🏠 (../../../index.html) » Collection Index (../../index.html) » Collections in the Community Namespace (../index.html) » Community.Crypto (index.html) » community.crypto.openssl_pkcs12 – Generate OpenSSL PKCS#12 archive

You are reading the latest community version of the Ansible documentation. Red Hat subscribers, select **2.9** in the version selection to the left for the most recent Red Hat release.

community.crypto.openssl_pkcs12 – Generate OpenSSL PKCS#12 archive

Note

This plugin is part of the community.crypto collection (https://galaxy.ansible.com/community/crypto) (version 1.9.3).

To install it use: `ansible-galaxy collection install community.crypto`.

To use it in a playbook, specify: `community.crypto.openssl_pkcs12`.

Search this site

Synopsis

- This module allows one to (re-)generate PKCS#12.
- The module can use the cryptography Python library, or the pyOpenSSL Python library. By default, it tries to detect which one is available, assuming none of the *iter_size* and *maciter_size* options are used. This can be overridden with the *select_crypto_backend* option.

Requirements

The below requirements are needed on the host that executes this module.

- PyOpenSSL >= 0.15 or cryptography >= 3.0

Parameters

Parameter	Choices/Defaults	Comments
action string	**Choices:** • **export** ← • parse	`export` or `parse` a PKCS#12.
attributes string *added in 2.3 of ansible.builtin*		The attributes the resulting file or directory should have. To get supported flags look at the man page for *chattr* on the target system. This string should contain the attributes in the same order as the one displayed by *lsattr*. The `=` operator is assumed as default, otherwise `+` or `-` operators need to be included in the string. aliases: attr
backup boolean	**Choices:** • **no** ← • yes	Create a backup file including a timestamp so you can get the original output file back if you overwrote it with a new one by accident.
certificate_path path		The path to read certificates and private keys from. Must be in PEM format.
force boolean	**Choices:** • **no** ← • yes	Should the file be regenerated even if it already exists.

friendly_name string		Specifies the friendly name for the certificate and private key. aliases: name
group string		Name of the group that should own the file/directory, as would be fed to *chown*.
iter_size integer		Number of times to repeat the encryption step. This is not considered during idempotency checks. This is only used by the `pyopenssl` backend. When using it, the default is `2048`.
maciter_size integer		Number of times to repeat the MAC step. This is not considered during idempotency checks. This is only used by the `pyopenssl` backend. When using it, the default is `1`.
mode raw		The permissions the resulting file or directory should have. For those used to */usr/bin/chmod* remember that modes are actually octal numbers. You must either add a leading zero so that Ansible's YAML parser knows it is an octal number (like `0644` or `01777`) or quote it (like `'644'` or `'1777'`) so Ansible receives a string and can do its own conversion from string into number. Giving Ansible a number without following one of these rules will end up with a decimal number which will have unexpected results. As of Ansible 1.8, the mode may be specified as a symbolic mode (for example, `u+rwx` or `u=rw,g=r,o=r`). If `mode` is not specified and the destination file **does not** exist, the default `umask` on the system will be used when setting the mode for the newly created file. If `mode` is not specified and the destination file **does** exist, the mode of the existing file will be used. Specifying `mode` is the best way to ensure files are created with the correct permissions. See CVE-2020-1736 for further details.

other_certificates list / elements=path		List of other certificates to include. Pre Ansible 2.8 this parameter was called *ca_certificates*. Assumes there is one PEM-encoded certificate per file. If a file contains multiple PEM certificates, set *other_certificates_parse_all* to `true`. aliases: ca_certificates
other_certificates_parse_all boolean *added in 1.4.0 of community.crypto*	**Choices:** - **no** ← - yes	If set to `true`, assumes that the files mentioned in *other_certificates* can contain more than one certificate per file (or even none per file).
owner string		Name of the user that should own the file/directory, as would be fed to *chown*.
passphrase string		The PKCS#12 password. **Note:** PKCS12 encryption is not secure and should not be used as a security mechanism. If you need to store or send a PKCS12 file safely, you should additionally encrypt it with something else.
path path / required		Filename to write the PKCS#12 file to.
privatekey_passphrase string		Passphrase source to decrypt any input private keys with.
privatekey_path path		File to read private key from.
return_content boolean *added in 1.0.0 of community.crypto*	**Choices:** - **no** ← - yes	If set to `yes`, will return the (current or generated) PKCS#12's content as *pkcs12*.
select_crypto_backend string *added in 1.7.0 of community.crypto*	**Choices:** - **auto** ← - cryptography - pyopenssl	Determines which crypto backend to use. The default choice is `auto`, which tries to use `cryptography` if available, and falls back to `pyopenssl`. If one of *iter_size* or *maciter_size* is used, `auto` will always result in `pyopenssl` to be chosen for backwards compatibility. If set to `pyopenssl`, will try to use the pyOpenSSL (https://pypi.org/project/pyOpenSSL/) library. If set to `cryptography`, will try to use the cryptography (https://cryptography.io/) library.

selevel string		The level part of the SELinux file context. This is the MLS/MCS attribute, sometimes known as the `range`. When set to `_default`, it will use the `level` portion of the policy if available.
serole string		The role part of the SELinux file context. When set to `_default`, it will use the `role` portion of the policy if available.
setype string		The type part of the SELinux file context. When set to `_default`, it will use the `type` portion of the policy if available.
seuser string		The user part of the SELinux file context. By default it uses the `system` policy, where applicable. When set to `_default`, it will use the `user` portion of the policy if available.
src path		PKCS#12 file path to parse.
state string	**Choices:** • absent • **present** ←	Whether the file should exist or not. All parameters except `path` are ignored when state is `absent`.
unsafe_writes boolean *added in 2.2 of ansible.builtin*	**Choices:** • **no** ← • yes	Influence when to use atomic operation to prevent data corruption or inconsistent reads from the target file. By default this module uses atomic operations to prevent data corruption or inconsistent reads from the target files, but sometimes systems are configured or just broken in ways that prevent this. One example is docker mounted files, which cannot be updated atomically from inside the container and can only be written in an unsafe manner. This option allows Ansible to fall back to unsafe methods of updating files when atomic operations fail (however, it doesn't force Ansible to perform unsafe writes). IMPORTANT! Unsafe writes are subject to race conditions and can lead to data corruption.

See Also

See also

community.crypto.x509_certificate (x509_certificate_module.html#ansible-collections-community-crypto-x509-certificate-module)

The official documentation on the **community.crypto.x509_certificate** module.

community.crypto.openssl_csr (openssl_csr_module.html#ansible-collections-community-crypto-openssl-csr-module)

The official documentation on the **community.crypto.openssl_csr** module.

community.crypto.openssl_dhparam (openssl_dhparam_module.html#ansible-collections-community-crypto-openssl-dhparam-module)

The official documentation on the **community.crypto.openssl_dhparam** module.

community.crypto.openssl_privatekey (openssl_privatekey_module.html#ansible-collections-community-crypto-openssl-privatekey-module)

The official documentation on the **community.crypto.openssl_privatekey** module.

community.crypto.openssl_publickey (openssl_publickey_module.html#ansible-collections-community-crypto-openssl-publickey-module)

The official documentation on the **community.crypto.openssl_publickey** module.

Examples

```yaml
- name: Generate PKCS#12 file
  community.crypto.openssl_pkcs12:
    action: export
    path: /opt/certs/ansible.p12
    friendly_name: raclette
    privatekey_path: /opt/certs/keys/key.pem
    certificate_path: /opt/certs/cert.pem
    other_certificates: /opt/certs/ca.pem
    # Note that if /opt/certs/ca.pem contains multiple certificates,
    # only the first one will be used. See the other_certificates_parse_all
    # option for changing this behavior.
    state: present

- name: Generate PKCS#12 file
  community.crypto.openssl_pkcs12:
    action: export
    path: /opt/certs/ansible.p12
    friendly_name: raclette
    privatekey_path: /opt/certs/keys/key.pem
    certificate_path: /opt/certs/cert.pem
    other_certificates_parse_all: true
    other_certificates:
      - /opt/certs/ca_bundle.pem
        # Since we set other_certificates_parse_all to true, all
        # certificates in the CA bundle are included and not just
        # the first one.
      - /opt/certs/intermediate.pem
        # In case this file has multiple certificates in it,
        # all will be included as well.
    state: present

- name: Change PKCS#12 file permission
  community.crypto.openssl_pkcs12:
    action: export
    path: /opt/certs/ansible.p12
    friendly_name: raclette
    privatekey_path: /opt/certs/keys/key.pem
    certificate_path: /opt/certs/cert.pem
    other_certificates: /opt/certs/ca.pem
    state: present
    mode: '0600'

- name: Regen PKCS#12 file
  community.crypto.openssl_pkcs12:
    action: export
    src: /opt/certs/ansible.p12
    path: /opt/certs/ansible.p12
    friendly_name: raclette
    privatekey_path: /opt/certs/keys/key.pem
    certificate_path: /opt/certs/cert.pem
    other_certificates: /opt/certs/ca.pem
    state: present
    mode: '0600'
    force: yes

- name: Dump/Parse PKCS#12 file
  community.crypto.openssl_pkcs12:
    action: parse
    src: /opt/certs/ansible.p12
    path: /opt/certs/ansible.pem
    state: present
```

Search this site

```
- name: Remove PKCS#12 file
  community.crypto.openssl_pkcs12:
    path: /opt/certs/ansible.p12
    state: absent
```

Return Values

Common return values are documented here (../../../reference_appendices/common_return_values.html#common-return-values), the following are the fields unique to this module:

Key	Returned	Description
backup_file string	changed and if *backup* is `yes`	Name of backup file created. **Sample:** /path/to/ansible.com.pem.2019-03-09@11:22~
filename string	changed or success	Path to the generate PKCS#12 file. **Sample:** /opt/certs/ansible.p12
pkcs12 string *added in 1.0.0 of community.crypto*	if *state* is `present` and *return_content* is `yes`	The (current or generated) PKCS#12's content Base64 encoded.
privatekey string	changed or success	Path to the TLS/SSL private key the public key was generated from. **Sample:** /etc/ssl/private/ansible.com.pem

Authors

- Guillaume Delpierre (@gdelpierre)