Property validation & state

Validation

There are several pieces of data validation information that may be returned. It is possible to ignore these options as all fields will be validated when the data is submitted to the server, but you will provide a better user experience if you provide immediate feedback to the user.

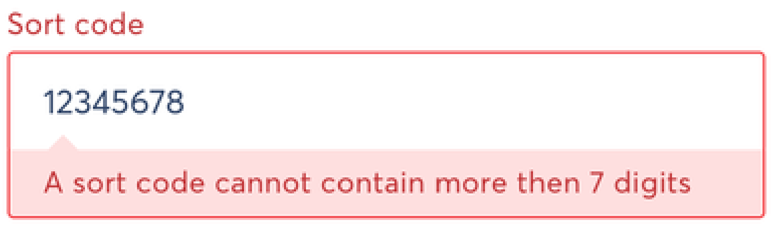
- **required** - indicates whether the field is required or optional
- **minimum/maximum** - applies to numeric and date fields
- **minLength/maxLength** - applies to string fields
- **pattern** - a regular expression pattern that applies to string fields

A field may contain custom validation messages, if present these will appear as a map from the validation type to a localised string. Any of the validation types could appear in this map.

It is safe to ignore this information, and use a generic string for all fields e.g. "The value is too short". However, better feedback may be provided to customers if you use custom messages when available.

```
sortCode: {
  type: "string",
  minLength: 6,
  maxLength: 7,
  validationMessages: {
    minLength: "A sort code must contain at least 6 digits",
    maxLength: "A sort code cannot contain more then 7 digits"
  }
}
```

When a user enters a value that fails validation, we would show the relevant validation message.



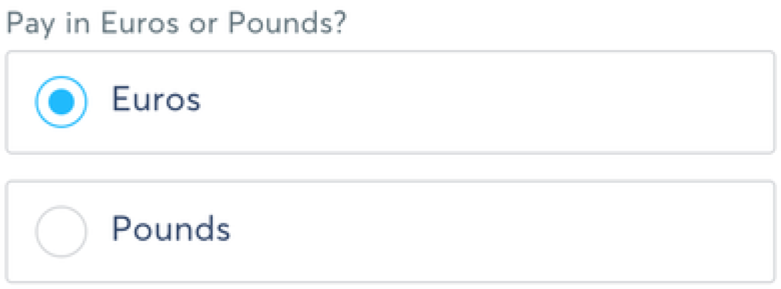
A sort code cannot contain more then 7 digits

Default values

We may include a ‘default’ property. It’s expected that you will pre set the field to the associated value when supplied. This is especially useful when choosing from a set of values:

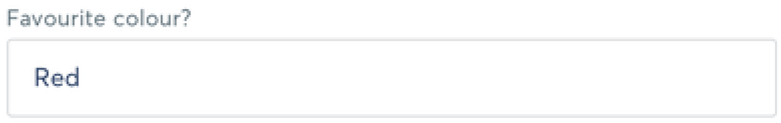
```
{
  type: "string",
  title: "Pay in Euros or Pounds?",
  default: "EUR",
  values: [{
    value: "EUR",
    label: "Euros,
  },{
    value: "GBP",
    label: "Pounds
  }]
}
```

Which may render as:



However, you may also see this on other fields:

```
{
  type: "string",
  title: "Favourite colour?",
  default: "Red"
}
```



Field states

We may suggest rendering the field in one of two states: disabled or hidden. For example, if our API expected a fixed value that was not customisable by the user we might supply disabled: true along with a default value. Or if the value can only be one enumerated value, and it is not relevant for the user to see, we might supply:

```
{
  enum: [‘my-value’],
  default: ‘my-value’,
  hidden: true
}
```