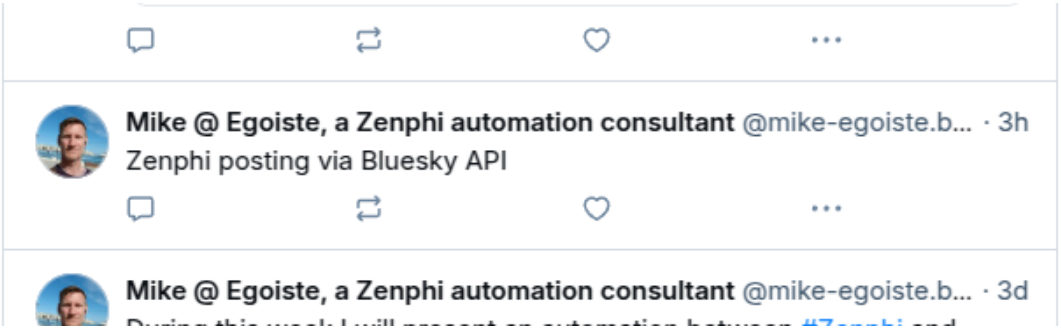
Connecting to Bluesky API Documentation

How to create a post in Bluesky Social using Zenphi API call



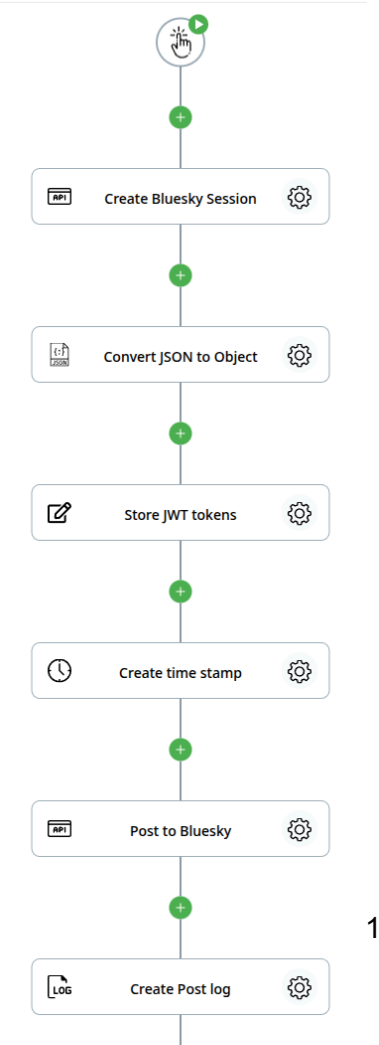
To use Bluesky we currently can call its API via the Zenphi's "Make HTTP Request"-action

Depending on the action you are looking to perform, there are different URLs to call. In the example below we will post a message to our timeline, and to do that we need to communicate with two URLs.

Prerequisites: An **App Password** (Basically your API Key). Obtain your password here : https://bsky.app/settings/app-passwords

I strongly recommend that you store the App Password in the Zenphi Vault and refer to it in your flow when needed.

The flow consists of the following steps:



A first API call is made with **Make HTTP Request** `/xrpc/com.atproto.server.createSession` asking to create a session. It will respond back with an *Access JWT* and a *Refresh JWT* token among other things.

We use **Convert JSON to Object** to extract the two JWT tokens. For your logging purposes, this is where you also extract other details.

Then we store the JWT tokens in two variables, aptly called AccessJWT and RefreshJWT with **Set Variable**

When creating a post (or record in Bluesky language), we need to provide an accurate timestamp. The **Current Date Time** will let us format this correctly.

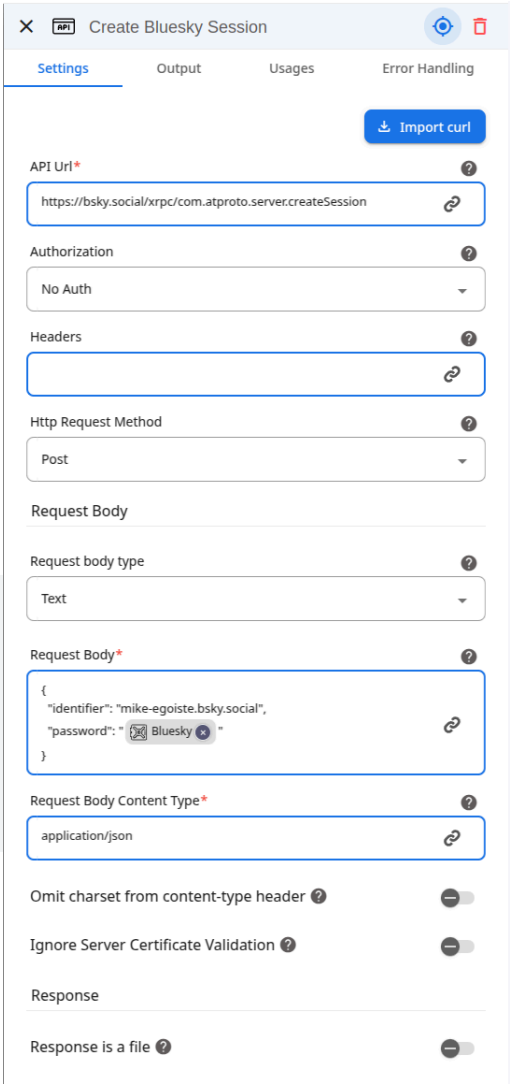
Now we are ready to post, again using the **Make HTTP Request**, and the URL we call is

`/xrpc/com.atproto.repo.createRecord` where the Access JWT token is presented, the text to be posted and the time stamp we created.

Log the response back for reference, if you for example want the ability to purge the post or otherwise keep track of what is going on.

Now, let's look at each action in the flow and how it is configured.

First we add a "**Make HTTP Request**" action (which I then renamed to "Create Bluesky Session").



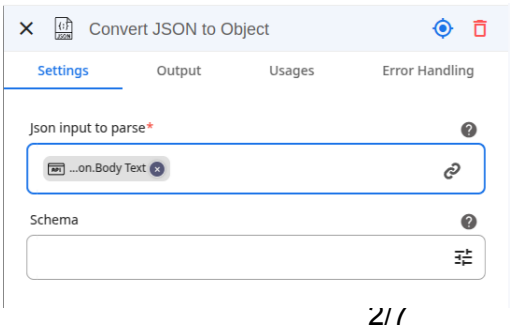
- *API Url* is set to `/xrpc/com.atproto.server.createSession` using https://bsky.social as entryway.
    - If you have a specific PDS, such as *nnnn.us-east.host.bsky.network*, you would use that for entryway instead.
- Set *Authorization* to "`No Auth`"
- Leave *Headers* empty
- Set *Http Request Method* to "`Post`"
- In the *Request Body*, enter the following:

```
Unset
{
  "identifier": "your-handle.bsky.social",
  "password": "your-app password"
}
```

    - If you saved your App Password in the Zenphi vault (recommended) you can simply refer to it here.
- Then set *Request Body Content Type* to `application/json`

Next, add the "**Convert JSON to Object**" action.

Set the field *Json input to parse* to the Body Text output from the Make HTTP Request action.



We also need to declare the JSON *Schema*, which you can do easily by logging the output of the JSON source and paste that into the Sample JSON Payload column and have it automatically set up.

This is the schema if you prefer, I have set the type size to really small save space:

Unset

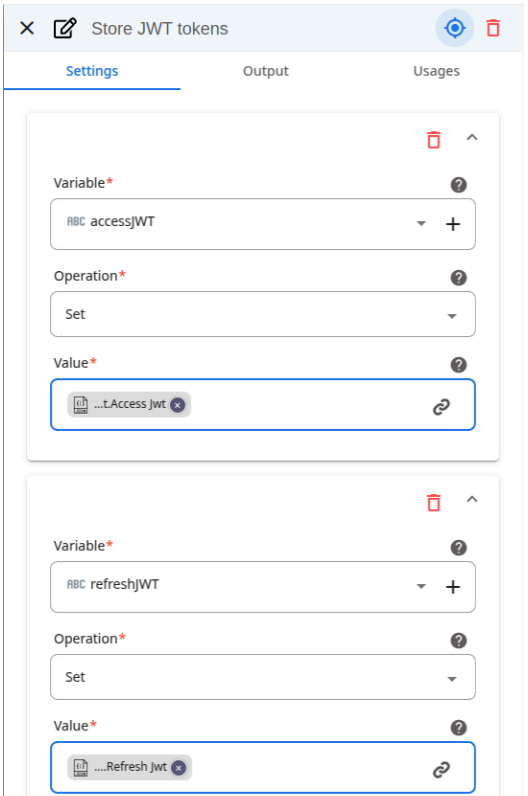
```
{
    "type": "object",
    "properties": {
        "did": {
            "type": "string",
            "title": "Did"
        },
        "didDoc": {
            "type": "object",
            "properties": {
                "@context": {
                    "type": "array",
                    "items": {
                        "type": "string"
                    },
                    "title": "Context"
                },
                "id": {
                    "type": "string",
                    "title": "Id"
                },
                "alsoKnownAs": {
                    "type": "array",
                    "items": {
                        "type": "string"
                    },
                    "title": "Also Known As"
                },
                "verificationMethod": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {
                            "id": {
                                "type": "string",
                                "title": "Id"
                            },
                            "type": {
                                "type": "string",
                                "title": "Type"
                            },
                            "controller": {
                                "type": "string",
                                "title": "Controller"
                            },
                            "publicKeyMultibase": {
                                "type": "string",
                                "title": "Public Key Multibase"
                            }
                        }
                    },
                    "title": "Verification Method"
                },
                "service": {
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {
                            "id": {
                                "type": "string",
                                "title": "Id"
                            },
```

```
                            "type": {
                                "type": "string",
                                "title": "Type"
                            },
                            "serviceEndpoint": {
                                "type": "string",
                                "title": "Service Endpoint"
                            }
                        }
                    },
                    "title": "Service"
                }
            },
            "title": "Did Doc"
        },
        "handle": {
            "type": "string",
            "title": "Handle"
        },
        "email": {
            "type": "string",
            "title": "Email"
        },
        "emailConfirmed": {
            "type": "boolean",
            "title": "Email Confirmed"
        },
        "emailAuthFactor": {
            "type": "boolean",
            "title": "Email Auth Factor"
        },
        "accessJwt": {
            "type": "string",
            "title": "Access Jwt"
        },
        "refreshJwt": {
            "type": "string",
            "title": "Refresh Jwt"
        },
        "active": {
            "type": "boolean",
            "title": "Active"
        }
    },
    "$schema": "http://json-schema.org/draft-04/schema"
}
```


As you can see, there is quite a lot of information available for your housekeeping tasks. The ones we are looking for is the JWT tokens *accessJwt* and *refreshJwt*

Store these two in two variables named to fit in the following action “**Set Variable**”.

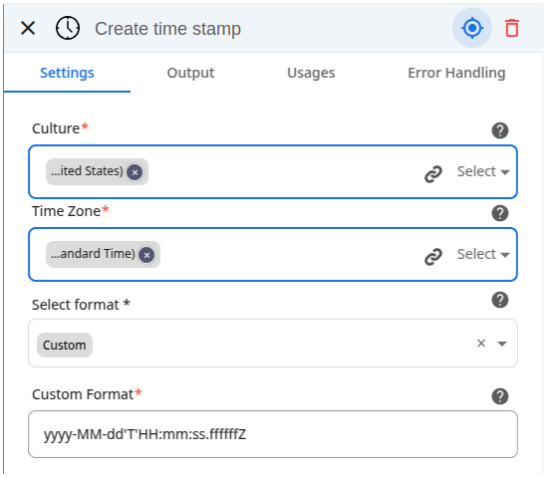
You store both variables in the same action step, as you see in the picture.



We now need a time stamp, and that is created with the action “**Current Date Time**”.

Set to your *Culture* and *Time Zone*

Set *Select format* to `Custom` and enter the following into *Custom Format* field:
`yyyy-MM-dd'T'HH:mm:ss.ffffffZ`



Finally we can do the call to create a record and get our post out there.

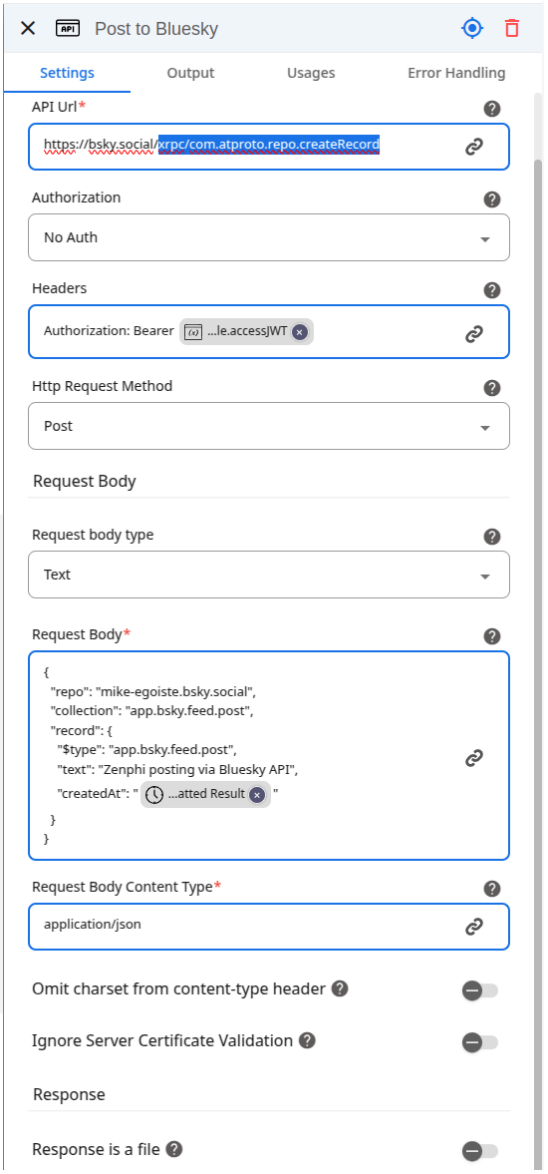
- *API Url* is set to `/xrpc/com.atproto.repo.createRecord`
- Set *Authorization* to “`No Auth`”
- Configure one line in *Headers* to say:

“`Authorization: Bearer [accessJWT token]`

- Set *Http Request Method* to “`Post`”
- In the *Request Body*, enter the following:

```
Unset
{
  "repo": "your-handle.bsky.social",
  "collection": "app.bsky.feed.post",
  "record": {
    "$type": "app.bsky.feed.post",
    "text": "Hello World!", //Your message
    "createdAt": "[Time Stamp Token]"
  }
}
```

- Set *Request Body Content Type* to `application/json`



The response back will contain some key information about this record / post that you may want to save. At least log the output for future reference.

The Status Code for a successful call is 200, useful if you are working with dynamic flows, handling several call events and accounts.

The response will be in a JSON formatted output, handle the response with the action Convert JSON to Object.

What if you want to post an image?

You will first need to upload the image to Bluesky blob service, (`xrpc/com.atproto.repo.uploadBlob`) in a second API call with the content type set to fit the image type, for example `Content-Type: image/jpeg`.

In the Request Body you enclose the raw binary data for the image.

The response back will look something like this:

```
Unset
{
  "blob": {
    "$type": "blob",
    "ref": {
      "$link": "bafkreipx9zldqt8m6u2a1c7g4f8"
    },
    "mimeType": "image/jpeg",
    "size": 848484
  }
}
```

Extract the ref value (“bafkreipx9zldqt8m6u2a1c7g4f8”), this is the link to the image that you need in your post.

And now create the post in the third call, referring to the blob reference in response above, you do the same as you did in the text post earlier, but extend the Request Body as follows:

```
Unset
{
  "repo": "your-handle.bsky.social",
  "collection": "app.bsky.feed.post",
  "record": {
    "$type": "app.bsky.feed.post",
    "text": "This is my post with an image!",
    "createdAt": "[Time-Stamp-Token]",
    "embed": {
      "$type": "app.bsky.embed.images",
```

```
      "images": [
        {
          "image": {
            "$link": "bafkreipx9zldqt8m6u2a1c7g4f8"
          },
          "alt": "An alt image description for accessibility."
        }
      ]
    }
  }
}
```

Also, the refreshJwt token, what is it for? It is to renew your accessJwt, it has a longer expiry than the accessJwt. Something very useful when you build this into an automation flow!

The full API reference for Bluesky is available at
https://docs.bsky.app/docs/category/http-reference

That is all, have fun! Let me know if you have comments or suggestions.

Mikael Klambro
Egoiste Zenphi Application Consultant