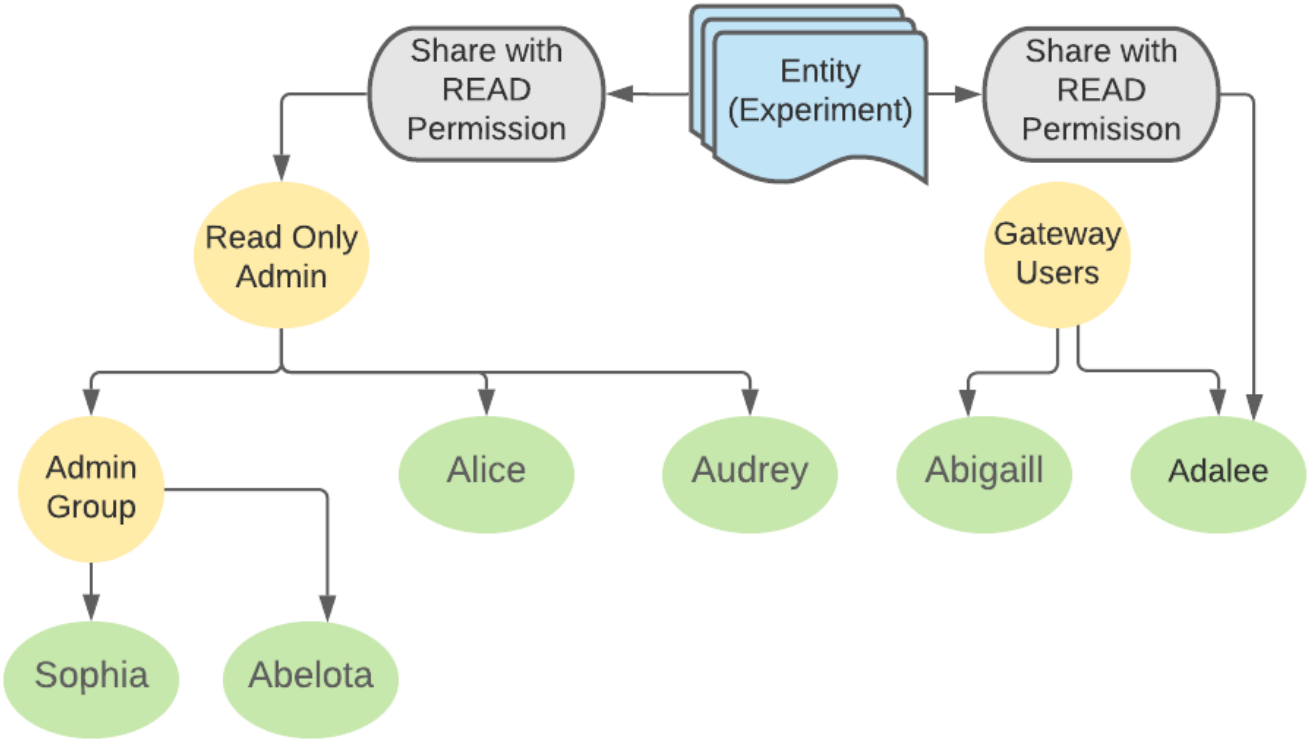
Gateways 2020:Custos Tutorial Hands-On Exercises

This tutorial explains how Custos Sharing Service can be used to impose fine-grained authorization to protect resources and provide specific permissions for users and groups to access a protected resource.

Use Case (Hirachical group-based authorization)

Creates an Experiment entity and assign read permission to a particular user group and explicitly to a specific user in another group. Evaluates permissions for each user by validating given permission. The following diagram depicts the user, group, and permission hierarchy.



According to the diagram, permissions shared with Group A should inherit to members of Group A and Group B. Permissions are shared with Adalee explicitly. So Adalee also should have access but Abigail should not have access to the resource.

User	Has Access	Access granted type
Alice	True	Through group A
Audrey	True	Through group A
Sophia	True	Through group B via group A
Abelota	True	Through group B via group A
Abigail	False	
Adalee	True	Direct sharing

Prerequisites

- Should have Google Account
- Should have docker and docker-compose installed
- Refer Docker for Windows and Mac and Docker Engine for Linux

Steps

- Download custos_tutorial_2020_notebook
- Visit https://colab.research.google.com/
    - Log in using any of your Google Accounts
- Click File upload File
- Upload downloaded file from the above link - 'custos_tutorial_2020.ipynb'
- Replace Custos Id and Custos Secret with provided credentials
- Run each step sequentially

Source Code

```
import os
import json
import random, string

from custos.clients.user_management_client import UserManagementClient
from custos.clients.group_management_client import GroupManagementClient
from custos.clients.resource_secret_management_client import
ResourceSecretManagementClient
from custos.clients.sharing_management_client import
SharingManagementClient
from custos.clients.identity_management_client import
IdentityManagementClient

from custos.transport.settings import CustosServerClientSettings
import custos.clients.utils.utilities as utl

from google.protobuf.json_format import MessageToJson

# read settings
custos_settings = CustosServerClientSettings(custos_host='custos.scigap.
org',
                                             custos_port='port',
                                             custos_client_id='custos
Id',
                                             custos_client_sec='custos
sec')

# create custos user management client
user_management_client = UserManagementClient(custos_settings)

# create custos group management client
group_management_client = GroupManagementClient(custos_settings)

# create custos resource secret client
resource_secret_client = ResourceSecretManagementClient(custos_settings)

# create sharing management client
sharing_management_client = SharingManagementClient(custos_settings)

# create identity management client
identity_management_client = IdentityManagementClient(custos_settings)

# obtain base 64 encoded token for tenant
b64_encoded_custos_token = utl.get_token(custos_settings
=custos_settings)

login_user_id = "admin"
login_user_password = "password"

created_groups = {}

resource_ids = []


def verifiy_admin_user():
    response = user_management_client.get_user(token
=b64_encoded_custos_token, username=login_user_id)
    user_management_client.update_user_profile(
        token=b64_encoded_custos_token,
        username=response.username,
        email=response.email,
        first_name=response.first_name,
        last_name=response.last_name)


def register_users(users):
    for user in users:
        print("Registering user: " + user['username'])
        user_management_client.register_user(token
```

```
=b64_encoded_custos_token,
                                                    username=user['username'],
                                                    first_name=user[
'first_name'],
                                                    last_name=user['last_name']
,
                                                    password=user['password'],
                                                    email=user['email'],
                                                    is_temp_password=False)

        user_management_client.enable_user(token
=b64_encoded_custos_token, username=user['username'])


def create_groups(groups):
    for group in groups:
        print("Creating group: " + group['name'])
        grResponse = group_management_client.create_groups(token
=b64_encoded_custos_token,
                                                                name=group[
'name'],
                                                                description
=group['description'],
                                                                owner_id
=group['owner_id'])
        resp = MessageToJson(grResponse)
        respData = json.loads(resp)
        created_groups[respData['groups'][0]['name']] = respData[
'groups'][0]['id']


def allocate_users_to_groups(user_group_mapping):
    for usr_map in user_group_mapping:
        group_id = created_groups[usr_map['group_name']]
        print("Assigning user " + usr_map['username'] + " to group " +
usr_map['group_name'])
        group_management_client.add_user_to_group(token
=b64_encoded_custos_token,
                                                      username=usr_map[
'username'],
                                                      group_id=group_id,
                                                      membership_type=
'Member'
                                                      )


def allocate_child_group_to_parent_group(gr_gr_mapping):
    for gr_map in gr_gr_mapping:
        child_id = created_groups[gr_map['child_name']]
        parent_id = created_groups[gr_map['parent_name']]
        print("Assigning child group " + gr_map['child_name'] + " to
parent group " + gr_map['parent_name'])
        group_management_client.add_child_group(token
=b64_encoded_custos_token,
                                                    parent_group_id
=parent_id,
                                                    child_group_id=child_id)


def create_permissions(permissions):
    for perm in permissions:
        print("Creating permission " + perm['id'])
        sharing_management_client.create_permission_type(token
=b64_encoded_custos_token,
                                                         client_id
=custos_settings.CUSTOS_CLIENT_ID,
                                                         id=perm['id'],
                                                         name=perm[
'name'],
                                                         description
=perm['description'])


def create_entity_types(entity_types):
    for type in entity_types:
        print("Creating entity types " + type['id'])
        sharing_management_client.create_entity_type(token
=b64_encoded_custos_token,
                                                        client_id
=custos_settings.CUSTOS_CLIENT_ID,
                                                        id=type['id'],
                                                        name=type['name'],
                                                        description=type[
'description'])
```

```
def register_resources(resources):
    for resource in resources:
        id = resource['name'].join(random.choice(string.ascii_letters)
for x in range(5))
        resource_ids.append(id)
        resource['id'] = id
        print("Register resources " + resource['name'] + " generated ID
: " + resource['id'])
        sharing_management_client.create_entity(token
=b64_encoded_custos_token,
                                                client_id
=custos_settings.CUSTOS_CLIENT_ID,
                                                id=resource['id'],
                                                name=resource['name'],
                                                description=resource[
'description'],
                                                owner_id=resource[
'user_id'],
                                                type=resource['type'],
                                                parent_id='')


def share_resource_with_user(sharings):
    for shr in sharings:
        print("Sharing entity " + shr['entity_id'] + " with user " + shr
['user_id'] + " with permission " + shr[
            'permission_type'])
        sharing_management_client.share_entity_with_users(token
=b64_encoded_custos_token,
                                                           client_id
=custos_settings.CUSTOS_CLIENT_ID,
                                                           entity_id=shr[
'entity_id'],
                                                           permission_type=shr['permission_type'],
                                                           user_id=shr[
'user_id']
                                                           )


def share_resource_with_group(gr_sharings):
    for shr in gr_sharings:
        group_id = created_groups[shr['group_name']]
        print("Sharing entity " + shr['entity_id'] + " with group " +
shr['group_name'] + " with permission " + shr[
            'permission_type'])
        sharing_management_client.share_entity_with_groups(token
=b64_encoded_custos_token,
                                                            client_id
=custos_settings.CUSTOS_CLIENT_ID,
                                                            entity_id=shr
['entity_id'],
                                                            permission_type=shr['permission_type'],
                                                            group_id
=group_id)


def check_user_permissions(users):
    for user in users:
        access = sharing_management_client.user_has_access(token
=b64_encoded_custos_token,
                                                            client_id
=custos_settings.CUSTOS_CLIENT_ID,
                                                            entity_id
=resource_ids[0],
                                                            permission_type="READ",
                                                            user_id=user[
'username'])
        usr = user['username']
        print("Access for user " + usr + " : " + str(access))


users = [
    {
        'username': 'alice',
        'first_name': 'Alice',
        'last_name': 'Aron',
        'password': '12345678',
        'email': 'alice@gmail.com'
    },
```

```
    {
        'username': 'audrey',
        'first_name': 'Audrey',
        'last_name': 'Aron',
        'password': '12345678',
        'email': 'audrey@gmail.com'
    },
    {
        'username': 'sophia',
        'first_name': 'Sophia',
        'last_name': 'Aron',
        'password': '12345678',
        'email': 'sophia@gmail.com'
    },
    {
        'username': 'abelota',
        'first_name': 'Abelota',
        'last_name': 'Aron',
        'password': '12345678',
        'email': 'abelota@gmail.com'
    },
    {
        'username': 'abigaill',
        'first_name': 'Abigaill',
        'last_name': 'Aron',
        'password': '12345678',
        'email': 'abigaill@gmail.com'
    },
    {
        'username': 'adalee',
        'first_name': 'Adalee',
        'last_name': 'Aron',
        'password': '12345678',
        'email': 'adalee@gmail.com'
    }
]

groups = [
    {
        'name': 'Admin',
        'description': 'Group for gateway read only admins',
        'owner_id': 'admin'
    },
    {
        'name': 'Read Only Admin',
        'description': 'Group for gateway admins',
        'owner_id': 'admin'
    },
    {
        'name': 'Gateway User',
        'description': 'Group  for gateway users',
        'owner_id': 'admin'
    }
]

user_group_mapping = [
    {
        'group_name': 'Admin',
        'username': 'alice'
    },
    {
        'group_name': 'Admin',
        'username': 'audrey'
    },
    {
        'group_name': 'Read Only Admin',
        'username': 'sophia'
    },
    {
        'group_name': 'Read Only Admin',
        'username': 'abelota'
    },
    {
        'group_name': 'Gateway User',
        'username': 'abigaill'
    },
    {
        'group_name': 'Gateway User',
        'username': 'adalee'
    }
]

child_gr_parent_gr_mapping = [
    {
```

```
        "child_name": 'Admin',
        "parent_name": 'Read Only Admin'
    }
]

permissions = [
    {
        'id': 'OWNER',
        'name': 'OWNER',
        'description': 'Owner permission'
    },
    {
        'id': 'READ',
        'name': 'READ',
        'description': 'Read permission'
    },
    {
        'id': 'WRITE',
        'name': 'WRITE',
        'description': 'WRITE permission'
    }
]
entity_types = [
    {
        'id': 'PROJECT',
        'name': 'PROJECT',
        'description': 'PROJECT entity type'
    },
    {
        'id': 'EXPERIMENT',
        'name': 'EXPERIMENT',
        'description': 'EXPERIMENT entity type'
    }
]

verifiy_admin_user()

# Register users
register_users(users)

# Create groups
create_groups(groups)

# Allocate users to groups
allocate_users_to_groups(user_group_mapping)

# Allocate child groups to parent group
allocate_child_group_to_parent_group(child_gr_parent_gr_mapping)

# Create permissions
create_permissions(permissions)

# Create entity types
create_entity_types(entity_types)



resources = [
    {
        'name': 'SEAGRD_EXP',
        'description': 'Register an experiment',
        'user_id': 'admin',
        'type': 'EXPERIMENT'
    }
]

# Register resources
register_resources(resources)

sharings = [
    {
        "entity_id": resource_ids[0],
        "permission_type": "READ",
        "type": "EXPERIMENT",
        "user_id": "adalee"
    }
]

gr_sharings = [{

    "entity_id": resource_ids[0],
    "permission_type": "READ",
    "type": "EXPERIMENT",
    "group_name": 'Read Only Admin'
```

```
}]


# Share resource with users
share_resource_with_user(sharings)

# Share resource with group
share_resource_with_group(gr_sharings)

# Check user permissions
check_user_permissions(users)
```

- Output

```
Registering user: alice
Registering user: audrey
Registering user: sophia
Registering user: abelota
Registering user: abigaill
Registering user: adalee
Creating group: Admin
Creating group: Read Only Admin
Creating group: Gateway User
Assigning user alice to group Admin
Assigning user audrey to group Admin
Assigning user sophia to group Read Only Admin
Assigning user abelota to group Read Only Admin
Assigning user abigaill to group Gateway User
Assigning user adalee to group Gateway User
Assigning child group Admin to parent group Read Only Admin
Creating permission OWNER
Creating permission READ
Creating permission WRITE
Creating entity types PROJECT
Creating entity types EXPERIMENT
Register resources SEAGRD_EXP generated ID : OSEAGRD_EXPNSEAGRD_EXPLSEAGRD_EXPdSEAGRD_EXPR
Sharing entity OSEAGRD_EXPNSEAGRD_EXPLSEAGRD_EXPdSEAGRD_EXPR with user abigaill with permission READ
Sharing entity OSEAGRD_EXPNSEAGRD_EXPLSEAGRD_EXPdSEAGRD_EXPR with group Read Only Admin with permission
READ
Access for user alice : True
Access for user audrey : True
Access for user sophia : True
Access for user abelota : True
Access for user abigaill :False
Access for user adalee :True
```

Verify via sample web gateway

- Download docker-compose.yml into any location in your computer
- Open up the file and add your Custos client Id and client secret
- Run docker-compose up -d
- Go to localhost:8080
- Login using testuser credentials

you should be able to see users, groups, and sharings created through python SDK are synced with the local setup demo gateway.