Ansible module architecture

If you are working on the `ansible-core` code, writing an Ansible module, or developing an action plugin, you may need to understand how Ansible's program flow executes. If you are just using Ansible Modules in playbooks, you can skip this section.

Types of modules

Ansible supports several different types of modules in its code base. Some of these are for backwards compatibility and others are to enable flexibility.

Action plugins

Action plugins look like modules to anyone writing a playbook. Usage documentation for most action plugins lives inside a module of the same name. Some action plugins do all the work, with the module providing only documentation. Some action plugins execute modules. The `normal` action plugin executes modules that don't have special action plugins. Action plugins always execute on the controller.

Some action plugins do all their work on the controller. For example, the :ref:`debug <debug_module>` action plugin (which prints text for the user to see) and the :ref:`assert <assert_module>` action plugin (which tests whether values in a playbook satisfy certain criteria) execute entirely on the controller.

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_program_flow_modules.rst`, **line 28**); ***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_program_flow_modules.rst`, **line 28**); ***backlink***

Unknown interpreted text role "ref".

Most action plugins set up some values on the controller, then invoke an actual module on the managed node that does something with these values. For example, the :ref:`template <template_module>` action plugin takes values from the user to construct a file in a temporary location on the controller using variables from the playbook environment. It then transfers the temporary file to a temporary file on the remote system. After that, it invokes the :ref:`copy module <copy_module>` which operates on the remote system to move the file into its final location, sets file permissions, and so on.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-`

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_program_flow_modules.rst, line 33); *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_program_flow_modules.rst, line 33); *backlink*

Unknown interpreted text role "ref".

New-style modules

All of the modules that ship with Ansible fall into this category. While you can write modules in any language, all official modules (shipped with Ansible) use either Python or PowerShell.

New-style modules have the arguments to the module embedded inside of them in some manner. Old-style modules must copy a separate file over to the managed node, which is less efficient as it requires two over-the-wire connections instead of only one.

Python

New-style Python modules use the :ref:`Ansiballz` framework for constructing modules. These modules use imports from `ansible.module_utils` to pull in boilerplate module code, such as argument parsing, formatting of return values as :term:`JSON`, and various file operations.

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_program_flow_modules.rst, line 58); *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_program_flow_modules.rst, line 58); *backlink*

Unknown interpreted text role "term".

Note

In Ansible, up to version 2.0.x, the official Python modules used the :ref:`module_replacer` framework. For module authors, :ref:`Ansiballz` is largely a superset of :ref:`module_replacer` functionality, so you usually do not need to understand the differences between them.

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_program_flow_modules.rst, line 63); *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_program_flow_modules.rst, line 63); *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_program_flow_modules.rst, line 63); *backlink*

Unknown interpreted text role "ref".

PowerShell

New-style PowerShell modules use the :ref:`module_replacer` framework for constructing modules. These modules get a library of PowerShell code embedded in them before being sent to the managed node.

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_program_flow_modules.rst, line 73); *backlink*

Unknown interpreted text role "ref".

JSONARGS modules

These modules are scripts that include the string `<<INCLUDE_ANSIBLE_MODULE_JSON_ARGS>>` in their body. This string is replaced with the JSON-formatted argument string. These modules typically set a variable to that value like this:

```
json_arguments = """<<INCLUDE_ANSIBLE_MODULE_JSON_ARGS>>"""
```

Which is expanded as:

```
json_arguments = """{"param1": "test's quotes", "param2": "\"To be or not to be\" - Hamlet"}"""
```

Note

Ansible outputs a :term:`JSON` string with bare quotes. Double quotes are used to quote string values, double quotes inside of string values are backslash escaped, and single quotes may appear unescaped inside of a string value. To use JSONARGS, your scripting language must have a way to handle this type of string. The example uses Python's triple quoted strings to do this. Other scripting languages may have a similar quote character that won't be confused by any quotes in the JSON or it may allow you to define your own start-of-quote and end-of-quote characters. If the language doesn't give you any of these then you'll need to write a :ref:`non-native JSON module <flow_want_json_modules>` or :ref:`Old-style module <flow_old_style_modules>` instead.

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_program_flow_modules.rst, **line 96**); *backlink*

Unknown interpreted text role "term".

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_program_flow_modules.rst, **line 96**); *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_program_flow_modules.rst, **line 96**); *backlink*

Unknown interpreted text role "ref".

These modules typically parse the contents of `json_arguments` using a JSON library and then use them as native variables throughout the code.

Non-native want JSON modules

If a module has the string `WANT_JSON` in it anywhere, Ansible treats it as a non-native module that accepts a filename as its only command line parameter. The filename is for a temporary file containing a :term:`JSON` string containing the module's parameters. The module needs to open the file, read and parse the parameters, operate on the data, and print its return data as a JSON encoded dictionary to stdout before exiting.

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_program_flow_modules.rst, **line 116**); *backlink*

Unknown interpreted text role "term".

These types of modules are self-contained entities. As of Ansible 2.1, Ansible only modifies them to change a shebang line if present.

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_program_flow_modules.rst, **line 126**)

Unknown directive type "seealso".

```
.. seealso:: Examples of Non-native modules written in ruby are in the `Ansible
    for Rubyists <https://github.com/ansible/ansible-for-rubyists>`_ repository.
```

Binary modules

From Ansible 2.2 onwards, modules may also be small binary programs. Ansible doesn't perform any magic to make these portable to different systems so they may be specific to the system on which they were compiled or require other binary runtime dependencies. Despite these drawbacks, you may have to compile a custom module against a specific binary library if that's the only way to get access to certain resources.

Binary modules take their arguments and return data to Ansible in the same way as :ref:`want JSON modules <flow_want_json_modules>`.

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_program_flow_modules.rst, **line 141**); *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_program_flow_modules.rst, line 144)

Unknown directive type "seealso".

```
.. seealso:: One example of a `binary module
    <https://github.com/ansible/ansible/blob/devel/test/integration/targets/binary_modules/library/hellowo
    written in go.
```

Old-style modules

Old-style modules are similar to :ref:`want JSON modules <flow_want_json_modules>`, except that the file that they take contains `key=value` pairs for their parameters instead of :term:`JSON`. Ansible decides that a module is old-style when it doesn't have any of the markers that would show that it is one of the other types.

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_program_flow_modules.rst, line 153); *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_program_flow_modules.rst, line 153); *backlink*

Unknown interpreted text role "term".

How modules are executed

When a user uses :program:`ansible` or :program:`ansible-playbook`, they specify a task to execute. The task is usually the name of a module along with several parameters to be passed to the module. Ansible takes these values and processes them in various ways before they are finally executed on the remote machine.

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_program_flow_modules.rst, line 164); *backlink*

Unknown interpreted text role "program".

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_program_flow_modules.rst, line 164); *backlink*

Unknown interpreted text role "program".

Executor/task_executor

The TaskExecutor receives the module name and parameters that were parsed from the :term:`playbook <playbooks>` (or from the command line in the case of :command:`/usr/bin/ansible`). It uses the name to decide whether it's looking at a module or an :ref:`Action Plugin <flow_action_plugins>`. If it's a module, it loads the :ref:`Normal Action Plugin <flow_normal_action_plugin>` and passes the name, variables, and other information about the task and play to that Action Plugin for further processing.

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_program_flow_modules.rst, line 175); *backlink*

Unknown interpreted text role "term".

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_program_flow_modules.rst, line 175); *backlink*

Unknown interpreted text role "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_program_flow_modules.rst, line 175); *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_program_flow_modules.rst, line 175); *backlink*

Unknown interpreted text role "ref".

The `normal` action plugin

The `normal` action plugin executes the module on the remote host. It is the primary coordinator of much of the work to actually execute the module on the managed machine.

- It loads the appropriate connection plugin for the task, which then transfers or executes as needed to create a connection to that host.
- It adds any internal Ansible properties to the module's parameters (for instance, the ones that pass along `no_log` to the module).
- It works with other plugins (connection, shell, become, other action plugins) to create any temporary files on the remote machine and cleans up afterwards.
- It pushes the module and module parameters to the remote host, although the :ref:`module_common <flow_executor_module_common>` code described in the next section decides which format those will take.

> **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_program_flow_modules.rst`, line 199); *backlink*
>
> Unknown interpreted text role "ref".

- It handles any special cases regarding modules (for instance, async execution, or complications around Windows modules that must have the same names as Python modules, so that internal calling of modules from other Action Plugins work.)

Much of this functionality comes from the *BaseAction* class, which lives in :file:`plugins/action/__init__.py`. It uses the `Connection` and `Shell` objects to do its work.

> **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_program_flow_modules.rst`, line 206); *backlink*
>
> Unknown interpreted text role "file".

> **Note**
>
> When :term:`tasks <tasks>` are run with the `async:` parameter, Ansible uses the `async` Action Plugin instead of the `normal` Action Plugin to invoke it. That program flow is currently not documented. Read the source for information on how that works.
>
> > **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_program_flow_modules.rst`, line 211); *backlink*
> >
> > Unknown interpreted text role "term".

Executor/module_common.py

Code in :file:`executor/module_common.py` assembles the module to be shipped to the managed node. The module is first read in, then examined to determine its type:

> **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_program_flow_modules.rst`, line 221); *backlink*
>
> Unknown interpreted text role "file".

- :ref:`PowerShell <flow_powershell_modules>` and :ref:`JSON-args modules <flow_jsonargs_modules>` are passed through :ref:`Module Replacer <module_replacer>`.

> **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_program_flow_modules.rst`, line 225); *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_program_flow_modules.rst`, line 225); *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_program_flow_modules.rst`, line 225); *backlink*
>
> Unknown interpreted text role "ref".

- New-style :ref:`Python modules <flow_python_modules>` are assembled by :ref:`Ansiballz`.

> **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_program_flow_modules.rst`, line 226); *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_program_flow_modules.rst`, line 226); *backlink*
>
> Unknown interpreted text role "ref".

- :ref:`Non-native-want-JSON <flow_want_json_modules>`, :ref:`Binary modules <flow_binary_modules>`, and :ref:`Old-Style modules <flow_old_style_modules>` aren't touched by either of these and pass through unchanged.

> **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_program_flow_modules.rst`, line 227); *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_program_flow_modules.rst`, line 227); *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_program_flow_modules.rst`, line 227); *backlink*
>
> Unknown interpreted text role "ref".

After the assembling step, one final modification is made to all modules that have a shebang line. Ansible checks whether the interpreter in the shebang line has a specific path configured via an `ansible_$X_interpreter` inventory variable. If it does, Ansible substitutes that path for the interpreter path given in the module. After this, Ansible returns the complete module data and the module type to the :ref:`Normal Action <flow_normal_action_plugin>` which continues execution of the module.

> **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_program_flow_modules.rst`, line 229); *backlink*
>
> Unknown interpreted text role "ref".

Assembler frameworks

Ansible supports two assembler frameworks: Ansiballz and the older Module Replacer.

Module Replacer framework

The Module Replacer framework is the original framework implementing new-style modules, and is still used for PowerShell modules. It is essentially a preprocessor (like the C Preprocessor for those familiar with that programming language). It does straight substitutions of specific substring patterns in the module file. There are two types of substitutions:

- Replacements that only happen in the module file. These are public replacement strings that modules can utilize to get helpful boilerplate or access to arguments.
  - `from ansible.module_utils.MOD_LIB_NAME import *` is replaced with the contents of the :file:`ansible/module_utils/MOD_LIB_NAME.py` These should only be used with :ref:`new-style Python modules <flow_python_modules>`.

    > **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_program_flow_modules.rst`, line 258); *backlink*
    >
    > Unknown interpreted text role "file".

    > **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_program_flow_modules.rst`, line 258); *backlink*
    >
    > Unknown interpreted text role "ref".

  - `#<<INCLUDE_ANSIBLE_MODULE_COMMON>>` is equivalent to `from ansible.module_utils.basic import *` and should also only apply to new-style Python modules.

- `# POWERSHELL_COMMON` substitutes the contents of :file:`ansible/module_utils/powershell.ps1`. It should only be used with :ref:`new-style Powershell modules <flow_powershell_modules>`.

  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_program_flow_modules.rst, line 264); backlink

  Unknown interpreted text role "file".

  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_program_flow_modules.rst, line 264); backlink

  Unknown interpreted text role "ref".

- Replacements that are used by `ansible.module_utils` code. These are internal replacement patterns. They may be used internally, in the above public replacements, but shouldn't be used directly by modules.
  - `"<<ANSIBLE_VERSION>>"` is substituted with the Ansible version. In :ref:`new-style Python modules <flow_python_modules>` under the :ref:`Ansiballz` framework the proper way is to instead instantiate an *AnsibleModule* and then access the version from :attr:`AnsibleModule.ansible_version`.

    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_program_flow_modules.rst, line 270); 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_program_flow_modules.rst, line 270); backlink

    Unknown interpreted text role "ref".

  - `"<<INCLUDE_ANSIBLE_MODULE_COMPLEX_ARGS>>"` is substituted with a string which is the Python `repr` of the :term:`JSON` encoded module parameters. Using `repr` on the JSON string makes it safe to embed in a Python file. In new-style Python modules under the Ansiballz framework this is better accessed by instantiating an *AnsibleModule* and then using :attr:`AnsibleModule.params`.

    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_program_flow_modules.rst, line 275); backlink

    Unknown interpreted text role "term".

    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_program_flow_modules.rst, line 275); backlink

    Unknown interpreted text role "attr".

  - `<<SELINUX_SPECIAL_FILESYSTEMS>>` substitutes a string which is a comma separated list of file systems which have a file system dependent security context in SELinux. In new-style Python modules, if you really need this you should instantiate an *AnsibleModule* and then use :attr:`AnsibleModule._selinux_special_fs`. The variable has also changed from a comma separated string of file system names to an actual python list of filesystem names.

    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_program_flow_modules.rst, line 281); backlink

    Unknown interpreted text role "attr".

  - `<<INCLUDE_ANSIBLE_MODULE_JSON_ARGS>>` substitutes the module parameters as a JSON string. Care must be taken to properly quote the string as JSON data may contain quotes. This pattern is not substituted in new-style Python modules as they can get the module parameters another way.
  - The string `syslog.LOG_USER` is replaced wherever it occurs with the `syslog_facility` which was named in :file:`ansible.cfg` or any `ansible_syslog_facility` inventory variable that applies to this host. In new-style Python modules this has changed slightly. If you really need to access it, you should instantiate an *AnsibleModule* and then use :attr:`AnsibleModule._syslog_facility` to access it. It is no longer the actual syslog facility and is now the name of the syslog facility. See the :ref:`documentation on internal arguments <flow_internal_arguments>` 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_program_flow_modules.rst, line 293); backlink

    Unknown interpreted text role "file".

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_program_flow_modules.rst, line 293); *backlink*

Unknown interpreted text role "attr".

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_program_flow_modules.rst, line 293); *backlink*

Unknown interpreted text role "ref".

Ansiballz framework

The Ansiballz framework was adopted in Ansible 2.1 and is used for all new-style Python modules. Unlike the Module Replacer, Ansiballz uses real Python imports of things in :file:`ansible/module_utils` instead of merely preprocessing the module. It does this by constructing a zipfile -- which includes the module file, files in :file:`ansible/module_utils` that are imported by the module, and some boilerplate to pass in the module's parameters. The zipfile is then Base64 encoded and wrapped in a small Python script which decodes the Base64 encoding and places the zipfile into a temp directory on the managed node. It then extracts just the Ansible module script from the zip file and places that in the temporary directory as well. Then it sets the PYTHONPATH to find Python modules inside of the zip file and imports the Ansible module as the special name, `__main__`. Importing it as `__main__` causes Python to think that it is executing a script rather than simply importing a module. This lets Ansible run both the wrapper script and the module code in a single copy of Python on the remote machine.

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_program_flow_modules.rst, line 308); *backlink*

Unknown interpreted text role "file".

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_program_flow_modules.rst, line 308); *backlink*

Unknown interpreted text role "file".

Note

- Ansible wraps the zipfile in the Python script for two reasons:
  - for compatibility with Python 2.6 which has a less functional version of Python's `-m` command line switch.
  - so that pipelining will function properly. Pipelining needs to pipe the Python module into the Python interpreter on the remote node. Python understands scripts on stdin but does not understand zip files.
- Prior to Ansible 2.7, the module was executed via a second Python interpreter instead of being executed inside of the same process. This change was made once Python-2.4 support was dropped to speed up module execution.

In Ansiballz, any imports of Python modules from the :py:mod:`ansible.module_utils` package trigger inclusion of that Python file into the zipfile. Instances of `#<<INCLUDE_ANSIBLE_MODULE_COMMON>>` in the module are turned into `from ansible.module_utils.basic import *` and :file:`ansible/module-utils/basic.py` is then included in the zipfile. Files that are included from :file:`module_utils` are themselves scanned for imports of other Python modules from :file:`module_utils` to be included in the zipfile as well.

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_program_flow_modules.rst, line 335); *backlink*

Unknown interpreted text role "py:mod".

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_program_flow_modules.rst, line 335); *backlink*

Unknown interpreted text role "file".

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_program_flow_modules.rst, line 335); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-

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_program_flow_modules.rst, line 335); *backlink*

Unknown interpreted text role "file".

Warning

At present, the Ansiballz Framework cannot determine whether an import should be included if it is a relative import. Always use an absolute import that has :py:mod:`ansible.module_utils` in it to allow Ansiballz to determine that the file should be included.

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_program_flow_modules.rst, line 345); *backlink*

Unknown interpreted text role "py:mod".

Passing args

Arguments are passed differently by the two frameworks:

- In :ref:`module_replacer`, module arguments are turned into a JSON-ified string and substituted into the combined module file.

  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_program_flow_modules.rst, line 358); *backlink*

  Unknown interpreted text role "ref".

- In :ref:`Ansiballz`, the JSON-ified string is part of the script which wraps the zipfile. Just before the wrapper script imports the Ansible module as `__main__`, it monkey-patches the private, `_ANSIBLE_ARGS` variable in `basic.py` with the variable values. When a :class:`ansible.module_utils.basic.AnsibleModule` is instantiated, it parses this string and places the args into :attr:`AnsibleModule.params` where it can be accessed by the module's other code.

  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_program_flow_modules.rst, line 359); *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_program_flow_modules.rst, line 359); *backlink*

  Unknown interpreted text role "class".

  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_program_flow_modules.rst, line 359); *backlink*

  Unknown interpreted text role "attr".

Warning

If you are writing modules, remember that the way we pass arguments is an internal implementation detail: it has changed in the past and will change again as soon as changes to the common module_utils code allow Ansible modules to forgo using :class:`ansible.module_utils.basic.AnsibleModule`. Do not rely on the internal global `_ANSIBLE_ARGS` variable.

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_program_flow_modules.rst, line 362); *backlink*

Unknown interpreted text role "class".

Very dynamic custom modules which need to parse arguments before they instantiate an `AnsibleModule` may use `_load_params` to retrieve those parameters. Although `_load_params` may change in breaking ways if necessary to support changes in the code, it is likely to be more stable than either the way we pass parameters or the internal global variable.

Note

Prior to Ansible 2.7, the Ansible module was invoked in a second Python interpreter and the arguments were then passed to the script over the script's stdin.

Internal arguments

Both :ref:`module_replacer` and :ref:`Ansiballz` send additional arguments to the module beyond those which the user specified in the playbook. These additional arguments are internal parameters that help implement global Ansible features. Modules often do not need to know about these explicitly as the features are implemented in :py:mod:`ansible.module_utils.basic` but certain features need support from the module so it's good to know about them.

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_program_flow_modules.rst, line 380); *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_program_flow_modules.rst, line 380); *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_program_flow_modules.rst, line 380); *backlink*

Unknown interpreted text role "py:mod".

The internal arguments listed here are global. If you need to add a local internal argument to a custom module, create an action plugin for that specific module - see `_original_basename` in the copy action plugin for an example.

_ansible_no_log

Boolean. Set to True whenever a parameter in a task or play specifies `no_log`. Any module that calls :py:meth:`AnsibleModule.log` handles this automatically. If a module implements its own logging then it needs to check this value. To access in a module, instantiate an `AnsibleModule` and then check the value of :attr:`AnsibleModule.no_log`.

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_program_flow_modules.rst, line 392); *backlink*

Unknown interpreted text role "py:meth".

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_program_flow_modules.rst, line 392); *backlink*

Unknown interpreted text role "attr".

Note

`no_log` specified in a module's argument_spec is handled by a different mechanism.

_ansible_debug

Boolean. Turns more verbose logging on or off and turns on logging of external commands that the module executes. If a module uses :py:meth:`AnsibleModule.debug` rather than :py:meth:`AnsibleModule.log` then the messages are only logged if `_ansible_debug` is set to `True`. To set, add `debug: True` to :file:`ansible.cfg` or set the environment variable :envvar:`ANSIBLE_DEBUG`. To access in a module, instantiate an `AnsibleModule` and access :attr:`AnsibleModule._debug`.

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_program_flow_modules.rst, line 402); *backlink*

Unknown interpreted text role "py:meth".

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_program_flow_modules.rst, line 402); *backlink*

Unknown interpreted text role "py:meth".

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_program_flow_modules.rst, line 402); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-

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_program_flow_modules.rst, line 402); *backlink*

Unknown interpreted text role "envvar".

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_program_flow_modules.rst, line 402); *backlink*

Unknown interpreted text role "attr".

_ansible_diff

Boolean. If a module supports it, tells the module to show a unified diff of changes to be made to templated files. To set, pass the `--diff` command line option. To access in a module, instantiate an *AnsibleModule* and access :attr:`AnsibleModule._diff`.

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_program_flow_modules.rst, line 413); *backlink*

Unknown interpreted text role "attr".

_ansible_verbosity

Unused. This value could be used for finer grained control over logging.

_ansible_selinux_special_fs

List. Names of filesystems which should have a special SELinux context. They are used by the *AnsibleModule* methods which operate on files (changing attributes, moving, and copying). To set, add a comma separated string of filesystem names in :file:`ansible.cfg`:

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_program_flow_modules.rst, line 426); *backlink*

Unknown interpreted text role "file".

```
# ansible.cfg
[selinux]
special_context_filesystems=nfs,vboxsf,fuse,ramfs,vfat
```

Most modules can use the built-in `AnsibleModule` methods to manipulate files. To access in a module that needs to know about these special context filesystems, instantiate an `AnsibleModule` and examine the list in :attr:`AnsibleModule._selinux_special_fs`.

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_program_flow_modules.rst, line 436); *backlink*

Unknown interpreted text role "attr".

This replaces :attr:`ansible.module_utils.basic.SELINUX_SPECIAL_FS` from :ref:`module_replacer`. In module replacer it was a comma separated string of filesystem names. Under Ansiballz it's an actual list.

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_program_flow_modules.rst, line 440); *backlink*

Unknown interpreted text role "attr".

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_program_flow_modules.rst, line 440); *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_program_flow_modules.rst, line 444)

Unknown directive type "versionadded".

```
.. versionadded:: 2.1
```

_ansible_syslog_facility

This parameter controls which syslog facility Ansible module logs to. To set, change the `syslog_facility` value in :file:`ansible.cfg`. Most modules should just use :meth:`AnsibleModule.log` which will then make use of this. If a module has to use this on its own, it

should instantiate an *AnsibleModule* and then retrieve the name of the syslog facility from :attr:`AnsibleModule._syslog_facility`. The Ansiballz code is less hacky than the old :ref:`module_replacer` code:

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_program_flow_modules.rst, line 449); *backlink*

Unknown interpreted text role "file".

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_program_flow_modules.rst, line 449); *backlink*

Unknown interpreted text role "meth".

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_program_flow_modules.rst, line 449); *backlink*

Unknown interpreted text role "attr".

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_program_flow_modules.rst, line 449); *backlink*

Unknown interpreted text role "ref".

```
# Old module_replacer way
import syslog
syslog.openlog(NAME, 0, syslog.LOG_USER)

# New Ansiballz way
import syslog
facility_name = module._syslog_facility
facility = getattr(syslog, facility_name, syslog.LOG_USER)
syslog.openlog(NAME, 0, facility)
```

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_program_flow_modules.rst, line 467)

Unknown directive type "versionadded".

```
.. versionadded:: 2.1
```

_ansible_version

This parameter passes the version of Ansible that runs the module. To access it, a module should instantiate an *AnsibleModule* and then retrieve it from :attr:`AnsibleModule.ansible_version`. This replaces :attr:`ansible.module_utils.basic.ANSIBLE_VERSION` from :ref:`module_replacer`.

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_program_flow_modules.rst, line 472); *backlink*

Unknown interpreted text role "attr".

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_program_flow_modules.rst, line 472); *backlink*

Unknown interpreted text role "attr".

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_program_flow_modules.rst, line 472); *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_program_flow_modules.rst, line 478)

Unknown directive type "versionadded".

```
.. versionadded:: 2.1
```

Module return values & Unsafe strings

At the end of a module's execution, it formats the data that it wants to return as a JSON string and prints the string to its stdout. The normal action plugin receives the JSON string, parses it into a Python dictionary, and returns it to the executor.

If Ansible templated every string return value, it would be vulnerable to an attack from users with access to managed nodes. If an unscrupulous user disguised malicious code as Ansible return value strings, and if those strings were then templated on the controller, Ansible could execute arbitrary code. To prevent this scenario, Ansible marks all strings inside returned data as `Unsafe`, emitting any Jinja2 templates in the strings verbatim, not expanded by Jinja2.

Strings returned by invoking a module through `ActionPlugin._execute_module()` are automatically marked as `Unsafe` by the normal action plugin. If another action plugin retrieves information from a module through some other means, it must mark its return data as `Unsafe` on its own.

In case a poorly-coded action plugin fails to mark its results as "Unsafe," Ansible audits the results again when they are returned to the executor, marking all strings as `Unsafe`. The normal action plugin protects itself and any other code that it calls with the result data as a parameter. The check inside the executor protects the output of all other action plugins, ensuring that subsequent tasks run by Ansible will not template anything from those results either.

Special considerations

Pipelining

Ansible can transfer a module to a remote machine in one of two ways:

- it can write out the module to a temporary file on the remote host and then use a second connection to the remote host to execute it with the interpreter that the module needs
- or it can use what's known as pipelining to execute the module by piping it into the remote interpreter's stdin.

Pipelining only works with modules written in Python at this time because Ansible only knows that Python supports this mode of operation. Supporting pipelining means that whatever format the module payload takes before being sent over the wire must be executable by Python via stdin.

Why pass args over stdin?

Passing arguments via stdin was chosen for the following reasons:

- When combined with :ref:`ANSIBLE_PIPELINING`, this keeps the module's arguments from temporarily being saved onto disk on the remote machine. This makes it harder (but not impossible) for a malicious user on the remote machine to steal any sensitive information that may be present in the arguments.

  > **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_program_flow_modules.rst`, **line 525**); *backlink*
  >
  > Unknown interpreted text role "ref".

- Command line arguments would be insecure as most systems allow unprivileged users to read the full commandline of a process.
- Environment variables are usually more secure than the commandline but some systems limit the total size of the environment. This could lead to truncation of the parameters if we hit that limit.

AnsibleModule

Argument spec

The `argument_spec` provided to `AnsibleModule` defines the supported arguments for a module, as well as their type, defaults and more.

Example `argument_spec`:

```
module = AnsibleModule(argument_spec=dict(
    top_level=dict(
        type='dict',
        options=dict(
            second_level=dict(
                default=True,
                type='bool',
            )
        )
    )
))
```

This section will discuss the behavioral attributes for arguments:

type: `type` allows you to define the type of the value accepted for the argument. The default value for `type` is `str`. Possible values are:

- str
- list
- dict
- bool
- int
- float
- path

- raw
- jsonarg
- json
- bytes
- bits

The `raw` type, performs no type validation or type casting, and maintains the type of the passed value.

elements: `elements` works in combination with `type` when `type='list'`. `elements` can then be defined as `elements='int'` or any other type, indicating that each element of the specified list should be of that type.

default: The `default` option allows sets a default value for the argument for the scenario when the argument is not provided to the module. When not specified, the default value is `None`.

fallback: `fallback` accepts a `tuple` where the first argument is a callable (function) that will be used to perform the lookup, based on the second argument. The second argument is a list of values to be accepted by the callable.

The most common callable used is `env_fallback` which will allow an argument to optionally use an environment variable when the argument is not supplied.

Example:

```
username=dict(fallback=(env_fallback, ['ANSIBLE_NET_USERNAME']))
```

choices: `choices` accepts a list of choices that the argument will accept. The types of `choices` should match the `type`.

required: `required` accepts a boolean, either `True` or `False` that indicates that the argument is required. When not specified, `required` defaults to `False`. This should not be used in combination with `default`.

no_log: `no_log` accepts a boolean, either `True` or `False`, that indicates explicitly whether or not the argument value should be masked in logs and output.

> **Note**
>
> In the absence of `no_log`, if the parameter name appears to indicate that the argument value is a password or passphrase (such as "admin_password"), a warning will be shown and the value will be masked in logs but **not** output. To disable the warning and masking for parameters that do not contain sensitive information, set `no_log` to `False`.

aliases: `aliases` accepts a list of alternative argument names for the argument, such as the case where the argument is `name` but the module accepts `aliases=['pkg']` to allow `pkg` to be interchangeably with `name`

options: `options` implements the ability to create a sub-argument_spec, where the sub options of the top level argument are also validated using the attributes discussed in this section. The example at the top of this section demonstrates use of `options`. `type` or `elements` should be `dict` is this case.

apply_defaults: `apply_defaults` works alongside `options` and allows the `default` of the sub-options to be applied even when the top-level argument is not supplied.

In the example of the `argument_spec` at the top of this section, it would allow `module.params['top_level']['second_level']` to be defined, even if the user does not provide `top_level` when calling the module.

removed_in_version: `removed_in_version` indicates which version of ansible-core or a collection a deprecated argument will be removed in. Mutually exclusive with `removed_at_date`, and must be used with `removed_from_collection`.

Example:

```
option = {
  'type': 'str',
  'removed_in_version': '2.0.0',
  'collection_name': 'testns.testcol',
},
```

removed_at_date: `removed_at_date` indicates that a deprecated argument will be removed in a minor ansible-core release or major collection release after this date. Mutually exclusive with `removed_in_version`, and must be used with `removed_from_collection`.

Example:

```
option = {
  'type': 'str',
  'removed_at_date': '2020-12-31',
  'collection_name': 'testns.testcol',
},
```

removed_from_collection: Specifies which collection (or ansible-core) deprecates this deprecated argument. Specify `ansible.builtin` for ansible-core, or the collection's name (format `foo.bar`). Must be used with `removed_in_version` or `removed_at_date`.

deprecated_aliases: Deprecates aliases of this argument. Must contain a list or tuple of dictionaries having some the following keys:

name: The name of the alias to deprecate. (Required.)

version: The version of ansible-core or the collection this alias will be removed in. Either `version` or `date` must be specified.

date: The a date after which a minor release of ansible-core or a major collection release will no longer contain this alias.. Either `version` or `date` must be specified.

collection_name: Specifies which collection (or ansible-core) deprecates this deprecated alias. Specify `ansible.builtin` for ansible-core, or the collection's name (format `foo.bar`). Must be used with `version` or `date`.

Examples:

```
option = {
  'type': 'str',
  'aliases': ['foo', 'bar'],
  'depecated_aliases': [
    {
      'name': 'foo',
      'version': '2.0.0',
      'collection_name': 'testns.testcol',
    },
    {
      'name': 'foo',
      'date': '2020-12-31',
      'collection_name': 'testns.testcol',
    },
  ],
},
```

mutually_exclusive: If `options` is specified, `mutually_exclusive` refers to the sub-options described in `options` and behaves as in :ref:`argument_spec_dependencies`.

> **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_program_flow_modules.rst, line 710); *backlink***
>
> Unknown interpreted text role "ref".

required_together: If `options` is specified, `required_together` refers to the sub-options described in `options` and behaves as in :ref:`argument_spec_dependencies`.

> **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_program_flow_modules.rst, line 714); *backlink***
>
> Unknown interpreted text role "ref".

required_one_of: If `options` is specified, `required_one_of` refers to the sub-options described in `options` and behaves as in :ref:`argument_spec_dependencies`.

> **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_program_flow_modules.rst, line 718); *backlink***
>
> Unknown interpreted text role "ref".

required_if: If `options` is specified, `required_if` refers to the sub-options described in `options` and behaves as in :ref:`argument_spec_dependencies`.

> **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_program_flow_modules.rst, line 722); *backlink***
>
> Unknown interpreted text role "ref".

required_by: If `options` is specified, `required_by` refers to the sub-options described in `options` and behaves as in :ref:`argument_spec_dependencies`.

> **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_program_flow_modules.rst, line 726); *backlink***
>
> Unknown interpreted text role "ref".

Dependencies between module options

The following are optional arguments for `AnsibleModule()`:

```
module = AnsibleModule(
  argument_spec,
  mutually_exclusive=[
    ('path', 'content'),
  ],
  required_one_of=[
    ('path', 'content'),
  ],
)
```

mutually_exclusive: Must be a sequence (list or tuple) of sequences of strings. Every sequence of strings is a list of option names which are mutually exclusive. If more than one options of a list are specified together, Ansible will fail the module with an error.

Example:

```
mutually_exclusive=[
  ('path', 'content'),
  ('repository_url', 'repository_filename'),
],
```

In this example, the options `path` and `content` must not specified at the same time. Also the options `repository_url` and `repository_filename` must not be specified at the same time. But specifying `path` and `repository_url` is accepted.

To ensure that precisely one of two (or more) options is specified, combine `mutually_exclusive` with `required_one_of`.

required_together: Must be a sequence (list or tuple) of sequences of strings. Every sequence of strings is a list of option names which are must be specified together. If at least one of these options are specified, the other ones from the same sequence must all be present.

Example:

```
required_together=[
  ('file_path', 'file_hash'),
],
```

In this example, if one of the options `file_path` or `file_hash` is specified, Ansible will fail the module with an error if the other one is not specified.

required_one_of: Must be a sequence (list or tuple) of sequences of strings. Every sequence of strings is a list of option names from which at least one must be specified. If none one of these options are specified, Ansible will fail module execution.

Example:

```
required_one_of=[
  ('path', 'content'),
],
```

In this example, at least one of `path` and `content` must be specified. If none are specified, execution will fail. Specifying both is explicitly allowed; to prevent this, combine `required_one_of` with `mutually_exclusive`.

required_if: Must be a sequence of sequences. Every inner sequence describes one conditional dependency. Every sequence must have three or four values. The first two values are the option's name and the option's value which describes the condition. The further elements of the sequence are only needed if the option of that name has precisely this value.

If you want that all options in a list of option names are specified if the condition is met, use one of the following forms:

```
('option_name', option_value, ('option_a', 'option_b', ...)),
('option_name', option_value, ('option_a', 'option_b', ...), False),
```

If you want that at least one option of a list of option names is specified if the condition is met, use the following form:

```
('option_name', option_value, ('option_a', 'option_b', ...), True),
```

Example:

```
required_if=[
  ('state', 'present', ('path', 'content'), True),
  ('force', True, ('force_reason', 'force_code')),
],
```

In this example, if the user specifies `state=present`, at least one of the options `path` and `content` must be supplied (or both). To make sure that precisely one can be specified, combine `required_if` with `mutually_exclusive`.

On the other hand, if `force` (a boolean parameter) is set to `true`, `yes` etc., both `force_reason` and `force_code` must be specified.

required_by: Must be a dictionary mapping option names to sequences of option names. If the option name in a dictionary key is specified, the option names it maps to must all also be specified. Note that instead of a sequence of option names, you can also specify one single option name.

Example:

```
required_by={
  'force': 'force_reason',
  'path': ('mode', 'owner', 'group'),
},
```

In the example, if `force` is specified, `force_reason` must also be specified. Also, if `path` is specified, then three three options `mode`, `owner` and `group` also must be specified.

Declaring check mode support

To declare that a module supports check mode, supply `supports_check_mode=True` to the `AnsibleModule()` call:

```
module = AnsibleModule(argument_spec, supports_check_mode=True)
```

The module can determine whether it is called in check mode by checking the boolean value `module.check_mode`. If it evaluates to `True`, the module must take care not to do any modification.

If `supports_check_mode=False` is specified, which is the default value, the module will exit in check mode with `skipped=True` and message `remote module (<insert module name here>) does not support check mode.`

Adding file options

To declare that a module should add support for all common file options, supply `add_file_common_args=True` to the `AnsibleModule()` call:

```
module = AnsibleModule(argument_spec, add_file_common_args=True)
```

You can find a list of all file options here. It is recommended that you make your `DOCUMENTATION` extend the doc fragment `ansible.builtin.files` (see :ref:`module_docs_fragments`) in this case, to make sure that all these fields are correctly documented.

> **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_program_flow_modules.rst`, line 860); *backlink*
>
> Unknown interpreted text role "ref".

The helper functions `module.load_file_common_arguments()` and `module.set_fs_attributes_if_different()` can be used to handle these arguments for you:

```
argument_spec = {
  'path': {
    'type': 'str',
    'required': True,
  },
}

module = AnsibleModule(argument_spec, add_file_common_args=True)
changed = False

# TODO do something with module.params['path'], like update it's contents

# Ensure that module.params['path'] satisfies the file options supplied by the user
file_args = module.load_file_common_arguments(module.params)
changed = module.set_fs_attributes_if_different(file_args, changed)

module.exit_json(changed=changed)
```