

Value Types

Compose Derafu Value types for optional and required array reads

Anonymous · 26/10/2026

Value Types in Array Readers

ArrayReader accepts `Derafu\Value\ValueType<T>` for generic reads and collections. The type system belongs to [Derafu Value](#), including primitive checks, explicit conversions, dates, backed enums, unions, nullable types and custom types.

```
use Derafu\Array\ArrayReader;
use Derafu\Value\ValueType;

$reader = ArrayReader::fromArray([
    'quantities' => [3 => '12.0', 8 => '2'],
    'mode' => 'fast',
], 'settings.php');

$reader->requireArrayOf('quantities', ValueType::toInt());
// [3 => 12, 8 => 2]

$reader->get('mode', ValueType::literal('fast', 'safe'), 'safe');
// 'fast'
```

Validation and Conversion

`ValueType::int()` checks an existing integer; `ValueType::toInt()` requests conversion through `ValueCaster`. Built-in getters delegate to these same types: `requireInt()` is strict, while `requireAsInt()` converts explicitly. PHPStan follows the generic return type through these operations.

Arrays preserve arbitrary integer/string keys. Maps require string keys; lists require consecutive integer keys starting at zero. Empty collections are accepted. A type such as `mapOf()` checks every value, not a schema of required field names.

Nullability and Presence

A nullable type accepts null when parsed directly, but reader methods apply their own presence rules first: optional reads return the default for a missing or null key, and required reads reject both. `hasKey()` distinguishes a present null key from a missing key.

```
use Derafu\Array\ArrayReader;
use Derafu\Value\ValueType;

$reader = ArrayReader::fromArray(['count' => null]);
$reader->get('count', ValueType::toInt()->nullable(), 0); // 0
$reader->hasKey('count'); // true
$reader->hasValue('count'); // false
```

Backed Enums

```
use Derafu\Array\ArrayReader;
use Derafu\Value\ValueType;

enum Status: string
{
    case Pending = 'pending';
    case Approved = 'approved';
}

$reader = ArrayReader::fromArray(['status' => 'approved']);
$reader->require('status', ValueType::backedEnum(Status::class));
// Status::Approved
```

Backing values use their exact int/string type, without scalar coercion. See [ValueType](#) for enum factories, dates, custom type callbacks and union order.

Error Context

ValueType reports the full nested path and the expected element type. ArrayReader adapts a value type mismatch to Derafu\Array\Exception\TypeMismatchException, preserving the Value exception as its cause. The reader keeps its own translated key-oriented messages and source context. See [Translations](#).

Last updated on 26/10/2026