使用元数据对与API和Python的自定义报告

目录

简介

本文描述如何与在Python脚本内的API为了自定义报告一道使用元数据。

先决条件

要求

Cisco 建议您了解以下主题：

- CloudCenter
- Python

使用的组件

本文档不限于特定的软件和硬件版本。

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您使用的是真实网络，请确保您已经了解所有命令的潜在影响。

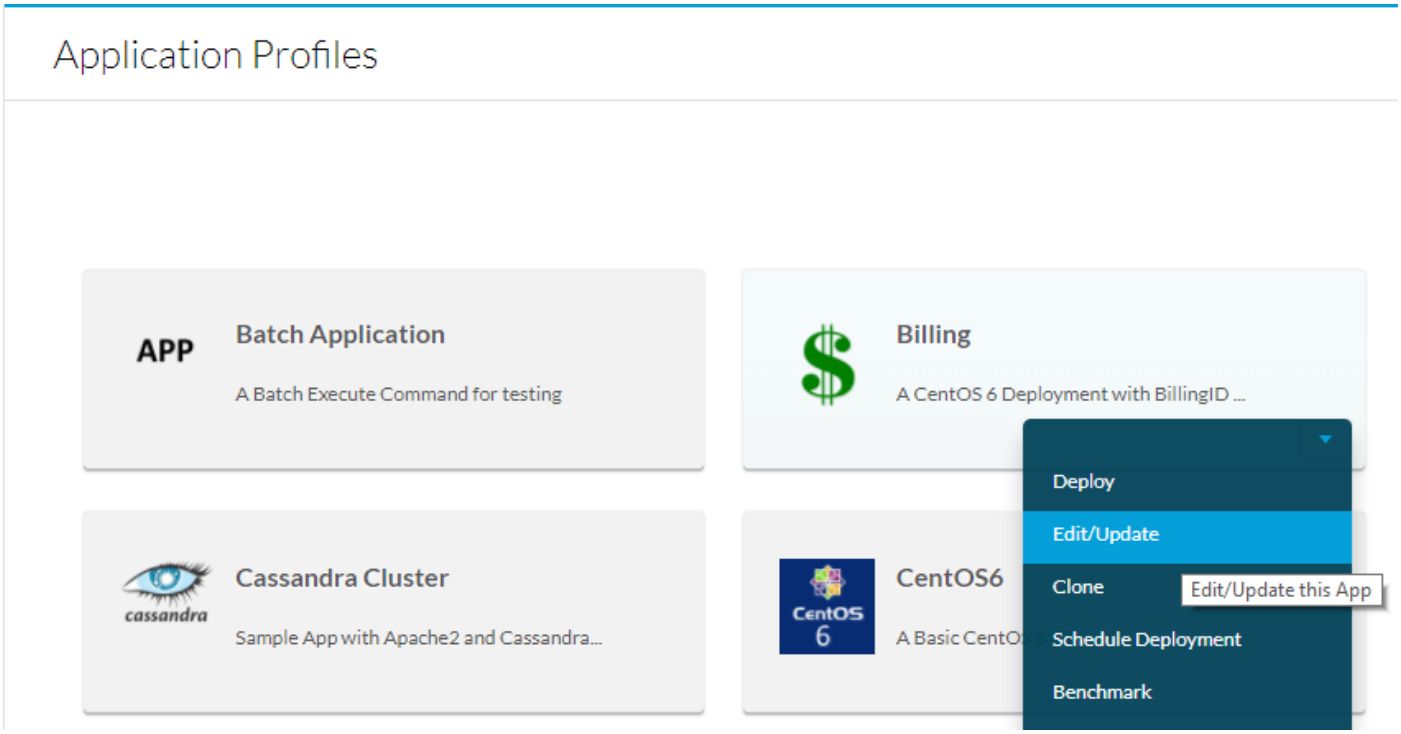
背景信息

CloudCenter提供报告的一些箱外，然而不允许方式报告根据自定义过滤器。为了使用API为了与元数据一道获取信息直接地从数据库，附加对工作，您能允许自定义报告。

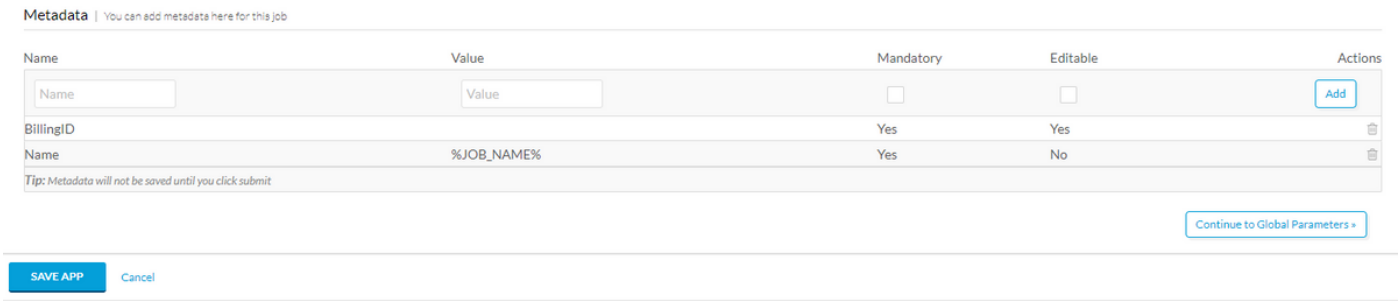
设置元数据

在a必须添加元数据每应用级，那么需要跟踪与使用自定义报告将必须被修改的每应用程序。

为了执行此，导航到**应用配置文件**，然后选择下拉式能将编辑的App的然后选择如镜像所显示的请**编辑/更新**。



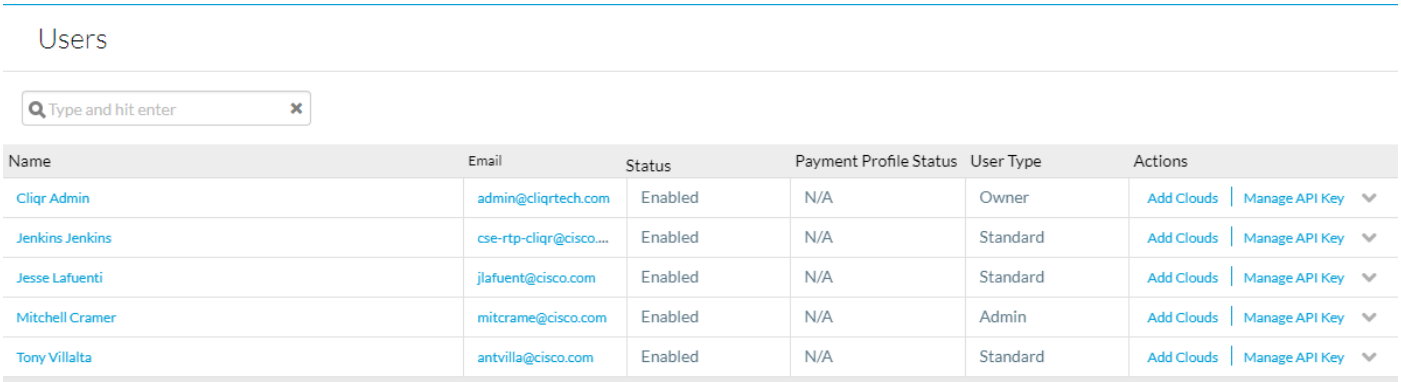
移动到**基本信息的**底部并且添加元数据标记，例如**BillingID**，如果此元数据将由suer填好使必须和编辑可能。如果它是宏，则请填写默认值，并且请勿使编辑可能。在您填好元数据后，如镜像所显示，请选择**添加**然后**保存App**。



采集API密钥

为了处理API呼叫，用户名和API密钥将要求。这些密钥提供同样水平访问象用户，因此，如果所有用户部署将被添加在报告，推荐为了获得承租人API密钥的admin。如果广泛子承租人将一起被记录，对所有的根承租人需要访问部署环境或者所有子承租人admins API密钥将要求。

要获得API密钥请导航对**Admin > Users >管理API锁上**，复制用户名并且为用户锁上要求。



创建自定义报告

在您创建创建报道的Python脚本前，请保证Python和小核安装对此。然后运行**小核安装制成表**，制成表是该的库自动地格式化报告的把柄。

两示例报告在表里附加到此指南，第一收集关于所有部署的信息然后输出它。第二使用同一信息创建与使用的一自定义报告BillingID元数据。此脚本详细解释使用作为指南。

```
import datetime
import json
import sys
import requests
##pip install tabulate
from tabulate import tabulate
from operator import itemgetter
from decimal import Decimal
```

时间用于准确地计算日期，这执行创建最最近的X天的报告。

json用于帮助解析json数据， api呼叫输出。

sys使用系统呼叫。

请求用于简化进行的Web要求API呼叫。

制成表使用自动地格式化表。

itemgetter用于作为iterator排序第2个表。

十进制用于舍入开销到两小数位位置。

```
if(len(sys.argv)==1):
        days = -1
elif(len(sys.argv)==2):
        try:
                days = int(sys.argv[1])
                if(days < 1):
                        raise ValueError('Less than 1')
                start=datetime.datetime.now()+datetime.timedelta(days*-1)
        except ValueError:
                print("Number of days must be an integer greater than 0")
                exit()
else:
        print("Enter number of days to report on, or leave blank to report all time")
        exit()
```

此部分用于解析几天编号line命令参数。

如果没有line命令参数(sys.argv ==1)，则报告将执行在所有时刻。

如果有一line命令参数检查它是否是大于或等于1的整数，如果在几天该编号报告，如果没有，请返回错误。

如果有超过一参数返回每错误。

```
departments = []
users = ['user1','user2','user3']
passwords = ['user1Key','user2Key','user3Key']
```

部门是将拿着最终输出的列表。

用户是将做API呼叫所有用户的列表，如果有多个转租人每个用户会是一个不同的转租人的admin。

密码是用户API密钥的列表，用户命令，并且密钥需要是相同的为了能将使用的正确密钥。

```
for j in xrange(0,len(users)):
        jobs = []
        r = requests.get('https://ccm2.cisco.com/v1/jobs', auth=(users[j], passwords[j]),
headers={'Accept': 'application/json'})
        data = r.json()
        for i in xrange(0,len(data["jobs"])):
                test = datetime.datetime.strptime((data["jobs"][i]["startTime"]), '%Y-%m-%d
%H:%M:%S.%f')
                if(days != -1):
                        if(start < test):
                                jobs.append([data["jobs"][i]["id"],'None',
data["jobs"][i]["cost"]["totalCost"],data["jobs"][i]["status"],data["jobs"][i]["displayName"],da
ta["jobs"][i]["startTime"]])
                else:
                        jobs.append([data["jobs"][i]["id"],'None',
data["jobs"][i]["cost"]["totalCost"],data["jobs"][i]["status"],data["jobs"][i]["displayName"],da
ta["jobs"][i]["startTime"]])
        for id in jobs:
                q = requests.get('https://ccm2.cisco.com/v1/jobs/'+id[0], auth=(users[j],
passwords[j]), headers={'Accept': 'application/json'})
                data2 = q.json()
                id[2]=round(id[2],2)
                for i in xrange(0,len(data2["metadatas"])):
                        if('BillingID' == data2["metadatas"][i]["name"]):
                                id[1]=data2["metadatas"][i]["value"]
                added=0
                for i in xrange(0,len(departments)):
                        if(departments[i][0]==id[1]):
                                departments[i][1]+= 1
                                departments[i][2]+=id[2]
                                added=1
                if(added==0):
                        departments.append([id[1],1,id[2]])
```

j在xrange(0,len(users)) ： 是为了环路能重复通过每用户定义在上一个代码大块，这是处理所有API呼叫的主环路。

工作是将使用保持工作的信息的一临时列表，当被校对到列表时。

r = requests.get .....是第一API呼叫，这一个列出所有工作，欲知更多信息看到ListJobs。

结果在**数据的**json格式然后存储。

i在xrange(0,len(data["jobs"])) ： 通过从上一个API呼叫返回的所有工作重复。

每工作的时期从json被拉并且转换对时间对象，然后与被输入的line命令参数比较发现是否在区域内。

如果它是，它是从被添附对工作列表的json的此信息： **id， totalCost，状态，名称，开始时间**。不是使用所有此信息，亦不是可以返回的此所有信息。 列表乔布斯显示可以相似地被添加返回的所有信息。

在您通过从该用户后返回的所有工作重复，您移动向**为在工作的id** ：哪些通过得到的所有工作重复，在您检查起始日期后。

q = requests.get (.....是第二API呼叫，这一个列出所有相关的信息对从第一API呼叫被采取的工作ID。 欲知更多信息请参阅GetJob详细信息。

json文件在**data2**然后存储**。**

开销，在**id[2]**存储被舍入到两小数位位置。

i在xrange(0,len(data2["metadatas"])) ：通过所有元数据重复关联与工作。

如果有呼叫**BillingID的**元数据那么在工作信息存储。

已添加用于的标志确定**BillingID**是否已经被添加了到**部门**列表。

i在xrange(0,len(departments)) ：通过被添加了的所有部门重复。

如果此工作是已经存在部门的一部分，则工作计数由一个重复，并且开销被添加到该部门的总成本。

否则，新的一行然后被添附对部门以工作计数1和总成本相等与开销此一工作。

```
departments = sorted(departments, key=itemgetter(1))
print(tabulate(departments,headers=['Department','Number of Jobs','Total Cost']))
```

部门=排序(部门， key=itemgetter(1))由乔布斯编号排序部门。

打印(请制成表(部门， headers= ['Department'，'Number of Jobs'， 'Total Cost']))打印表创建由制成表与三个报头。

相关信息

- CloudCenter API
- **技术支持和文档 - Cisco Systems**