A2019 Salesforce Package

Readme

Version 1.0

10/26/2020

Table of Contents

1. Introduction 3

1.1 Overview 3
1.2 Use Cases 3

2. Requirements & Prerequisites 4

2.1 System Requirements 4
2.2 Prerequisites 4

3. Getting Started 8

3.1 Quick Start 8
3.1.1 Setup 8
3.1.2 Configuration 8

4. Support & FAQs 12

4.1 Support 12
4.2 FAQs 12

Appendix A: Record of Changes 13

Appendix B: References 14

1. Introduction

This document contains all essential information for the user to make full use of this A2019 Package. It includes a description of the functions and capabilities and step-by-step procedures for setup & configuration of the Package.

1.1 Overview

The Salesforce package includes 5 actions that allow RPA developers to interface with their Salesforce environment to read, write, update, and delete both standard and custom Salesforce objects.

1.2 Use cases

The Salesforce Package include 5 actions for interfacing with/managing your Salesforce Env.

- **Authenticate**: To authenticate against your salesforce env and generate an access token (used via sessions for all subsequent commands)
- **Delete Object**: Deleting Objects in Salesforce based on the Object ID
- **Insert Object**: Creating new Objects in Salesforce based on their type with support for a dictionary of key value pairs to set object attributes
- **Update Object**: Updating attributes on existing Salesforce Objects based on the Object ID with a dictionary of key value pairs containing the attribute name and target value.
- **Execute SOQL**: This allows for the retrieval of Salesforce Objects returned as a JSON.

Using these actions, users can take any number of actions on their Salesforce environment – from validating data in other systems matches the data that has been recorded in Salesforce to migrating data from 3rd party applications into Salesforce directly.

The goal of this package is for the available actions to be usable for any developer with limited to no understanding of the Salesforce API.

2. Requirements & Prerequisites

2.1 System Requirements

Enterprise A2019 (Cloud deployed) and Community Edition device requirements.
Review the machine hardware specifications, operating system versions, and browser types supported by Automation Anywhere Enterprise for creating and running bots and command packages as an Enterprise A2019 (Cloud deployed) or Community Edition user on your local machine.

Salesforce access with the ability to create (or have someone create for you) a Connected App. This connected app will generate the client ID and client secret necessary for authentication.

2.2 Prerequisites

Automation Anywhere Enterprise A2019

Windows Bot Runner

Salesforce Env with API v49+ (older versions may work, but have not been explicitly tested)

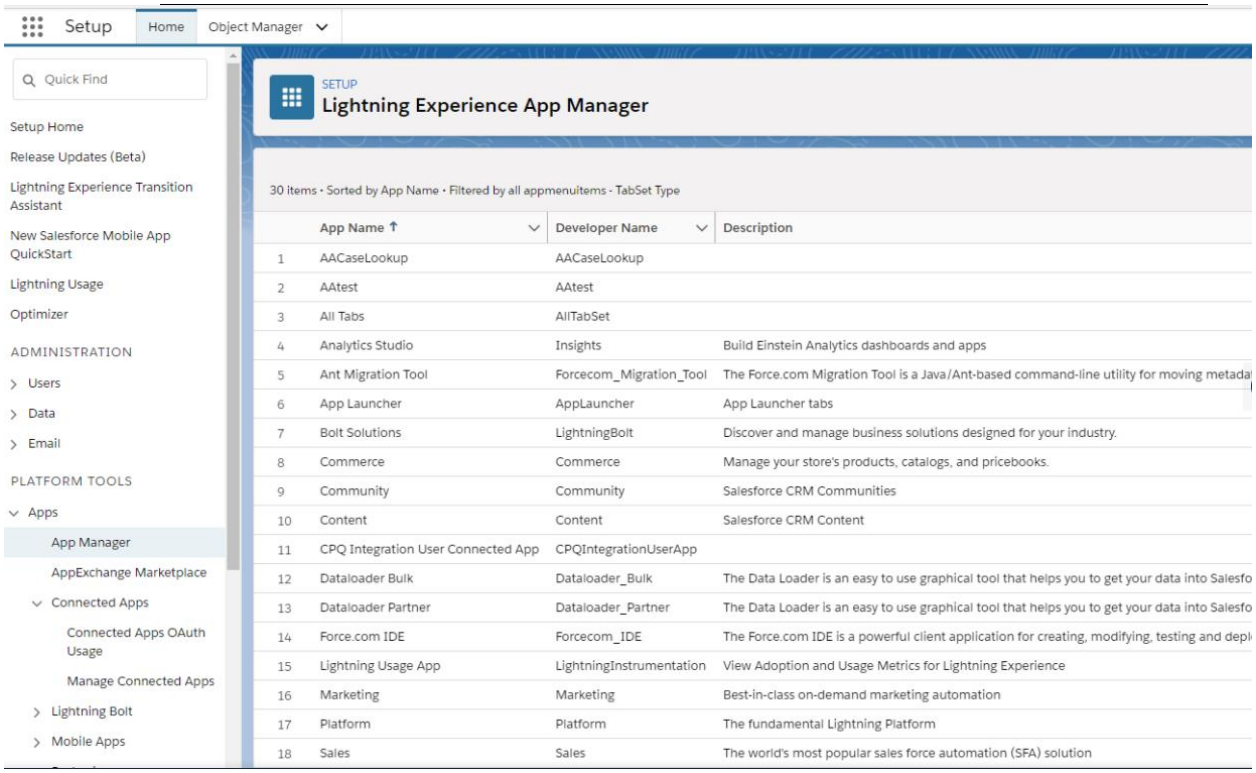
Salesforce Connected App (with client ID and client secret)

Note: This package can be used with the Salesforce dev account, so even in the absence of trying it out on your organization's Salesforce env, you can use it with a dev account as that's how it was originally developed/tested.

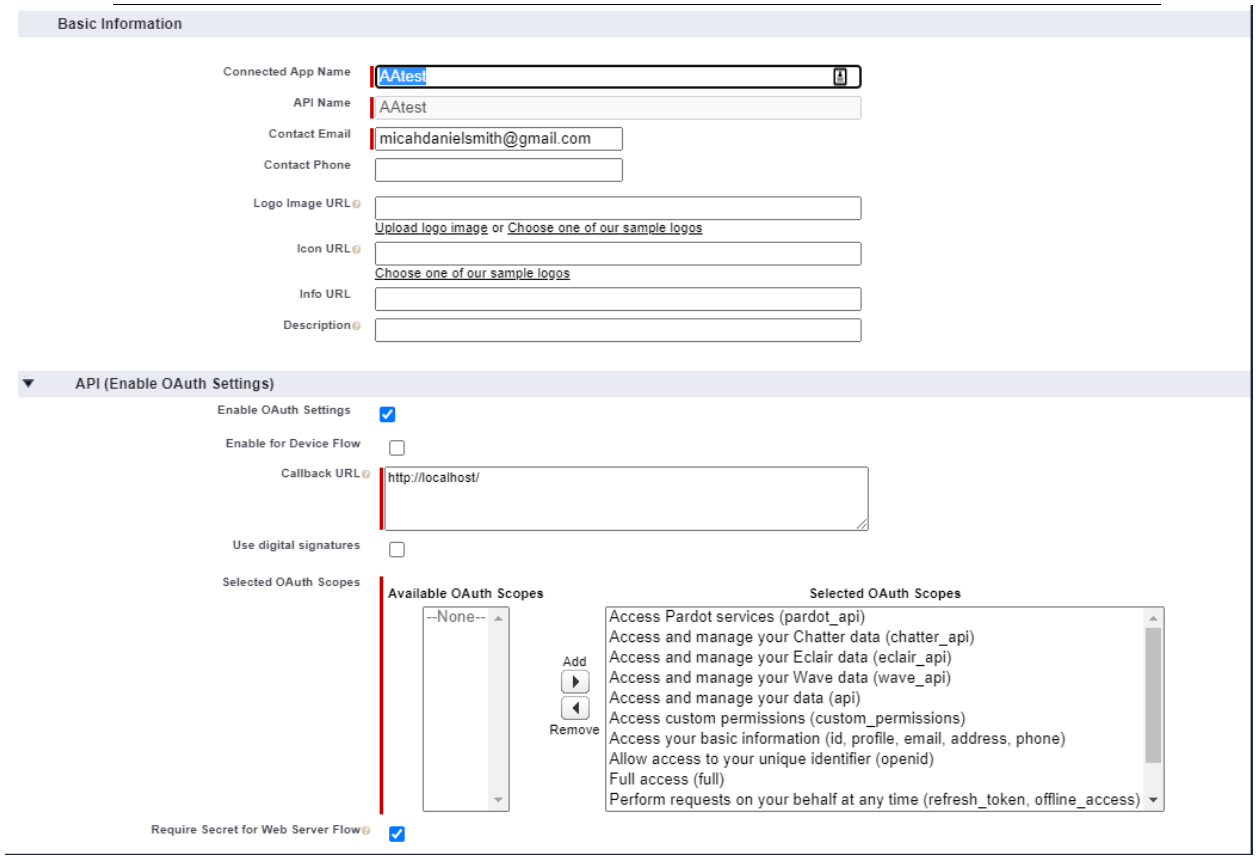
2.3 Creating a Connected App in Salesforce

Once you have a Salesforce env, you'll need to create a Connected App in Salesforce. The Connected App will allow you to generate a Consumer Key and Consumer Secret which are needed for your Salesforce Authentication. To create a Connected App:

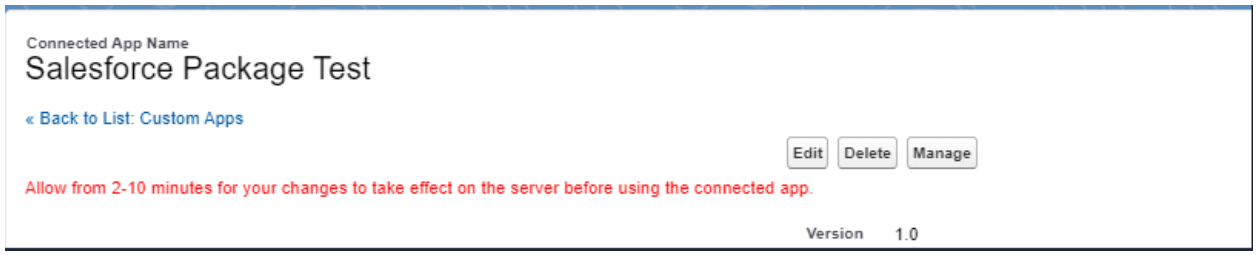
1. Login to your Salesforce developer environment.
2. Mine defaults me to the setup page, if yours does not, navigate to https://na174.lightning.force.com/lightning/setup (replacing na174.lightning.force with the default URL that Salesforce grants for your login)
3. On the left side navigation under Platform Tools, you should find Apps.
4. Navigate to Apps > App Manager to display all connected apps.



5. In the upper right hand corner, select the button that says “New Connected App”. You’ll need to fill out some basic information for your app…most of which you won’t have to reference in the package. Consult with your Salesforce Admin on the appropriate security settings, the remainder of this setup assumes setting up a connected app for a test env.
    a. Give an app Name and Contact details
    b. Click the checkbox for Enable OAuth Settings…this will be needed so our package can authenticate to Salesforce.
        i. For Callback URL just use **http://localost/** unless your organization uses a specific callback.
        ii. For selected OAuth Scopes, use All Available OAuth Scopes since this setup assumes establishing a connected app for test purposes only.
        iii. The box for **Require Secret for Web Server Flow** is checked by default - leave it checked.
        iv. Everything else can be left blank. Press **Save** to continue.



c. You'll get a message indicating it can take *x* amount of time for your changes to take effect on the server side. Press **Continue**
d. You'll be redirected to the Manage Connected Apps page with the specific details for your new connected app.
e. Copy down the **Consumer Key** and the **Consumer Secret**.
    i. Be sure when you copy them that you don't accidentally end up with an extra character. For some reason when copying the Consumer Secret I kept getting a carriage return at the end, so copy them into a notepad++ document or other text editor before pasting them into the Authenticate action to validate there were no additional characters.
f. Press the **Manage** button towards the top of the connected app.



g. By default, Salesforce Connected Apps use **Enforced IP restrictions**. Press the **Edit Policies** button.
h. In the **IP Relaxation** drop down, select **Relax IP restrictions** and press **Save** at the bottom of the screen.
    i. Note: This step is optional, but if your bot runners or other teammates using the same Connected App are in different geographic regions, they may not be able to authenticate via the Salesforce Package - so loosening these restrictions (especially for a test env) can help.

3. Getting Started

3.1 Quick Start

3.1.1 Setup

1. Install the package from Bot Store into your Control Room
2. Enable the package named **Salesforce** and set as default.
3. Navigate to **Salesforce – AutomationAnywhere** in the **Bot Store** folder to examine the installed bot.

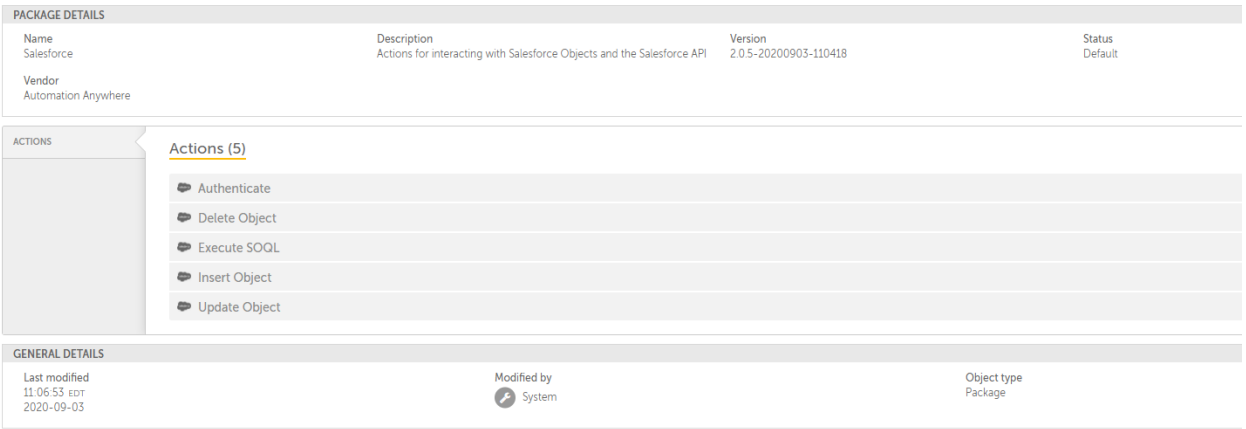


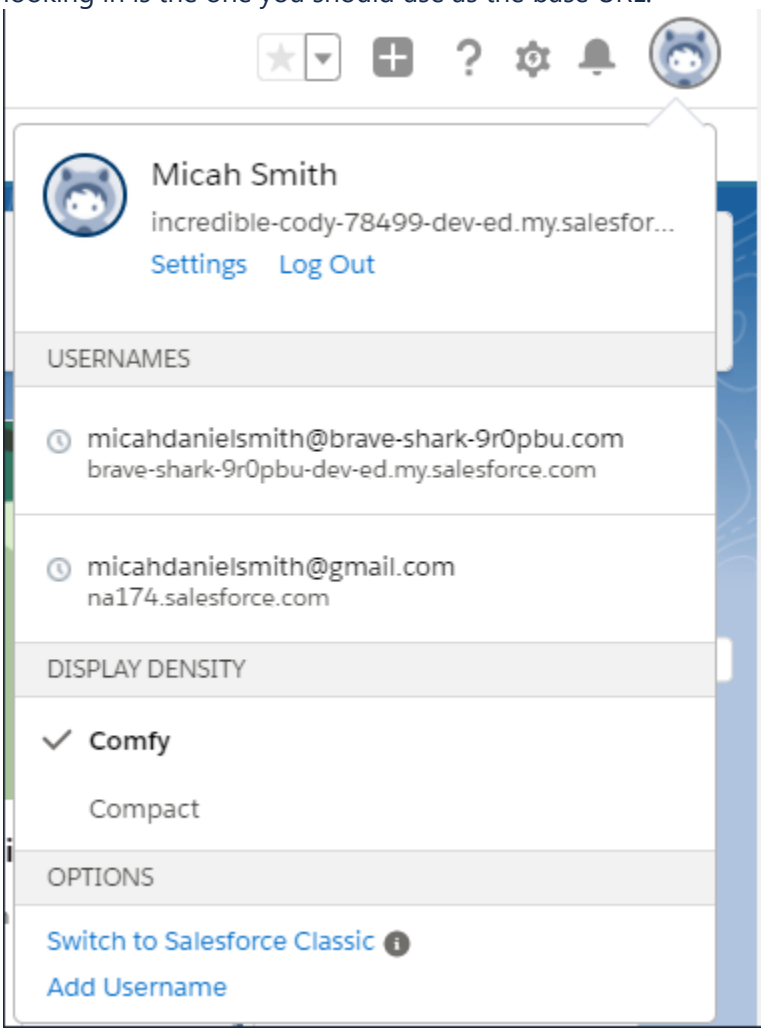
Figure 1: Note that the Version # and Last modified date may be different than what is seen in your environment.

3.1.2 Configuration and Use

There are 5 actions that make up this package:

1. **Authenticate**: This action is the first action which must be used when connecting to your Salesforce environment. This action requires the following values:
    a. **Base URL**: This is the base login URL for your SF env. For me (and the example shown, that means https://na174.salesforce.com - if you get an error, you may try using this, however it seems to be that whatever URL that shows up under your username after

looking in is the one you should use as the base URL.



b. **Client ID:** This is the Consumer Key we get from the Salesforce connected app. The Salesforce API uses the phrase "Client ID" so that language is carried over here to the field label.
c. **Client Secret**: This is the Consumer Secret we got from the Salesforce connected app. The Salesforce API uses the phrase "Client Secret" so that language is carried over here to the field label.
d. **Salesforce Username**: This is the username you used to login to Salesforce's UI
e. **Salesforce Password**: This is the password you used to login to Salesforce's UI
f. **Access Token Response**: You really don't need this value because its passed to the other actions via the session, but you should check to make sure the Access Token response doesn't include "Error retrieving access token…" as that's what I'm returning to the bot if the Auth fails.
    i. Note**:** The included sample bot displays the returns the result of the Authentication in a variable named sAuthToken and displays the result in a message box. This is purely for demonstration purposes. The token itself is

stored in the session and therefore would not need to be referenced via the variable.

2. **Insert Object**: The Insert Object action is used for creating new Objects (custom or standard) of an existing object type. The following fields are required for inserting an object.
    a. **Object Type:** The name of the object you intend to insert. Examples include Case, User, Opportunity, Location, or any custom objects that have been created.
    b. **Object Values:** This should be a dictionary (key-value pairs) of all the values required to create an object.
        i. In the sample bot included, the bot is creating a case by setting 4 total values.
        ii. **Note**: Certain objects (standard and custom) may have fields which are required for object creation. Check with your Salesforce env to identify the required data for each specific object type.
    c. **Insert Object Response**: On success, the ID of the newly created object is returned. Should an error occur, the error message would be returned on failure.
3. **Execute SOQL (Read Matching Objects):** SOQL is the Salesforce Object Query Language which allows users to query their Salesforce environment to return matching object details in the form of JSON response. This can be used for returning objects which may need to be validated against another system, or for getting Object iD's of matching objects which may need to be updated.
    a. **SOQL Query to Execute:** This is where you enter the SOQL query you wish for the bot to execute. If you aren't familiar with the Salesforce Query Language (admittedly I was not before building this package) – check out the Salesforce Developer Workbench (https://workbench.developerforce.com/query.php)
        i. The SOQL query builder in particular is helpful in building and testing SOQL queries – especially when custom Objects and Custom attributes have been added as this query builder shows the proper field names for each. Additionally, by running the queries in this tool first, you can see the expected result before trying to replicate the same from within your bot.
    b. **JSON Response**: The response from the Execute SOQL action is the raw JSON returned from Salesforce. This approach gives developers more control over how they wish to parse the response values and make use of the returned data. Developers are strongly encouraged to parse the response using the JSON Object Manager package available from Bot Store.
4. **Update Object**: The Update Object action allows bot builders to update object attributes after providing the object type itself, the object ID of the object to be

updated, and a dictionary (key value pairs) of the attributes to be updated with their target values. Note: Be aware of the object attributes which may be auto-generated vs those that can be updated via the Salesforce API.

a. **Object Type for Updating:** This is the object type. Same pattern that was used in the insert object action applies.

b. **ObjectID of the Object to Update:** To update an object, the object's ID is required. Likely this value would be extracted from the JSON response of a SOQL Query action, though it is possible that the object's id came from some other source.

c. **Fields to Update (dictionary):** Dictionary (key value pairs) of the object attribute fields and their target values.

d. **Result of Object Update:** Assuming a successful update, this result will return as "Object updated successfully" – otherwise the resulting error message will be displayed.

5. **Delete Object** The delete object action enables developers to delete Salesforce objects based on the object type and object ID.

   a. **Object Type for Deletion:** The object type is required for object deletion. This matches the same object types that were used in the insert and update actions.

   b. **Object ID for Object:** Like the update action, the delete action requires that developers provide the specific object ID for deletion. This will typically come as a result of an execute SOQL action response.

   c. **Delete Object Response:** Upon successful delete, this string value will return with the object ID of the successfully deleted object as well as a "… was successfully deleted" message. If the delete was not successful, the error message returned from the Salesforce API would be returned.

4. Support & FAQs

4.1 Support

Free bots are not officially supported through Automation Anywhere. You can get access to Community Support through the following channels:

- You can get access to Community Support, connecting with other Automation Anywhere customers and developers on APeople – the Bot Building Forum, the Bot Store Support Forum, or the Developers Everywhere Group.
- Automation Anywhere also provides a Product Documentation portal which can be accessed for more information about our products and guidance on Enterprise A2019.

4.2 FAQs

For questions relating to Enterprise A2019: See the Enterprise A2019 FAQs.

Appendix A: Record of Changes

No.	Version Number	Date of Change	Author	Notes
1.0	*1.0*	*10/28/2020*	*Micah Smith*	*Initial Salesforce Package Release*

Appendix B: References

No.	Topic	Reference Link
1	Overview of Enterprise A2019	Click here
2	Guidance: Building basic A2019 bots	Click here
3	Guidance: Building A2019 action packages	Click here
4	APeople Community Forum	Click here
5	Automation Anywhere University	Click here