Automating Q!365 projects - A workflow example

July 30, 2021

Contents

This is an example showing how to automate folder creation via triggers on Account Object creation/update.

It is recommended to write a project readme documenting the values used to configure your Q!365 instance.

Prerequisites

1. Microsoft Azure
2. Microsoft 365
3. Salesforce
4. Salesforce CLI (sfdx)

Setting Up

Install Q!365 in your Salesforce Org and set it up according to the Q!365 Admin Documentation.

SFDX

Navigate to your workspace folder via CLI, and execute:

```
sfdx force:project:create -p pkg -n $project
```

where `$project` is the name of the directory for your project files.

Enter the directory:

```
cd $project
```

It is recommended to use git for version control. Initialize a git repository and connect it to github with:

```
git init
git remote add origin git@github.com:$organization/$project.git
```

Connect the sfdx Project with your Salesforce Org:

```
sfdx auth:web:login -a $project -r https://test.salesforce.com
sfdx force:config:set defaultusername=$project
```

From now on, opening the Salesforce Org via sfdx should work:

```
sfdx force:org:open
```

Optionally only generate the login link, for manual copy-pasting:

```
sfdx force:org:open -r
```

Automation Workflow

Microsoft Graph Explorer

for executing HTTP requests

> Make sure to Sign In to your Account, and set permission `Sites.ReadWrite.All` via `...` > `Select permissions`

Run query:

> GET https://graph.microsoft.com/v1.0/sites?search=

The Response preview-window should contain the Sharepoint Site Id in the form of:

```
company.sharepoint.com,some-alphanumerics-with-dashes,some-alphanumerics-with-dashes
```

Using the Sharepoint Site Id, acquire the Drive Id with:

> GET https://graph.microsoft.com/v1.0/sites/{site-id}/drives

Using the Drive Id, acquire the Accounts Id:

> GET https://graph.microsoft.com/v1.0/drives/{drive-id}/root/children

Salesforce

The Service Account can be configured under `Setup` > `Home` > `Named Credentials`

Navigate to `Setup` > `Home` > `Custom Settings` > `New` to create a new Custom Settings object, name it `Q365_Project_Settings`.

On this Custom Settings Object, create new Custom Fields of type Text with a length of 255 and the following Field Labels:

1. Template Drive
2. Template Accounts
3. Destination Root
4. Destination Drive
5. Destination Accounts

Populate the Field Labels via the `Manage` button on the Object with the appropriate Ids from GraphExplorer. Destination Root can be set to the string 'root' for simple use cases.

SFDX

See the respective files in this repository for code examples.

Download the Custom Object definition:

```
sfdx force:source:retrieve -m 'CustomObject: Q365_Project_Settings__c'
```

Create Apex classes for configuration and directory creation:

```
sfdx force:apex:class:create -d pkg/main/default/classes -n SharepointConfig
sfdx force:apex:class:create -d pkg/main/default/classes -n SharepointUtilities
sfdx force:apex:class:create -d pkg/main/default/classes -n SharepointFoldersCtrl
```

Create Apex Triggers for the Sharepoint Folders to be used as template structures:

```
sfdx force:apex:trigger:create -d pkg/main/default/triggers -n SharepointFoldersAccounts
```

Create a `$project/package.xml` file detailing your changes.

Deploy your changes to the org with:

```
sfdx force:source:deploy -x package.xml
```

If deployment fails, it does so detailing the errors responsible. Iterate fixing them until deployment succeeds.

If using git for version control, commit your changes.

Tips

List all SFDX commands:

```
sfdx commands
```

Show help for SFDX command COMMAND:

```
sfdx help COMMAND
```

```
sfdx auth:list
```

```
sfdx config:list
```

```
sfdx force:org:list
```