WOG CENTRAL ACCOUNT MANAGEMENT

API Interface Specification

Abstract

This document describes interface specification of CAM API that will be implemented in Agency Applications

WOG CAM Central Team

cam@tech.gov.sg

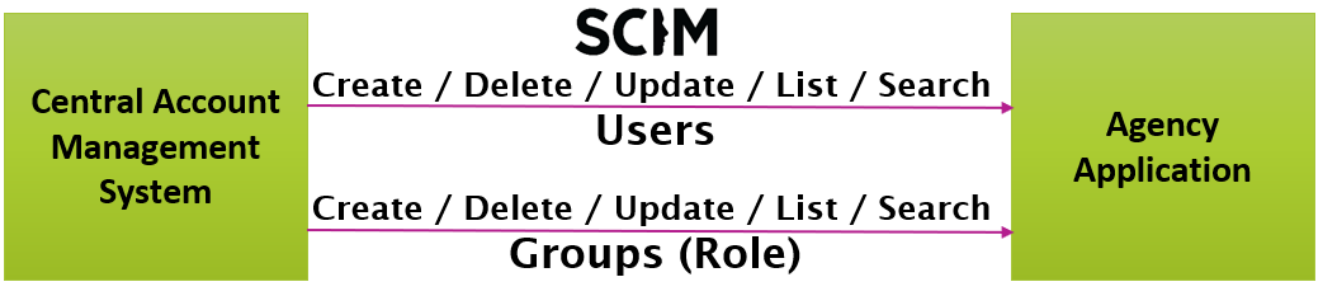
Contents

Document Change Log

Version	Release Date	Summary of Change	Updated by
0.1	01 Jul 2020	Initial baseline	Yohan S
0.2	24 Oct 2020	Extend SCIM user and group object attributes based on requirements	Yohan S
0.3	26 Nov 2020	Specify mandatory attributes in the response payload	Yohan S
0.4	04 Dec 2020	Add Group Access Right Info attribute in group object to facilitate access review	Yohan S
0.5	17 Feb 2021	1. Add 404 response code, body, and schemas code according to rfc7644 for all APIs 2. API Authentication a. Update Request Timestamp format b. Update Example of HTTPS Authorisation Header c. Update .NET and Java sample code to include timestamp check 3. Remove “Create User API” 4. Remove Appendix	Tai Yet CHONG
1.0	1 March 2021	1. Release version 1.0	Tai Yet CHONG
1.1	23 March 2021	1. Updated extendedUser and extendedGroup schema and their json schema filenames. 2. Added mandatory attributes “isPrivileged” attribute to extendedUser schema 3. Removed ListUserResponse and ListGroupResponse schemas and their schema files to use the standard RFC7644 ListResponse schema. 4. Removed extendedPatchop and its schema file from Disable or Enabler User API, Remove User API, Add or Remove User from Group API & Remove Group API. 5. Updated response syntax in Get User and Get User List API by grouping the enterprise and extended CAM users into a json container – to distinguish the core attributes and extended attributes. 6. Updated response syntax in Get Group and Get Group List API by grouping the extended CAM groups into a json container – to distinguish the core attributes and extended attributes. 7. Replaced .NET API Client Simulator with Postman API Client script under section Tools and Sample Code.	Tai Yet CHONG

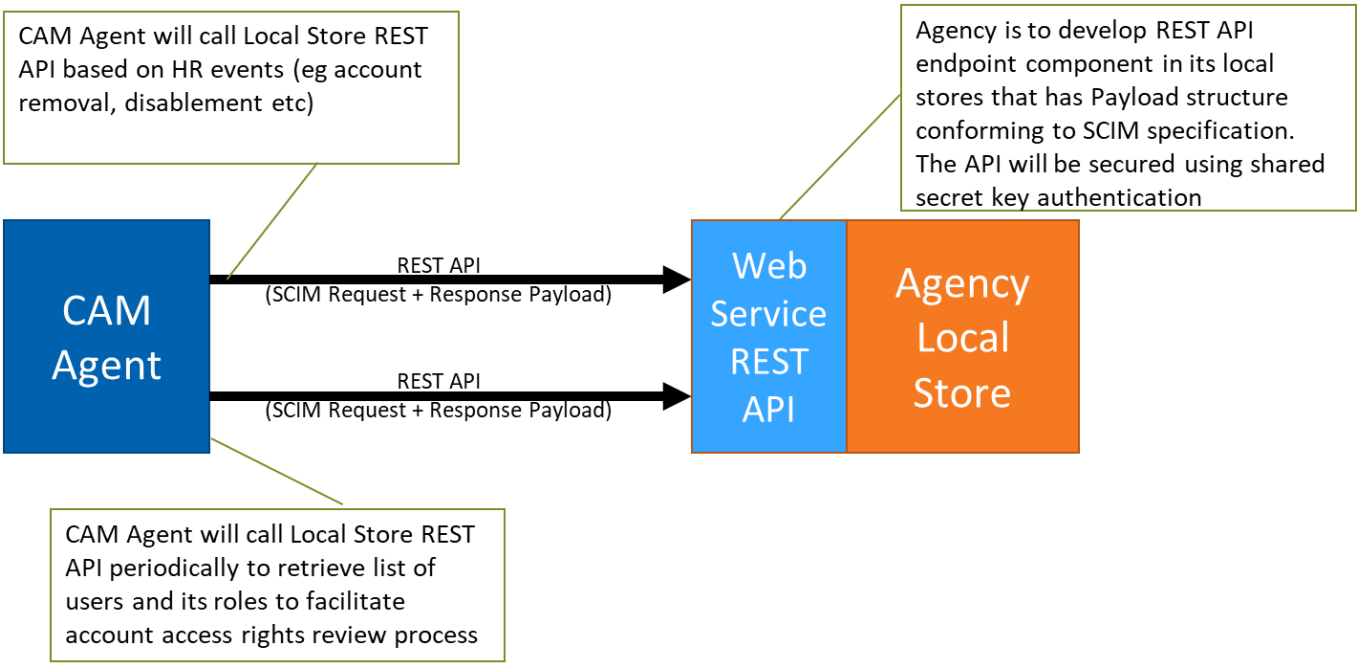
1. Background

In order to integrate with WOG CAM System, agency applications (either bespoke or COTS) will have to follow and implement standardised application interface specified for WOG CAM to manage user accounts and access rights. WOG CAM Agent will be deployed in agency hosting environment to interface with agency applications.



The interface standard will be based on SCIM (System for Cross-domain Identity Management) specification which is described in following IETF.org (Internet Engineering Task Force) pages

- https://tools.ietf.org/html/rfc7642
- https://tools.ietf.org/html/rfc7643
- https://tools.ietf.org/html/rfc7644



Following are the list of events and scenarios that will trigger CAM Agent to call Agency Application's REST API

Event	Scenario Examples	List of API Calls to Agency Application	When it is triggered	Remarks
User Account Retrieval	CAM retrieves detailed user account of a user.	Get User API	Ad-hoc	
User List Retrieval	CAM retrieves list of users with a given filter list for review purpose.	Get User List API	Daily	
User Account Disablement/Enablement	CAM receives notification to disable a user who is inactive for > 90 days or to reinstate a disabled user.	Disable or Enable User	Ad-hoc	
User Account Removal	CAM removes a user upon the user account termination notification.	Remove User API	Ad-hoc	
Group Info Retrieval	CAM retrieves info of a group and list of users attached to it for review purpose.	Get Group API	Ad-hoc	
Group List Retrieval	CAM retrieves list of groups with a given filter list for review purpose.	Get Group List API	Daily	
Group Membership Provisioning/Removal	CAM adds user(s) into or removes user(s) from a group upon an access right provisioning notification.	Add or Remove User from Group	Ad-hoc	For future use
Group Removal	CAM detaches all uses from a group and removes the group upon a group removal notification.	Remove Group API	Ad-hoc	For future use

2. API Specification

API Authentication

How to Authenticate REST API

Every REST API calls initiated by CAM Agent would be encrypted and need to be authenticated by Agency Application.

Following are pre-requisite and information that need to be provided by Agency Application to CAM Agent:

- Agency Application would need to **expose REST API** in **HTTPS** (by acquiring and installing WOG Cert in its Local Store)
- Generate **Account ID** and **Secret Key** for CAM Agent. Account ID and Secret Key are 10-50 alphanumeric characters that are unique for each Agency Application. Different Agency Application should generate different pair of Account ID and Secret Key for CAM Agent

CAM Agent will be calling Agency Application by providing **Authorization header**, **Account ID**, **Nonce** and **Request Timestamp.**

Authorization header will be passed as Bearer HTTP Authorization header request

Account ID, Nonce and **Request Timestamp** will be passed as Request Parameters.

Item	Description
HTTP Authorization Header	Signed Authentication token that will be sent by CAM Agent.
Account ID and **Secret Key**	A pair of 10-50 alphanumeric characters sting that are used by Agency Application system to identify CAM Agent.
Nonce	A random positive 32-bit integer number that is uniquely generated for each API call.
Request Timestamp (TS)	Timestamp of the request express in number of milliseconds since the UNIX epoch (January 1, 1970 00:00:00 UTC).

Agency Application would need to perform following checks for authentication

- Ensure that the signed authentication token sent by CAM Agent through **HTTP Authorization Header** is valid. See below on how to authenticate the HTTP Authorization Header
- Ensure that the **Nonce** parameter is a positive 32-bit integer number. Optionally, Agency Application may choose to ensure the uniqueness of the nonce (up to 24 hours)
- Ensure that the **Request Timestamp (TS)** is within **accepted 3 minutes grace period** (the grace period should be configurable at Agency Application)

How to Authenticate HTTP Authorization Header

Following are the steps to validate HTTPS Authorisation Header:

- Combine HTTP Method, URL and sorted parameters in name value pairs into following structure <http method>&<encoded url>&<sorted parameters as name value pairs>. HTTP method (GET, PUT, PATCH, etc) must be in uppercase and the rest must be in lowercase

  Example:

  ```
  HTTP Request POST https://intranet.app1.agency1.gov.sg/scim/api/users/info?
  accountId=1234567&nonce=09832472134&ts=5678973453
  ```

  Combined string will be:

  ```
  POST&https://intranet.app1.agency1.gov.sg/scim/api/users/info?accountid=1234567&nonce=09832472134&ts=567
  8973453
  ```

- **Hash** the above combined string with **HMAC-SHA256** algorithm
- **Sign** the hashed value using **Secret Key** of the Account ID
- **Encode** the above signature value with **Base64**
- **Compare** the received **HTTP Authorisation Header** with above Base64-**encoded signature**. If it matches, that means the Authorisation Header is valid

Sample Code on How to Authenticate Authorization Header

Following are sample code in .NET and Java on how to perform check on authorization header from request header.

Sample Code in .NET (ASP.Net REST API using .NET Core 3+)

```
private void checkAccountAndAuthorizationHeader(AuthorizationFilterContext context)
{
   Func<string, byte[]> _ = (s) => { return Encoding.UTF8.GetBytes(s); };
   Func<string, int> setUnauthorizedCode = (s) => {
      context.Result = new ContentResult()
      {
         StatusCode = 401,
         Content = JsonConvert.SerializeObject(new {error = "Unauthorised Request", errorCode = s})
      };
      return 0;
   };

   //Get accountID and secretKey
   string acctID = _configuration[SCIM_API_AccountID];
   string secretKey = _configuration[SCIM_API_SecretKey];

   //Check AccountID
   string acctIDq = context.HttpContext.Request.Query["accountid"].ToString();
   if (string.IsNullOrEmpty(acctIDq) || acctIDq != acctID)
```

```
    {
        setUnauthorizedCode("Invalid account ID");
        return;
    }

    //Construct string for authorisation check
    string httpMethod = context.HttpContext.Request.Method.ToUpper();
    string protocol = context.HttpContext.Request.IsHttps ? "https" : "http";
    string hostName = context.HttpContext.Request.Host.Value.ToLower();
    string url = context.HttpContext.Request.Path.ToString();
    string queryStringSorted = context.HttpContext.Request.Query.OrderBy(x => x.Key)
        .Select(x => $"{x.Key}={x.Value}").Aggregate((x, y) => $"{x}&{y}").ToLower();

    string combinedString = $"{httpMethod}&{protocol}://{hostName}{url}?{queryStringSorted}";

    string authString = Convert.ToBase64String(new HMACSHA256(_(secretKey)).ComputeHash(_(combinedString)));

    //Get authorisation header from request
    var authHeader = getAuthorizationHeader(context.HttpContext.Request.Headers);

    if (authHeader != authString)
    {
        setUnauthorizedCode("Invalid authorization header");
        return;
    }
}

private void checkNonce(AuthorizationFilterContext context)
{
    // TODO: to replace this with proper NONCE handling that cater for scalability deployment and housekeeping mechanism
    string nonce = context.HttpContext.Request.Query["nonce"].ToString();
    if (_nonceValidator.checkNonce(nonce))
    {
        return;
    }
    context.Result = new ContentResult()
    {
        StatusCode = 401,
        Content = JsonConvert.SerializeObject(new { error = "Unauthorised Request", errorCode = "Invalid nonce" })
    };
}

private void checkTimeStamp(AuthorizationFilterContext context)
{
    string reqdtq = context.HttpContext.Request.Query["ts"].ToString();
    string gracePeriod = _configuration[SCIM_API_RequestExpiry];
    if (!string.IsNullOrEmpty(reqdtq)) {
        try
        {
            long gracePeriodl = long.Parse(gracePeriod);
            long lreqDt = long.Parse(reqdtq);

            DateTime reqDateTime = new DateTime(1970, 1, 1).AddMilliseconds(lreqDt);
            if (DateTime.Now.Subtract(reqDateTime).TotalSeconds <= gracePeriodl)
            {
                return;
```

```
        }
      } catch (Exception ex)
      {
        // TODO: log the exception
      }
    }

    context.Result = new ContentResult()
    {
      StatusCode = 401,
      Content = JsonConvert.SerializeObject(new { error = "Unauthorised Request", errorCode = $"Invalid request date time" })
    };
}


public void OnAuthorization(AuthorizationFilterContext context)
{
    checkNonce(context);
    if (context.Result == null) checkTimeStamp(context);
    if (context.Result == null) checkAccountAndAuthorizationHeader(context);
}
```

Sample Code in Java (Spring Boot REST API Java 8+)

```
private ErrorResponse checkAccountAndAuthorizationHeader(HttpServletRequest request, Map<String, String> reqHeaders, Map<String,
String> reqParams)  {

    String acctIDq = reqParams.get("accountid");
    if (!accessID.equals(acctIDq))
    {
      return new ErrorResponse(HttpStatus.UNAUTHORIZED, new ErrorResponseMessage("Unauthorised Request","Invalid account
ID"));
    }

    String httpMethod = request.getMethod().toUpperCase();
    String protocol = request.getScheme().toLowerCase();
    String hostName = request.getServerName().toLowerCase();
    String hostPort = request.getServerPort() + "";
    hostName = hostPort == "80" ? hostName : String.format("%s:%s", hostName, hostPort);
    String requestURI = request.getRequestURI().toLowerCase();
    String parameters = constructAndSortParam(reqParams);

    String combinedString = String.format("%s&%s://%s%s?%s", httpMethod, protocol,hostName,requestURI,parameters);

    try {
      SecretKeySpec signingKey = new SecretKeySpec(secretKey.getBytes(), "HMACSHA256");

      Mac mac = Mac.getInstance("HMACSHA256");
      mac.init(signingKey);
      byte[] rawHmac = mac.doFinal(combinedString.getBytes());
      String authString = new String(Base64.getEncoder().encodeToString(rawHmac));

      String authHeader = getAuthorizationHeader(reqHeaders);

      if (!authHeader.equals(authString))
      {
```

```
        return new ErrorResponse(HttpStatus.UNAUTHORIZED, new ErrorResponseMessage("Unauthorised Request","Invalid authorization
header"));
      }
      return null;
    } catch (Exception ex) {
      //TODO Log exception here
      return new ErrorResponse(HttpStatus.UNAUTHORIZED, new ErrorResponseMessage("Unauthorised Request","Invalid authorization
header"));
    }
}

public Object validateActionRequest(final ProceedingJoinPoint joinPoint) throws Throwable {
        Object joinPointResult;

        HttpServletRequest request = ((ServletRequestAttributes) RequestContextHolder
            .getRequestAttributes()).getRequest();

        Map<String, String> reqParams = convertMap(request.getParameterMap());

        ErrorResponse e = checkNonce(reqParams);
        if (e == null) { e = checkTimeStamp(reqParams); }
        if (e == null) { e = checkAccountAndAuthorizationHeader(request, reqParams); }

        if (e != null)
        {
                return new ResponseEntity<ErrorResponseMessage>(e.getMessage(), e.getHttpStatus());
        }
        else
        {
                joinPointResult = joinPoint.proceed();
                return joinPointResult;
        }
}

private ErrorResponse checkNonce(Map<String, String> reqParams) {
      String nonce = reqParams.getOrDefault("nonce", "");
        if (nonceValidator.checkNonce(nonce)) {
                return null;
        }
        return new ErrorResponse(HttpStatus.UNAUTHORIZED, new ErrorResponseMessage("Unauthorised Request","Invalid nonce"));

}

private ErrorResponse checkTimeStamp(Map<String, String> reqParams) {
        try {
                String reqdtq = reqParams.getOrDefault("ts", "").toString();
                if (!reqdtq.equals("") ) {
                        long reqdtl = Long.parseLong(reqdtq);
                        long currentMillisecond = System.currentTimeMillis();
                        if (currentMillisecond - reqdtl <= (requestExpiryGracePeriod * 1000)) {
                                return null;
                        }
                }
        } catch (Exception e) {
                // TODO: Log exception here
        }
        return new ErrorResponse(HttpStatus.UNAUTHORIZED, new ErrorResponseMessage("Unauthorised Request","Invalid request
date time"));
}
```

List of CAM Agent API

Following are list of the API. The API specification is also available in OpenAPI 3.0 format and is available at Appendix section

1. Get User API

Description

API to get user detail info from Local User Store

API Request Element

Item	Value and Description
URL Path	/users/info
Parameters	• accountId • nonce • ts See section API Authentication to find out more on accountId, nonce and ts
Verb	POST
Authentication Scheme	Bearer <HTTP Authorization Header> See section API Authentication to find out more on HTTP Authorization Header

Request Body	<pre>{
	  "userId": "<User ID>"
	}</pre>
	UserID is unique identifier of a user in the user store of agency application.

API Response Element

Response Code and Response Body	Description

200
<pre>{
  "schemas": [
  "urn:ietf:params:scim:schemas:core:2.0:User",
  "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
  "urn:ietf:params:scim:schemas:extension:cam:2.0:User"
  ],
  "id": "<User ID>",
  "externalId": "<External User ID>",
  "meta": {
  "resourceType": "User",
  "created": "<Created Date Time>",
  "lastModified": "<Last Modified Date Time>"
  },
  "userName": "<UserName>",
  "displayName": "<DisplayName>",
  "name": {
  "formatted": "<Formatted Name>",
  "familyName": "<Family Name>",
  "givenName": "<Given Name>"
  },
  "active": <Active Status>,
  "emails": [
  {
  "value": "<Email>",
  "type": "work",
  "primary": <Email Primary Status></pre>
	OK Response
	
	Items marked with (*) are mandatory.
	
	• User ID(*) : Unique Internal User ID in User Store
	• External User ID : Unique External User ID (eg NRIC/FIN No). If it's not available, it should be same as User ID
	• Created Date Time: Created Date Time in UTC as specified in https://tools.ietf.org/html/rfc3339
	• Last Modified Date Time : Last Modified Date Time in UTC as specified in https://tools.ietf.org/html/rfc3339
	• UserName(*) : Unique alias of User ID that is usually user friendly name. If its not available, it should be same as User ID
	• DisplayName : Full name of user in the system
	• Formatted Name : The formatted full name of user in the system

<table>
<tr>
<td>

<pre>
    }
  ],
  "profileUrl": "<Profile URL>",
  "title": "<User Title>",
  "userType": "<User Type>",
  "groups": [
    {
      "value": "<Group ID>",
      "$ref": "<Group URI>",
      "displayName": "<Group Name>"
    }
  ],
  "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User": {
    "organization": "<Agency Code>",
    "division": "<Division>",
    "department": "<Department>",
    "manager": {
      "value": "<RO User ID>",
      "$ref": "<RO User Profile URI>",
      "displayName": "<RO Name>"
    }
  },
  "urn:ietf:params:scim:schemas:extension:cam:2.0:User": {
    "lastLogin": "<Last Login Date Time>",
    "lastPasswordChanged": "<Last Password Changed Date Time>",
    "isPrivileged": <Is Privilege User>
  }
}
</pre>

</td>
<td>

- Family Name : User family name
- Given Name : User given name
- Active Status(*) : User enable / disable status (true or false)

Emails(*) (NOTE: there can be more than 1 set of emails and 1 must be set as primary)

- Email(*) : Work email address.
- Email Primary Status(*): Primary status (true or false)
- Profile URL : The referent URI of user profile. If it's not available, leave it blank
- User Title : User title or designation
- User Type: Type of User (AgencyUser, Temp, Intern, Vendor, Other)

Groups (NOTE: there can be more than 1 set of groups)

- Group ID(*) : Authorisation Group ID where user belongs to
- Group URI : The reference URI of Group profile. If it's not available, leave it blank
- Group Name : The display name of authorisation group

Enterprise

- Agency Code : Agency Code ID
- Division : Division of user
- Department : Department of user
- RO User ID : RO User ID
- RO User Profile URI : The reference URI of RO profile. If it's not available, leave it blank
- RO Name : RO Name

User Extension

- Last Login Date Time : Last login date time of user in UTC as specified in https://tools.ietf.org/html/rfc3339
- Last Password Changed Date Time: Date time where user last changed password in UTC as specified in https://tools.ietf.org/html/rfc3339
- Is Privilege User(*): Is user with privilege access (true or false)

JSON Schema for SCIM to validate the response body:

- user.schema.json
- enterpriseUser.schema.json

</td>
</tr>
</table>

	• extensionCamUser.schema.json

401
{
"schemas": [
"urn:ietf:params:scim:api:messages:2.0:Error"
],
"detail": "<Error Message>",
"status": "401"
}
	Unauthorised request.
	
	Possible value of <Error Message>
	
	• Invalid account
	• Invalid authorisation header
	• Invalid time stamp
	• Invalid nonce
	
	JSON Schema for SCIM to validate the response body:
	
	• RFC 7644 - Error Response Handling

404
{
"schemas": [
"urn:ietf:params:scim:api:messages:2.0:Error"
],
"detail": "<Error Message>",
"status": "404"
}
	User was not found
	
	• <Error Message> : brief description of the error. It may contain an error code that can be used to lookup detail error message / stack trace in application log for troubleshooting purpose
	
	JSON Schema for SCIM to validate the response body:
	
	• RFC 7644 - Error Response Handling

500
{
"schemas": [
"urn:ietf:params:scim:api:messages:2.0:Error"
],
"detail": "<Error Message>",
"status": "500"
}
	Other Type of Error.
	
	• <Error Message> : brief description of the error. It may contain an error code that can be used to lookup detail error message / stack trace in application log for troubleshooting purpose
	
	JSON Schema for SCIM to validate the response body:
	
	• RFC 7644 - Error Response Handling

Example

Request

```
POST /users/info?accountId=ab012312&nonce=92834923&ts=823748237

{
  "userId": "345234523"
}
```

Response

```
HTTP/1.1 200 OK
{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
    "urn:ietf:params:scim:schemas:extension:cam:2.0:User"
  ],
  "id": "345234523",
  "externalId": "S1234567Q",
  "meta": {
    "resourceType": "User",
    "created": "2018-03-27T19:59:26.000Z",
    "lastModified": "2018-03-27T19:59:26.000Z"
  },
  "userName": "john2134",
  "displayName": "John Lee",
  "name": {
    "formatted": "John Lee",
    "familyName": "Lee",
    "givenName": "John"
  },
  "active": true,
  "emails": [
    {
      "value": "John.lee@domain.com",
      "type": "work",
      "primary": true
    }
  ],
  "profileUrl": "",
  "title": "Project Manager",
  "userType": "AgencyUser",
  "groups": [
    {
      "value": "e9e30dba-f08f-4109-8486-d5c6a331660a",
      "$ref": "https://example.com/v2/Groups/e9e30dba-f08f-4109-8486-d5c6a331660a",
      "display": "Doc Approval Group"
    },
    {
      "value": "fc348aa8-3835-40eb-a20b-c726e15c55b5",
      "$ref": "https://example.com/v2/Groups/fc348aa8-3835-40eb-a20b-c726e15c55b5",
      "displayName": "Verifier Group"
    }
  ],
  "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User": {
    "organization": "MOM",
    "division": "WP",
    "department": "WPD",
    "manager": {
      "value": "876987687",
      "$ref": "https://example.com/v2/User/e9e30dba-f08f-4109-8486-d5c6a331660a",
      "displayName": "Thomas Tan"
    }
```

```
  },
  "urn:ietf:params:scim:schemas:extension:cam:2.0:User": {
    "lastLogin": "2019-03-27T19:59:26.000Z",
    "lastPasswordChanged": "2019-03-20T19:59:26.000Z",
    "isPrivileged": false
  }
}
```

2. Get User List API

Description

API to get list of user info from Local User Store.

API Request Element

Item	Value and Description
URL Path	/users/findbycriteria
Parameters	• accountId • nonce • ts See section API Authentication to find out more on accountId, nonce and ts
Verb	POST
Authentication Scheme	Bearer <HTTP Authorization Header> See section API Authentication to find out more on HTTP Authorization Header

Request Body	<pre>{
	  "filter": "<filter criteria>",
	  "startIndex": <startIndex>,
	  "itemsPerPage": <itemsPerPage>,
	  "ascOrderBy": "<ascOrderBy>",
	  "descOrderBy": "<descOrderBy>"
	}</pre>
	The default behaviour is to list all users in local store if there is no filter parameter specified
	• filter : Filtering criteria for the list. It will be either by *groupId* or *groupName.* It must supports operators "eq" and "like" and followed by keyword enclosed in single quote. Example are as follows.
	  • *filter=groupId eq '123'* → retrieve list of users having group id equals to '123'
	  • *filter=groupName like 'ADMIN'* → retrieve list of users having group name containing string 'ADMIN'
	• startIndex : Starting pagination index (starting with 1)
	• itemsPerPage : Number of records returned per API
	• ascOrderBy : To indicate the list should be ordered in ascending by either *groupID, groupName, userId*, or *userName*
	• descOrderBy : To indicate the list should be ordered in descending by either *groupID, groupName, userId*, or *userName*

API Response Element

Response Code and Response Body	Description

200
 	
{
"schemas": [
"urn:ietf:params:scim:api:messages:2.0:ListResponse"
],
"totalResults": <Total Results>,
"Resources": [
{
"schemas": [
"urn:ietf:params:scim:schemas:core:2.0:User",
"urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
"urn:ietf:params:scim:schemas:extension:cam:2.0:User"
],
"id": "<User ID>",
"externalId": "<External User ID>",
"meta": {
"resourceType": "User",
"created": "<Created Date Time>",
"lastModified": "<Last Modified Date Time>"
},
"userName": "<UserName>",
"displayName": "<DisplayName>",
"name": {
"formatted": "<Formatted Name>",
"familyName": "<Family Name>",
"givenName": "<Given Name>"
},
"active": <Active Status>,
"emails": [
{
"value": "<Email>",
"type": "work",
"primary": <Email Primary Status>
}
],
"profileUrl": "<Profile URL>",
"title": "<User Title>",
"userType": "<User Type>",
"groups": [
{
"value": "<Group ID>",
"$ref": "<Group URI>",
"displayName": "<Group Name>"
}
],
"urn:ietf:params:scim:schemas:extension:enterprise:2.0:User": {
	OK Response
	
	Items marked with (*) are mandatory.
	
	• Total Results : Unique Internal User ID in User Store
	• Starting Index Page : The 1-based index of first result
	• No of Items per Page : The number of resources returned in list if pagination/itemsPerPage is defined
	
	<u>List of user resources</u>
	
	• User ID(*) : Unique Internal User ID in User Store
	• External User ID : Unique External User ID (eg NRIC/FIN No). If it's not available, it should be same as User ID
	• Created Date Time: Created Date Time in UTC as specified in https://tools.ietf.org/html/rfc3339
	• Last Modified Date Time : Last Modified Date Time in UTC as specified in https://tools.ietf.org/html/rfc3339
	• UserName(*) : Unique alias of User ID that is usually user friendly name. If its not available, it should be same as User ID
	• DisplayName : Full name of user in the system
	• Formatted Name : The formatted full name of user in the system
	• Family Name : User family name
	• Given Name : User given name
	• Active Status(*) : User enable / disable status (true or false)
	
	<u>Emails(*)</u> (NOTE: there can be more than 1 set of emails and 1 must be set as primary)
	
	• Email(*) : Work email address.
	• Email Primary Status(*): Primary status (true or false)
	• Profile URL : The referent URI of user profile. If it's not available, leave it blank
	• User Title : User title or designation
	• User Type: Type of User (AgencyUser, Temp, Intern, Vendor, Other)

<table>
<tr>
<td>

<pre>
            "organization": "<Agency Code>",
            "division": "<Division>",
            "department": "<Department>",
            "manager": {
               "value": "<RO User ID>",
               "$ref": "<RO User Profile URI>",
               "displayName": "<RO Name>"
            }
         },
         "urn:ietf:params:scim:schemas:extension:cam:2.0:User": {
            "lastLogin": "<Last Login Date Time>",
            "lastPasswordChanged": "<Last Password Changed Date
Time>",
            "isPrivileged": <Is Privilege User>
         }
      }
   ],
   "startIndex": <Starting Index Page>,
   "itemsPerPage": <No of Items per Page>
}
</pre>

</td>
<td>

Groups (NOTE: there can be more than 1 set of groups)

- Group ID(*) : Authorisation Group ID where user belongs to
- Group URI : The reference URI of Group profile. If it's not available, leave it blank
- Group Name : The display name of authorisation group

Enterprise

- Agency Code : Agency Code ID
- Division : Division of user
- Department : Department of user
- RO User ID : RO User ID
- RO User Profile URI : The reference URI of RO profile. If it's not available, leave it blank
- RO Name : RO Name

User Extension

- Last Login Date Time : Last login date time of user in UTC as specified in https://tools.ietf.org/html/rfc3339
- Last Password Changed Date Time: Date time where user last changed password in UTC as specified in https://tools.ietf.org/html/rfc3339
- Is Privilege User(*): Is user with privilege access (true or false)

JSON Schema for SCIM to validate the response body:

- RFC 7644 - List Response
- user.schema.json
- enterpriseUser.schema.json
- extensionCamUser.schema.json

</td>
</tr>
<tr>
<td>

401

<pre>
{
   "schemas": [
      "urn:ietf:params:scim:api:messages:2.0:Error"
   ],
   "detail": "<Error Message>",
   "status": "401"
}
</pre>

</td>
<td>

Unauthorised request.

Possible value of <Error Message>

- Invalid account
- Invalid authorisation header
- Invalid time stamp
- Invalid nonce

JSON Schema for SCIM to validate the response body:

</td>
</tr>
</table>

	• RFC 7644 - Error Response Handling

404
{
"schemas": [
"urn:ietf:params:scim:api:messages:2.0:Error"
],
"detail": "<Error Message>",
"status": "404"
}
	User was not found
	• <Error Message> : brief description of the error. It may contain an error code that can be used to lookup detail error message / stack trace in application log for troubleshooting purpose
	JSON Schema for SCIM to validate the response body:
	• RFC 7644 - Error Response Handling

500
{
"schemas": [
"urn:ietf:params:scim:api:messages:2.0:Error"
],
"detail": "<Error Message>",
"status": "500"
}
	Other Type of Error.
	• <Error Message> : brief description of the error. It may contain an error code that can be used to lookup detail error message / stack trace in application log for troubleshooting purpose
	JSON Schema for SCIM to validate the response body:
	• RFC 7644 - Error Response Handling

Example

Request

```
POST /users/findbycriteria?accountID=AB012312&nonce=92834923&ts=823748237

{
  "filter": "groupName like 'admin'",
  "startIndex": 1,
  "itemsPerPage": 20,
  "ascOrderBy": "userName"
}
```

Response

```
HTTP/1.1 200 OK
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 2,
  "Resources": [
    {
      "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:User",
        "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
        "urn:ietf:params:scim:schemas:extension:cam:2.0:User"
      ],
```

```
    "id": "982734098520735243",
    "externalId": "58342554-38d6-4ec8-948c-50044d0a33fd",
    "meta": {
      "resourceType": "User",
      "created": "2018-03-27T19:59:26.000Z",
      "lastModified": "2018-03-27T19:59:26.000Z"
    },
    "userName": "thomas.lee",
    "displayName": "Thomas Lee",
    "name": {
      "formatted": "Thomas Lee",
      "familyName": "Lee",
      "givenName": "Thomas"
    },
    "active": true,
    "emails": [
      {
        "value": "thomas.lee@domain.com",
        "type": "work",
        "primary": true
      }
    ],
    "title": "Project Manager",
    "userType": "AgencyUser",
    "groups": [
      {
        "value": "e9e30dba-f08f-4109-8486-d5c6a331660a",
        "displayName": "Deployer Admin Group"
      },
      {
        "value": "fc348aa8-3835-40eb-a20b-c726e15c55b5",
        "displayName": "Site Admin Configurator Group"
      }
    ],
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User": {
      "organization": "MOM",
      "division": "WP",
      "department": "WPD",
      "manager": {
        "value": "876987687",
        "$ref": "https://example.com/v2/User/e9e30dba-f08f-4109-8486-d5c6a331660a",
        "displayName": "Thomas Tan"
      }
    },
    "urn:ietf:params:scim:schemas:extension:cam:2.0:User": {
      "lastLogin": "2019-03-27T19:59:26.000Z",
      "lastPasswordChanged": "2019-03-20T19:59:26.000Z",
      "isPrivileged": false
    }
  },
  {
    "schemas": [
      "urn:ietf:params:scim:schemas:core:2.0:User",
      "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
      "urn:ietf:params:scim:schemas:extension:cam:2.0:User"
    ],
    "id": "982734098520735777",
    "externalId": "58342554-38d6-4ec8-948c-50044d0a5678d",
    "meta": {
      "resourceType": "User",
      "created": "2018-03-27T19:59:26.000Z",
      "lastModified": "2018-03-27T19:59:26.000Z"
    },
    "userName": "tan.ahkow",
    "displayName": "Tan Ah Kow",
    "name": {
      "formatted": "Tan Ah Kow",
```

```
            "familyName": "Tan ",
            "givenName": "Ah Kow"
          },
          "active": true,
          "emails": [
            {
              "value": "ah.kow@domain.com",
              "type": "work",
              "primary": true
            }
          ],
          "title": "Project Manager",
          "userType": "AgencyUser",
          "groups": [
            {
              "value": "e9e30dba-f08f-4109-8486-d5c6a331660a",
              "displayName": "Deployer Admin Group"
            },
            {
              "value": "fc348aa8-3835-40eb-a20b-c726e15c55b5",
              "displayName": "Site Admin Configurator Group"
            }
          ],
          "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User": {
            "organization": "MOM",
            "division": "WP",
            "department": "WPD",
            "manager": {
              "value": "876988888",
              "$ref": "https://example.com/v2/User/e9e30dba-f08f-4109-8486-d5c6a331660a",
              "displayName": "Michael Tan"
            }
          },
          "urn:ietf:params:scim:schemas:extension:cam:2.0:User": {
            "lastLogin": "2019-03-27T19:59:26.000Z",
            "lastPasswordChanged": "2019-03-20T19:59:26.000Z",
            "isPrivileged": false
          }
        }
      ],
      "startIndex": 1,
      "itemsPerPage": 20
    }
```


3. Disable or Enable User API

Description

API to disable or enable user in Local User Store

API Request Element

Item	Value and Description
URL Path	/users/update
Parameters	• accountId • nonce • ts See section API Authentication to find out more on accountId, nonce and ts
Verb	POST

Authentication Scheme	Bearer <HTTP Authorization Header> See section API Authentication to find out more on HTTP Authorization Header

Request Body	<pre>{
	  "schemas": [
	  "urn:ietf:params:scim:api:messages:2.0:PatchOp"
	  ],
	  "userId": "<User ID>",
	  "Operations": [
	  {
	  "op": "Replace",
	  "path": "active",
	  "value": <Activity Status>
	  }
	  ]
	}</pre>
	Following are list of fields
	• User ID : unique identifier of a user in the user store of agency application.
	• Activity Status : activity status to be set (true or false)
	NOTE: *Operations* attribute will only contain 1 operation element
	
	JSON Schema for SCIM to validate the request body:
	• patchop.schema.json

API Response Element

Response Code and Response Body	Description

200
 	
<pre>{
  "schemas": [
  "urn:ietf:params:scim:schemas:core:2.0:User",
  "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
  "urn:ietf:params:scim:schemas:extension:cam:2.0:User"
  ],
  "id": "<User ID>",
  "externalId": "<External User ID>",
  "meta": {
  "resourceType": "User",
  "created": "<Created Date Time>",
  "lastModified": "<Last Modified Date Time>"
  },
  "userName": "<UserName>",
  "displayName": "<DisplayName>",
  "name": {
  "formatted": "<Formatted Name>",
  "familyName": "<Family Name>",
  "givenName": "<Given Name>"
  },
  "active": <Active Status>,
  "emails": [
  {
  "value": "<Email>",
  "type": "work",</pre>
	OK Response
	
	Items marked with (*) are mandatory.
	
	• User ID(*) : Unique Internal User ID in User Store
	• External User ID : Unique External User ID (eg NRIC/FIN No). If it's not available, it should be same as User ID
	• Created Date Time: Created Date Time in UTC as specified in https://tools.ietf.org/html/rfc3339
	• Last Modified Date Time : Last Modified Date Time in UTC as specified in https://tools.ietf.org/html/rfc3339
	• UserName(*) : Unique alias of User ID that is usually user friendly name. If its not available, it should be same as User ID
	• DisplayName : Full name of user in the system
	• Formatted Name : The formatted full name of user in the system
	• Family Name : User family name
	• Given Name : User given name

```
        "primary": <Email Primary Status>
      }
    ],
    "profileUrl": "<Profile URL>",
    "title": "<User Title>",
    "userType": "<User Type>",
    "groups": [
      {
        "value": "<Group ID>",
        "$ref": "<Group URI>",
        "displayName": "<Group Name>"
      }
    ],
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User": {
      "organization": "<Agency Code>",
      "division": "<Division>",
      "department": "<Department>",
      "manager": {
        "value": "<RO User ID>",
        "$ref": "<RO User Profile URI>",
        "displayName": "<RO Name>"
      }
    },
    "urn:ietf:params:scim:schemas:core:2.0:extendedUser": {
      "lastLogin": "<Last Login Date Time>",
      "lastPasswordChanged": "<Last Password Changed Date Time>",
      "isPrivileged": <Is Privilege User>
    }
}
```


- Active Status(*) : User enable / disable status (true or false)

Emails(*) (NOTE: there can be more than 1 set of emails and 1 must be set as primary)

- Email(*) : Work email address.
- Email Primary Status(*): Primary status (true or false)
- Profile URL : The referent URI of user profile. If it's not available, leave it blank
- User Title : User title or designation
- User Type: Type of User (AgencyUser, Temp, Intern, Vendor, Other)

Groups (NOTE: there can be more than 1 set of groups)

- Group ID(*) : Authorisation Group ID where user belongs to
- Group URI : The reference URI of Group profile. If it's not available, leave it blank
- Group Name(*) : The display name of authorisation group

Enterprise

- Agency Code : Agency Code ID
- Division : Division of user
- Department : Department of user
- RO User ID : RO User ID
- RO User Profile URI : The reference URI of RO profile. If it's not available, leave it blank
- RO Name : RO Name

Extended User

- Last Login Date Time : Last login date time of user in UTC as specified in https://tools.ietf.org/html/rfc3339
- Last Password Changed Date Time: Date time where user last changed password in UTC as specified in https://tools.ietf.org/html/rfc3339
- Is Privilege User(*): Is user with privilege access (true or false)

JSON Schema for SCIM to validate the response body:

- user.schema.json

	• enterpriseUser.schema.json • extensionCamUser.schema.json

401
{
"schemas": [
"urn:ietf:params:scim:api:messages:2.0:Error"
],
"detail": "<Error Message>",
"status": "401"
}
	Unauthorised request.
	
	Possible value of <Error Message>
	
	• Invalid account
	• Invalid authorisation header
	• Invalid time stamp
	• Invalid nonce
	
	JSON Schema for SCIM to validate the response body:
	
	• RFC 7644 - Error Response Handling

404
{
"schemas": [
"urn:ietf:params:scim:api:messages:2.0:Error"
],
"detail": "<Error Message>",
"status": "404"
}
	User was not found
	
	• <Error Message> : brief description of the error. It may contain an error code that can be used to lookup detail error message / stack trace in application log for troubleshooting purpose
	
	JSON Schema for SCIM to validate the response body:
	
	• RFC 7644 - Error Response Handling

500
{
"schemas": [
"urn:ietf:params:scim:api:messages:2.0:Error"
],
"detail": "<Error Message>",
"status": "500"
}
	Other Type of Error.
	
	• <Error Message> : brief description of the error. It may contain an error code that can be used to lookup detail error message / stack trace in application log for troubleshooting purpose
	
	JSON Schema for SCIM to validate the response body:
	
	• RFC 7644 - Error Response Handling

Example

Request

```
POST /users/update?accountId=AB012312&nonce=92834923&ts=823748237

{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:PatchOp"
  ],
  "userId": "28342342",
  "Operations": [
    {
      "op": "Replace",
      "path": "active",
      "value": false
    }
  ]
}
```

Response

```
HTTP/1.1 200 OK
{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User",
    "urn:ietf:params:scim:schemas:extension:cam:2.0:User"
  ],
  "id": "28342342",
  "externalId": "S1234567Q",
  "meta": {
    "resourceType": "User",
    "created": "2018-03-27T19:59:26.000Z",
    "lastModified": "2018-03-27T19:59:26.000Z"
  },
  "userName": "john2134",
  "displayName": "John Lee",
  "name": {
    "formatted": "John Lee",
    "familyName": "Lee",
    "givenName": "John"
  },
  "active": false,
  "emails": [
    {
      "value": "John.lee@domain.com",
      "type": "work",
      "primary": true
    }
  ],
  "profileUrl": "",
  "title": "Project Manager",
  "userType": "AgencyUser",
  "groups": [
    {
```

```
        "value": "e9e30dba-f08f-4109-8486-d5c6a331660a",
        "$ref": "https://example.com/v2/Groups/e9e30dba-f08f-4109-8486-d5c6a331660a",
        "displayName": "Doc Approval Group"
      },
      {
        "value": "fc348aa8-3835-40eb-a20b-c726e15c55b5",
        "$ref": "https://example.com/v2/Groups/fc348aa8-3835-40eb-a20b-c726e15c55b5",
        "displayName": "Verifier Group"
      }
    ],
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User": {
      "organization": "MOM",
      "division": "WP",
      "department": "WPD",
      "manager": {
        "value": "876987687",
        "$ref": "https://example.com/v2/User/e9e30dba-f08f-4109-8486-d5c6a331660a",
        "displayName": "Thomas Tan"
      }
    },
    "urn:ietf:params:scim:schemas:extension:cam:2.0:User": {
      "lastLogin": "2019-03-27T19:59:26.000Z",
      "lastPasswordChanged": "2019-03-20T19:59:26.000Z",
      "isPrivileged": false
    }
}
```


4. Remove User API

Description

API Request Element

Item	Value and Description
URL Path	/users/remove
Parameters	• accountId • nonce • ts See section API Authentication to find out more on accountId, nonce and ts
Verb	POST
Authentication Scheme	Bearer <HTTP Authorization Header> See section API Authentication to find out more on HTTP Authorization Header

Request Body	{
	"userId": "<User ID>"
	}
	
	UserID : unique identifier of a user in the user store of agency application.

API Response Element

Response Code and Response Body	Description
204	OK - No Content

401
{
"schemas": [
"urn:ietf:params:scim:api:messages:2.0:Error"
],
"detail": "<Error Message>",
"status": "401"
}
	Unauthorised request.
	
	Possible value of <Error Message>
	• Invalid account
	• Invalid authorisation header
	• Invalid time stamp
	• Invalid nonce
	
	JSON Schema for SCIM to validate the response body:
	• RFC 7644 - Error Response Handling

404
{
"schemas": [
"urn:ietf:params:scim:api:messages:2.0:Error"
],
"detail": "<Error Message>",
"status": "404"
}
	User was not found
	• <Error Message> : brief description of the error. It may contain an error code that can be used to lookup detail error message / stack trace in application log for troubleshooting purpose
	
	JSON Schema for SCIM to validate the response body:
	• RFC 7644 - Error Response Handling

500
{
"schemas": [
"urn:ietf:params:scim:api:messages:2.0:Error"
],
"detail": "<Error Message>",
"status": "500"
}
	Other Type of Error.
	• <Error Message> : brief description of the error. It may contain an error code that can be used to lookup detail error message / stack trace in application log for troubleshooting purpose
	
	JSON Schema for SCIM to validate the response body:
	• RFC 7644 - Error Response Handling

Example

Request

```
POST /users/remove?accountId=AB012312&nonce=92834923&ts=823748237

{
  "userId": "user123123"
}
```

Response

```
HTTP/1.1 204 OK
```

5. Get Group API

Description

API to get user authorisation group info from Local User Store

API Request Element

Item	Value and Description
URL Path	/groups/info
Parameters	• accountId • nonce • ts See section API Authentication to find out more on accountId, nonce and ts
Verb	POST
Authentication Scheme	Bearer <HTTP Authorization Header> See section API Authentication to find out more on HTTP Authorization Header

Request Body	<pre>{
	"groupId": "<Group ID>"
	}</pre>
	GroupID is unique identifier of a group in the User Store.

API Response Element

Response Code and Response Body	Description

200
 	
<pre>{
"schemas": [
"urn:ietf:params:scim:schemas:core:2.0:Group",
"urn:ietf:params:scim:schemas:extension:cam:2.0:Group"
],
"id": "<GroupID>",
"externalId": "<External Group ID>",
"meta": {
"resourceType": "Group",
"created": "<Created Date Time>",
"lastModified": "<Last Modified Date Time>"
},
"displayName": "<DisplayName>",
"members": [
{
"value": "<User ID>",
"$ref": "<User URI>",</pre>
	OK Response
	
	Items marked with (*) are mandatory.
	
	• Group ID(*) : Unique Internal Group ID in User Store
	• External Group ID : Unique External Group ID. If it's not available, it should be same as User ID
	• Created Date Time: Created Date Time in UTC as specified in https://tools.ietf.org/html/rfc3339
	• Last Modified Date Time : Last Modified Date Time in UTC as specified in https://tools.ietf.org/html/rfc3339
	• Display Name(*): Full name of group in the system
	• User ID(*) : User ID that is a member of the group

<table>
<tr>
<td>

<pre>
        "type": "User"
      },
      {
        "value": "<User ID>",
        "$ref": "<User URI>",
        "type": "User"
      }
    ],
    "urn:ietf:params:scim:schemas:extension:cam:2.0:Group": {
      "groupAccessRightInfo": "<GroupAccessRightInfo>"
    }
}
</pre>

</td>
<td>

- User URI : The reference URI of User profile. If it's not available, leave it blank

Extended Group

- GroupAccessRightInfo: Information of access rights such as roles, data and functions that this group is entitled to

JSON Schema for SCIM to validate the response body:

- group.schema.json
- extensionCamGroup.schema.json

</td>
</tr>
<tr>
<td>

401

<pre>
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:Error"
  ],
  "detail": "<Error Message>",
  "status": "401"
}
</pre>

</td>
<td>

Unauthorised request.

Possible value of <Error Message>

- Invalid account
- Invalid authorisation header
- Invalid time stamp
- Invalid nonce

JSON Schema for SCIM to validate the response body:

- RFC 7644 - Error Response Handling

</td>
</tr>
<tr>
<td>

404

<pre>
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:Error"
  ],
  "detail": "<Error Message>",
  "status": "404"
}
</pre>

</td>
<td>

User was not found

- <Error Message> : brief description of the error. It may contain an error code that can be used to lookup detail error message / stack trace in application log for troubleshooting purpose

JSON Schema for SCIM to validate the response body:

- RFC 7644 - Error Response Handling

</td>
</tr>
<tr>
<td>

500

<pre>
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:Error"
  ],
  "detail": "<Error Message>",
  "status": "500"
}
</pre>

</td>
<td>

Other Type of Error.

- <Error Message> : brief description of the error. It may contain an error code that can be used to lookup detail error message / stack trace in application log for troubleshooting purpose

JSON Schema for SCIM to validate the response body:

- RFC 7644 - Error Response Handling

</td>
</tr>
</table>

Example

Request

```
POST /groups/info?accountId=AB012312&nonce=92834923&ts=823748237

{
   "groupId": "40734ae655284ad3abcc"
}
```

Response

```
HTTP/1.1 200 OK
{
   "schemas": [
      "urn:ietf:params:scim:schemas:core:2.0:Group",
      "urn:ietf:params:scim:schemas:extension:cam:2.0:Group"
   ],
   "id": "40734ae655284ad3abcc",
   "externalId": "60f1bb27-2e1e-402d-bcc4-ec999564a194",
   "meta": {
      "resourceType": "Group",
      "created": "2018-03-27T19:59:26.000Z",
      "lastModified": "2018-03-27T19:59:26.000Z"
   },
   "displayName": "Administrator Group",
   "members": [
      {
         "value": "e9e30dba-f08f-4109-8486-d5c6a331660a",
         "$ref": "https://app1.domain.gov.sg/UserProfile/e9e30dba-f08f-4109-8486-d5c6a331660a",
         "display": "Thomas Lee"
      },
      {
         "value": "e9e30dba-f08f-4109-8486-d5c6a331660a",
         "$ref": "https://app1.domain.gov.sg/UserProfile/e9e30dba-f08f-4109-8486-d5c6a331660a",
         "display": "John Tan"
      }
   ],
   "urn:ietf:params:scim:schemas:extension:cam:2.0:Group": {
      "groupAccessRightInfo": "AdminRole=CreateUser,EditUser,DeleteUser,RetrieveUSer; OperatorRole=CreateBackup,DeleteBackup"
   }
}
```

6. Get Group List API

Description

API Request Element

Item	Value and Description
URL Path	/groups/findbycriteria
Parameters	• accountId • nonce • ts See section API Authentication to find out more on accountId, nonce and ts
Verb	POST
Authentication Scheme	Bearer <HTTP Authorization Header> See section API Authentication to find out more on HTTP Authorization Header

Request Body	<pre>{
	"filter": "<filter criteria>",
	"startIndex": <startIndex>,
	"itemsPerPage": <itemsPerPage>,
	"ascOrderBy": "<ascOrderBy>",
	"descOrderBy": "<descOrderBy>"
	}</pre>
	The default behaviour is to list all groups in local store if there is no filter parameter specified
	• filter : Filtering criteria for the list. It will be either by *groupName.* It must supports operators "eq" and "like" and followed by keyword enclosed in single quote. Example are as follows.
	  • *filter=groupName like 'ADMIN'* → retrieve list of users having group name containing string 'ADMIN'
	• startIndex : Starting pagination index (starting with 1)
	• itemsPerPage : Number of records returned per API
	• ascOrderBy : To indicate the list should be ordered in ascending by either *groupID* or *groupName*
	• descOrderBy : To indicate the list should be ordered in descending by either *groupID* or *groupName*

API Response Element

Response Code and Response Body	Description

200
 	
<pre>{
"schemas": [
"urn:ietf:params:scim:api:messages:2.0:ListResponse"
],
"totalResults": <Total Result>,
"Resources": [
{
"schemas": [
"urn:ietf:params:scim:schemas:core:2.0:Group",
"urn:ietf:params:scim:schemas:extension:cam:2.0:Group"</pre>
	OK Response
	
	Items marked with (*) are mandatory.
	
	• Total Results : Unique Internal User ID in User Store
	• Starting Index Page : The 1-based index of first result
	• No of Items per Page : The number of resources returned in list if pagination/itemsPerPage is defined

<table>
<tr>
<td>

<pre>
        ],
        "id": "<Group ID>",
        "externalId": "<External Group ID>",
        "meta": {
          "resourceType": "Group",
          "created": "<Created Date Time>",
          "lastModified": "<Last Modified Date Time>"
        },
        "displayName": "<Display Name>",
        "members": [],
        "urn:ietf:params:scim:schemas:extension:cam:2.0:Group": {
          "groupAccessRightInfo": "<GroupAccessRightInfo>"
        }
      }
  ],
  "startIndex": <Start Index Page>,
  "itemsPerPage": <No of Items per Page>
}
</pre>

</td>
<td>

List of group resources

- Group ID(*) : Unique Internal Group ID in User Store
- External Group ID : Unique External Group ID. If it's not available, it should be same as User ID
- Created Date Time: Created Date Time in UTC as specified in https://tools.ietf.org/html/rfc3339
- Last Modified Date Time : Last Modified Date Time in UTC as specified in https://tools.ietf.org/html/rfc3339
- Display Name(*): Full name of group in the system

Extended Group

- GroupAccessRightInfo: Information of access rights such as roles, data and functions that this group is entitled to

JSON Schema for SCIM to validate the response body:

- RFC 7644 - List Response
- group.schema.json
- extensionCamGroup.schema.json

</td>
</tr>
<tr>
<td>

401

<pre>
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:Error"
  ],
  "detail": "<Error Message>",
  "status": "401"
}
</pre>

</td>
<td>

Unauthorised request.

Possible value of <Error Message>

- Invalid account
- Invalid authorisation header
- Invalid time stamp
- Invalid nonce

JSON Schema for SCIM to validate the response body:

- RFC 7644 - Error Response Handling

</td>
</tr>
<tr>
<td>

404

<pre>
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:Error"
  ],
  "detail": "<Error Message>",
  "status": "404"
}
</pre>

</td>
<td>

User was not found

- <Error Message> : brief description of the error. It may contain an error code that can be used to lookup detail error message / stack trace in application log for troubleshooting purpose

</td>
</tr>
</table>

	JSON Schema for SCIM to validate the response body: • RFC 7644 - Error Response Handling

500
 	
{
"schemas": [
"urn:ietf:params:scim:api:messages:2.0:Error"
],
"detail": "<Error Message>",
"status": "500"
}
	Other Type of Error.
	
	• <Error Message> : brief description of the error. It may contain an error code that can be used to lookup detail error message / stack trace in application log for troubleshooting purpose
	
	JSON Schema for SCIM to validate the response body:
	
	• RFC 7644 - Error Response Handling

Example

Request

```
POST /groups/findbycriteria?accountID=AB012312&nonce=92834923&ts=823748237

{
  "filter": "groupName like 'admin'",
  "startIndex": 1,
  "itemsPerPage": 20,
  "ascOrderBy": "groupName"
}
```

Response

```
HTTP/1.1 200 OK
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "totalResults": 2,
  "Resources": [
    {
      "id": "40734ae655284ad3abcc",
      "externalId": "60f1bb27-2e1e-402d-bcc4-ec999564a194",
      "meta": {
        "resourceType": "Group",
        "created": "2018-03-27T19:59:26.000Z",
        "lastModified": "2018-03-27T19:59:26.000Z"
      },
      "members": [],
      "displayName": "backup admin group",
      "urn:ietf:params:scim:schemas:extension:cam:2.0:Group": {
        "groupAccessRightInfo": "AdminRole=CreateUser,EditUser,DeleteUser,RetrieveUSer; OperatorRole=CreateBackup,DeleteBackup"
```

```
        }
      },
      {
        "id": "40734ae655284ad3abee",
        "externalId": "60f1bb27-2e1e-402d-bcc4-ec999564a194",
        "meta": {
          "resourceType": "Group",
          "created": "2018-03-27T19:59:26.000Z",
          "lastModified": "2018-03-27T19:59:26.000Z"
        },
        "members": [],
        "displayName": "super admin group",
        "urn:ietf:params:scim:schemas:extension:cam:2.0:Group": {
          "groupAccessRightInfo": "AdminRole=CreateUser,EditUser,DeleteUser,RetrieveUSer; OperatorRole=CreateBackup,DeleteBackup"
        }
      }
    ],
    "startIndex": 1,
    "itemsPerPage": 20
}
```


7. Add or Remove User from Group API

Description

API Request Element

Item	Value and Description
URL Path	/groups/update
Parameters	• accountId • nonce • ts See section API Authentication to find out more on accountId, nonce and ts
Verb	POST
Authentication Scheme	Bearer <HTTP Authorization Header> See section API Authentication to find out more on HTTP Authorization Header

Request Body	{
	"schemas": [
	"urn:ietf:params:scim:api:messages:2.0:PatchOp"
	],
	"groupId": "<Group ID>",
	"Operations": [
	{
	"op": "<Ops Type>",
	"path": "members",
	"value": "<User ID>"
	}
	]
	}

	Following are list of fields • Ops Type: Add or Remove • Group ID : unique identifier of a group in the User Store • User ID : User ID to be added or removed from the group NOTE: *Operations* attribute will only contain 1 operation element JSON Schema for SCIM to validate the request body: • patchop.schema.json

API Response Element

Response Code and Response Body	Description
204	OK – No Content

401
{
"schemas": [
"urn:ietf:params:scim:api:messages:2.0:Error"
],
"detail": "<Error Message>",
"status": "401"
}
	Unauthorised request.
	
	Possible value of <Error Message>
	
	• Invalid account
	• Invalid authorisation header
	• Invalid time stamp
	• Invalid nonce
	
	JSON Schema for SCIM to validate the response body:
	
	• RFC 7644 - Error Response Handling

404
{
"schemas": [
"urn:ietf:params:scim:api:messages:2.0:Error"
],
"detail": "<Error Message>",
"status": "404"
}
	User was not found
	
	• <Error Message> : brief description of the error. It may contain an error code that can be used to lookup detail error message / stack trace in application log for troubleshooting purpose
	
	JSON Schema for SCIM to validate the response body:
	
	• RFC 7644 - Error Response Handling

500
{
"schemas": [
"urn:ietf:params:scim:api:messages:2.0:Error"
],
"detail": "<Error Message>",
"status": "500"
}
	Other Type of Error.
	
	• <Error Message> : brief description of the error. It may contain an error code that can be used to lookup detail error message / stack trace in application log for troubleshooting purpose

	JSON Schema for SCIM to validate the response body: • RFC 7644 - Error Response Handling

Example

Request

```
POST /groups/update?accountId=AB012312&nonce=92834923&ts=823748237

{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:PatchOp"
  ],
  "groupId": "28342342",
  "Operations": [
    {
      "op": "Add",
      "path": "members",
      "value": "878273649324"
    }
  ]
}
```

Response

```
HTTP/1.1 204 OK
```

8. Remove Group API

Description

API Request Element

Item	Value and Description
URL Path	
Parameters	• accountId • nonce • ts See section API Authentication to find out more on accountId, nonce and ts
Verb	POST
Authentication Scheme	Bearer <HTTP Authorization Header> See section API Authentication to find out more on HTTP Authorization Header

Request Body	<pre>{
	  "groupId": "<Group ID>"
	}</pre>
	Group ID : unique identifier of a group in the User Store.

API Response Element

Response Code and Response Body	Description
204	OK – No Content

401
<pre>{
  "schemas": [
  "urn:ietf:params:scim:api:messages:2.0:Error"
  ],
  "detail": "<Error Message>",
  "status": "401"
}</pre>
	Unauthorised request.
	
	Possible value of <Error Message>
	
	• Invalid account
	• Invalid authorisation header
	• Invalid time stamp
	• Invalid nonce
	
	JSON Schema for SCIM to validate the response body:
	
	• RFC 7644 - Error Response Handling

404
<pre>{
  "schemas": [
  "urn:ietf:params:scim:api:messages:2.0:Error"
  ],
  "detail": "<Error Message>",
  "status": "404"
}</pre>
	User was not found
	
	• <Error Message> : brief description of the error. It may contain an error code that can be used to lookup detail error message / stack trace in application log for troubleshooting purpose
	
	JSON Schema for SCIM to validate the response body:
	
	• RFC 7644 - Error Response Handling

500
<pre>{
  "schemas": [
  "urn:ietf:params:scim:api:messages:2.0:Error"
  ],
  "detail": "<Error Message>",
  "status": "500"
}</pre>
	Other Type of Error.
	
	• <Error Message> : brief description of the error. It may contain an error code that can be used to lookup detail error message / stack trace in application log for troubleshooting purpose
	
	JSON Schema for SCIM to validate the response body:
	
	• RFC 7644 - Error Response Handling

Example

Request

```
POST /groups/remove?accountId=AB012312&nonce=92834923&ts=823748237
```

```
{
   "groupId": "group123"
}
```

Response

```
HTTP/1.1 204 OK
```

3. Tools and Sample Code

Postman API Client Script

Postman API Client Script has been provided in order to facilitate the development of WOG CAM API interface in agency applications. The script is developed with Postman v8.0, exported & saved in Postman Collection v2.1 format.

The script includes the followings:

- Sample API request use cases of all scenarios.
- Built in API authentication mechanism for every API request (auto generated nonce, request timestamp and authorization header).
- Validation of request and response payload against JSON schema.
    - Schemas are conforming to JSON Schema Draft-07 (http://JSON Schema Draft-07).
    - Please take note some of the schemas are combined for ease of validation implementation. This is transparent to the API development.

The script is developed with Postman v8.0, exported & saved in Postman Collection v2.1 format.

Following are steps to run the script:

- Import the script into Postman.
- Select CamScimPostmanScript collection.
    - Select Pre-request Script and update protocol, host and port details of targeted test URL.
- Select an API request to test.
- Examine the response and console.

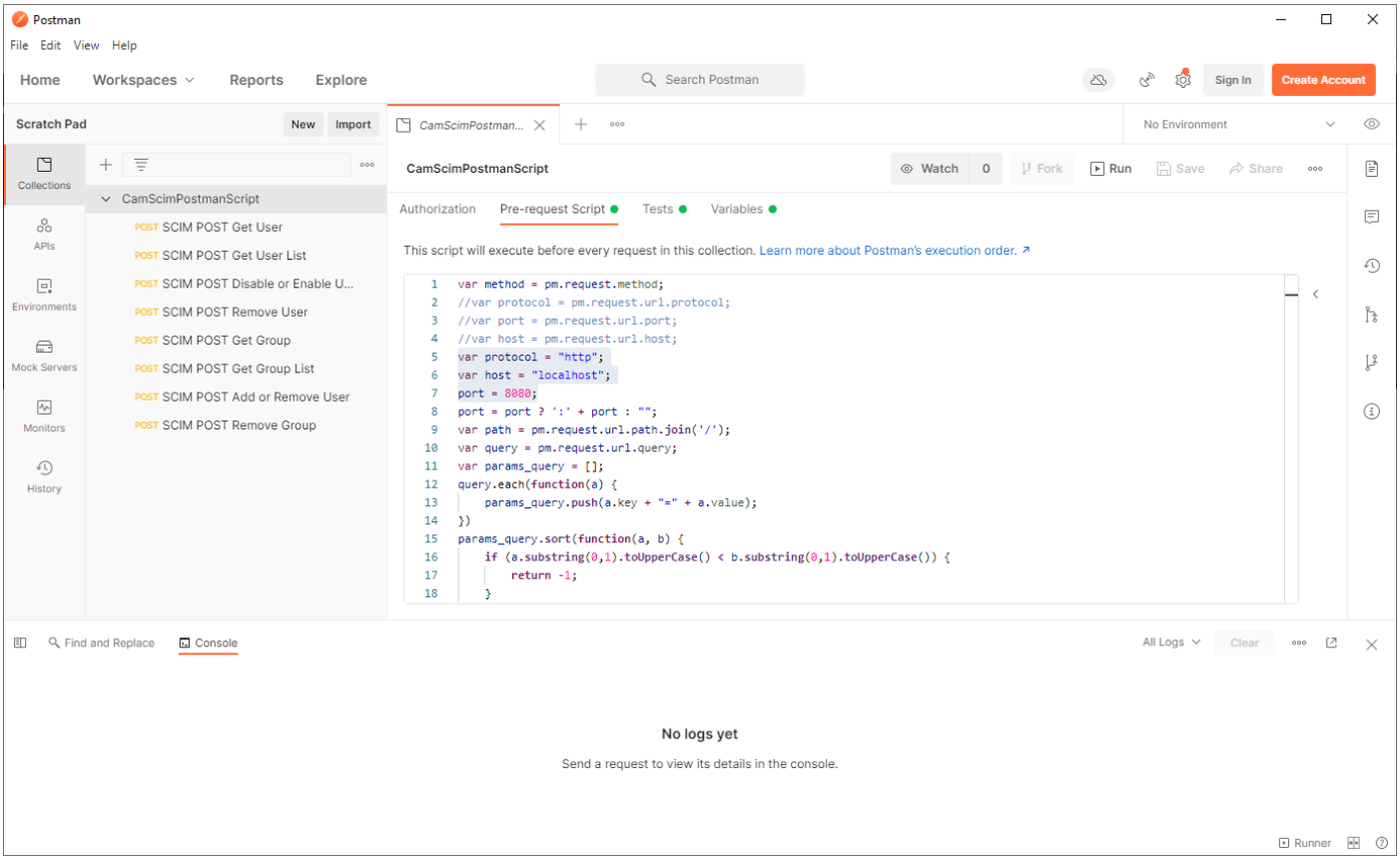


Fig 1. Screenshot of Postman API Client script: select Pre-request Script and update protocol, host and port details of targeted test URL.

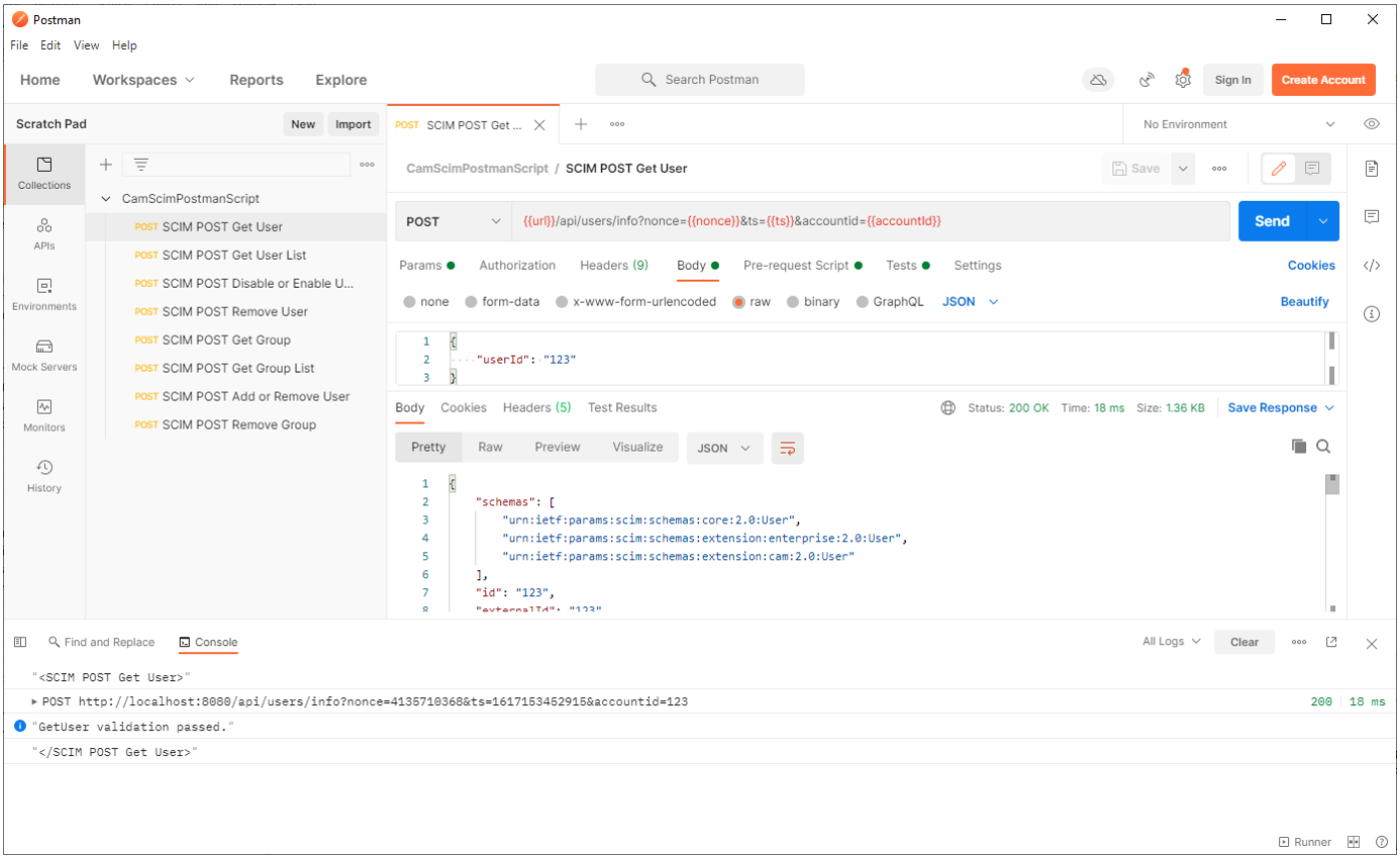


Fig 2. Screenshot of Postman API Client script: response detail and validation result

Sample Codes

Sample REST API application code in .NET and Java have been provided as starter kit to facilitate the development of WOG CAM API in agency applications. This application code can be run as a standalone app accessing the local user store (such as via database) or it can be an application module that is part of agency application.

The sample application code has all REST API interfaces and has ready-to-use module to authenticate API request based on API authentication mechanism specified in earlier section of this documents

.NET Sample codes

Following are the specifications

- Written in C# .NET Core 3.0 and Visual Studio 2019
- Can be configured to run in IIS or as a stand alone web server
- Project structed is based on Domain Driven Design
- It has domain object model and mock services for all API interface

Steps to run the sample codes

- Extract the .NET sample codes into a folder
- Run Visual Studio 2019 and open VS solution file (GT.CAM.SCIM.WebApp.Sln)
- Open Program.cs and uncomment the relevant section for your deployment environment (eg IIS or Kestrel)
- Compile and publish.
    - NOTE: to run ASP.NET core in IIS, configure app pool to be ‘No Managed Code’ and set hostingModel attribute in web.config to be “OutOfProcess”

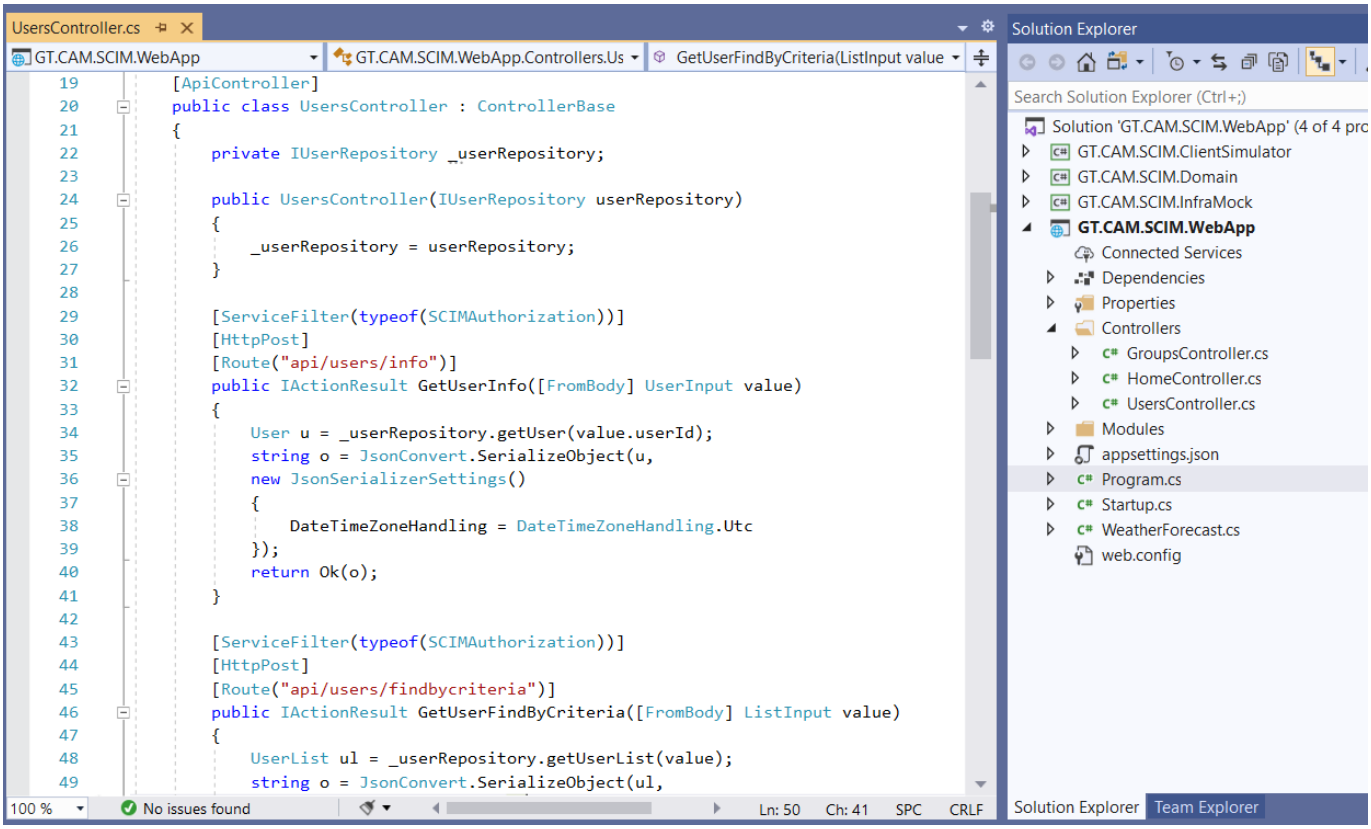


Fig 3. ASP.NET Core API Interface Sample Codes

Java Sample codes

Following are the specifications

- Written in Java 11 and Spring Boot framework
- Customised for Eclipse IDE and built using Maven
- Project structed is based on Domain Driven Design
- It has domain object model and mock services for all API interface

Steps to run the sample codes are as follows.

- Extract Java sample codes into a folder
- Run Eclipse IDE JEE. Ensure that you have installed Spring plugin for Eclipse (download from https://marketplace.eclipse.org/content/spring-tool-suite-sts-eclipse)
- Import the Java codes as Maven project
- Configure Maven Build with Goals ‘spring-boot:run’ in Eclipse to run the application on built-in Spring Boot Tomcat web server
- Configure Spring-Boot App Run with main type set to ‘com.sao.scim.web.MainApplication’ in Eclipse

Note that above steps are meant for Eclipse IDE and Maven. It is possible to use other tools to run the code (e.g. IntelliJ IDE and Gradle build) and it would need to configure the IDE and build settings separately.

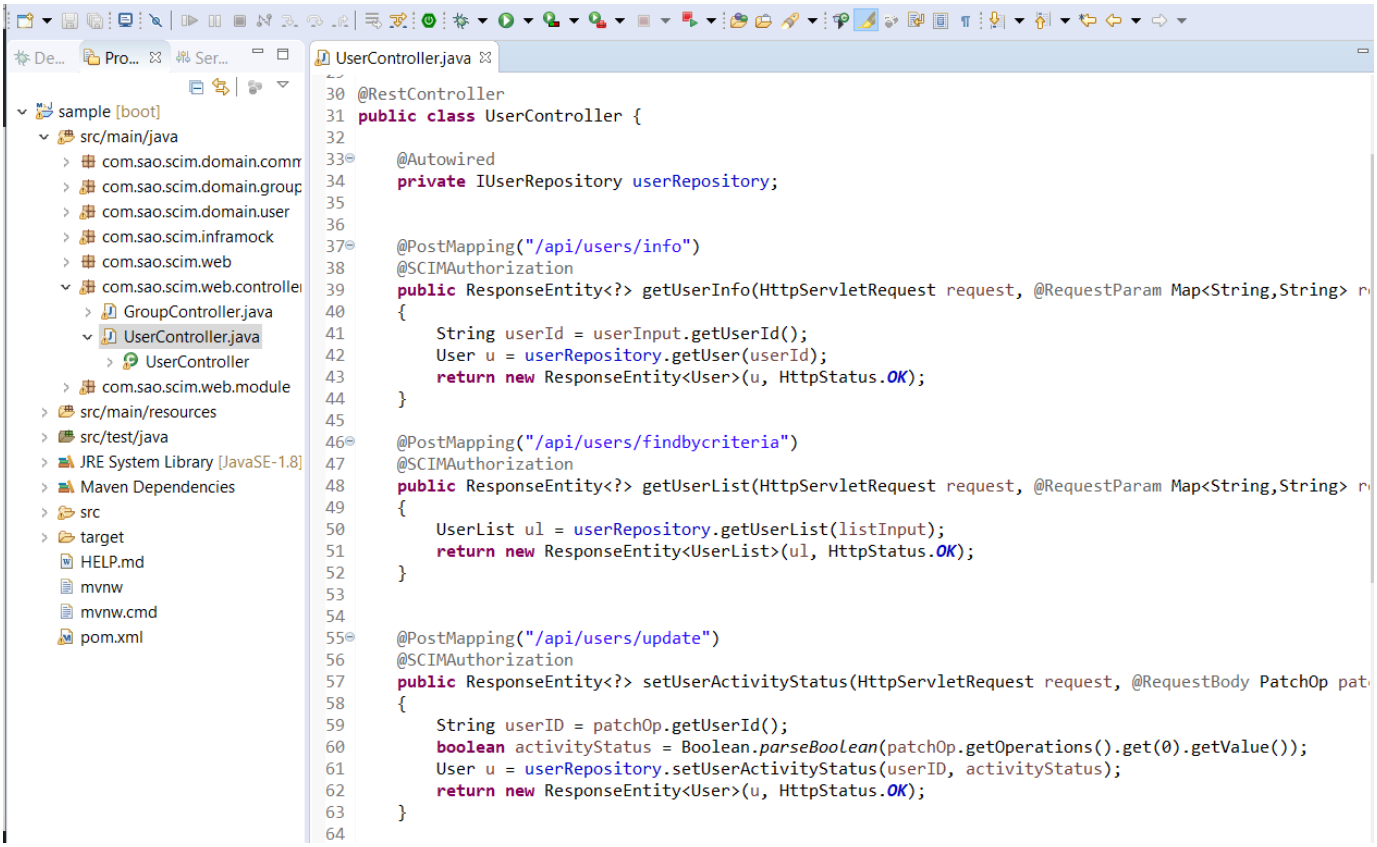


Fig 4. Java Spring Boot API Interface Sample Code