

Settings

- [General](#)
 - [What setting types are available?](#)
- [Model Settings](#)
 - [How do I add model settings to a model?](#)
- [Setting Definitions](#)
 - [How do I add setting definitions to a model?](#)

General

What setting types are available?

As of 15/09/2026, there are ten setting types available:

Type	Description
String	Stores text values.
Boolean	Stores <code>true</code> / <code>false</code> values.
Integer	Stores whole-number values.
Float	Stores decimal values.
Array	Stores multiple values of different types.
Eloquent	Stores eloquent models.
Encrypted	Stores encrypted values.
Date	Stores a date.
Date Range	Stores a start and end date.
Cron	Stores a cron expression.

Base Setting

All setting types inherit the base setting which exposes the following methods.

Method	Description
<code>map(\Closure \$map)</code>	Provide a map to format the setting value prior to retrieval.

<code>default(\$default)</code>	Sets the default value for the setting.
<code>section(string \$section)</code>	Assigns the setting to a specific section.
<code>label(string \$label)</code>	Sets the display label for the setting.
<code>hint(string \$hint)</code>	Adds hint text for the setting.
<code>required()</code>	Marks the setting as required.
<code>addRule(\$rule)</code>	Adds a validation rule to the setting.
<code>addRules(array \$rules)</code>	Adds multiple validation rules to the setting.
<code>removeRule(\$rule)</code>	Removes a validation rule from the setting.
<code>removeRules(array \$rules)</code>	Removes multiple validation rules from the setting.
<code>showIf(string \$key, \$value = true)</code>	Conditionally displays the setting input based on another setting's value.
<code>showWhen(string \$key, \$value = true)</code>	Alias for <code>showIf()</code> .

String Setting

String settings store text values and provide additional methods for controlling how the value is entered.

Method	Description
<code>__construct(string \$key)</code>	Creates a new string setting using the specified <code>\$key</code> .
<code>textarea()</code>	Displays using a <code>textarea</code> instead of a text input.
<code>wyiswyg()</code>	Displays using a <code>wyiswyg</code> editor instead of a text input.

String settings use the [Dropdown Rules](#), [Size Rules](#) & [String Rules](#) traits.

Example Usage

```
\Aero\Common\Facades\Settings::group('acme', function (Aero\Common\Settings\SettingGroup
$group) {
    // Basic
    $group->string('order_reference_prefix')->default('WEB');

    // Textarea
```

```

$group->string('order_notes')->textarea();

// WYSIWYG editor
$group->string('rich_text')->wysiwyg();

// Associative dropdown
$group->string('short_date_format')->associative()
    ->in([
        'D jS M, H:i' => now()->format('D jS M, H:i'),
        'D jS M Y, H:i' => now()->format('D jS M Y, H:i'),
    ])
    ->default('D jS M, H:i')
    ->section('Date & Time');
});

```

Boolean Setting

Boolean settings store `true`/`false` values and provide additional methods for controlling how multiple boolean values are handled.

Method	Description
<code>__construct(string \$key)</code>	Creates a new boolean setting using the specified <code>\$key</code> .
<code>onlyReturnTrueKeys()</code>	Only return keys corresponding to <code>true</code> values (when using <code>multiple()</code>)
<code>onlyReturnFalseKeys()</code>	Only return keys corresponding to <code>false</code> values (when using <code>multiple()</code>)

Boolean settings use the [Can Have Multiple](#) trait.

Example Usage

```

\Aero\Common\Facades\Settings::group('acme', function (Aero\Common\Settings\SettingGroup
$group) {
    // Basic

```

```

$group->boolean('is_enabled')->default(true);

// Multiple values
$group->boolean('outstanding_order_states')
    ->multiple([
        \Aero\Cart\Models\OrderStatus::CANCELED => false,
        \Aero\Cart\Models\OrderStatus::ON_HOLD => true,
        \Aero\Cart\Models\OrderStatus::SUCCESSFUL => true,
        \Aero\Cart\Models\OrderStatus::COMPLETE => false,
        \Aero\Cart\Models\OrderStatus::PROCESSING => true,
        \Aero\Cart\Models\OrderStatus::CLOSED => false,
        \Aero\Cart\Models\OrderStatus::PARTIALLY_DISPATCHED => true,
        \Aero\Cart\Models\OrderStatus::DISPATCHED => false,
        \Aero\Cart\Models\OrderStatus::PARTIALLY_RETURNED => false,
        \Aero\Cart\Models\OrderStatus::RETURNED => false,
    ])
    ->onlyReturnTrueKeys();
});

```

Integer Setting

Integer settings store whole-number values and provide additional methods for constraining the value and defining allowed options.

Method	Description
<code>__construct(string \$key)</code>	Creates a new integer setting using the specified <code>\$key</code> .

Integer settings use the [Dropdown Rules & Size Rules](#) traits.

Example Usage

```

\Aero\Common\Facades\Settings::group('acme', function (Aero\Common\Settings\SettingGroup
$group) {
    // Basic
    $group->integer('failed_attempts_limit')->default(5);

```

```
// Dropdown
$group->integer('per_page')
    ->in([24, 48, 96, 240, 360])
    ->default(24);

});
```

Float Setting

Float settings store decimal values and provide additional methods for constraining the value and defining allowed options.

Method	Description
<code>__construct(string \$key)</code>	Creates a new float setting using the specified <code>\$key</code> .
<code>step(float \$step)</code>	Sets the step increment used by the float setting, uses <code>0.01</code> by default.

Float settings use the [Dropdown Rules](#) & [Size Rules](#) traits.

Example Usage

```
\Aero\Common\Facades\Settings::group('acme', function (Aero\Common\Settings\SettingGroup
$group) {
    // Basic
    $group->float('commission_percentage')->default(1.2)

    // Dropdown
    $group->float('fallback_tax_rate')
        ->in([0.0, 5.0, 12.5, 20.0])
        ->default(20.0);
});
```

Array Setting

Array settings store arrays of values and provide additional methods for defining the structure and validation rules for each item.

Method	Description
<code>__construct(string \$key)</code>	Creates a new array setting using the specified <code>\$key</code> .
<code>associative()</code>	Enables associative mode.
<code>definition(array \$definition)</code>	Defines the structure and validation rules for each array item.

Array settings use the [Size Rules](#) & [Can Sort](#) traits.

Example Usage

```
\Aero\Common\Facades\Settings::group('acme', function (Aero\Common\Settings\SettingGroup
$group) {
    // Basic
    $group->array('allowed_upload_types')
        ->default([
            'image/jpeg',
            'image/png',
            'image/gif',
            'image/svg+xml',
            'image/webp',
        ]);

    // Custom structure (each definition has a key input by default)
    $group->array('form_fields')
        ->definition([
            'type' => ['required', 'string', 'in:text,number'],
            'required' => ['boolean'],
        ])
        ->default([
            'first_name' => [
                'type' => 'text',
                'required' => true,
            ],
            'age' => [
                'type' => 'number',
```

```

        'required' => true,
    ]
});
});

```

Eloquent Setting

Eloquent settings store references to Eloquent models and provide additional methods for searching and sorting the results.

Method	Description
<code>__construct(string \$key, string \$class)</code>	Creates a new eloquent setting using the specified <code>\$key</code> for the model with class <code>\$class</code> .
<code>dontLoadModel()</code>	Disables loading the eloquent model on retrieval, returns the id instead.
<code>dontLoadModels()</code>	Alias for <code>dontLoadModel()</code>
<code>searchField(\$searchField, \$displayField = null)</code>	Configures a single field to use when searching for models, with an optional field for displaying the model name.
<code>searchFields(array \$searchFields, \$displayField = null)</code>	Adds one or more fields to use when searching for models and optionally sets the display field.

Eloquent settings use the [Can Have Multiple](#) & [Can Sort](#) traits.

Example Usage

```

\Aero\Common\Facades\Settings::group('acme', function (Aero\Common\Settings\SettingGroup
$group) {
    // Single
    $group->eloquent('image', \Aero\Common\Models\Image::class);

    // Multiple
    $group->eloquent('vip_customer_groups', \Aero\Account\Models\CustomerGroup::class)-
>multiple();
});

```

Encrypted Setting

Encrypted settings store encrypted string values and provide additional methods for constraining the value and validating its format.

Method	Description
<code>__construct(string \$key)</code>	Creates a new encrypted setting using the specified <code>\$key</code> .

Encrypted settings use the [Size Rules](#) & [String Rules](#) traits.

Example Usage

```
\Aero\Common\Facades\Settings::group('acme', function (Aero\Common\Settings\SettingGroup $group) {  
    $group->encrypted('api_key')->default('secret');  
});
```

Date Setting

Date settings store date values and provide additional methods for validating the date.

Method	Description
<code>__construct(string \$key)</code>	Creates a new date setting using the specified <code>\$key</code> .
<code>beforeToday()</code>	Validates the date is in the past.
<code>beforeDate(\$date)</code>	Validates the date is before some specified date.
<code>afterToday()</code>	Validates the date is in the future.
<code>afterDate(\$date)</code>	Validates the date is after some specified date.

Example Usage

```
\Aero\Common\Facades\Settings::group('acme', function (Aero\Common\Settings\SettingGroup $group) {  
    $group->date('today')->default('today');
```

```
});
```

Date Range Setting

Date range settings store date ranges.

Method	Description
<code>__construct(string \$key)</code>	Creates a new date range setting using the specified <code>\$key</code> .

Example Usage

```
\Aero\Common\Facades\Settings::group('acme', function (Aero\Common\Settings\SettingGroup $group) {  
    $group->dateRange('available_days')->default([  
        'start' => '2026-01-01 07:00:00',  
        'end' => '2026-12-31 15:00:00',  
    ]);  
});
```

Cron Setting

Cron settings store cron expressions for defining scheduled tasks.

Method	Description
<code>__construct(string \$key)</code>	Creates a new cron setting using the specified <code>\$key</code> .

Example Usage

```
\Aero\Common\Facades\Settings::group('acme', function (Aero\Common\Settings\SettingGroup $group) {  
    $group->cron('delete_old_logs')->default('0 0 * * *'); //Every day at midnight  
});
```

Traits

Dropdown Rules

Method	Description
<code>in(\$in)</code>	Defines the allowed values for the dropdown. Accepts an array of options or a closure that resolves to an array.
<code>associative()</code>	Configures the dropdown to use associative array keys as the allowed values.

The order is important: `associative()` should be used before `in()` if required.

Size Rules

Method	Description
<code>between(\$min, \$max)</code>	Validates that the value is between the specified minimum and maximum values.
<code>max(\$size)</code>	Validates that the value does not exceed the specified maximum size.
<code>min(\$size)</code>	Validates that the value is at least the specified minimum size.
<code>size(\$size)</code>	Validates that the value has exactly the specified size.
<code>greaterThan(\$value)</code>	Validates that the value is greater than the specified value.
<code>lessThan(\$value)</code>	Validates that the value is less than the specified value.

String Rules

Method	Description
<code>email(string \$options = 'rfc,dns')</code>	Validates the value is an email using the comma-separated values provided.
<code>uuid()</code>	Validates the value is a uuid.
<code>url()</code>	Validates the value is a url.

<code>ip()</code>	Validates the value is an ip.
<code>startsWith(string \$needle)</code>	Validates the value starts with <code>\$needle</code> .
<code>startsWithOneOfMany(array \$needles)</code>	Validates the value starts with one of many <code>\$needles</code> .

Can Have Multiple

Method	Description
<code>multiple(array \$defaults = [])</code>	Enables multiple-value selection. Optionally sets default values and, when no options are configured, derives the available options from the default array keys.
<code>options(array \$options = [])</code>	Sets the available options.
<code>addMultipleRules(array \$rules)</code>	Adds validation rules, applied to each selected value.
<code>addMultipleRule(\$rule)</code>	Adds a single validation rule, applied to each selected value.
<code>removeMultipleRules(array \$rules)</code>	Removes validation rules from being applied to each selected value.
<code>removeMultipleRule(\$rule)</code>	Removes a single validation rule from being applied to each selected value.

Can Sort

Method	Description
<code>sortable()</code>	Enables sorting for the setting.

Model Settings

How do I add model settings to a model?

Model settings provide a consistent way to store configurable settings against an Eloquent model. There are various steps needed to be followed to setup model settings for a model.

First, add the `\Aero\Common\Traits\CanHaveSettings` to the model and optionally define a `RELATION_KEY`:

```
class Acme extends Model
{
    use \Aero\Common\Traits\CanHaveSettings;

    public const RELATION_KEY = 'acme' //optional

    //...
}
```

If you added a `RELATION_KEY` make sure to add it to the relation morph map.

Then update create/update requests for this model to wrap the existing rules with `::getSettingRules()`:

```
class CreateAcmeRequest extends AeroRequest
{
    public function rules(): array
    {
        return Acme::getSettingRules([
            //...
        ]);
    }
}
```

This adds the validation rules required by the model settings alongside the request's normal validation rules.

Finally, after validating the requests, save the model using `saveSettings()`:

```
$data = $request->validated();  
  
$acme->saveSettings($data['settings'] ?? [], true);
```

For UI support, you must include the model settings partial in your views:

```
@include('admin::settings.model-settings', ['model' => $acme ?? new Acme()])
```

Setting Definitions

How do I add setting definitions to a model?

Setting definitions allow you to define settings via the admin UI.

To add a setting definition for a model, you must add the model type, you can do this using the model class or relation key.

For example:

```
// By class
\Aero\Common\Models\SettingDefinition::addModelType(Acme::class, 'Acme');

// By relation key
\Aero\Common\Models\SettingDefinition::addModelType(Acme::RELATION_KEY, 'Acme');
```

The model you're adding must support model settings for setting definitions to work.