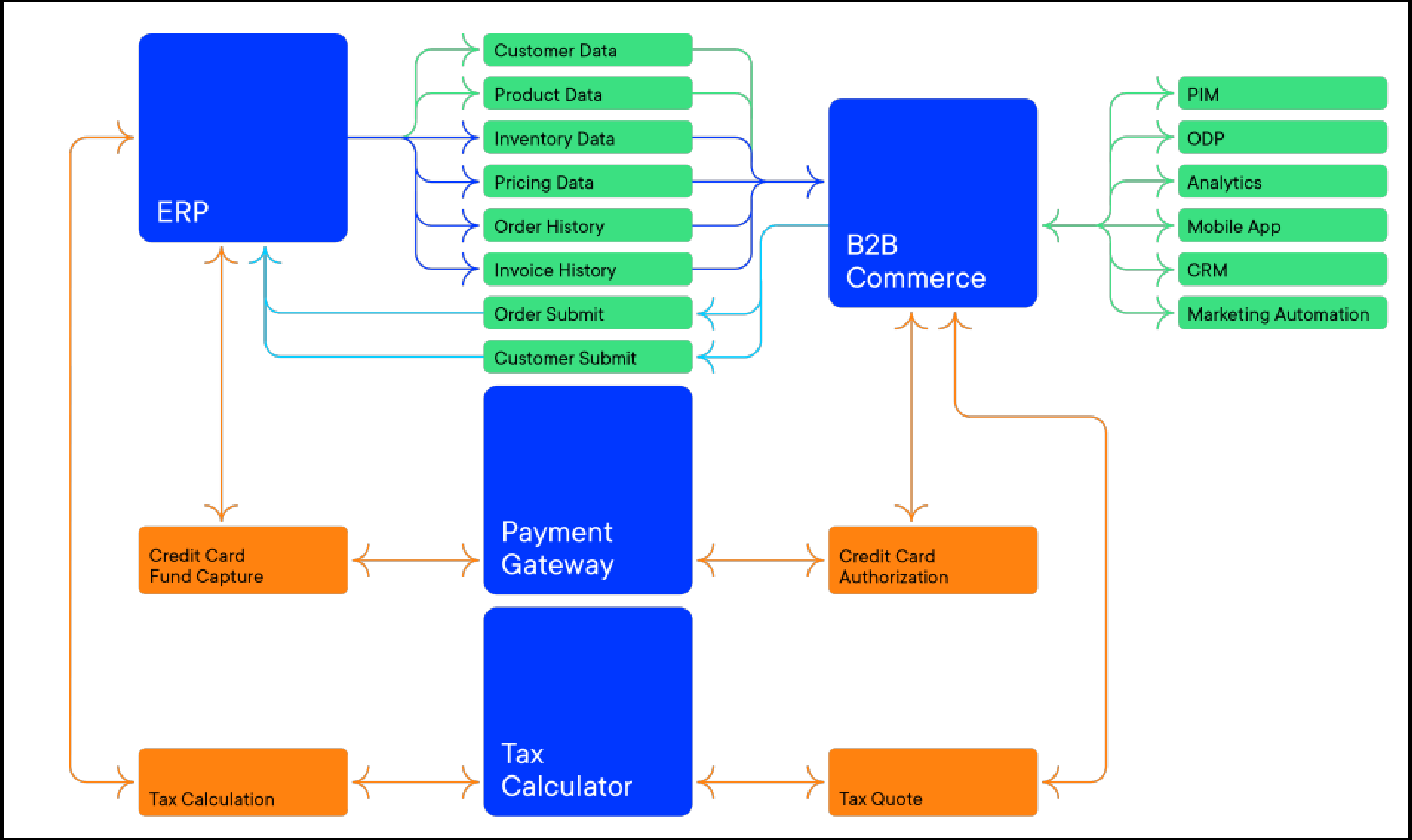
System Integration

- Optimizely B2B Commerce is shipped with an amazing integration layer out of the box. It simplifies connection with many external systems like APIs, ERPs, PIMs, CRMs, and even Flat Files
- The administration console can be used to create connections to external systems. Jobs can be configured to fetch data from an external system and map it back into the B2B Commerce model. All that without writing a line of code.
- The integration layer is extensible. You can develop your own processors and use them within customs jobs.
- The Windows Integration Service (WIS) can be installed in the customer infrastructure when external systems are not reachable from the outside. An internal WIS is also provided as part of the SaaS platform for cloud connection to external systems .
- You can set recurring jobs and link jobs together to schedule and coordinate data synchronization tasks.
- Notification mechanism to inform you if a specific job has failed
- Pre-coded jobs are provided by the platform for fetching real-time prices/inventory and submitting orders

System Integration Architecture



System Integration Typical Architecture



Existing Connectors

/ Optimizely B2B Commerce solution comes with standard connectors, reducing the amount of work required to connect to external systems .

/ There is no additional cost for adding one of those to your project.

/ It is possible to customize those connectors to match the customer's needs. Note that it is not possible to add a new Payment Gateway connector and that you must ask Optimizely to create it for you because of their PCI-DSS compliance.

/ Those standard connectors are very good starting points for creating your own connectors

/ Project in 2023 to add the possibility for partners to create their own connectors and publish them in the Optimizely Tech Marketplace

ERP

A+
Acumatica
Infor CSD
Infor FACTS
Infor SX.e
IFS OnPremise
IFS Cloud
Prophet 21

Payment

Adyen
APS
Authorize.Net
Bambora
CardConnect
CenPOS
CyberSource
Evo
Moneris
Orbital
Paypal
Payeezy
Stripe
And many more…

Punchout

cXML

Tax

Avalara
CyberSource
Vertex

Search

Elasticsearch
Hawksearch

Section 08

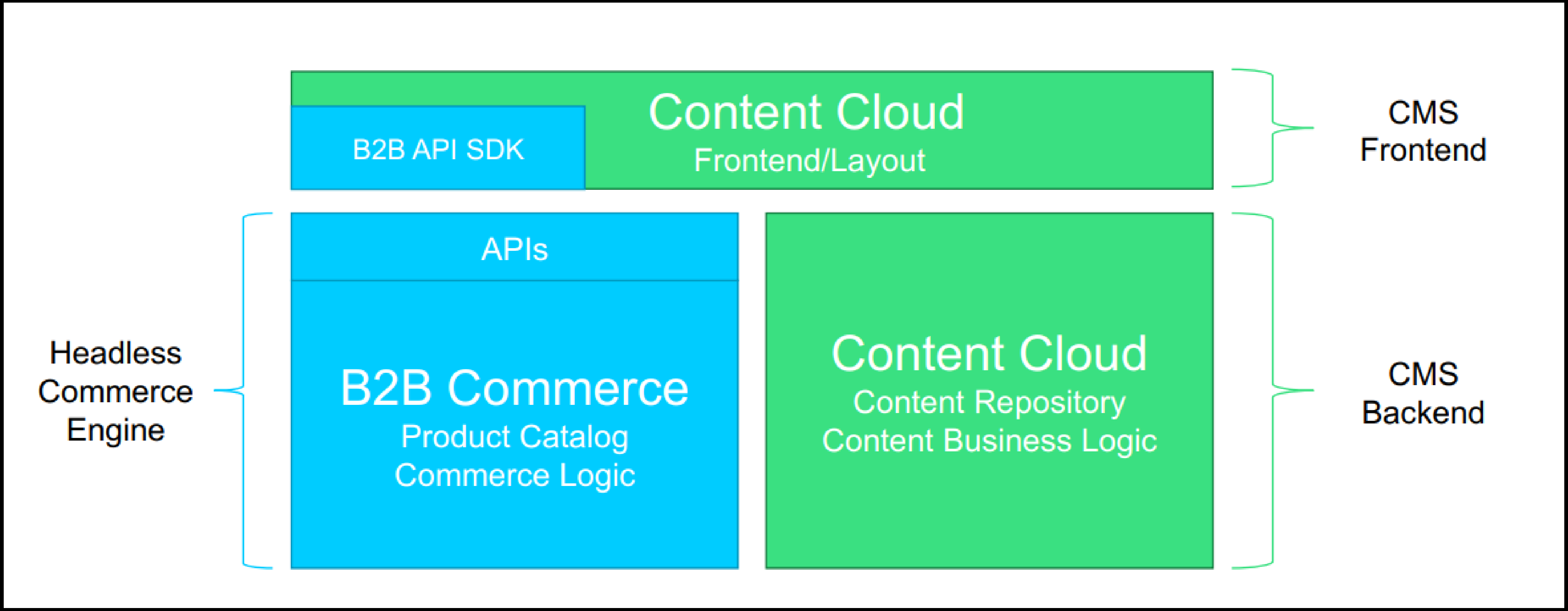
Add-Ons

Add-Ons

Name	Availability
B2B Analytics	Available
Mobile And VMI	Available, VMI in beta
PIM	Available
Content Cloud Integration	Released in Q4 2022
Product Recommendations	Q4 2022
Experimentation	2023

Content Cloud Integration

B2B API SDK connects B2B Commerce into Content Cloud – all B2B Commerce functionality can be used within the Content Cloud App.



Section 09

Q&A

Thank You

Alain Marsolais, Optimizely Practice Director
alain.marsolais@valtech.com

valtech_