

How can I add a custom listing rule?

To add a custom listing rule you need to create a class that extends

```
\Aero\Catalog\Rules\ListingRule\AttributedWithRule.
```

For example we will add a rule that checks if the listing has specific attributes.

First, we need to index the attribute ids so that they can be queried easily:

```
\Aero\Search\Elastic\Documents\ListingDocument::add('search-result', function
(\Aero\Search\Elastic\Documents\ListingDocument $document) {
    return [
        'attribute_ids' => $document->getModel()->attributes->map(function
(\Aero\Catalog\Models\Attribute $attribute) {
            return (string) $attribute->getKey();
        }),
    ];
});
```

Now, we can make the rule:

```
class AttributedWithRule extends \Aero\Catalog\Rules\ListingRule
{
    public const KEY = 'attributes';

    public string $name = 'have one of these attributes';

    public string $view = 'acme::attributes';

    protected array $options = [
        'attributes' => [],
    ];

    public function viewData(array $context): array
    {
```

```

return [
    'attributes' => collect($this->options['attributes'] ?? [])
        ->flatten()
        ->map(function ($id) use ($context) {
            return $context['attributes'][$id] ?? null;
        })
        ->toArray(),
];
}

public function apply(\Aero\Search\Contracts\SearchBuilder $builder):
?\Elastica\Query\BoolQuery
{
    $attributeIds = $this->options['attributes'] ?? [];

    if (! is_array($attributeIds)) {
        return null;
    }

    return (new BoolQuery())->addMust(
        new \Elastica\Query\Terms('search-result.attribute_ids', $attributeIds),
    );
}

public function optionRules(): array
{
    return [
        'attributes' => 'required|array',
        'attributes.*' => 'required|distinct|int|exists:attributes,id',
    ];
}
}

```

For the UI to work we must build the context for our rule:

```

\Aero\Admin\Pipelines\ListingRulesContextBuilder::extend(function
(\Aero\Admin\Pipelines\ListingRulesContextBuilder $pipeline, \Closure $next) {
    $attributeIds = $pipeline->rules
        ->where('type', \Aerocargo\Acme\AttributedWithRule::KEY)
        ->flatMap(function (\Aero\Catalog\Models\ListingRule $rule) {

```

```

        return collect($rule->options['attributes'] ?? [])->flatten()->toArray();
    })
    ->unique();

$pipeline->data[\Aerocargo\Acme\AttributedWithRule::KEY] = [
    'attributes' => $attributeIds->isNotEmpty()
        ? Attribute::query()
            ->select('id', 'name')
            ->whereIn('id', $attributeIds)->get()
            ->mapWithKeys(function (\Aero\Catalog\Models\Attribute $attribute) {
                $id = $attribute->getKey();

                return [
                    $id => [
                        'value' => (string) $id,
                        'name' => $attribute->getAttribute('name'),
                    ]
                ];
            })
            ->toArray()
        : [],
];

return $next($pipeline);
});

```

And add the view:

```

<template v-for="attribute in builder.rule.view_data.attributes ?? []">
    <input type="hidden" :name="{{ $name }}" + '[attributes][]' :value="attribute.value">
</template>

<label :for="{{ $id }}" + '-attributes-option">Attributes</label>
<searchable-select :id="{{ $id }}" + '-attributes-option' class="mt-2 w-full"
    :input-name="'rules[' + builder.group_index + '][' + builder.rule_index +
    '][view_data][attributes]'"
    url="{{ route('admin.catalog.attributes.search') }}" track-by="value"
    label="name" :multiple="true"
    v-model="builder.rule.view_data.attributes"></searchable-select>

```

Revision #1

Created 2026-09-24 13:02:29 UTC by Ethan Payne

Updated 2026-09-24 13:13:39 UTC by Ethan Payne