API および大蛇が付いているカスタム レポートにメタデータを利用して下さい

目次

概要

この資料に大蛇スクリプト内の API カスタム レポートと共にメタデータを使用する方法を記述されています。

前提条件

要件

次の項目に関する知識が推奨されます。

- CloudCenter
- 大蛇

使用するコンポーネント

このドキュメントは、特定のソフトウェアやハードウェアのバージョンに限定されるものではありません。

本書の情報は、特定のラボ環境にあるデバイスに基づいて作成されたものです。 このドキュメントで使用するすべてのデバイスは、初期（デフォルト）設定の状態から起動しています。 稼働中のネットワークで作業を行う場合、コマンドの影響について十分に理解したうえで作業してください。

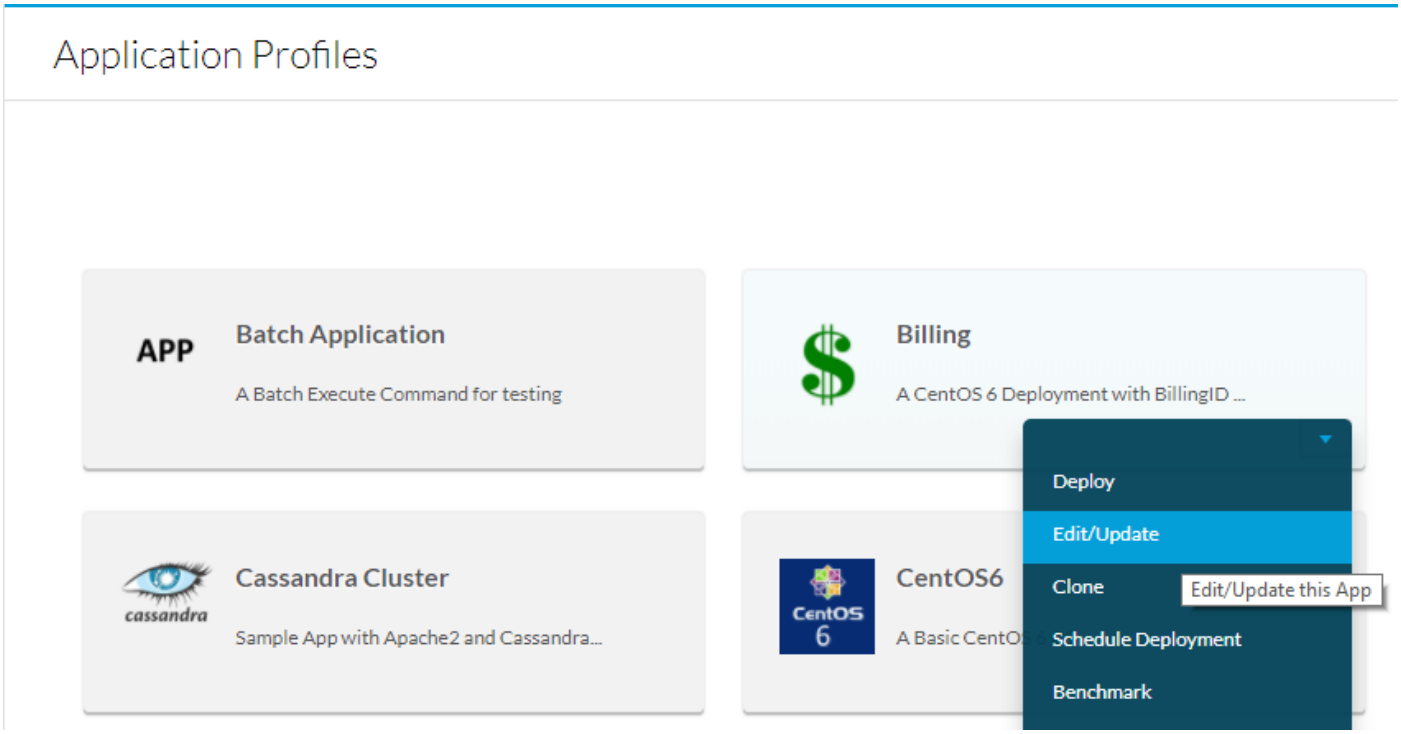
背景説明

CloudCenter はカスタム フィルターに基づいてレポートのための方法を可能にしないどんなに、レポート独自にを提供します。 メタデータと共にジョブに接続するデータベースからの情報を直接、つかむために API を使用するためにカスタム レポートを考慮に入れることができます。

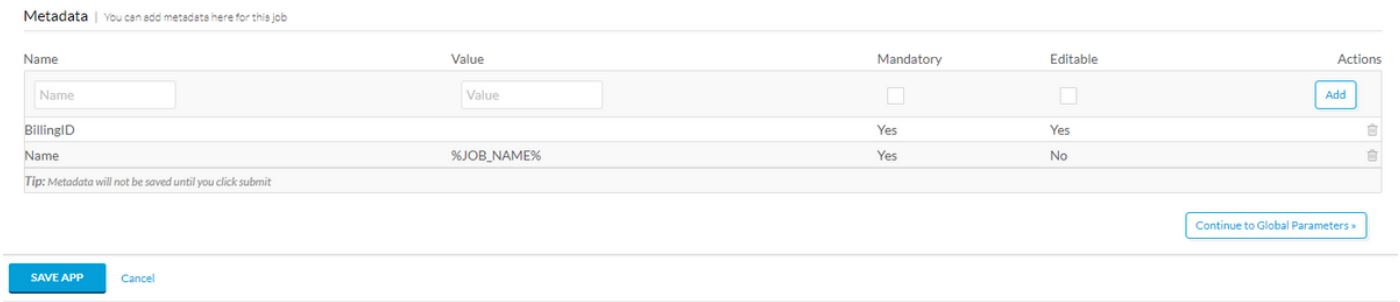
メタデータを設定して下さい

メタデータは修正されなければならないカスタム レポートの使用とトラッキングされる必要がある水平なアプリケーションごとの a でそう各アプリケーション追加する必要があります。

次にアプリケーションがおよび**/アップデート** イメージに示すように『Edit』 を選択 するために編集されることができるようにこれ、ナビゲートを**アプリケ- ションプロファイル**にするため、次にドロップダウンの選択するため。



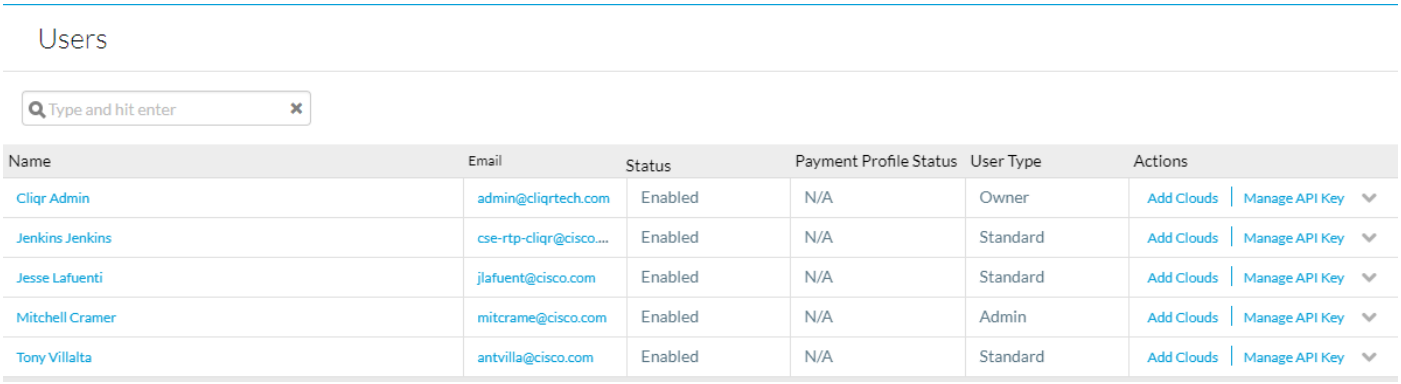
このメタデータが suer によって記入されるべきなら**基本情報**の下部のにスクロールし、メタデータ タグ付けします追加して下さい、たとえば **BillingID は**、必須および編集可能にします。 それがちょうどマクロである場合、デフォルト値を記入し、編集可能にしないで下さい。 メタデータに記入した後、イメージに示すようにそれから**保存アプリケーションを**『Add』 を選択 して下さい。



API キーを収集して下さい

API 呼び出しを処理するために、ユーザ名がおよび API キーは必要となります。 これらのキーはユーザと同レベルのアクセスを提供します、従ってすべてのユーザ配備がレポートに追加されるべきなら借用者 API キーの admin を得ることを推奨します。 複数の補助的な借用者が一緒に記録されるべきならルート借用者必要はすべての配置環境にアクセスします、またはすべての補助的な借用者 admin の API キーが必要となります。

API キー ナビゲートを **Admin > Users** に得るために **> API キーを管理し**、ユーザ名をコピーし、必要なユーザ向けにキー入力して下さい。



カスタム レポートを作成して下さい

レポートを作成する大蛇スクリプトを作成する前に、大蛇およびピップがそれでインストールされていたことを確認して下さい。 それからレポートを自動的にフォーマットすることを処理する実行**ピップ インストールは**ですライブラリ**表にしましたり**、表にします。

2 つのサンプル レポートは表でこのガイドに、第 1 単に収集しますすべての配備についての情報をそれから出力しますそれを接続されます。 第 2 は BillingID メタデータの使用でカスタム レポートを作成するのに同じ情報を使用します。 このスクリプトはガイドとして使用するために詳しく説明されます。

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

datetime が正確に日付を計算するのにこれされます最新 X 日のレポートを作成するために使用されています。

json データの解析を助けるのに **json が** API 呼び出しの出力使用されています。

sys はシステムコールのために使用されます。

要求が API 呼び出しのための Web 要求のすを簡素化するのに使用されています。

自動的に表をフォーマットするのに使用されています**表にして下さい**。

イテレータとして **itemgetter が**第 2 表をソートするのに使用されています。

小数点が 2 つの小数点以下にコストを四捨五入するのに使用されています。

```
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()
```

この部分が日数のコマンド・ライン パラメータを解析するのに使用されています。

コマンド・ライン パラメータ（sys.argv ==1）がなければ、レポートはすべての時間の間行われます。

1 に等しい、 またはそれ以上であるのは整数である場合 1 コマンド・ライン パラメータ チェックがあれば、それがその日数で報告される場合、ない、エラーを返す場合。

エラー複数のパラメータ戻りがあれば。

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

部門は最終的な出力を保持するリストです。

ユーザは複数の転借人がある場合、API 呼び出しを作るすべてのユーザのリストです各ユーザは別の転借人の admin です。

パスワードはユーザ API キーのリスト、ユーザの順序で、使用されるべき正しいキーのために同一である必要をキー入力します。

```
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 のための xrange(0,len(users)) : これは前のコード チャンクの各ユーザが定義するによって繰り返すべきループのためですすべての API 呼び出しを扱うメイン ループです。

ジョブはリストに照合される間、ジョブのための情報を保持するのに使用する一時リストです。

r = requests.get は.....最初の API コール**です**、この 1 つはすべてのジョブをリストします、詳細については ListJobs を見ます。

結果は**データ**の json 形式でそれから保存されます。

i のための xrange(0,len(data["jobs"])) : 前の API コールから戻ったすべてのジョブによって繰り返します。

各ジョブの時間は json から引っ張られ、datetime オブジェクトに変換されます、そして境界の内にあるかどうか見ることを入力されるコマンド・ライン パラメータと比較されます。

そうである場合、それはジョブ リストに追加 される json からのこの情報です: **ID、totalCost、ステータス、名前、開始時刻**。 この情報すべてが使用されません、返すことができるこのすべての情報はあります。 リスト ジョブは同じように追加することができる完全に返される情報を示します。

そのユーザから戻るすべてのジョブによって繰り返した後**ジョブの ID のために**に移動します: 繰り返すかどれが開始日をチェックした後奪取 されたすべてのジョブによって。

q = requests.get (.....第 2 API コールは最初の API コールから奪取 されたジョブ ID に、この 1 リストしますすべての関連情報を**あります**。 詳細については GetJob 詳細を参照して下さい。

json ファイルは **data2** でそれから保存されます。

コストは 2 つの小数点以下に、**id[2]** で保存される四捨五入されます。

i のための xrange(0,len(data2["metadatas"])) : ジョブと関連付けられるすべてのメタデータによって繰り返します。

BillingID と呼ばれるメタデータがあればジョブ情報で保存されます。

BillingID が既に**部門**リストに追加されていたかどうか確認するのに使用されるフラグが**追加されます**。

i のための xrange(0,len(departments)) : 追加されたすべての部門を通って繰り返します。

このジョブがあれば既に存在 して いる 部門ジョブ数は 1 つによって繰り返されます、およびコストの一部はその部門のためのトータルコストに追加されます。

そうでなかったら、それから 1 のジョブこの 1 つのジョブのコストと等しい数およびトータルコストで復帰改行文字は部門に追加 されます。

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

部門はジョブの数 = **(部門、key=itemgetter(1))** ソートします部門を**ソートしました。**

プリント (表にして下さい (ジョブ」、「トータルコスト」の部門、headers= [「部門」、「数])) 作成される表を表にします 3 つのヘッダと印刷します。

関連情報

- CloudCenter API
- **テクニカル サポートとドキュメント – Cisco Systems**