Overview

This page describes the options that are available when creating `template.json` files.

When using the Template Engine the design is to allow you to create a template which is still a "runnable project" A "runnable project" is a project that can be executed as a normal project can. Instead of updating your source files to be tokenized you define replacements, and other processing, mostly in an external file, the `template.json` file. This requires that a few changes have to be done on the fly during the preparation of the package, and in the `template.json` we define all the operations needed to create a well working package. You can find a sample of a complex `template.json` file here Sample.

Category	Description
Package Definition	Metadata of the package
Content Manipulation	Configuration of content changes in source files
Output Management	Configuration of the final output

Package Definition

Name	Description
identity	A unique name for this template
author	The author of the template
classifications	Zero or more characteristics of the template which may be used in search. In this field you define the values shown as Tags in `dotnet new`
name	The name for the template. This is displayed as the template name when using `dotnet new`
groupIdentity	The ID of the group this template belongs to. When combined with the `tags`section, this allows multiple templates to be displayed as one, with the the decision for which one to use being presented as a choice in each one of the pivot categories (keys) Details

Name	Description
tags	You can use tags to improve the metadata of your project. To specify the project language you have to add the tag `language` Details, and if you want to make your project searchable via the `dotnet new --type` command you have to use the tag `type`
shortName	A default shorthand for selecting the template (applies to environments where the template name is specified by the user - not selected via a GUI). this is the name shown as Short Name in `dotnet new` list of templates, and is the name to use command list to run this template
postActions	Enables actions to be performed after the project is created. See https://github.com/dotnet/templating/wiki/Post-Action-Registry

Content Manipulation

Name	Description	Default
sources	The set of mappings in the template content to user directories. It's defined as any array of Source	If not specified, an implicit source is created with `"source": "./"` and `"target": "./"`, all other properties in the source are set to their defaults
guids	A list of guids which appear in the template source and should be replaced in the template output. For each guid listed, a replacement guid is generated, and replaces all occurrences of the source guid in the output	

Name	Description	Default
symbols	See Below	

Source Definition

Name	Description	Default
source	The path in the template content (relative to the directory containing the .template.config folder) that should be processed	`./`
target	The path (relative to the directory the user has specified) that content should be written to	`./`
include	The set of globbing patterns indicating the content to process in the path referred to by `source`	["**/*"]
exclude	The set of globbing patterns indicating the content that was included by sources.include that should not be processed	`[ "**/[Bb]in/**", "**/[Oo]bj/**", ".template.config/**/*", "**/*.filelist", "**/*.user", "**/*.lock.json" ]`
copyOnly	The set of globbing patterns indicating the content that was included by `include`, that hasn't been excluded by sources.exclude that should be placed in the user's directory without modification	`[ "**/node_modules/**/*" ]`

Name	Description	Default
rename	The set of explicit renames to perform. Each key is a path to a file in the source, each value is a path to the target location - only the values will be evaluated with the information the user supplies	
condition	A Boolean-evaluable condition to indicate if the sources configuration should be included or ignored. If the condition evaluates to `true` or is not provided, the sources config will be used for creating the template. If it evaluates to `false`, the sources config will be ignored	
modifiers	A list of additional source information which gets added to the top-level source information, based on evaluation the corresponding source.modifiers.condition. Is defined as an array of Modifier	

Modifier Definition

Name	Description
condition	A Boolean-evaluable condition to indicate if the sources.modifiers instance should be included or ignored. If the condition evaluates to true or is not provided, the sources.modifiers instance will be used for creating the template. If it evaluates to false, the sources.modifiers config will be ignored.
include	Include configuration specific to this sources.modifiers instance, contingent on the corresponding sources.modifiers.condition. See sources.include for more info.
exclude	Exclude configuration specific to this sources.modifiers instance, contingent on the corresponding sources.modifiers.condition. See sources.exclude for more info
copyOnly	CopyOnly configuration specific to this sources.modifiers instance, contingent on the corresponding sources.modifiers.condition. See sources.copyonly for more info

Symbols Introduction The symbols section defines variables and their values, the values may be the defined in terms of other symbols. When a defined symbol name is encountered anywhere in the template definition, it is replaced by the value defined in this configuration. The symbols configuration is a collection of key-value pairs. The keys are the symbol names, and the value contains key-value-pair configuration information on how to assign the symbol a value.

Name	Description	Applies to
type	Defines the high level configuration of symbol. The valid types are: `parameter`, `computed`, and `generated`. The type indicates how the configuration is processed to generate the symbol value: `Parameter`: A symbol for which the config provides literal and/or default values `Computed`: A symbol for which the config provides a Boolean predicate whose evaluation result is the computed symbol result `Generated`: A symbol whose value gets computed by a built-in symbol value generator. See Details	
dataType	Indicates limitations on the valid values a symbol may be assigned. At this point, the only valid datatype is “choice”, which also requires providing symbols.choices configuration for the symbol	`type=parameter`
defaultValue	The value assigned to the symbol if no value for it is provided by the user or host	`type=parameter`
binding		
replaces	Text value that will be replaced	
fileRename	The fileRename element defines the filenames which will be replaced	

Name	Description	Applies to
description	Human readable text describing the meaning of the symbol. This has no effect on template generation	
isRequired	Indicates if the parameter is required or not.	Only used when `type` is set to `parameter`
choices	List of available choices.	Only used when `type` is `choice`
value	Value for replacement	`type=computed`

There are 3 places from which a symbol can acquire its value:

- Template configuration (this).
- Host provided values, which override template configuration values. For example, the host may provide the operating system kind as "Linux", overriding a template default value of "Windows".
- User provided values, which override host & template values. The manner in which these are provided to the template generation broker is specific to the broker. This may be via additional configuration files, command line parameters, inputs to a UI, etc.

Output Management

Name	Description	Default
sourceName	The name in the source tree to replace with the name the user specifies. The value to be replaced with can be given using the `-n --name` options while running a template. The template engine will look for any occurrance of the name present in the config file and replace it in file names and file contents. If no name is specified by the host, the current directory is used.	

Name	Description	Default
placeholderFilename	A filename that will be completely ignored expect to indicate that its containing directory should be copied. This allows creation of empty directory in the created template, by having a corresponding source directory containing just the placeholder file. Completely empty directories are ignored.	If not specified, a default value of `"-.-"` is used.
primaryOutputs	A list of important output paths created during template generation. These paths need to be added to the newly created project at the end of template creation. It's defined as an array of Primary Output	

Primary Output Definition

Name	Description
path	One instance of a primary output path.
condition	If the condition evaluates to true, the corresponding path will be included in the output list. If false, the path is ignored. If no condition is provided for a path, the condition is defaulted to true.

Creation of Multiple Templates shown as one

TODO: Coming soon