Bridging

Bridging directories

Bridges' job is to turn the contents of an external data directory into a data structure that can then be then constructed on the Matrix server by the Provisioner. See State representation for the details description of the data structure being produced.

See specific bridges (in the sidebar) to learn more about how GS interprets the contents of specific data sources.

Bridges run continously and trigger provisioning whenever they observer changes in the data source.

LDAP

The LDAP bridge will periodically (according to its configuration) fetch the LDAP tree from the server (filtering out the things it doesn't find interesting).

To enable maximum flexibility it "flattens" the LDAP tree so that the users' (and groups') place in directory tree doesn't matter.

Groups, OrgUnits and Domains (if found), will all be flattened and treated like a container for users. It makes it possible to use their DNs[^note] (fully qualified names) to assign users to spaces and power levels in those spaces.

[^note]: CNs are also allowed here for backwards compatibility reasons, but **only for groups**. It is however advised to avoid using CNs and use DNs instead, since they are guaranteed to be unique across the LDAP tree. GS' behaviour is undefined when mapping groups with duplicate names.

For example, for the following LDAP tree:

```
- Company (Domain) (`dc=company`)
  - Alfred (User) (`cn=alfred,cn=company`)
  - Engineering (OrgUnit) (`ou=engineering,cn=company`)
    - Barbara (User) (`cn=barbara,ou=engineering,cn=company`)
    - Moderators (Group) (`cn=moderators,ou=engineering,cn=company`)
      - Charlie (User) (`cn=charlie,ou=engineering,cn=company`)
```

Company, Engineering and Moderators will all be treated as if they were an group. We could then use the following space mapping configuration with it:

```
spaces:
  id: root
  name: "Company"
  groups:
    - externalId: `dc=company` # or leave it empty with the same result
  subspaces:
    - id: engineering
      name: Engineering
      groups:
        - externalId: `ou=engineering,cn=company`
        - externalId: `cn=moderators,ou=engineering,cn=company`
          powerLevel: 50
```

Example Configuration

When using the helm chart, the authentication schema is automatically used to configure GroupSync LDAP source. If you want to override some settings, you can always implement the following configuration:

```
source:
  type: 'ldap'
  # LDAP will be checked for changes every this many seconds
  check_interval_seconds: 60
  # The following can be copied straight from Synapse's homeserver.yaml
  # if you're already using its LDAP password provider
  uri: "ldap://element.test"
  # The base ou we specify here will become the root space
  base: "ou=employees,dc=element,dc=test"
  # Optional. An LDAP filter to use when searching for entries
  filter: '(!(ou=Domain Controllers))'
  # Make sure the account you use here has enough permissions to perform searches on your `base`
  bind_dn: "ELEMENT\\administrator"
  bind_password: "donkey.8"
  # Needs `uid` to be able to determine Matrix localparts for users
  # and `name`s to pick the right names for spaces
  attributes:
    uid: "sAMAccountName"
```

```
    name: "name"
  # If the LDAP server requires a client certificate, enable this option.
  # cert:
    # The path to the file
    # file: "./my-cert-file.pem"
    # OR the PEM-encoded cert itself
    # cert: "foobar"
    # Passphrase for the cert, if required.
    # passphrase: "passphrase"
```

Microsoft Graph

The MsGraph bridge will periodically (according to its configuration) perform the following API calls:

- `/organization`, to determine the name of the organization
- `/users` to get the list of users in the org
- `/groups` to get the list of groups in the org
- `/groups/<id>/members` to get a list of members for a particular group

In order to perform the queries successfully, Advanced Identity Management's Application needs to have the following permissions granted in Azure:

- `User.Read.All`
- `GroupMember.Read.All`

It emits a list of users and groups as-is, without performing any transformations on them.

Example Configuration

Using MS-Graph requires the following GroupSync configuration :

```
source:
  type: 'ms-graph-ad'

  # This is the "Tenant ID" from your Azure Active Directory Overview
  tenant_id: 'b9355cb3-feed-dead-beef-9cc325f0335b'
```

```
# Register your app in "App registrations". This will be its "Application (client) ID"
client_id: '5c955b66-18b3-42de-bb5a-13b5a202d4fc'

# Go to "Certificates & secrets", and click on "New client secret".
# This will be the "Value" of the created secret (not the "Secret ID").
client_secret: 'yOb7Q~Km~~YMKzpeq73swJj3kOeJpUwXSZamr'
# For the bridge to be able to operate correctly, navigate to API permissions and unsure
# it has access to GroupMember.Read.All and User.Read.All
# Application permissions for Microsoft Graph. Remember to grant the admin consent for those.

# Optional. The url to reach Graph on. Override if your deployment uses a specific graph endpoint.
base_url: 'https://graph.microsoft.com/'

# Optional. Specific scopes to set for graph to use.
scopes: ['https://graph.microsoft.com/.default']
```

SCIM

The SCIM bridge maintains an HTTP service that conforms to the SCIM protocol (RFC 7644) and provisions a Matrix server with the SCIM resources sent to it.

Configuration

The following options are available when configuring the SCIM bridge:

- `base_url` (optional) - the URL prefix for each route. For instance, with `base_url` set to `/scim/azure` the requests need to be hitting `/scim/azure/Users` etc. Useful when running behind a non-rewriting proxy. Set to an empty string by default.
- `client` – a structure with the following fields:
  - `id` - a string specifying the name of the client, e.g. the name of the organization. Must be unique across SCIM bridge instances running on the server. Changing this value after some SCIM resources have been provisioned is equivalent to creating a new user/group database and a new Matrix Space. The value will also be the default name of the organization Space (which can later be changed by Space Moderators).
  - `token` - access token for the SCIM client, as per RFC 6750
  - `attributeMapping` - a structure with the following fields:

    - `username` - the SCIM user attribute to use when determining the Matrix username. If you're using OIDC, make sure it matches its setup.
- `synchronous_provisioning` (optional) - a boolean flag. If set to true, SCIM responses won't be sent before the Matrix provisioning finishes, and any Matrix errors may cause SCIM requests to fail and potentially leave the server in an invalid state. Useful for testing. False by default, and it's strongly recommended to leave it that way.

Example configuration

Configuring the SCIM bridge requires to configure the following values. When using ESS Helm Chart, you need to set `groupSync.enableSCIM` to expose the SCIM ingress. It will be abailable under GroupSync ingress if it is enabled, or Synapse ingress at `/scim/v2` path.

```
source:
  type: 'scim'
  client:
    # Unique ID for the SCIM client.
    # This will be used to keep track of the managed Space and User/Group storage in Matrix.
    id: 'element-ad'
    # You can set up multiple client tokens with different permission levels.
    rbac:
        # Bearer token for the client, as per RFC 6750
      - token: 'foo-bar-baz'
        # What's the token allowed to do: in this case, everything (read+write on all endpoints).
        # The format for these is 'access:scope', access being 'read', 'write' or '*' for both,
        # scope being 'users', 'groups' or '*' for everything.
        roles: ['*:*']
        # You can specify permissions for anyone who presents a valid Matrix access_token for an admin user
      - synapse_user: 'admin'
        # ...and assign more fine-tuned permissions to it
        roles: ['read:*', 'write:groups']
    attributeMapping:
      # The SCIM user attribute that'll be used as the Matrix username for provisioned users
      username: 'externalId'
  # Should SCIM user creation register a Matrix account for the user.
  # Possible values are 'yes', 'no' and 'if-missing'
  # - 'yes' will register Matrix accounts on the server upon a SCIM create user request,
  #   and error out if the user with that username already exists.
  # - 'if-missing' will register Matrix accounts unless they exist already.
  #   This is useful if some users have their user accounts created independently before the SCIM bridge was set
```

```
up.
  # - 'no' will not create user accounts, only work with existing ones.
  register_users: 'no'
  # Optional: Should SCIM responses wait for Matrix provisioning to complete.
  # It is recommended to leave it as false. HTTP responses will be sent quicker,
  # and Matrix provisioning may still fail in the background (to be retried later).
  synchronous_provisioning: false
  # Optional: Configure a mailer to send email notifications to newly registered, activated and deactivated users.
  # mailer:
  #   # The email address emails will be sent from
  #   from: 'element@element.com'
  #   # Path to a directory with email templates.
  #   # Each template should be a directory containing 'subject.pug', 'text.pug' and 'html.pug',
  #   # all using https://pugjs.org/ as a template language.
  #   # Group sync ships with standard, Element-branded templates in templates/
  #   templates_path: './templates'
  #   # SMTP transport configuration, as per https://nodemailer.com/smtp/,
  #   # except that we default `secure` to `true` and `port` to 465.
  #   transport:
  #     host: 'smtp.example.com'
  #     auth:
  #       user: 'mailer'
  #       pass: 'mailerpass.8'
```

Revision #2
Created 15 May 2025 09:06:53 by Gaël Goinvic
Updated 27 May 2025 06:39:39 by Gaël Goinvic