# Module format and documentation

If you want to contribute your module to most Ansible collections, you must write your module in Python and follow the standard format described below. (Unless you're writing a Windows module, in which case the :ref:`Windows guidelines <developing_modules_general_windows>` apply.) In addition to following this format, you should review our :ref:`submission checklist <developing_modules_checklist>`, :ref:`programming tips <developing_modules_best_practices>`, and :ref:`strategy for maintaining Python 2 and Python 3 compatibility <developing_python_3>`, as well as information about :ref:`testing <developing_testing>` before you open a pull request.

**System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 8); *backlink*

Unknown interpreted text role "ref".

**System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 8); *backlink*

Unknown interpreted text role "ref".

**System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 8); *backlink*

Unknown interpreted text role "ref".

**System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 8); *backlink*

Unknown interpreted text role "ref".

**System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 8); *backlink*

Unknown interpreted text role "ref".

Every Ansible module written in Python must begin with seven standard sections in a particular order, followed by the code. The sections in order are:

- Python shebang & UTF-8 coding
- Copyright and license
- ANSIBLE_METADATA block
- DOCUMENTATION block
- EXAMPLES block
- RETURN block
- Python imports
- Testing module documentation

**Note**

Why don't the imports go first?

Keen Python programmers may notice that contrary to PEP 8's advice we don't put `imports` at the top of the file. This is because the `DOCUMENTATION` through `RETURN` sections are not used by the module code itself; they are essentially extra docstrings for the file. The imports are placed after these special variables for the same reason as PEP 8 puts the imports after the introductory comments and docstrings. This keeps the active parts of the code together and the pieces which are purely informational apart. The decision to exclude E402 is based on readability (which is what PEP 8 is about). Documentation strings in a module are much more similar to module level docstrings, than code, and are never utilized by the module itself. Placing the imports below this documentation and closer to the code, consolidates and groups all related code in a congruent manner to improve readability, debugging and understanding.

**Warning**

**Copy old modules with care!**

Some older Ansible modules have `imports` at the bottom of the file, `Copyright` notices with the full GPL prefix, and/or `DOCUMENTATION` fields in the wrong order. These are legacy files that need updating - do not copy them into new modules. Over time we are updating and correcting older modules. Please follow the guidelines on this page!

## Python shebang & UTF-8 coding

Begin your Ansible module with `#!/usr/bin/python` - this "shebang" allows `ansible_python_interpreter` to work. Follow the shebang immediately with `# -*- coding: utf-8 -*-` to clarify that the file is UTF-8 encoded.

## Copyright and license

After the shebang and UTF-8 coding, add a copyright line with the original copyright holder and a license declaration. The license declaration should be ONLY one line, not the full GPL prefix.:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Copyright: (c) 2018, Terry Jones <terry.jones@example.org>
# GNU General Public License v3.0+ (see COPYING or https://www.gnu.org/licenses/gpl-3.0.txt)
```

Major additions to the module (for instance, rewrites) may add additional copyright lines. Any legal review will include the source control history, so an exhaustive copyright header is not necessary. Please do not edit the existing copyright year. This simplifies project administration and is unlikely to cause any interesting legal issues. When adding a second copyright line for a significant feature or rewrite, add the newer line above the older one:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Copyright: (c) 2017, [New Contributor(s)]
# Copyright: (c) 2015, [Original Contributor(s)]
# GNU General Public License v3.0+ (see COPYING or https://www.gnu.org/licenses/gpl-3.0.txt)
```

## ANSIBLE_METADATA block

Since we moved to collections we have deprecated the METADATA functionality, it is no longer required for modules, but it will not break anything if present.

## DOCUMENTATION block

After the shebang, the UTF-8 coding, the copyright line, and the license section comes the `DOCUMENTATION` block. Ansible's online module documentation is generated from the `DOCUMENTATION` blocks in each module's source code. The `DOCUMENTATION` block must be valid YAML. You may find it easier to start writing your `DOCUMENTATION` string in an :ref:`editor with YAML syntax highlighting <other_tools_and_programs>` before you include it in your Python file. You can start by copying our example documentation string into your module file and modifying it. If you run into syntax issues in your YAML, you can validate it on the YAML Lint website.

**System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 72); *backlink***

Unknown interpreted text role "ref".

Module documentation should briefly and accurately define what each module and option does, and how it works with others in the underlying system. Documentation should be written for broad audience--readable both by experts and non-experts.

- Descriptions should always start with a capital letter and end with a full stop. Consistency always helps.
- Verify that arguments in doc and module spec dict are identical.
- For password / secret arguments `no_log=True` should be set.
- For arguments that seem to contain sensitive information but **do not** contain secrets, such as "password_length", set `no_log=False` to disable the warning message.
- If an option is only sometimes required, describe the conditions. For example, "Required when I(state=present)."
- If your module allows `check_mode`, reflect this fact in the documentation.

To create clear, concise, consistent, and useful documentation, follow the :ref:`style guide <style_guide>`.

**System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 82); *backlink***

Unknown interpreted text role "ref".

Each documentation field is described below. Before committing your module documentation, please test it at the command line and as HTML:

- As long as your module file is :ref:`available locally <local_modules>`, you can use `ansible-doc -t module my_module_name` to view your module documentation at the command line. Any parsing errors will be obvious - you can view details by adding `-vvv` to the command.

**System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 86); *backlink*

Unknown interpreted text role "ref".

- You should also :ref:`test the HTML output <testing_module_documentation>` of your module documentation.

**System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 87); *backlink*

Unknown interpreted text role "ref".

## Documentation fields

All fields in the `DOCUMENTATION` block are lower-case. All fields are required unless specified otherwise:

**module:**
- The name of the module.
- Must be the same as the filename, without the `.py` extension.

**short_description:**
- A short description which is displayed on the :ref:`list_of_collections` page and `ansible-doc -l`.

  **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 101); *backlink*

  Unknown interpreted text role "ref".

- The `short_description` is displayed by `ansible-doc -l` without any category grouping, so it needs enough detail to explain the module's purpose without the context of the directory structure in which it lives.
- Unlike `description:`, `short_description` should not have a trailing period/full stop.

**description:**
- A detailed description (generally two or more sentences).
- Must be written in full sentences, in other words, with capital letters and periods/full stops.
- Shouldn't mention the module name.
- Make use of multiple entries rather than using one long paragraph.
- Don't quote complete values unless it is required by YAML.

**version_added:**
- The version of Ansible when the module was added.
- This is a string, and not a float, for example, `version_added: '2.1'`.
- In collections, this must be the collection version the module was added to, not the Ansible version. For example, `version_added: 1.0.0`.

**author:**
- Name of the module author in the form `First Last (@GitHubID)`.
- Use a multi-line list if there is more than one author.
- Don't use quotes as it should not be required by YAML.

**deprecated:**
- Marks modules that will be removed in future releases. See also :ref:`module_lifecycle`.

  **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 128); *backlink*

  Unknown interpreted text role "ref".

**options:**
- Options are often called *parameters* or *arguments*. Because the documentation field is called *options*, we will use that term.
- If the module has no options (for example, it's a `_facts` module), all you need is one line: `options: {}`.
- If your module has options (in other words, accepts arguments), each option should be documented thoroughly. For each module option, include:

  **option-name:**
  - Declarative operation (not CRUD), to focus on the final state, for example *online:*, rather than *is_online:*.
  - The name of the option should be consistent with the rest of the module, as well as other modules in the same category.
  - When in doubt, look for other modules to find option names that are used for the same purpose, we like to offer consistency to our users.

  **description:**

| | |
|---|---|
| | • Detailed explanation of what this option does. It should be written in full sentences.<br>• The first entry is a description of the option itself; subsequent entries detail its use, dependencies, or format of possible values.<br>• Should not list the possible values (that's what `choices:` is for, though it should explain what the values do if they aren't obvious).<br>• If an option is only sometimes required, describe the conditions. For example, "Required when I(state=present)."<br>• Mutually exclusive options must be documented as the final sentence on each of the options. |
| **required:** | • Only needed if `true`.<br>• If missing, we assume the option is not required. |
| **default:** | • If `required` is false/missing, `default` may be specified (assumed 'null' if missing).<br>• Ensure that the default value in the docs matches the default value in the code.<br>• The default field must not be listed as part of the description, unless it requires additional information or conditions.<br>• If the option is a boolean value, you can use any of the boolean values recognized by Ansible: (such as true/false or yes/no). Choose the one that reads better in the context of the option. |
| **choices:** | • List of option values.<br>• Should be absent if empty. |
| **type:** | • Specifies the data type that option accepts, must match the `argspec`.<br>• If an argument is `type='bool'`, this field should be set to `type: bool` and no `choices` should be specified.<br>• If an argument is `type='list'`, `elements` should be specified. |
| **elements:** | • Specifies the data type for list elements in case `type='list'`. |
| **aliases:** | • List of optional name aliases.<br>• Generally not needed. |
| **version_added:** | • Only needed if this option was extended after initial Ansible release, in other words, this is greater than the top level *version_added* field.<br>• This is a string, and not a float, for example, `version_added: '2.3'`.<br>• In collections, this must be the collection version the option was added to, not the Ansible version. For example, `version_added: 1.0.0`. |
| **suboptions:** | |

- If this option takes a dict or list of dicts, you can define the structure here.
- See :ref:`ansible_collections.azure.azcollection.azure_rm_securitygroup_module`, :ref:`ansible_collections.azure.azcollection.azure_rm_azurefirewall_module`, and :ref:`ansible_collections.openstack.cloud.baremetal_node_action_module` for examples.

**System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 191); *backlink***

Unknown interpreted text role "ref".

**System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 191); *backlink***

Unknown interpreted text role "ref".

**System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 191); *backlink***

Unknown interpreted text role "ref".

**requirements:**

- List of requirements (if applicable).
- Include minimum versions.

**seealso:**

- A list of references to other modules, documentation or Internet resources
- In Ansible 2.10 and later, references to modules must use the FQCN or `ansible.builtin` for modules in `ansible-core`.
- A reference can be one of the following formats:

```
seealso:

# Reference by module name
- module: cisco.aci.aci_tenant

# Reference by module name, including description
- module: cisco.aci.aci_tenant
  description: ACI module to create tenants on a Cisco ACI fabric.

# Reference by rST documentation anchor
- ref: aci_guide
  description: Detailed information on how to manage your ACI infrastructure using Ans

# Reference by rST documentation anchor (with custom title)
- ref: The official Ansible ACI guide <aci_guide>
  description: Detailed information on how to manage your ACI infrastructure using Ans

# Reference by Internet resource
- name: APIC Management Information Model reference
  description: Complete reference of the APIC object model.
  link: https://developer.cisco.com/docs/apic-mim-ref/
```

- If you use `ref:` to link to an anchor that is not associated with a title, you must add a title to the ref for the link to work correctly.
- You can link to non-module plugins with `ref:` using the rST anchor, but plugin and module anchors are never associated with a title, so you must supply a title when you link to them. For example `ref: namespace.collection.plugin_name lookup plugin <ansible_collections.namespace.collection.plugin_name_lookup>`.

**notes:**

- Details of any important information that doesn't fit in one of the above sections.
- For example, whether `check_mode` is or is not supported.

## Linking and other format macros within module documentation

You can link from your module documentation to other module docs, other resources on docs.ansible.com, and resources elsewhere on the internet with the help of some pre-defined macros. The correct formats for these macros are:

- `L()` for links with a heading. For example: `See L(Ansible Automation Platform,https://www.ansible.com/products/automation-platform).` As of Ansible 2.10, do not use `L()` for relative links between Ansible documentation and collection documentation.
- `U()` for URLs. For example: `See U(https://www.ansible.com/products/automation-platform) for an overview.`
- `R()` for cross-references with a heading (added in Ansible 2.10). For example: `See R(Cisco IOS Platform Guide,ios_platform_options)`. Use the RST anchor for the cross-reference. See :ref:`adding_anchors_rst` for details.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 247); *backlink*
>
> Unknown interpreted text role "ref".

- `M()` for module names. For example: `See also M(ansible.builtin.yum) or M(community.general.apt_rpm).`

There are also some macros which do not create links but we use them to display certain types of content in a uniform way:

- `I()` for option names. For example: `Required if I(state=present).` This is italicized in the documentation.
- `C()` for files, option values, and inline code. For example: `If not set the environment variable C(ACME_PASSWORD) will be used.` or `Use C(var | foo.bar.my_filter) to transform C(var) into the required format.` This displays with a mono-space font in the documentation.
- `B()` currently has no standardized usage. It is displayed in boldface in the documentation.
- `HORIZONTALLINE` is used sparingly as a separator in long descriptions. It becomes a horizontal rule (the `<hr>` html tag) in the documentation.

> **Note**
>
> For links between modules and documentation within a collection, you can use any of the options above. For links outside of your collection, use `R()` if available. Otherwise, use `U()` or `L()` with full URLs (not relative links). For modules, use `M()` with the FQCN or `ansible.builtin` as shown in the example. If you are creating your own documentation site, you will need to use the intersphinx extension to convert `R()` and `M()` to the correct links.

> **Note**
>
> To refer to a group of modules in a collection, use `R()`. When a collection is not the right granularity, use `C(..)`:
>
> - `Refer to the R(kubernetes.core collection, plugins_in_kubernetes.core) for information on managing kubernetes clusters.`
> - `The C(win_*) modules (spread across several collections) allow you to manage various aspects of windows hosts.`

> **Note**
>
> Because it stands out better, use `seealso` for general references over the use of notes or adding links to the description.

## Documentation fragments

If you are writing multiple related modules, they may share common documentation, such as authentication details, file mode settings, `notes:` or `seealso:` entries. Rather than duplicate that information in each module's `DOCUMENTATION` block, you can save it once as a doc_fragment plugin and use it in each module's documentation. In Ansible, shared documentation fragments are contained in a `ModuleDocFragment` class in lib/ansible/plugins/doc_fragments/ or the equivalent directory in a collection. To include a documentation fragment, add `extends_documentation_fragment: FRAGMENT_NAME` in your module documentation. Use the fully qualified collection name for the FRAGMENT_NAME (for example, `kubernetes.core.k8s_auth_options`).

Modules should only use items from a doc fragment if the module will implement all of the interface documented there in a manner that behaves the same as the existing modules which import that fragment. The goal is that items imported from the doc fragment will behave identically when used in another module that imports the doc fragment.

By default, only the `DOCUMENTATION` property from a doc fragment is inserted into the module documentation. It is possible to define additional properties in the doc fragment in order to import only certain parts of a doc fragment or mix and match as appropriate. If a property is defined in both the doc fragment and the module, the module value overrides the doc fragment.

Here is an example doc fragment named `example_fragment.py`:

```
class ModuleDocFragment(object):
    # Standard documentation
    DOCUMENTATION = r'''
    options:
      # options here
    '''

    # Additional section
    OTHER = r'''
    options:
      # other options here
    '''
```

To insert the contents of `OTHER` in a module:

```
extends_documentation_fragment: example_fragment.other
```

Or use both :

```
extends_documentation_fragment:
  - example_fragment
  - example_fragment.other
```

**System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 319)**

Unknown directive type "versionadded".

```
.. versionadded:: 2.8
```

Since Ansible 2.8, you can have user-supplied doc_fragments by using a `doc_fragments` directory adjacent to play or role, just like any other plugin.

For example, all AWS modules should include:

```
extends_documentation_fragment:
- aws
- ec2
```

:ref:`docfragments_collections` describes how to incorporate documentation fragments in a collection.

**System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 331); *backlink***

Unknown interpreted text role "ref".

## EXAMPLES block

After the shebang, the UTF-8 coding, the copyright line, the license section, and the `DOCUMENTATION` block comes the `EXAMPLES` block. Here you show users how your module works with real-world examples in multi-line plain-text YAML format. The best examples are ready for the user to copy and paste into a playbook. Review and update your examples with every change to your module.

Per playbook best practices, each example should include a `name:` line:

```
EXAMPLES = r'''
- name: Ensure foo is installed
  namespace.collection.modulename:
    name: foo
    state: present
'''
```

The `name:` line should be capitalized and not include a trailing dot.

Use a fully qualified collection name (FQCN) as a part of the module's name like in the example above. For modules in `ansible-core`, use the `ansible.builtin.` identifier, for example `ansible.builtin.debug`.

If your examples use boolean options, use yes/no values. Since the documentation generates boolean values as yes/no, having the examples use these values as well makes the module documentation more consistent.

If your module returns facts that are often needed, an example of how to use them can be helpful.

## RETURN block

After the shebang, the UTF-8 coding, the copyright line, the license section, `DOCUMENTATION` and `EXAMPLES` blocks comes the `RETURN` block. This section documents the information the module returns for use by other modules.

If your module doesn't return anything (apart from the standard returns), this section of your module should read: `RETURN = r''' # '''` Otherwise, for each value returned, provide the following fields. All fields are required unless specified otherwise.

**return name:** Name of the returned field.

  **description:** Detailed description of what this value represents. Capitalized and with trailing dot.

| | |
|---|---|
| **returned:** | When this value is returned, such as `always`, `changed` or `success`. This is a string and can contain any human-readable content. |
| **type:** | Data type. |
| **elements:** | If `type='list'`, specifies the data type of the list's elements. |
| **sample:** | One or more examples. |
| **version_added:** | Only needed if this return was extended after initial Ansible release, in other words, this is greater than the top level *version_added* field. This is a string, and not a float, for example, `version_added: '2.3'`. |
| **contains:** | Optional. To describe nested return values, set `type: dict`, or `type: list/elements: dict`, or if you really have to, `type: complex`, and repeat the elements above for each sub-field. |

Here are two example `RETURN` sections, one with three simple fields and one with a complex nested field:

```
RETURN = r'''
dest:
    description: Destination file/path.
    returned: success
    type: str
    sample: /path/to/file.txt
src:
    description: Source file used for the copy on the target machine.
    returned: changed
    type: str
    sample: /home/httpd/.ansible/tmp/ansible-tmp-1423796390.97-147729857856000/source
md5sum:
    description: MD5 checksum of the file after running copy.
    returned: when supported
    type: str
    sample: 2a5aeecc61dc98c4d780b14b330e3282
'''

RETURN = r'''
packages:
    description: Information about package requirements.
    returned: success
    type: dict
    contains:
        missing:
            description: Packages that are missing from the system.
            returned: success
            type: list
            elements: str
            sample:
                - libmysqlclient-dev
                - libxml2-dev
        badversion:
            description: Packages that are installed but at bad versions.
            returned: success
            type: list
            elements: dict
            sample:
                - package: libxml2-dev
                  version: 2.9.4+dfsg1-2
                  constraint: ">= 3.0"
'''
```

## Python imports

After the shebang, the UTF-8 coding, the copyright line, the license, and the sections for `DOCUMENTATION`, `EXAMPLES`, and `RETURN`, you can finally add the python imports. All modules must use Python imports in the form:

```
from module_utils.basic import AnsibleModule
```

The use of "wildcard" imports such as `from module_utils.basic import *` is no longer allowed.

## Testing module documentation

To test Ansible documentation locally please :ref:`follow instruction<testing_module_documentation>`.

**System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\dev_guide\[ansible-devel][docs][docsite][rst][dev_guide]developing_modules_documenting.rst`, line 453); *backlink***

Unknown interpreted text role "ref".