Register New IMEI Units On RMS 4.1 Using Pytohn Automatic Script Universal Small Cell (USC)

Contents

Introduction

This document describes the process of how to registered new IMEI in Remote Management Services (RMS) using an automatic script.

Prerequisites

Requirements

There are no specific requirements for this document.

Components Used

This document is not restricted to specific software and hardware versions.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Background Information

Before running this process please check with customer permission, user and pass to the Remote Management Services (RMS) elastic server (root user).

In the beginning of the procedure and at the end, Check the id pool usage report to see how many celling left.

duplicate cellid can be created on network if there is not enough cellid's on pool.

Configure

Run register procedure :

Step 1. Create csv file according to this format, as shown in the image:

	A	B	C	D	E
	EID	Select Area Manually	Area	RFProfile_New	Activated
	001B67-352639055652167	TRUE	Israel	13dBm_Multicell_Open	TRUE

Step 2 . FTP the csv file to RMS elastic 1 server with user/pass

Location library: **/intucell/scripts**

Step 3. Open putty/command line to elastic1.

Step 4. Run command : **cd /intucell/scripts**.

Step 5. Run command : **python sc_eid_registration_prod.py csvfile.csv**

python sc_eid_registration_prod.py

```
import re
import subprocess
import sys
import csv
import datetime
import time
import os

def run(file_name):
    #import pdb;pdb.set_trace()
    print "Starting.............................."
    eids = readFromCSV(file_name)
    csvFilename="sc_eid_registration.csv"
    f = open('regEid.txt','w')
    for row in eids:
        header='Content-Type: application/xml'
        myURL="http://192.168.166.129:8083/pmg"
        #print "EID=%s"%eid
        myXml="<?xml version='1.0' encoding='UTF-8'?><Register
xmlns='http://www.cisco.com/ca/sse/PMGMessages-v2_0_0'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xsi:schemaLocation='http://www.cisco.com/ca/sse/PMGMessages-v2_0_0 pmg-messages-
v2_0_0.xsd'><TxnID>Register-TxnID-
0</TxnID><EID>%s</EID><Activated>true</Activated><GroupMemberships><Group><Name>Israel</Name><Ty
```

```
pe>Area</Type></Group><Group><Name>%s</Name><Type>RFProfile</Type></Group></GroupMemberships></R
egister>"%(row['eid'],row['profile'])
        cmd='curl -X POST %s -vv -u "pmguser:pmguser" --digest -H "%s" -d
"%s"'%(myURL,header,myXml)
        cmd_ = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE)
        ok=False
        for line in cmd_.stdout:
            x = line.find("Success")
            if x!=-1:
                f.write("%s --- Register OK\n"%row['eid'])
                ok=True
                break
        if not ok:
            f.write("%s --- Register Fail\n"%row['eid'])
    f.close() # you


def readFromCSV(csvFilename):
    eid_list=[]
    with open(csvFilename, "U") as f:
        reader = csv.DictReader(f)
        for line in reader:
            try:
                eid=str(line["EID"]).strip()
                profile=str(line["RFProfile_New"]).strip()
                if len(eid)>0 and len(profile)>0:
                    eid_list.append({'eid':eid,'profile':profile})
            except Exception as e:
                print "readFromCSV Error: %s"%e
                f.close()
                sys.exit()
        f.close()
        return eid_list



if __name__ == '__main__':
   try:
        usage="usage: python sc_eid_registration_prod.py <file name>"
        l=len(sys.argv)
        if l==2:
            run(sys.argv[1])
        else:
            print usage
   except Exception as e:
        print "ERROR:%"%e
        print usage
   finally:
        sys.exit()
```

The report file is automatically created when a script is done.

Report example:

regEid.txt

```
001B67-352639055637721 --- Register OK
001B67-352639055637242 --- Register OK
001B67-352639055637218 --- Register OK
001B67-352639055637036 --- Register OK
001B67-352639055636947 --- Register OK
001B67-352639055636897 --- Register OK
001B67-352639055636830 --- Register OK
```

```
001B67-352639055636780 --- Register OK
001B67-352639055636764 --- Register OK
001B67-352639055636228 --- Register OK
001B67-352639055636137 --- Register OK
001B67-352639055635741 --- Register OK
001B67-352639055635295 --- Register OK
001B67-352639055635220 --- Register OK
001B67-352639055634959 --- Register OK
001B67-352639055633985 --- Register OK
001B67-352639055480304 --- Register OK
001B67-352639055480221 --- Register OK
001B67-352639055480130 --- Register OK
001B67-352639055480056 --- Register OK
001B67-352639055479785 --- Register OK
001B67-352639055479611 --- Register OK
001B67-352639055479546 --- Register OK
001B67-352639055479405 --- Register OK
001B67-352639055471162 --- Register OK
001B67-352639055470214 --- Register OK
001B67-352639055469539 --- Register OK
001B67-352639053871033 --- Register OK
001B67-352639053870704 --- Register OK
001B67-352639053863915 --- Register OK
001B67-352639053592746 --- Register OK
001B67-352639055781081 --- Register OK
001B67-352639055781073 --- Register OK
001B67-352639055781065 --- Register OK
001B67-352639055780877 --- Register OK
001B67-352639055780869 --- Register OK
001B67-352639055651912 --- Register OK
001B67-352639055651839 --- Register OK
001B67-352639055651789 --- Register OK
001B67-352639055651706 --- Register OK
001B67-352639055651672 --- Register OK
001B67-352639055651664 --- Register OK
001B67-352639055651656 --- Register OK
```

Verify

There is currently no verification procedure available for this configuration.

Troubleshoot

There is currently no specific troubleshooting information available for this configuration.