Integrating Asana with Teamchat

Step-1: CREATING A USER ACCOUNT

Register for a GoToMeeting account at https://app.asana.com/. This will be your account for handling with projects.

Step-2: CREATING A NEW APP

REGISTERING AN APPLICATION

Developers seeking to implement an Asana Connect application must first register their app to receive a Client ID and Client Secret. Fortunately, this process is fast and easy: visit your Account Settings dialog and click the Apps tab. There will be a link to “Register New Application”. As with all OAuth apps, you must supply an Application URL as well as a Redirect URL that successful (or failed) authentications will redirect to. Additionally, you can set a name and an icon to enhance the recognizability of the application. When users are prompted to authorize your app, they will be more likely to click “Allow” if it is clear who they are passing their authorization on to.

Once you have created an app, the details view will include a Client ID, needed to uniquely identify your app to the Asana API, as well as a Client Secret. Applications implementing the Authorization Code Grant flow use the secret to securely identify themselves when retrieving the user’s bearer token.

Step-4: FETCHING ACCESS TOKEN

So how does OAuth with Asana work exactly? What is it these examples do, and how precisely would you implement it yourself?

Obviously, a working knowledge of the OAuth 2.0 spec (we're using Draft 31) is useful. If you're familiar with the spec — or intend to use an out-of-the-box OAuth 2.0 library rather than hand-coding it — all you need to know is this:

Quick Reference

The endpoint for user authorization is https://app.asana.com/-/oauth_authorize

The endpoint for token exchange is https://app.asana.com/-/oauth_token

Applications can be created from the "Apps" tab of your account settings, where you will find your Client ID and Client Secret. (Quick Link)

We support both the Authorization Code Grant flow, and the Implicit Grant flow.

Calls to the API made with the header Authorization: Bearer $TOKEN will automatically be authorized to act on behalf of the user.

Note: The currect OAuth implementation does not support scopes or other flows.

Authorization Code Grant

To actually implement the Authorization Code Grant flow (the most typical flow for most applications), there are basically three steps:

Redirect a user to the authorization endpoint so that they can approve access of your app to their Asana account

Receive a redirect back from the authorization endpoint with a code embedded in the parameters

Exchange the code for a token via the token exchange endpoint

The token that you have at the end can be used to make calls to the Asana API on the authorizing user's behalf.

Implicit Grant

The implement the Implicit Grant flow, which is suitable for in-browser web apps in JavaScript or other applications that might have difficulty making arbitrary HTTP POST requests to the token exchange endpoint, there are only two steps:

Redirect a user to the authorization endpoint so that they can approve access of your app to their Asana account

Receive a redirect back from the authorization endpoint with a token embedded in the fragment portion (the bit following the #) of the URL.

This token can then be used to access the API, in this case typically using JSONP.

User Authorization Endpoint

Request

Your app redirects the user to https://app.asana.com/-/oauth_authorize, passing parameters along as a standard query string:

client_id - required The Client ID uniquely identifies the application making the request.

redirect_uri - required The URI to redirect to on success or error. This must match the Redirect URL specified in the application settings.

response_type - required Must be one of either code (if using the Authorization Code Grant flow) or token (if using the Implicit Grant flow). Other flows are currently not supported.

state - optional Encodes state of the app, which will be returned verbatim in the response and can be used to match the response up to a given request.

Response

If either the client_id or redirect_uri do not match, the user will simply see a plain-text error. Otherwise, all errors will be sent back to the redirect_uri specified.

The user then sees a screen giving them the opportunity to accept or reject the request for authorization. In either case, the user will be redirected back to the redirect_uri.

If using the response_type=code, your app will receive the following parameters in

the query string on successful authorization:

code - This is the code your app can exchange for a token

state - The state parameter that was sent with the authorizing request

If using the response_type=token, your app will receive the following parameters in the URL fragment (the bit following the #):

token - This is the token your app can use to make requests of the API

state - The state parameter that was sent with the authorizing request

Token Exchange Endpoint

Request

If your app received a code from the authorization endpoint, it can now be exchanged for a proper token, optionally including a refresh_token, which can be used to request new tokens when the current one expires without needing to redirect or reauthorize the user.

Your app makes a POST request to https://app.asana.com/-/oauth_token, passing the parameters as part of a standard form-encoded post body.

grant_type - required Must be authorization_code

client_id - required The Client ID uniquely identifies the application making the request.

client_secret - required The Client Secret belonging to the app, found in the details pane of the developer view

redirect_uri - required Must match the redirect_uri specified in the original request

code - required The code you are exchanging for an auth token

Alternatively, if you are exchanging a refresh_token, the grant_type should be refresh_token and instead of sending code=... you would send refresh_token=....

Response

In the response, you will receive a JSON payload with the following parameters:

access_token - The token to use in future requests against the API

expires_in - The number of seconds the token is valid, typically 3600 (one hour)

token_type - The type of token, in our case, bearer

refresh_token - If exchanging a code, a long-lived token that can be used to get new access tokens when old ones expire.

data - A JSON object encoding a few key fields about the logged-in user, currently id, name, and email.tps://app.asana.com/api/1.0/users/me

Step-5: USING ACCESS TOKEN TO CALL FUNCTIONS

Get available workspaces

```
GET    /workspaces
GET    /workspaces/workspace-id
This method returns the workspace records, described above.

Get all available workspaces

# Request
curl -u <api_key>: https://app.asana.com/api/1.0/workspaces

# Response
{
```

```
 "data": [
 {
  "id": 1337,
  "name": "Space Cats"
 },
 ...
 ]
}
```

Create Project

```
POST    /projects
POST    /teams/team-id/projects
POST    /workspaces/workspace-id/projects
```

This method creates a new project and returns its full record.

Every project is required to be created in a specific workspace or organization, and this workspace cannot be changed once set. Note that you can use the workspace parameter regardless of whether or not it is an organization.

If the workspace for your project is an organization, you must also supply a team to share the project with.

Create a project

```
# Request
curl -u <api_key>: https://app.asana.com/api/1.0/projects \
-d "name=Things to Buy" \
-d "notes=These are things we want to purchase." \
```

```
-d "workspace=14916"


# Response
HTTP/1.1 201
{
  "data": {
    "id": 1331,
    "name": "Things to Buy",
    "notes": "These are things we want to purchase.",
    ...
  }
}
```

Delete Project

DELETE /projects/project-id

A specific, existing project can be deleted by making a DELETE request on the URL for that project.

This method returns an empty data response on success.

Delete a project.

```
# Request
curl -u <api_key>: https://app.asana.com/api/1.0/projects/1331
```

```
# Response
HTTP/1.1 200
{
  "data": {
  }
}
```

Get tasks in a project

```
GET    /projects/project-id/tasks
Returns the list of tasks in this project. Tasks can exist in more than one project at a time.

Get visible tasks on a project

# Request
curl -u <api_key>: https://app.asana.com/api/1.0/projects/1331/tasks

# Response
HTTP/1.1 200
{
  "data": [
  {
    "id": 2001,
    "name": "Catnip"
  },
  {
    "id": 2002,
```

```
   "name": "Kitty litter"
  },
  ...
  ]
}
```

Create Task

```
GET    /projects/project-id/tasks
Returns the list of tasks in this project. Tasks can exist in more than one project at a time.


Get visible tasks on a project


# Request
curl -u <api_key>: https://app.asana.com/api/1.0/projects/1331/tasks


# Response
HTTP/1.1 200
{
  "data": [
  {
    "id": 2001,
    "name": "Catnip"
  },
  {
    "id": 2002,
    "name": "Kitty litter"
  },
  ...
  ]
}
```

Delete task

DELETE /tasks/task-id

A specific, existing task can be deleted by making a DELETE request on the URL for that task.

This method returns an empty data response on success.

Delete a task

Request

curl -u <api_key>: https://app.asana.com/api/1.0/tasks/1001

Response

HTTP/1.1 200

```
{
  "data": {
  }
}
```