

Orders

- [How do I block orders?](#)
- [How do I unblock orders?](#)
- [How do I customize the order reference?](#)
- [What is the order item builder?](#)
- [How do I add to the order activity timeline?](#)
- [How do I remove the manufacturers name from the order items name?](#)

How do I block orders?

Orders can be blocked by adding a blocker when some condition is met.

For example, you can prevent potentially fraudulent orders from proceeding to payment by checking the order's email domain and IP address against configured blocked values.

First, define some settings:

```
\Aero\Common\Facades\Settings::group('checkout', static function
(\Aero\Common\Settings\SettingGroup $group) {
    $group->array('blocked_email_domains')->section('Fraud');
    $group->array('blocked_ip_addresses')->section('Fraud');
});
```

An `is_fraudulent` getter can then be macro'ed onto the order model:

```
\Aero\Cart\Models\Order::macro('getIsFraudulentAttribute', function () {
    $orderEmailDomain = Str::after($this->email, '@');

    return in_array($orderEmailDomain, setting('checkout.blocked_email_domains'))
        || in_array($this->ip, setting('checkout.blocked_ip_addresses'));
});
```

Middleware can then be added to the checkout payment page to block the order if fraud has been detected:

```
\Aero\Checkout\Http\Responses\CheckoutPaymentPage::middleware(function
(\Illuminate\Http\Request $request, \Closure $next) {
    $order = \Aero\Cart\Facades\Cart::order();

    if ($order->is_fraudulent) {
        $order->setBlocker('fraud', [
            'reason' => 'We suspect that this order is fraudulent. Please contact our support
team if you have any questions.',
        ]);
    }

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

```
});
```

The checkout payment page can then be extended to set a custom view to display the blocker reason:

```
\Aero\Checkout\Http\Responses\CheckoutPaymentPage::extend(function
(\Aero\Checkout\Http\Responses\CheckoutPaymentPage $page, \Closure $next) {
    $blocker = $page->cart->order()->getBlocker('fraud');

    if ($blocker) {
        $page->setData('blocker', $blocker);
        $page->getData('sections')->put('payment_method', 'acme::error');
    }

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

The `acme::error` view:

```
<div class="payment-method checkout__section">
    <div class="checkout__error">{{ blocker.reason }}</div>
</div>
```

Parameters

Parameter	Type	Description
<code>\$key</code>	<code>string</code>	The key of the blocker to set.
<code>\$value</code>	<code>mixed</code>	The value of the blocker to set.

How do I unblock orders?

You can remove an existing blocker from an order using the `unsetBlocker` method.

For example, you can make it so some customer groups are exempt from being blocked

```
\Aero\Common\Facades\Settings::group('checkout', static function
(\Aero\Common\Settings\SettingGroup $group) {
    $group->eloquent('exempt_customer_groups', \Aero\Account\Models\CustomerGroup::class)-
>multiple()->dontLoadModels()->section('Fraud');
});
```

Middleware can then be added to the checkout payment page to unblock the order for customers of that group:

```
\Aero\Checkout\Http\Responses\CheckoutPaymentPage::middleware(function
(\Illuminate\Http\Request $request, \Closure $next) {
    $order = \Aero\Cart\Facades\Cart::order();

    $customerGroupId = optional($order->customer)->customer_group_id ?? null;

    if (in_array($customerGroupId, setting('checkout.exempt_customer_groups'))) {
        collect($order->blockers)->keys()
            ->each(function (string $key) use ($order) {
                $order->unsetBlocker($key);
            });
    }

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

Parameters

Parameter	Type	Description
<code>\$key</code>	<code>string</code>	The key of the blocker to unset.

How do I customize the order reference?

The order reference generator can be customized to control the format of order references that are generated when an order is created.

For example, to use the order reference prefix followed by the order ID:

```
\Aero\Cart\Models\Order::setReferenceGenerator(function (\Aero\Cart\Models\Order $order) {  
    $prefix = setting('order_reference_prefix');  
  
    return "{$prefix}{$order->getKey()}";  
});
```

The generator receives the order and should return a string to be used as the order reference.

Parameters

Parameter	Type	Description
\$order	\Aero\Cart\Models\Order	The order to generate a reference for.

What is the order item builder?

The order item builder is run whenever a cart item is transformed into an order item. It builds the order item data from the cart item.

For example, you can add a custom attribute to the order item:

```
\Aero\Cart\OrderItemBuilder::extend(function (\Aero\Cart\OrderItemBuilder $builder, \Closure $next) {  
    $cartItem = $builder->item();  
  
    $builder->setAttributes([  
        'acme' => $cartItem->meta->get('acme'),  
    ]);  
  
    return $next($builder);  
});
```

Parameters

Parameter	Type	Description
<code>\$builder</code>	<code>\Aero\Cart\OrderItemBuilder</code>	The order item builder.
<code>\$next</code>	<code>\Closure</code>	The next step in the order item builder.

Methods

Method	Description
<code>item()</code>	Returns the cart item being transformed.
<code>order()</code>	Returns the order that the order item will be added to once transformed.

<code>setAttribute(string \$key, \$value)</code>	Sets an attribute to be added to the order item.
<code>setAttributes(array \$attributes)</code>	Sets attributes to be added to the order item (merges them with existing attributes).
<code>getAttribute(string \$key)</code>	Returns the attribute corresponding to <code>\$key</code> or <code>null</code> if it doesn't exist
<code>getAttributes()</code>	Returns the array of attributes to be set on the order item

How do I add to the order activity timeline?

To add something to the order activity timeline you must create a class that extends

`\Aero\Admin\Activity`:

```
class OrderFlagsEditActivity extends \Aero\Admin\Activity
{
    protected $view = 'admin::activity.edit-flags';

    protected $key = 'order-flags-activity';

    protected $name = 'Order Flag Edits';

    protected $flags;

    public static function create(\Carbon\Carbon $time, ?\Aero\Admin\Models\Admin $admin,
\Illuminate\Support\Collection $flags): self
    {
        return (new self($time))->admin($admin)->flags($flags);
    }

    public function flags(\Illuminate\Support\Collection $flags): self
    {
        $this->flags = $flags;

        return $this;
    }

    public function viewData(): array
    {
        $name = $this->getAdminName();

        return [
            'id' => Str::random(),
```

```

        'time' => $this->getTime(),
        'title' => $this->getTitle($name),
        'admin' => $name,
        'changes' => $this->getChanges(),
    ];
}

protected function getTitle($adminName): string
{
    return $adminName ? "{$adminName} edited this order's flags" : "This order's flags
were edited";
}

protected function getChanges()
{
    return $this->flags->groupBy('state');
}
}

```

And a view to handle rendering the activity:

```

<div class="w-full [ aero-accordion ]">
    <input type="checkbox" id="{{ $id }}Open" aria-hidden="true">

    <div class="flex items-center">
        <label class="aero-accordion-label aero-accordion-label--inline justify-start flex-1"
for="{{ $id }}Open" aria-hidden="true">
            <div class="flex items-center">
                <div class="flex-1">
                    <div class="timeline-title">
                        {{ $title }}
                    </div>
                </div>
            </div>
        </label>

        @include('admin::activity.partials.time', compact('time'))
    </div>

    <div class="visually-hidden">

```

```

        <div>
            {{ $title }}
        </div>
    </div>

<div class="[ aero-accordion-content ]">
    <div class="pl-6">
        @foreach($changes as $state => $stateChanges)
            <div class="py-2">
                <div class="font-medium mb-2">{{ ucfirst($state) }}</div>

                @foreach($stateChanges as $change)
                    <div class="flex items-center mt-1">
                        &bull;&nbsp;
                        @include('admin::icons.flag', [
                            'color' => $change->flag->color
                        ])
                        @if($state === \Aero\Admin\Models\FlagHistory::ADDED)
                            <span>&nbsp;{{ $change->flag->name }} @if($change-
>new_value)({{ $change->new_value }})@endif</span>
                        @elseif($state === \Aero\Admin\Models\FlagHistory::UPDATED)
                            <span>&nbsp;{{ $change->flag->name }} ({{ $change->old_value
?? view('admin::resource-lists.placeholder') }} => {{ $change->new_value }})</span>
                        @else
                            <span>&nbsp;{{ $change->flag->name }} @if($change-
>old_value)({{ $change->old_value }})@endif</span>
                        @endif
                    </div>
                @endforeach
            </div>
        @endforeach
    </div>
</div>
</div>

```

Finally, you must extend `\Aero\Cart\Models\Order` to add this activity:

```

\Aero\Cart\Models\Order::extendActivity(function (\Aero\Cart\Models\Order $order) {
    $order->loadMissing('flagHistoryGroups.admin', 'flagHistoryGroups.histories.flag')-
>flagHistoryGroups->each(function (\Aero\Admin\Models\FlagHistoryGroup $group) use ($order) {

```

```
$order->addActivity(\Aero\Admin\Activities\OrderFlagsEditActivity::create(  
    $group->created_at,  
    $group->admin,  
    $group->histories  
));  
});  
});
```

How do I remove the manufacturers name from the order items name?

By default if a product name doesn't contain the manufacturer name already, the manufacturer name is appended to the front of the product name when converting a cart item to an order item.

If you don't want the manufacturer name appended to the product name you can set the static

`Aero\Cart\Models\OrderItem::$nameContainsManufacturer` boolean to `false`:

```
\Aero\Cart\Models\OrderItem::$nameContainsManufacturer = false;
```