

Account Area

- [General](#)

- [How do I add a Custom CSS File to the account area?](#)
- [How can I configure the account area?](#)

- [Slots](#)

- [What are the available account area slots?](#)
- [How do I inject into account area slots?](#)

- [Pages](#)

- [What are the default pages in the account area?](#)
- [How do I create a custom page in the account area?](#)
- [How do I add a section to an existing account area page?](#)

- [Sets](#)

- [What are the default sets in the account area?](#)
- [How do I create a custom set in the account area?](#)

- [Forms](#)

- [What are the default forms in the account area?](#)
- [How do I create a custom form in the account area?](#)

General

How do I add a Custom CSS File to the account area?

You can add a custom CSS file to the account area by creating a file named `account-area.css` in your public directory.

Example contents where we're overriding the default colours:

```
:root {  
  --bpa-color-primary: deepskyblue;  
  --bpa-color-success: forestgreen;  
  --bpa-color-warning: darkorange;  
  --bpa-color-error: red;  
  --bpa-color-helpbox: aliceblue;  
  --bpa-color-content-background: ghostwhite;  
}
```

You can also place this file in your theme if preferred.

How can I configure the account area?

ENV Variables

Variable	Default	Description
AERO_ACCOUNT_AREA_THEME	aerocommerce/account-area	Defines the namespace that should be registered as having the views and resources for the account area. This should only be changed under unique circumstances.
AERO_ACCOUNT_AREA_LAYOUT	layouts.main	Defines the layout that the account area views use. If the layout provided doesn't exist, account-area::layouts.blank will be used.

Config File

If you would like to change any of the config values you can publish the config file by running:

```
php artisan vendor:publish --provider="Aero\AccountArea\ServiceProvider"
```

Once the file is published you will be able to edit it through the config php file created at `/config/aero/account-area.php`.

date_format

This defines the format used when displaying any dates in the account area. `js F Y` by default, e.g. `25/09/2026`.

subscription_status_state_class

Maps subscription status states to specific colors, e.g. `active => success`. So when a subscription has a status with active state, the orb will have the class `bpa-indicator--success` (green).

order_status_state_class

Maps order status states to specific colors, e.g. `active => success`. So when an order has a status with active state, the orb will have the class `bpa-indicator--success` (green).

order_status_radio_options

This defines the order statuses shown in the radio options on the orders page and what order status states they map to.

order_years

This defines the number of previous years to display in the years dropdown on the orders page.

page_sections

This defines the sections that make up each page by default. You can add/edit/remove sections through code, it is not necessary to adjust these values usually.

form_sections

This defines the sections that make up each form by default. You can add/edit/remove sections through code, it is not necessary to adjust these values usually.

slots

This defines the views that are injected into slots by default.

links

This defines the links that should be in the navbar by default. You can add/edit/remove links through code, it is not necessary to adjust these values usually.

Slots

What are the available account area slots?

Navbar

- `navbar.info`

Addresses

- `addresses.address.card.footer`

Orders

- `orders.order.card.dispatched`
- `orders.order.card.not-dispatched`
- `order.header.buttons`
- `orders.order.card.header`
- `orders.order.card.footer`

Subscriptions

- `subscriptions.subscription.card.header`
- `subscriptions.subscription.card.footer`

How do I inject into account area slots?

Account Area Slots allow you to add content within existing account area pages. The examples below show how to inject a view into the orders page.

For a simple use case, simply provide a view:

```
\Aero\AccountArea\AccountAreaSlot::inject('order.header.buttons', 'acme::button');
```

For more advanced use cases, instead of passing a view directly, you can provide a callback function:

```
\Aero\AccountArea\AccountAreaSlot::inject('order.header.buttons', function ($data) {  
    // add custom variables to the $data array  
  
    return view('acme::button', $data);  
});
```

These injections should typically sit inside the `boot` method of a Service Provider

Pages

What are the default pages in the account area?

Class	Route	Description
<code>AccountLoginPage</code>	<code>login</code>	A page where users can log in.
<code>AccountRegisterPage</code>	<code>register</code>	A page where guests can register.
<code>AccountPasswordForgotPage</code>	<code>password.forgot</code>	A page where users can send a password reset email to their email.
<code>AccountPasswordResetPage</code>	<code>password.reset</code>	A page where users can reset their password.
<code>AccountOrdersPage</code>	<code>account.orders</code>	A page that lists all of a user's orders and lets them filter/search.
<code>AccountOrderViewPage</code>	<code>account.view-order</code>	A page that shows more information about an order.
<code>AccountAddressesPage</code>	<code>account.addresses</code>	A page that lists all of the users addresses.
<code>AccountAddressNewPage</code>	<code>account.new-address</code>	A page where users can create a new address,
<code>AccountAddressEditPage</code>	<code>account.edit-address</code>	A page where users can edit one of their addresses.
<code>AccountDetailsPage</code>	<code>account.details</code>	A page where users can update their account details and change their password.
<code>AccountInvoicePage</code>	<code>account.view-invoice</code>	A simple invoice for an order - currently not linked to from anywhere.
<code>AccountOverviewPage</code>	<code>account</code>	The first page a user sees after they log in - currently redirects to the orders page.

The above classes reside in the `Aero\AccountArea\Http\Responses` namespace.

How do I create a custom page in the account area?

To create and register a custom account area page you need to use the `Aero\AccountArea\AccountArea::registerPage()` helper method.

This method expects you to pass a class that extends `Aero\AccountArea\AccountAreaPage`:

```
\Aero\AccountArea\AccountArea\AccountArea::registerPage(\Aerocargo\Acme\CustomPage::class);
```

Your account area page class must implement the `title()`, `route()`, and `routeName()` methods that all should return a string.

It's common practice to use [Laravel Localization](#) for the title and route so that they can change depending on the language used. You should not use this for the `routeName` as the `routeName` is used in code to refer to the route and therefore should never change.

You should also define a protected static steps and middleware array and a protected sections array:

- The static steps array can be used to define any extra response steps required for your page.
- The middleware array can be used to define any middleware required for your page (the most useful being 'account' which means users need to be logged in to view your page).
- The sections array defines the views that make up your page.

For example, here's how [aerocargo/wishlists](#) defines its page:

```
class WishlistPage extends \Aero\AccountArea\AccountAreaPage
{
    use \Aero\AccountArea\Traits\HasAccountAreaLink;

    protected static $steps = [
        Steps\AttachWishlist::class,
    ];

    protected static $middleware = [
```

```

        'account',
    ];

    protected $sections = [
        'navbar' => 'account-area::sections.navbar',
        'page-header' => 'account-area::partials.page-header',
        'wishlist' => 'wishlist::account-area-wishlist',
    ];

    static function title(): string
    {
        return __('My Wishlist');
    }

    static function route(): string
    {
        return __('wishlist');
    }

    static function routeName(): string
    {
        return 'account.wishlist';
    }

    public static function toLink(): array
    {
        return [
            'icon' => 'wishlist::icons.heart',
            'text' => 'Your Wishlist',
        ];
    }
}

```

You can add your page to the navbar using the `addLink()` method:

```

\Aero\AccountArea\AccountAreaPage::extend(function (\Aero\AccountArea\AccountAreaPage $page,
\Closure $next) {
    $page->addLink(\Aerocargo\Wishlist\Http\Responses\WishlistPage::class);

    return $next($page);
}

```

```
});
```

For `addLink` to work, your page needs to use the `\Aero\AccountArea\Traits\HasAccountAreaLink` trait and implement the `toLink()` method (see the wishlist example above)

How do I add a section to an existing account area page?

You can use the `addSection()` method(s) to add a section to an existing form:

```
\Aero\AccountArea\Http\Responses\AccountLoginPage::extend(function
(\Aero\AccountArea\Http\Responses\AccountLoginPage $page, \Closure $next) {
    // Add it as the first section
    $page->addSection('acme', 'acme::section');

    // Add it after a specific section
    $page->addSectionAfter('acme', 'acme::custom', 'login-form');

    // Add it before a specific section
    $page->addSectionBefore('acme', 'acme::custom', 'login-form');

    return $next($page);
});
```

If you need to see the keys of the existing columns in order to position your new column appropriately, you can dump them out:

```
\Aero\AccountArea\Http\Responses\AccountLoginPage::extend(function
(\Aero\AccountArea\Http\Responses\AccountLoginPage $page, \Closure $next) {
    dd($page->getSections()->keys()->toArray());

    return $next($page);
});
```

Sets

What are the default sets in the account area?

Class	Method	Route
AccountLoginSet	Post	login
AccountLogoutSet	Post	logout
AccountRegisterSet	Post	register
AccountPasswordForgotSet	Post	password.forgot
AccountPasswordResetSet	Post	password.reset
AccountAddressNewSet	Post	account.new-address
AccountAddressEditSet	Put	account.edit-address
AccountAddressMakeDefaultSet	Put	account.make-default-address
AccountDetailsSet	Put	account.details
AccountAddressDelete	Delete	account.delete-address

The above classes reside in the `Aero\AccountArea\Http\Responses` namespace.

How do I create a custom set in the account area?

Sets are used for POST/PUT/DELETE requests.

To create and register a custom account area set you need to use one of the following methods:

- `Aero\AccountArea\AccountArea::registerPost()`
- `Aero\AccountArea\AccountArea::registerPut()`
- `Aero\AccountArea\AccountArea::registerDelete()`

These methods expect you to pass a class that extends `Aero\AccountArea\AccountAreaSet`.

```
\Aero\AccountArea\AccountArea\AccountArea::registerDelete(\Aerocargo\Acme\CustomDelete::class)
;
```

Your account area set class must implement the `route()` and `routeName()` methods that all should return a string.

You should also define a protected static steps and middleware array:

- The steps array can be used to define any extra response steps required for your set.
- The middleware array can be used to define any middleware required for your set (the most useful being 'account' which means the user needs to be logged in).

For example, here's how [aerocargo/wishlists](#) defines its delete:

```
class WishlistDelete extends \Aero\AccountArea\AccountAreaSet
{
    protected static $steps = [
        Steps\DeleteWishlist::class,
    ];

    protected static $middleware = [
        'account',
    ];
}
```

```
static function route(): string
{
    return __('wishlist');
}

static function routeName(): string
{
    return 'account.wishlist';
}
}
```

Forms

What are the default forms in the account area?

Class	Description
<code>AccountAddressEditForm</code>	The edit address form.
<code>AccountAddressNewForm</code>	The new address form.
<code>AccountDetailsForm</code>	The account details form.
<code>AccountLoginForm</code>	The login form.
<code>AccountPasswordForgotForm</code>	The forgotten password form.
<code>AccountPasswordResetForm</code>	The password reset form.
<code>AccountRegisterForm</code>	The register form.

The above classes reside in the `Aero\AccountArea\Http\Forms` namespace.

How do I create a custom form in the account area?

To create a form you need to create a class that extends the `\Aero\AccountArea\AccountAreaForm` class:

- The `$sections` array lists the views used to render the form.
- The `method()` method defines the method that the form will use (this should be `post`, `get`, `put`, or `delete`).
- The `route()` method should return the URL for the form.
 - A `$data` variable is accessible in case the route requires parameters that are available in the data.

For example, here's how the login form is defined:

```
class LoginForm extends \Aero\AccountArea\AccountAreaForm
{
    protected $sections = [
        'email' => 'account-area::partials.email-input',
        'password' => 'account-area::partials.password-input',
        'forgot-password' => 'account-area::partials.login-forgot-password-link',
        'submit' => 'account-area::partials.login-button',
    ];

    protected function method(): string
    {
        return 'post';
    }

    protected function route($data): string
    {
        return route('login');
    }
}
```

Don't forget to register your form:

```
\Aero\AccountArea\AccountArea::registerForm(MyForm::class);
```