**Building Business Logic for Industries Digital Commerce**

So how do we make our product model feasible with Industries Digital Commerce? Fortunately, some of that heavy lifting is taken care of with the **Digital Commerce SDK**.  
  
Consumers demand that communications, media, and energy and utilities companies provide a highly-differentiated user experience, and to do that, front-end developers require complete control of all elements of the user experience. Industries deliver this level of control by providing a layer of business logic as part of the Digital Commerce solution in theDigital Commerce SDK. With this software development kit (SDK), you can assemble custom applications utilizing a library of functions, improving solution time to market while still providing a highly-customized user experience.



**What is the Industries Digital Commerce SDK?**

The Industries Digital Commerce SDK is a pure JavaScript library that abstracts and simplifies the use of Digital Commerce APIs. It is one of the many SDK libraries that Industries offers as part of its solution.

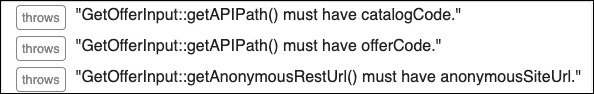
In some cases, the SDK invokes **more than one API** to complete a request.

The SDK can be shared by different applications user interfaces and like any other Industries SDK, it can be extended and overridden for custom implementations.

**Why use the Industries Digital Commerce SDK?**

The SDK provides for error checking and parameter validation before data is passed to the Digital Commerce APIs. **This** **ensures that Industries data structure rules are enforced and that data is not written incorrectly to the Cart** **or to the Shared Catalog**.

Here is an example of the errors that can be thrown if the getOffer method does not receive all of the parameters needed for the API it calls:



Additionally, the SDK can reduce UI development time by providing consolidated common application and business logic methods.

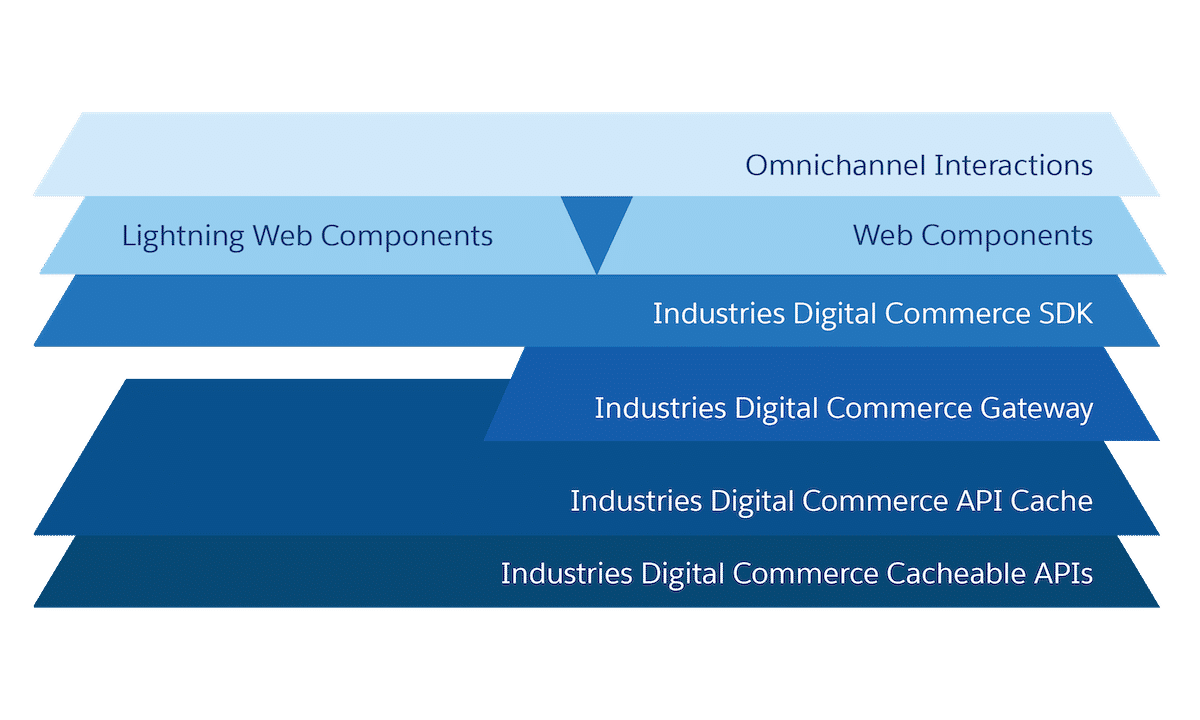
**When should the Industries Digital Commerce SDK be used?**

Early and often!

In order to build robust and successful digital commerce applications, Industries recommends using the Digital Commerce SDK in *almost all cases*, rather than calling the Digital Commerce APIs directly. This will allow you to take advantage of the many benefits of the Digital Commerce SDK, and ensure your application is extensible and upgradeable.

**How do you use the Industries Digital Commerce SDK?**

The following lessons in this course describe exactly how to use the SDK. But in big-picture terms, you call the Digital Commerce SDK from the user interface using either **Digital Commerce Lightning Web Components** (when developing a Salesforce on-platform application) or using**Digital Commerce Web Components** (when developing an off-platform application).



**The Industries Digital Commerce User Experience**

When you read through the SDK documentation and imagine how you might implement it, you should think about how users will be interacting with your site. Generally speaking, there are four phases in any commerce experience: **Browse**, **Configure**, **Cart**, and **Checkout**.

Below, we'll cover what users typically expect in a commerce experience and how the Industries Digital Commerce solution aligns with it.

**Browse Phase**First, users browse the available offers in the catalog. This is the **Browse** phase.

The Industries Digital Commerce solution supports this phase by giving you methods and functions to retrieve catalog offers and individual offer details.

**Configure Phase**Second, users may select an offer and configure it, which may change the pricing and availability, we'll call this the **Configure** phase.  
  
The Industries Digital Commerce SDK supports this phase by giving you methods and functions to get updated pricing and to perform offer validation. In this phase, the virtual "basket" is not shown to the user.

**Cart Phase**Third, users may add the configured offer to their cart (a container of saved line items), and we'll call this the **Cart** phase. This phase may be skipped if the user directly orders the offer ("buy now") from the offer page.

**Checkout Phase**Finally, when the user is ready to review their order, enter in or retrieve shipping, billing, and payment information, and finally submit their cart so that it may become an order, this is the **Checkout** phase.

The Industries Digital Commerce SDK supports this phase by using the create cart API to create an order from the user's temporary cart.

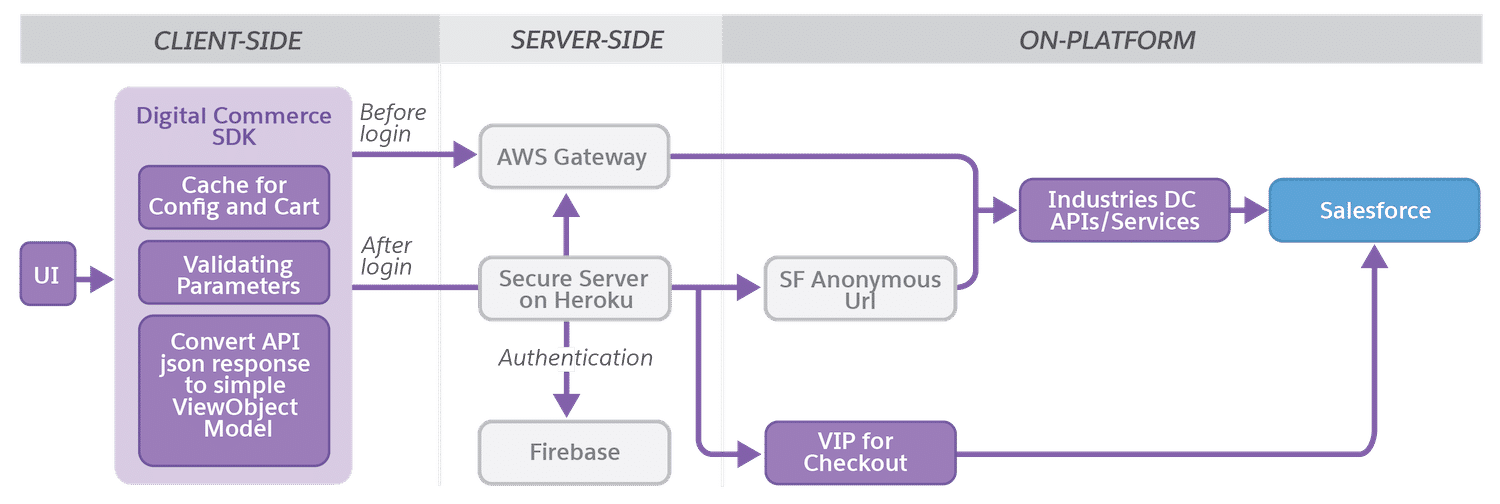
Now that you have a general understanding of how the Industries Digital Commerce SDK works, you will have the option of using the SDK purely on the client-side, or on the server-side.

**Comparing Client-Side and Server-Side SDK Implementation Methods**

The Digital Commerce SDKs can be implemented in two ways, one of which utilizes a server-side SDK. There are three Digital Commerce SDKs. The **Digital Commerce SDK**, the **Proxy SDK**, and the **ServerSDK**.  
  
The architecture can be difficult to keep track of so let's start out by reviewing a diagram that connects the components, services, and platforms.

**The Client-Side Industries Digital Commerce SDK Method**

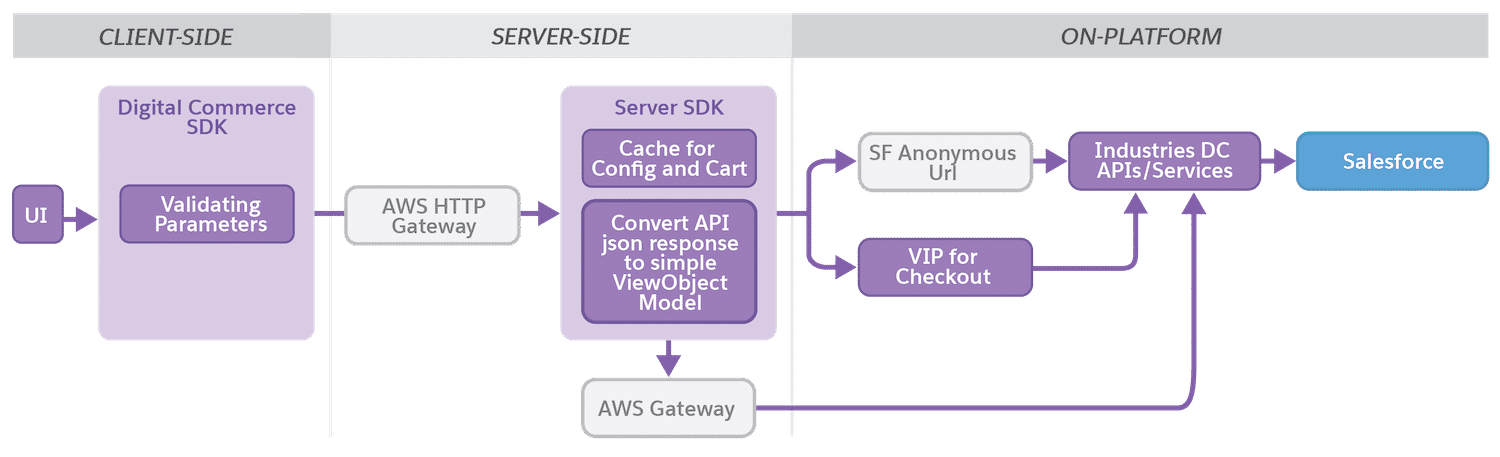
This is an architecture illustration for the Digital Commerce UI with a logged-in users scenario:



This method uses a **purely client-side SDK (Digital Commerce SDK)**, redirecting calls to an AWS Gateway, secure server (e.g., Heroku) with an OmniStudio Integration Procedure, and/or SF anonymous URL. Ultimately all of those methods will connect to the cacheable APIs. The main distinction of this method is that **all SDK calls are made from the client**.

**The Server-Side Industries Digital Commerce SDK Method**

This is an architecture illustration for the Digital Commerce UI with a logged-in users scenario:



The other method uses a **client-side proxy SDK**, redirecting all calls to a **server-side SDK**running on a Node.js server. In this case, you are utilizing the **Digital Commerce** **Proxy SDK**on the client-side and the **ServerSDK**on the server-side.  
  
Alternatively, you can opt to use the Digital Commerce SDK from the first method, instead of the Proxy SDK. In this case, all calls will be made from the client until the user signs in, after which all calls will be made from the ServerSDK.

So why use this new method? It has a number of advantages. It supports third-party authentication systems (e.g., Firebase), it does not expose Salesforce account IDs or org details/tokens to the client, provides secure endpoint URLs, supports the *isloggedin* cacheable API parameter, and requires less custom application logic for iOS/Android native clients.

**Choosing an SDK Implementation Method**

Ultimately there are some considerations to be made when deciding between the two SDK implementation methods. We recommend you read more about the pros and cons on the ServerSDK Overview documentation page.

**ServerSDK Installation**

The ServerSDK is a reference implementation that serves as a middle layer between the user interface and the Digital Commerce APIs. ServerSDK runs SDK methods on any Node.js supported server and provides REST APIs that you can interact with.

The first thing you need to do is obtain the serverSDK reference app zip file from the Salesforce Industries Process Library. Please follow all of the **ServerSDK Installation** steps listed on the Server SDK Installation documentation page.

**Server-Side Deployment**

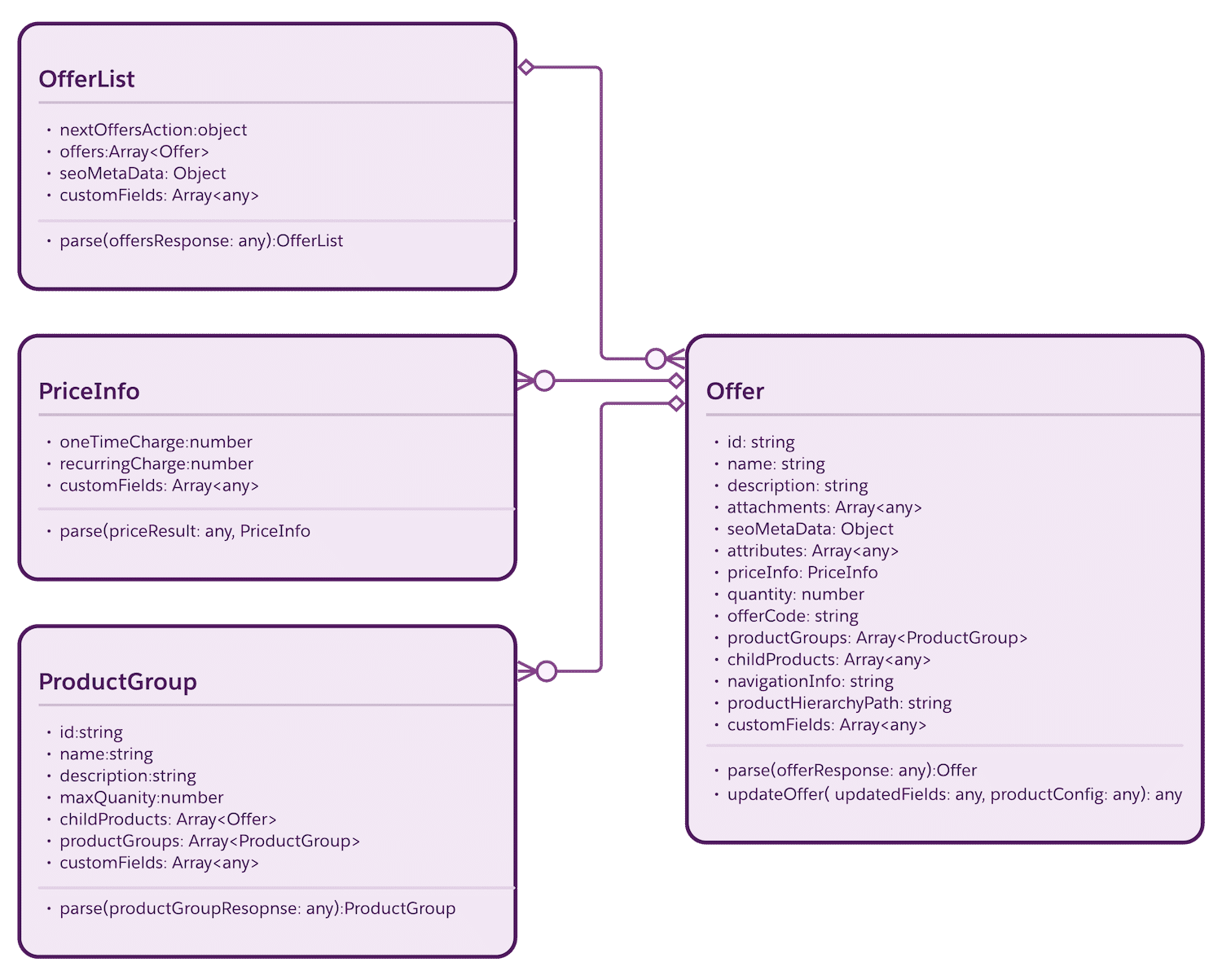
On the documentation page, you can also find steps for deploying the app locally or in a Heroku environment. In the videos below, we will demonstrate how to run the ServerSDK on Heroku.

**The Industries Digital Commerce SDK Object Model**

One of the big benefits of the Industries Digital Commerce SDK is its object model. The object model is entirely separate from sObjects or the Industries Communications platform data model. The SDK's object model describes how data will be formatted in the SDK response. The object model's logical representation also carries over to the web components in the user interface.  
  
Understanding the existing object model is key if you plan on extending the SDK.

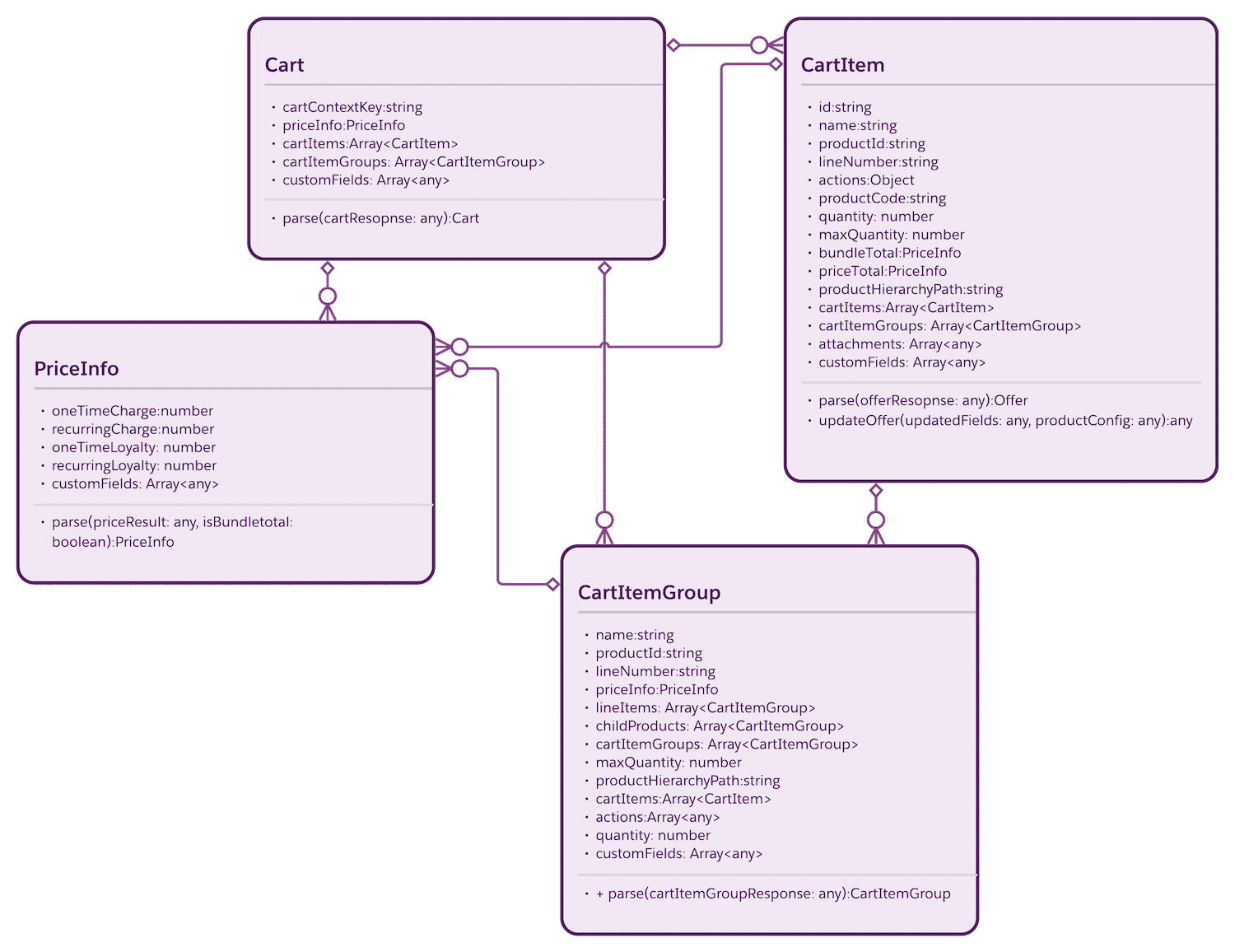
**The OfferList Object**

The OfferList object includes an array of Offer objects, which include the PriceInfo and ProductGroup objects. The OfferList object serves as the parent with its offers array. PriceInfo and ProductGroup objects contain data specific to the offer you have requested.



The Cart Object

The Cart object serves as the parent of an array that includes the CartItem, PriceInfo, and CartItemGroup objects. PriceInfo and CartItemGroup contain data specific to the CartItem you have called for.



**Defining the Industries Digital Commerce SDK**

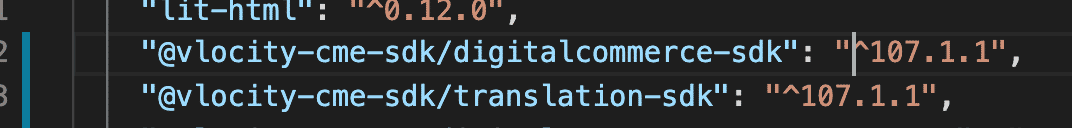
The Digital Commerce SDK was designed using a Singleton pattern to allow the instantiation of its class to one "single" instance.

Using the Singleton pattern provides the following benefits:

* Shared single instance
* Minimized memory usage
* Easier data sharing
* Browser caching as a singleton object

**SDK Versioning and the npm Registry**

Industries' SDKs are versioned like the managed package in your org. They are versioned and distributed using a protected npm registry.  In addition, they follow [npm's semantic versioning specification](https://docs.npmjs.com/about-semantic-versioning" \t "_blank). Non-breaking changes will be automatically distributed to your development environment.



**Getting an Industries Digital Commerce SDK Sample App**

The Industries Digital Commerce sample app is available to Industries customers and partners as a working reference to learn more about a possible Digital Commerce solution. It is not intended to be deployed to a production environment.

To access the sample app, visit the Salesforce Industries Process Library and select Digital Commerce under Filter by Category. It is listed as "Explore & Checkout for Off-Platform Usage".

**Installing the Industries Digital Commerce SDK**

To install the Digital Commerce SDK, you will need an authorization token, which you can get by creating a case with [Salesforce Industries Support](https://help.salesforce.com/). Once you have the token, here are the steps to follow to import the SDK's node\_module dependencies:

1. Open **Visual Studio Code**.  
  
2. Click **File** at the top-left corner of the screen.  
  
3. Click **New File**.  
  
4. **Copy and paste** the following into the window that just opened: (**replace "YOUR\_EMAIL\_ID"**with your email and **"YOUR\_AUTH\_TOKEN"** with the token Salesforce Industries support provided you):

email=YOUR\_EMAIL\_ID

always-auth=true

auth=YOUR\_AUTH\_TOKEN

registry=https://repo.vlocity.com/repository/npm-public/

5. Press **CTRL+S**or **CMD+S**on your keyboard to save the file.  
  
6. Name the file **.npmrc**.  
7. It should prompt a message, click **Use "."**.  
  
8..**Add the digitalcommerce-sdk dependency** into your package.json file (see video and use "109.0.1" as the version).  
  
9. Press **CTRL+S** or **CMD+S** on your keyboard to save the file.  
  
10.**Open the Terminal command line** within Visual Studio Code, type **npm install**,and then **press enter**.

Easy, right? Now in the**node\_modules** folder, you will see a new folder called **@vlocity-cme-sdk** that contains a folder called **digitalcommerce-sdk**. This is your ticket to Digital Commerce fun and games.

You can also move this folder inside the public/static resource of your application.

**Instantiating the Digital Commerce SDK**

To instantiate the SDK, you create a single instance of the DigitalCommerceSDKConfig object. The object consists of a UserContext and a DataSourceService. In an end-to-end digital commerce process flow, you may have one instance for anonymous users and, later in the process flow, you will instantiate another for authenticated users. To create an instance, you use one of the following methods, depending on whether you are developing the instance for anonymous users or authenticated users.

[createConfigForAnonymousUser](https://s3.amazonaws.com/vlocity/vlocitysdk/v104/classes/_digitalcommerce_digitalcommerce_sdk_.digitalcommerce.html#createconfigforanonymoususer)

[createConfigForLoginUser](https://s3.amazonaws.com/vlocity/vlocitysdk/v104/classes/_digitalcommerce_digitalcommerce_sdk_.digitalcommerce.html#createconfigforloginuser)

Defining the SDK Methods

Industries provide several methods to support the customer's journey through browsing, configuring, adding a product or promotion to the cart, and submitting the order. We will explore these methods as they pertain to the four phases of the Digital Commerce User Experience Model.

Browse Phase Methods

During the **Browse**phase, the user is browsing a catalog of products or promotions.

This can be done by anonymous or authenticated users.

During the Browse phase, you can use the following methods to navigate the catalog and review offers.

* [getOffers](https://s3.amazonaws.com/vlocity/vlocitysdk/cmt/109.0/classes/_digitalcommerce_digitalcommerce_sdk_.digitalcommerce.html#getoffers) - Returns a list of offers (products and/or promotions) from a catalog
* [getOffer](https://s3.amazonaws.com/vlocity/vlocitysdk/cmt/109.0/classes/_digitalcommerce_digitalcommerce_sdk_.digitalcommerce.html#getoffer) - Returns an offer and its details.

Configure Phase Methods

During the **Configure**phase, the user is configuring an offer by changing quantities, adding or deleting child products in the bundle, and configuring attribute values. Once the offer is configured, the validateOffer method is used to check for configuration errors.

This can be done by anonymous or authenticated users.

During this phase, use the getSelectedOffer method to return the offers that were viewed during the Browse phase from the cache. This is useful when the user is navigating through multiple pages. Note that no API calls are made in these methods.  
  
Use the validateOffer method to check the offer's configuration.

* [getSelectedOffer](https://s3.amazonaws.com/vlocity/vlocitysdk/cmt/109.0/classes/_digitalcommerce_digitalcommerce_sdk_.digitalcommerce.html#getselectedoffer) - Returns the offer for the given offer code that is being configured.
* [validateOffer](https://s3.amazonaws.com/vlocity/vlocitysdk/cmt/109.0/classes/_digitalcommerce_digitalcommerce_sdk_.digitalcommerce.html#validateoffer) - Validates the configuration of an offer; invokes ConfigureOfferDetails API to validate the offer and update pricing.

Cart Phase Methods

During the **Cart**phase, the user is adding, updating or deleting line items in the cart.

During this phase, use the following methods to manage the items in the cart. Once the cart is ready, use the checkoutCart method, which submits an order.

* [addToCart](https://s3.amazonaws.com/vlocity/vlocitysdk/cmt/109.0/classes/_digitalcommerce_digitalcommerce_sdk_.digitalcommerce.html#addtocart) - Adds or removes one or more offers into a cart.
* [updateItemsInCart](https://s3.amazonaws.com/vlocity/vlocitysdk/cmt/109.0/classes/_digitalcommerce_digitalcommerce_sdk_.digitalcommerce.html#updateitemsincart) - Replaces the item in cart with the provided offer configurations.
* [checkoutCart](https://s3.amazonaws.com/vlocity/vlocitysdk/cmt/109.0/classes/_digitalcommerce_digitalcommerce_sdk_.digitalcommerce.html#checkoutcart) - Creates an order with the items the user has configured in the cart.

Checkout Phase Methods

During the **Checkout**phase, the user is ready to checkout and submit the order.

Prior to submitting an order, the user must authenticate.  
  
Using the checkoutCart method to create an order for the items the user has configured:

When the Industries Digital Commerce SDK Is Not Enough

When developing your application, you may find that you have requirements that do not precisely map to Industries SDK's existing methods. Or you may want to modify how the SDK's response is formatted to meet the needs of your business application. In rare cases, you may need to override the existing methods.

**This is absolutely normal.** Your business has unique requirements, which drive your competitive advantage in the marketplace. But you're the one who is responsible for making this happen. How can you adapt Industries SDK to your company's unique requirements?

Lucky for you, Industries has designed the Digital Commerce SDK to allow you to extend its functions, transform its response, or even override a particular method.

Extending the Industries Digital Commerce SDK

You can add new methods to the Digital Commerce SDK using the [extend](https://s3.amazonaws.com/vlocity/vlocitysdk/cme/sdk/104.1.0/classes/_digitalcommerce_digitalcommerce_sdk_.digitalcommerce.html#extend) method.

Transforming the Industries Digital Commerce SDK Response

Sometimes you may find that the Digital Commerce SDK is returning WAY too much information for your liking. (Remember the SDK's object model that we looked at earlier? That is what will be returned by default.)

You can trim the SDK's response using a customObjectMap function as shown in the following video.    
  
Trimming the SDK response using the customObjectMap function:

Overriding the Industries Digital Commerce SDK

In rare cases, you may need to override an existing method using the [override](https://s3.amazonaws.com/vlocity/vlocitysdk/cme/sdk/104.1.0/classes/_digitalcommerce_digitalcommerce_sdk_.digitalcommerce.html#override) method. You should use this method sparingly because it may have unintended consequences, affect upgrades, and make troubleshooting more difficult.

When Things Go Wrong

Debugging the SDK is best done by checking that the SDK is fully installed, instantiated, and passing properties correctly. Start with the checklist below, and then use the following video series to help with troubleshooting.

1. Make sure the SDK is loaded properly.
2. Add a debugger to check the instance.
3. The context object can indicate whether the SDK is properly loaded or not.
4. Make sure the SDK properties are passed correctly.
5. If the status code is a 200, the API call was successful.
6. Finally, check the error.