Reusing Teamwork Cloud session

When using cURL, user credentials (user name/password or a token) can be provided in a command-line argument. A new session is created during authentication. It is **very** important to note that a Teamwork Cloud session remains open after a REST API returns. This implies that a Teamwork Cloud session is not closed automatically. If the user's credentials are still in use (without sending a valid cookie), the license count will be used up, and new sessions cannot be created. The user needs to wait until the session is timed out. The default timeout is 15 minutes.

The correct way to use REST API is to log in only once, reuse the session for all subsequent calls, and log out once finished.

The following example is a Bash script used for creating a comma-separated list of email addresses for all users in Teamwork Cloud, to facilitate notification sending.

```
#!/bin/bash
# Script to export a unique list of email address for all users in Teamwork Cloud into a singe CSV list
(email_list.csv)
# Author:  Benjamin Krajmalnik
#
#
SERVER=127.0.0.1
RESTBASEURL=https://$SERVER:8111/osmc
USERURL=${RESTBASEURL}/admin/users
LOGINURL=${RESTBASEURL}/login
LOGOUTURL=${RESTBASEURL}/logout
username=Administrator
password=Administrator
rm -f email*.*
curl -k -s -c cookie.txt --insecure -u $username:$password $LOGINURL
curl -k -s -c cookie.txt -b cookie.txt --insecure -X GET "$USERURL" -H "accept: application/json" > users.json
sed 's/\"\,\"/\n/g' users.json > users.txt
sed -i 's/\[\"//g' users.txt
sed -i 's/\"\]//g' users.txt
while read username; do
               curl -k -s -c cookie.txt -b cookie.txt --insecure -X GET "$USERURL/$username" -H "accept:
application/json" >> email.txt
done < users.txt
curl -s -c cookie.txt -b cookie.txt $LOGOUTURL
grep -E -o "\b[a-zA-Z0-9.-]+@[a-zA-Z0-9.-]+\.[a-zA-Z0-9.-]+\b" email.txt > email.lst
sed -i 's/\n/;/g' email.lst
sort email.lst | uniq > email_unique.lst
tr '\n' ',' < email_unique.lst > email_list.csv
truncate -s-1 email_list.csv
```

The first execution of cURL logs in to the server. The server returns a session ID in a cookie. The cookie is saved into the **cookie.txt** file. The remaining cURL calls send the cookie back to the server to reuse the open session.

The example below shows how to use PowerShell to create internal users from a CSV file.

```
# ---------------------------------------------------------------
# Script to import users from CSV file into Teamwork Cloud
# Author:  Benjamin Krajmalnik
# Format: non-quoted CSV, no headers
#  userLogin,userPassword,fullUserName,departmentName,emailAddress
#
# Usage:
# PowerShell.exe -ExecutionPolicy Bypass -File ImportUsersFromCSV.ps1
# ---------------------------------------------------------------
Add-Type @"
  using System.Net;
  using System.Security.Cryptography.X509Certificates;
  public class TrustAllCertificatesPolicy : ICertificatePolicy
  {
     public bool CheckValidationResult
     (
       ServicePoint srvPoint, X509Certificate certificate
       ,
       WebRequest request, int certificateProblem
     )
     {
       return true;
     }
  }
"@
[System.Net.ServicePointManager]::CheckCertificateRevocationList = $false;
[System.Net.ServicePointManager]::ServerCertificateValidationCallback += { $true; };
[System.Net.ServicePointManager]::CertificatePolicy = New-Object TrustAllCertificatesPolicy
[System.Net.ServicePointManager]::SecurityProtocol = [System.Net.SecurityProtocolType]::Tls12
$Server = Read-Host -Prompt 'Input the Teamwork Cloud Server Name or IP address'
$Admin = Read-Host -Prompt 'Input the Teamwork Cloud Administrator User'
$Passwd = Read-Host -Prompt 'Input the Teamwork Cloud Server Administrator Password'
$Userfile = Read-Host -Prompt 'Input the full path name of the file containing the users'
$RESTBASEURL = "https://"+$Server+":8111/osmc/"
$ADDUSERURL = $RESTBASEURL + "admin/users"
$LOGINURL = $RESTBASEURL   + "login"
$LOGOUTURL = $RESTBASEURL + "logout"
$pair="$($Admin):$($Passwd)"
$bytes = [System.Text.Encoding]::ASCII.GetBytes($pair)
$base64 = [System.Convert]::ToBase64String($bytes)
$basicAuthValue = "Basic $base64"
$headers = @{ Authorization = $basicAuthValue }
$RestError = $null
Try {
$response = Invoke-WebRequest -Uri "$LOGINURL" -Method Post -Headers $headers -SessionVariable 'Session'
} Catch {
        $RestError = $_
}
foreach($line in [System.IO.File]::ReadLines($Userfile))
{
                  $login,$password,$username,$department,$email = $line.split(',').trim()

                  $JSON=@"
                  {
                                              "userName": "$login",
                                              "password": "$password",
                                              "otherAttributes": {
                                                            "mobile": "",
                                                            "name": "$username",
                                                            "department": "$department",
                                                            "email": "$email"
                                              },
                                              "enabled": true
                              }
"@
                              $response = Invoke-WebRequest  -Uri "$ADDUSERURL" -Method Post -Body $JSON -
ContentType "application/json"  -WebSession $Session
}

$response = Invoke-WebRequest  -Uri "$LOGOUTURL" –Method Get -WebSession $Session
```