Account Engagement API Quick Start

Getting started can be daunting, especially if you're not an experienced developer. This tutorial walks you through a few real-life tasks so you can start automating and extending Account Engagement.

In this guide:

We implement authentication using an OAuth2.0 flow.

Make some example calls and complete a basic task.

Before you begin, make sure you have:

A Salesforce admin to create a connected app and gather information.

A Salesforce user with SSO enabled for Account Engagement.

A configured Account Engagement Business Unit.

Selected which OAuth flow works best for your case. This tutorial uses the Web Server OAuth flow.

This tutorial requires background knowledge of connected apps and authentication. If you're unfamiliar with these topics, we **strongly** recommend that you check out these resources first:

Salesforce Help: Connected App and OAuth Terminology

Trailhead module: Connected App Basics

Trailhead module: Build a Connected App for API Integration

After you familiarize yourself with the basics, come back to this tutorial.

Warning

The calls you make during this tutorial have the potential to impact both your Salesforce and Account Engagement settings and data. If you're just testing the API, make sure that you're using a sandbox account. If you're using this tutorial to set up your production account, use caution.

To authenticate to use Account Engagement APIs, you must use an SSO-enabled Salesforce user who has Account Engagement access, and an OAuth flow with a connected app. Implementing authentication can be tricky, so we walk through an example here.

The first step in authentication is creating a connected app. A connected app is a framework that enables an external application to integrate with Salesforce using APIs and standard protocols like OAuth. Connected apps use these protocols to authenticate, authorize, and provide single sign-on (SSO) for external apps. To put it simply, a connected app represents an integration.

To set up a connected app, have a Salesforce admin follow these steps:

From Salesforce Setup, in the Quick Find box, enter **App Manager**, and then select **App Manager**.

Click **New Connected App**.

Name the connected app, and enter contact details for the app owner.

In the API section, select the **Enable OAuth Flows**.

Enter a Callback URL. The callback URL is used to redirect users after authentication in browser-based flows. In this example, we use `https://my.example.com/myapp`. If you're using a browser-based flow (such as Web Server Flow or User Agent Flow), the URL must match the URL you pass as a callback to OAuth endpoints. If you aren't using a browser-based flow, the URL entered isn't used.

Under **Selected OAuth Scopes**, add **Access Account Engagement services**, which give the app access to Account Engagement. For more complex scenarios, such as using refresh tokens, select other scopes as well.

Save the connected app.

This just scratches the surface of connected apps. If you'd like to learn more, like how to restrict access to certain users, check out these resources:

Connected App Salesforce Help

Connected App Trailhead

Now that you have a connected app, let's gather some important information you need handy to finish setting up authentication. The primary details are:

Connected App Consumer Key: a unique identifier for your connected app.

Connected App Consumer Secret: a password for the connected app.

Business Unit IDs: Because a Salesforce org can have multiple business units, the ID routes the API request to the correct business unit. You need the ID even if you have only one business unit.

Salesforce User: A Salesforce user with SSO enabled for Account Engagement.

Since all this information is sensitive, carefully consider how to securely share these details with other team members.
From Salesforce Setup, in the Quick Find box, enter **App Manager**, and then select **App Manager**.
Go to your connected app and select **View**.
Copy your consumer key.
Reveal and copy your consumer secret.
From Marketing Setup, in the Quick Find box, enter **Pardot**, and then select **Pardot Account Setup**.
Copy the business unit ID for the Account Engagement instance you want to use.
For the integration user, we recommend that you create a unique user for each specific app integration. The user must be SSO-enabled and have access to Account Engagement.
Salesforce has several OAuth flows to meet your unique security and integration needs. In this tutorial, we use the Web Server OAuth flow. This flow is ideal for when you don't want to store the end user's credentials in your system. To learn more about your options and how to choose the best OAuth flow for your integration, check out Salesforce OAuth Help Documentation.
Have your integration direct the user to Salesforce's OAuth authorization endpoint:
`https://login.salesforce.com/services/oauth2/authorize?response_type=code&client_id=CLIENT_ID&redirect_uri=https://my.example.com/myapp`
Replace `CLIENT_ID` with your connected app consumer key.
Replace `https://my.example.com/myapp` with your redirect URI.
If the user doesn't have an active session, they're prompted to log into Salesforce.
The user grants the app permission.
After the user has logged in and allowed the app, Salesforce redirects you back to the redirect_uri passed in to the authorize endpoint:`https://my.example.com/myapp?code=<CODE>`
Your server-side code exchanges this code for an access token by making a POST request to the Salesforce OAuth token endpoint:
After Salesforce validates the connected app credentials and authorization code, the endpoint responds with an access token:
The access token can be used to make calls to the Account Engagement API. The code was exchanged for the access token on the server side and not from the user's browser. Because of how the code was exchanged, there's no opportunity for malicious JavaScript to steal the access token.
After you've implemented authorization, you can start working with the Account Engagement API.
The Account Engagement API lets your application access current data within Account Engagement. Through the API, you can perform several common operations on Account Engagement objects including the following:
`create`: Creates an instance of the object with the specified parameters.
`read`: Retrieves information about the specified object.
`query`: Retrieves objects that match specified criteria.
`update`: Updates elements of an existing object.
`upsert`: Updates elements of an existing object if it exists. If the object doesn't exist, one is created using the supplied parameters.
Developers must authenticate using a Salesforce OAuth endpoint or the Account Engagement API login endpoint before issuing Account Engagement API requests. Refer to the Authentication section for details about this procedure.
Keep in mind a few considerations when you perform requests. For update requests, only the fields specified in the request are updated. All others are left unchanged. If a required field is cleared during an update, the request is declined.
All requests to the API must:
Use either HTTP GET or POST. Version 5 also supports PUT and DELETE requests.
Pass access token in an HTTP Authorization header.
Pass Account Engagement Business Unit ID in an HTTP Pardot-Business-Unit-Id header.
Use the correct URL for your Account Engagement environment.
In this example, we want to get a list of all custom fields in a business unit.
Before you begin, make sure you've authenticated and have a valid access token. For simplicity in this example, these calls are written in HTTP.
Let's break down the call and identify its parts, starting with the uniform resource identifier (URI). `GET 'https://<ENVIRONMENT_URL>/api/<OBJECT>/version/<API_VERSION>/do/<OPERATION>?format=<FORMAT>`
Environment URL: In the example, we use pi.pardot.com, but you use the URL that matches your environment type. Demos, developer orgs, and sandbox environments are hosted on the domain pi.demo.pardot.com. Training and production environments

are hosted on the domain pi.pardot.com.

Object: The object we're requesting data for. In our example, we're querying the CustomField object.

API version: The version of the API you're using. Here, we use v4.

Operation: The operation you're performing. Here, we use `query`.

Format: The output format, either XML or JSON. Here, we use `json`.

Now that we've explained the URI, let's examine the header lines in the call.

`'Authorization: Bearer <ACCESS TOKEN>'` lets Account Engagement know that you've authenticated and have permission to access data.

`'Pardot-Business-Unit-Id: <BUSINESS UNIT ID>'` lets Account Engagement know which business unit you want data from.

`Host`: The URL for your Account Engagement environment.

The call returns the business unit's custom fields and their metadata.

Learn more about each object's fields in the Object Field Reference documentation.

Warning: If you're following along with your Account Engagement instance, this next example creates a list in your account. Either use a sandbox account, or tailor the example to fit your needs.

In this example, we create a list for a spring nurturing campaign.

Before you begin, make sure you've authenticated and have a valid access token.

We went over the URI and the authorization and business unit headers in the GET example. Let's cover the other components:

`Content-Type: application/x-www-form-urlencoded`: Lets the server know what kind of data the request includes.

`name=Spring Leads Nurture`: Sets the list's internal name to "Spring Leads Nurture".

`title=Spring Leads`: Titles the list "Spring Leads".

`description=A list to nurture spring leads`: Adds a text description to the list.

When we send this request to the server, Account Engagement responds by creating a list with the criteria that we specified. The response looks like this:

Let's take the example a step further and add a prospect to our new list. Remember that completing this task changes data in Account Engagement, so proceed with caution.

Before you begin, make sure you've authorized and have a valid access token. For simplicity in this example, we make these calls in our command-line interface using cURL.

First, copy the list ID for the list you want to add a prospect to. Then, find the IDs for the prospect you want to add to the list. You can use `query` on the Prospect object to get ID.

List ID: 1000

Prospect ID: XXXXX

Notice that the URI includes the prospect's ID: XXXXX. When we send this request, Account Engagement adds that prospect to the list with the ID 1000.

That's a wrap! Now that you've made some basic calls, we hope you feel confident about using the Account Engagement API to extend and automate your business!

Connected Apps

Authorize Apps with OAuth

Connected App and OAuth Terminology

Find a Trailblazer Community Group near you and talk with other Account Engagement developers about how they use APIs.