Tableau Group Subscription Sync

What is "Tableau Group Subscription Sync"?

- It is a Python script that synchronizes Tableau subscriptions with one or more user groups.
- If there are users added/removed from the user groups, the script will update the subscriptions accordingly.

Who should use this script?

- Tableau administrators who need group subscription capabilities.
- Tableau administrators who need to use a service account as subscription owner but the enterprise authentication restricts service account login.

Why is this script needed?

- Tableau (as of v2019.3) does not have the group subscription capability. Although when you first create subscriptions, you can specify user groups as subscribers; however, the user groups you entered are not linked to the subscriptions, only group members are linked to the subscriptions.
- This means if you need to maintain subscriptions for all members in the user groups, you must manually add/remove subscriptions when group membership changes.
- The script automates add/remove subscriber process. It can be scheduled to run at desire intervals.
- An added side benefit of using this script is subscription owner can be assigned to a service account. *Actually, this is a key benefit for us. Due to security restrictions, we cannot sign-in to Tableau using a service account (AD authentication restriction). The main benefit of assigning subscription owner to a service account instead of to a user is when the user leaves the company; there is no need to recreate the subscriptions.*

How does the script work?

The script utilizes Tableau Server Client (TSC) library, which was published by Tableau for the Python programming language.

Below is a list of steps the script performs:

1. Login to a Tableau environment
2. Retrieves subscribers base on the following information:
   - Subscription Schedule
   - Subscription Subject Line
   - Subscription Target (Workbook/View)
3. Retrieves group members for desired user groups
4. Compare subscribers vs. group members
5. Add missing subscribers
6. Remove subscribers who are no longer group members
7. Logout

Installation/Setup

1. Download Python from Anaconda website below and run the installation:
    - https://www.anaconda.com/distribution/
2. Install Tableau Server Client (TSC) library
    - pip install tableauserverclient
3. Download "Tableau Group Subscription Sync" script (TableauGroupSubSync.py) from https://github.com/BiToolkits/Tableau
4. For Windows, setup DOS environment variable if script is running from Command Line:

   Edit the Windows Properties Environment Path variable to add following paths.

   ```
   C:\ProgramData\Anaconda3;C:\ProgramData\Anaconda3\scripts;C:\ProgramData\Anaconda3\Library\bin
   ```

How to use the script

1. Enter required Tableau subscription information under the "Initialization" section of the code. There is no need to change anything else in the script
2. Save the script to a file, such as TableauGroupSubSync.py
3. Run the script

Assumptions and Limitations

- Make sure user subscription option is enabled in TSM; otherwise, error 400063: Bad Request - errors.bad_request.detail.generic_create_subscription.
- A subscriber must have email address for the script to be able to create subscription; otherwise, the script will error out. *Note there is a bug in TSC library where the Email attribute in UserItem class contains NULL for all AD users; thus, the script cannot set to exclude subscribers who don't have email address (such as service accounts).*
- The script checks the subscriber's Site Role info and only create subscriptions for subscribers who have assigned Tableau license.
- The script deletes subscriptions whose subscribers are Unlicensed. *Note this is to prevent a RuntimeException error to show up in Background Task Status.*
- Custom Views are currently not supported in REST API. There are two requests to support Custom Views in REST API. Please vote for them.
    - https://community.tableau.com/ideas/8729
    - https://community.tableau.com/ideas/9173
- Each project name must be unique within the site; otherwise, the script may not be able to pick out the correct workbook if the workbook names are not unique.

```
# ========================================================
# Program: Tableau Group Subscription Sync
# Purpose: To automate Tableau Dashboard Subscription by User Groups
# Author: Chung Lau (BiToolkits@gmail.com)
# Download: https://github.com/BiToolkits/Tableau
# Copyright (c) Pacific Life
# --------------------------------------------------------
#
# --------------------------------------------------------
# Date     Init Ver     Description
#--------- ---- ------  ----------------------------------
# 10/15/19  cl  1.0     Created
#
# ========================================================

#==========================
# Initialization (Be careful with case sensitivity unless stated otherwise)
#==========================
sScheduleName = "Daily Test Schedule"     # if *, get all subscriptions -- useful for
'RemoveUnlicensed' action
sSubjectLine = "My Subscription"
sProjectName = "Sales Project"
sWorkbookName = "Sales Workbook"
sTargetName = "Leaderboard"               # Target is what to subscribe, can be a view or
workbook.
sTargetType = 'View'                # 'View' or 'Workbook'
asUserGroups = [                        # comma (,) separate each group
name "Test User Group 1",
"Test User Group 2"
]
sServerName = 'https://myserver.com'
sLoginID = 'userid'
sLoginPassword = 'password'
sLoginSite = 'Production'


#==========================
# End Initialization
#==========================

import tableauserverclient as TSC
# Max items to fetch from Tableau is 1000
# If you have more than 1000 item, you need to use pagination
request_option  =  TSC.RequestOptions()
request_option.pagesize = 1000

# get object ID from a desire End Point
def msGetID(pvclsEndPoint, pvsName):
        for clsItem in TSC.Pager(pvclsEndPoint, request_option):
                if clsItem.name == pvsName:
                        return(clsItem.id)
        return("")

# get Workbook ID
def msGetWorkbookID(pvsName, pvsProjectID):
        for clsItem in TSC.Pager(clsServer.workbooks, request_option):
                if clsItem.name == pvsName and clsItem.project_id == pvsProjectID:
                        return(clsItem.id)
        return("")

# get View ID
def msGetViewID(pvsName, pvsWorkbookID):
        for clsItem in TSC.Pager(clsServer.views, request_option):
                if clsItem.name == pvsName and clsItem.workbook_id == pvsWorkbookID:
                        return(clsItem.id)
        return("")

# function to return the 1st element of the
# two elements passed as the paramater
def mvSortFirst(val):
        return val[0]
```

```
# get subscribers (user id, subscription id) from a Subscription
def masGetSubscribers(pvsScheduleID, pvclsTarget, pvsSubjectLine):
        asSubscribers = []
        for clsSubItem in TSC.Pager(clsServer.subscriptions, request_option):
                if clsSubItem.schedule_id == pvsScheduleID and clsSubItem.target.id ==
pvclsTarget.id and clsSubItem.target.type == pvclsTarget.type and clsSubItem.subject ==
pvsSubjectLine :
                        asSubscribers.append([clsSubItem.user_id, clsSubItem.id])
        return (asSubscribers)

# get user ids (excluding Unlicnesed users and users with no email) from a group
def masGetGroupMembers(pvsGroupName):
        matching_groups = list(TSC.Pager(clsServer.groups, request_option))
        asUserIDs = []
        for mygroup in matching_groups:
                if mygroup.name == pvsGroupName:
                        pagination_item = clsServer.groups.populate_users(mygroup)
                        for vUser in mygroup.users:
                                if "Unlicensed" not in vUser.site_role: # **Tableau API Bug: AD
user in Tableau has no email** and vUser.email is not None:
                                        asUserIDs.append(vUser.id)
                                        #print (vUser.name, vUser.email, vUser.site_role)
        return (asUserIDs)

# get all users in the system
def masGetAllUsers():
        #-- limited output asAllUsers, pagination_item = clsServer.users.get()
        asAllUsers = list(TSC.Pager(clsServer.users, request_option))
        return (asAllUsers)

# Target class is needed when query subscriptions
class clsSubscriptionTarget:
        def init (clsSubTarget, id, type):
                clsSubTarget.id = id
                clsSubTarget.type = type

def mPrintUserName(pvsText, pvasUserName):
        print (pvsText)
        for sUserName in pvasUserName:
                print ("    ", sUserName)
        return

#==========================
# MAIN CODE
#==========================

# create an auth object
clsTabAuth = TSC.TableauAuth(sLoginID, sLoginPassword, sLoginSite)

# create an instance for your server
clsServer = TSC.Server(sServerName)

# call the sign-in method with the auth object
clsServer.auth.sign_in(clsTabAuth)

with clsServer.auth.sign_in(clsTabAuth):
        sScheduleID = msGetID(clsServer.schedules, sScheduleName)
        sProjectID = msGetID(clsServer.projects, sProjectName)
        sWorkbookID = msGetWorkbookID(sWorkbookName, sProjectID)
        print ("Schedule Name, ID ==> ", sScheduleName, sScheduleID)
        print ("Subject Line ==> ", sSubjectLine)
        print ("Project ==> ", sProjectName, sProjectID)
        print ("Workbook ==> ", sWorkbookName, sWorkbookID)

        # Setup subscription Target
        if sTargetType == "View":
                sTargetID = msGetViewID(sTargetName, sWorkbookID)
        else:
                sTargetID = sWorkbookID
        clsTarget = clsSubscriptionTarget(sTargetID, sTargetType)
```

```
        print ("Target Name, ID, Type ==> ", sTargetName, sTargetID, sTargetType)

        # Get group members
        asUserIDList = []
        for sUserGroup in asUserGroups:
                asUserIDList = asUserIDList + masGetGroupMembers(sUserGroup)
                print ("Group Member Count ==> ", sUserGroup, len(asUserIDList))

        asUserIDList = list(dict.fromkeys(asUserIDList))    # De-dup
        print ("Unique Group Member Count ==> ", len(asUserIDList))

        # Get subscribers
        asSubscriberList = masGetSubscribers(sScheduleID, clsTarget, sSubjectLine) # returns
pairs of user ids and sub ids
        asSubUserIDs = []
        for asSubscriber in asSubscriberList:
                asSubUserIDs.append(asSubscriber[0])
        print ("Subscription Count ==> ", len(asSubscriberList))

        # Calculate Add/Remove users
        asSubUserIDsAdd = list(set(asUserIDList) - set(asSubUserIDs))
        asSubUserIDsRemove = list(set(asSubUserIDs) - set(asUserIDList))

        # Resolve Add user id to user login id for display purpose
        asAllUsers = masGetAllUsers()
        asSubUsersAdd = [asUser for asUser in asAllUsers if asUser.id in asSubUserIDsAdd]
        asSubUsersAdd.sort(key=lambda x: x.name)
        print ("Add Users ==> ", [asUser.name for asUser in asSubUsersAdd])

        # Resolve Remove user id to user login id for display purpose
        asSubsRemove = []
        for sUserID in asSubUserIDsRemove:
                asUserName = [asUser.name for asUser in asAllUsers if asUser.id == sUserID]
                asSubID = [asSub[1] for asSub in asSubscriberList if asSub[0] == sUserID]
                asSubsRemove.append([asUserName[0], asSubID[0]])
        asSubsRemove.sort(key=lambda x: x[0])
        print ("Remove Users ==> ", [asUser[0] for asUser in asSubsRemove])

        # Add subscriptions
        for asUser in asSubUsersAdd:
                print ("Add Subscription for User:", asUser.name)
                # create a new SubscriptionItem object.
                clsNewSub = TSC.SubscriptionItem(sSubjectLine, sScheduleID, asUser.id, clsTarget)
                # create the new subscription to the site
                clsNewSub = clsServer.subscriptions.create(clsNewSub)

        # Remove subscriptions (including Unlicensed)
        for asSub in asSubsRemove:
                print ("Remove Subscription:", asSub[0])
                clsNewSub = clsServer.subscriptions.delete(asSub[1])

clsServer.auth.sign_out

#============================
# END CODE
#============================
```