autossl Documentation

Release 0.9.5

Thibaud Castaing

Sep 14, 2023

Contents

1 How to use autossl module 1

2 Api reference 7

3 Changes 23

4 What 29

5 Installation 31

6 Tests 33

7 Contributing 35

Python Module Index 37

Index 39

CHAPTER 1

How to use autossl module

1.1 Blueprinting

All SSL certificates are blueprinted with a yaml file defining:

- name of the certificate (used to identify easily the certificate)
- information of server (or list of servers) where certificate must be deployed.
- certificate details: type (DV, OV, ..), common name, san, renewal delay, ...
- storage: where artifacts generated will be stored
- tracking: what tracking mechanism will be used to track the operations performed (certificate renewal, deployment, ...)

Note that configuration linked to storage, tracking can be put either in a dedicated blueprint in order to reuse same global config for several certificates or in each certificate blueprint.

- **Certificate blueprint**

```
---
name: tst.example.autossl.com

servers:
  - type: autossl.server.local.LocalServer
    parameters:
      path: /etc/ssl/my_certificates
  - type: autossl.server.local.LocalServer
    parameters:
      path: /etc/ssl_path2/my_certificates

certificate:
  type: DV
  certificate_authority: LetsEncrypt
  common_name: tst.example.autossl.com
```

(continues on next page)

(continued from previous page)

```
    san:
      - tst1.example.autossl.com
      - tst2.example.autossl.com
      - tst3.example.autossl.com
```

- **Global configuration blueprint**

```
---

certificate_authorities:
  - name: Sectigo
    key: Sectigo
    certificate_types: ['OV', 'DV']
    chain_of_trust:
      # intermediate certificate
      - |
        -----BEGIN CERTIFICATE-----
        MIIGGTCCBAGgAwIBAgIQE31TnKp8MamkM3AZaIR6jTANBgkqhkiG9w0BAQwFADCBiDELMAkGA1UE
        BhMCVVMxEzARBgNVBAgTCk5ldyBKZXJzZXkxFDASBgNVBAcTC0plcnNleSBDaXR5MR4wHAYDVQQK
        ExVUaGUgVVNFUlRSVVNUIE5ldHdvcmsxLjAsBgNVBAMTJVVTRVJUcnVzdCBSU0EgQ2VydGlmaWNh
        dGlvbiBBdXRob3JpdHkwHhcNMTgxMTAyMDAwMDAwWhcNMzAxMjMxMjM1OTU5WjCBlTELMAkGA1UE
        BhMCR0IxGzAZBgNVBAgTEkdyZWF0ZXIgTWFuY2hlc3RlcjEQMA4GA1UEBxMHU2FsZm9yZDEYMBYG
        A1UEChMPU2VjdGlnbyBMaW1pdGVkMT0wOwYDVQQDEzRTZWN0aWdvIFJTQSBPcmdhbml6YXRpb24g
        VmFsaWRhdGlvbiBTZWN1cmUgU2VydmVyIENBMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKC
        AQEAnJMCRkVKUkiS/FeN+S3qU76zLNXYqKXsW2kDwB0Q9lkz3v4HSKjojHpnSvH1jcM3ZtAykffE
        nQRgxLVK4oOLp64m1F06XvjRFnG7ir1xon3IzqJgJLBSoDpFUd54k2xiYPHkVpy3O/c8Vdjf1Xox
        fDV/ElFw4Sy+BKzL+k/hfGVqwECn2XylY4QZ4ffK76q06Fha2ZnjJt+OErK43DOyNtoUHZZYQkBu
        CyKFHFEirsTIBkVtkuZntxkj5Ng2a4XQf8dS48+wdQHgibSov4o2TqPgbOuEQc6lL0giE5dQYkUe
        CaXMn2xXcEAG2yDoG9bzk4unMp63RBUJ16/9fAEc2wIDAQABo4IBbjCCAWowHwYDVR0jBBgwFoAU
        U3m/WqorSs9UgOHYm8Cd8rIDZsswHQYDVR0OBBYEFBfZ1iUnZ/kxwklD2TA2RIxsqU/rMA4GA1Ud
        DwEB/wQEAwIBhjASBgNVHRMBAf8ECDAGAQH/AgEAMB0GA1UdJQQWMBQGCCsGAQUFBwMBBggrBgEF
        BQcDAjAbBgNVHSAEFDASMAYGBFUdIAAwCAYGZ4EMAQICMFAGA1UdHwRJMEcwRaBDoEGGP2h0dHA6
        Ly9jcmwudXNlcnRydXN0LmNvbS9VU0VSVHJ1c3RSU0FDZXJ0aWZpY2F0aW9uQXV0aG9yaXR5LmNy
        bDB2BggrBgEFBQcBAQRqMGgwPwYIKwYBBQUHMAKGM2h0dHA6Ly9jcnQudXNlcnRydXN0LmNvbS9V
        U0VSVHJ1c3RSU0FBZGRUcnVzdENBLmNydDAlBggrBgEFBQcwAYYZaHR0cDovL29jc3AudXNlcnRy
        dXN0LmNvbTANBgkqhkiG9w0BAQwFAAOCAgEAThNAlsnD5m5bwOO69Bfhrgkfyb/LDCUW8nNTs3Ya
        t6tIBtbNAHwgRUNFbBZaGxNh10m6pAKkrOjOzi3JKnSj3N6uq9BoNviRrzwB93fVC8+Xq+uH5xWo
        +jBaYXEgscBDxLmPbYox6xU2JPti1Qucj+lmveZhUZeTth2HvbC1bP6mESkGYTQxMD0gJ3NR0N6F
        g9N3OSBGltqnxloWJ4Wyz04PToxcvr44APhL+XJ71PJ616IphdAEutNCLFGIUi7RPSRnR+xVzBv0
        yjTqJsHe3cQhifa6ezIejpZehEU4z4CqN2mLYBd0FUiRnG3wTqN3yhscSPr5z0noX0+FCuKPkBur
        cEya67emP7SsXaRfz+bYipaQ908mgWB2XQ8kd5GzKjGfFlqyXYwcKapInI5v03hAcNt37N3j0VcF
        cC3mSZiIBYRiBXBWdoY5TtMibx3+bfEOs2LEPMvAhblhHrrhFYBZlAyuBbuMf1a+HNJav5fyakyw
        xnB2sJCNwQs2uRHY1ihc6k/+JLcYCpsM0MF8XPtpvcyiTcaQvKZN8rG61ppnW5YCUtCC+cQKXA0o
        4D/I+pWVidWkvklsQLI+qGu41SWyxP7x09fn1txDAXYw+zuLXfdKiXyaNb78yvBXAfCNP6CHMntH
        WpdLgtJmwsQt6j8k9Kf5qLnjatkYYaA7jBU=
        -----END CERTIFICATE-----

  - name: Let's Encrypt
    key: LetsEncrypt
    type: autossl.ca_manager.acme_v2_http01.AcmeHttp01
    certificate_types: ['DV']
    acme_api:
      production: https://acme-v02.api.letsencrypt.org
      staging: https://acme-staging-v02.api.letsencrypt.org
    # specify where acme account key is located
    storage:
      type: autossl.storage.local.LocalFileStorage
```

(continues on next page)

(continued from previous page)

```
      name: lets_encrypt_account_key
      parameters:
        path: /etc/ca_account_keys/
    chain_of_trust:
      # intermediate certificate
      - |
        -----BEGIN CERTIFICATE-----
        MIIEkjCCA3qgAwIBAgIQCgFBQgAAAVOFc2oLheynCDANBgkqhkiG9w0BAQsFADA/MSQwIgYDVQQK
        ExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMTDkRTVCBSb290IENBIFgzMB4X
        DTE2MDMxNzE2NDA0NloXDTIxMDMxNzE2NDA0NlowSjELMAkGA1UEBhMCVVMxFjAUBgNVBAoTDUxl
        dCdzIEVuY3J5cHQxIzAhBgNVBAMTGkxldCdzIEVuY3J5cHQgQXV0aG9yaXR5IFgzMIIBIjANBgkq
        hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAnNMM8FrlLke3cl03g7NoYzDq1zUmGSXhvb418XCSL7e4
        S0EFq6meNQhY7LEqxGiHC6PjdeTm86dicbp5gWAf15Gan/PQeGdxyGkOlZHP/uaZ6WA8SMx+yk13
        EiSdRxta67nsHjcAHJyse6cF6s5K671B5TaYucv9bTyWaN8jKkKQDIZ0Z8h/pZq4UmEUEz9l6YKH
        y9v6Dlb2honzhT+Xhq+w3Brvaw2VFn3EK6BlspkENnWAa6xK8xuQSXgvopZPKiAlKQTGdMDQMc2P
        MTiVFrqoM7hD8bEfwzB/onkxEz0tNvjj/PIzark5McWvxI0NHWQWM6r6hCm21AvA2H3DkwIDAQAB
        o4IBfTCCAXkwEgYDVR0TAQH/BAgwBgEB/wIBADAOBgNVHQ8BAf8EBAMCAYYwfwYIKwYBBQUHAQEE
        czBxMDIGCCsGAQUFBzABhiZodHRwOi8vaXNyZy50cnVzdGlkLm9jc3AuaWRlbnRydXN0LmNvbTA7
        BggrBgEFBQcwAoYvaHR0cDovL2FwcHMuaWRlbnRydXN0LmNvbS9yb290cy9kc3Ryb290Y2F4My5w
        N2MwHwYDVR0jBBgwFoAUxKexpHsscfrb4UuQdf/EFWCFiRAwVAYDVR0gBE0wSzAIBgZngQwBAgEw
        PwYLKwYBBAGC3xMBAQEwMDAuBggrBgEFBQcCARYiaHR0cDovL2Nwcy5yb290LXgxLmxldHNlbmNy
        eXB0Lm9yZzA8BgNVHR8ENTAzMDGgL6AthitodHRwOi8vY3JsLmlkZW50cnVzdC5jb20vRFNUUk9P
        VENBWDNDUkwuY3JsMB0GA1UdDgQWBBSoSmpjBH3duubRObemRWXv86jsoTANBgkqhkiG9w0BAQsF
        AAOCAQEA3TPXEfNjWDjdGBX7CVW+dla5cEilaUcne8IkCJLxWh9KEik3JHRRHGJouM2VcGfl96S8
        TihRzZvoroed6ti6WqEBmtzw3Wodatg+VyOeph4EYpr/1wXKtx8/wApIvJSwtmVi4MFU5aMqrSDE
        6ea73Mj2tcMyo5jMd6jmeWUHK8so/joWUoHOUgwuX4Po1QYz+3dszkDqMp4fklxBwXRsW10KXzPM
        TZ+sOPAveyxindmjkW8lGy+QsRlGPfZ+G6Z6h7mjem0Y+iWlkYcV4PIWL1iwBi8saCbGS5jN2p8M
        +X+Q7UNKEkROb3N6KOqkqm57TH2H3eDJAkSnh6/DNFu0Qg==
        -----END CERTIFICATE-----
      # root certificate
      - |
        -----BEGIN CERTIFICATE-----
        MIIDSjCCAjKgAwIBAgIQRK+wgNajJ7qJMDmGLvhAazANBgkqhkiG9w0BAQUFADA/MSQwIgYDVQQK
        ExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMTDkRTVCBSb290IENBIFgzMB4X
        DTAwMDkzMDIxMTIxOVoXDTIxMDkzMDE0MDExNVowPzEkMCIGA1UEChMbRGlnaXRhbCBTaWduYXR1
        cmUgVHJ1c3QgQ28uMRcwFQYDVQQDEw5EU1QgUm9vdCBDQSBYMzCCASIwDQYJKoZIhvcNAQEBBQAD
        ggEPADCCAQoCggEBAN+v6ZdQCINXtMxiZfaQguzH0yxrMMpb7NnDfcdAwRgUi+DoM3ZJKuM/IUmT
        rE4Orz5Iy2Xu/NMhD2XSKtkyj4zl93ewEnu1lcCJo6m67XMuegwGMoOifooUMM0RoOEqOL15CjH9
        UL2AZd+3UWODyOKIYepLYYHsUmu5ouJLGiifSKOeDNoJjj4XLh7dIN9bxiqKqy69cK3FCxolkHRy
        xXtqqzTWMIn/5WgTe1QLyNau7Fqckh49ZLOMxt+/yUFw7BZy1SbsOFU5Q9D8/RhcQPGX69Wam40d
        utolucbY38EVAjqr2m7xPi71XAicPNaDaeQQmxkqtilX4+U9m5/wAl0CAwEAAaNCMEAwDwYDVR0T
        AQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMCAQYwHQYDVR0OBBYEFMSnsaR7LHH62+FLkHX/xBVghYkQ
        MA0GCSqGSIb3DQEBBQUAA4IBAQCjGiybFwBcqR7uKGY3Or+Dxz9LwwmglSBd49lZRNI+DT69ikug
        dB/OEIKcdBodfpga3csTS7MgROSR6cz8faXbauX+5v3gTt23ADq1cEmv8uXrAvHRAosZy5Q6XkjE
        GB5YGV8eAlrwDPGxrancWYaLbumR9YbK+rlmM6pZW87ipxZzR8srzJmwN0jP41ZL9c8PDHIyh8bw
        RLtTcm1D9SZImlJnt1ir/md2cXjbDaJWFBM5JDGFoqgCWjBH4d1QB7wCCZAA62RjYJsWvIjJEubS
        fZGL+T0yjWW06XyxV3bqxbYoOb8VZRzI9neWagqNdwvYkQsEjgfbKbYK7p2CNTUQ
        -----END CERTIFICATE-----


organization:
  company_name: Autossl corporation
  street_address: Newbury street
  city: Boston
  state: Massachusetts
  postal_code: '02115'
  country_code: US
```

(continues on next page)

(continued from previous page)

```
storage:
  type: autossl.storage.gitscm.GitStorage
  credentials: git_credentials
  parameters:
    git_url: https://git.autossl.com/autossl/my_certs.git
  data:
    # type of data to store/retrieve in this storage
    - type: key
    - type: csr
    - type: crt

tracking:
  type: autossl.tracking.local.LocalFileTracking
  parameters:
    log_folder: /var/log/ssl_logs
  data:
    - type: yaml
    - type: csr
    - type: crt

credentials:
  git_credentials:
    type: UserPassword

...
```

1.2 Command line options

All commands accepts the following options

- *–config* (optional) is the global blueprint yaml file
- *–blueprint* is the certificate blueprint yaml file

Both *–config* and *–blueprint* files can also be merged in a single blueprint and in that case use only *–blueprint* option. If same section (*tracking*, *storage*, . . .) appears in both global config and certificate blueprint, global config is ignored and section from certificate blueprint will be used

1.3 Monitoring

The *check* action allow to monitor certificates deployed on servers and provide status.

```
$ autossl \
  --config global_config.yaml \
  --blueprint example.autossl.com.yaml check
INFO:autossl:Processing blueprint example.autossl.com.yaml
INFO:autossl.server.base:[LocalServer - AUTOSSL_MACHINE:/etc/ssl_path_1] - example.
↪autossl.com - 2019-05-20T17:37:28 => valid (42 days remaining)
INFO:autossl.ssl:Following domains not covered by certificate: [new.example.autossl.
↪com]
INFO:autossl.manager:Certificate definition changed for 'example.autossl.com' on␣
↪server '[LocalServer - AUTOSSL_MACHINE:/etc/ssl_path_1]'
```

(continues on next page)

(continued from previous page)

```
INFO:autossl.server.base:[LocalServer - AUTOSSL_MACHINE:/etc/ssl_path_2] - example.
↪autossl.com - 2019-05-20T17:37:28 => valid (42 days remaining)
INFO:autossl.ssl:Following domains not covered by certificate: [new.example.autossl.
↪com]
INFO:autossl.manager:Certificate definition changed for 'example.autossl.com' on␣
↪server '[LocalServer - AUTOSSL_MACHINE:/etc/ssl_path_2]'
```

1.4 Renewal

Process to renew certificate is the same, whatever the CA used (Sectigo, Let's Encrypt, ...) or the type of certificate requested. Renewal can be requested for 1 or several blueprints.

Depending on the type of certificate requested and the CA, automated certificate renewal may or not be possible.

For each blueprint, the flow starts with the following:

- compare blueprint with stored certificate: checking for close expiration, change of certificate content
- compare blueprint with existing certificate on the server(s): same checks than before + track servers with missing certificate
- generate a csr based on blueprint
- call tracking api and send it specified files in config (generally blueprint and CSR)
- then, when supported by specified CA, certificate is generated automatically with CA specified renewal method protocol (see details below) and also sent to tracking api

```
$ autossl \
  --blueprint --blueprint example.autossl.com.yaml \
  renew --force\
INFO:autossl.ssl_manager:Processing blueprint example.autossl.com.yaml
INFO:autossl.ssl_manager:Force renewal for 'auto_example.autossl.com'
Continue ? (y/n)y
INFO:autossl.ssl_manager:Start renewal process for certificate 'auto_example.autossl.
↪com'
INFO:autossl.ssl_manager:Tracking record created: TR 98765432: SSL certificate for␣
↪example.autossl.com
INFO:autossl.ssl_manager:Processing blueprint example.autossl.com.yaml
INFO:autossl.manager:Start renewal process for certificate 'example.autossl.com.yaml'
INFO:autossl.acme.acme_manager:Parsing account key...
INFO:autossl.acme.acme_manager:Registering account...
INFO:autossl.acme.acme_manager:Already registered!
INFO:autossl.acme.acme_manager:Starting validation for domain example.autossl.com
INFO:autossl.server.local:Deploy challenge on LocalServer AUTOSSL_MACHINE:/etc/acme_
↪dir
INFO:autossl.acme.acme_manager:example.autossl.com verified!
INFO:autossl.server.local:Cleanup challenge from LocalServer AUTOSSL_MACHINE:/etc/
↪acme_dir
INFO:autossl.acme.acme_manager:Signing certificate...
INFO:autossl.acme.acme_manager:Certificate signed
```

1.5 Deployment

To perform the deployment, several information are required: - certificate - private key - ssl blueprint - tracking record ID (optional)

All those information can be directly given in command line or can be retrieved directly from configured storage and/or tracking record.

1) from tracking record and blueprint (or global config)

At least global config is needed to identify tracking type and retrieve data from specified tracking record. If only global config specified, full blueprint must be attached to tracking record to know where to deploy this certificate.

```
$ autossl --config global_config.yaml deploy --tracking-record 12345678
```

2) with all information from command line

```
$ autossl --config global_config.yaml deploy \
   --private-key example.autossl.com.key \
   --certificate example.autossl.com.crt \
   --tracking-record 12345678
```

These commands will:

- retrieve all needed artifacts (yaml blueprint, new certificate, ...) if not already given in command line
- ensure certificate is compatible with yaml blueprint, private key, CA certificate chain
- deploy key+certificate in all servers listed in yaml blueprint
- update tracking record with status of the deployment and set it as completed

```
$ autossl --config global_config.yaml deploy \
   --tracking-record 98765432 \
   --private-key /etc/keys/example.autossl.com.key
INFO:autossl.manager:Blueprint: example.autossl.com.yaml
INFO:autossl.manager:Certificate: example.autossl.com.crt
INFO:autossl.manager:PrivateKey: example.autossl.com.key
INFO:autossl.server.base:[LocalServer - slave-ql6n8] - example.autossl.com - 2019-07-
↪10T08:43:29 => valid (90 days remaining)
INFO:autossl.server.local:Certificate/Key example.autossl.com updated successfully on␣
↪[LocalServer - AUTOSSL_MACHINE:/etc/ssl_path_1].
INFO:autossl.server.local:Certificate/Key example.autossl.com updated successfully on␣
↪[LocalServer - AUTOSSL_MACHINE:/etc/ssl_path_2].
```

- global config is needed here to know how to retrieve tracking record specified
- –private-key is the path to the certificate private key (can also be retrieved automatically from configured storage or tracking record)
- –certificate is the path to the new certificate (can also be retrieved automatically from configured storage or tracking record)
- –tracking-record is the tracking record created in renewal step above

Note that using tracking record is optional, and you can directly give certificate blueprint, private key and SSL certificate in input of deploy.

CHAPTER 2

Api reference

2.1 autossl.ca_manager

class autossl.ca_manager.base.**CaManager**(*ca_config*, *staging=True*, *storage_api=None*, ***kwargs*)

Bases: object

get_signed_certificate(*ssl_blueprint=None*, *csr_path=None*, *servers_api=None*)

Get PEM encoded certificate using current Certificate Authority implementation

Parameters

- **ssl_blueprint** (ssl.SslBlueprint) –
- **csr_path** (*pathlib.Path*) – path to CSR file
- **servers_api** (*list(server.base.Server)*) – list of api instances to each server

Returns PEM encoded signed certificate as bytes

Return type bytes

is_automated_renewal_supported

Check is current CA supports automated renewal

Returns True, if this CA implementation supports automated renewal

Return type bool

class autossl.ca_manager.acme_v2_http01.**AcmeHttp01**(*ca_config*, *staging=True*, *storage_api=None*, ***kwargs*)

Bases: *autossl.ca_manager.base.CaManager*

get_signed_certificate(*ssl_blueprint=None*, *csr_path=None*, *servers_api=None*)

Get PEM encoded certificate using current Certificate Authority implementation

Parameters

- **ssl_blueprint** (ssl.SslBlueprint) –

- **csr_path** (*pathlib.Path*) – path to CSR file
- **servers_api** (*list(server.base.Server)*) – list of api instances to each server

Returns PEM encoded signed certificate as bytes

Return type bytes

is_automated_renewal_supported
Check is current CA supports automated renewal

Returns True, if this CA implementation supports automated renewal

Return type bool

class autossl.ca_manager.local.**LocalCa**(*ca_config*, *staging=True*, *storage_api=None*, *ca_private_key=None*, *ca_certificate=None*, *certificate_validity_days=90*, ***kwargs*)
Bases: *autossl.ca_manager.base.CaManager*

Class implementing a certificate authority based on a private key retrieved from CA storage

get_signed_certificate(*ssl_blueprint=None*, *csr_path=None*, *servers_api=None*)
Get PEM encoded certificate using current Certificate Authority implementation

Parameters

- **ssl_blueprint** (ssl.SslBlueprint) –
- **csr_path** (*pathlib.Path*) – path to CSR file
- **servers_api** (*list(server.base.Server)*) – list of api instances to each server

Returns PEM encoded signed certificate as bytes

Return type bytes

is_automated_renewal_supported
Check is current CA supports automated renewal

Returns True, if this CA implementation supports automated renewal

Return type bool

2.2 autossl.server

class autossl.server.base.**Server**(*crt_name*, *deploy_full_chain=False*, ***kwargs*)
Bases: object

create_acme_challenge(*token*, *key_authorization*)
Create token on server with specified value

Parameters

- **token** – challenge key
- **key_authorization** – challenge value

delete_acme_challenge(*token*)
Delete challenge created on server

Parameters **token** (*str*) – challenge key to delete from server

deploy_cert(*key*, *cert*, ***kwargs*)
Deploy input certificate on server

Parameters

- **key** (*pathlib.Path*) – path to local private key
- **cert** (*pathlib.Path*) – path to local public certificate

Raises ***exception.DeployCertificateError*** – if unexpected error occurred during deployment on server

get_certificate_information()
Retrieve certificate information from server.

Must be implemented for each type of server.

Returns SSL certificate information

Return type *autossl.ssl.SslCertificate*

Raises ***autossl.exception.CertificateNotFound*** – if certificate does not exist yet on server

get_description()
Get description of this server

Returns server description

Return type str

is_expired(*expiration_delay=0*)
Check for expiration of specified certificate

Parameters **expiration_delay** (*int*) – Number of days before real expiration we consider a renewal needed

Returns True is certificate is going to expire in less than expiration_delay days

Return type bool

is_same(*common_name=None*, *sans=None*, *exact_match=False*)
Check if current certificate deployed on server is covering all specified domains

Parameters

- **common_name** (*str*) – Common name
- **sans** (*list*) – list of Subject Alternate Names
- **exact_match** (*bool*) – if True, certificate must exactly match input domains if False, input domain will also match wilcard certificate and additional domains in certificate will be ignored

Returns True is certificate is already covering all domains

class autossl.server.local.**LocalServer**(*crt_name*, *path*, *acme_dir=None*, ***kwargs*)
Bases: *autossl.server.base.Server*

create_acme_challenge(*token*, *key_authorization*)
Create token on server with specified value

Parameters

- **token** – challenge key
- **key_authorization** – challenge value

delete_acme_challenge(*token*)
Delete challenge created on server

Parameters **token** (*str*) – challenge key to delete from server

deploy_cert(*key*, *cert*, ***kwargs*)
Deploy input certificate on server

Parameters

- **key** (*pathlib.Path*) – path to local private key
- **cert** (*pathlib.Path*) – path to local public certificate

Raises ***exception.DeployCertificateError*** – if unexpected error occurred during deployment on server

get_certificate_information()
Retrieve certificate information from server.

Must be implemented for each type of server.

Returns SSL certificate information

Return type *autossl.ssl.SslCertificate*

Raises ***autossl.exception.CertificateNotFound*** – if certificate does not exist yet on server

get_description()
Get description of this server

Returns server description

Return type str

2.3 autossl.storage

class autossl.storage.base.**Storage**(*tracking_record_id=None*, ***kwargs*)
Bases: object

retrieve_data(*name*, *data_type*, ***kwargs*)
Retrieve data from storage

Parameters

- **name** (*str*) – identifier of data to retrieve
- **data_type** (*ssl.DataType*) – type of data to retrieve
- ****kwargs** (*dict*) – optional key/value parameters from blueprint to retrieve data

Returns requested data

Return type bytes

Raises ***exception.NotFound*** – when requested data are missing in storage

save_data(*name*, *data_type*, *content=None*, *local_path=None*, ***kwargs*)
Save specified content in storage

Parameters

- **name** (*str*) – name of the content to be stored on server side

- **data_type** (ssl.DataType) – type of data to save
- **content** (*bytes*) – content to be stored on server side
- **local_path** (*pathlib.Path or str*) – local path to a file to store
- ****kwargs** (*dict*) – optional key/value parameters from blueprint to save data

Either one of *content* or *local_path* must be specified but not both

class autossl.storage.local.**LocalFileStorage**(*path*, *tracking_record_id=None*, ***kwargs*)

Bases: *autossl.storage.base.Storage*

retrieve_data(*name*, ***kwargs*)

Retrieve data from storage

Parameters

- **name** (*str*) – identifier of data to retrieve
- **data_type** (ssl.DataType) – type of data to retrieve
- ****kwargs** (*dict*) – optional key/value parameters from blueprint to retrieve data

Returns requested data

Return type bytes

Raises ***exception.NotFound*** – when requested data are missing in storage

save_data(*name*, *content=None*, *local_path=None*, ***kwargs*)

Save specified content in storage

Parameters

- **name** (*str*) – name of the content to be stored on server side
- **data_type** (ssl.DataType) – type of data to save
- **content** (*bytes*) – content to be stored on server side
- **local_path** (*pathlib.Path or str*) – local path to a file to store
- ****kwargs** (*dict*) – optional key/value parameters from blueprint to save data

Either one of *content* or *local_path* must be specified but not both

class autossl.storage.gitscm.**GitStorage**(*git_url*, *folder=None*, *tracking_record_id=None*, *config_user_name=None*, *config_user_email=None*, ***kwargs*)

Bases: *autossl.storage.base.Storage*

retrieve_data(*name*, ***kwargs*)

Retrieve data from storage

Parameters

- **name** (*str*) – identifier of data to retrieve
- **data_type** (ssl.DataType) – type of data to retrieve
- ****kwargs** (*dict*) – optional key/value parameters from blueprint to retrieve data

Returns requested data

Return type bytes

Raises ***exception.NotFound*** – when requested data are missing in storage

save_data(*name*, *content=None*, *local_path=None*, ***kwargs*)
Save specified content in storage

Parameters

- **name** (*str*) – name of the content to be stored on server side
- **data_type** (ssl.DataType) – type of data to save
- **content** (*bytes*) – content to be stored on server side
- **local_path** (*pathlib.Path or str*) – local path to a file to store
- ****kwargs** (*dict*) – optional key/value parameters from blueprint to save data

Either one of *content* or *local_path* must be specified but not both

autossl.storage.gitscm.**git_url_with_username_password**(*git_url*, *username*, *password*)

2.4 autossl.tracking

class autossl.tracking.base.**Tracking**(*ssl_blueprint_path*, ***kwargs*)
Bases: object

close_for_failure(*message*)
Specify action is completed with a failed status

Parameters **message** (*str*) – custom message

close_for_success(*message*)
Specify action is completed with a success status

Parameters **message** (*str*) – custom message

create(*tracking_type*, *servers=None*)
Create a tracking record with details of current SSL blueprint

Parameters

- **tracking_type** (TrackingType) – Type of tracking. Can be used to customized tracking record content.
- **servers** (*list*) – List of servers in scope of the action. All servers from config if None specified here.

Returns Identifier for the created record

Return type str

refresh(*record_id*)
Update current tracking instance with last changes from tracking record on server side

Parameters **record_id** – identifier of the record to refresh

retrieve_data(*name=None*, *data_type=None*, ***kwargs*)
Retrieve specified data from tracking system

Parameters

- **name** (*str*) – Name of file/data to retrieve
- **data_type** (ssl.DataType) – type of data to retrieve
- ****kwargs** – generic key/value parameters

Returns file content

Return type bytes

save_data(*name*, *data_type*, *local_path=None*, *content=None*, ***kwargs*)
Save input data in tracking system

Parameters

- **name** (*str*) – name of the file to attach to the tracking record
- **data_type** (ssl.DataType) – type of data to save
- **local_path** (*pathlib.Path*) – local path to file to attach to the tracking record
- **content** (*bytes*) – content of the file to attach to the tracking record
- ****kwargs** – generic key/value parameters

update(*message*)
Update tracking record

Parameters **message** (*str*) – text to add to tracking record

class autossl.tracking.base.**TrackingType**
Bases: enum.Enum

list of tracking types supported.

Renewal = 'renewal'

Synchronize = 'synchronize'

2.5 autossl.credential

class autossl.credential.**CredentialType**
Bases: enum.Enum

list of credentials types supported

ApiKeyAndId = 'api_key_and_api_id'

UserPassword = 'user_password'

autossl.credential.**get_api_key_and_id**(*name*, *credentials=None*, *separator=None*)

autossl.credential.**get_credentials**(*name*, *global_config*, *credentials*, *extra_parameters=None*)
Get structured form of specified credential based on its type and ready to be passed to any api

Parameters

- **name** (*str*) – name of the credential
- **global_config** (*dict*) – credential global configuration
- **credentials** (*dict*) – structured credentials dict
- **extra_parameters** (*dict*) – extra parameters to add to current credential

Returns structured credentials

Return type dict

autossl.credential.**get_user_password**(*name*, *credentials=None*, *separator=None*)

2.6 autossl.exception

exception autossl.exception.**AutoSslException**(*msg*, *original_exception=None*)
Bases: Exception

Generic exception for autossl

Allow to chain exceptions keeping track of origin exception

exception autossl.exception.**CertificateNotFound**(*msg*, *original_exception=None*)
Bases: *autossl.exception.NotFound*

Requested certificate not present on server

exception autossl.exception.**DefinitionMismatch**(*msg*, *original_exception=None*)
Bases: *autossl.exception.InvalidCertificate*

Certificate is not matching blueprint definition

exception autossl.exception.**DeployCertificateError**(*msg*, *original_exception=None*)
Bases: *autossl.exception.AutoSslException*

Unexpected error when trying to deploy new certificate

exception autossl.exception.**ExpiredCertificate**(*msg*, *original_exception=None*)
Bases: *autossl.exception.InvalidCertificate*

Certificate is expiring

exception autossl.exception.**HttpCodeException**(*request_exception*)
Bases: *autossl.exception.AutoSslException*

exception autossl.exception.**InvalidCertificate**(*msg*, *original_exception=None*)
Bases: *autossl.exception.AutoSslException*

Certificate is not matching expected criteria

exception autossl.exception.**InvalidTrustChain**(*msg*, *original_exception=None*)
Bases: *autossl.exception.InvalidCertificate*

Certificate is not compatible with CA certificate specified

exception autossl.exception.**KeyMismatch**(*msg*, *original_exception=None*)
Bases: *autossl.exception.InvalidCertificate*

Certificate does not match private key

exception autossl.exception.**NotFound**(*msg*, *original_exception=None*)
Bases: *autossl.exception.AutoSslException*

Requested data not found

exception autossl.exception.**SslBlueprintInconsistency**(*msg*, *original_exception=None*)
Bases: *autossl.exception.AutoSslException*

SSL blueprint definition contains inconsistencies

2.7 autossl.manager

Script to check and renew automatically SSL certificates on a server

class autossl.manager.**SslManager**(*global_config=None*, *blueprint_path=None*, *credentials=None*, *staging=True*)

Bases: object

deploy(*tracking_record_id=None*, *certificate_path=None*, *private_key_path=None*, *all_servers=False*)

Deploy certificate/key on servers

if certificate/key file are specified in input, they will be used, else they will be retrieved from configured storage.

If tracking record identifier is specified, certificate can also be retrieved from there, and this record will be used to track the change. If no tracking record specified, a new one will be created

Parameters

- **tracking_record_id** (*str*) – tracking record identifier
- **certificate_path** (*pathlib.Path*) – local path to SSL certificate file
- **private_key_path** (*pathlib.Path*) – local path to SSL certificate private key
- **all_servers** (*bool*) – if True, deploy certificate/key on all configured servers, else only out of synch servers will be updated.

deploy_certificate(*key_path*, *crt_path*, *servers_list*)

Deploy input SSL certificate on servers

Parameters

- **key_path** (*pathlib.Path*) – path to private key This is optional, if not provided, private key will be automatically retrieved from SecretServer
- **crt_path** (*pathlib.Path*) – path to certificate
- **servers_list** – list of server configuration on which to deploy the certificate.

get_and_check_artifacts(*tracking_record_id=None*, *certificate_path=None*, *private_key_path=None*, *folder=None*)

Retrieve currently stored certificate/key and check if valid for deployment

Parameters

- **tracking_record_id** (*str or None*) – tracking record identifier
- **certificate_path** (*pathlib.Path or None*) – local path to SSL certificate file. Automatically retrieved if not specified.
- **private_key_path** (*pathlib.Path or None*) – local path to SSL certificate private key. Automatically retrieved if not specified.
- **folder** (*pathlib.Path or None*) – folder where artifacts will be stored.

Returns tuple of (certificate path, private key path)

Return type tuple(pathlib.Path, pathlib.Path)

get_ca_manager_api()

get_ca_storage_api()

get_certificate_information(*working_directory*)

Retrieve certificate information for the blueprint.

Parameters **working_directory** (*pathlib.Path*) – directory in which the ssl certificate will be downloaded

Returns SSL certificate information

Return type *autossl.ssl.SslCertificate*

Raises **autossl.exception.NotFound** – if certificate does not exist in storage

get_file(*file_type*, *file_identifier*, *output_folder*, *output_filename=None*, *api_names=None*)
Retrieve specified stored data

Parameters

- **file_type** (ssl.DataType) – type of data to retrieve
- **file_identifier** (*str*) – identifier of the data to retrieve
- **output_folder** (*pathlib.Path*) – which folder content will be written
- **output_filename** (*str*) – name of file to write (default: same than 'file_identifier' parameter)
- **api_names** (*list*) – list of api in which to search data

Returns local file path to the retrieved content

Return type pathlib.Path

get_renewal_status()
Get details status of the certificate for each server from blueprint: expired, modified, missing, ...

Returns a 2-tuple with (Boolean renewal needed, Array servers to update)

Return type tuple

The checks performed are the following

1) it is a new certificate
2) cert is close to expiration
3) cert definition has been modified (ex: new san)
4) new server has been added

get_server_api(*server_parameters*)

get_storage_api()

get_tracking_api()

renew(*force=False*)
Request a renewal and proceed with automated renewal right after (if applicable)

Parameters **force** (*bool*) – request renewal even if not needed

renew_certificate()
Perform automated renewal of the certificate using ACME protocol

Will interact with the CA to validate ownership of the domains using ACME protocol. In case of any error, input TR will be automatically closed as rejected and exception logged in that TR In case of success, certificate is directly attached to the TR

request_renewal(*force=False*)
Request renewal of the certificate for specified blueprint

it is first checking that a renewal is needed. Then it is generating a new CSR for the specified blueprint. A new tracking record is created with CSR and blueprint attached If automated renewal is supported, certificate is generated automatically with CA and attached to TR Else, TR is simply sent to 'SSL Certificate Service' team

Parameters **force** (*bool*) – request renewal even if not needed

Returns True if a renewal is needed

Return type bool

save_file(*file_type*, *file_path=None*, *file_content=None*, *api_names=None*)
Save specified content wherever it is configured in blueprint

Parameters

- **file_type** (ssl.DataType) – type of data to save
- **file_path** (*pathlib.Path*) – path to a local file to save
- **file_content** (*bytes*) – content to save
- **api_names** (*list*) – list of api in which to save data

Raises **IOError** – if none of 'file_path' or 'file_content' parameter are specified

2.8 autossl.ssl

class autossl.ssl.**CertificateAuthorityConfig**(*certificate_authorities*, *certificate_authority_key*)

Bases: object

get_acme_api(*staging=False*)

get_chain_of_trust()
Return list of certificate to have full chain of trust: intermediate, root :return: list of certificate starting intermediate until root certificate :rtype: list

get_storage_config()
Get configuration of CA storage api

Returns CA storage configuration

Return type dict

get_supported_certificate_types()
Get list of certificate types currently supported by CA

Returns list of certificate types currently supported by CA

Return type list

is_acme_supported()
Check if CA supports ACME protocol

Returns True if CA supports ACME protocol

Return type bool

is_certificate_supported(*cert_type*)
Check if specified certificate type is supported by CA

Parameters **cert_type** (*str*) – type of certificate to check (ex: DV)

Returns True if CA supports this certificate type

Return type bool

class autossl.ssl.**DataType**
Bases: enum.Enum

list of data types supported

Blueprint = 'yaml'

Certificate = 'crt'

CertificateSigningRequest = 'csr'

PrivateKey = 'key'

class autossl.ssl.**SslBlueprint**(*yaml_path=None*, *global_config_path=None*)
Bases: object

domains
Get domains covered by this blueprint

Returns list of domains in blueprint

Return type set

get_chain_of_trust()
Return list of certificates to have full chain of trust: intermediate, root :return: list of certificates starting intermediate until root certificate :rtype: list

get_config(*name*, *path=None*, *default=None*)

validate()
Validate data extracted from blueprint

Raises **ValueError** – if content of specified blueprint is not valid

class autossl.ssl.**SslCertificate**(*x509_path=None*, *common_name=None*, *sans=None*, *expiration=None*)
Bases: object

domains

init_from_x509(*x509_path*)

Parameters **x509_path** (*pathlib.Path*) – path to PEM certificate

is_expired(*expiration_delay=0*)
Check for expiration

Parameters **expiration_delay** (*int*) – Number of days before real expiration we consider a renewal needed

Returns True is certificate is going to expire in less than expiration_delay days

Return type bool

is_same(*common_name=None*, *sans=None*, *exact_match=False*)
Check if current certificate is covering all specified domains

Parameters

- **common_name** (*str*) – Common name
- **sans** (*list*) – list of Subject Alternate Names

- **exact_match** (*bool*) – if True, certificate must exactly match input domains if False, input domain will also match wilcard certificate and additional domains in certificate will be ignored

Returns True is certificate is already covering all domains

class autossl.ssl.**SslCertificateConfig**(*certificate_type*, *certificate_authority*, *common_name=None*, *sans=None*, *organization=None*, *chain_of_trust=None*, *exact_match=False*, *private_key_reuse=False*, *private_key_size=2048*, *renewal_delay=30*, *is_ca=False*)

Bases: object

domains

set_attr_if_not_none(*attr_name*, *value*)
Set attribute value if value is not None

Parameters

- **attr_name** (*str*) – attribute name
- **value** – attribute value

validate(*ca_config*)

autossl.ssl.**check_certificate_with_key**(*key_path*, *crt_path*)
Check whether a private key matches a certificate

For this, we compare RSAPublicNumbers from public key in certificate with the RSAPublicNumbers which makes up the RSA public key associated with this RSA private key.

Parameters

- **key_path** (*pathlib.Path*) – path to private key
- **crt_path** (*pathlib.Path*) – path to SSL certificate

Returns True, if certificate matches private key

Return type bool

autossl.ssl.**check_chain_of_trust**(*chain_of_trust*, *crt_path*)
Check that input certificate matches chain of trust

Parameters

- **chain_of_trust** (*list*) – list of certificates of the chain of trust (intermediate CA, root CA)
- **crt_path** (*pathlib.Path*) – local path to certificate to verify

Raises ***exception.InvalidTrustChain*** – if input certificate does not match chain of trust specified

autossl.ssl.**generate_csr**(*name*, *common_name=None*, *company_name=None*, *street_address=None*, *city=None*, *state=None*, *postal_code=None*, *country_code=None*, *email_address=None*, *sans=None*, *key_content=None*, *key_size=2048*, *output_path=None*, *is_ca=False*)
Generate a CSR for specified parameters

if a private key is given, it will be used to generate CSR, else a new one will be created

Parameters

- **name** – name of file generated (without extension)

- **common_name** – common name
- **company_name** – company name
- **street_address** – company street address
- **city** – company city
- **state** – company state
- **postal_code** – company postal code
- **country_code** – company country
- **email_address** – contact email
- **sans** – list of SANs to be covered
- **key_content** (*byte*) – optional private key content to generate CSR
- **key_size** – size of private key to generate CSR, if no key in input
- **output_path** – local path where to generate files
- **is_ca** – True if the requested certificate is for a CA

Returns tuple(key_content, csr_path) with content of private key and path to csr file

Return type tuple(bytes, pathlib.Path)

autossl.ssl.**get_domains**(*common_name=None*, *sans=None*)

Get unique list of domains for input criteria

Parameters

- **common_name** (*str or None*) – Certificate common name
- **sans** (*list(str) or None*) – Certificate SANs List

Returns unique list of domains

Return type set(str)

autossl.ssl.**get_domains_from_x509**(*file_path*, *file_type*)

Retrieve the list of domains covered by specified x509 file (CSR or CRT)

Parameters

- **file_path** (*pathlib.Path*) – path to x509 file
- **file_type** (DataType) – type of x509 file. Supported types: [DataType.CertificateSigningRequest, DataType.Certificate]

Returns list of domain

Return type set

autossl.ssl.**get_expiration**(*crt_path*)

autossl.ssl.**is_domain_list_matching**(*domains_to_check*, *reference_domains*, *exact_match=False*)

Check if a list of domains are covered by another list of domains

For example, test.example.com and test2.example.com are covered by *.example.com

Parameters

- **domains_to_check** – list of domains to check
- **reference_domains** – list of reference domains to compare with

- **exact_match** – If True, domains_to_check and reference_domains must be the same If False, domains_to_check can be only a subset of reference_domains

Returns True if domains_to_check are covered by reference_domains

Return type bool

autossl.ssl.**is_domain_matching**(*domain_to_check*, *reference_domain*, *exact_match=False*)
Check if a domain is matching another domain

For example, test.example.com is matching by *.example.com

Parameters

- **domain_to_check** – the domain to check
- **reference_domain** – the reference domain to compare with
- **exact_match** – If True, domain_to_check and reference_domain must be the same If False, domain_to_check can be only a subset of reference_domain

Returns True if domain_to_check is matching reference_domain

Return type bool

autossl.ssl.**sign**(*csr*, *ca_key*, *ca_cert*, *validity_days*)
Sign a certificate request with a key (CA)

Parameters

- **csr** (*bytes, PEM encoded*) – certificate request to sign
- **ca_key** (*bytes, PEM encoded*) – the signing key
- **ca_cert** (*bytes, PEM encoded*) – the signing certificate
- **validity_days** (*int*) – certificate validity duration (in days)

Returns the signed certificate

Return type bytes, PEM encoded

2.9 autossl.util

class autossl.util.**TempDir**(*path=None*)
Bases: object

autossl.util.**check_http_response_ok**(*response*)
Validate http response code

all codes not in 2xx will raise an exception

Parameters **response** (*requests.Response*) – requests Http response

Returns same http response

Return type requests.Response

Raises ***exception.HttpCodeException*** – if http status code in not in 2xx

autossl.util.**str_to_class**(*class_path*)
Dynamically import and return class type from full module and class path

Parameters **class_path** (*str*) –

Returns Type of the class to instantiate

Return type type

Raises

- **ImportError** – if module does not exist
- **AttributeError** – if class not found in specified module

CHAPTER 3

Changes

3.1 0.9.0 (20/01/2020)

- First public release
- ACMEv1 support removed (v1 is deprecated since quite some time and will be disabled for new domains in June 2020)

3.2 0.8.0 (09/01/2020)

- [Feature] support for ACMEv2 HTTP01 challenge
- [Feature] ability to read credentials from environment variables
- [Improvement] migrate from simple string path to pathlib for easier path manipulation
- [Improvement] use byte as default encoding rather than str to avoid useless conversions
- [Improvement] use relative imports
- [Improvement] add missing doc

3.3 0.7.5 (16/12/2019)

- [Bug] ACME http01 - CA account key was deleted too early in the process while was still needed for renewal

3.4 0.7.4 (12/12/2019)

- [Improvement]: move acme section from blueprint directly in CA configuration
- [Improvement]: remove dedicated acme_storage and only use 1 generic storage class for CA manager

3.5 0.7.3 (11/12/2019)

- [Improvement]: avoid leaking credentials in git storage logs

3.6 0.7.2 (06/12/2019)

- [Bug]: fix invalid attribute in acme_v1_http01 renewal

3.7 0.7.1 (05/12/2019)

- [Bug]: fix module import on initialization

3.8 0.7.0 (05/12/2019)

- [Improvement]: generic CA managers for automatic certificate renewal, the type is determined from *type* attribute in blueprint CA. Supported values are : *autossl.ca_manager.acme_v1_http01.AcmeHttp01* and *autossl.ca_manager.local.LocalCa*
- [Feature]: certificate signing from a CA key/certificate available in storage

3.9 0.6.0 (29/11/2019)

- [Feature]: support deployment of full certificate chain on any type of server
- [Feature]: when chain of trust is specified (in global config or ssl blueprint) always verify it before deployment

3.10 0.5.6 (19/11/2019)

- [Bug]: to_be_renewed flag returned by *manager.get_renewal_status* was still true when stored certificate was valid if at least 1 server was invalid
- [Feature]: *manager.get_and_check_artifacts* ability to try first retrieving artifacts from tracking when tracking ID specified, as we consider tracking the most up to date in that case

3.11 0.5.5 (18/11/2019)

- [Feature] support for CA certificates request

3.12 0.5.4 (12/11/2019)

- [Improvement]: python 3.8 support
- [Bug]: credentials in input of storage.gitscm were ignored, directly add them in input git url for http

3.13 0.5.3 (12/11/2019)

- DO NOT USE

3.14 0.5.2 (06/11/2019)

- [Bug]: fix equality operator on SslCertificate object as sans comparison must ignore sorting
- [Bug]: do not directly compare certificates in SslManager and always call 'is_same' from Server api as each server can customize/override its behavior

3.15 0.5.1 (29/10/2019)

- [Technical]: no change

3.16 0.5.0 (14/10/2019)

- [Improvement]: support certificates without servers

3.17 0.4.4 (22/08/2019)

- [Bug]: chain of trust in global config was ignored
- [Feature] add retries in case Incapsula return internal error at deployment (as happening quite often)

3.18 0.4.3 (05/08/2019)

- [Bug]: fix acme *_get_new_challenge* raising decoding error in python 3

3.19 0.4.2 (26/07/2019)

- [Bug]: fix deployment of full certificate chain of trust on Incapsula for python3 (base64 encoding issue)

3.20 0.4.1 (25/07/2019)

- [Bug]: Certificates deployed on Incapsula server type must contain the full chain – root CA , intermediate CA, and the origin server certificates, this is now default behavior for Incapsula and can be activated for any type of server.

3.21 0.4.0 (13/05/2019)

- [Feature]: plugins now have access to file type (Certificate, private key, ...) when retrieving/saving data in storage and tacking to be able to customize behavior.

3.22 0.3.3 (06/05/2019)

- [Bug]: fix exception raised when Incaspula site has no SSL certificate deployed yet

3.23 0.3.2 (29/04/2019)

- [Improvement]: add email to 'Subject' section of the certificate

3.24 0.3.1 (26/04/2019)

- [Improvement]: better error handling during certificate deployment: deploy everywhere possible and report errors in tracking record
- [Improvement]: sanitize Incapsula tests removing all Amadeus specifics
- [Improvement]: update documentation

3.25 0.3.0 (15/03/2019)

- [Bug]: deploy from record was failing as looking first for data in storage without caching exception
- [Feature]: automatically save data in other apis when found only in a specific one
- [Feature]: support tracking in local file to more easily test different api orchestrations
- [Feature]: Support Incapsula server type
- [Feature]: ability to deploy existing certificate to outdated or new servers thanks to synchronize option
- [Feature]: support storage of credentials in local file ~/.autossl in ini format

3.26 0.2.3 (19/02/2019)

- [Bug]: host is an optional parameter in server configuration + fix for credential enum
- [Feature]: possibility to specify only global config without certificate information for deployment to retrieve blueprint from storage/tracking

3.27 0.2.2 (14/02/2019)

- [Bug]: fix package delivery: unable to access subpackages from external module

3.28 0.2.1 (14/02/2019)

- [Bug]: fix package delivery (missing subfolders)
- [Feature]: add 'version' option in command line to display package information

3.29 0.2.0 (13/02/2019)

- [Feature] make autossl generic to support any type of storage for artifacts persistency and tracking mechanism

3.30 0.1.13 (07/02/2019)

- [Improvement]: Do not block renewal if challenge cannot be verified from local machine as validation will be performed anyway by Certificate Authority

3.31 0.1.12 (07/02/2019)

- [Improvement]: Do not block renewal if server is not reachable from local machine

3.32 0.1.11 (19/12/2018)

- [Improvement]: Remove prompt when renew is called with force option

3.33 0.1.10 (19/12/2018)

- [Improvement]: Modify delivery to ensure proper artifact publication

3.34 0.1.9 (19/12/2018)

- [Improvement]: Return tracking record Id when applicable in ssl_manager.renew method

3.35 0.1.8 (10/12/2018)

- [Improvement]: support custom certificate filename for each server

3.36 0.1.7 (13/09/2018)

- [Improvement]: allow no servers section specified in ssl blueprint to just manage certificate request without server interaction

3.37 0.1.6 (27/08/2018)

- [Improvement]: add possibility to use any servers type. No automatic checks for now, they will always generate new certificates

3.38 0.1.5 (06/07/2018)

- [Improvement]: add retry capability in case of connection error for all http connections

3.39 0.1.4 (06/06/2018)

- [Improvement]: add retry capability to challenge creation/deletion in case of connection error for automated certificate renewal

3.40 0.1.3 (24/04/2018)

- [Bug]: fix default parameters for command line

3.41 0.1.2 (24/04/2018)

- [Bug]: add missing entry point for command line in setup.py

3.42 0.1.1 (11/04/2018)

- [Bug]: fix config files missing in package delivery

3.43 0.1.0 (06/04/2018)

- First delivery

AutoSSL Python module to automate SSL certificates monitoring, renewal and deployment

Copyright Copyright (c) 2019 Amadeus sas

License MIT

Documentation https://autossl.readthedocs.io

Development https://github.com/AmadeusITGroup/AutoSSL

CHAPTER 4

What

autossl is a module for Python 2.7+/3.5+ that can be used to to automate SSL certificate monitoring, renewal and deployment.

This module can be customized with plugins mechanism to support any type of:

- **server**: where the certificate is deployed, can be 1 or more server, potentially of different types
- **storage**: where to store your artifacts (private key, public certificate, . . .)
- **tracking mechanism**: how to track renewal process (ticket creation)
- **renewal method**: how to get a new certificate (local CA, ACME protocol, ...)

It's providing a command line interface with simple actions: *check*, *renew*, *deploy*. All configuration is provided thanks to blueprints in Yaml

It can then be run by any tool able to use a command line (cron, jenkins pipeline, . . .) to manage all your certificates from a central place.

CHAPTER 5

Installation

For a basic installation, just run

```
$ pip install autossl
```

to support optional features, you may need extra dependencies, for that install autossl with corresponding *keyword*:

```
$ pip install autossl[keyword]
```

See available *keywords* and associated extra dependencies in table below:

keyword	additional dependencies	extra features
all	all packages below	all features below
acme	acme	renewal using ACME protocol
git	GitPython	artifacts storage in git repository

CHAPTER 6

Tests

tests require few more python packages. To install them, run:

```
$ pip install -r requirements_dev.txt
```

Clone the repository, then to execute the test suite with your current python version, run:

```
$ pytest -sv tests
```

CHAPTER 7

Contributing

7.1 Bug Reports

Bug reports are hugely important! Before you raise one, though, please check through the GitHub issues, both open and closed, to confirm that the bug hasn't been reported before.

7.2 Feature Requests

If you think a feature is missing and could be useful in this module, feel free to raise a feature request through the GitHub issues

7.3 Code Contributions

When contributing code, please follow this project-agnostic contribution guide.

Python Module Index

a

autossl.ca_manager.acme_v2_http01, 7
autossl.ca_manager.base, 7
autossl.ca_manager.local, 8
autossl.credential, 13
autossl.exception, 14
autossl.manager, 14
autossl.server.base, 8
autossl.server.local, 9
autossl.ssl, 17
autossl.storage.base, 10
autossl.storage.gitscm, 11
autossl.storage.local, 11
autossl.tracking.base, 12
autossl.util, 21

Index

A

AcmeHttp01 (*class in autossl.ca_manager.acme_v2_http01*), 7
ApiKeyAndId (*autossl.credential.CredentialType attribute*), 13
autossl.ca_manager.acme_v2_http01 (*module*), 7
autossl.ca_manager.base (*module*), 7
autossl.ca_manager.local (*module*), 8
autossl.credential (*module*), 13
autossl.exception (*module*), 14
autossl.manager (*module*), 14
autossl.server.base (*module*), 8
autossl.server.local (*module*), 9
autossl.ssl (*module*), 17
autossl.storage.base (*module*), 10
autossl.storage.gitscm (*module*), 11
autossl.storage.local (*module*), 11
autossl.tracking.base (*module*), 12
autossl.util (*module*), 21
AutoSslException, 14

B

Blueprint (*autossl.ssl.DataType attribute*), 18

C

CaManager (*class in autossl.ca_manager.base*), 7
Certificate (*autossl.ssl.DataType attribute*), 18
CertificateAuthorityConfig (*class in autossl.ssl*), 17
CertificateNotFound, 14
CertificateSigningRequest (*autossl.ssl.DataType attribute*), 18
check_certificate_with_key() (*in module autossl.ssl*), 19
check_chain_of_trust() (*in module autossl.ssl*), 19
check_http_response_ok() (*in module autossl.util*), 21
close_for_failure() (*autossl.tracking.base.Tracking method*), 12
close_for_success() (*autossl.tracking.base.Tracking method*), 12
create() (*autossl.tracking.base.Tracking method*), 12
create_acme_challenge() (*autossl.server.base.Server method*), 8
create_acme_challenge() (*autossl.server.local.LocalServer method*), 9
CredentialType (*class in autossl.credential*), 13

D

DataType (*class in autossl.ssl*), 18
DefinitionMismatch, 14
delete_acme_challenge() (*autossl.server.base.Server method*), 8
delete_acme_challenge() (*autossl.server.local.LocalServer method*), 9
deploy() (*autossl.manager.SslManager method*), 15
deploy_cert() (*autossl.server.base.Server method*), 8
deploy_cert() (*autossl.server.local.LocalServer method*), 10
deploy_certificate() (*autossl.manager.SslManager method*), 15
DeployCertificateError, 14
domains (*autossl.ssl.SslBlueprint attribute*), 18
domains (*autossl.ssl.SslCertificate attribute*), 18
domains (*autossl.ssl.SslCertificateConfig attribute*), 19

E

ExpiredCertificate, 14

G

generate_csr() (*in module autossl.ssl*), 19
get_acme_api() (*autossl.ssl.CertificateAuthorityConfig method*), 17
get_and_check_artifacts() (*autossl.manager.SslManager method*), 15

get_api_key_and_id() (*in module autossl.credential*), 13
get_ca_manager_api() (*autossl.manager.SslManager method*), 15
get_ca_storage_api() (*autossl.manager.SslManager method*), 15
get_certificate_information() (*autossl.manager.SslManager method*), 15
get_certificate_information() (*autossl.server.base.Server method*), 9
get_certificate_information() (*autossl.server.local.LocalServer method*), 10
get_chain_of_trust() (*autossl.ssl.CertificateAuthorityConfig method*), 17
get_chain_of_trust() (*autossl.ssl.SslBlueprint method*), 18
get_config() (*autossl.ssl.SslBlueprint method*), 18
get_credentials() (*in module autossl.credential*), 13
get_description() (*autossl.server.base.Server method*), 9
get_description() (*autossl.server.local.LocalServer method*), 10
get_domains() (*in module autossl.ssl*), 20
get_domains_from_x509() (*in module autossl.ssl*), 20
get_expiration() (*in module autossl.ssl*), 20
get_file() (*autossl.manager.SslManager method*), 16
get_renewal_status() (*autossl.manager.SslManager method*), 16
get_server_api() (*autossl.manager.SslManager method*), 16
get_signed_certificate() (*autossl.ca_manager.acme_v2_http01.AcmeHttp01 method*), 7
get_signed_certificate() (*autossl.ca_manager.base.CaManager method*), 7
get_signed_certificate() (*autossl.ca_manager.local.LocalCa method*), 8
get_storage_api() (*autossl.manager.SslManager method*), 16
get_storage_config() (*autossl.ssl.CertificateAuthorityConfig method*), 17
get_supported_certificate_types() (*autossl.ssl.CertificateAuthorityConfig method*), 17
get_tracking_api() (*autossl.manager.SslManager method*), 16
get_user_password() (*in module autossl.credential*), 13
git_url_with_username_password() (*in module autossl.storage.gitscm*), 12
GitStorage (*class in autossl.storage.gitscm*), 11

H

HttpCodeException, 14

I

init_from_x509() (*autossl.ssl.SslCertificate method*), 18
InvalidCertificate, 14
InvalidTrustChain, 14
is_acme_supported() (*autossl.ssl.CertificateAuthorityConfig method*), 17
is_automated_renewal_supported (*autossl.ca_manager.acme_v2_http01.AcmeHttp01 attribute*), 8
is_automated_renewal_supported (*autossl.ca_manager.base.CaManager attribute*), 7
is_automated_renewal_supported (*autossl.ca_manager.local.LocalCa attribute*), 8
is_certificate_supported() (*autossl.ssl.CertificateAuthorityConfig method*), 17
is_domain_list_matching() (*in module autossl.ssl*), 20
is_domain_matching() (*in module autossl.ssl*), 21
is_expired() (*autossl.server.base.Server method*), 9
is_expired() (*autossl.ssl.SslCertificate method*), 18
is_same() (*autossl.server.base.Server method*), 9
is_same() (*autossl.ssl.SslCertificate method*), 18

K

KeyMismatch, 14

L

LocalCa (*class in autossl.ca_manager.local*), 8
LocalFileStorage (*class in autossl.storage.local*), 11
LocalServer (*class in autossl.server.local*), 9

N

NotFound, 14

P

PrivateKey (*autossl.ssl.DataType attribute*), 18

R

refresh() (*autossl.tracking.base.Tracking method*), 12

renew() (*autossl.manager.SslManager method*), 16
renew_certificate() (*autossl.manager.SslManager method*), 16
Renewal (*autossl.tracking.base.TrackingType attribute*), 13
request_renewal() (*autossl.manager.SslManager method*), 16
retrieve_data() (*autossl.storage.base.Storage method*), 10
retrieve_data() (*autossl.storage.gitscm.GitStorage method*), 11
retrieve_data() (*autossl.storage.local.LocalFileStorage method*), 11
retrieve_data() (*autossl.tracking.base.Tracking method*), 12

S

save_data() (*autossl.storage.base.Storage method*), 10
save_data() (*autossl.storage.gitscm.GitStorage method*), 11
save_data() (*autossl.storage.local.LocalFileStorage method*), 11
save_data() (*autossl.tracking.base.Tracking method*), 13
save_file() (*autossl.manager.SslManager method*), 17
Server (*class in autossl.server.base*), 8
set_attr_if_not_none() (*autossl.ssl.SslCertificateConfig method*), 19
sign() (*in module autossl.ssl*), 21
SslBlueprint (*class in autossl.ssl*), 18
SslBlueprintInconsistency, 14
SslCertificate (*class in autossl.ssl*), 18
SslCertificateConfig (*class in autossl.ssl*), 19
SslManager (*class in autossl.manager*), 14
Storage (*class in autossl.storage.base*), 10
str_to_class() (*in module autossl.util*), 21
Synchronize (*autossl.tracking.base.TrackingType attribute*), 13

T

TempDir (*class in autossl.util*), 21
Tracking (*class in autossl.tracking.base*), 12
TrackingType (*class in autossl.tracking.base*), 13

U

update() (*autossl.tracking.base.Tracking method*), 13
UserPassword (*autossl.credential.CredentialType attribute*), 13

V

validate() (*autossl.ssl.SslBlueprint method*), 18
validate() (*autossl.ssl.SslCertificateConfig method*), 19