Heroku Solution Kit for Commerce

Clienteling App

The purpose of this document is to showcase best practices for building a specific use-case. This kit is for developing a clienteling app leveraging Salesforce B2C Commerce and Heroku. Examples of this include API-First or API-Driven experiences driven by B2C Commerce REST APIs (OCAPI) but hosted and developed leveraging Heroku.

Version 1 - May 2019

Future updates to come
## CONTENTS

### WHAT IS A SOLUTION KIT?  
4

### BUSINESS VALUE OVERVIEW  
5

### USE CASE OVERVIEW  
6
- Clienteling App for In-Store  
  6
- Introducing the NTO Team: Developer and Business Leaders  
  7
- Collect Requirements  
  7
  - Store Associate-Facing Requirements  
  8
  - Application Requirements  
  9

### UNDERSTANDING HEROKU PREREQUISITES  
10
- Sign Up for a Free Heroku Account  
  10
- Install the Heroku CLI to Provision Development Accelerators  
  10
- Get to Know the Anatomy of a Heroku Application  
  11

#### Establishing the Application Footprint  
12
- Select a Buildpack  
  12
- Supported Buildpacks  
  13
- Provision and Evaluate Heroku Add-Ons  
  14
- Heroku Add-Ons for the NTO Mobile Application  
  14
- Provision and Evaluate Heroku Buttons  
  15
- Heroku Buttons for the NTO Storefront Application  
  15

#### Putting it All Together  
16
- Understanding the Heroku Application Development Model  
  16

### INTEGRATING WITH B2C COMMERCE  
17

#### Develop Applications With Commerce Cloud's REST APIs  
17
- Identify OCAPI Endpoints  
  18
- Authenticate a Customer  
  19
- Create a Customer Session  
  19
- Retrieve Storefront Products  
  19
- Create or Retrieve a Customer’s Basket  
  20
- Add a Product to or Remove it from a Customer’s Basket  
  20

#### OCAPI Performance Best Practices  
20
WHAT IS A SOLUTION KIT?

Salesforce solution kits are your go-to when you want to extend Commerce Cloud B2C Commerce beyond the traditional storefront. Heroku lets you use the programming language of your choice with the Commerce Cloud API, which empowers developers to focus on innovation and business value instead of plumbing and technical puzzles. This makes it easy to deliver personalized shopping experiences that increase customer conversion, engagement, and loyalty.

This solution kit provides:

- An overview of the business value
- An example use case
- How to get started with Heroku on your own implementation
- Tips on integrating with B2C Commerce

The workflows and best practices you'll discover in this kit empower IT owners and developers with a foundation for custom commerce experiences that'll connect your brand with today's shoppers.

How Do I Learn More?

- Interested in Heroku? - Sign up for a free Heroku account. Learn more about Heroku pricing and availability.
  - Trailhead: Develop Apps in Heroku

- Interested in Commerce Cloud? - Check out these Trailhead Trails to learn more about B2C Commerce. Contact your Salesforce Account Manager for a demo and more information.
  - Trailhead: Developing on Salesforce B2C Commerce
  - Trailhead: Commerce Cloud Basics
  - Trailhead: Commerce Cloud Features
BUSINESS VALUE OVERVIEW

The In-Person Shopping Experience is Here to Stay

Successful brands create rich shopping experiences that ensure they stand out from the competition. In fact, some experts have predicted that the customer experience will soon overtake price and product selection as the key differentiator of brands. As a result, brands are looking to orchestrate a unified customer experience across all touchpoints, including online, mobile, and in stores. You might think that the explosive growth of mobile commerce has rendered physical stores unimportant, but nothing could be further from the truth. Even with the convenience and power of online commerce, nearly half of all consumers still prefer to shop in person.

Unify the Customer Experience via Your Store Associates with Heroku and the Salesforce Commerce Cloud Platform

Consumers expect a consistent branded experience across all sales channels, and that unified experience strengthens their engagement with a brand. Oftentimes, an online storefront session leads to a brick & mortar store visit to check out a product in person, or a shopper already in the store uses the mobile app to further research a product they've found. Providing your in-store personnel with a branded clienteling application helps them maintain consistent interactions with customers.

The Commerce Cloud B2C platform enables powerful customer experiences with its APIs, and Heroku lets you access that power any way you choose.

- **Commerce Cloud for API-driven commerce** - Deliver amazing experiences to every shopper with Commerce APIs for merchandising, search, data, AI, and more.

- **Heroku platform for rapid, scalable app development** - Empower your developers with embedded data services, incredible scalability, and a powerful runtime accessible with any programming language, from the most app-centric platform-as-a-service on the market.

- **Heroku and open-source ecosystem** - Enhance the capabilities of your apps with Heroku add-ons, open-source components, and third-party services.

- **Partner ecosystem** - Engage with ISV and SI partners who specialize in creating great mobile experiences for shoppers and store associates.
USE CASE OVERVIEW

Clienteling App for In-Store

Our merchant, Northern Trail Outfitters (NTO), is an outerwear, apparel, and gear retailer for outdoor recreation and fitness. The typical NTO customer is an avid outdoors enthusiast willing to spend more for quality products, but not averse to great deals. The executive team understands that to best engage with their sophisticated shoppers, they should equip their store associates with a tool that empowers them to provide an exceptional and personalized in-store experience.

NTO decides to create their own version of a luxury clienteling app, which will enable store associates to manage personalized relationships with customers by tracking their preferences, product interests, special dates, and other relevant information, facilitating customer communications, and integrating with catalog and inventory data.

For example, a store associate logs into the clienteling app and receives a notification that the latest backpacking gear has arrived in stock. He then runs a report that identifies local customers who have expressed an interest in that type of product. Once they've been identified, he fires off text messages and emails informing them of the new arrivals.

When one of these customers comes into the store and mentions the communication, the associate brings up that customer's account on the app and reviews his preferences and purchase history. He's then able to discuss the new products in the context of that customer's personal interests, including being able to quickly determine whether his preferred sizes, colors, and styles are in stock. In addition, he can mention a few other products that the Einstein personalization engine has recommended specifically for that customer! If the shopper is ready to make a purchase, he can initiate the sale right in the app, and even order an out-of-stock style from another location.
Introducing the NTO Team: Developer and Business Leaders

Understanding the Developer Persona

This team runs lean and mean, maintaining the B2C Commerce storefront experience running on Commerce Cloud while also managing integrations to external systems. Made up of a combination of full-stack, front-end, and back-end engineers that have varying levels of experience deploying custom web applications, the team wants to make this project a reality. They also know that they have some real constraints to work through:

- **Capacity**: The team has existing responsibilities, and won't be able to dedicate 100% of their resources to this application.

- **Timeline**: The business would like this application developed and deployed in time for camping season.

Understanding the Business Leader Persona

The business leaders are responsible for creating innovative experiences that are reflected in their bottom line and aggressive growth projections. When it comes to making decisions, they have some key factors in mind:

- **Budget**: As with any lean organization, overhead and operational costs must be kept low and predictable.

- **Scale**: As the NTO brand continues to grow and expand, they need to invest in technology that can scale at the pace needed to meet growing customer demand and stay ahead of the competition.

Undaunted by these challenges, the team presses forward with identifying and addressing requirements. Requirements that spell out customer-facing and internal application goals will give them what they need to think about the implementation while respecting their organizational constraints.

Collect Requirements

Requirements are the foundation of successful application development. While NTO application development team members are firm believers in user stories, they also realize they first need to draft a collection of user-facing and internal application requirements to frame their effort.
Store Associate-Facing Requirements

These requirements explain the desired customer experience and clienteling app capabilities.

- **Ordering**
  - Shop for a customer with or without logging in to their account.
  - Locate product inventory and place an order for the customer to ship to home, to their store, or to another store.
  - Support multi-shipment orders.

- **Customer Profiles**
  - Share the same customer profile data used in storefront and mobile channels.
  - Search by name or email address.
  - Access and modify a customer profile: review order history, maintain contact and address information, reset password, etc.
  - Retrieve promotions and pricing available to a particular shopper.
  - Identify any customer groups a customer belongs to.
  - Maintain a unified basket: Retrieve items from and push items to a customer's web basket.

- **Merchandising**
  - Connect to the existing Commerce Cloud storefront for pricing, promotions, customers, inventory, etc.
  - Search for products by ID or name, and browse products by category.
  - Surface Einstein personalized recommendations.

- **Secure Payment-Present Transactions**
  - Accept mobile wallets (Paypal, Apple Pay, Samsung Pay, etc.)

- **Special Point-of-Sale Features**
○ Enable permission-managed price overrides.
○ Offer sales dashboard and reporting.
○ Manage permission groups to control access to elements of the application.
○ Include store associate ID in order exports.

Application Requirements

These requirements describe features and capabilities that must be satisfied by the clienteling application to ensure that the solution is secure, maintainable, and scalable to customers. They describe application development goals.

● Integrate with the B2C Commerce storefront via OCAPI for product data.
● Scale OCAPI SHOP API request performance.
● Secure credentials for all integrated systems.
● Monitor the application's uptime, availability, and engagement.
● Deliver a high-performance user experience with zero latency.
● Capture and consolidate all log files for easy inspection and triaging.
● Simplify application deployment and versioning.
● Ensure predictable cost management and forecasting.

With these requirements in place, the team can begin to think about ways to accelerate their application development.

As Commerce Cloud developers with customer application experience, the NTO team has an idea of the effort necessary to stand up this application from scratch. Unfortunately, they don't have the resources to do it — nor do they want to make themselves responsible for maintaining, scaling, and tuning a stand-alone application through to go-live. Fortunately for them, though, one of their team members suggests that they look into leveraging Heroku.
UNDERSTANDING HEROKU PREREQUISITES

Here, we show you how to get started with Heroku’s amazing developer experience.

Sign Up for a Free Heroku Account

Heroku allows developers to collaboratively create apps, connect to databases, and expand with add-on services. Commerce Cloud developers are invited to sign up for free and experience Heroku for themselves. The majority of Heroku build-packs, add-ons, and buttons are supported in the free version. Some of the features of Heroku's free tier include:

- Run apps for free using your monthly pool of free dyno hours
- Unverified accounts receive a pool of 550 free dyno hours
- Verified accounts receive an additional 450 free dyno hours
- Dyno hours can be shared across any of your free apps
- 1 web dyno/1 worker dyno/1 one-off dyno maximum per app
- 512 MB RAM per dyno
- Free apps sleep automatically after 30 mins of inactivity to conserve dyno hours
- Free apps wake automatically when a web request is received
- Access the Heroku Dashboard and Heroku CLI for app management
- Verified accounts can have custom domains for every free app
- Up to 5 (unverified accounts) or 100 (verified accounts) free apps

Install the Heroku CLI to Provision Development Accelerators

The Heroku Command Line Interface (CLI) makes it easy to create and manage your Heroku apps directly from the terminal. It’s an essential part of using Heroku. Using the CLI, you can authenticate, manage your application and its resources, provision
add-ons, and scale your application. Developers can install platform-specific instances of the CLI via the links below:

- MacOS Installer
  ```sh
brew tap heroku/brew && brew install heroku
  ```

- Windows 64-bit Installer
- Windows 32-bit Installer
- Ubuntu 16+
  ```sh
  sudo snap install --classic heroku
  ```

- NPM Installer
  ```sh
  npm install -g heroku
  ```

Armed with a Heroku account and CLI, your team is ready to begin experiencing Heroku for yourselves. A great place to begin is this Getting Started video from TrailheadX on YouTube As well as the Heroku's Getting Started guide for a deeper dive into the platform.

Get to Know the Anatomy of a Heroku Application

Just as Heroku makes it easy for developers to get started by letting them use the programming language of their choice, the Heroku Dev Center empowers and educates developers on how to maximize their platform's capabilities and scalability. Developers looking for a deeper dive into Heroku's architecture and principles of success should review the following articles:

- How Heroku Works provides a high-level technical description of the platform. It ties together many of the concepts encountered while writing, configuring, deploying and running applications on the Heroku platform.

- Architecting Applications on Heroku explains a common set of architectural and development best practices. Many of these concepts have their origins in the Twelve Factor methodology and are detailed here in the context of building your app for Heroku.

- Development and Configuration Principles includes several principles of application development and configuration central to the Heroku development experience.

- Runtime Principles explains the core runtime principles that applications must follow to ensure that they can be properly managed and scale effectively.
• **Erosion Resistance** describes how Heroku applications are resistant to the wear, tear, scalability struggles, and maintenance headaches of other platforms.

• **Principles of Management and Visibility** explains how Heroku handles many of the execution and orchestration concerns inherent in running applications. It covers many useful auxiliary functions as well, including running one-off tasks and gaining visibility into your application.

**Establishing the Application Footprint**

**Identify Buildpacks, Add-Ons, and Buttons**

Buildpacks, add-ons, and Buttons provide scalable features to help you build your Heroku application. The first step in Heroku development is to decide which ones you'll use.

- **Buildpack.** Buildpacks correspond to programming languages. They take code written in a language and transform it into a packaged, compiled copy. This compiled copy can then be scaled via the Heroku platform through its CLI and Dashboard tooling.

- **Add-ons.** Add-ons are components that play a supporting role in your application. They handle specific application activities such as data storage, monitoring, analytics, and security, to name a few. Add-ons eliminate the need for you to create sub-systems from scratch to manage these activities.

- **Buttons.** Buttons provide an easy way to get an app up and running quickly on the Heroku platform. They are pointers to deployable and configurable source code repositories. Clicking a Heroku Button initiates deployment of the app, provides an option to configure the app, and delivers the running app on the web.

- You also have to consider how you'll use Heroku's features with OCAPI. It provides two REST APIs (SHOP and DATA) that can be used to enable the shopping experience, as well as an administrative data management experience via service interactions.

**Select a Buildpack**

Buildpacks are responsible for transforming deployed code into a **slug**, which can then be executed on a **dyno**. A buildpack is composed of a set of scripts. Depending on the programming language, the scripts retrieve dependencies, output generated assets or compiled code, and more.
It's important to select a buildpack that aligns with your team’s development preferences and capabilities.

**Supported Buildpacks**

The Heroku platform includes a collection of [officially supported buildpacks](https://elements.heroku.com), as well as [third-party supported buildpacks](https://elements.heroku.com). If Heroku’s official buildpacks don’t meet your requirements, hundreds of custom, third-party buildpacks are available in the [Elements marketplace](https://elements.heroku.com) or via the CLI.

Officially supported buildpacks include the following:

<table>
<thead>
<tr>
<th>Buildpack</th>
<th>Shorthand</th>
<th>Documentation</th>
<th>Runtime versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ruby</td>
<td>heroku/ruby</td>
<td><a href="https://elements.heroku.com">Documentation</a></td>
<td><a href="https://elements.heroku.com">Runtime versions</a></td>
</tr>
<tr>
<td>Node.js</td>
<td>heroku/nodejs</td>
<td><a href="https://elements.heroku.com">Documentation</a></td>
<td><a href="https://elements.heroku.com">Runtime versions</a></td>
</tr>
<tr>
<td>Clojure</td>
<td>heroku/clojure</td>
<td><a href="https://elements.heroku.com">Documentation</a></td>
<td><a href="https://elements.heroku.com">Runtime versions</a></td>
</tr>
<tr>
<td>Python</td>
<td>heroku/python</td>
<td><a href="https://elements.heroku.com">Documentation</a></td>
<td><a href="https://elements.heroku.com">Runtime versions</a></td>
</tr>
<tr>
<td>Java</td>
<td>heroku/java</td>
<td><a href="https://elements.heroku.com">Documentation</a></td>
<td><a href="https://elements.heroku.com">Runtime versions</a></td>
</tr>
<tr>
<td>Gradle</td>
<td>heroku/gradle</td>
<td><a href="https://elements.heroku.com">Documentation</a></td>
<td><a href="https://elements.heroku.com">Runtime versions</a></td>
</tr>
<tr>
<td>Grails 3.xx</td>
<td>heroku/gradle</td>
<td><a href="https://elements.heroku.com">Documentation</a></td>
<td><a href="https://elements.heroku.com">Runtime versions</a></td>
</tr>
<tr>
<td>Scala</td>
<td>heroku/scala</td>
<td><a href="https://elements.heroku.com">Documentation</a></td>
<td><a href="https://elements.heroku.com">Runtime versions</a></td>
</tr>
<tr>
<td>Play 2.xx</td>
<td>heroku/scala</td>
<td><a href="https://elements.heroku.com">Documentation</a></td>
<td><a href="https://elements.heroku.com">Runtime versions</a></td>
</tr>
<tr>
<td>PHP</td>
<td>heroku/php</td>
<td><a href="https://elements.heroku.com">Documentation</a></td>
<td><a href="https://elements.heroku.com">Runtime versions</a></td>
</tr>
<tr>
<td>Go</td>
<td>heroku/go</td>
<td><a href="https://elements.heroku.com">Documentation</a></td>
<td><a href="https://elements.heroku.com">Runtime versions</a></td>
</tr>
</tbody>
</table>

For the NTO storefront app project, the development team chose the [Node.js buildpack](https) — their developers have extensive javascript experience and work with it every day in maintaining and extending their Commerce Cloud storefront.

```
heroku buildpacks:set heroku/nodejs
Buildpack set. Next release on random-app-1234 will use heroku/nodejs.
Run `git push heroku master` to create a new release using this buildpack.
```
Provision and Evaluate Heroku Add-Ons

Add-ons are installed onto applications using the Heroku Dashboard or the CLI. Most add-ons offer multiple plans, with different features, capabilities, and prices. Add-on plans are priced by the month, and prorated to the second. The majority of add-ons available via the Elements Marketplace have free plans — enabling developers to try one before committing it to an application's architecture.

- Look for add-ons that can accelerate your particular application or sub-system development.
- Use a free plan to evaluate whether an add-on is a fit by observing it in action.
- If an add-on isn't right for your project, go back to its category and evaluate other options.

Heroku Add-Ons for the NTO Mobile Application

The NTO development team selected the following collection of add-ons to accelerate their application development. These add-ons were chosen because they addressed core application framework needs and met several application requirements.

- **Still Alive** for monitoring
  
  heroku addons:create stillalive:developer

- **LogDNA** for logging
  
  heroku addons:create logdna:quaco

- **Heroku Postgres** for data capture
  
  heroku addons:create heroku-postgresql:hobby-dev

- **Heroku Redis** for caching
  
  heroku addons:create heroku-redis:hobby-dev

- **RuntimeError** for error tracking
  
  heroku addons:create runtimeerror:standard

- **Ice** to securely encrypt application secrets
  
  heroku addons:create ice:test

- **User Agent Identifier** and **IP to Earth** for customer identification
Provision and Evaluate Heroku Buttons

A Heroku Button is simply a pointer to a source code repository with information about how to configure and deploy it. When you click a Heroku Button, the Heroku platform deploys the linked code.

Since Heroku Buttons contain source code, they are fantastic accelerators for developers. Buttons often represent sample applications, reference applications, or stand-alone solutions that can be leveraged simply by including the Button.

Heroku Buttons for the NTO Storefront Application

The NTO development team chose two buttons for their project. These buttons are being used to accelerate their development, as well as improve performance of the application.

- The Node.js Getting Started on Heroku Button provides an Express.js application shell that the NTO development team can use to leverage their JavaScript skills.

  $ git clone https://github.com/heroku/node-js-getting-started.git
  $ cd node-js-getting-started
  $ npm install
  $ npm start

- The Reverse Proxy on Heroku Button will accelerate the performance of Commerce Cloud’s Open Commerce APIs. The team learned via the Commerce Cloud Platform documentation that a reverse proxy can be used to improve OCAPI performance for Shop API requests.
Putting it All Together

Understanding the Heroku Application Development Model

The Heroku model simplifies development and deployment of custom applications, as well as customer-facing access to them. Developers can build applications using languages they love, extend them using a combination of add-ons and Buttons, and scale them using Heroku's management capabilities.
INTEGRATING WITH B2C COMMERCE

Develop Applications With Commerce Cloud's REST APIs

Just as Heroku enables developers to quickly stand up, harden, and deploy custom applications, Commerce Cloud supports those applications with its collection of easy-to-leverage REST endpoints:

- The OCAPI SHOP API exposes the storefront functionality required to enable customer-facing shopping experiences via custom applications. Its functions include:
  - Authenticate customers
  - Perform API driven product searches
  - Provide product details (including variation availability)
  - Enable add to cart and checkout behaviors

- The OCAPI DATA API exposes management and integration capabilities for the data objects used by the Commerce Cloud platform. Its functions include:
  - Create and manage data objects
  - Manipulate the storefront configuration
  - Remotely execute jobs and import processes
  - Implement continuous integration
Explore The Commerce Cloud Open Commerce API Explorer

The **OCAPI API Explorer** lets you explore OCAPI's endpoints and practice calling them.

**Shop API**

**customers**
- `/customers`
  - `GET` `/customers`
  - `GET` `/customers/auth`
  - `GET` `/customers/password_reset`
  - `GET` `/customers/(customer_id)`

**baskets**
- `/baskets`
  - `POST` `/baskets`
  - `GET` `/baskets/(basket_id)`

**categories**
- `/categories`
  - `GET` `/categories` (with parameters `q`)

**content**
- `/content`
  - `GET` `/content` (with parameters `q`)

**content_search**
- `/content_search`
  - `GET` `/content_search` (with parameters `q`)

**custom_objects**
- `/custom_objects`
  - `POST` `/custom_objects` (with parameters `q`)

**customers**
- `/customers`
  - `GET` `/customers` (with parameters `q`)

**folders**
- `/folders`
  - `GET` `/folders` (with parameters `q`)

**gift_certificate**
- `/gift_certificate`
  - `GET` `/gift_certificate` (with parameters `q`)

**order_search**
- `/order_search`
  - `GET` `/order_search` (with parameters `q`)

**orders**
- `/orders`
  - `GET` `/orders` (with parameters `q`)

---

**Identify OCAPI Endpoints**

With requirements identified for OCAPI integration — specific OCAPI endpoints and corresponding requests should be identified so that they can be modeled prior to application development. Emphasize that OCAPI's documentation and API Explorer enable developers to explore OCAPI's SHOP and DATA apis.
Authenticate a Customer

Commerce Cloud supports the JSON Web Token (JWT) standard to authenticate against OCAPI Shop API resources. Custom applications can obtain a JWT for a customer by using the /customers/auth endpoint. When initializing a new customer experience, the customer must be authenticated as a guest or registered user.

When you request a JWT using the /customers/auth endpoint, you must specify your client ID as a URL or header parameter. You must also specify the type of customer:

- Specify "type":"guest" to request a JWT for a guest customer.
- Specify "type":"credentials" to request a JWT for a registered customer. (For a registered customer, you must also include the customer login and password in the HTTP Basic Authentication scheme.)
- Specify "type":"session" to request a JWT for a customer (guest or registered) associated with a session. (You have to provide valid dwsid anddwsecuretoken cookies.)

API Explorer Link: SHOP API /customers/auth endpoint

Create a Customer Session

Once a JWT token has been created, it can be exchanged for a customer session. The created session is associated with the JWT's customer and can be used in conjunction with stateless OCAPI calls that would use the JWT token. The OCAPI session bridge can then be used to allow seamless interactions between a customer's storefront and OCAPI sessions.

API Explorer Link: SHOP API /sessions endpoint

Retrieve Storefront Products

The products resource can be used to retrieve various views of product information for single or multiple products. This includes product metadata, availability information, and pricing details, as well as imagery and product detail links.

API Explorer Link: SHOP API /product endpoints
Create or Retrieve a Customer’s Basket

The baskets resource can be used to create and retrieve basket information. For registered customers, the /customers/{customer_id}/baskets endpoint can also be leveraged to retrieve the total number of baskets open and available to a customer.

**API Explorer Links:** SHOP API /baskets endpoints and /customers endpoints

Add a Product to or Remove it from a Customer’s Basket

Individual products can be added to a customer's cart via the /baskets/{basket_id}/items endpoint. Additionally, products can be modified in or removed from a customer’s cart via the baskets resource.

**API Explorer Link:** SHOP API /baskets endpoints

OCAPI Performance Best Practices

The Commerce Cloud platform documentation includes guidance on how to develop high-performing OCAPI applications. This documentation explains how OCAPI requests differ from storefront requests, how OCAPI requests are cached, how OCAPI cache keys are generated, and how to improve the performance of OCAPI requests.