

# Translation Manager

## including a new Text-based property in translations:

Out-of-the-box, Translation Manager will recognise all core of text-based properties and a large number of the popular package based properties available for Umbraco.

However, if you have developed your own custom properties or installed a package that Translation Manager doesn't recognise you can extend the list of text properties it understands by adding them to the `translations.config` file within your site's `Config` folder.

## Text / Html only Properties

The Mappers / Text config allows for the configuration of simple property editors that only store a simple text string or HTML as their value.

```
<mitters>
  <text>
    <text>My.Boxed.Content,My.Sample.Content</text>
    <html>My.Boxed.Content</html>
  </text>
</mitters>
```

In this example to property editors `My.Boxed.Content` and `My.Sample.Content` both contain a text-based value and will be interpreted as text by Translation Manager, here `My.BoxedContent` also contains HTML (i.e is a rich text control) so Translation Manager will use the config to treat the text as HTML during translation.

## Complex Properties

If you have a property editor that stores its data in a more complex way you can still extend Translation manager to parse and recognise the text within the data. For this, there are two options

### 1: Using the Custom JSON Mapper

If your property editor stores it's values within a JSON object - you can use the config file to tell Translation Manager how to interpret your properties data.

If for example, you have a property that stores its data in the following format

```
{
  "Key", "some-key-value",
  "Title", "my property title",
  "Content", "<p>some rich text content</p>"
}
```

Then you could use the config options (below) to define the Title and Content fields as text-based.

Within the mappers section of the config file you can define this using the <custom> or <grid> config nodes (see below for details of how the grid naming works)

```
<mappers>
  <custom>
    <config alias="My.Custom.Property">
      <properties>
        <property name="Title" alias="Umbraco.Textbox" />
        <property name="Content" alias="Umbraco.TinyMCEv3" />
      </properties>
    </config>
  </custom>
</mappers>
```

This config tells translation manager to treat all My .Custom, Property values as JSON and extract the title and content values when creating and importing translations.

## 2: Write a custom ValueMapper

The config file should let you configure most non-complex property editors without the need to resort to code, however if you are storing complex data or nesting other property editors within your own editor then you can write your own mapper to with these values

A class implementing the `IValueMapper` interface, will be discovered by translation manager and used where the editor aliases match that of a given property editor.

We have some example code to help you should you need to write your own mapper:

- [Example Stacked Content Value Mapper](#) (Included in Translation Manager)
- [Example for RJP.MultiURL Picker](#)

## Help! I don't know what format the control is storing data in?

If you didn't develop the property editor, then it's highly likely you don't know what format it is storing its data in.

The quickest way to find out what format things are stored in is to look in the `umbraco.config` file within the `app_data` folder.

This file is the Umbraco cache on disk, and you should not edit it - but you can open it up and quickly search it for a property name you know is using the required data type and you will be able to see the format of the data you are looking for.

If your value is storing something other than text you will likely see a piece of JSON looking something like this:

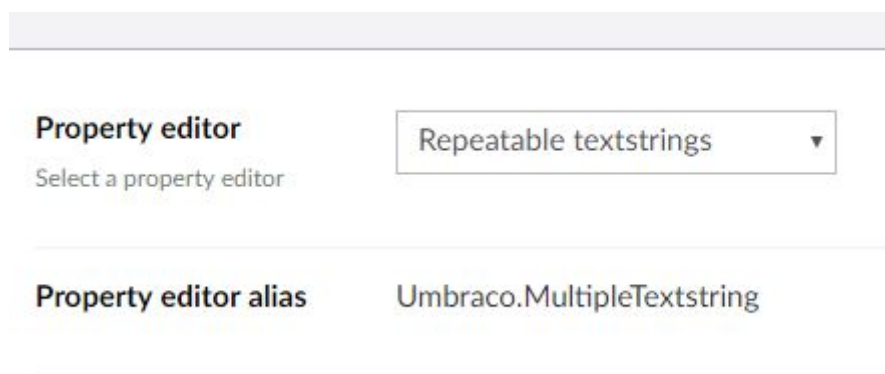
```
"value": {
  "key": "Main Translation set thing",
  "borderColor": "#EEFFEE",
  "definition": "<p>Some text will need translating</p>"
},
```

*In this case you would use this to build a config similar to the one in the first part of this document.*

## Property editor naming

For the most part, when looking for property editors, Translation Manager will use the Property Editor Alias set within the property editor - this is for example `Umbraco.TextString` for textboxes, and `Jumoo.StyledText` for the Styled Textbox property editor.

When you have a custom property editor you should use your property editor alias (alias in the package.manifest file) as the value in the configuration. This can also be seen within Umbraco when looking at the data type in the developer section



The screenshot shows a configuration interface for a property editor. It consists of two rows, each with a label on the left and a value on the right. The first row is labeled "Property editor" and has a dropdown menu with the text "Repeatable textstrings" and a downward arrow. Below the label "Property editor" is the text "Select a property editor". The second row is labeled "Property editor alias" and has the text "Umbraco.MultipleTextstring".

## Grid Names

When looking for custom editors within the Grid property editor, translation manager will use the folder the view is stored in, so for example if your grid editor contains the following config :

```
"gridEditors": [{
  "name": "Boxed Content",
  "alias": "BoxedContent",
  "view": "~/App_Plugins/BoxedContent/Editor/editor.html",
  "render":
  "~/App_Plugins/BoxedContent/Website/boxedcontent.cshtml",
  "icon": "icon-pushpin"
}],
```

The editor alias for translation manager will be `BoxedContent` - which is the alias name, this is the name you should use for all configuration and custom value mappers.