

- [Home](#)
- [User](#)
- [Developer](#)
- [Hub](#)
- [API](#)
- [Release Notes](#)

# Shipments - Developer Guide | Spree Commerce

## Overview

This guide explains how Spree represents shipping options and how it calculates expected costs, and shows how you can configure the system with your own shipping methods. After reading it you should know:

- how shipments and shipping are implemented in Spree
- how to specify your shipping structure
- how split shipments work
- how to configure products for special shipping treatment
- how to capture shipping instructions

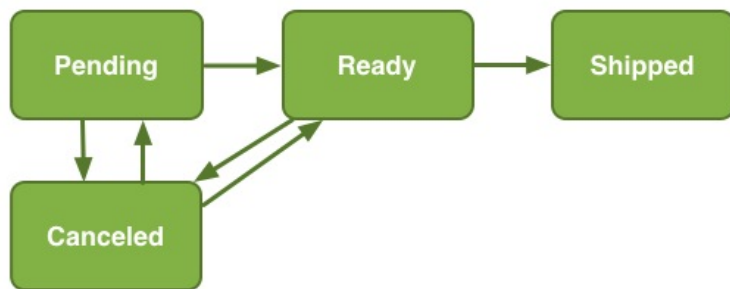
Spree uses a very flexible and effective system to calculate shipping, accommodating the full range of shipment pricing: from simple flat rate to complex product-type- and weight-dependent calculations.

The Shipment model is used to track how items are delivered to the buyer.

Shipments have the following attributes:

- **number:** The unique identifier for this shipment. It begins with the letter H and ends in an 11-digit number. This number is shown to the users, and can be used to find the order by calling `Spree::Shipment.find_by(number: number)`.
- **tracking:** The identifier given for the shipping provider (i.e. FedEx, UPS, etc).
- **shipped\_at:** The time when the shipment was shipped.
- **state:** The current state of the shipment.
- **stock\_location\_id:** The ID of the Stock Location where the items for this shipment will be sourced from.

A shipment can go through many different states, as illustrated below.



An explanation of the different states:

- pending: The shipment has backordered inventory units and/or the order is not paid for.
- ready: The shipment has *no* backordered inventory units and the order is paid for.
- shipped: The shipment is on its way to the buyer.
- canceled: When an order is cancelled, all of its shipments will also be cancelled. When this happens, all items in the shipment will be restocked. If an order is “resumed”, then the shipment will also be resumed.

Explaining each piece of the shipment world inside of Spree separately and how each piece fits together can be a cumbersome task. Fortunately, using a few simple examples makes it much easier to grasp. In that spirit, the examples are shown first in this guide.

## Examples

### Simple Setup

Consider you sell T-shirts to the US and Europe and ship from a single location, and you work with 2 deliverers:

- USPS Ground (to US)
- FedEx (to EU)

and their pricing is as follow:

- USPS charges \$5 for one T-shirt and \$2 for each additional one
- FedEx charges \$10 each, regardless of the quantity

To achieve this setup you need the following configuration:

- Shipping Categories: All your products are the same, so you only need to define one default shipping category. Each of your products would then need to be assigned to this shipping category.
- 1 Stock Location: You are shipping all items from the same location, so you can use the default.
- 2 Shipping Methods (Configuration->Shipping Methods) as follows:

Name	Zone	Calculator
USPS Ground	US	Flexi Rate(\$5,\$2)
FedEx EU_VAT		FlatRate-per-item(\$10)

### Advanced Setup

Consider you sell products to a single zone (US) and you ship from 2 locations (Stock Locations):

- New York
- Los Angeles

and you work with 3 deliverers (Shipping Methods):

- FedEx
- DHL
- US postal service

and your products can be classified into 3 Shipping Categories:

- Light
- Regular
- Heavy

and their pricing is as follow:

FedEx charges:

- \$10 for all light items regardless of how many you have
- \$2 per regular item
- \$20 for the first heavy item and \$15 for each additional one

DHL charges:

- \$5 per item if it's light or regular
- \$50 per item if it's heavy

USPS charges:

- \$8 per item if it's light or regular
- \$20 per item if it's heavy

To achieve this setup you need the following configuration:

- 4 Shipping Categories: Default, Light, Regular and Heavy
- 3 Shipping Methods (Configuration->Shipping Methods): FedEx, DHL, USPS
- 2 Stock Locations (Configuration->Stock Locations): New York, Los Angeles

S. Category / S. Method	DHL	FedEx	USPS
Light	Per Item (\$5)	Flat Rate (\$10)	Per Item (\$8)
Regular	Per Item (\$5)	Per Item (\$2)	Per Item (\$8)
Heavy	Per Item (\$50)	Flexi Rate(\$20,\$15)	Per Item (\$20)

## Design & Functionality

To properly leverage Spree's shipping system's flexibility you must understand a few key concepts:

- Shipping Methods
- Zones
- Shipping Categories
- Calculators (through Shipping Rates)

### Shipping Methods

Shipping methods are the actual services used to send the product. For example:

- UPS Ground
- UPS One Day
- FedEx 2Day
- FedEx Overnight
- DHL International

Each shipping method is only applicable to a specific Zone. For example, you wouldn't be able to get a package delivered internationally using a domestic-only shipping method. You can't ship from Dallas, USA to Rio de Janeiro, Brazil using UPS Ground (a US-only carrier).

If you are using shipping categories, these can be used to qualify or disqualify a given shipping method.

**Note:** Shipping methods can now have multiple shipping categories assigned to them. This allows the shipping methods available to an order to be determined by the shipping categories of the items in a shipment.

## Zones

Zones serve as a mechanism for grouping geographic areas together into a single entity. You can read all about how to configure and use Zones in the [Zones Guide](#).

The Shipping Address entered during checkout will define the zone the customer is in and limit the Shipping Methods available to him.

## Shipping Categories

Shipping Categories are useful if you sell products whose shipping pricing vary depending on the type of product (TVs and Mugs, for instance).

For simple setups, where shipping for all products is priced the same (ie. T-shirt-only shop), all products would be assigned to the default shipping category for the store.

Some examples of Shipping Categories would be:

- Light (for lightweight items like stickers)
- Regular
- Heavy (for items over a certain weight)

Shipping Categories are created in the admin interface (“Configuration” -> “Shipping Categories”) and then assigned to products (“Products” -> “Edit”).

\*\*\*\*\* TODO \*\*\*\*\*

Follow up: on a clean install + seed data I ended up with two Shipping Categories - “Default Shipping” and “Default”

\*\*\*\*\*

During checkout, the shipping categories of the products in your order will determine which calculator will be used to price its shipping for each Shipping Method.

## Calculators

A Calculator is the component responsible for calculating the shipping price for each available Shipping Method.

Spree ships with 5 default Calculators:

- Flat rate (per order)
- Flat rate (per item/product)
- Flat percent
- Flexible rate
- Price sack

Flexible rate is defined as a flat rate for the first product, plus a different flat rate for each additional product.

You can define your own calculator if you have more complex needs. In that case, check out the [Calculators Guide](#).

## UI

### What the Customer Sees

In the standard system, there is no mention of shipping until the checkout phase.

After entering a shipping address, the system displays the available shipping options and their costs for each shipment in the order. Only the shipping options whose zones include the *shipping* address are presented.

The customer must choose a shipping method for each shipment before proceeding to the next stage. At the confirmation step, the shipping cost will be shown and included in the order’s total.

-- . . . . . -- . . . . .

You can enable collection of extra *shipping instructions* by setting the option `Spree::Config.shipping_instructions` to `true`. This is set to `false` by default. See [Shipping Instructions](#) below.

## What the Order’s Administrator Sees

`Shipment` objects are created during checkout for an order. Initially each records just the shipping method and the order it applies to. The administrator can update the record with the actual shipping cost and a tracking code, and may also (once only) confirm the dispatch. This confirmation causes a shipping date to be set as the time of confirmation.

## Advanced Shipping Methods

Spree comes with a set of calculators that should fit most of the shipping situations that may arise. If the calculators that come with Spree are not enough for your needs, you might want to use an extension - if one exists to meet your needs - or create a custom one.

### Extensions

There are a few Spree extensions which provide additional shipping methods, including special support for fees imposed by common carriers, or support for bulk orders. See the [Spree Extension Registry](#) for the latest information.

### Writing Your Own

For more detailed information, check out the section on [Calculators](#).

Your calculator should accept an array of `LineItem` objects and return a cost. It can look at any reachable data, but typically uses the address, the order and the information from variants which are contained in the `line_items`.

## Product Configuration

Store administrators can assign products to specific `ShippingCategories` or include extra information in variants to enable the calculator to determine results.

Each product has a `ShippingCategory`, which adds product-specific information to the calculations beyond the standard information from the shipment. Standard information includes:

- Destination address
- Variants and quantities
- Weight and dimension information, if given, for a variant

`ShippingCategory` is basically a wrapper for a string. One use is to code up specific rates, eg. “Fixed \$20” or “Fixed \$40”, from which a calculator could extract imposed prices (and not go through its other calculations).

### Variant Configuration

Variants can be specified with weight and dimension information. Some shipping method calculators will use this information if it is present.

## Shipping Instructions

The option `Spree::Config[:shipping_instructions]` controls collection of additional shipping instructions. This is turned off (set to `false`) by default. If an order has any shipping instructions attached, they will be shown in an order’s shipment admin page and can also be edited at that stage. Observe that instructions are currently attached to the *order* and not to actual *shipments*.

## The Active Shipping Extension

The popular `spree_active_shipping` extension harnesses the `active_shipping` gem to interface with carrier APIs such as USPS, Fedex and UPS, ultimately providing Spree-compatible calculators for the different delivery services of those carriers.

To install the spree-active-shipping extension add the following to your Gemfile:

```
gem 'spree_active_shipping'
gem 'active_shipping', :git => 'git://github.com/Shopify/active_shipping.git'
```

and run `bundle install` from the command line.

As an example of how to use the [spree\\_active\\_shipping extension](#) we'll demonstrate how to configure it to work with the USPS API. The other carriers follow a very similar pattern.

For each USPS delivery service you want to offer (e.g. "USPS Media Mail"), you will need to create a `ShippingMethod` with a descriptive name ("Configuration" -> "Shipping Methods") and a `Calculator` (registered in the `active_shipping` extension) that ties the delivery service and the shipping method together.

## Default Calculators

The `spree_active_shipping` extension comes with several pre-configured calculators out of the box. For example, here are the ones provided for the USPS carrier:

```
def activate
  [
    #... calculators for Fedex and UPS not shown ...
    Calculator::Usps::MediaMail,
    Calculator::Usps::ExpressMail,
    Calculator::Usps::PriorityMail,
    Calculator::Usps::PriorityMailSmallFlatRateBox,
    Calculator::Usps::PriorityMailRegularMediumFlatRateBoxes,
    Calculator::Usps::PriorityMailLargeFlatRateBox
  ].each(&:register)
end
```

Each USPS delivery service you want to make available at checkout has to be associated with a corresponding shipping method. Which shipping methods are made available at checkout is ultimately determined by the zone of the customer's shipping address. The USPS' basic shipping categories are domestic and international, so we'll set up zones to mimic this distinction. We need to set up two zones then - a domestic one, consisting of the USA and its territories; and an international one, consisting of all other countries.

With zones in place, we can now start adding some shipping methods through the admin panel. The only other essential requirement to calculate the shipping total at checkout is that each product and variant be assigned a weight.

The `spree_active_shipping` gem needs some configuration variables set in order to consume the carrier web services.

```
# these can be set in an initializer in your site extension
Spree::ActiveShipping::Config.set(:usps_login => "YOUR_USPS_LOGIN")
Spree::ActiveShipping::Config.set(:fedex_login => "YOUR_FEDEX_LOGIN")
Spree::ActiveShipping::Config.set(:fedex_password => "YOUR_FEDEX_PASSWORD")
Spree::ActiveShipping::Config.set(:fedex_account => "YOUR_FEDEX_ACCOUNT")
Spree::ActiveShipping::Config.set(:fedex_key => "YOUR_FEDEX_KEY")
```

## Adding Additional Calculators

Additional delivery services that are not pre-configured as a calculator in the `spree_active_shipping` extension can be easily added. Say, for example, you need First Class International Parcels via the US Postal Service.

First, create a calculator class that inherits from `Calculator::Usps::Base` and implements a description class method:

```
class Calculator::Usps::FirstClassMailInternationalParcels < Calculator::Usps::Base
  def self.description
    "USPS First-Class Mail International Package"
  end
end
```

Note that, unlike calculators that you write yourself, these calculators do not have to implement a `compute` instance method that returns a shipping amount. The subclasses take care of that requirement.

There is one gotcha to bear in mind: the string returned by the `description` method must *exactly* match the name of the USPS delivery service. To determine the exact spelling of the delivery service, you'll need to examine what gets returned from the API:

```
class Calculator::ActiveShipping < Calculator
  def compute(line_items)
    #....
    rates = retrieve_rates(origin, destination, packages(order))
    # the key of this hash is the name you need to match
    # raise rates.inspect

    return nil unless rates
    rate = rates[self.description].to_f + (Spree::ActiveShipping::Config[:handling_fee].to_f || 0.0)
    return nil unless rate
    # divide by 100 since active_shipping rates are expressed as cents

    return rate/100.0
  end

  def retrieve_rates(origin, destination, packages)
    #....
    # carrier is an instance of ActiveMerchant::Shipping::USPS
    response = carrier.find_rates(origin, destination, packages)
    # turn this beastly array into a nice little hash
    h = Hash[*response.rates.collect { |rate| [rate.service_name, rate.price] }.flatten]
    #....
  end
end
```

As you can see in the code above, the `spree_active_shipping` gem returns an array of services with their corresponding prices, which the `retrieve_rates` method converts into a hash. Below is what would get returned for an order with an international destination:

```
{
  "USPS Priority Mail International Flat Rate Envelope"=>1345,
  "USPS First-Class Mail International Large Envelope"=>376,
  "USPS USPS GXG Envelopes"=>4295,
  "USPS Express Mail International Flat Rate Envelope"=>2895,
  "USPS First-Class Mail International Package"=>396,
  "USPS Priority Mail International Medium Flat Rate Box"=>4345,
  "USPS Priority Mail International"=>2800,
  "USPS Priority Mail International Large Flat Rate Box"=>5595,
  "USPS Global Express Guaranteed Non-Document Non-Rectangular"=>4295,
  "USPS Global Express Guaranteed Non-Document Rectangular"=>4295,
  "USPS Global Express Guaranteed (GXG)"=>4295,
  "USPS Express Mail International"=>2895,
  "USPS Priority Mail International Small Flat Rate Box"=>1345
}
```

From all of the viable shipping services in this hash, the `compute` method selects the one that matches the description of the calculator. At this point, an optional flat handling fee (set via preferences) can be added:

```
rate = rates[self.description].to_f + (Spree::ActiveShipping::Config[:handling_fee].to_f || 0.0)
```

Finally, don't forget to register the calculator you added. In extensions, this is accomplished with the `activate` method:

```
def activate
  Calculator::Usps::FirstClassMailInternationalParcels.register
end
```

## Filtering Shipping Methods On Criteria Other Than the Zone

Ordinarily, it is the zone of the shipping address that determines which shipping methods are displayed to a customer at checkout. Here is how the availability of a shipping method is determined:

```
class Spree::Stock::Estimator
  def shipping_methods(package)
    shipping_methods = package.shipping_methods
    shipping_methods.delete_if { |ship_method| !ship_method.calculator.available?(package.contents) }
    shipping_methods.delete_if { |ship_method| !ship_method.include?(order.ship_address) }
    shipping_methods.delete_if { |ship_method| !(ship_method.calculator.preferences[:currency].nil? || ship_method.calculator.preferences[:currency] == currency) }
    shipping_methods
  end
end
```

Unless overridden, the calculator's `available?` method returns `true` by default. It is, therefore, the zone of the destination address that filters out the shipping methods in most cases. However, in some circumstances it may be necessary to filter out additional shipping methods.

Consider the case of the USPS First Class domestic shipping service, which is not offered if the weight of the package is greater than 13oz. Even though the USPS API does not return the option for First Class in this instance, First Class will appear as an option in the checkout view with an unfortunate value of 0, since it has been set as a Shipping Method.

To ensure that First Class shipping is not available for orders that weigh more than 13oz, the calculator's `available?` method must be overridden as follows:

```
class Calculator::Usps::FirstClassMailParcels < Calculator::Usps::Base
  def self.description
    "USPS First-Class Mail Parcel"
  end

  def available?(order)
    multiplier = 1.3
    weight = order.line_items.inject(0) do |weight, line_item|
      weight + (line_item.variant.weight ? (line_item.quantity * line_item.variant.weight * multiplier) : 0)
    end
    #if weight in ounces > 13, then First Class Mail is not available for the order
    weight > 13 ? false : true
  end
end
```

## Split Shipments


### Introduction

Split shipments are a new feature as of Spree 2.0 that addresses the needs of complex Spree stores that require sophisticated shipping and warehouse logic. This includes detailed inventory management and allows for shipping from multiple locations.



SHIPPING METHOD


PACKAGE FROM DEFAULT

IMAGE	NAME	QUANTITY	PRICE
	Spree Tote	2	\$15.99

SHIPPING METHOD

☒ UPS Ground (USD) \$5.00
 ☐ UPS Two Day (USD) \$10.00
 ☐ UPS One Day (USD) \$15.00

PACKAGE FROM LONDON

IMAGE	NAME	QUANTITY	PRICE
	Ruby on Rails Mug	3	\$13.99

SHIPPING METHOD

☒ UPS Ground (USD) \$5.00
 ☐ UPS Two Day (USD) \$10.00
 ☐ UPS One Day (USD) \$15.00

ORDER SUMMARY

Item Total:	\$73.95
North America 5.0%:	\$1.60
Shipping:	\$5.00
Shipping:	\$5.00
Order Total:	\$85.55

## Creating Proposed Shipments

This section steps through the basics of what is involved in determining shipments for an order. There are a lot of pieces that make up this process. They are explained in detail in the [Components of Split Shipments](#) section of this guide.

The process of determining shipments for an order is triggered by calling `create_proposed_shipments` on an `Order` object while transitioning to the `delivery` state during checkout. This process will first delete any existing shipments for an order and then determine the possible shipments available for that order.

`create_proposed_shipments` will initially call `Spree::Stock::Coordinator.new(@order).packages`. This will return an array of packages. In order to determine which items belong in which package when they are being built, Spree uses an object called a `Splitter`, described in more detail [below](#).

After obtaining the array of available packages, they are converted to shipments on the order object. Shipping rates are determined and inventory units are created during this process as well.

At this point, the checkout process can continue to the delivery step.

## Components of Split Shipments

This section describes the four main components that power split shipments: [The Coordinator](#), [The Packer](#), [The Prioritizer](#), and [The Estimator](#).

### The Coordinator

The `Spree::Stock::Coordinator` is the starting point for determining shipments when calling `create_proposed_shipments` on an order. Its job is to go through each `StockLocation` available and determine what can be shipped from that location.

The `Spree::Stock::Coordinator` will ultimately return an array of packages which can then be easily converted into shipments for an order by calling `to_shipment` on them.

### The Packer

A `Spree::Stock::Packer` object is an important part of the `create_proposed_shipments` process. Its job is to determine possible packages for a given `StockLocation` and order. It uses rules defined in classes

A `Spree::Stock::Packer` object is an important part of the `create_proposed_shipments` process. Its job is to determine possible packages for a given `StockLocation` and order. It uses rules defined in classes known as `Splitters` to determine what packages can be shipped from a `StockLocation`.

For example, we may have two splitters for a stock location. One splitter has a rule that any order weighing more than 50lbs should be shipped in a separate package from items weighing less. Our other splitter is a catch-all for any item weighing less than 50lbs. So, given one item in an order weighing 60lbs and two items weighing less, the `Packer` would use the rules defined in our splitters to come up with two separate packages: one containing the single 60lb item, the other containing our other two items.

## Default Splitters

Spree comes with two default splitters which are run in sequence. This means that the first splitter takes the packages array from the order, and each subsequent splitter takes the output of the splitter that came before it.

Let's take a look at what the default splitters do:

- **Shipping Category Splitter:** Splits an order into packages based on items' shipping categories. This means that each package will only have items that all belong to the same shipping category.
- **Weight Splitter:** Splits an order into packages based on a weight threshold. This means that each package has a mass weight. If a new item is added to the order and it causes a package to go over the weight threshold, a new package will be created so that all packages weigh less than the threshold. You can set the weight threshold by changing `Spree::Stock::Splitter::Weight.threshold` (defaults to 150) in an initializer.

## Custom Splitters

Note that splitters can be customized, and creating your own can be done with relative ease. By inheriting from `Spree::Stock::Splitter::Base`, you can create your own splitter.

For an example of a simple splitter, take a look at Spree's [weight based splitter](#). This splitter pulls items with a weight greater than 150 into their own shipment.

After creating your splitter, you need to add it to the array of splitters Spree uses. To do this, add the following to your application's spree initializer `spree.rb` file:

```
Rails.application.config.spree.stock_splitters << Spree::Stock::Splitter::CustomSplitter
```

You can also completely override the splitters used in Spree, rearrange them, etc. To do this, add the following to your `spree.rb` file:

```
Rails.application.config.spree.stock_splitters = [  
  Spree::Stock::Splitter::CustomSplitter,  
  Spree::Stock::Splitter::ShippingCategory  
]
```

Or if you don't want to split packages just set the option above to an empty array. e.g. a store with the following configuration in `spree.rb` won't have any package splitted.

```
Rails.application.config.spree.stock_splitters = []
```

If you want to add different splitters for each `StockLocation`, you need to decorate the `Spree::Stock::Coordinator` class and override the `splitters` method.

## The Prioritizer

A `Spree::Stock::Prioritizer` object will decide which `StockLocation` should ship which package from an order. The prioritizer will attempt to come up with the best shipping situation available to the user.

By default, the prioritizer will first select packages where the items are on hand. Then it will try to find packages where items are backordered. During this process, the `Spree::Stock::Adjuster` is also used to ensure each package has the correct number of items.

The prioritizer is also a customization point. If you want to customize which packages should take priority for the order during this process, you can override the `sort_packages` method in `Spree::Stock::Prioritizer`.

### Customizing the Adjuster

## Customizing the Adjuster

The Adjuster visits each package in an order and ensures the correct number of items are in each package. To customize this functionality, you need to do two things:

- Subclass the [Spree::Stock::Adjuster](#) class and override the `adjust` method to get the desired functionality.
- Decorate the `Spree::Stock::Coordinator` and override the `prioritize_packages` method, passing in your custom adjuster class to the `Prioritizer` initializer. For example, if our adjuster was called `Spree::Stock::CustomAdjuster`, we would do the following:

```
Spree::Stock::Coordinator.class_eval do
  def prioritize_packages(packages)
    prioritizer = Prioritizer.new(order, packages, Spree::Stock::CustomAdjuster)
    prioritizer.prioritized_packages
  end
end
```

## The Estimator

The `Spree::Stock::Estimator` loops through the packages created by the packer in order to calculate and attach shipping rates to them. This information is then returned to the user so they can select shipments for their order and complete the checkout process.

- [Tutorials](#)
  - [Getting Started](#)
  - [Extensions](#)
  - [Deface Overrides](#)
- [Source Code](#)
  - [About](#)
  - [Navigating](#)
  - [Getting Help](#)
  - [Contributing](#)
- [The Core](#)
  - [Addresses](#)
  - [Adjustments](#)
  - [Calculators](#)
  - [Inventory](#)
  - [Orders](#)
  - [Payments](#)
  - [Preferences](#)
  - [Products](#)
  - [Promotions](#)
  - [Shipments](#)
  - [Taxation](#)
- [Customization](#)
  - [Authentication](#)
  - [Internationalization \(i18n\)](#)
  - [View](#)
  - [Account](#)

- [Assets](#)
- [Logic](#)
- [Use S3](#)
- [Checkout](#)

- **[Deployment](#)**

- [Ninefold](#)
- [Heroku](#)
- [Shelly Cloud](#)
- [Ansible Ubuntu Deployment](#)
- [Manual Ubuntu Deployment](#)
- [Deployment Options](#)
- [Deployment Service](#)
- [Deployment Tips](#)
- [Requesting/Configuring SSL](#)

- **[Advanced Topics](#)**

- [Search Engine Optimization](#)
- [Developer Tips](#)
- [Migrating to Spree](#)
- [Security](#)
- [Testing Spree Applications](#)

- **[Upgrade Guides](#)**

- [0.60.x to 0.70.x](#)
- [0.70.x to 1.0.x](#)
- [1.0.x to 1.1.x](#)
- [1.1.x to 1.2.x](#)
- [1.2.x to 1.3.x](#)
- [1.3.x to 2.0.x](#)
- [2.0.x to 2.1.x](#)
- [2.1.x to 2.2.x](#)
- [2.2.x to 2.3.x](#)

## Product

- [Platform](#)
- [Hub](#)
- [Support](#)
- [Privacy Policy](#)
- [Terms of Service](#)

## Developers

- [Overview](#)
- [Documentation](#)
- [Community](#)
- [License](#)

## Partners

## Partners

- [Pro Services](#)
- [Training](#)
- [Partnership Program](#)
- [Payments](#)

## About Spree Commerce

Spree Commerce is an automated enterprise solution built specifically for ecommerce. We effectively manage your operations so you can focus on serving your customers and growing your business.

## Take it for a test drive

You can even create your own personal store.

[Try The Demo](#)

Spree, Spree Commerce and “Behind the Best Storefronts” are all trademarks of Spree Commerce Inc.

- 
- 
- 
-