

## How to use ACME unittests code

ACME unittests code를 사용하는 방법에 대한 문서(Linux)



## **Table of Contents**

- l . 환경 설정
- II. config.py
- Ⅲ. runTest.py와 test 코드
- Ⅳ. 실제 사용 화면

# CHAPTER '

환경 설정

#### 1. 환경 설정

### 1. OS 환경



- OS 환경
- ✔ Ubuntu 20.04.4 LTS 버전을 사용한다.

(단, OS의 경우 특별한 제약이 있는 것은 아님)

```
cumulus@cumulus:~$ cat /etc/*release*
DISTRIB ID=Ubuntu
DISTRIB RELEASE=20.04
DISTRIB CODENAME=focal
DISTRIB DESCRIPTION="Ubuntu 20.04.4 LTS"
NAME="Ubuntu"
VERSION="20.04.4 LTS (Focal Fossa)"
ID=ubuntu
ID LIKE=debian
PRETTY_NAME="Ubuntu 20.04.4 LTS"
VERSION ID="20.04"
HOME URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG REPORT URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION CODENAME=focal
UBUNTU CODENAME=focal
```

#### 1. 환경 설정

### 2. python 가상 환경 생성 및 실행



- module 설치를 위한 가상 환경
- ✔ 가상 환경을 설치할 디렉토리를 생성한 후 이동한다.

```
cumulus@cumulus:~$ mkdir venvscumulus@cumulus:~$ cd venvscumulus@cumulus:~/venvs$ []
```

- ✔ 아래 명령어를 사용하여 python 가상 환경을 생성한다.
  - cumulus@cumulus:~/venvs\$ python3 -m venv acmecumulus@cumulus:~/venvs\$ ls acmecumulus@cumulus:~/venvs\$ []
- ✓ 그 후 가상환경을 실행하여 모듈을 설치할 준비를 한다.
  - cumulus@cumulus:~/venvs\$ source acme/bin/activate
    (acme) cumulus@cumulus:~/venvs\$ []

### 3. ACME에 필요한 모듈 설치



- o python 모듈 설치
- ✔ 필요한 모듈을 설치하기 위해 아래 명령어를 입력한다.

```
    (acme) cumulus@cumulus:~$ cd ACME-oneM2M-CSE/
    (acme) cumulus@cumulus:~/ACME-oneM2M-CSE$ pip3 install -r requirements.txt
```

✔ 다음 단계를 위하여 "tests" 디렉토리로 이동한다.

```
(acme) cumulus@cumulus:~/ACME-oneM2M-CSE$ cd tests
(acme) cumulus@cumulus:~/ACME-oneM2M-CSE/tests$ ls
               앞으로 사용하게 될
pycache
                                     testREQ.pv
config.py
                 python파일들
                                     testRemote.py
init.py
                                     testRemote Annc.py
                                     testRemote GRP.py
mattReauests
                  testFCNT.pv
runTests.py
                  testFCNT FCI.py
                                     testRemote Requests.py
testACP.py
                  testGRP.py
                                     testRequests.py
testACTR.py
                  testLoad.py
                                     testSUB.py
testAE.py
                  testMgmtObj.py
                                     testTS.py
testAddressing.pv
                  testMisc.py
                                     testTSB.py
testCIN.py
                  testNOD.py
                                     testTS_TSI.py
testCNT.py
                  testPCH.py
                                     testUpperTester.py
testCNT_CIN.py
                  testPCH_PCU.py
(acme) cumulus@cumulus:~/ACME-oneM2M-CSE/tests$ ||
```

# CHAPTER

2

config.py

### 1. config.py 코드 구조



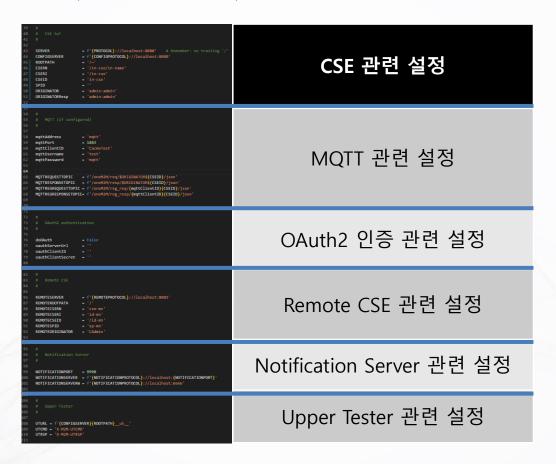
- o config.py 코드 구조
- ✔ BINDING 변수를 찾아 사용할 프로토콜에 맞추어 설정한다(단, 코드의 버전업에 따라 변수의 위치가 변할 수 있음).

```
BINDING
                               = 'http'
                                            이번 문서에서는
12
     if BINDING == 'mqtt':
                                           HTTP 프로토콜을
                                          사용할 예정이므로
                               = 'mqtt'
                                                                  https, mqtt
        PROTOCOL
                                          다음과 같이 기입함
                               = 'http'
        CONFIGPROTOCOL
        NOTIFICATIONPROTOCOL
                               = 'http'
                                           # possible values: http, https, mqtt
        REMOTEPROTOCOL
                               = 'http'
     elif BINDING == 'http':
                                          # possible values: http, https, mqtt
21
        PROTOCOL
                               = 'http'
        CONFIGPROTOCOL
                               = 'http'
                                          # possible values: http, https, mqtt
                                          # possible values: http, https, mqtt
        NOTIFICATIONPROTOCOL
                               = 'http'
                                          # possible values: http, https, mqtt
                               = 'http'
        REMOTEPROTOCOL
     elif BINDING == 'https':
                                           # possible values: http, https, mqtt
        PROTOCOL
                               = 'https'
                                           # possible values: http, https, mqtt
        CONFIGPROTOCOL
                               = 'https'
                                           # possible values: http, https, mqtt
                               = 'http'
        NOTIFICATIONPROTOCOL
        REMOTEPROTOCOL
                               = 'http'
```

## 1. config.py 코드 구조



- o config.py 코드 구조
- ✔ 그 후 사용하게 될 CSE의 정보, MQTT 관련 정보, OAuth2를 사용할 경우 등에 대한 설정이 나타난다.



## 2. config.py 설정



- o config.py 코드 설정
- ✔ 이번 테스트를 위한 설정으로 CSE 서버와 관련된 정보만 바꿔주도록 하였다.

이때, 각 정보의 기입란에는 반드시 끝에 '/'를 적지 않도록 한다.

```
CSE SuT
                        = f'{PROTOCOL}://localhost:8080' # Remember: no trailing '/'
     SERVER
                        = f'{CONFIGPROTOCOL}://localhost:8080'
    CONFIGSERVER
45
    ROOTPATH
                        = '/in-cse/in-name'
46
    CSERN
                        = '/in-cse'
    CSERI
                        = 'in-cse'
48
    CSEID
49
    SPID
                        = 'admin:admin'
    ORIGINATOR
50
     ORIGINATORResp
                        = 'admin:admin'
```

## CHAPTER

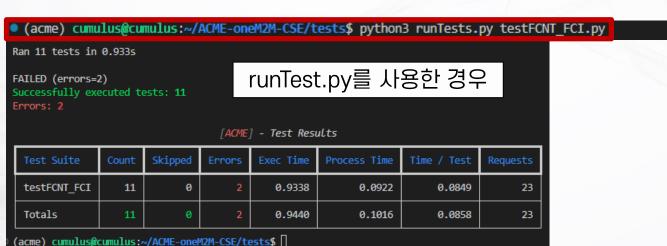
3

runTest.py와 test 코드

### 1. runTest.py

- o runTest.py란?
- ✔ unittest 코드를 실행하고 결과를 보기 쉽게 정리해주는 코드로 크게 수정할 필요는 없다.



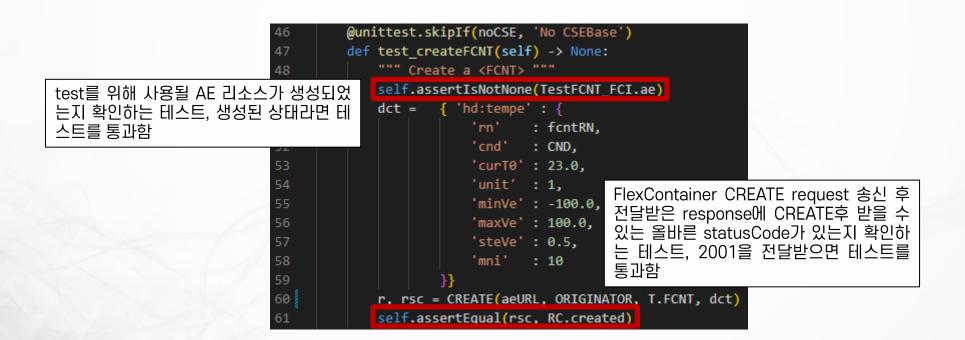


- testFCNT\_FCl.py
- ✔ 이 문서에서는 FlexContainer의 FlexContainerInstance가 올바르게 구현되었는지에 대한 unittest를 실시한다.
- ✓ 그러나 모든 unittest 코드들은 ACME를 기준으로 작성되었기 때문에 직접 개발한 oneM2M 플랫폼에는
  올바르지 않게 동작할 가능성이 존재한다.
- ✔ 때문에 test 코드의 간단한 구조와 디버깅 시 자신의 oneM2M 플랫폼에 맞출 수 있는 방법을 설명하겠다.

- testFCNT\_FCl.py
- ✓ main 코드를 보면 run 함수를 실행하는 것을 알 수 있고 run 함수에 어떤 테스트를 실시할 것인지
  TestFCNT\_FCI의 인자를 보고 직관적으로 알 수 있다.

```
# TODO other FCNT controlling attributes
      # TODO Add similar tests from testCNT CIN for mni, etc
248
      def run(testVerbosity:int, testFailFast:bool) -> Tuple[int, int, int]:
          suite = unittest.TestSuite()
250
          suite.addTest(TestFCNT_FCI('test_createFCNT'))
          suite.addTest(TestFCNT FCI('test attributesFCNT'))
          suite.addTest(TestFCNT FCI('test updateFCNT'))
          suite.addTest(TestFCNT_FCI('test_retrieveFCNTLa01'))
          suite.addTest(TestFCNT FCI('test updateFCNTMni'))
          suite.addTest(TestFCNT_FCI('test_updateLBL'))
          suite.addTest(TestFCNT FCI('test updateMNInoFCICreated'))
          suite.addTest(TestFCNT_FCI('test_createFCIFail'))
258
          suite.addTest(TestFCNT FCI('test updateFCIFail'))
          suite.addTest(TestFCNT FCI('test updateFCNTMniNull'))
          suite.addTest(TestFCNT_FCI('test_deleteFCNT'))
          result = unittest.TextTestRunner(verbosity=testVerbosity, failfast=testFailFast).run(suite)
          printResult(result)
          return result.testsRun, len(result.errors + result.failures), len(result.skipped)
      if name == ' main ':
          _, errors, _ = run(2, True)
          sys.exit(errors)
270
```

- testFCNT\_FCl.py
- ✓ 전달한 인자를 가지고 수행할 함수를 찾게 된다.예를 들어 'test\_createFCNT'함수를 살펴보면 아래와 같다.
- ✔ 테스트 통과 유무는 붉은색 표시가 된 self.assert [조건]에서 조건이 참이 되면 테스트를 통과하게 된다.



- testFCNT\_FCl.py
- ✔ 만약 테스트를 수행할 때 oneM2M 표준 문서에서 설명한대로 잘 구현했지만, 테스트 통과 Error가 발생할 경우, 이것이 ACME에 의존적인 부분이라 판단된다면 해당 부분을 주석 처리하거나 따로 수정하여 테스트를 진행하면 됨

```
@unittest.skipIf(noCSE, 'No CSEBase')
def test_attributesFCNT(self) -> None:
    """ Validate <FCNT> attributes """
   r, rsc = RETRIEVE(fcntURL, ORIGINATOR)
    self.assertEqual(rsc, RC.OK, r)
    self.assertEqual(findXPath(r, 'hd:tempe/ty'), T.FCNT, r)
   self.assertEqual(findXPath(r, 'hd:tempe/pi'), findXPath(TestFCNT FCI.ae,'m2m:ae/ri'), r)
   self.assertEqual(findXPath(r, 'hd:tempe/rn'), fcntRN, r)
   self.assertIsNotNone(findXPath(r, 'hd:tempe/ct'), r)
   self.assertIsNotNone(findXPath(r, 'hd:tempe/lt'), r)
   self.assertIsNotNone(findXPath(r, 'hd:tempe/et'), r)
   #self.assertIsNotNone(findXPath(r, 'hd:tempe/st'), r)
   seit.assertiswone(tindxPath(r, nd:tempe/cr), r)
                                                                     단순히 예시를 위한 코드로
   self.assertEqual(findXPath(r, 'hd:tempe/cnd'), CND, r)
                                                                     주석 처리한 코드가 실제로
   self assertEqual(findYDath(r 'hd:temne/curTA') '23 A', r)
   ##self.assertIsNone(findXPath(r, 'hd:tempe/tarTe'), r)
                                                                     ACME에 의존적인 부분은
   selt.assertEqual(findXPath(r, 'hd:tempe/unit'), '1', r)
   self.assertEqual(findXPath(r, 'hd:tempe/minVe'), '-100.0', r)
   self.assertEqual(findXPath(r, 'hd:tempe/maxVe'), '100.0', r)
   self.assertEqual(findXPath(r, 'hd:tempe/steVe'), '0.5', r)
   #self.assertIsNotNone(findXPath(r, 'hd:tempe/st'), r)
   #self.assertEqual(findXPath(r, 'hd:tempe/st'), 0, r)
   self.assertIsNotNone(findXPath(r, 'hd:tempe/mni'), r)
    self.assertEqual(findXPath(r, 'hd:tempe/mni'), 10, r)
   self.assertIsNotNone(findXPath(r, 'hd:tempe/cni'), r)
   self.assertEqual(findXPath(r, 'hd:tempe/cni'), 1, r)
   self assertIsNotNone(findXDath(r 'hd:temne/chs') r)
    #self.assertGreater(findXPath(r, 'hd:tempe/cbs'), 0, r
```

# CHAPTER '

4

## 실제 사용 화면

#### Ⅳ. 실제 사용 화면

### 1. 결과

- 결과
- ✔ 11개의 테스트가 성공했고 2개의 Error 상황이 발생하였다.
- ✔ 이때 유의할 점은 이 test code는 ACME의 구현에 의존하여 개발되었기 때문에 Error를 출력했다고 하여 반드시 개발한 oneM2M 플랫폼이 잘못 구현된 것은 아니다.
- ✓ refer. <a href="https://github.com/seslabSJU/OM2M\_FCIN\_TESTSUITE">https://github.com/seslabSJU/OM2M\_FCIN\_TESTSUITE</a>

