# Drupal 8 configuration schema cheat sheet

1.3 - Jan 20. 2015.

Configuration schema in Drupal 8 is used to describe the structure of configuration files. It is then applied to:

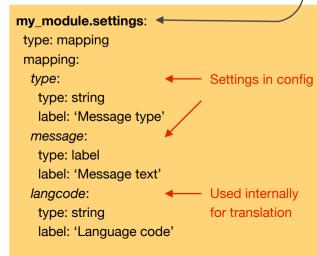
- Typecast configuration to ensure type consistency (see StorableConfigBase::castValue())
- Automated persistence of configuration entity properties (see ConfigEntityBase::toArray())
- Automated generation of the configuration translation user interface (see the core module)

### A simple example

#### config/install/my\_module.settings.yml

type: warning message: 'Hello!' langcode: en

#### config/schema/my\_module.schema.yml



### Basic schema types

Core provides the following data types. Contributed modules may define new base types. More are defined in *core.data\_types.schema.yml*.

### Scalar types

boolean

integer

float

string

email

uri

### Subtyping

All of configuration schema is basically subtyping from existing types. The simple example earlier is subtyping *mapping* with defined keys that have their own types.

#### List types

mapping: known keys sequence: unknown keys

#### **Common subtypes**

label: short & translatable text: long & translatable Types *route*, *filter*, *mail*, etc. are provided for common complex Drupal data structures.

The only difference

between the two list

types is in mappings

you know the keys.

### Dynamic type with [%parent]

Exact types may not be known ahead of time and may depend on the data. Schema allows to define types based on the data as well. Let's say the type of *message* may depend on the *type* value: either a list of messages or a simple warning message. Let's use *'multiple'* for the list case and keep *'warning'* for the single line message.

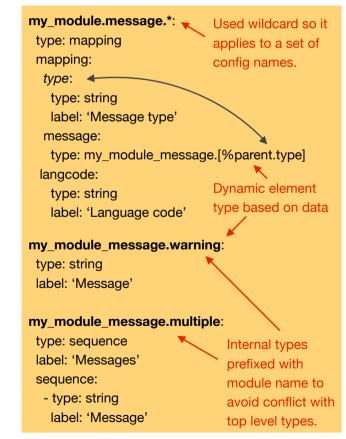
### $config/install/my\_module.message.single.yml$

type: warning message: 'Hello!' langcode: en

### config/install/my\_module.message.multiple.yml

type: multiple message: - 'Hello!' - 'Hi!' langcode: en

#### config/schema/my\_module.schema.yml



Chaining is possible as %parent.%parent.type, etc.

### Dynamic type with [type]

If the data to vary your type by is under the data to be typed, that is when [type] becomes useful.

config/install/my\_module.message.single.yml

message:

type: warning

value: 'Hello!'

langcode: en

#### config/install/my\_module.message.multiple.yml

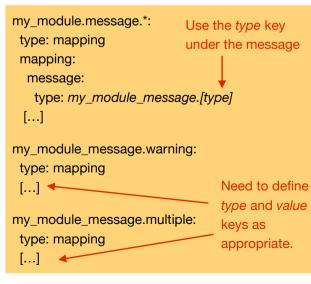
#### message:

type: multiple

value:

- 'Hello!'
- 'Hi!'
- langcode: en

### config/schema/my\_module.schema.yml



You may also define a **my\_module\_message\_base** base type that includes common keys like 'type' and extend from that with any custom keys per type.

# Dynamic type with [%key]

config/install/my\_module.messages.yml

messages:	
'single:1': 'Hello!'	
'single:2': 'Hi!'	bitrary message lis
'multiple:1':	
- 'Good morning!'	
- 'Good night!'	
langcode: en	

This is now a list of arbitrary message elements.

### config/schema/my\_module.schema.yml

### my module.messages: type: mapping mapping: messages: Type is in the key as type: sequence prefix, e.g. 'single:1' label: 'Messages' sequence: - type: my module message.[%key] langcode: type: string label: 'Language code' my module message.single:\*: type: string label: 'Message' my module message.multiple:\*: type: sequence label: 'Messages' Wildcard to match prefix. sequence:

- type: string

label: 'Message'

## Schema debugging

To debug configuration schemas use the Configuration Inspector module (<u>http://drupal.org/project/</u> <u>config\_inspector</u>) which helps you find schema mismatches with active configuration and inspect how your schema is applied to your configuration.

# Schema testing

- All TestBase deriving tests in core now use \$strictConfigSchema = TRUE which results in strict scheme adherence testing for all configuration saved. Only opt out of this if you really need to. Your schema should match your data and pass this test.
- Use SchemaCheckTestTrait in your test to check for specific config files only.

### More documentation

See <u>https://www.drupal.org/node/1905070</u> for even more configuration schema documentation and examples.

### Issues?

- For issues with core configuration schemas, tag them with 'Configuration schema' and 'Configuration system' and pick the appropriate module as component.
- For issues with the configuration schema system itself, use the 'configuration system' component and also tag with 'Configuration schema'.