+++ title = “OAuth authentication” description = “Grafana OAuthentication Guide” keywords = [“grafana”, “configuration”, “documentation”, “oauth”] weight = 500 +++

Generic OAuth authentication

You can configure many different OAuth2 authentication services with Grafana using the generic OAuth2 feature. Examples:

- Generic OAuth authentication
    - Set up OAuth2 with Auth0
    - Set up OAuth2 with Bitbucket
    - Set up OAuth2 with Centrify
    - Set up OAuth2 with OneLogin
    - JMESPath examples
        * Role mapping
        * Groups mapping

This callback URL must match the full HTTP address that you use in your browser to access Grafana, but with the suffixed path of `/login/generic_oauth`.

You may have to set the `root_url` option of `[server]` for the callback URL to be correct. For example in case you are serving Grafana behind a proxy.

Example config:

```
[auth.generic_oauth]
name = OAuth
icon = signin
enabled = true
client_id = YOUR_APP_CLIENT_ID
client_secret = YOUR_APP_CLIENT_SECRET
scopes =
empty_scopes = false
auth_url =
token_url =
api_url =
allowed_domains = mycompany.com mycompany.org
allow_sign_up = true
tls_skip_verify_insecure = false
tls_client_cert =
tls_client_key =
tls_client_ca =
use_pkce = true
```

Set `api_url` to the resource that returns OpenID UserInfo compatible information.

You can also specify the SSL/TLS configuration used by the client.

- Set `tls_client_cert` to the path of the certificate.
- Set `tls_client_key` to the path containing the key.
- Set `tls_client_ca` to the path containing a trusted certificate authority list.

`tls_skip_verify_insecure` controls whether a client verifies the server's certificate chain and host name. If it is true, then SSL/TLS accepts any certificate presented by the server and any host name in that certificate. *You should only use this for testing*, because this mode leaves SSL/TLS susceptible to man-in-the-middle attacks.

Set `empty_scopes` to true to use an empty scope during authentication. By default, Grafana uses `user:email` as scope.

Email address

Grafana determines a user's email address by querying the OAuth provider until it finds an e-mail address:

1. Check for the presence of an e-mail address via the `email` field encoded in the OAuth `id_token` parameter.
2. Check for the presence of an e-mail address using the JMESPath specified via the `email_attribute_path` configuration option. The JSON used for the path lookup is the HTTP response obtained from querying the UserInfo endpoint specified via the `api_url` configuration option. **Note**: Only available in Grafana v6.4+.
3. Check for the presence of an e-mail address in the `attributes` map encoded in the OAuth `id_token` parameter. By default Grafana will perform a lookup into the attributes map using the `email:primary` key, however, this is configurable and can be adjusted by using the `email_attribute_name` configuration option.
4. Query the `/emails` endpoint of the OAuth provider's API (configured with `api_url`), then check for the presence of an email address marked as a primary address.
5. If no email address is found in steps (1-4), then the email address of the user is set to an empty string.

Roles

Grafana checks for the presence of a role using the JMESPath specified via the `role_attribute_path` configuration option. The JMESPath is applied to the `id_token` first. If there is no match, then the UserInfo endpoint specified via the `api_url` configuration option is tried next. The result after evaluation of the `role_attribute_path` JMESPath expression should be a valid Grafana role, for example, `Viewer`, `Editor` or `Admin`.

For more information, refer to the JMESPath examples.

Groups / Teams

Similarly, group mappings are made using JMESPath with the `groups_attribute_path` configuration option. The `id_token` is attempted first, followed by the UserInfo from the `api_url`. The result of the JMESPath expression should be a string array of groups.

Furthermore, Grafana will check for the presence of at least one of the teams specified via the `team_ids` configuration option using the JMESPath specified via the `team_ids_attribute_path` configuration option. The JSON used for the path lookup is the HTTP response obtained from querying the Teams endpoint specified via the `teams_url` configuration option (using `/teams` as a fallback endpoint). The result should be a string array of Grafana Team IDs. Using this setting ensures that only certain teams is allowed to authenticate to Grafana using your OAuth provider.

Login

Customize user login using `login_attribute_path` configuration option. Order of operations is as follows:

1. Grafana evaluates the `login_attribute_path` JMESPath expression against the ID token.
2. If Grafana finds no value, then Grafana evaluates expression against the JSON data obtained from UserInfo endpoint. The UserInfo endpoint URL is specified in the `api_url` configuration option.

You can customize the attribute name used to extract the ID token from the returned OAuth token with the `id_token_attribute_name` option.

You can set the user's display name with JMESPath using the `name_attribute_path` configuration option. It operates the same way as the `login_attribute_path` option.

> **Note:** `name_attribute_path` is available in Grafana 7.4+.

PKCE

> Available in Grafana v8.3 and later versions.

IETF's RFC 7636 introduces "proof key for code exchange" (PKCE) which introduces additional protection against some forms of authorization code interception attacks. PKCE will be required in OAuth 2.1.

You can enable PKCE in Grafana by setting `use_pkce` to `true` in the `[auth.generic_oauth]` section.

```
use_pkce = true
```

Grafana always uses the SHA256 based `S256` challenge method and a 128 bytes (base64url encoded) code verifier.

Set up OAuth2 with Auth0

1. Create a new Client in Auth0
   - Name: Grafana
   - Type: Regular Web Application
2. Go to the Settings tab and set:
   - Allowed Callback URLs: `https://<grafana domain>/login/generic_oauth`
3. Click Save Changes, then use the values at the top of the page to configure Grafana:

   ```
   [auth.generic_oauth]
   enabled = true
   allow_sign_up = true
   team_ids =
   allowed_organizations =
   name = Auth0
   client_id = <client id>
   client_secret = <client secret>
   scopes = openid profile email
   auth_url = https://<domain>/authorize
   token_url = https://<domain>/oauth/token
   api_url = https://<domain>/userinfo
   use_pkce = true
   ```

Set up OAuth2 with Bitbucket

```
[auth.generic_oauth]
name = BitBucket
enabled = true
allow_sign_up = true
client_id = <client id>
client_secret = <client secret>
scopes = account email
auth_url = https://bitbucket.org/site/oauth2/authorize
token_url = https://bitbucket.org/site/oauth2/access_token
api_url = https://api.bitbucket.org/2.0/user
teams_url = https://api.bitbucket.org/2.0/user/permissions/workspaces
team_ids_attribute_path = values[*].workspace.slug
team_ids =
allowed_organizations =
```

Set up OAuth2 with Centrify

1. Create a new Custom OpenID Connect application configuration in the Centrify dashboard.

2. Create a memorable unique Application ID, e.g. “grafana”, “grafana_aws”, etc.
3. Put in other basic configuration (name, description, logo, category)
4. On the Trust tab, generate a long password and put it into the OpenID Connect Client Secret field.
5. Put the URL to the front page of your Grafana instance into the “Resource Application URL” field.
6. Add an authorized Redirect URI like https://your-grafana-server/login/generic_oauth
7. Set up permissions, policies, etc. just like any other Centrify app
8. Configure Grafana as follows:

```
[auth.generic_oauth]
name = Centrify
enabled = true
allow_sign_up = true
client_id = <OpenID Connect Client ID from Centrify>
client_secret = <your generated OpenID Connect Client Secret"
scopes = openid profile email
auth_url = https://<your domain>.my.centrify.com/OAuth2/Authorize/<Application ID>
token_url = https://<your domain>.my.centrify.com/OAuth2/Token/<Application ID>
api_url = https://<your domain>.my.centrify.com/OAuth2/UserInfo/<Application ID>
```

Set up OAuth2 with OneLogin

1. Create a new Custom Connector with the following settings:

   - Name: Grafana
   - Sign On Method: OpenID Connect
   - Redirect URI: `https://<grafana domain>/login/generic_oauth`
   - Signing Algorithm: RS256
   - Login URL: `https://<grafana domain>/login/generic_oauth`

   then:

2. Add an App to the Grafana Connector:

   - Display Name: Grafana

   then:

3. Under the SSO tab on the Grafana App details page you’ll find the Client ID and Client Secret.

   Your OneLogin Domain will match the URL you use to access OneLogin.

   Configure Grafana as follows:

```
[auth.generic_oauth]
name = OneLogin
enabled = true
allow_sign_up = true
client_id = <client id>
client_secret = <client secret>
scopes = openid email name
auth_url = https://<onelogin domain>.onelogin.com/oidc/2/auth
token_url = https://<onelogin domain>.onelogin.com/oidc/2/token
api_url = https://<onelogin domain>.onelogin.com/oidc/2/me
team_ids =
allowed_organizations =
```

JMESPath examples

To ease configuration of a proper JMESPath expression, you can test/evaluate expressions with custom payloads at http://jmespath.org/.

Role mapping

If therole_attribute_path property does not return a role, then the user is assigned the `Viewer` role by default. You can disable the role assignment by setting `role_attribute_strict = true`. It denies user access if no role or an invalid role is returned.

Basic example:

In the following example user will get `Editor` as role when authenticating. The value of the property `role` will be the resulting role if the role is a proper Grafana role, i.e. `Viewer`, `Editor` or `Admin`.

Payload:

```
{
    ...
    "role": "Editor",
    ...
}
```

Config:

```
role_attribute_path = role
```

Advanced example:

In the following example user will get `Admin` as role when authenticating since it has a role `admin`. If a user has a role `editor` it will get `Editor` as role, otherwise `Viewer`.

Payload:

```
{
    ...
    "info": {
        ...
        "roles": [
            "engineer",
            "admin",
        ],
        ...
    },
    ...
}
```

Config:

```
role_attribute_path = contains(info.roles[*], 'admin') && 'Admin' || contains(info.roles[*]
```

Groups mapping

> Available in Grafana Enterprise v8.1 and later versions.

With Team Sync you can map your Generic OAuth groups to teams in Grafana so that the users are automatically added to the correct teams.

Generic OAuth groups can be referenced by group ID, like `8bab1c86-8fba-33e5-2089-1d1c80ec267d` or `myteam`.

[Learn more about Team Sync]({{< relref "team-sync.md" >}})

Config:

```
groups_attribute_path = info.groups
```

Payload:

```
{
    ...
    "info": {
        ...
        "groups": [
            "engineers",
            "analysts",
        ],
        ...
    },
    ...
}
```