fomod Documentation

Release 5.x

Daniel Nunes and the fomod team

Dec 08, 2017

Contents

fomod is game-agnostic format for mod installers, written in xml.

This documentation contains both the tutorial, for beginners, with the most widely used options, and a collection of conventions to make some specifics clearer for both *fomod* authors and installer authors.

This documentation was written for the `5.x` major releases.

CHAPTER 1

Contents

1.1 A fomod Tutorial

At the end of each section there will be a link to an example package where you can see all that was discussed so far.

Let's jump right in - we have finished our mod and we need to provide an installer.

1.1.1 A Simple Installer

We'll start with a simple example, here's how our package looks:

```
.
- example.plugin
- readme.txt
```

Don't forget that the . (dot) simbolizes our current directory.

So we need to install our **example.plugin** to wherever plugins are installed for this game. We don't really care where exactly that is, we'll leave that to the actual installer to figure out. From now on, let's call this destination folder, **dest**.

We start by creating a folder named **fomod** and then creating two files under it named **info.xml** and **ModuleConfig.xml**. Your package should now look like:

```
.
- fomod
|   - info.xml
|   - ModuleConfig.xml
- example.plugin
- readme.txt
```

Start with **info.xml**. You could type this in and be done with it:

```xml
<fomod/>
```

... but that's not really helpful, is it? The purpose of the **info.xml** file is to provide extra metadata for your package, so other people and apps understand what it is about. So let's fill it in properly:

```xml
<fomod>
    <Name>Example Mod</Name>

    <Author>Example Author</Author>

    <Version MachineVersion="1.2.3">
        1.2.3
    </Version>

    <Description>
        This is an example mod.
    </Description>

    <Website>
        https://example.website.com/example-mod
    </Website>
</fomod>
```

See? It didn't hurt and now everyone else knows a little more about our mod! It should be pretty much self-explanatory but if you need a reminder on xml feel free to pause here and look at W3Schools.

Moving on! The **ModuleConfig.xml** is where the magic happens. Coincidentally, it's also the more complex and mind-numbing of the two. So we'll start slow and build our way through all the options. At least the more useful ones.

```xml
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://qconsulting.ca/fo3/ModConfig5.0.xsd">

    <moduleName>Example Mod</moduleName>

    <requiredInstallFiles>
        <file source="example.plugin"/>
    </requiredInstallFiles>

</config>
```

Ok, bit of a mouthful. The *config* tag has the url of the schema as an attribute and it serves as the root of the entire tree. *moduleName* should be pretty much self-explanatory, no?

Now, *requiredInstallFiles*. This tag serves as a root to any files and folders that are ALWAYS installed with your mod. Simple enough. That's all we want for now. The *file* tag under it specifies what to install. The attribute *source* says where the source file will be found. If you needed to install a folder, instead of listing all files in that folder you could use the *folder* tag, it has exactly the same attributes as *file*.

And that's it. We've just made a tiny installer that will successfully install **example.plugin** for our users.

Example 01

1.1.2 Dependencies Network

Right, our installer is a little too simple. Let's say you added a few more things to your plugin, that depended on another plugin. Why waste time reiventing the wheel?

So now you need to make sure the other mod is installed before your own or it won't work. Let's say our plugin depends on **depend1.plugin**:

```xml
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://qconsulting.ca/fo3/ModConfig5.0.xsd">

    <moduleName>Example Mod</moduleName>

    <moduleDependencies operator="And">
        <fileDependency file="depend1.plugin" state="Active"/>
    </moduleDependencies>

    <requiredInstallFiles>
        <file source="example.plugin"/>
    </requiredInstallFiles>

</config>
```

(Pay attention to the order of the tags! It's important!)

moduleDependencies lists all the dependencies our mod needs fulfilled. It is the first thing the actual installer will check, even before installing the files in *requiredInstallFiles*. This dependency list is actually a shared format (meaning other tags will follow the same rules, even if their tag is different), so we'll refer back here whenever another shows up.

The *operator* attribute shows how the dependencies will be resolved:

- "And", every single dependency needs to be met
- "Or", at least one dependency needs to be met

fileDependency, much like the *file* tag, specifies a file, which in this case needs to exist in the **dest** folder. The *file* attribute is, unsurprisingly, the file to depend on, and *state* is which state the file can be in ("Active", "Inactive" and "Missing").

And that's it, you now successfully depen... Awww shucks. You forgot another dependency!

You also depend on another mod, but here the author was a bit messy. He changed the name of the installed file when he updated the version! You should never, ever, do this, but not everyone is as amazing, beautiful and articulate as we are.

So now you depend on another two files, **depend2v1.plugin** and **depend2v2.plugin**. But your mod works with both, so you don't really care which the user has installed and you can't put both under the "And" operator since the user will only have one of them installed. Now we enter the domain of nested dependencies:

```xml
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://qconsulting.ca/fo3/ModConfig5.0.xsd">

    <moduleName>Example Mod</moduleName>

    <moduleDependencies operator="And">
        <fileDependency file="depend1.plugin" state="Active"/>
        <dependencies operator="Or">
            <fileDependency file="depend2v1.plugin" state="Active"/>
            <fileDependency file="depend2v2.plugin" state="Active"/>
        </dependencies>
    </moduleDependencies>

    <requiredInstallFiles>
        <file source="example.plugin"/>
    </requiredInstallFiles>

</config>
```

The *dependencies* tag works exactly like *moduleDependencies* (remember what I said before?). It has the same attribute (*operator*, and it works the same way), the same possible children. You can even have another *dependencies* within it!

So how does it all resolve? Let's start from the top:

- *moduleDependencies*'s *operator* is "And" so we need to meet all dependencies;
- First, the dependency on **depend1.plugin** is always mandatory;
- Second, the nested *dependencies* has to be met too, so we go down:
  - This *operator* is "Or" so at least on of these files has to exist;
  - If either **depend2v1.plugin** or **depend2v2.plugin** exist, this is met.
- And we go back up and check if if they're all met. If they are, installation moves on and if not, installation stops here and the actual installer complains!

To finish off this section, there might be another useful tag to use with *moduleDependencies*: *gameDependency*. It's used like this:

```
<moduleDependencies>
    <gameDependency version="1.0"/>
</moduleDependencies>
```

It pretty much just specifies a minimum version of the game that the mod needs to be able to run.

And finally, you now successfully depend on two other mods to install!

Example 02

1.1.3 A Step Forward

And we finally get to the most important part of the installer - the installation steps.

You've worked a bit more on your mod and now you offer users a choice between two features:

```
.
- fomod
|   - info.xml
|   - ModuleConfig.xml
|   - option_a.png
|   - option_b.png
- example_a.plugin
- example_b.plugin
- readme.txt
```

So now let's go step-by-step in understanding how to present this to the user:

```
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://qconsulting.ca/fo3/ModConfig5.0.xsd">

    <moduleName>Example Mod</moduleName>

    <moduleDependencies operator="And">
        <fileDependency file="depend1.plugin" state="Active"/>
        <dependencies operator="Or">
            <fileDependency file="depend2v1.plugin" state="Active"/>
            <fileDependency file="depend2v2.plugin" state="Active"/>
        </dependencies>
    </moduleDependencies>

    <installSteps order="Explicit">
        <installStep name="Choose Option">
            <optionalFileGroups order="Explicit">
                <group name="Select an option:" type="SelectExactlyOne">
                    <plugins order="Explicit">

                        <plugin name="Option A">
                            <description>Select this to install Option A!</
↪description>
                            <image path="fomod/option_a.png"/>
                            <files>
                                <file source="example_a.plugin"/>
                            </files>
                            <typeDescriptor>
                                <type name="Recommended"/>
                            </typeDescriptor>
                        </plugin>

                        <plugin name="Option B">
                            <description>Select this to install Option B!</
↪description>
                            <image path="fomod/option_b.png"/>
                            <files>
                                <file source="example_b.plugin"/>
                            </files>
                            <typeDescriptor>
                                <type name="Optional"/>
                            </typeDescriptor>
                        </plugin>

                    </plugins>
                </group>
            </optionalFileGroups>
        </installStep>
    </installSteps>

</config>
```

Don’t panic. First, *requiredInstallFiles* was removed since we no longer need it.

installSteps is the root tag for this portion. All it does is contain the individual steps and set the order they appear in via the *order* attribute. Like with the *optionalFileGroups* and *plugins* tags, you’ll want to keep this value to “Explicit”. For more info on this take a look at the *Tips and Tricks*.

Next, *installStep*. The step itself and for now all it does is name the step (*name* attribute) and hold the next tag.

optionalFileGroups, has the same *order* attribute as *installSteps* and does nothing more than holding groups.

group is an interesting tag - all options, or *plugins*, below here will be grouped and all groups in the same step will be visible at once. This allows the user to make several choices in the same step and is incredibly useful for you (less work) as long as these choices don’t require interaction between them (which we’ll get to in the next section!).

So the *group* tag holds the *plugins* and you get to define the name of the group (*name* attribute) and its type. I won’t waste time explaining them since they’re so simple and self-explanatory: “SelectAny”, “SelectAll”, “SelectExactly-One”, “SelectAtMostOne” and “SelectAtLeastOne”.

Unlike the previous tag, *plugins* is boring. Same deal as *optionalFileGroups* but with plugins.

plugin is where all the magic happens. This corresponds to an option the user can take during installation. The *name* attribute is what the option will be called and *description* the... description. While it is not required to set an *image* for this option it is highly recommended.

In *files* you set the files you want to install if this option is selected, exactly the same way as *requiredInstallFiles*. Lastly, *typeDescriptor* is a bit complex but for what we want and need most of the time what you see in the example is enough. In the *name* attribute in *type* you have a choice between:

- "Optional", where the option is... optional. Yep.
- "Required", where the user doesn't really have a choice. Useful for including small readmes during the installation and hoping the user reads them this way.
- "Recommended", where the option is usually pre-selected. Be careful as implementation of this varies.

There are actually two more possible but they're useless.

To finish this section here's a little piece of advice - try to keep your files the same name regardless of version and user options. Other's tools may be depending on it and it's considered general courtesy to do so. So our example's package and *installSteps* should look like this instead:

```
.
- fomod
|   - info.xml
|   - ModuleConfig.xml
|   - option_a.png
|   - option_b.png
- option_a
|   - example.plugin
- option_b
|   - example.plugin
- readme.txt
```

```xml
<installSteps order="Explicit">
    <installStep name="Choose Option">
        <optionalFileGroups order="Explicit">
            <group name="Select an option:" type="SelectExactlyOne">
                <plugins order="Explicit">
                    <plugin name="Option A">
                        <description>Select this to install Option A!</description>
                        <image path="fomod/option_a.png"/>
                        <files>
                            <folder source="option_a"/>
                        </files>
                        <typeDescriptor>
                            <type name="Recommended"/>
                        </typeDescriptor>
                    </plugin>
                    <plugin name="Option B">
                        <description>Select this to install Option B!</description>
                        <image path="fomod/option_b.png"/>
                        <files>
                            <folder source="option_b"/>
                        </files>
                        <typeDescriptor>
                            <type name="Optional"/>
                        </typeDescriptor>
                    </plugin>
                </plugins>
            </group>
        </optionalFileGroups>
    </installStep>
</installSteps>
```

Example 03

1.1.4 Flags and You

Your mod's new version now features a choice between textures: Blue or Red. But you needed to make a version of each texture for each plugin version:

```
.
- fomod
|   - info.xml
|   - ModuleConfig.xml
|   - option_a.png
|   - option_b.png
|   - texture_blue.png
|   - texture_red.png
- plugin_a
|   - example.plugin
- plugin_b
|   - example.plugin
- texture_blue_a
|   - texture.tga
- texture_blue_b
|   - texture.tga
- texture_red_a
|   - texture.tga
- texture_red_b
    - texture.tga
```

Ugh, it's getting complex. Let's see what we can make of our steps:

```
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://qconsulting.ca/fo3/ModConfig5.0.xsd">

    <moduleName>Example Mod</moduleName>

    <moduleDependencies operator="And">
        <fileDependency file="depend1.plugin" state="Active"/>
        <dependencies operator="Or">
            <fileDependency file="depend2v1.plugin" state="Active"/>
            <fileDependency file="depend2v2.plugin" state="Active"/>
        </dependencies>
    </moduleDependencies>

    <installSteps order="Explicit">

        <installStep name="Choose Option">
            <optionalFileGroups order="Explicit">
                <group name="Select an option:" type="SelectExactlyOne">
                    <plugins order="Explicit">

                        <plugin name="Option A">
                            <description>Select this to install Option A!</
↪description>
```

```
                        <image path="fomod/option_a.png"/>
                        <files>
                            <folder source="option_a"/>
                        </files>
                        <conditionFlags>
                            <flag name="option_a">selected</flag>
                        </conditionFlags>
                        <typeDescriptor>
                            <type name="Recommended"/>
                        </typeDescriptor>
                    </plugin>

                    <plugin name="Option B">
                        <description>Select this to install Option B!</
↪description>
                        <image path="fomod/option_b.png"/>
                        <files>
                            <folder source="option_b"/>
                        </files>
                        <conditionFlags>
                            <flag name="option_b">selected</flag>
                        </conditionFlags>
                        <typeDescriptor>
                            <type name="Optional"/>
                        </typeDescriptor>
                    </plugin>

                </plugins>
            </group>
        </optionalFileGroups>
    </installStep>

    <installStep name="Choose Texture">
        <visible>
            <flagDependency flag="option_a" value="selected"/>
        </visible>
        <optionalFileGroups order="Explicit">
            <group name="Select a texture:" type="SelectExactlyOne">
                <plugins order="Explicit">

                    <plugin name="Texture Blue">
                        <description>Select this to install Texture Blue!</
↪description>
                        <image path="fomod/texture_blue.png"/>
                        <files>
                            <folder source="texture_blue_a"/>
                        </files>
                        <typeDescriptor>
                            <type name="Optional"/>
                        </typeDescriptor>
                    </plugin>

                    <plugin name="Texture Red">
                        <description>Select this to install Texture Red!</
↪description>
                        <image path="fomod/texture_red.png"/>
                        <files>
                            <folder source="texture_red_a"/>
                            </files>
                            <typeDescriptor>
                                <type name="Optional"/>
                            </typeDescriptor>
                        </plugin>

                    </plugins>
                </group>
            </optionalFileGroups>
        </installStep>

        <installStep name="Choose Texture">
            <visible>
                <flagDependency flag="option_b" value="selected"/>
            </visible>
            <optionalFileGroups order="Explicit">
                <group name="Select a texture:" type="SelectExactlyOne">
                    <plugins order="Explicit">

                        <plugin name="Texture Blue">
                            <description>Select this to install Texture Blue!</
↪description>
                            <image path="fomod/texture_blue.png"/>
                            <files>
                                <folder source="texture_blue_b"/>
                            </files>
                            <typeDescriptor>
                                <type name="Optional"/>
                            </typeDescriptor>
                        </plugin>

                        <plugin name="Texture Red">
                            <description>Select this to install Texture Red!</
↪description>
                            <image path="fomod/texture_red.png"/>
                            <files>
                                <folder source="texture_red_b"/>
                            </files>
                            <typeDescriptor>
                                <type name="Optional"/>
                            </typeDescriptor>
                        </plugin>

                    </plugins>
                </group>
            </optionalFileGroups>
        </installStep>

    </installSteps>

</config>
```

The most obvious change was the addition of two new steps, but we'll get there later.

First let's talk about the existing step. A couple of new tags were added: *conditionFlags* and *flag*. *conditionFlags* works much like *files* but for flags - sets the flag to the value you want whenever the option is selected.

Within *plugin* at least one of either *conditionFlags* or *plugin* must exist in any order.

A flag is like a marker with a name and a value that you control. It does nothing by itself but is amazing at communicating things throughout the installer - here we use it to tell the other two steps what was the option the user chose. To resume, set the *flag* name with the *name* attribute and its value with the element's text.

And on to the two last steps. The new tag here is *visible*, which is a *dependency network*. This tag manages whether the step is visible or not - if its conditions are met then the step is shown to the user, otherwise it's skipped.

In the first of these two steps we're installing the textures that correspond with option A in the first step so we make sure to depend on the flag we set on option A for visiblity. In the last one we do the opposite!

That's it really, most of you can now go on making installers for your mods. As you can see, fomod is actually pretty simple! And for the brave ones or those who need a little more to spice up their installer I'll be waiting for you at the next section!

Example 04

1.1.5 The Installation Matrix

So you've finished reading the last section and maybe you thought - *"so if for each choice dependent on a previous one I have to make a new install step, what if I had 10 choices for the user? 20?"* - and you thought very well. In truth, if you followed the previous and you had the user make 10 dependent choices between two options you'd need to make 1023 ($a_n = -1 + 2^n$, where n is the number of choices for the user to make) installation steps.

Instead, you could create an installation matrix:

```
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://qconsulting.ca/fo3/ModConfig5.0.xsd">

    <moduleName>Example Mod</moduleName>

    <moduleDependencies operator="And">
        <fileDependency file="depend1.plugin" state="Active"/>
        <dependencies operator="Or">
            <fileDependency file="depend2v1.plugin" state="Active"/>
            <fileDependency file="depend2v2.plugin" state="Active"/>
        </dependencies>
    </moduleDependencies>

    <installSteps order="Explicit">
        <installStep name="Choose Option">
            <optionalFileGroups order="Explicit">

                <group name="Select an option:" type="SelectExactlyOne">
                    <plugins order="Explicit">

                        <plugin name="Option A">
                            <description>Select this to install Option A!</
↪description>
                            <image path="fomod/option_a.png"/>
                            <conditionFlags>
                                <flag name="option_a">selected</flag>
                            </conditionFlags>
                            <typeDescriptor>
                                <type name="Recommended"/>
                            </typeDescriptor>
                        </plugin>

                        <plugin name="Option B">
                            <description>Select this to install Option B!</
    ↪description>
                            <image path="fomod/option_b.png"/>
                            <conditionFlags>
                                <flag name="option_b">selected</flag>
                            </conditionFlags>
                            <typeDescriptor>
                                <type name="Optional"/>
                            </typeDescriptor>
                        </plugin>

                    </plugins>
                </group>

                <group name="Select a texture:" type="SelectExactlyOne">
                    <plugins order="Explicit">

                        <plugin name="Texture Blue">
                            <description>Select this to install Texture Blue!</
    ↪description>
                            <image path="fomod/texture_blue.png"/>
                            <conditionFlags>
                                <flag name="texture_blue">selected</flag>
                            </conditionFlags>
                            <typeDescriptor>
                                <type name="Optional"/>
                            </typeDescriptor>
                        </plugin>

                        <plugin name="Texture Red">
                            <description>Select this to install Texture Red!</
    ↪description>
                            <image path="fomod/texture_red.png"/>
                            <conditionFlags>
                                <flag name="texture_red">selected</flag>
                            </conditionFlags>
                            <typeDescriptor>
                                <type name="Optional"/>
                            </typeDescriptor>
                        </plugin>

                    </plugins>
                </group>

            </optionalFileGroups>
        </installStep>
    </installSteps>

    <conditionalFileInstalls>
        <patterns>
            <pattern>
                <dependencies operator="And">
                    <flagDependency flag="option_a" value="selected"/>
                    <flagDependency flag="texture_blue" value="selected"/>
                </dependencies>
                <files>
                    <folder source="option_a"/>
                    <folder source="texture_blue_a"/>
                </files>
            </pattern>
            <pattern>
                <dependencies operator="And">
                    <flagDependency flag="option_a" value="selected"/>
                    <flagDependency flag="texture_red" value="selected"/>
                </dependencies>
                <files>
                    <folder source="option_a"/>
                    <folder source="texture_red_a"/>
                </files>
            </pattern>
            <pattern>
                <dependencies operator="And">
                    <flagDependency flag="option_b" value="selected"/>
                    <flagDependency flag="texture_blue" value="selected"/>
                </dependencies>
                <files>
                    <folder source="option_b"/>
                    <folder source="texture_blue_b"/>
                </files>
            </pattern>
            <pattern>
                <dependencies operator="And">
                    <flagDependency flag="option_b" value="selected"/>
                    <flagDependency flag="texture_red" value="selected"/>
                </dependencies>
                <files>
                    <folder source="option_b"/>
                    <folder source="texture_red_b"/>
                </files>
            </pattern>
        </patterns>
    </conditionalFileInstalls>

</config>
```

Granted, the number of matrix items (*pattern* tags) you'll need to create in this specific 2 options/choice example is always going to be higher ($a_n = 2^n$, where n is the number of choices for the user to make) than the number of installation steps needed for the same number of choices, BUT you can better organize your steps into groups since they're no longer dependent on each other and this matrix is mostly copy-paste while replacing a few things, while the steps need careful adjustements of the *visible*, *files* and *conditionFlags* tags.

It also looks much better this way.

As you may have understood by now, *conditionalFileInstalls* allows you to create a matrix of *pattern* tags. The mod manager/installer will run through each of these, check *dependencies* and if they match, install anything under *files*. We've talked about these tags before, they work exactly the same way.

And that's it really. All major sections were talked about and you're ready to tackle 99.9% of the *fomod* installers out there. There are a few minor things that also exist but they're so rarely needed that they can be safely ignored by most people. To take a look at some of them continue on to *Tips and Tricks* and if you need something else that isn't even covered there head on over to *Specification* for a complete and exhaustive look at the schema.

Hope you learned something and good luck!

Example 05

1.2 Tips and Tricks

AKA a F.A.Q. for which we couldn't come up with questions

1.2.1 The Schema Location

Maybe you've noticed this appearing as the schema location in the examples: *http://qconsulting.ca/fo3/ModConfig5.0.xsd*. It is not an accident - some mod managers, and I won't name names here, use the link text (not the file the link points to, but the link itself) to check which fomod version the installer is using.

Conclusion - you can't change it unless you're sure your users will never use those mod managers. Hopefully they'll drop that if a new schema version is created.

1.2.2 The Explicit Order

As promised during the tutorial - the fabled *order* attribute.

This is actually pretty simple and, if I might say so, completely ridiculous. Let me show you the options for this attribute:

- "Ascending" - the default
- "Descending"
- "Explicit"

The first two sort your stuff alphabetically with no regards to how you sorted them out and since (the *order* attribute is optional, meaning you can omit it) the default is "Ascending" you'll need to put *order="Explicit"* everywhere in your installer.

No idea why it was decided to put "Ascending" as the default or who even thought it would be a good idea to sort things alphabetically here.

1.2.3 The Type Descriptor

Coming soon!

1.3 Specification

The *fomod* format initially arose from the need to provide users with a simpler way of installing mods without the need to download multiple files.

It was created and initially maintaned by a currently unknown developer.

The *fomod* files (discussed below) can be thought of as a blueprint for a mod manager or an independent mod installer to create a GUI (graphical user interface) to simplify user installation.

1.3.1 Structure

A *fomod* installers requires a specific package structure. Assuming the current directory (`.`) is the package:

```
.
- example.plugin
- readme.txt
```

A `fomod` folder is needed and within it two files that are going to be described below: *Info* and *Config*.

A final structure should resemble:

```
.
- fomod
|   - info.xml
|   - ModuleConfig.xml
- example.plugin
- readme.txt
```

1.3.2 Info File

There is no defined schema for this file but it is required anyway. A proposed schema to fit the majority of mod managers could be:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" version="5.x">
    <xs:element name="fomod">
        <xs:annotation>
            <xs:documentation>
                The following tags are to be filled in according
                to their tags, shouldn't be hard to figure out.
            </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Name" type="xs:string" minOccurs="0"/>
                <xs:element name="Author" type="xs:string" minOccurs="0"/>
                <xs:element name="Version" minOccurs="0">
                    <xs:complexType>
                        <xs:simpleContent>
                            <xs:extension base="xs:string">
                                <xs:attribute name="MachineVersion" type="xs:string"/>
                                    <xs:annotation>
                                        <xs:documentation>
                                            This attribute is used for providing a
                                            machine-readable version.
                                            Examples can be found here - https://en.
↪wikipedia.org/wiki/Software_versioning

                                            Semantic versioning is recommended -␣
↪https://semver.org/
                                        </xs:documentation>
                                    </xs:annotation>
                            </xs:extension>
                        </xs:simpleContent>
                    </xs:complexType>
                </xs:element>
                <xs:element name="Description" type="xs:string" minOccurs="0"/>
                <xs:element name="Website" type="xs:string" minOccurs="0"/>
                <xs:element name="Id" type="xs:string" minOccurs="0"/>
                <xs:any processContents="skip" minOccurs="0" maxOccurs="unbounded">
```

```
                    <xs:annotation>
                        <xs:documentation>
                            This element is used solely for allowing extensions
                            since this is merely a proposed schema.
                        </xs:documentation>
                    </xs:annotation>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

1.3.3 Config File

You can find a complete reference for this file here. However, since it is generally easier to understand, it is recommended that you look through the actual schema.