DEVELOPER DOCUMENTATION

Authenticating with the Consensus API using OAuth 2.0

A complete guide to building applications that connect to the Consensus Platform API and MCP Server using the OAuth 2.0 Authorization Code flow with PKCE.

Audience: Developers & Partners **Applies to:** Consensus Platform API V2

Overview

The Consensus Platform API uses **OAuth 2.0 Authorization Code flow with PKCE** for application authentication. The interactive API docs let you generate a short-lived bearer token to test endpoints in the browser, but any application that needs to access the API on behalf of a user must implement OAuth.

This guide walks you through:

- Creating OAuth client credentials in the Consensus web app
- Implementing the full Authorization Code + PKCE flow
- Refreshing tokens so your app can run continuously
- Revoking and introspecting tokens

> **Base URLs.** All examples in this guide use the production base URL https://app.goconsensus.com. The OAuth endpoints live under /api/auth/v1.0/oauth2.

What you'll need

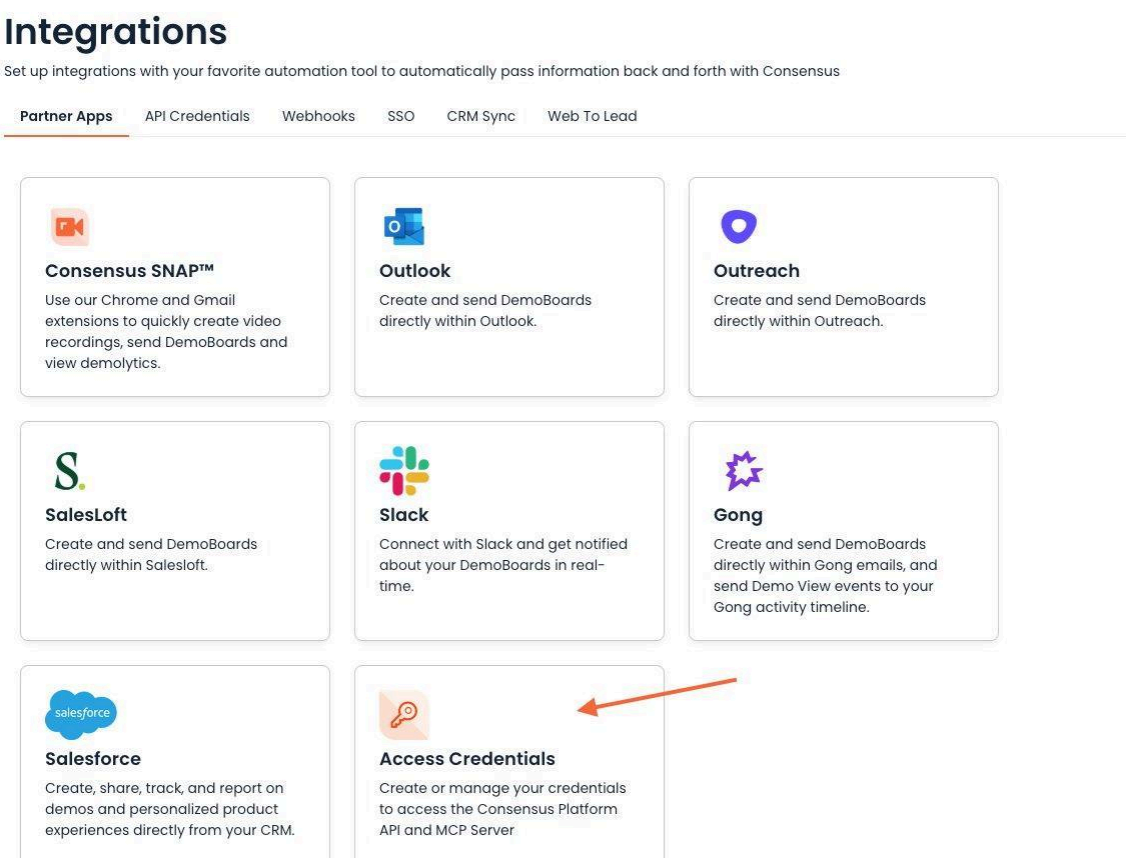
- A Consensus account with access to the **Integrations** settings
- A registered **Callback URL** for your application
- Your application's `client_id` and `client_secret` (generated in Step 1)

Step 1: Create access credentials

Before you can authenticate, you need to register your application and obtain a `client_id` and `client_secret` from the Consensus web app.

1.1 Open the Access Credentials app

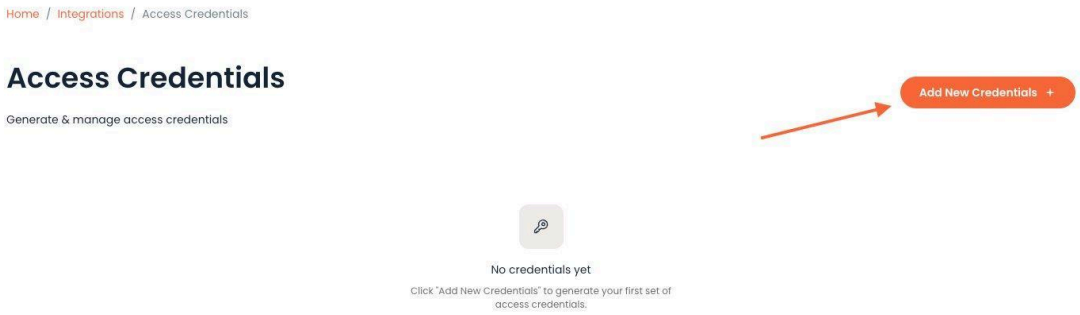
From the Consensus web app, navigate to **Integrations** and select the **Access Credentials** tile.



Integrations → Access Credentials

1.2 Add a new credential set

Click **Add New Credentials** in the top-right corner of the Access Credentials page.

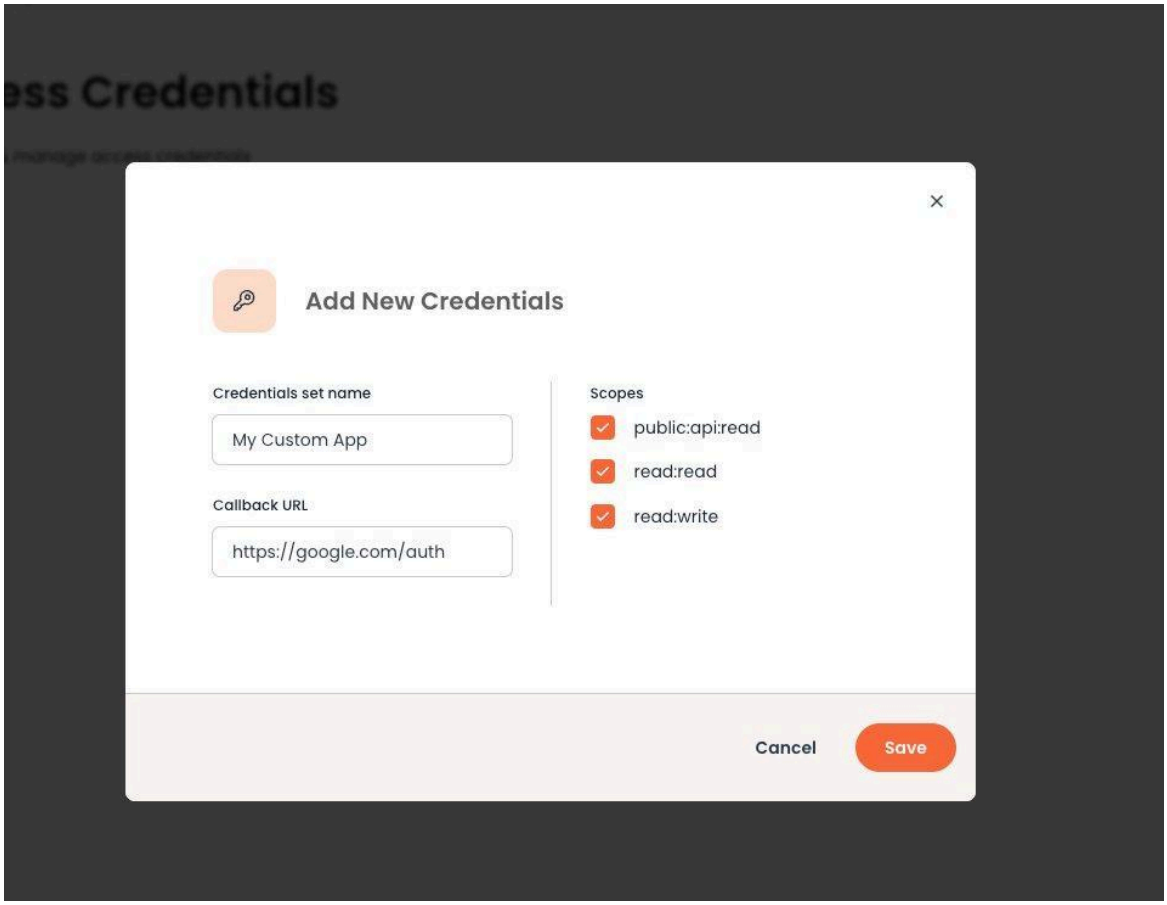


Click Add New Credentials to register a new OAuth client

1.3 Configure your application

Fill in the form with your application details:

- **Credentials set name** — A label for this credential set (e.g., the name of your app or integration).
- **Callback URL** — The exact URL Consensus will redirect to after the user authorizes your app. This must match the `redirect_uri` sent in your authorization request.
- **Scopes** — Select the scopes your application needs. Available scopes:
  - `public:api:read` — Read access to public API endpoints
  - `read:read` — Read scope for resources
  - `read:write` — Write scope for resources



Configure name, callback URL, and scopes, then click Save

1.4 Copy your client ID and client secret

Once saved, your new credentials will appear in the list. Click the eye icon next to **Client secret** to reveal it.

Your new credential set with client_id, client_secret, callback URL, and scopes

Store your client secret securely. Treat it like a password. Never commit it to source control or expose it in client-side code. If your secret is ever leaked, revoke the credential set and create a new one.

Step 2: Authorize the user

The OAuth flow begins by sending the user to the Consensus authorization endpoint. Consensus uses **PKCE (Proof Key for Code Exchange)**, so before redirecting, generate a `code_verifier` and its corresponding `code_challenge`.

2.1 Generate PKCE values

The `code_verifier` is a cryptographically random string. The `code_challenge` is the SHA-256 hash of the verifier, base64-url encoded.

```
// Node.js example
const crypto = require('crypto');

function base64URLEncode(buffer) {
  return buffer.toString('base64')
    .replace(/\+/g, '-')
    .replace(/\//g, '_')
    .replace(/=+$/, '');
}

const codeVerifier = base64URLEncode(crypto.randomBytes(32));
const codeChallenge = base64URLEncode(
  crypto.createHash('sha256').update(codeVerifier).digest()
);

// Store codeVerifier in your session - you'll need it in Step 3
// Send codeChallenge in the authorization URL
```

2.2 Redirect the user to the authorize endpoint

`GET` `https://app.goconsensus.com/api/auth/v1.0/oauth2/authorize`

Construct the authorization URL with the following query parameters:

Parameter	Required	Description
`response_type`	Yes	Must be `code`
`client_id`	Yes	Your client ID from Step 1

Parameter	Required	Description
`redirect_uri`	Yes	Must match the callback URL registered for your credentials
`scope`	Yes	Space-separated list of scopes (e.g., `public:api:read read:read`)
`state`	Yes	An opaque random string to prevent CSRF. Verify it on callback.
`code_challenge`	Yes	The PKCE challenge generated in 2.1
`code_challenge_method`	Yes	Must be `S256`

Example

```
https://app.goconsensus.com/api/auth/v1.0/oauth2/authorize
  ?response_type=code
  &client_id=52d8ce6c9e97a74ac...
  &redirect_uri=https%3A%2F%2Fyourapp.com%2Fcallback
  &scope=public%3Aapi%3Aread%20read%3Aread
  &state=xyz123
  &code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
  &code_challenge_method=S256
```

2.3 Handle the callback

After the user grants consent, Consensus redirects to your `redirect_uri` with two query parameters:

```
https://yourapp.com/callback?code=AUTH_CODE_HERE&state=xyz123
```

> **Validate the state value** matches the one you sent in Step 2.2. If it doesn't, abort the flow — this protects against CSRF attacks.

Authorization codes are single-use and short-lived. Exchange them for tokens immediately.

Step 3: Exchange the code for tokens

`POST` `https://app.goconsensus.com/api/auth/v1.0/oauth2/token`

Send a `POST` request with `Content-Type: application/x-www-form-urlencoded` and the following body parameters:

Parameter	Description
`grant_type`	Must be `authorization_code`
`code`	The authorization code from Step 2.3
`redirect_uri`	Must match the `redirect_uri` from Step 2.2
`client_id`	Your client ID
`client_secret`	Your client secret
`code_verifier`	The PKCE verifier you generated in Step 2.1

Example request

```
curl -X POST https://app.goconsensus.com/api/auth/v1.0/oauth2/token \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "grant_type=authorization_code" \
  -d "code=AUTH_CODE_HERE" \
  -d "redirect_uri=https://yourapp.com/callback" \
  -d "client_id=YOUR_CLIENT_ID" \
  -d "client_secret=YOUR_CLIENT_SECRET" \
  -d "code_verifier=YOUR_CODE_VERIFIER"
```

Example response

```
{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...",
  "refresh_token": "def50200a8b2c...",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

The access token is a JWT. Store both tokens securely on your server — never in the browser or in client-side code.

Step 4: Call the Consensus API

Include the access token as a Bearer token in the `Authorization` header for every API request.

```
curl https://app.goconsensus.com/api/v2/<endpoint> \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN"
```

Refer to the Consensus API V2 documentation for the full list of available endpoints.

Step 5: Refresh access tokens

Access tokens expire after one hour (`expires_in: 3600`). To keep your application running continuously without prompting the user to re-authorize, use the refresh token to request a new access token.

`POST` `https://app.goconsensus.com/api/auth/v1.0/oauth2/token`

Parameter	Description
`grant_type`	Must be `refresh_token`
`refresh_token`	The refresh token from Step 3
`client_id`	Your client ID
`client_secret`	Your client secret

Example request

```
curl -X POST https://app.goconsensus.com/api/auth/v1.0/oauth2/token \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "grant_type=refresh_token" \
  -d "refresh_token=YOUR_REFRESH_TOKEN" \
  -d "client_id=YOUR_CLIENT_ID" \
  -d "client_secret=YOUR_CLIENT_SECRET"
```

Refresh token rotation. Each refresh request may return a new refresh token along with the new access token. Always replace your stored refresh token with the latest value returned.

Recommended pattern

Refresh proactively before tokens expire. A common approach is to refresh when the token is within 5 minutes of expiry, or to handle 401 responses by refreshing and retrying once.

Revoking tokens

Use the revoke endpoint when a user disconnects your app or you need to invalidate a token.

`POST` `https://app.goconsensus.com/api/auth/v1.0/oauth2/revoke`

```
curl -X POST https://app.goconsensus.com/api/auth/v1.0/oauth2/revoke \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "token=YOUR_REFRESH_TOKEN" \
  -d "token_type_hint=refresh_token"
```

Set `token_type_hint` to `access_token` or `refresh_token` depending on which token you're revoking. Revoking a refresh token also invalidates any access tokens issued from it.

Inspecting a token

Use the introspection endpoint to check whether a token is still active and view its metadata.

`POST` `https://app.goconsensus.com/api/auth/v1.0/oauth2/introspect`

```
curl -X POST https://app.goconsensus.com/api/auth/v1.0/oauth2/introspect \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "token=YOUR_ACCESS_TOKEN" \
  -d "token_type_hint=access_token"
```

Getting user info

The `userinfo` endpoint returns information about the user associated with the current access token.

`GET` `https://app.goconsensus.com/api/auth/v1.0/oauth2/userinfo`

```
curl https://app.goconsensus.com/api/auth/v1.0/oauth2/userinfo \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN"
```

Endpoint reference

Method	Endpoint	Purpose
GET	`/api/auth/v1.0/oauth2/authorize`	Start the authorization flow
POST	`/api/auth/v1.0/oauth2/token`	Exchange auth code for tokens, or refresh tokens

Method	Endpoint	Purpose
POST	`/api/auth/v1.0/oauth2/revoke`	Revoke an access or refresh token
POST	`/api/auth/v1.0/oauth2/introspect`	Check the status of a token
GET	`/api/auth/v1.0/oauth2/userinfo`	Get info about the authenticated user

Need help?

If you run into issues implementing OAuth or have questions about the Consensus API, contact your Consensus partner manager or reach out through the standard support channels.