perplexity

how can I Authentication in baseca,p via rest API

To authenticate with Basecamp via its REST API, you need to use **OAuth 2.0**, as it is the standard authentication protocol for Basecamp 3 and 4 APIs[1] [2] [3]. Here's a step-by-step guide to performing authentication:

1. Register Your Application

- Go to the Basecamp developer portal at https://launchpad.37signals.com/integrations.
- Register a new app to receive your **Client ID** and **Client Secret**.
- Specify your **Redirect URI** (the URL Basecamp will redirect the user to after authorization).

2. Redirect the User for Authorization

- Construct an authorization URL:
  ```
  https://launchpad.37signals.com/authorization/new?type=web_server&client_id=YOUR_CLIE
  ```
- Redirect the user to this URL.
- The user logs in and grants your app permission.

3. Handle the Authorization Callback

- After the user authorizes, Basecamp redirects to your `redirect_uri` with a code like:
  ```
  https://your-app.com/callback?code=AUTH_CODE
  ```
- Extract the `code` parameter.

4. Exchange Code for Access Token

- Make a POST request to:
  ```
  https://launchpad.37signals.com/authorization/token
  ```
- With the parameters:
  - `type=web_server`
  - `client_id=YOUR_CLIENT_ID`
  - `client_secret=YOUR_CLIENT_SECRET`
  - `redirect_uri=YOUR_REDIRECT_URI`
  - `code=AUTH_CODE`

- The response contains your **access token** and **refresh token**.

5. Make Authenticated API Requests

- Use the access token in the HTTP `Authorization` header:

```
Authorization: Bearer ACCESS_TOKEN
```

6. Refresh Your Token

- When your access token expires, exchange your refresh token for a new access token:

```
POST to https://launchpad.37signals.com/authorization/token with:
  type=refresh
  client_id=YOUR_CLIENT_ID
  client_secret=YOUR_CLIENT_SECRET
  refresh_token=YOUR_REFRESH_TOKEN
```

- This will return a new access token.

Example Authorization Request (JavaScript/Node with Axios)

```
// Exchange code for access token
const response = await axios.post('https://launchpad.37signals.com/authorization/token',
  type: 'web_server',
  client_id: process.env.CLIENT_ID,
  client_secret: process.env.CLIENT_SECRET,
  code: authCode,
  redirect_uri: process.env.REDIRECT_URI
});
const accessToken = response.data.access_token;
```

Example Authenticated Request

```
const userData = await axios.get('https://3.basecampapi.com/ACCOUNT_ID/my/profile.json',
  headers: {
    'Authorization': `Bearer ${accessToken}`
  }
});
```

Note: For Basecamp Classic (not 3 or 4), you can use API tokens in requests, but modern Basecamp APIs require OAuth 2.0[4] [5] [6].

This flow can be implemented in any language or framework. Most Basecamp API libraries simply automate these steps[7].

Summary: Register your app, handle the OAuth 2.0 authorization flow, obtain an access token, and use it in your API requests[1] [2] [3].

⁂

please can how I do this:

To complete the OAuth 2.0 setup for Basecamp API authentication, you need to provide a **Redirect URI**—this is the URL Basecamp will send users back to after they authorize or deny access to your application. Here's how you do it:

1. **Choose Your Redirect URI**
    - This should be an endpoint in your application that will handle the OAuth callback (e.g., `https://yourapp.com/oauth/callback`).
2. **Enter Redirect URI in Basecamp Developer Settings**
    - When registering your app at the Basecamp developer portal, you'll see a field like in the image you provided labeled "Redirect URI (required)."
    - Enter your chosen Redirect URI (e.g., `https://yourapp.com/oauth/callback`) in that box.
3. **Implement the Redirect URI Handler in Your Application**
    - Your application needs to have a route at that exact URL which processes the query parameters sent by Basecamp (such as `code`).
4. **Use the Same Redirect URI in OAuth Requests**
    - When sending the user to the Basecamp authorization URL and when exchanging the `code` for a token, ensure you use the **exact** Redirect URI you specified here.

Example:

- Enter in the form: `https://yourapp.com/oauth/callback`

What happens next?

- Basecamp will redirect users to your URI with `?code=...` after they approve your app.
- Your app captures that code and exchanges it for an access token as described in the previous instructions.

Tip: The Redirect URI must match exactly—including protocol (`http` or `https`) and any trailing slashes—otherwise, the OAuth exchange will fail[8].

If you haven't implemented this yet, just enter the URI you expect to use; you can update it later in the Basecamp developer settings.

Let me know if you'd like step-by-step sample code for handling the callback in a specific programming language!

[8]

⁂

pleae: Let me know if you’d like step-by-step sample code for handling the callback in a specific programming language!

Here’s a step-by-step example of how to handle the OAuth callback in **Node.js/Express** after Basecamp redirects the user back to your redirect URI with a `code`. This example assumes your redirect URI is `https://yourapp.com/oauth/callback`.

Step 1: Set Up the Express Route

```
const express = require('express');
const axios = require('axios');
const app = express();
require('dotenv').config(); // To easily manage your secrets

// This is your registered redirect URI:
app.get('/oauth/callback', async (req, res) => {
  const authCode = req.query.code; // Get the code from the query string

  if (!authCode) {
    return res.status(400).send('No code found in the callback request.');
  }

  try {
    // Step 2: Exchange code for access token
    const tokenResponse = await axios.post('https://launchpad.37signals.com/authorizatio
      type: 'web_server',
      client_id: process.env.CLIENT_ID,
      client_secret: process.env.CLIENT_SECRET,
      redirect_uri: process.env.REDIRECT_URI,
      code: authCode
    });

    // Get the tokens from the response
    const { access_token, refresh_token, expires_at } = tokenResponse.data;

    // Step 3: Use or store the tokens as needed
    // You should securely save the tokens—this is just an example
    res.json({ access_token, refresh_token, expires_at });

    // Step 4: Redirect or continue flow as needed
    // For example, res.redirect('/dashboard');
  } catch (error) {
    res.status(500).send('Error exchanging code for token: ' + error.message);
  }
});

app.listen(3000, () => {
  console.log('OAuth callback server listening on port 3000');
});
```

Environment Variables (`.env`)

```
CLIENT_ID=your_basecamp_client_id
CLIENT_SECRET=your_basecamp_client_secret
REDIRECT_URI=https://yourapp.com/oauth/callback
```

Key Points:

- The `/oauth/callback` route receives the `code` from Basecamp.
- You POST the code and app credentials back to Basecamp to get the access token and refresh token.
- With the access token, you can now make authenticated API requests to Basecamp on behalf of the user.
- Always securely store the received tokens.

Let me know if you want similar code for a different language or need help with another part of the flow!

⁂

1. https://github.com/basecamp/bc3-api
2. https://rollout.com/integration-guides/basecamp-3/how-to-build-a-public-basecamp-3-integration-building-the-auth-flow
3. https://www.devzery.com/post/basecamp-api-your-guide-to-custom-project-management-integrations
4. https://github.com/basecamp/basecamp-classic-api
5. https://classic.basecamp-help.com/article/587-api-integrations
6. https://www.fancyhands.com/help/basecamp-setup
7. https://pypi.org/project/basecampapi/
8. image.jpg