

Validation

Validation

Anonymous · 20/10/2026

Validation

`Vault` and `Store` have a schema, and they keep their data valid: a change that breaks the schema is rejected, and the container stays as it was. `Bag` and `Journal` have no schema and accept everything.

When the Data Is Checked

Operation	What is checked
The constructor	The data against the schema.
<code>set()</code> and <code>\$container['key'] = ...</code>	The data that would result from the change.
<code>clear()</code> and <code>unset(\$container['key'])</code>	The data that would result from the change.
<code>setSchema()</code>	The stored data against the new schema.
<code>validate()</code>	The stored data against the current schema.

The data is also **normalized**: the defaults are applied and, in a `Vault`, the normalizers run. What is stored after a `set()` is the resolved data, as after the constructor.

```
use Derafu\Container\Vault;

$vault = new Vault(
    ['host' => 'imap.example.com'],
    [
        'host' => ['types' => 'string', 'required' => true],
        'port' => ['types' => 'int', 'default' => 993],
    ]
);

$vault->set('port', 'abc');
// InvalidOptionsException: The option "port" with value "abc" is expected to be
// of type "int", but is of type "string".

$vault->requireInt('port'); // 993: nothing changed.
```

The change is made on a copy and the container takes it only if it is valid, so there is nothing to undo

and nothing is silenced: the exception always reaches the caller.

Removing Keys

`clear()` is checked like any other change. Removing a key with a default gives it back its default; removing a required key is rejected:

```
$vault->clear('port'); // Fine: port is 993 again.  
$vault->clear('host'); // MissingOptionsException: The required option "host" is  
missing.  
$vault->clear();      // The same: an empty container does not satisfy the schema.
```

Changing the Schema

`setSchema()` checks (and normalizes) the data that the container already has. If it does not comply, neither the schema nor the data change.

```
$vault = new Vault(['host' => 'a.example.com']); // No schema: anything goes.  
  
$vault->setSchema([  
    'host' => ['types' => 'string', 'required' => true],  
    'port' => ['types' => 'int', 'default' => 993],  
]);  
  
$vault->requireInt('port'); // 993: the default was applied to the stored data.
```

The Schema of a Vault

The schema is an array: each key is a field, and its value says how the field is validated. The keys of the field are:

Key	Meaning
types	A type, or an array of types, that the value can have: <code>string</code> , <code>int</code> , <code>float</code> , <code>bool</code> , <code>array</code> , <code>null</code> , <code>object</code> , a class name... (the types of <code>OptionsResolver</code>).
required	<code>true</code> when the field has to be present.
choices	The values that are accepted.
default	The value of the field when it is not given.
normalizer	A closure fn (<code>Options \$options</code> , <code>mixed \$value</code>): <code>mixed</code> that converts the value after it is validated.
schema	The schema of the fields inside an array field. It nests as deep as needed.

```
$vault = new Vault(  

```

```
[ 'price' => '12.50', 'status' => 'active', 'user' => [ 'name' => 'Ada' ] ],
[
  'price' => [
    'types' => [ 'string', 'float' ],
    'required' => true,
    'normalizer' => fn ($options, $value) => (float) $value,
  ],
  'status' => [ 'types' => 'string', 'choices' => [ 'active', 'inactive' ] ],
  'user' => [
    'types' => 'array',
    'schema' => [
      'name' => [ 'types' => 'string', 'required' => true ],
      'age' => [ 'types' => 'int', 'default' => 18 ],
    ],
  ],
],
];

$vault->requireFloat('price');    // 12.5
$vault->requireInt('user.age');   // 18
$vault->set('status', 'other');   // Rejected: not one of the choices.
```

A field whose configuration is not valid (a `types` that is not a string or an array of strings, a field that is not an array...) throws `TranslatableInvalidArgumentException` when the schema is used.

Keys the Schema Does Not Define

By default a key that is not in the schema is rejected:

```
$vault->set('endpoint', 'x');
// UndefinedOptionsException: The option "endpoint" does not exist.
```

The policy can be changed in three places, from the most general to the most specific:

1. The constructor: `new Vault($data, $schema, allowUndefinedKeys: true)`. The `Vault` keeps it and uses it in `set()`, `clear()`, `setSchema()` and `validate()`.
2. The key `__allowUndefinedKeys` in a schema (`true` or `false`), which overrides the constructor for that level.
3. `validate(true)` or `validate(false)`, only for that call.

A nested schema inherits the policy of the level above and can override it with its own `__allowUndefinedKeys`:

```
$vault = new Vault($data, [
  '__allowUndefinedKeys' => true,           // The root accepts more keys...
  'user' => [
    'schema' => [
```

```

        '__allowUndefinedKeys' => false,           // ...but `user` does not.
        'name' => ['types' => 'string'],
    ],
],
});

```

The Exceptions of a Vault

The errors of the data are the exceptions of `symfony/options-resolver` (`InvalidOptionsException`, `MissingOptionsException`, `UndefinedOptionsException`), which extend `InvalidArgumentException`. They are not translated. The errors of the schema itself are `TranslatableInvalidArgumentException`.

The Schema of a Store

A Store is validated with JSON Schema (draft 2020-12). You can give the schema as an array; `$schema` and `type: object` are added if they are missing.

```

use Derafu\Container\Store;

$store = new Store(
    ['name' => 'Ada'],
    [
        'required' => ['name'],
        'properties' => [
            'name' => ['type' => 'string'],
            'lang' => ['type' => 'string', 'default' => 'es'],
        ],
    ],
);

$store->get('lang');           // "es": the default of the property.
$store->set('name', 5);         // Rejected.
$store->clear('name');         // Rejected: `name` is required.

```

The errors are `TranslatableInvalidArgumentException`, with every problem found and where it is:

```

Failed to validate the data against the JSON schema. The data (integer) must
match the type: string in /name.

```

JSON Schema defaults are applied to the data, but the data must be an object at its root: an empty container is validated as `{}`.

What Is Not Checked

The validation guards the methods of the container. These ways in do not go through it:

- `collection()` returns the live `ArrayCollection`. Changing it changes the container without validating. It is a deliberate way for the code that needs to work with the collection directly.
- An object stored in the container can be changed by whoever holds it.

`validate()` checks the stored data again, for when you used one of those:

```
$vault->collection()->set('port', 'abc');    // Not checked.  
$vault->validate();                          // InvalidOptionsException.
```

`validate()` does not change the data. In a `Vault` it takes an optional argument to override the policy of undefined keys for that call.

Last updated on 20/10/2026