Authentication and Authorization Guide

/ ForgeRock Identity Cloud 7.2

Latest update: 7.2.0

ForgeRock AS.
201 Mission St., Suite 2900
San Francisco, CA 94105, USA
+1 415-599-1100 (US)
www.forgerock.com

Abstract

Guide to configuring authentication and authorization.

Table of Contents

Overview

This guide covers authentication, authorization, and delegated administration.

Quick Start

		
Authentication	Authorization & Access Control	Delegated Administration
Authenticate users securely.	Protect REST endpoints with secure authorization and access control.	Use privileges to give fine-grained administrative access to specific users.

ForgeRock Identity Platform™ serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see https://www.forgerock.com.

The ForgeRock Common REST API works across the platform to provide common ways to access web resources and collections of resources.

Chapter 1
Authentication

Authentication is the process of verifying who is requesting access to a resource. The user or application making the request presents credentials, making it possible to prove that the requester is who they claim to be. The goal is to authorize access to specific Identity Cloud resources, depending on the confirmed identity of the user or application making the request.

Authenticate through AM

When you use Identity Cloud and AM together in a *platform deployment*, you configure Identity Cloud to use AM bearer tokens for authentication, instead of setting up traditional authentication modules. This delegates all authentication to AM. In this configuration, Identity Cloud uses an `rsFilter` that replaces all other authentication methods.

With AM bearer tokens, all Identity Cloud endpoints that require authentication are accessed using an authorization header that contains the bearer token, instead of `X-OpenIDM-Username` and `X-OpenIDM-Password`. Endpoints that allow anonymous access can be accessed without a token.

Important

- From Identity Cloud 7.0 onwards, using AM bearer tokens for authentication is the *only supported method* of integrating Identity Cloud with AM.
- To use AM bearer tokens for authentication, your AM configuration must include at least the following configuration:
  - Two OAuth 2.0 clients: an `idm-resource-server` client to introspect the bearer token, and an `idm-provisioning` client used by AM to provision users in Identity Cloud. For information on configuring these clients, see Configure OAuth Clients in the *Platform Setup Guide*.
  - An OAuth 2 provider service.
  - An Identity Cloud provisioning service.

  Your Identity Cloud authentication configuration *(You can manage the authentication configuration over REST at the config/authentication endpoint.)* must include the `rsFilter` configuration and *no other* authentication methods.

+ *Sample rsFilter Authentication*

```
{
    "rsFilter" : {
```

```
        "clientId" : "",
        "clientSecret" : "",
        "tokenIntrospectUrl" : "http://am.example:8080/openam/oauth2/introspect",
        "scopes" : [ ],
        "cache" : {
            "maxTimeout" : "300 seconds"
        },
        "augmentSecurityContext" : {
            "type" : "text/javascript",
            "source" : "require('auth/orgPrivileges').assignPrivilegesToUser(resource, security,
 properties, subjectMapping, privileges, 'privileges', 'privilegeAssignments');"
        },
        "subjectMapping" : [
            {
                "resourceTypeMapping" : {
                    "usr" : "managed/user"
                },
                "propertyMapping" : {
                    "sub" : "_id"
                },
                "userRoles" : "authzRoles/*",
                "additionalUserFields" : [
                    "adminOfOrg",
                    "ownerOfOrg"
                ],
                "defaultRoles" : [
                    "internal/role/openidm-authorized"
                ]
            }
        ],
        "staticUserMapping" : [
            {
                "subject" : "(usr!amadmin)",
                "localUser" : "internal/user/openidm-admin",
                "roles" : [
                    "internal/role/openidm-authorized",
                    "internal/role/openidm-admin"
                ]
            },
            {
                "subject" : "(age!idm-provisioning)",
                "localUser" : "internal/user/idm-provisioning",
                "roles" : [
                    "internal/role/platform-provisioning"
                ]
            }
        ],
        "anonymousUserMapping" : {
            "localUser" : "internal/user/anonymous",
            "roles" : [
                "internal/role/openidm-reg"
            ]
        }
    }
}
```

The `rsFilter` configuration includes the following properties:

`clientId`

The client ID of the AM OAuth 2.0 client used to introspect the bearer token (the `idm-resource-server`) client, in this example).

`clientSecret`

The client secret of the AM OAuth 2.0 client used to introspect the bearer token. Identity Cloud will encrypt this field if it isn't encrypted already.

`tokenIntrospectUrl`

The URI to reach the `oauth2/introspect` endpoint in AM, for example, `http://am.example:8080/openam/oauth2/introspect`.

`scopes`

Any scopes that are required to be present in the access token. This will vary depending on your configuration. For more information about scopes, see About Scopes in the AM *OAuth 2.0 Guide*.

`cache`

Sets the `maxTimeout`, in seconds, after which the token is removed from the cache.

`augmentSecurityContext`

Specifies a script that is executed only after a successful authentication request. The script helps to populate the expected security context. For more information, see "The `augmentSecurityContext` Trigger" in the *Scripting Guide*.

`subjectMapping`

An array of mappings that let you map AM realms to Identity Cloud managed object types. For example:

```
"subjectMapping" : [
    {
        "resourceTypeMapping" : {
            "usr" : "managed/realm-name_user"
        },
        "propertyMapping" : {
            "sub" : "_id"
        },
        "userRoles" : "authzRoles/*",
        "additionalUserFields" : [
            "adminOfOrg",
            "ownerOfOrg"
        ],
        "defaultRoles" : [
            "internal/role/openidm-authorized"
        ]
    }
],
```

Each `subjectMapping` includes the following properties:

- Either a `resourceTypeMapping` or a `queryOnResource` property:

  - `resourceTypeMapping`: Maps the identity type of a subject claim in AM to a resource collection in Identity Cloud. In the access token, the subject claim is a compound identity that consists of the claim `type` and subject name, separated by a `!`.

    To use a `resourceTypeMapping`, unique Oauth2 subject claims must be enabled in AM. (From AM 7.1, these are enabled by default.) For more information about subject claims, see *About the Subject and the Subname Claims* in the section on /oauth2/userinfo.

  - `queryOnResource`: The Identity Cloud resource to check for the authenticating user; for example, `managed/realm-name_user`.

  Both the `resourceTypeMapping` and the `queryOnResource` properties support a dynamic handlebars template that lets a single subject mapping match multiple realms, if the managed objects are named prescriptively, and based on the realm name. For example:

  ```
  "resourceTypeMapping" : {
      "usr" : "managed/{{substring realm 1}}"
  }
  ```

  This configuration lets an access token with the realm `employee` map to an Identity Cloud `managed/employee`, and an access token with the realm `contractor` map to an Identity Cloud `managed/contractor`. The configuration is useful if your AM and Identity Cloud deployments use a consistent realm and managed object naming.

- `realm`: The AM realm to which this subject mapping applies. A value of `/` specifies the top-level realm. If this property is absent, the mapping can apply to any realm, which is useful if the `resourceTypeMapping` or `queryOnResource` uses a dynamic handlebars template.

  You cannot have more than one mapping for the same realm, and you cannot have more than one mapping that has no `realm` in the configuration.

- `propertyMapping`: Maps fields in the AM access token to properties in Identity Cloud. This mapping is used to construct the query filter that locates the authenticating user. The default configuration maps the subject (`sub`) in the access token to the `_id` in Identity Cloud.

- `userRoles`: Determines the field to consult for locating the authorization roles; usually `authzRoles`, unless you have changed how user roles are stored. This field must be a relationship field. Identity Cloud uses the `_refId` from the array elements to populate the user roles in the security context.

- `additionalUserFields`: Determines the field to consult for locating the authorization roles; usually `authzRoles`, unless you have changed how user roles are stored. This field must be a relationship field. Identity Cloud uses the `_refId` from the array elements to populate the user roles in the security context.

- `defaultRoles`: The default roles that should be applied to a user who authenticates using the `rsFilter`.

Although you can configure an array of subject mappings, only one mapping is selected and used during the authentication process. If there is a `realm` attribute in the access token, that realm is used to select an appropriate mapping. If no mapping is defined for the access token's realm, or if the realm is not provided in the access token, the authentication uses a mapping that does not define a `realm`.

Note

If you have a remote connector server that is authenticating against AM, you must add a subject mapping specifically for the connector server. For example:

```
{
    "subject" : "RCS-OAuth-clientId",
    "localUser" : "internal/user/idm-provisioning"
}
```

The `subject` must reflect the OAuth2 client in AM that has been set up for the remote connector server. The `localUser` can be any existing user. Do not assign that user any roles to ensure that the connector server bearer token cannot be used for any other purpose.

`staticUserMapping`

Maps AM users to a matching Identity Cloud user. Can contain multiple user mappings, each with three properties: `subject`, `localUser`, and `roles`.

- `subject` of the access token (the AM user). If you have specified a `resourceTypeMapping`, the static user mapping includes the full new subject string to match service accounts or static subject mappings, for example:

  ```
  "subject" : "(usr!amadmin)"
  ```

- `localUser` is the Identity Cloud user you want to associate with the AM user identified in `subject`. For example, if `subject` is set to `(usr!amadmin)`, you might set the corresponding `localUser` to `internal/user/openidm-admin`.

- `roles` the default Identity Cloud roles that this mapped user will have after they authenticate.

Note

The `idm-provisioning` subject is a service account used by AM to provision users in Identity Cloud. You *must* include this subject in your `staticUserMapping`. For example:

```
{
    "subject": "(age!idm-provisioning)",
    "localUser": "internal/user/idm-provisioning",
    "roles" : [
        "internal/role/platform-provisioning"
    ]
}
```

anonymousUserMapping

The default user that will be used when no access token is included in the request. Contains two properties: `localUser` and `userRoles`.

- `localUser`: the Identity Cloud user resource referenced when no specific user is identified. For example, `internal/user/anonymous`.
- `roles`: the default roles that the anonymous user will have, usually `internal/role/openidm-reg`.

Test Authentication Through AM

1. Obtain a bearer token from AM. For example:

```
curl \
--header "X-OpenAM-Username: amAdmin" \
--header "X-OpenAM-Password: password" \
--header "Accept-API-Version: resource=2.0, protocol=1.0" \
--request POST \
--data "grant_type=client_credentials" \
--data "client_id=idm-provisioning" \
--data "client_secret=openidm" \
--data "scope=fr:idm:*" \
"http://am.example.com:8080/am/oauth2/realms/root/access_token"
{
  "access_token": "z4uKDWiv4wnxKY7OjeG04PujG8E",
  "scope": "fr:idm:*",
  "token_type": "Bearer",
  "expires_in": 3599
}
```

2. Authenticate to Identity Cloud using that bearer token:

```
curl \
--request GET \
--header "Content-Type: application/json" \
--header "Authorization: Bearer z4uKDWiv4wnxKY7OjeG04PujG8E" \
'https://tenant-name.forgeblocks.com/openidm/info/login'
{
  "_id": "login",
  "authenticationId": "idm-provisioning",
  "authorization": {
    "id": "idm-provisioning",
    "roles": [
      "internal/role/platform-provisioning"
    ],
    "component": "internal/user"
  }
}
```

See the Platform Setup Guide for complete instructions on setting up Identity Cloud to use AM bearer tokens for authentication.

Authentication and Roles

When a user authenticates, they are given a set of default *internal roles*. These roles determine how much access the user has to Identity Cloud. Identity Cloud includes a number of default internal roles, on the `openidm/internal/roles` endpoint. You can configure additional internal roles to customize how you restrict access to the server.

The following internal roles are defined by default (in `conf/repo.init.json`):

openidm-admin

Identity Cloud administrator role, excluded from the reauthorization required policy definition by default.

openidm-authorized

Default role for any user who authenticates with a username and password.

openidm-cert

Default role for any user who authenticates with mutual SSL authentication.

This role applies only to mutual authentication. The shared secret (certificate) must be adequately protected. The `openidm-cert` role is excluded from the reauthorization required policy definition by default.

openidm-reg

Assigned to users who access Identity Cloud with the default anonymous account.

The `openidm-reg` role is excluded from the reauthorization required policy definition by default.

openidm-tasks-manager

Role for users who can be assigned to workflow tasks.

platform-provisioning

Role for platform provisioning access. If you are not planning to run AM and Identity Cloud together as a platform, you can safely remove this role.

When a user authenticates, Identity Cloud calculates that user's roles as follows:

- Each authentication module includes a `defaultUserRoles` property. Depending on how the user authenticates, Identity Cloud assigns the roles listed in that module's `defaultUserRoles` property to the user on authentication. The `defaultUserRoles` property is specified as an array. For most authentication modules, the user obtains the `openidm-authorized` role on authentication. For example:

```
{
    "name" : "MANAGED_USER",
    "properties" : {
        ...
        "defaultUserRoles" : [
            "internal/role/openidm-authorized"
        ]
    },
    ...
}
```

- The `userRoles` property in an authentication module maps to an attribute (or list of attributes) in a user entry that contains that user's authorization roles. This attribute is usually `authzRoles`, unless you have changed how user roles are stored.

  Any internal roles that are conditionally applied are also calculated and included in the user's `authzRoles` property at this point.

- If the authentication module includes a `groupRoleMapping`, `groupMembership`, or `groupComparison` property, Identity Cloud can assign additional roles to the user, depending on the user's group membership on an *external* system. For more information, see "Use Groups to Control Access to Identity Cloud" in the *Object Modeling Guide*.

Note

The roles calculated in sequence are cumulative. Roles with temporal restrictions are not included in that list if the current time is outside of the time assigned to the role.

Dynamic Role Calculation

By default, Identity Cloud calculates a user's roles only on authentication. You can configure Identity Cloud to recalculate a user's roles dynamically, with each request, instead of only when the user reauthenticates. To enable this feature, set `enableDynamicRoles` to `true` in the `JWT_SESSION` session module in `authentication.json`:

To enable dynamic role calculation through the Admin UI, select Configure > Authentication > Session > Enable Dynamic Roles.

Roles, Authentication, and the Security Context

The Security Context (`context.security`), consists of a principal (defined by the `authenticationId` property) and an access control element (defined by the `authorization` property).

If authentication is successful, the authentication framework sets the principal. Identity Cloud stores that principal as the `authenticationId`.

The `authorization` property includes an `id`, an array of `roles`, and a `component`, that specifies the resource against which authorization is validated.

Chapter 2

Authorization and Access Control

Identity Cloud provides role-based authorization that restricts direct HTTP access to REST interface URLs. This access control applies to direct HTTP calls only. Access for internal calls (for example, calls from scripts) is not affected by this mechanism.

Authorization and Roles

When a user authenticates, they are given a set of default *roles*, as described in "Authentication and Roles". The authorization configuration grants access rights to users, based on these roles acquired during authentication.

You can use internal and managed roles to restrict access, with the following caveats:

- Internal roles are not meant to be provisioned or synchronized with external systems.
- Internal roles cannot be given assignments.
- Event scripts (such as `onCreate`) cannot be attached to internal roles.
- The internal role schema is not configurable.

Authorization roles are referenced in a user's `authzRoles` property by default, and are assigned when the user authenticates.

By default, managed users are assigned the `openidm-authorized` role when they authenticate. The following request shows the authorization roles for user psmith when that user logs in to the server:

```
curl \
--header "X-OpenIDM-Username: psmith" \
--header "X-OpenIDM-Password: Passw0rd" \
--header "Accept-API-Version: resource=1.0" \
--cacert ca-cert.pem \
--request GET \
"https://localhost:8443/openidm/managed/realm-name_user/openidm/info/login"
{
  "_id": "login",
  "authenticationId": "psmith",
  "authorization": {
    "userRolesProperty": "authzRoles",
    "component": "managed/realm-name_user",
    "authLogin": false,
    "authenticationIdProperty": "username",
    "roles": [
      "internal/role/openidm-authorized"
    ],
    "ipAddress": "0:0:0:0:0:0:0:1",
    "authenticationId": "psmith",
    "protectedAttributeList": [
      "password"
    ],
    "id": "psmith",
    "moduleId": "MANAGED_USER",
    "queryId": "credential-query"
  }
}
```

The authorization implementation is configured in two files:

- `openidm/bin/defaults/script/router-authz.js`
- *`project-dir`*`/conf/access.json`

Identity Cloud calls the `router-authz.js` script for each request, through an `onRequest` hook defined in the `router.json` file. `router-authz.js` references your project's access configuration (`access.json`) to determine the allowed HTTP requests. If access is denied, according to the configuration defined in `access.json`, the `router-authz.js` script throws an exception, and Identity Cloud denies the request.

`router.json` also defines an `onResponse` script, `relationshipFilter`. This provides additional filtering to ensure that the user has the appropriate access to see the data of the related object. You can change this behavior by extending or updating `/bin/defaults/script/relationshipFilter.js`, or by removing the `onResponse` script if you don't want additional filtering on relationships. For more information about relationships, see "*Relationships Between Objects*" in the *Object Modeling Guide*.)

Note

You can configure delegated administration to grant access that bypasses this access control.

The Router Authorization Script

The router authorization script (`router-authz.js` contains a number of functions that enforce access rules. For example, the following function controls whether users with a certain role can start a specified process:

```
function isAllowedToStartProcess() {
    var processDefinitionId = request.content._processDefinitionId;
    var key = request.content._key;
    return isProcessOnUsersList(function (process) {
        return (process._id === processDefinitionId) || (process.key === key);
    });
}
```

You can extend the default authorization mechanism by defining additional functions in `router-authz.js` and by creating new access control rules in `access.json`.

Important

Some authorization-related functions in `router-authz.js` should *not* be altered, because they affect the security of the server. Such functions are indicated in the comments in that file.

Configure Access Control in `access.json`

The `access.json` configuration includes a set of rules that govern access to specific endpoints. These rules are tested in the order in which they appear in the file. You can define more than one rule for the same endpoint. If one rule passes, the request is allowed. If all the rules fail, the request is denied.

The following rule (from a default `access.json` file) shows the access configuration structure:

```
{
    "pattern"   : "system/*",
    "roles"     : "internal/role/openidm-admin",
    "methods"   : "action",
    "actions"   : "test,testConfig,createconfiguration,liveSync,authenticate"
}
```

This rule specifies that users with the `openidm-admin` role can perform the listed actions on all `system` endpoints.

The parameters in each access rule are as follows:

`pattern`

The REST endpoint for which access is being controlled. `"*"` specifies access to all endpoints in that path. For example, `"managed/realm-name_user/*"` specifies access to all managed user objects.

`roles`

A comma-separated list of the roles to which this access configuration applies.

The `roles` referenced here align with the object's security context (`security.authorization.roles`). The `authzRoles` relationship property of a managed user produces this security context value during authentication.

`methods`

A comma-separated list of the methods that can be performed with this access. Methods can include `create, read, update, delete, patch, action, query`. A value of `"*"` indicates that all methods are allowed. A value of `""` indicates that no methods are allowed.

`actions`

A comma-separated list of the allowed actions. The possible actions depend on the resource (URL) that is being exposed. Note that the `actions` in the default `access.json` file do not list all the supported actions in the *Scripting Guide* on each resource.

A value of `"*"` indicates that all actions exposed for that resource are allowed. A value of `""` indicates that no actions are allowed.

`customAuthz`

An optional parameter that lets you define a custom function for additional authorization checks. Custom functions are defined in `router-authz.js`.

`excludePatterns`

An optional parameter that lets you specify endpoints to which access should not be granted.

Change the Access Configuration Over REST

You can manage the access configuration at the endpoint `openidm/config/access`. To change an access rule, first get the current access configuration, amend it to change the access rule, then submit the updated configuration in a PUT request. This example restricts access to the `info` endpoint to users who have authenticated:

+ *Get the Current Access Configuration*

```
curl \
--header "Authorization: Bearer *token*" \
--header "Accept-API-Version: resource=1.0"  \
--request GET \
"https://tenant-name.forgeblocks.com/openidm/config/access"
{
  "_id": "access",
  "configs": [
    {
      "pattern": "info/*",
      "roles": "*",
```

```
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "authentication",
      "roles": "*",
      "methods": "read,action",
      "actions": "login,logout"
    },
    {
      "pattern": "identityProviders",
      "roles": "*",
      "methods": "action",
      "actions": "getAuthRedirect,handlePostAuth,getLogoutUrl"
    },
    {
      "pattern": "identityProviders",
      "roles": "internal/role/openidm-authorized",
      "methods": "action",
      "actions": "normalizeProfile"
    },
    {
      "pattern": "identityProviders",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "config/ui/themeconfig",
      "roles": "*",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "info/uiconfig",
      "roles": "*",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "config/selfservice/kbaConfig",
      "roles": "*",
      "methods": "read",
      "actions": "*",
      "customAuthz": "checkIfAnyFeatureEnabled(['registration', 'passwordReset'])"
    },
    {
      "pattern": "config/ui/dashboard",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "info/features",
      "roles": "*",
      "methods": "query",
      "actions": "*"
    },
    {
```

```
      "pattern": "privilege",
      "roles": "*",
      "methods": "action",
      "actions": "listPrivileges"
    },
    {
      "pattern": "privilege/*",
      "roles": "*",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "selfservice/registration",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements",
      "customAuthz": "checkIfAnyFeatureEnabled('registration')"
    },
    {
      "pattern": "selfservice/socialUserClaim",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements",
      "customAuthz": "checkIfAnyFeatureEnabled('registration')"
    },
    {
      "pattern": "selfservice/reset",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements",
      "customAuthz": "checkIfAnyFeatureEnabled('passwordReset')"
    },
    {
      "pattern": "selfservice/username",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements",
      "customAuthz": "checkIfAnyFeatureEnabled('retrieveUsername')"
    },
    {
      "pattern": "selfservice/profile",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements"
    },
    {
      "pattern": "selfservice/termsAndConditions",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements"
    },
    {
      "pattern": "selfservice/kbaUpdate",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements"
    },
    {
      "pattern": "policy/*",
```

```
      "roles": "*",
      "methods": "action",
      "actions": "validateObject",
      "customAuthz": "context.current.name === 'selfservice'"
    },
    {
      "pattern": "policy/selfservice/registration",
      "roles": "*",
      "methods": "action,read",
      "actions": "validateObject",
      "customAuthz": "checkIfAnyFeatureEnabled('registration')"
    },
    {
      "pattern": "policy/selfservice/reset",
      "roles": "*",
      "methods": "action,read",
      "actions": "validateObject",
      "customAuthz": "checkIfAnyFeatureEnabled('passwordReset')"
    },
    {
      "pattern": "selfservice/kba",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*",
      "customAuthz": "checkIfAnyFeatureEnabled('kba')"
    },
    {
      "pattern": "managed/realm-name_user",
      "roles": "internal/role/openidm-reg",
      "methods": "create",
      "actions": "*",
      "customAuthz": "checkIfAnyFeatureEnabled('registration') && isSelfServiceRequest() &&
onlyEditableManagedObjectProperties('user', [])"
    },
    {
      "pattern": "managed/realm-name_user",
      "roles": "*",
      "methods": "query",
      "actions": "*",
      "customAuthz": "checkIfAnyFeatureEnabled(['registration', 'retrieveUsername', 'passwordReset'])
&& isSelfServiceRequest()"
    },
    {
      "pattern": "managed/realm-name_user/*",
      "roles": "*",
      "methods": "read",
      "actions": "*",
      "customAuthz": "checkIfAnyFeatureEnabled(['retrieveUsername', 'passwordReset']) &&
isSelfServiceRequest()"
    },
    {
      "pattern": "managed/realm-name_user/*",
      "roles": "*",
      "methods": "patch,action",
      "actions": "patch",
      "customAuthz": "(checkIfAnyFeatureEnabled(['registration', 'passwordReset'])
|| checkIfProgressiveProfileIsEnabled()) && isSelfServiceRequest() &&
onlyEditableManagedObjectProperties('user', [])"
    },
```

```
    {
      "pattern": "external/email",
      "roles": "*",
      "methods": "action",
      "actions": "send",
      "customAuthz": "checkIfAnyFeatureEnabled(['registration', 'retrieveUsername', 'passwordReset'])
&& isSelfServiceRequest()"
    },
    {
      "pattern": "schema/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "consent",
      "roles": "internal/role/openidm-authorized",
      "methods": "action,query",
      "actions": "*"
    },
    {
      "pattern": "*",
      "roles": "internal/role/openidm-admin",
      "methods": "*",
      "actions": "*",
      "excludePatterns": "repo,repo/*"
    },
    {
      "pattern": "system/*",
      "roles": "internal/role/openidm-admin",
      "methods": "create,read,update,delete,patch,query",
      "actions": ""
    },
    {
      "pattern": "system/*",
      "roles": "internal/role/openidm-admin",
      "methods": "script",
      "actions": "*"
    },
    {
      "pattern": "system/*",
      "roles": "internal/role/openidm-admin",
      "methods": "action",
      "actions": "test,testConfig,createconfiguration,liveSync,authenticate"
    },
    {
      "pattern": "repo",
      "roles": "internal/role/openidm-admin",
      "methods": "*",
      "actions": "*",
      "customAuthz": "disallowCommandAction()"
    },
    {
      "pattern": "repo/*",
      "roles": "internal/role/openidm-admin",
      "methods": "*",
      "actions": "*",
      "customAuthz": "disallowCommandAction()"
    },
```

```
    {
      "pattern": "repo/link",
      "roles": "internal/role/openidm-admin",
      "methods": "action",
      "actions": "command",
      "customAuthz": "request.additionalParameters.commandId === 'delete-mapping-links'"
    },
    {
      "pattern": "managed/*",
      "roles": "internal/role/platform-provisioning",
      "methods": "create,read,query,patch"
    },
    {
      "pattern": "internal/role/*",
      "roles": "internal/role/platform-provisioning",
      "methods": "read,query"
    },
    {
      "pattern": "profile/*",
      "roles": "internal/role/platform-provisioning",
      "methods": "create,read,action,update",
      "actions": "*"
    },
    {
      "pattern": "policy/*",
      "roles": "internal/role/platform-provisioning",
      "methods": "read,action",
      "actions": "*"
    },
    {
      "pattern": "schema/*",
      "roles": "internal/role/platform-provisioning",
      "methods": "read"
    },
    {
      "pattern": "consent",
      "roles": "internal/role/platform-provisioning",
      "methods": "action,query",
      "actions": "*"
    },
    {
      "pattern": "selfservice/kba",
      "roles": "internal/role/platform-provisioning",
      "methods": "read"
    },
    {
      "pattern": "selfservice/terms",
      "roles": "internal/role/platform-provisioning",
      "methods": "read"
    },
    {
      "pattern": "identityProviders",
      "roles": "internal/role/platform-provisioning",
      "methods": "read"
    },
    {
      "pattern": "external/email",
      "roles": "internal/role/platform-provisioning",
      "methods": "action",
```

```
      "actions": "sendTemplate"
    },
    {
      "pattern": "policy/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read,action",
      "actions": "*"
    },
    {
      "pattern": "config/ui/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "authentication",
      "roles": "internal/role/openidm-authorized",
      "methods": "action",
      "actions": "reauthenticate"
    },
    {
      "pattern": "*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read,action,delete",
      "actions": "bind,unbind",
      "customAuthz": "ownDataOnly()"
    },
    {
      "pattern": "*",
      "roles": "internal/role/openidm-authorized",
      "methods": "update,patch,action",
      "actions": "patch",
      "customAuthz": "ownDataOnly() && onlyEditableManagedObjectProperties('user', []) &&
reauthIfProtectedAttributeChange()"
    },
    {
      "pattern": "selfservice/user/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "patch,action",
      "actions": "patch",
      "customAuthz": "(request.resourcePath === 'selfservice/user/' +
context.security.authorization.id) && onlyEditableManagedObjectProperties('user', [])"
    },
    {
      "pattern": "endpoint/getprocessesforuser",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "endpoint/gettasksview",
      "roles": "internal/role/openidm-authorized",
      "methods": "query",
      "actions": "*"
    },
    {
      "pattern": "workflow/taskinstance/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "action",
```

```
      "actions": "complete",
      "customAuthz": "isMyTask()"
    },
    {
      "pattern": "workflow/taskinstance/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read,update",
      "actions": "*",
      "customAuthz": "canUpdateTask()"
    },
    {
      "pattern": "workflow/processinstance",
      "roles": "internal/role/openidm-authorized",
      "methods": "create",
      "actions": "*",
      "customAuthz": "isAllowedToStartProcess()"
    },
    {
      "pattern": "workflow/processdefinition/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "*",
      "actions": "read",
      "customAuthz": "isOneOfMyWorkflows()"
    },
    {
      "pattern": "managed/realm-name_user",
      "roles": "internal/role/openidm-cert",
      "methods": "patch,action",
      "actions": "patch",
      "customAuthz": "isQueryOneOf({'managed/realm-name_user': ['for-userName']}) &&
restrictPatchToFields(['password'])"
    },
    {
      "pattern": "internal/usermeta/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*",
      "customAuthz": "ownRelationship()"
    },
    {
      "pattern": "internal/notification/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read,delete",
      "actions": "*",
      "customAuthz": "ownRelationship()"
    },
    {
      "pattern": "managed/realm-name_user/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read,query",
      "actions": "*",
      "customAuthz": "ownRelationshipCollection(['idps','_meta','_notifications'])"
    },
    {
      "pattern": "notification",
      "roles": "internal/role/openidm-authorized",
      "methods": "action",
      "actions": "deleteNotificationsForTarget",
```

```
      "customAuthz": "request.additionalParameters.target ===
 (context.security.authorization.component + '/' + context.security.authorization.id)"
    },
    {
      "pattern": "managed/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*",
      "customAuthz": "ownIDP()"
    }
  ]
}
```

+ *Replace the Access Configuration*

```
curl \
--header "Authorization: Bearer *token*" \
--header "Content-type: application/json" \
--header "Accept-API-Version: resource=1.0"  \
--request PUT \
--data '{
  "_id": "access",
  "configs": [
    {
      "pattern": "info/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "authentication",
      "roles": "*",
      "methods": "read,action",
      "actions": "login,logout"
    },
    {
      "pattern": "identityProviders",
      "roles": "*",
      "methods": "action",
      "actions": "getAuthRedirect,handlePostAuth,getLogoutUrl"
    },
    {
      "pattern": "identityProviders",
      "roles": "internal/role/openidm-authorized",
      "methods": "action",
      "actions": "normalizeProfile"
    },
    {
      "pattern": "identityProviders",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "config/ui/themeconfig",
      "roles": "*",
      "methods": "read",
```

```
      "actions": "*"
    },
    {
      "pattern": "info/uiconfig",
      "roles": "*",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "config/selfservice/kbaConfig",
      "roles": "*",
      "methods": "read",
      "actions": "*",
      "customAuthz": "checkIfAnyFeatureEnabled(['registration', 'passwordReset'])"
    },
    {
      "pattern": "config/ui/dashboard",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "info/features",
      "roles": "*",
      "methods": "query",
      "actions": "*"
    },
    {
      "pattern": "privilege",
      "roles": "*",
      "methods": "action",
      "actions": "listPrivileges"
    },
    {
      "pattern": "privilege/*",
      "roles": "*",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "selfservice/registration",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements",
      "customAuthz": "checkIfAnyFeatureEnabled('registration')"
    },
    {
      "pattern": "selfservice/socialUserClaim",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements",
      "customAuthz": "checkIfAnyFeatureEnabled('registration')"
    },
    {
      "pattern": "selfservice/reset",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements",
      "customAuthz": "checkIfAnyFeatureEnabled('passwordReset')"
```

```
    },
    {
      "pattern": "selfservice/username",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements",
      "customAuthz": "checkIfAnyFeatureEnabled('retrieveUsername')"
    },
    {
      "pattern": "selfservice/profile",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements"
    },
    {
      "pattern": "selfservice/termsAndConditions",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements"
    },
    {
      "pattern": "selfservice/kbaUpdate",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements"
    },
    {
      "pattern": "policy/*",
      "roles": "*",
      "methods": "action",
      "actions": "validateObject",
      "customAuthz": "context.current.name === 'selfservice'"
    },
    {
      "pattern": "policy/selfservice/registration",
      "roles": "*",
      "methods": "action,read",
      "actions": "validateObject",
      "customAuthz": "checkIfAnyFeatureEnabled('registration')"
    },
    {
      "pattern": "policy/selfservice/reset",
      "roles": "*",
      "methods": "action,read",
      "actions": "validateObject",
      "customAuthz": "checkIfAnyFeatureEnabled('passwordReset')"
    },
    {
      "pattern": "selfservice/kba",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*",
      "customAuthz": "checkIfAnyFeatureEnabled('kba')"
    },
    {
      "pattern": "managed/realm-name_user",
      "roles": "internal/role/openidm-reg",
      "methods": "create",
      "actions": "*",
```

```
      "customAuthz": "checkIfAnyFeatureEnabled('registration') && isSelfServiceRequest() &&
onlyEditableManagedObjectProperties('user', [])"
    },
    {
      "pattern": "managed/realm-name_user",
      "roles": "*",
      "methods": "query",
      "actions": "*",
      "customAuthz": "checkIfAnyFeatureEnabled(['registration', 'retrieveUsername', 'passwordReset'])
&& isSelfServiceRequest()"
    },
    {
      "pattern": "managed/realm-name_user/*",
      "roles": "*",
      "methods": "read",
      "actions": "*",
      "customAuthz": "checkIfAnyFeatureEnabled(['retrieveUsername', 'passwordReset']) &&
isSelfServiceRequest()"
    },
    {
      "pattern": "managed/realm-name_user/*",
      "roles": "*",
      "methods": "patch,action",
      "actions": "patch",
      "customAuthz": "(checkIfAnyFeatureEnabled(['registration', 'passwordReset'])
|| checkIfProgressiveProfileIsEnabled()) && isSelfServiceRequest() &&
onlyEditableManagedObjectProperties('user', [])"
    },
    {
      "pattern": "external/email",
      "roles": "*",
      "methods": "action",
      "actions": "send",
      "customAuthz": "checkIfAnyFeatureEnabled(['registration', 'retrieveUsername', 'passwordReset'])
&& isSelfServiceRequest()"
    },
    {
      "pattern": "schema/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "consent",
      "roles": "internal/role/openidm-authorized",
      "methods": "action,query",
      "actions": "*"
    },
    {
      "pattern": "*",
      "roles": "internal/role/openidm-admin",
      "methods": "*",
      "actions": "*",
      "excludePatterns": "repo,repo/*"
    },
    {
      "pattern": "system/*",
      "roles": "internal/role/openidm-admin",
      "methods": "create,read,update,delete,patch,query",
```

```
      "actions": ""
    },
    {
      "pattern": "system/*",
      "roles": "internal/role/openidm-admin",
      "methods": "script",
      "actions": "*"
    },
    {
      "pattern": "system/*",
      "roles": "internal/role/openidm-admin",
      "methods": "action",
      "actions": "test,testConfig,createconfiguration,liveSync,authenticate"
    },
    {
      "pattern": "repo",
      "roles": "internal/role/openidm-admin",
      "methods": "*",
      "actions": "*",
      "customAuthz": "disallowCommandAction()"
    },
    {
      "pattern": "repo/*",
      "roles": "internal/role/openidm-admin",
      "methods": "*",
      "actions": "*",
      "customAuthz": "disallowCommandAction()"
    },
    {
      "pattern": "repo/link",
      "roles": "internal/role/openidm-admin",
      "methods": "action",
      "actions": "command",
      "customAuthz": "request.additionalParameters.commandId === 'delete-mapping-links'"
    },
    {
      "pattern": "managed/*",
      "roles": "internal/role/platform-provisioning",
      "methods": "create,read,query,patch"
    },
    {
      "pattern": "internal/role/*",
      "roles": "internal/role/platform-provisioning",
      "methods": "read,query"
    },
    {
      "pattern": "profile/*",
      "roles": "internal/role/platform-provisioning",
      "methods": "create,read,action,update",
      "actions": "*"
    },
    {
      "pattern": "policy/*",
      "roles": "internal/role/platform-provisioning",
      "methods": "read,action",
      "actions": "*"
    },
    {
      "pattern": "schema/*",
```

```
      "roles": "internal/role/platform-provisioning",
      "methods": "read"
    },
    {
      "pattern": "consent",
      "roles": "internal/role/platform-provisioning",
      "methods": "action,query",
      "actions": "*"
    },
    {
      "pattern": "selfservice/kba",
      "roles": "internal/role/platform-provisioning",
      "methods": "read"
    },
    {
      "pattern": "selfservice/terms",
      "roles": "internal/role/platform-provisioning",
      "methods": "read"
    },
    {
      "pattern": "identityProviders",
      "roles": "internal/role/platform-provisioning",
      "methods": "read"
    },
    {
      "pattern": "external/email",
      "roles": "internal/role/platform-provisioning",
      "methods": "action",
      "actions": "sendTemplate"
    },
    {
      "pattern": "policy/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read,action",
      "actions": "*"
    },
    {
      "pattern": "config/ui/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "authentication",
      "roles": "internal/role/openidm-authorized",
      "methods": "action",
      "actions": "reauthenticate"
    },
    {
      "pattern": "*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read,action,delete",
      "actions": "bind,unbind",
      "customAuthz": "ownDataOnly()"
    },
    {
      "pattern": "*",
      "roles": "internal/role/openidm-authorized",
      "methods": "update,patch,action",
```

```
      "actions": "patch",
      "customAuthz": "ownDataOnly() && onlyEditableManagedObjectProperties('user', []) &&
reauthIfProtectedAttributeChange()"
    },
    {
      "pattern": "selfservice/user/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "patch,action",
      "actions": "patch",
      "customAuthz": "(request.resourcePath === 'selfservice/user/' +
context.security.authorization.id) && onlyEditableManagedObjectProperties('user', [])"
    },
    {
      "pattern": "endpoint/getprocessesforuser",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "endpoint/gettasksview",
      "roles": "internal/role/openidm-authorized",
      "methods": "query",
      "actions": "*"
    },
    {
      "pattern": "workflow/taskinstance/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "action",
      "actions": "complete",
      "customAuthz": "isMyTask()"
    },
    {
      "pattern": "workflow/taskinstance/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read,update",
      "actions": "*",
      "customAuthz": "canUpdateTask()"
    },
    {
      "pattern": "workflow/processinstance",
      "roles": "internal/role/openidm-authorized",
      "methods": "create",
      "actions": "*",
      "customAuthz": "isAllowedToStartProcess()"
    },
    {
      "pattern": "workflow/processdefinition/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "*",
      "actions": "read",
      "customAuthz": "isOneOfMyWorkflows()"
    },
    {
      "pattern": "managed/realm-name_user",
      "roles": "internal/role/openidm-cert",
      "methods": "patch,action",
      "actions": "patch",
      "customAuthz": "isQueryOneOf({'managed/realm-name_user': ['for-userName']}) &&
restrictPatchToFields(['password'])"
```

```
    },
    {
      "pattern": "internal/usermeta/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*",
      "customAuthz": "ownRelationship()"
    },
    {
      "pattern": "internal/notification/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read,delete",
      "actions": "*",
      "customAuthz": "ownRelationship()"
    },
    {
      "pattern": "managed/realm-name_user/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read,query",
      "actions": "*",
      "customAuthz": "ownRelationshipCollection(['idps','_meta','_notifications'])"
    },
    {
      "pattern": "notification",
      "roles": "internal/role/openidm-authorized",
      "methods": "action",
      "actions": "deleteNotificationsForTarget",
      "customAuthz": "request.additionalParameters.target ===
 (context.security.authorization.component + '/' + context.security.authorization.id)"
    },
    {
      "pattern": "managed/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*",
      "customAuthz": "ownIDP()"
    }
  ]
}' \
"https://tenant-name.forgeblocks.com/openidm/config/access"
{
  "_id": "access",
  "configs": [
    {
      "pattern": "info/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "authentication",
      "roles": "*",
      "methods": "read,action",
      "actions": "login,logout"
    },
    {
      "pattern": "identityProviders",
      "roles": "*",
      "methods": "action",
```

```
      "actions": "getAuthRedirect,handlePostAuth,getLogoutUrl"
    },
    {
      "pattern": "identityProviders",
      "roles": "internal/role/openidm-authorized",
      "methods": "action",
      "actions": "normalizeProfile"
    },
    {
      "pattern": "identityProviders",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "config/ui/themeconfig",
      "roles": "*",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "info/uiconfig",
      "roles": "*",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "config/selfservice/kbaConfig",
      "roles": "*",
      "methods": "read",
      "actions": "*",
      "customAuthz": "checkIfAnyFeatureEnabled(['registration', 'passwordReset'])"
    },
    {
      "pattern": "config/ui/dashboard",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "info/features",
      "roles": "*",
      "methods": "query",
      "actions": "*"
    },
    {
      "pattern": "privilege",
      "roles": "*",
      "methods": "action",
      "actions": "listPrivileges"
    },
    {
      "pattern": "privilege/*",
      "roles": "*",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "selfservice/registration",
```

```
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements",
      "customAuthz": "checkIfAnyFeatureEnabled('registration')"
    },
    {
      "pattern": "selfservice/socialUserClaim",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements",
      "customAuthz": "checkIfAnyFeatureEnabled('registration')"
    },
    {
      "pattern": "selfservice/reset",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements",
      "customAuthz": "checkIfAnyFeatureEnabled('passwordReset')"
    },
    {
      "pattern": "selfservice/username",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements",
      "customAuthz": "checkIfAnyFeatureEnabled('retrieveUsername')"
    },
    {
      "pattern": "selfservice/profile",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements"
    },
    {
      "pattern": "selfservice/termsAndConditions",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements"
    },
    {
      "pattern": "selfservice/kbaUpdate",
      "roles": "*",
      "methods": "read,action",
      "actions": "submitRequirements"
    },
    {
      "pattern": "policy/*",
      "roles": "*",
      "methods": "action",
      "actions": "validateObject",
      "customAuthz": "context.current.name === 'selfservice'"
    },
    {
      "pattern": "policy/selfservice/registration",
      "roles": "*",
      "methods": "action,read",
      "actions": "validateObject",
      "customAuthz": "checkIfAnyFeatureEnabled('registration')"
    },
    {
```

```
      "pattern": "policy/selfservice/reset",
      "roles": "*",
      "methods": "action,read",
      "actions": "validateObject",
      "customAuthz": "checkIfAnyFeatureEnabled('passwordReset')"
    },
    {
      "pattern": "selfservice/kba",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*",
      "customAuthz": "checkIfAnyFeatureEnabled('kba')"
    },
    {
      "pattern": "managed/realm-name_user",
      "roles": "internal/role/openidm-reg",
      "methods": "create",
      "actions": "*",
      "customAuthz": "checkIfAnyFeatureEnabled('registration') && isSelfServiceRequest() &&
onlyEditableManagedObjectProperties('user', [])"
    },
    {
      "pattern": "managed/realm-name_user",
      "roles": "*",
      "methods": "query",
      "actions": "*",
      "customAuthz": "checkIfAnyFeatureEnabled(['registration', 'retrieveUsername', 'passwordReset'])
&& isSelfServiceRequest()"
    },
    {
      "pattern": "managed/realm-name_user/*",
      "roles": "*",
      "methods": "read",
      "actions": "*",
      "customAuthz": "checkIfAnyFeatureEnabled(['retrieveUsername', 'passwordReset']) &&
isSelfServiceRequest()"
    },
    {
      "pattern": "managed/realm-name_user/*",
      "roles": "*",
      "methods": "patch,action",
      "actions": "patch",
      "customAuthz": "(checkIfAnyFeatureEnabled(['registration', 'passwordReset'])
|| checkIfProgressiveProfileIsEnabled()) && isSelfServiceRequest() &&
onlyEditableManagedObjectProperties('user', [])"
    },
    {
      "pattern": "external/email",
      "roles": "*",
      "methods": "action",
      "actions": "send",
      "customAuthz": "checkIfAnyFeatureEnabled(['registration', 'retrieveUsername', 'passwordReset'])
&& isSelfServiceRequest()"
    },
    {
      "pattern": "schema/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*"
```

```
    },
    {
      "pattern": "consent",
      "roles": "internal/role/openidm-authorized",
      "methods": "action,query",
      "actions": "*"
    },
    {
      "pattern": "*",
      "roles": "internal/role/openidm-admin",
      "methods": "*",
      "actions": "*",
      "excludePatterns": "repo,repo/*"
    },
    {
      "pattern": "system/*",
      "roles": "internal/role/openidm-admin",
      "methods": "create,read,update,delete,patch,query",
      "actions": ""
    },
    {
      "pattern": "system/*",
      "roles": "internal/role/openidm-admin",
      "methods": "script",
      "actions": "*"
    },
    {
      "pattern": "system/*",
      "roles": "internal/role/openidm-admin",
      "methods": "action",
      "actions": "test,testConfig,createconfiguration,liveSync,authenticate"
    },
    {
      "pattern": "repo",
      "roles": "internal/role/openidm-admin",
      "methods": "*",
      "actions": "*",
      "customAuthz": "disallowCommandAction()"
    },
    {
      "pattern": "repo/*",
      "roles": "internal/role/openidm-admin",
      "methods": "*",
      "actions": "*",
      "customAuthz": "disallowCommandAction()"
    },
    {
      "pattern": "repo/link",
      "roles": "internal/role/openidm-admin",
      "methods": "action",
      "actions": "command",
      "customAuthz": "request.additionalParameters.commandId === 'delete-mapping-links'"
    },
    {
      "pattern": "managed/*",
      "roles": "internal/role/platform-provisioning",
      "methods": "create,read,query,patch"
    },
    {
```

```
      "pattern": "internal/role/*",
      "roles": "internal/role/platform-provisioning",
      "methods": "read,query"
    },
    {
      "pattern": "profile/*",
      "roles": "internal/role/platform-provisioning",
      "methods": "create,read,action,update",
      "actions": "*"
    },
    {
      "pattern": "policy/*",
      "roles": "internal/role/platform-provisioning",
      "methods": "read,action",
      "actions": "*"
    },
    {
      "pattern": "schema/*",
      "roles": "internal/role/platform-provisioning",
      "methods": "read"
    },
    {
      "pattern": "consent",
      "roles": "internal/role/platform-provisioning",
      "methods": "action,query",
      "actions": "*"
    },
    {
      "pattern": "selfservice/kba",
      "roles": "internal/role/platform-provisioning",
      "methods": "read"
    },
    {
      "pattern": "selfservice/terms",
      "roles": "internal/role/platform-provisioning",
      "methods": "read"
    },
    {
      "pattern": "identityProviders",
      "roles": "internal/role/platform-provisioning",
      "methods": "read"
    },
    {
      "pattern": "external/email",
      "roles": "internal/role/platform-provisioning",
      "methods": "action",
      "actions": "sendTemplate"
    },
    {
      "pattern": "policy/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read,action",
      "actions": "*"
    },
    {
      "pattern": "config/ui/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*"
```

```
    },
    {
      "pattern": "authentication",
      "roles": "internal/role/openidm-authorized",
      "methods": "action",
      "actions": "reauthenticate"
    },
    {
      "pattern": "*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read,action,delete",
      "actions": "bind,unbind",
      "customAuthz": "ownDataOnly()"
    },
    {
      "pattern": "*",
      "roles": "internal/role/openidm-authorized",
      "methods": "update,patch,action",
      "actions": "patch",
      "customAuthz": "ownDataOnly() && onlyEditableManagedObjectProperties('user', []) &&
reauthIfProtectedAttributeChange()"
    },
    {
      "pattern": "selfservice/user/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "patch,action",
      "actions": "patch",
      "customAuthz": "(request.resourcePath === 'selfservice/user/' +
context.security.authorization.id) && onlyEditableManagedObjectProperties('user', [])"
    },
    {
      "pattern": "endpoint/getprocessesforuser",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*"
    },
    {
      "pattern": "endpoint/gettasksview",
      "roles": "internal/role/openidm-authorized",
      "methods": "query",
      "actions": "*"
    },
    {
      "pattern": "workflow/taskinstance/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "action",
      "actions": "complete",
      "customAuthz": "isMyTask()"
    },
    {
      "pattern": "workflow/taskinstance/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read,update",
      "actions": "*",
      "customAuthz": "canUpdateTask()"
    },
    {
      "pattern": "workflow/processinstance",
      "roles": "internal/role/openidm-authorized",
```

```
      "methods": "create",
      "actions": "*",
      "customAuthz": "isAllowedToStartProcess()"
    },
    {
      "pattern": "workflow/processdefinition/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "*",
      "actions": "read",
      "customAuthz": "isOneOfMyWorkflows()"
    },
    {
      "pattern": "managed/realm-name_user",
      "roles": "internal/role/openidm-cert",
      "methods": "patch,action",
      "actions": "patch",
      "customAuthz": "isQueryOneOf({'managed/realm-name_user': ['for-userName']}) &&
restrictPatchToFields(['password'])"
    },
    {
      "pattern": "internal/usermeta/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*",
      "customAuthz": "ownRelationship()"
    },
    {
      "pattern": "internal/notification/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read,delete",
      "actions": "*",
      "customAuthz": "ownRelationship()"
    },
    {
      "pattern": "managed/realm-name_user/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read,query",
      "actions": "*",
      "customAuthz": "ownRelationshipCollection(['idps','_meta','_notifications'])"
    },
    {
      "pattern": "notification",
      "roles": "internal/role/openidm-authorized",
      "methods": "action",
      "actions": "deleteNotificationsForTarget",
      "customAuthz": "request.additionalParameters.target ===
(context.security.authorization.component + '/' + context.security.authorization.id)"
    },
    {
      "pattern": "managed/*",
      "roles": "internal/role/openidm-authorized",
      "methods": "read",
      "actions": "*",
      "customAuthz": "ownIDP()"
    }
  ]
```

```
}
```

Grant Internal Authorization Roles Manually

Apart from the default roles that users get when they authenticate, you can grant internal authorization roles manually, over REST or through the Admin UI. This mechanism works in the same way as the granting of managed roles. For information about granting managed roles, see Grant Roles to a User in the *Object Modeling Guide*. To grant an internal role manually through the Admin UI:

1. Select Manage > User and click the user to whom you want to grant the role.
2. Select the Authorization Roles tab and click Add Authorization Roles.
3. Select Internal Role as the Type, click in the Authorization Roles field to select from the list of defined Internal Roles, then click Add.

To manually grant an internal role over REST, add a reference to the internal role to the user's `authzRoles` property. The following command adds the `openidm-admin` role to user bjensen (with ID `9dce06d4-2fc1-4830-a92b-bd35c2f6bcbb`):

```
curl \
--header "Authorization: Bearer *token*" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--cacert ca-cert.pem \
--request PATCH \
--data '[
  {
    "operation": "add",
    "field": "/authzRoles/-",
    "value": {"_ref" : "internal/role/openidm-admin"}
  }
]' \
"https://localhost:8443/openidm/managed/realm-name_user/9dce06d4-2fc1-4830-a92b-bd35c2f6bcbb"
{
  "_id": "9dce06d4-2fc1-4830-a92b-bd35c2f6bcbb",
  "_rev": "0000000050c62938",
  "mail": "bjensen@example.com",
  "givenName": "Barbara",
  "sn": "Jensen",
  "description": "Created By CSV",
  "userName": "bjensen",
  "telephoneNumber": "1234567",
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": []
}
```

You can also grant internal roles dynamically using conditional role grants in the *Object Modeling Guide*.

Note

Because internal roles are not managed objects, you cannot manipulate them in the same way as managed roles. Therefore you cannot add a user to an internal role, as you would to a managed role.

To add users directly to an internal role, add the users as values of the role's `authzMembers` property. For example:

```
curl \
--header "Authorization: Bearer *token*" \
--header "Content-Type: application/json" \
--cacert ca-cert.pem \
--request POST \
--data '{"_ref":"managed/realm-name_user/bjensen"}' \
"https://localhost:8443/openidm/internal/role/3042798d-37fd-49aa-bae3-52598d2c8dc4/authzMembers?
_action=create"
```

Administrative Users

The default Identity Cloud administrative user is `openidm-admin`. In a production environment, you might want to replace this user with a managed or internal user with the same roles, specifically the `openidm-admin` and `openidm-authorized` roles.

You can create either an internal or managed user with the same roles as the default `openidm-admin` user. To add these roles to an existing managed user, see "Grant Internal Authorization Roles Manually". The following procedure creates a new administrative internal user (`admin`):

1. Create an internal user:

```
curl \
--header "Authorization: Bearer *token*" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--cacert ca-cert.pem \
--request PUT \
--data '{
  "password": "Passw0rd"
}' \
"https://localhost:8443/openidm/internal/user/admin"
{
  "_id": "admin",
  "_rev": "00000000210f6746"
}
```

2. Add a `STATIC_USER` authentication module to the authentication configuration:

   + *Using REST*

```
curl \
--header "Authorization: Bearer *token*" \
```

```
--header "Content-Type: application/json" \
--header "Accept-API-Version: resource=1.0" \
--cacert ca-cert.pem \
--request PATCH \
--data '[
  {
    "operation": "add",
    "field": "/serverAuthContext/authModules/-",
    "value": {
      "name" : "STATIC_USER",
      "properties" : {
        "queryOnResource" : "internal/user",
        "username" : "admin",
        "password" : "Passw0rd",
        "defaultUserRoles" : [
          "internal/role/openidm-authorized",
          "internal/role/openidm-admin"
        ]
      },
      "enabled" : true
    }
  }
]' \
"https://localhost:8443/openidm/config/authentication"
{
  "_id": "authentication",
  "serverAuthContext": {
    ...
    "authModules": [
      ...
      {
        "name": "STATIC_USER",
        "properties": {
          "queryOnResource": "internal/user",
          "username": "admin",
          "password": "{encrypted password}",
          "defaultUserRoles": [
            "internal/role/openidm-authorized",
            "internal/role/openidm-admin"
          ]
        },
        "enabled": true
      },
      ...
    ]
  }
}
```

3. To verify the changes, perform a REST call or log in to the Admin UI as the new admin user. For example, query the list of internal users:

```
curl \
--header "X-OpenIDM-Username: admin" \
--header "X-OpenIDM-Password: Passw0rd" \
--header "Accept-API-Version: resource=1.0" \
--cacert ca-cert.pem \
--request GET \
"https://localhost:8443/openidm/internal/user?_queryFilter=true"
{
  "result": [
    {
      "_id": "admin",
      "_rev": "00000000f8e1665a"
    }
  ],
  ...
}
```

4. (Optional) *After* you have verified the new admin user, you can delete or disable the `openidm-admin` user:

   + *Delete 'openidm-admin' User*

   1. Delete the `openidm-admin` object:

      ```
      curl \
      --header "X-OpenIDM-Username: admin" \
      --header "X-OpenIDM-Password: Passw0rd" \
      --header "Accept-API-Version: resource=1.0" \
      --cacert ca-cert.pem \
      --request DELETE \
      "https://localhost:8443/openidm/internal/user/openidm-admin"
      {
        "_id": "openidm-admin",
        "_rev": "00000000210f6746"
      }
      ```

   2. Delete the authentication module for `"username" : "openidm-admin"`:

      + *Using REST*

      a. Get the current authentication configuration:

```
curl \
--header "Authorization: Bearer *token*" \
--header "Accept-API-Version: resource=1.0" \
--cacert ca-cert.pem \
--request GET \
"https://localhost:8443/openidm/config/authentication"
{
  "_id": "authentication",
  "serverAuthContext": {
    ...
    "authModules": [
      ...
      {
        "name": "STATIC_USER",
        "properties": {
          "queryOnResource": "internal/user",
          "username": "openidm-admin",
          "password": "&{openidm.admin.password}",
          "defaultUserRoles": [
            "internal/role/openidm-authorized",
            "internal/role/openidm-admin"
          ]
        },
        "enabled": true
      },
      ...
    ]
  }
}
```

b. Remove the authentication module for `"username" : "openidm-admin"`, and replace the authentication configuration:

```
curl \
--header "Authorization: Bearer *token*" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--cacert ca-cert.pem \
--request PUT \
--data '{
  "_id": "authentication",
  "serverAuthContext": {
    "sessionModule": {
      "name": "JWT_SESSION",
      "properties": {
        "maxTokenLifeMinutes": 120,
        "tokenIdleTimeMinutes": 30,
        "sessionOnly": true,
        "isHttpOnly": true,
        "enableDynamicRoles": false
      }
    },
    "authModules": [
      {
        "name": "STATIC_USER",
        "properties": {
          "queryOnResource": "internal/user",
          "username": "anonymous",
```

```
          "password": {
            "$crypto": {
              "type": "x-simple-encryption",
              "value": {
                "cipher": "AES/CBC/PKCS5Padding",
                "stableId": "openidm-sym-default",
                "salt": "xBlTp67ze4Ca5LTocXOpoA==",
                "data": "mdibV6UabU2M+M5MK7bjFQ==",
                "keySize": 16,
                "purpose": "idm.config.encryption",
                "iv": "36D2+FumKbaUsndNQ+/+5w==",
                "mac": "ZM8GMnh0n80QwtSH6QsNmA=="
              }
            }
          },
          "defaultUserRoles": [
            "internal/role/openidm-reg"
          ]
        },
        "enabled": true
      },
      {
        "name": "STATIC_USER",
        "properties": {
          "queryOnResource": "internal/user",
          "username": "admin",
          "password": "{encrypted password}",
          "defaultUserRoles": [
            "internal/role/openidm-authorized",
            "internal/role/openidm-admin"
          ]
        },
        "enabled": true
      },
      {
        "name": "MANAGED_USER",
        "properties": {
          "augmentSecurityContext": {
            "type": "text/javascript",
            "source": "require('auth/customAuthz').setProtectedAttributes(security)"
          },
          "queryId": "credential-query",
          "queryOnResource": "managed/realm-name_user",
          "propertyMapping": {
            "authenticationId": "username",
            "userCredential": "password",
            "userRoles": "authzRoles"
          },
          "defaultUserRoles": [
            "internal/role/openidm-authorized"
          ]
        },
        "enabled": true
      },
      {
        "name": "SOCIAL_PROVIDERS",
        "properties": {
          "defaultUserRoles": [
            "internal/role/openidm-authorized"
```

```
          ],
          "augmentSecurityContext": {
            "type": "text/javascript",
            "globals": {},
            "file": "auth/populateAsManagedUserFromRelationship.js"
          },
          "propertyMapping": {
            "userRoles": "authzRoles"
          }
        },
        "enabled": true
      }
    ]
  }
}' \
"https://localhost:8443/openidm/config/authentication"
```

+ *Disable 'openidm-admin' User*

Change the `enabled` state of the authentication module for `"username" : "openidm-admin"`:

+ *Using REST*

1. Get the current authentication configuration:

```
curl \
--header "Authorization: Bearer *token*" \
--header "Accept-API-Version: resource=1.0" \
--cacert ca-cert.pem \
--request GET \
"https://localhost:8443/openidm/config/authentication"
{
  "_id": "authentication",
  "serverAuthContext": {
    ...
    "authModules": [
      ...
      {
        "name": "STATIC_USER",
        "properties": {
          "queryOnResource": "internal/user",
          "username": "openidm-admin",
          "password": "&{openidm.admin.password}",
          "defaultUserRoles": [
            "internal/role/openidm-authorized",
            "internal/role/openidm-admin"
          ]
        },
        "enabled": true
      },
      ...
    ]
  }
}
```

2. Change the enabled state of the authentication module for `"username" : "openidm-admin"`, and replace the authentication configuration:

```
curl \
--header "Authorization: Bearer *token*" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--cacert ca-cert.pem \
--request PUT \
--data '{
  "_id": "authentication",
  "serverAuthContext": {
    "sessionModule": {
      "name": "JWT_SESSION",
      "properties": {
        "maxTokenLifeMinutes": 120,
        "tokenIdleTimeMinutes": 30,
        "sessionOnly": true,
        "isHttpOnly": true,
        "enableDynamicRoles": false
      }
    },
    "authModules": [
      {
        "name": "STATIC_USER",
        "properties": {
          "queryOnResource": "internal/user",
          "username": "anonymous",
```

```
          "password": {
            "$crypto": {
              "type": "x-simple-encryption",
              "value": {
                "cipher": "AES/CBC/PKCS5Padding",
                "stableId": "openidm-sym-default",
                "salt": "xBlTp67ze4Ca5LTocXOpoA==",
                "data": "mdibV6UabU2M+M5MK7bjFQ==",
                "keySize": 16,
                "purpose": "idm.config.encryption",
                "iv": "36D2+FumKbaUsndNQ+/+5w==",
                "mac": "ZM8GMnh0n80QwtSH6QsNmA=="
              }
            }
          },
          "defaultUserRoles": [
            "internal/role/openidm-reg"
          ]
        },
        "enabled": true
      },
      {
        "name": "STATIC_USER",
        "properties": {
          "queryOnResource": "internal/user",
          "username": "openidm-admin",
          "password": "&{openidm.admin.password}",
          "defaultUserRoles": [
            "internal/role/openidm-authorized",
            "internal/role/openidm-admin"
          ]
        },
        "enabled": false
      },
      {
        "name": "MANAGED_USER",
        "properties": {
          "augmentSecurityContext": {
            "type": "text/javascript",
            "source": "require('auth/customAuthz').setProtectedAttributes(security)"
          },
          "queryId": "credential-query",
          "queryOnResource": "managed/realm-name_user",
          "propertyMapping": {
            "authenticationId": "username",
            "userCredential": "password",
            "userRoles": "authzRoles"
          },
          "defaultUserRoles": [
            "internal/role/openidm-authorized"
          ]
        },
        "enabled": true
      },
      {
        "name": "SOCIAL_PROVIDERS",
        "properties": {
          "defaultUserRoles": [
            "internal/role/openidm-authorized"
```

```
          ],
          "augmentSecurityContext": {
            "type": "text/javascript",
            "globals": {},
            "file": "auth/populateAsManagedUserFromRelationship.js"
          },
          "propertyMapping": {
            "userRoles": "authzRoles"
          }
        },
        "enabled": true
      }
    ]
  }
}' \
"https://localhost:8443/openidm/config/authentication"
```

Chapter 3

Delegated Administration

Delegated administration lets you give fine-grained administrative access to specific users, based on a *privilege* mechanism.

How Privileges Restrict Administrative Access

Privileges enable you to grant administrative access to specific endpoints and objects, without needing to grant full administrative access to the server. For example, you might want to allow users with a help desk or support role to update the information of another user, without allowing them to delete user accounts or change the Identity Cloud system configuration.

You can use privileges to delegate specific administrative capabilities to non-administrative users, without exposing the Admin UI to those users. If a user has been granted a privilege that allows them to see a list of users and user information, for example, they can access this list directly through the End User UI.

Note

A delegated administrator does not have access to the same methods over REST as a regular administrator. Identity Cloud does not allow delegated administrator requests such as POST or DELETE. To add or remove relationships, use PATCH. For examples, see "Managed Roles" in the *Object Modeling Guide*.

The privilege mechanism requires dynamic role calculation, which is disabled by default. To enable it, set the `enableDynamicRoles` property to `true` in your `conf/authentication.json` file, or select Configure > Authentication > Session > Enable Dynamic Roles in the Admin UI. For more information about dynamic role calculation, see "Dynamic Role Calculation".

For more information on managing privileges over REST, see "Privileges" in the *REST API Reference*.

Determine Access Privileges

Identity Cloud determines what access a user has as follows:

1. Identity Cloud checks the `onRequest` script specified in `router.json`. By default, this script calls `router-authz.js`.
2. If access requirements are not satisfied, Identity Cloud then checks for any privileges associated with the user's roles.

`onResponse` and `onFailure` scripts are supported when using privileges. `onFailure` scripts are called only if both the `onRequest` script *and* the privilege filter fail. `onRequest`, `onResponse`, and `onFailure` scripts are not required for the privilege mechanism.

Create Privileges

Privileges are assigned to internal roles. A privilege specifies the following information:

- The service path to which users with that internal role have access.
- The methods and actions allowed on that service path.
- The specific attributes of the objects at that service path, to which access is allowed.

You can use a query filter within a privilege so that the privilege applies to only a subset of managed objects.

Note

If you are creating an internal authorization role that lets users access the Admin UI, you must *also* add the new role to the `roles` list in the `/path/to/openidm/conf/ui-configuration.json` file. For example:

```
"roles" : {
    "internal/role/openidm-authorized" : "ui-user",
    "internal/role/openidm-admin" : "ui-admin",
    "internal/role/support" : "ui-admin"
}
```

If you do not add the role here, users attempting to log in to the Admin UI will see the following error:

```
You are logged in but do not have access to this page.
```

The `privileges` property is an array, and can contain multiple privileges. Each privilege can contain:

accessFlags

A list of attributes within a managed object that you wish to give access to. Each attribute has two fields:

- `attribute`—the name of the property you are granting access to.

- `readOnly` (boolean)—determines what level of access is allowed.

Attributes marked as `"readOnly": true` can be viewed but not edited. Attributes marked as `"readOnly": false` can be both viewed and edited. Attributes that are not listed in the `accessFlags` array cannot be viewed or edited.

Note

- Privileges are not automatically aware of changes to the managed object schema. If new properties are added, removed, or made mandatory, you must update any existing privileges to account for these changes. When a new property is added, it has a default permission level of `NONE` in existing privileges, including when the privilege is set to access all attributes.
- Identity Cloud applies policy validation when creating or updating a privilege, to ensure that all required properties are writable when the `CREATE` permission is assigned. This validation does not run when schema changes are made, however, so you must verify that any existing privileges adhere to defined policies.

actions

A list of the specific actions allowed if the `ACTION` permission has been specified. Allowed actions must be explicitly listed.

description (optional)

A description of the privilege.

filter (optional)

This property lets you apply a static or dynamic query filter to the privilege, which can be used to limit the scope of what the privilege allows the user to access.

Static Filter Example

To allow a delegated administrator to access information only about users for the `stateProvince` Washington, include a static filter such as:

```
filter : "stateProvince eq \"Washington\""
```

Dynamic Filter Example

Dynamic filters insert values from the authenticated resource. To allow a delegated administrator to access information only about users in their own `stateProvince`, include a dynamic filter by wrapping the parameter in curly braces:

```
filter : "stateProvince eq \"{{stateProvince}}\""
```

Users with query filter privileges cannot edit the properties specified in the filter in ways that would cause the privilege to lose access to the object. For example, if a user with either of the

preceding example privileges attempted to edit another user's `stateProvince` field to anything not matching the query filter, the request would return a `403 Forbidden` error.

> **Note**
>
> Privilege filters are another layer of filter *in addition to* any other query filters you create. This means any output must satisfy all filters to be included.

name

The name of the privilege being created.

path

The path to the service you want to allow members of this privilege to access. For example, `managed/realm-name_user`.

permissions

A list of permissions this privilege allows for the given path. The following permissions are available:

- `VIEW`—allows reading and querying the path, such as viewing and querying managed users.
- `CREATE`—allows creation at the path, such as creating new managed users.
- `UPDATE`—allows updating or patching existing information, such as editing managed user details.
- `DELETE`—allows deletion, such as deleting users from `managed/realm-name_user`.
- `ACTION`—allows users to perform actions at the given path, such as custom scripted actions.

> **Note**
>
> Actions that require additional filtering on the results of the action are not currently supported.

Adding Privileges Using the Admin UI

The easiest way to modify privileges is using the Admin UI.

1. From the navigation bar, click Manage > Role.
2. From the Roles page, click the Internal tab, and then click an existing role (or create a new role).
3. From the *Role Name* page, click the Privileges tab.

   Identity Cloud displays the current privileges for the role.

4. To add privileges, click Add Privileges.

   - In the Add a privilege window, enter information, as necessary, and click Add.

+ *Adding Privileges Using REST*

The following example creates a new `support` role with privileges that let members view, create, and update information about users, but not delete users:

```
curl \
--header "Authorization: Bearer *token*" \
--header "Accept-API-Version: resource=1.0" \
--header "Content-Type: application/json" \
--cacert ca-cert.pem \
--request PUT \
--data '{
  "name": "support",
  "description": "Support Role",
  "privileges": [ {
    "name": "support",
    "description": "Support access to user information.",
    "path": "managed/realm-name_user",
    "permissions": [
      "VIEW", "UPDATE", "CREATE"
    ],
    "actions": [],
    "filter": null,
    "accessFlags": [
      {
        "attribute" : "userName",
        "readOnly" : false
      },
      {
        "attribute" : "mail",
        "readOnly" : false
      },
      {
        "attribute" : "givenName",
        "readOnly" : false
      },
      {
        "attribute" : "sn",
        "readOnly" : false
      },
      {
        "attribute" : "accountStatus",
        "readOnly" : true
      }
    ]
  } ]
}' \
"https://localhost:8443/openidm/internal/role/support"
{
  "_id": "support",
  "_rev": "00000000bfbac2ed",
  "name": "support",
  "description": "Support Role",
```

```
  "temporalConstraints": [],
  "condition": null,
  "privileges": [
    {
      "name": "support",
      "description": "Support access to user information.",
      "path": "managed/realm-name_user",
      "permissions": [
        "VIEW",
        "UPDATE",
        "CREATE"
      ],
      "actions": [],
      "filter": null,
      "accessFlags": [
        {
          "attribute": "userName",
          "readOnly": false
        },
        {
          "attribute": "mail",
          "readOnly": false
        },
        {
          "attribute": "givenName",
          "readOnly": false
        },
        {
          "attribute": "sn",
          "readOnly": false
        },
        {
          "attribute": "accountStatus",
          "readOnly": true
        }
      ]
    }
  ]
}
```

Policies Related to Privileges

When creating privileges, Identity Cloud runs policies found in `policy.json` and `policy.js`, including the five policies used for validating privileges:

`valid-accessFlags-object`

Verifies that `accessFlag` objects are correctly formatted. Only two fields are permitted in an `accessFlag` object: `readOnly`, which must be a boolean; and `attribute`, which must be a string.

`valid-array-items`

Verifies that each item in an array contains the properties specified in `policy.json`, and that each of those properties satisfies any specific policies applied to it. By default, this is used to verify

that each privilege contains `name`, `path`, `accessFlags`, `actions`, and `permissions` properties, and that the `filter` property is valid if included.

`valid-permissions`

Verifies that the permissions set on the privilege are all valid and can be achieved with the `accessFlags` that have been set. It checks:

- `CREATE` permissions must have write access to all properties required to create a new object.
- `CREATE` and `UPDATE` permissions must have write access to at least one property.
- `ACTION` permissions must include a list of allowed actions, with at least one action included.
- If any attributes have write access, then the privilege must also have either `CREATE` or `UPDATE` permission.
- All permissions listed must be valid types of permission: `VIEW`, `CREATE`, `UPDATE`, `ACTION`, or `DELETE`. Also, no permissions are repeated.

`valid-privilege-path`

Verifies that the `path` specified in the privilege is a valid object with a schema for Identity Cloud to reference. Only objects with a schema (such as `managed/realm-name_user`) can have privileges applied.

`valid-query-filter`

Verifies that the query filter used to filter privileges is a valid query.

For more information about policies and creating custom policies, see "*Use Policies to Validate Data*" in the *Object Modeling Guide*.

Use Privileges to Create a Delegated Administrator

You can use the Identity Cloud REST API to create an `internal/role` with privileges that have object, array, and relationship type attribute access. You can then use that role as a delegated administrator to perform operations on those attributes.

Use the following example to create a delegated administrator:

Note

If you want to experiment with delegated administrators in Postman, download and import this Postman collection.

+ *Step 1. Create a Managed Role*

To ensure a role object exists when roles are requested, you must create a managed role.

```
curl \
--header "Content-Type: application/json" \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "If-None-Match: *" \
--request PUT \
--data '{
  "name": "testManagedRole",
  "description": "a managed role for test"
}' \
"https://tenant-name.forgeblocks.com/openidm/managed/realm-name_role/testManagedRole"
{
  "_id": "testManagedRole",
  "_rev": "00000000e0945865",
  "name": "testManagedRole",
  "description": "a managed role for test"
}
```

+ *Step 2. Create a "Manager" User*

```
curl \
--header "Content-Type: application/json" \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "If-None-Match: *" \
--request PUT \
--data '{
  "userName": "psmith",
  "sn": "Smith",
  "givenName": "Patricia",
  "mail": "psmith@example.com",
  "telephoneNumber": "082082082",
  "password": "Passw0rd"
}' \
"https://tenant-name.forgeblocks.com/openidm/managed/realm-name_user/psmith"
{
  "_id": "psmith",
  "_rev": "000000008fefe160",
  "userName": "psmith",
  "sn": "Smith",
  "givenName": "Patricia",
  "mail": "psmith@example.com",
  "telephoneNumber": "082082082",
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": []
}
```

+ *Step 3. Create Additional Users*

In this step, you'll create two users with the following attributes:

- `preferences`

- `manager`
- `roles`

```
curl \
--header "Content-Type: application/json" \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "If-None-Match: *" \
--request PUT \
--data '{
  "userName": "scarter",
  "sn": "Carter",
  "givenName": "Steven",
  "mail": "scarter@example.com",
  "telephoneNumber": "082082082",
  "password": "Passw0rd",
  "preferences": {
    "updates": true,
    "marketing": false
  },
  "manager": {"_ref" : "managed/realm-name_user/psmith"},
  "roles": [{"_ref" : "managed/realm-name_role/testManagedRole"}]
}' \
"https://tenant-name.forgeblocks.com/openidm/managed/realm-name_user/scarter"
{
  "_id": "scarter",
  "_rev": "00000000a8d501f8",
  "userName": "scarter",
  "sn": "Carter",
  "givenName": "Steven",
  "mail": "scarter@example.com",
  "telephoneNumber": "082082082",
  "preferences": {
    "updates": true,
    "marketing": false
  },
  "accountStatus": "active",
  "effectiveRoles": [
    {
      "_ref": "managed/realm-name_role/testManagedRole"
    }
  ],
  "effectiveAssignments": []
}
```

```
curl \
--header "Content-Type: application/json" \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "If-None-Match: *" \
--request PUT \
--data '{
  "userName": "jdoe",
  "sn": "Doe",
  "givenName": "John",
  "mail": "jdoe@example.com",
  "telephoneNumber": "082082082",
```

```
  "password": "Passw0rd",
  "preferences": {
    "updates": true,
    "marketing": false
  },
  "manager": {"_ref" : "managed/realm-name_user/psmith"},
  "roles": [{"_ref" : "managed/realm-name_role/testManagedRole"}]
}' \
"https://tenant-name.forgeblocks.com/openidm/managed/realm-name_user/jdoe"
{
  "_id": "jdoe",
  "_rev": "00000000b174fbd4",
  "userName": "jdoe",
  "sn": "Doe",
  "givenName": "John",
  "mail": "jdoe@example.com",
  "telephoneNumber": "082082082",
  "preferences": {
    "updates": true,
    "marketing": false
  },
  "accountStatus": "active",
  "effectiveRoles": [
    {
      "_ref": "managed/realm-name_role/testManagedRole"
    }
  ],
  "effectiveAssignments": []
}
```

+ *Step 4. Create Another User*

You will delegate an internal/role with privileges to this user in the next step:

```
curl \
--header "Content-Type: application/json" \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "If-None-Match: *" \
--request PUT \
--data '{
  "userName": "bjensen",
  "sn": "Jensen",
  "givenName": "Barbara",
  "mail": "bjensen@example.com",
  "telephoneNumber": "082082082",
  "password": "Passw0rd"
}' \
"https://tenant-name.forgeblocks.com/openidm/managed/realm-name_user/bjensen"
{
  "_id": "bjensen",
  "_rev": "0000000022fae330",
  "userName": "bjensen",
  "sn": "Jensen",
  "givenName": "Barbara",
  "mail": "bjensen@example.com",
  "telephoneNumber": "082082082",
  "accountStatus": "active",
  "effectiveRoles": [],
  "effectiveAssignments": []
}
```

+ *Step 5. Create an "internal/role"*

This role will have the following privileges:

- A `managed/realm-name_user` privilege with accessFlags attributes that are of types: "String", "boolean", and "number"; but also for:
  - An object type that is not a relationship (`preferences`).
  - An object type that is a relationship (`manager`).
  - Array types that are relationships (`roles`, `authzRoles`, `reports`).
- A `managed/realm-name_role` privilege for viewing details of the "roles" property of a managed user.
- An `internal/role` privilege for viewing the details of the "authzRoles" property of a managed user.

Note

You can populate the privilege `filter` field to apply a finer level of permissions for what a delegated administrator can see or do with certain objects. The `filter` field is omitted in this example to allow all.

For properties that are *not* relationships, such as `preferences`, you can't specify finer-grained permissions. For example, you can't set permissions on `preferences/marketing`.

```
curl \
--header "Content-Type: application/json" \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "If-None-Match: *" \
--request PUT \
--data '{
  "name": "internal_role_with_object_array_and_relationship_privileges",
  "description": "an internal role that has privileges for object & array types and relationships",
  "privileges": [
    {
      "name": "managed_user_privilege",
      "path": "managed/realm-name_user",
      "permissions": [
        "VIEW",
        "CREATE",
        "UPDATE",
        "DELETE"
      ],
      "actions": [],
      "accessFlags": [
        {
          "attribute": "userName",
          "readOnly": false
        },
        {
          "attribute": "password",
          "readOnly": false
        },
        {
          "attribute": "givenName",
          "readOnly": false
        },
        {
          "attribute": "sn",
          "readOnly": false
        },
        {
          "attribute": "mail",
          "readOnly": false
        },
        {
          "attribute": "description",
          "readOnly": false
        },
        {
          "attribute": "accountStatus",
          "readOnly": false
        },
        {
          "attribute": "telephoneNumber",
          "readOnly": false
        },
        {
```

```
          "attribute": "postalAddress",
          "readOnly": false
        },
        {
          "attribute": "city",
          "readOnly": false
        },
        {
          "attribute": "postalCode",
          "readOnly": false
        },
        {
          "attribute": "country",
          "readOnly": false
        },
        {
          "attribute": "stateProvince",
          "readOnly": false
        },
        {
          "attribute": "preferences",
          "readOnly": false
        },
        {
          "attribute": "roles",
          "readOnly": false
        },
        {
          "attribute": "manager",
          "readOnly": false
        },
        {
          "attribute": "authzRoles",
          "readOnly": false
        },
        {
          "attribute": "reports",
          "readOnly": false
        }
      ]
    },
    {
      "name": "managed_role_privilege",
      "path": "managed/realm-name_role",
      "permissions": [
        "VIEW"
      ],
      "actions": [],
      "accessFlags": [
        {
          "attribute": "name",
          "readOnly": true
        },
        {
          "attribute": "description",
          "readOnly": true
        }
      ]
    },
```

```
    {
      "name": "internal_role_privilege",
      "path": "internal/role",
      "permissions": [
        "VIEW"
      ],
      "actions": [],
      "accessFlags": [
        {
          "attribute": "name",
          "readOnly": true
        },
        {
          "attribute": "description",
          "readOnly": true
        },
        {
          "attribute": "authzMembers",
          "readOnly": true
        }
      ]
    }
  ]
}' \
"https://tenant-name.forgeblocks.com/openidm/internal/role/testInternalRole"
{
  "_id": "testInternalRole",
  "_rev": "0000000079775d19",
  "name": "internal_role_with_object_array_and_relationship_privileges",
  "description": "an internal role that has privileges for object & array types and relationships",
  "temporalConstraints": null,
  "condition": null,
  "privileges": [
    {
      "name": "managed_user_privilege",
      "path": "managed/realm-name_user",
      "permissions": [
        "VIEW",
        "CREATE",
        "UPDATE",
        "DELETE"
      ],
      "actions": [],
      "accessFlags": [
        {
          "attribute": "userName",
          "readOnly": false
        },
        {
          "attribute": "password",
          "readOnly": false
        },
        {
          "attribute": "givenName",
          "readOnly": false
        },
        {
          "attribute": "sn",
          "readOnly": false
```

```
        },
        {
          "attribute": "mail",
          "readOnly": false
        },
        {
          "attribute": "description",
          "readOnly": false
        },
        {
          "attribute": "accountStatus",
          "readOnly": false
        },
        {
          "attribute": "telephoneNumber",
          "readOnly": false
        },
        {
          "attribute": "postalAddress",
          "readOnly": false
        },
        {
          "attribute": "city",
          "readOnly": false
        },
        {
          "attribute": "postalCode",
          "readOnly": false
        },
        {
          "attribute": "country",
          "readOnly": false
        },
        {
          "attribute": "stateProvince",
          "readOnly": false
        },
        {
          "attribute": "preferences",
          "readOnly": false
        },
        {
          "attribute": "roles",
          "readOnly": false
        },
        {
          "attribute": "manager",
          "readOnly": false
        },
        {
          "attribute": "authzRoles",
          "readOnly": false
        },
        {
          "attribute": "reports",
          "readOnly": false
        }
      ]
    },
```

```
    {
      "name": "managed_role_privilege",
      "path": "managed/realm-name_role",
      "permissions": [
        "VIEW"
      ],
      "actions": [],
      "accessFlags": [
        {
          "attribute": "name",
          "readOnly": true
        },
        {
          "attribute": "description",
          "readOnly": true
        }
      ]
    },
    {
      "name": "internal_role_privilege",
      "path": "internal/role",
      "permissions": [
        "VIEW"
      ],
      "actions": [],
      "accessFlags": [
        {
          "attribute": "name",
          "readOnly": true
        },
        {
          "attribute": "description",
          "readOnly": true
        },
        {
          "attribute": "authzMembers",
          "readOnly": true
        }
      ]
    }
  ]
}
```

+ *Step 6. Create the Relationship Between User and "internal/role"*

In this step, assign the internal/role from step 5 to the user created in step 4 by creating a relationship:

```
curl \
--header "X-OpenIDM-Username: openidm-admin" \
--header "X-OpenIDM-Password: openidm-admin" \
--header "Content-Type: application/json" \
--request POST \
--data '{
  "_ref": "managed/realm-name_user/bjensen",
  "_refProperties": {}
}' \
"https://tenant-name.forgeblocks.com/openidm/internal/role/testInternalRole/authzMembers?
_action=create"
{
  "_id": "732d3ab1-4319-41de-801b-80f4f4c97ef2",
  "_rev": "00000000e6dd99e0",
  "_ref": "managed/realm-name_user/bjensen",
  "_refResourceCollection": "managed/realm-name_user",
  "_refResourceId": "bjensen",
  "_refProperties": {
    "_id": "732d3ab1-4319-41de-801b-80f4f4c97ef2",
    "_rev": "00000000e6dd99e0"
  }
}
```

+ *Step 7. Perform Operations as a Delegated Administrator*

You can now perform operations as a delegated administrator, such as:

+ *Query All Users*

The query results display all users' properties that are allowed by the privileges:

```
curl \
--header "X-OpenIDM-Username: bjensen" \
--header "X-OpenIDM-Password: Passw0rd" \
--request GET \
"https://tenant-name.forgeblocks.com/openidm/managed/realm-name_user?
_queryFilter=true&_pageSize=100&_fields=*,*_ref/*"
{
  "result": [
    {
      "_id": "psmith",
      "_rev": "000000008fefe160",
      "userName": "psmith",
      "sn": "Smith",
      "givenName": "Patricia",
      "mail": "psmith@example.com",
      "telephoneNumber": "082082082",
      "accountStatus": "active",
      "reports": [
        {
          "_ref": "managed/realm-name_user/scarter",
          "_refResourceCollection": "managed/realm-name_user",
          "_refResourceId": "scarter",
          "_refProperties": {
            "_id": "c4e296ba-b0bb-44b8-a3e5-8d7c1656cef2",
```

```
            "_rev": "00000000e6f694a4"
          },
          "userName": "scarter",
          "sn": "Carter",
          "givenName": "Steven",
          "mail": "scarter@example.com",
          "telephoneNumber": "082082082",
          "preferences": {
            "updates": true,
            "marketing": false
          },
          "accountStatus": "active",
          "_rev": "00000000a8d501f8",
          "_id": "scarter"
        },
        {
          "_ref": "managed/realm-name_user/jdoe",
          "_refResourceCollection": "managed/realm-name_user",
          "_refResourceId": "jdoe",
          "_refProperties": {
            "_id": "1e3dd17d-a540-4652-984a-60bd60e546d5",
            "_rev": "0000000066ee928d"
          },
          "userName": "jdoe",
          "sn": "Doe",
          "givenName": "John",
          "mail": "jdoe@example.com",
          "telephoneNumber": "082082082",
          "preferences": {
            "updates": true,
            "marketing": false
          },
          "accountStatus": "active",
          "_rev": "00000000b174fbd4",
          "_id": "jdoe"
        }
      ],
      "manager": null,
      "roles": [],
      "authzRoles": [],
      "_notifications": [],
      "_meta": {
        "_ref": "internal/usermeta/0c15f08b-cf2e-4408-b302-4f46a40bf943",
        "_refResourceCollection": "internal/usermeta",
        "_refResourceId": "0c15f08b-cf2e-4408-b302-4f46a40bf943",
        "_refProperties": {
          "_id": "da3e2429-ae6f-4ea6-b5db-d3112f7c9d6a",
          "_rev": "00000000fd019b55"
        },
        "_rev": "000000003d8f5ca1",
        "_id": "0c15f08b-cf2e-4408-b302-4f46a40bf943"
      }
    },
    {
      "_id": "scarter",
      "_rev": "00000000a8d501f8",
      "userName": "scarter",
      "sn": "Carter",
      "givenName": "Steven",
```

```
      "mail": "scarter@example.com",
      "telephoneNumber": "082082082",
      "preferences": {
        "updates": true,
        "marketing": false
      },
      "accountStatus": "active",
      "reports": [],
      "manager": {
        "_ref": "managed/realm-name_user/psmith",
        "_refResourceCollection": "managed/realm-name_user",
        "_refResourceId": "psmith",
        "_refProperties": {
          "_id": "c4e296ba-b0bb-44b8-a3e5-8d7c1656cef2",
          "_rev": "00000000e6f694a4"
        },
        "userName": "psmith",
        "sn": "Smith",
        "givenName": "Patricia",
        "mail": "psmith@example.com",
        "telephoneNumber": "082082082",
        "accountStatus": "active",
        "_rev": "000000008fefe160",
        "_id": "psmith"
      },
      "roles": [
        {
          "_ref": "managed/realm-name_role/testManagedRole",
          "_refResourceCollection": "managed/realm-name_role",
          "_refResourceId": "testManagedRole",
          "_refProperties": {
            "_id": "352d7864-3143-4c56-ae11-8f75c96e980a",
            "_rev": "00000000b9ef9689"
          },
          "name": "testManagedRole",
          "description": "a managed role for test",
          "_rev": "00000000e0945865",
          "_id": "testManagedRole"
        }
      ],
      "authzRoles": [],
      "_notifications": [],
      "_meta": {
        "_ref": "internal/usermeta/6677aad2-def9-4507-9ea0-edd95da8da43",
        "_refResourceCollection": "internal/usermeta",
        "_refResourceId": "6677aad2-def9-4507-9ea0-edd95da8da43",
        "_refProperties": {
          "_id": "cc32ab82-084a-455c-bf97-3f2f2a71f848",
          "_rev": "00000000f4819bb6"
        },
        "_rev": "0000000090ae5c88",
        "_id": "6677aad2-def9-4507-9ea0-edd95da8da43"
      }
    },
    {
      "_id": "jdoe",
      "_rev": "00000000b174fbd4",
      "userName": "jdoe",
      "sn": "Doe",
```

```
      "givenName": "John",
      "mail": "jdoe@example.com",
      "telephoneNumber": "082082082",
      "preferences": {
        "updates": true,
        "marketing": false
      },
      "accountStatus": "active",
      "reports": [],
      "manager": {
        "_ref": "managed/realm-name_user/psmith",
        "_refResourceCollection": "managed/realm-name_user",
        "_refResourceId": "psmith",
        "_refProperties": {
          "_id": "1e3dd17d-a540-4652-984a-60bd60e546d5",
          "_rev": "0000000066ee928d"
        },
        "userName": "psmith",
        "sn": "Smith",
        "givenName": "Patricia",
        "mail": "psmith@example.com",
        "telephoneNumber": "082082082",
        "accountStatus": "active",
        "_rev": "000000008fefe160",
        "_id": "psmith"
      },
      "roles": [
        {
          "_ref": "managed/realm-name_role/testManagedRole",
          "_refResourceCollection": "managed/realm-name_role",
          "_refResourceId": "testManagedRole",
          "_refProperties": {
            "_id": "a3f6be90-3009-4e87-af46-257306617bd9",
            "_rev": "00000000b8f69498"
          },
          "name": "testManagedRole",
          "description": "a managed role for test",
          "_rev": "00000000e0945865",
          "_id": "testManagedRole"
        }
      ],
      "authzRoles": [],
      "_notifications": [],
      "_meta": {
        "_ref": "internal/usermeta/5b844d7e-c200-4b67-9fad-fa346740c79d",
        "_refResourceCollection": "internal/usermeta",
        "_refResourceId": "5b844d7e-c200-4b67-9fad-fa346740c79d",
        "_refProperties": {
          "_id": "42aa7cf0-6726-461b-92f9-1a22dab0b3c3",
          "_rev": "000000003aa1993e"
        },
        "_rev": "000000003e4f5bba",
        "_id": "5b844d7e-c200-4b67-9fad-fa346740c79d"
      }
    },
    {
      "_id": "bjensen",
      "_rev": "0000000022fae330",
      "userName": "bjensen",
```

```
      "sn": "Jensen",
      "givenName": "Barbara",
      "mail": "bjensen@example.com",
      "telephoneNumber": "082082082",
      "accountStatus": "active",
      "reports": [],
      "manager": null,
      "roles": [],
      "authzRoles": [
        {
          "_ref": "internal/role/testInternalRole",
          "_refResourceCollection": "internal/role",
          "_refResourceId": "testInternalRole",
          "_refProperties": {
            "_id": "732d3ab1-4319-41de-801b-80f4f4c97ef2",
            "_rev": "00000000e6dd99e0"
          },
          "_id": "testInternalRole",
          "name": "internal_role_with_object_array_and_relationship_privileges",
          "description": "an internal role that has privileges for object & array types and
 relationships",
          "_rev": "0000000079775d19"
        }
      ],
      "_notifications": [],
      "_meta": {
        "_ref": "internal/usermeta/0fbeb220-5e95-42b4-9bdd-0464e23194d4",
        "_refResourceCollection": "internal/usermeta",
        "_refResourceId": "0fbeb220-5e95-42b4-9bdd-0464e23194d4",
        "_refProperties": {
          "_id": "cbdb3794-1629-424d-8d7a-9e9b0c93287f",
          "_rev": "000000002b5199f1"
        },
        "_rev": "000000002fbc5b92",
        "_id": "0fbeb220-5e95-42b4-9bdd-0464e23194d4"
      }
    }
  ],
  "resultCount": 4,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```


+ *Read a Specified User's Preferences Object*

```
curl \
--header "X-OpenIDM-Username: bjensen" \
--header "X-OpenIDM-Password: Passw0rd" \
--request GET \
"https://tenant-name.forgeblocks.com/openidm/managed/realm-name_user/jdoe?_fields=preferences"
{
  "_id": "jdoe",
  "_rev": "00000000b174fbd4",
  "preferences": {
    "updates": true,
    "marketing": false
  }
}
```

+ *Query a Specified User's Roles*

```
curl \
--header "X-OpenIDM-Username: bjensen" \
--header "X-OpenIDM-Password: Passw0rd" \
--request GET \
"https://tenant-name.forgeblocks.com/openidm/managed/realm-name_user/scarter/roles?
_queryFilter=true&_fields=*"
{
  "result": [
    {
      "_id": "352d7864-3143-4c56-ae11-8f75c96e980a",
      "_rev": "00000000b9ef9689",
      "_refResourceCollection": "managed/realm-name_role",
      "_refResourceId": "testManagedRole",
      "_refResourceRev": "00000000e0945865",
      "name": "testManagedRole",
      "description": "a managed role for test",
      "_ref": "managed/realm-name_role/testManagedRole",
      "_refProperties": {
        "_id": "352d7864-3143-4c56-ae11-8f75c96e980a",
        "_rev": "00000000b9ef9689"
      }
    }
  ],
  "resultCount": 1,
  "pagedResultsCookie": null,
  "totalPagedResultsPolicy": "NONE",
  "totalPagedResults": -1,
  "remainingPagedResults": -1
}
```

+ *Read a Specified User's Manager*

```
curl \
--header "X-OpenIDM-Username: bjensen" \
--header "X-OpenIDM-Password: Passw0rd" \
--request GET \
"https://tenant-name.forgeblocks.com/openidm/managed/realm-name_user/scarter/manager?_fields=*"
{
  "_id": "c4e296ba-b0bb-44b8-a3e5-8d7c1656cef2",
  "_rev": "00000000e6f694a4",
  "_refResourceCollection": "managed/realm-name_user",
  "_refResourceId": "psmith",
  "_refResourceRev": "000000008fefe160",
  "userName": "psmith",
  "sn": "Smith",
  "givenName": "Patricia",
  "mail": "psmith@example.com",
  "telephoneNumber": "082082082",
  "accountStatus": "active",
  "_ref": "managed/realm-name_user/psmith",
  "_refProperties": {
    "_id": "c4e296ba-b0bb-44b8-a3e5-8d7c1656cef2",
    "_rev": "00000000e6f694a4"
  }
}
```

+ *Update a Specified User's Reports*

```
curl \
--header "X-OpenIDM-Username: bjensen" \
--header "X-OpenIDM-Password: Passw0rd" \
--header "Content-Type: application/json" \
--request PATCH \
--data '[ {
   "operation" : "replace",
   "field" : "reports",
   "value" : [{"_ref" : "managed/realm-name_user/scarter"}]
} ]' \
"https://tenant-name.forgeblocks.com/openidm/managed/realm-name_user/psmith"
{
  "_id": "psmith",
  "_rev": "000000008fefe160",
  "userName": "psmith",
  "sn": "Smith",
  "givenName": "Patricia",
  "mail": "psmith@example.com",
  "telephoneNumber": "082082082",
  "accountStatus": "active"
}
```

+ *Assign a Specified User's Manager*

```
curl \
--header "X-OpenIDM-Username: bjensen" \
--header "X-OpenIDM-Password: Passw0rd" \
--header "Content-Type: application/json" \
--request PATCH \
--data '[
  {
    "operation": "add",
    "field": "manager",
    "value": {"_ref" : "managed/realm-name_user/psmith"}
  }
]' \
https://tenant-name.forgeblocks.com/openidm/managed/realm-name_user/jdoe
{
  "_id": "jdoe",
  "_rev": "00000000b174fbd4",
  "userName": "jdoe",
  "sn": "Doe",
  "givenName": "John",
  "mail": "jdoe@example.com",
  "telephoneNumber": "082082082",
  "preferences": {
    "updates": true,
    "marketing": false
  },
  "accountStatus": "active"
}
```

+ *Remove a Specified User's Manager*

```
curl \
--header "X-OpenIDM-Username: bjensen" \
--header "X-OpenIDM-Password: Passw0rd" \
--header "Content-Type: application/json" \
--request PATCH \
--data '[
  {
     "operation": "remove",
     "field": "manager"
  }
]' \
https://tenant-name.forgeblocks.com/openidm/managed/realm-name_user/jdoe
{
  "_id": "jdoe",
  "_rev": "00000000b174fbd4",
  "userName": "jdoe",
  "sn": "Doe",
  "givenName": "John",
  "mail": "jdoe@example.com",
  "telephoneNumber": "082082082",
  "preferences": {
    "updates": true,
    "marketing": false
  },
  "accountStatus": "active"
}
```

+ *Update a Specified User's Manager*

```
curl \
--header "X-OpenIDM-Username: bjensen" \
--header "X-OpenIDM-Password: Passw0rd" \
--header "Content-Type: application/json" \
--request PATCH \
--data '[
  {
    "operation": "replace",
    "field": "manager",
    "value": {"_ref" : "managed/realm-name_user/jdoe"}
  }
]' \
"https://tenant-name.forgeblocks.com/openidm/managed/realm-name_user/scarter"
{
  "_id": "scarter",
  "_rev": "00000000a8d501f8",
  "userName": "scarter",
  "sn": "Carter",
  "givenName": "Steven",
  "mail": "scarter@example.com",
  "telephoneNumber": "082082082",
  "preferences": {
    "updates": true,
    "marketing": false
  },
  "accountStatus": "active"
}
```

+ *Delete a Specified User*

```
curl \
--header "X-OpenIDM-Username: bjensen" \
--header "X-OpenIDM-Password: Passw0rd" \
--header "Content-Type: application/json" \
--request DELETE \
"https://tenant-name.forgeblocks.com/openidm/managed/realm-name_user/psmith"
{
  "_id": "psmith",
  "_rev": "000000008fefe160",
  "userName": "psmith",
  "sn": "Smith",
  "givenName": "Patricia",
  "mail": "psmith@example.com",
  "telephoneNumber": "082082082",
  "accountStatus": "active"
}
```

+ *Create a User*

- Using POST:

```
curl \
--header "Content-Type: application/json" \
--header "X-OpenIDM-Username: bjensen" \
--header "X-OpenIDM-Password: Passw0rd" \
--request POST \
--data '{
  "userName": "psmith",
  "sn": "Smith",
  "givenName": "Patricia",
  "mail": "psmith@example.com",
  "telephoneNumber": "082082082",
  "password": "Passw0rd"
}' \
"https://tenant-name.forgeblocks.com/openidm/managed/realm-name_user"
{
  "_id": "e5f6a856-9f3c-49fd-904c-c5f87004b682",
  "_rev": "000000004bbde938",
  "userName": "psmith",
  "sn": "Smith",
  "givenName": "Patricia",
  "mail": "psmith@example.com",
  "telephoneNumber": "082082082",
  "accountStatus": "active"
}
```

- Using PUT:

```
curl \
--header "Content-Type: application/json" \
--header "X-OpenIDM-Username: bjensen" \
--header "X-OpenIDM-Password: Passw0rd" \
--header "If-None-Match: *" \
--request PUT \
--data '{
  "userName": "psmith",
  "sn": "Smith",
  "givenName": "Patricia",
  "mail": "psmith@example.com",
  "telephoneNumber": "082082082",
  "password": "Passw0rd"
}' \
"https://tenant-name.forgeblocks.com/openidm/managed/realm-name_user/psmith"
{
  "_id": "psmith",
  "_rev": "00000000658fe17a",
  "userName": "psmith",
  "sn": "Smith",
  "givenName": "Patricia",
  "mail": "psmith@example.com",
  "telephoneNumber": "082082082",
  "accountStatus": "active"
}
```

Note

For more examples, including working with filters, see the Postman collection.

Note

All patches are done with a PATCH request. Delegated administrator operations do not currently support using POST actions for patch requests (POST `_action=patch` will not work).

Get Privileges on a Resource

To determine which privileges a user has on a service, you can query the privilege endpoint for a given resource path or object, based on the user you are currently logged in as. For example, if bjensen is a member of the support role mentioned in the previous example, checking their privileges for the `managed/realm-name_user` resource would look like this:

```
curl \
--header "X-OpenIDM-UserName: bjensen" \
--header "X-OpenIDM-Password: Passw0rd" \
--header "Accept-API-Version: resource=1.0" \
--cacert ca-cert.pem \
--request GET \
"https://localhost:8443/openidm/privilege/managed/realm-name_user"
```

```
{
  "VIEW": {
    "allowed": true,
    "properties": [
      "userName",
      "givenName",
      "sn",
      "mail",
      "accountStatus"
    ]
  },
  "CREATE": {
    "allowed": true,
    "properties": [
      "userName",
      "givenName",
      "sn",
      "mail"
    ]
  },
  "UPDATE": {
    "allowed": true,
    "properties": [
      "userName",
      "givenName",
      "sn",
      "mail"
    ]
  },
  "DELETE": {
    "allowed": false
  },
  "ACTION": {
    "allowed": false,
    "actions": []
  }
}
```

In the above example, `accountStatus` is listed as a property for `VIEW`, but not for `CREATE` or `UPDATE`, because the privilege sets this property to be read only. Since both `CREATE` and `UPDATE` need the ability to write to a property, setting `readOnly` to false applies to both permissions. If you need more granular control, split these permissions into two privileges.

In addition to checking privileges for a resource, it is also possible to check privileges for specific objects within a resource, such as `managed/realm-name_user/scarter`.

Identity Cloud Glossary

correlation query	A correlation query specifies an expression that matches existing entries in a source repository to one or more entries in a target repository. A correlation query might be built with a script, but it is not the same as a correlation script. For more information, see "*Correlating Source Objects With Existing Target Objects*" in the *Synchronization Guide*.
correlation script	A correlation script matches existing entries in a source repository, and returns the IDs of one or more matching entries on a target repository. While it skips the intermediate step associated with a `correlation query`, a correlation script can be relatively complex, based on the operations of the script.
entitlement	An entitlement is a collection of attributes that can be added to a user entry via roles. As such, it is a specialized type of `assignment`. A user or device with an entitlement gets access rights to specified resources. An entitlement is a property of a managed object.
JCE	Java Cryptographic Extension, which is part of the Java Cryptography Architecture, provides a framework for encryption, key generation, and digital signatures.
JSON	JavaScript Object Notation, a lightweight data interchange format based on a subset of JavaScript syntax. For more information, see the JSON site.
JSON Pointer	A JSON Pointer defines a string syntax for identifying a specific value within a JSON document. For information about JSON Pointer syntax, see the JSON Pointer RFC.

JWT	JSON Web Token. As noted in the JSON Web Token draft IETF Memo, "JSON Web Token (JWT) is a compact URL-safe means of representing claims to be transferred between two parties." For Identity Cloud, the JWT is associated with the `JWT_SESSION` authentication module.
managed object	An object that represents the identity-related data managed by Identity Cloud. Managed objects are configurable, JSON-based data structures that Identity Cloud stores in its pluggable repository. The default configuration of a managed object is that of a user, but you can define any kind of managed object, for example, groups or roles.
mapping	A policy that is defined between a source object and a target object during reconciliation or synchronization. A mapping can also define a trigger for validation, customization, filtering, and transformation of source and target objects.
OSGi	A module system and service platform for the Java programming language that implements a complete and dynamic component model. For more information, see What is OSGi? Currently, only the Apache Felix container is supported.
reconciliation	During reconciliation, comparisons are made between managed objects and objects on source or target systems. Reconciliation can result in one or more specified actions, including, but not limited to, synchronization.
resource	An external system, database, directory server, or other source of identity data to be managed and audited by the identity management system.
REST	Representational State Transfer. A software architecture style for exposing resources, using the technologies and protocols of the World Wide Web. REST describes how distributed data objects, or resources, can be defined and addressed.
role	Identity Cloud distinguishes between two distinct role types - provisioning roles and authorization roles. For more information, see "Managed Roles" in the *Object Modeling Guide*.
source object	In the context of reconciliation, a source object is a data object on the source system, that Identity Cloud scans before attempting to find a corresponding object on the target system. Depending on the defined mapping, Identity Cloud then adjusts the object on the target system (target object).
synchronization	The synchronization process creates, updates, or deletes objects on a target system, based on the defined mappings from the source system. Synchronization can be scheduled or on demand.

system object	A pluggable representation of an object on an external system. For example, a user entry that is stored in an external LDAP directory is represented as a system object in Identity Cloud for the period during which Identity Cloud requires access to that entry. System objects follow the same RESTful resource-based design principles as managed objects.
target object	In the context of reconciliation, a target object is a data object on the target system, that Identity Cloud scans after locating its corresponding object on the source system. Depending on the defined mapping, Identity Cloud then adjusts the target object to match the corresponding source object.