How do I migrate to SSO while keeping my original accounts?

Issue

- When having previously used Matrix without SSO, after setting up SSO, new matrix accounts are created for all users. Users would prefer to keep using there existing accounts, whilst still making use of SSO.

Environment

- Element On-Premise
- Element Cloud
- SSO Configured

Resolution

Transferring SSO `external_ids` to original users

To transfer 'external_ids' from SSO accounts, to your original accounts you will need to use the Admin API.

Admin API Documentation

Getting an Access Token

Before being able to use the Admin API, you will need an admin account and it's 'Access Token', you can make a user a Synapse Admin by either following the steps in the link above (required for On-Premise), or following these steps on the EMS Control Panel:

1. Access the 'Server Admin' tab
2. Under the 'Users' tab, select the user that should be made Synapse Admin
3. Click the checkbox, next to 'Synapse Admin' and click 'Yes' to confirm

Once a user is a Synapse Admin, you can retrieve their 'Access Token' by logging in via the Element Matrix client:

1. Click on the users' profile icon in the top-left and select 'All Settings'
2. Open the 'Help & About' settings page, then scroll down to the 'Advanced' section
3. Click 'Access Token' to reveal the token, copy this to interact with the Admin API

Using an Access Token

Access tokens will need to be passed into all API requests as an Authorization Bearer token, see examples below:

Bash:

```
curl --request GET 'https://example.com/_synapse/admin/v2/users?from=0&limit=10&guests=false' \
--header 'Authorization: Bearer syt_adminToken'
```

Windows:

```
$headers.Add("Authorization", "Bearer syt_adminToken")
```

Python:

```
import requests
headers = {
    'Authorization': 'Bearer syt_adminToken',
}
```

Customising commands for your specific enviroment

The commands to follow are generic and will need to be modified to suit your specific environment:

- Replace `example.com` with your homseserver hostname.

    - Note, for EMS customers, this is always the .ems.host domain - even if your server uses Custom DNS.
- Replace `@user:example.com` with the Matrix ID of the user you want to edit.
    - Remember to replace `example.com` per the above step
- Replace `syt_adminToken` with an access token on your server with admin privileges.
    - See previous section for how to obtain
- Replace `saml` in `"auth_provider": "saml"`, with the SSO type you are using.
- Replace `email@address.com` in `"external_id": "email@address.com"` with the user's ID in your SSO provider.

Getting SSO users' `external_ids`

Admin API List Accounts

1. Using the link above, you can use a GET request to `/_synapse/admin/v2/users` to retrieve a list of all accounts on your home server. Follow the guidance on pagination to ensure all users are retrieved.
   Bash:

   ```
   curl --request GET 'https://example.com/_synapse/admin/v2/users?from=0&limit=10&guests=false' \
   --header 'Authorization: Bearer syt_adminToken'
   ```

   Windows:

   ```
   $headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
   $headers.Add("Authorization", "Bearer syt_adminToken")

   $response = Invoke-RestMethod
   'https://example.com/_synapse/admin/v2/users?from=0&limit=10&guests=false' -Method 'GET' -
   Headers $headers
   $response | ConvertTo-Json
   ```

Admin API Query User Account

2. For each user you can then use their `name` in another GET request to `/_synapse/admin/v2/users/<user_id>`, replacing `<user_id>` with `name`.
   Bash:

   ```
   curl --request GET 'https://example.com/_synapse/admin/v2/users/@user:example.com' \
   --header 'Authorization: Bearer syt_adminToken'
   ```

   Windows:

   ```
   $headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
   $headers.Add("Authorization", "Bearer syt_adminToken")
   ```

```
$response = Invoke-RestMethod 'https://example.com/_synapse/admin/v2/users/@user:example.com'
-Method 'GET' -Headers $headers
$response | ConvertTo-Json
```

3. You will find the `external_ids` for each user within the JSON output. You can programatically run through all users and generate a list of only those with `external_ids`, removing unneeded information. (See example below)
Python:

```
import requests

# REPLACE THESE VALUES WITH ACCESS TOKEN AND HOME SERVER URL
headers = {
    'Authorization': 'Bearer syt_adminToken',
}
url = 'https://example.com'

# GET LIST OF ALL USERS ON HOME SERVER
# OUTPUT: 'all_users' contains a list of all users
next_token = '0'
last_token = ''
all_users = []
get_users = requests.get(url + '/_synapse/admin/v2/users?from=' + next_token +
'&limit=10&guests=false', headers=headers).json()
for user in get_users['users']:
    all_users.append(user['name'])
while ('next_token' in get_users) and (next_token != last_token):
    next_token = get_users['next_token']
    get_users = requests.get(url + '/_synapse/admin/v2/users?from=' + next_token +
'&limit=10&guests=false', headers=headers).json()
    for user in get_users['users']:
        all_users.append(user['name'])

# FOR EACH USER, GET ALL INFO, EXCLUDE THOSE WITHOUT 'external_ids'
# OUTPUT: 'all_external_ids' contains a list of all users with external ids
all_external_ids = []
for user in all_users:
    get_user = requests.get(url + '/_synapse/admin/v2/users/' + user, headers=headers).json()
    if get_user['external_ids'].__len__() != 0:
        all_external_ids.append(
```

```
        {
            'sso_username': user,
            'original_username': '',
            'external_ids': get_user['external_ids']
        }
    )
```

Transferring SSO `external_ids` information

Admin API Create or Modify Account

With all `external_ids` collected, you will need to identify each SSO Account and the associated original account that you'd like to transfer the associated SSO information over too.

If using the Python example above, you will need to store the original username within `all_external_ids[X]['original_username']`, replacing X with the index of the SSO user. If you create a dictionary with `keys` named of the SSO username, and `values` of the desired Original username you could use the following to update `all_external_ids`:

Python:

```
# ADD REQUIRED ORIGINAL USERNAME
for user in all_external_ids:
    dict_storing_sso2orig = {'@example_sso_user:example.com': '@example_orig_user:example.com'}
    user['original_username'] = dict_storing_sso2orig[str(user['sso_username'])]
```

Once you have related all SSO Usernames to Original Usernames you can then, using the link above, use a PUT request to `/_synapse/admin/v2/users/<user_id>` to change the `external_ids` data for each account.

- Remember to remove the `external_ids` information from the soon to be defunct SSO accounts, this is done by sending blank data `{"external_ids":[]}`
- Adding of `external_ids` should follow the below JSON format:

```
{
  "external_ids": [
    {
      "auth_provider": "<provider1>",
      "external_id": "<user_id_provider_1>"
    },
    {
      "auth_provider": "<provider2>",
```

```
    "external_id": "<user_id_provider_2>"
  }
]
 }
```

Bash:

- Removing `external_ids`:

```
curl --request PUT 'https://example.com/_synapse/admin/v2/users/@user:example.com' \
--header 'Authorization: Bearer syt_adminToken' \
  --header 'Content-Type: application/json' \
  --data-raw '{"external_ids":[]}'
```

- Adding `external_ids`:

```
curl --location --request PUT 'https://example.ems.host/_synapse/admin/v2/users/@user:example.com'
\
--header 'Authorization: Bearer syt_adminToken' \
--header 'Content-Type: application/json' \
--data-raw '{
  "external_ids": [
    {
    "auth_provider": "saml",
    "external_id": "email@address.com"
    }
  ]
}'
```

Windows:

- Removing `external_ids`:

```
$headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Authorization", "Bearer syt_adminToken")
$headers.Add("Content-Type", "application/json")

$body = "{`"external_ids`": []}"

$response = Invoke-RestMethod
'https://example.ems.host/_synapse/admin/v2/users/@user:example.com' -Method 'PUT' -Headers
$headers -Body $body
```

```
$response | ConvertTo-Json
```

- Adding `external_ids`:

```
$headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Authorization", "Bearer syt_adminToken")
$headers.Add("Content-Type", "application/json")

$body = "{`"external_ids`": [{`"auth_provider`": `"saml`",`"external_id`":
`"email@address.com`"}]}"

$response = Invoke-RestMethod 'https://example.com/_synapse/admin/v2/users/@user:example.com'
-Method 'PUT' -Headers $headers -Body $body
$response | ConvertTo-Json
```

Continuing with the Python example, you can now use this to remove the `external_ids` from each SSO account, and add that information to the associated Original account.

Python:

```
# REMOVE 'external_ids' from 'sso_username' ACCOUNTS FROM 'all_external_ids' THEN
# UPDATE ALL 'original_username' ACCOUNTS FROM 'all_external_ids' WITH 'external_ids' FROM 'sso_username'
for user in all_external_ids:
    data = '{"external_ids":' + str(user['external_ids']).replace("'", '"').replace(" ", "") + '}'
    remove_sso = requests.put(url + '/_synapse/admin/v2/users/' + user['sso_username'], headers=headers,
data='{"external_ids":[]}')
    if remove_sso.status_code == 200:
        add_sso = requests.put(url + '/_synapse/admin/v2/users/' + user['original_username'], headers=headers,
data=data)
```

Python Example

The full python script is available below:

```
import requests

# REPLACE THESE VALUES WITH ACCESS TOKEN AND HOME SERVER URL
headers = {
    'Authorization': 'Bearer syt_adminToken',
```

```
}
url = 'https://example.com'

# GET LIST OF ALL USERS ON HOME SERVER
# OUTPUT: 'all_users' contains a list of all users
next_token = '0'
last_token = ''
all_users = []
get_users = requests.get(url + '/_synapse/admin/v2/users?from=' + next_token + '&limit=10&guests=false',
                         headers=headers).json()
for user in get_users['users']:
    all_users.append(user['name'])
while ('next_token' in get_users) and (next_token != last_token):
    next_token = get_users['next_token']
    get_users = requests.get(url + '/_synapse/admin/v2/users?from=' + next_token + '&limit=10&guests=false',
                             headers=headers).json()
    for user in get_users['users']:
        all_users.append(user['name'])

# FOR EACH USER, GET ALL INFO, EXCLUDE THOSE WITHOUT 'external_ids'
# OUTPUT: 'all_external_ids' contains a list of all users with external ids
all_external_ids = []
for user in all_users:
    get_user = requests.get(url + '/_synapse/admin/v2/users/' + user, headers=headers).json()
    if get_user['external_ids'].__len__() != 0:
        all_external_ids.append(
            {
                'sso_username': user,
                'original_username': '',
                'external_ids': get_user['external_ids']
            }
        )

# ADD REQUIRED ORIGINAL USERNAME
# REPLACE CONTENTS OF 'dict_storing_sso2orig' TO SET UP RELATED SSO -> ORIGINAL ACCOUNTS
# CHANGE 'readme' VARIABLE BELOW TO 'True' TO CONTINUE
readme = False
dict_storing_sso2orig = {'@example_sso_user:example.com': '@example_orig_user:example.com'}
```

```
for user in all_external_ids:
    if readme is True:
        user['original_username'] = dict_storing_sso2orig[str(user['sso_username'])]


# REMOVE 'external_ids' from 'sso_username' ACCOUNTS FROM 'all_external_ids' THEN
# UPDATE ALL 'original_username' ACCOUNTS FROM 'all_external_ids' WITH 'external_ids' FROM 'sso_username'
# CHANGE 'dict_storing_sso2orig_check' VARIABLE BELOW TO 'True' TO CONTINUE
dict_storing_sso2orig_check = False
for user in all_external_ids:
    if (dict_storing_sso2orig_check is True) and (dict_storing_sso2orig != {'@example_sso_user:example.com':
'@example_orig_user:example.com'}):
        data = '{"external_ids":' + str(user['external_ids']).replace("'", '"').replace(" ", "") + '}'
        remove_sso = requests.put(url + '/_synapse/admin/v2/users/' + user['sso_username'], headers=headers,
data='{"external_ids":[]}')
        if remove_sso.status_code == 200:
            add_sso = requests.put(url + '/_synapse/admin/v2/users/' + user['original_username'], headers=headers,
data=data)
```

Root Cause

If SSO is not setup prior to using matrix, new SSO duplicate accounts are created following it's configuration. Users would prefer to keep their existing accounts and associated setup (Rooms / etc.) so migrating `external_ids` from these new SSO accounts to the originals is required.

Revision #16
Created 2 February 2023 14:23:00 by Kieran Mitchell Lane
Updated 6 November 2024 12:49:38 by Kieran Mitchell Lane