Twitter Web Part Configuration Guide

Ocius Technology Group

http://ocius.technology

Author: Tim Wheeler

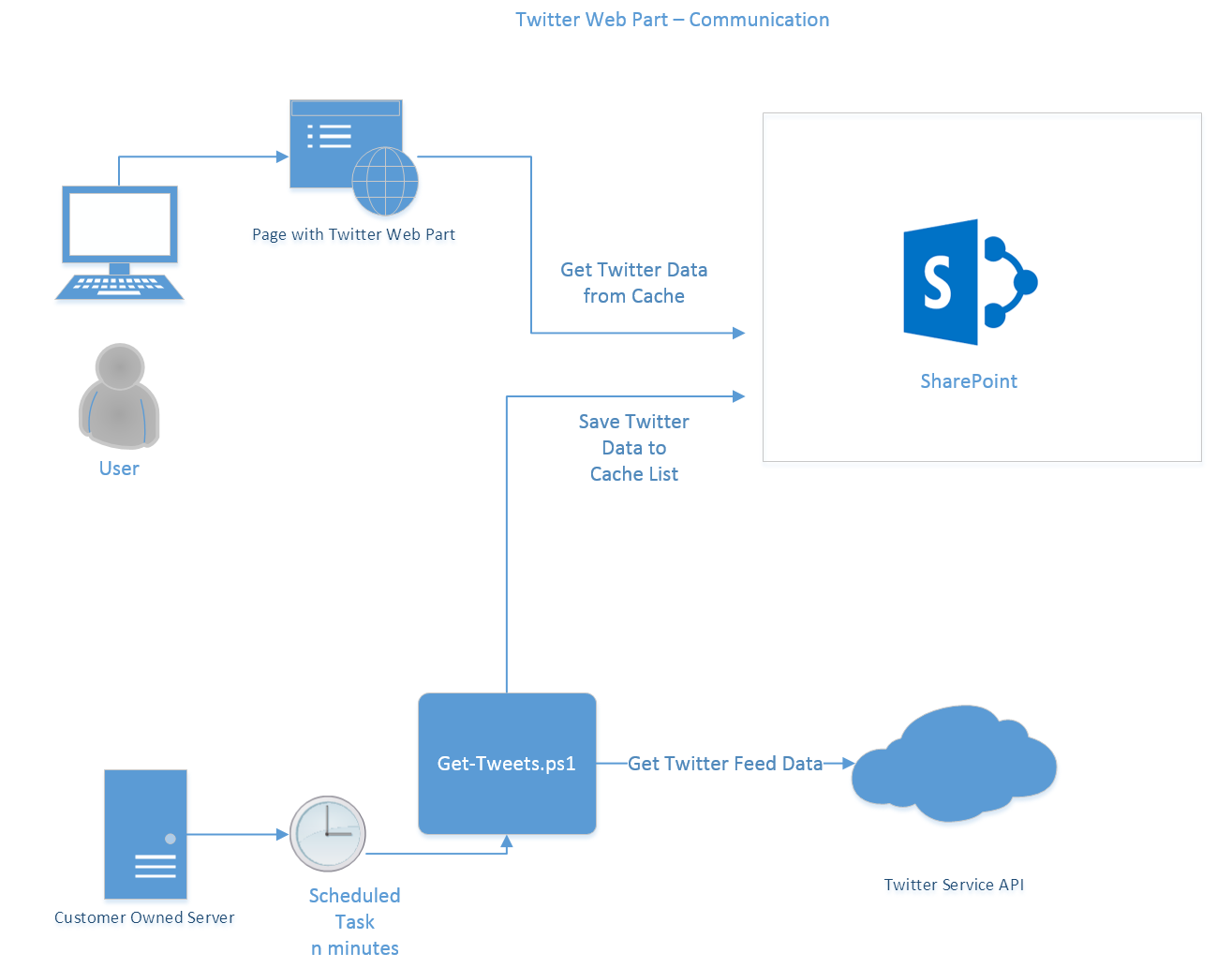
Blog: <http://blog.timwheeler.io>

Web: <http://ocius.technology>

# Summary

The Twitter Web Part displays a Twitter feed of a single account. It has configuration to support refresh and page size as well as the account name. This web part accesses content in a local site list called “Twitter Cache”. An external process is responsible for populating the list.

# Solution Overview



# Solution Requirements

This solution runs with:

* SharePoint 2013
* SharePoint 2016
* SharePoint Online

The Web Part uses:

* jQuery
* MomentJs
* Bootstrap Grid Styles

The Get-Tweets Server Solution uses:

* A scheduled task
* SharePoint CSOM
* PowerShell 3 or higher

# Communication Process

The Twitter Web part only interacts with SharePoint and uses the SharePoint Client Side Object Model. Data is retrieved from a list “Twitter Cache”. If the Refresh option is set in the Web Part configuration, then an automatic refresh from the Twitter Cache occurs on the specified interval.

# SharePoint Client Assemblies

The script references the SharePoint Client Side Object model assemblies, which are packaged in the original zip file along with the script. However, for security reasons if you prefer you can delete the included assemblies and download the originals’ from nuget.org. Be sure to place assemblies in the same location:

* <http://www.nuget.org/packages/Microsoft.SharePointOnline.CSOM/> (SP Online / SP2016)
* <http://www.nuget.org/packages/Microsoft.SharePoint2013.CSOM/> (SP2013)

# User Security Requirements

Users must have READ access to the Twitter Cache list. For security reasons only administrators and the Get-Tweets process account should have write access.

# Server Process – Get-Tweets

This process is implemented as a PowerShell script “Get-Tweets.ps1”. The script when executed with the correct parameters authenticates with Twitter then requests data from the Twitter API. The resulting response is JSON data, which is then saved to the SharePoint list “Twitter Cache” using the Client Side Object model.

The user does not interact with Twitter directly.

## Server Process – Setup Requirements

The Twitter API requires applications to be registered prior to accessing the API as the API does not allow anonymous access.

The process involves:

* [Sign up](https://twitter.com/signup) as a Twitter User
* [Register](https://apps.twitter.com/) as a Twitter App Developer
* [Register an App](https://apps.twitter.com/app/new)
* Configure App with Consumer Key and Consumer Secret (Keys and Access Tokens)
* Get the Bearer Token value by a one-off execution of the get-tweets.ps1 (See: Determining Configuration)
* Configure a scheduled task to repeatedly call get-tweets.ps1 (See: Creating a scheduled task)

Note: The access level for the app should be Read-only.

# Determining configuration

The get-tweets.ps1 script allows for 2 modes of execution:

* With a known Consumer Key and Secret, OR
* An encoded bearer/authentication token

When using the known Consumer Key and Secret, 2 calls are made. Firstly, to authenticate, then to request the twitter feed. However, if the bearer token method is used, only one call is ever made to Twitter. The ideal approach is to first run using the Consumer Key and Secret, copy the Bearer Token output, and then configure the scheduled task to use the Bearer Token only.

## Getting the bearer/authentication token

The following example uses the Consumer Key approach to authenticate and obtain a bearer token.

.\Get-Tweets.ps1 -siteUrl "https://mysharepointsite" -sponline:$true -username "myname@mysharepoint.onmicrosoft.com" -password "password" -screenName "TwitterScreenName" -consumerKey "consumer key" -consumerSecret "secret"

## Get-Tweets PowerShell Script Parameter Details

The following table indicates all the parameters and their meaning.

|  |  |
| --- | --- |
| Site URL | This is the URL to the root site collection in SharePoint |
| SPOnline | The script is compatible with SharePoint online. Set to $true for Online, $false of on premise. |
| Username | This is the user account that has write access to the SharePoint Twitter Cache list. (SharePoint access) |
| Password | Password associated with the Username. (SharePoint access). In interactive mode you can leave this blank and you will be prompted. However in a scheduled task this value must be provided. |
| ScreenName | This is the screen name for the Twitter account. |
| ConsumerKey | This is provided in the Twitter App Developer area. It is an identify for an App. The get-tweets.ps1 script, will be the App in this case. (Twitter access) |
| ConsumerSecret | The associated secret to the ConsumerKey. (Twitter access) |
| AuthToken | This is an encoded token that is provided once authenticated. This will allow the app to communicate directly with the feed and avoid future authentication. To obtain the AuthToken you should first use the ConsumerKey and ConsumerSecret. Note: You can specify AuthToken OR Consumer Key and Secret but not both. |

## Creating the Scheduled Task

The scheduled task should use the AuthToken method. To do this you specify the AuthToken parameter and omit the ConsumerKey and ConsumerSecret parameters.

For example:

.\Get-Tweets.ps1 -siteUrl "https://mysharepointsite" -sponline:$true -username "myname@mysharepoint.onmicrosoft.com" -password "password" -screenName "TwitterScreenName" –AuthToken "AAAAAAAAAABBAAAAAAAAAPO70gAAAAAAwLAJsaJlYZ2C9gepscI9H7lv91s%3DLdEyfgspIBCxfl8IiMTwM44psFF6QxiX2trvUzL3tRKASdlYTb"

This command accesses the feed and uses the AuthToken to implement the OAuth 2.0 authentication. The above token is fake and been added for demonstration purposes. You will need to execute the script with the Consumer Key and Secret to obtain your own bearer token.

Scheduled Task settings:

Run in PowerShell 3 or higher

Recommend that this script runs every minute between 7am and 12 pm. (Although you can change it to any value you like and make it run continually).

It does NOT need to run elevated

It does need internet access

It does need access to SharePoint

Note: The script currently does not support specifying a proxy and will use the user account settings it is running under to access the internet. (You may need to login with the service account and configure the system proxy on the server you intend to run the script on)