



# Zadara Cloud Services - Identity and Access User Guide

Release 24.03

**Zadara**

Apr 27, 2025



# CONTENTS

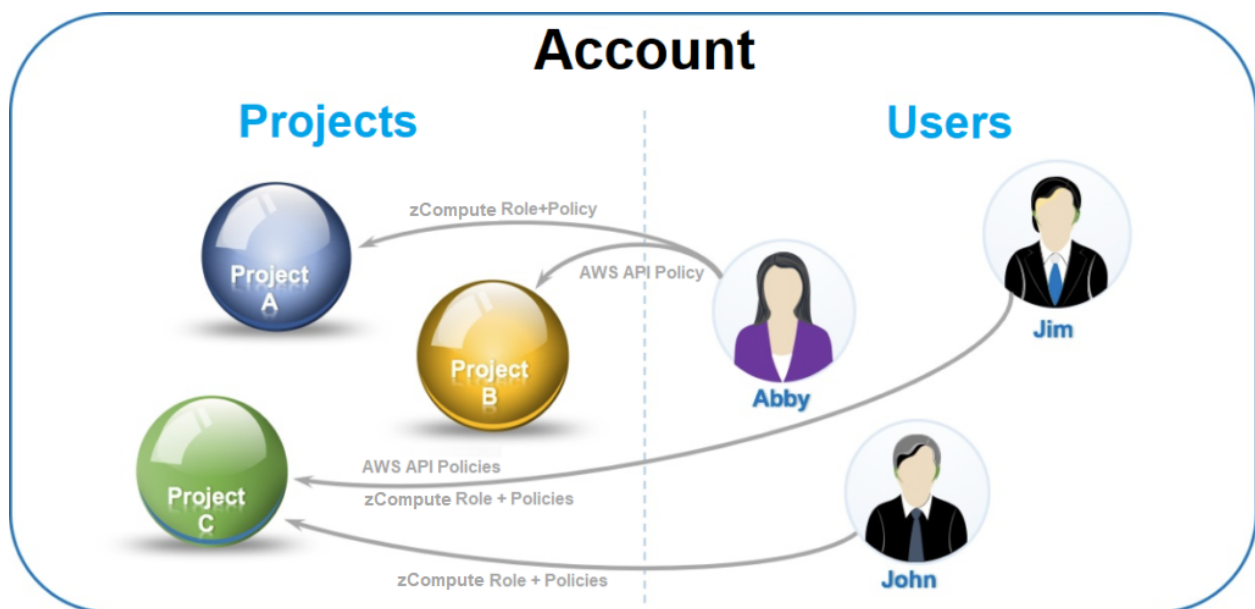
<b>1</b>	<b>Introduction to Identity and Access</b>	<b>1</b>
1.1	Accounts and Projects	1
1.2	Identity Provider (IdP)	2
1.3	Roles and Policies	2
<b>2</b>	<b>Accounts</b>	<b>5</b>
2.1	Overview	5
2.2	Basic Account Operations	5
2.3	Account Limits	6
<b>3</b>	<b>Projects</b>	<b>9</b>
3.1	Creating Projects	9
3.2	Enabling or Disabling Projects	9
3.3	Renaming Projects	10
3.4	Deleting Projects	10
3.5	Assigning a User to a Project	10
3.6	Project Limits	11
<b>4</b>	<b>Users</b>	<b>15</b>
4.1	Creating Users	15
4.2	Deleting Users	17
4.3	Managing Users Permissions	17
4.4	Modifying Users	18
4.5	Resetting User Passwords	19
<b>5</b>	<b>Connecting an Account to a Microsoft Active Directory Identity Provider</b>	<b>21</b>
5.1	Connecting an Account to an Active Directory Identity Provider	21
5.2	Viewing LDAP Groups	27
5.3	Viewing LDAP Users	28
5.4	Assigning zCompute Roles and permissions to an Active Directory User - UI	29
5.5	Assigning zCompute Roles and permissions to an Active Directory User - CLI	30
5.6	LDAP User Sign-on to zCompute	32
<b>6</b>	<b>Authentication using zCompute APIs</b>	<b>35</b>
6.1	API Endpoints	35
6.2	API Explorer	36
6.3	Generating a token at a bash prompt	40
6.4	Generating a token in Symp CLI	43
6.5	Python authentication example	44
<b>7</b>	<b>AWS API Policies</b>	<b>51</b>
7.1	Introduction	51

7.2	AWS IAM API Policies and AWS Roles Overview . . . . .	51
7.3	Managed AWS API Policies Supported by Zadara-IaaS . . . . .	53
7.4	Working with Managed AWS API Policies . . . . .	59
<b>8</b>	<b>AWS IAM Roles and Instance Profiles</b>	<b>61</b>
8.1	AWS IAM Roles . . . . .	61
8.2	Instance Profiles . . . . .	64
<b>9</b>	<b>Zadara Cloud Services Policies</b>	<b>69</b>
9.1	Overview . . . . .	69
9.2	Working with Zadara Cloud Services Policies . . . . .	69
<b>10</b>	<b>AWS Identity-and-Access Management (IAM)</b>	<b>79</b>
10.1	AWS-STS . . . . .	80

## INTRODUCTION TO IDENTITY AND ACCESS

### 1.1 Accounts and Projects

The virtual resources of the Zadara Cloud Services region are managed through an administrative hierarchy of accounts and projects as shown in figure below. The Zadara Cloud Services region consists of one or more accounts, each of which contains one or more projects. Virtual resources such as VLANs, instances, volumes, images and snapshots are created per project, account or region. Users or User Groups, which are members of an account, can be assigned different projects within their account through a Zadara Cloud Services role, together with one or more Zadara Cloud Services and AWS API policies. This assignment enables you to provide access to Zadara Cloud Services to numerous users or groups, while dividing and separating the resources that each user will be able to view, create, and manage.



See the video on the basics of zCompute Identity and Access:

## 1.2 Identity Provider (IdP)

Zadara Cloud Services users and groups can be accessed from an identity provider such as MS Active Directory. This is done by first connecting the Zadara Cloud Services account to the MS Active Directory. Connection with the identity provider must be configured independently for each Zadara Cloud Services account.

See [Connecting an Account to a Microsoft Active Directory Identity Provider](#) for detailed steps.

Once an account becomes connected to Active Directory, the newly available Active Directory users and groups must be assigned Zadara Cloud Services Project Roles and Policies. This can be done either through CLI commands or via the GUI.

Each of these users will now be able to login to Zadara Cloud Services with their Active Directory name and password.

If an account is not connected to an identity provider, users can be manually created within Zadara Cloud Services as members of this account.

---

✓ **Note:** An account with an IdP configured should have no local member users, because keeping local member users:

- Bypasses policies enforced by the IdP
- Defeats some of the purposes of using such an identity provider

Once an IdP is configured, changes cannot be applied to local users.

It is recommended to create and configure a “break-the-glass” local admin user before configuring an IdP.

---

## 1.3 Roles and Policies

Access to Zadara Cloud Services functionality is gained by assigning the user to one or more projects within this account. The user is then assigned policies and roles per project which are used as follows:

1. **Policies** define the ZCS and AWS functionality or APIs, which are being permitted for the user. Policies essentially determine **what** functionality is permitted. Multiple policies can be assigned.
2. **Roles** determines which of those policies or APIs can actually be assigned or used, by a user. Both ZCS and AWS roles are used. A user must be assigned a single ZCS role. In addition, a user may be associated with one or more AWS roles. Roles essentially determine the type of user **who** is permitted to perform this functionality.

### 1.3.1 ZCS Roles

There are three Zadara Cloud Services roles, each allowing different functionality, as follows:

1. **Member** - This role allows the user to use policies and APIs for creating, viewing, modifying and deleting virtual resources belonging to projects to which the user has been assigned. This is the standard role for most users.
2. **MSP or Tenant Admin** - In addition to allowing the use of those policies and APIs which are granted to a Member, the MSP or Tenant Admin role also allows the user to use policies and APIs for creating and managing new projects and users within a specific account, assigning to these users, per project, roles and Zadara Cloud Services and AWS policies. It is recommended that each account have at least one user with this role.
3. **Zadara Ops Admin** - In addition to allowing the use of those policies and APIs which are granted to a Member and an MSP/Tenant Admin, the Zadara Ops Admin role also allows the user to use policies and APIs for viewing, creating, managing and deleting all physical resources, such as nodes (servers), disks, storage pools, physical networks, etc., and administrative entities such as accounts. It is possible to create more than one user per region with Zadara Ops Admin rights.

---

✓ **Note:** The Zadara Ops Admin user `admin` is only available to a Zadara user.

---

Two important guidelines concerning the assignment of Zadara Cloud Services roles:

1. Only assign a single role per project.
2. When assigning multiple projects to a user, assign the same role for each project.

---

✓ **Note:** Effective user and group permissions are determined by the intersection of the user's and group's assigned roles and policies.

For example, if a user is assigned the Tenant Admin role, and one of their Symp policies is `ReadOnlyAdministrator`, then the effective permission for that user would be read-only permission on all of the account's resources and aspects.

---

When Zadara Cloud Services is installed, it comes with a single account containing one project and one user available to Zadara.

- The account is called `cloud_admin`.
- The project inside is called `default`.
- The special Zadara user is called `admin`.

This user, who serves as the System Administrator of the entire region, comes assigned to the 'default' project with the 'Admin' Zadara Cloud Services role, the 'FullAccess' Zadara Cloud Services Policy and the 'AdministratorAccess' AWS API policy. This built-in account, project and user cannot be deleted or modified in any way.

---

**Important:** It is highly recommended to create and keep a "break the glass" local account admin user, with its credentials vaulted for future use. (i.e. This is not a personal user who can leave the organization potentially causing the credentials to be lost, but rather a user for emergency "break the glass" use cases).

This admin user can be used for maintenance of the IdP connection if it stops working properly for any reason.

---

### 1.3.2 AWS Roles

AWS IAM Roles are policy-based tokens with temporary credentials, allowing a user temporary access to AWS services and actions which the user is normally not permitted to access. These users may be from different projects or even different accounts. These roles can also be embedded into specific instances allowing these instances access to the necessary actions.

---

✓ **Note:** The AWS IAM roles are independent of the Zadara Cloud Services roles, which together with Zadara Cloud Services policies, grant access to Zadara Cloud Services services and actions.

---

The AWS IAM role consists of the following:

1. Permissions which give access to certain Zadara Cloud Services, supported AWS services or actions.
2. Trust policy that defines the relationship between user per project and this role.
  1. This nature of the relationship may be 'allow' which grants permission to the specified users to assume the role, or 'deny' which prevents these users from assuming the role.
  2. This permission may be granted to multiple users of the same projects, different projects within the same account, or even users of different accounts.

3. The maximum session duration that can be requested when assuming this role.

### 1.3.3 Policies

Policies can be defined and assigned to a user per project for both Zadara Cloud Services and AWS API. While AWS APIs and Zadara Cloud Services functionality may overlap, they are essentially two independent areas of functionality, requiring separate sets of policies. A user can be granted access to both AWS API and Zadara Cloud Services functionality on the same project.

1. The two policy types support the granting of access to one area without granting access to the other. For example, users working with only AWS APIs do not need access to the Zadara Cloud Services API/GUI.
2. The Zadara Cloud Services API is more extensive and all of its functionality is not covered by the AWS APIs.
3. Even APIs that appear to be similar in AWS API and Zadara Cloud Services API, such as “create vm” and “runinstances”, actually permit different actions.



## ACCOUNTS

### 2.1 Overview

The virtual resources of the Zadara Cloud Services region are managed through an administrative hierarchy of accounts & projects. Users, who are members of an account, can be assigned different projects within their account. This assignment enables you to provide access to Zadara Cloud Services to numerous users, dividing and separating the resources that each user will be able to view, create, and manage.

### 2.2 Basic Account Operations

To see account information and perform basic account configurations, navigate to **Identity & Access > Account**.

In the upper half of the account view, the following account information is available:

- Account Id
- Account Name
- MFA Status
- Number of associated projects, groups, and users
- Number of associated compute resources (VM instances, vCPU's, RAM)
- Number of associated networking resources (networks, floating IP's, security groups)
- Amount of associated storage resources (volumes, images, snapshots)

In addition, the following configurations can be made:

- Enforce MFA (Multi-factored Authentication)
- Assign tags which can be used to allow better account visibility, filtering, or reporting.

In the lower half of the account view, 4 tabs are available for the following operations:

1. **Projects** - Projects associated with the account are displayed. To associate the account with a new project, click **Create Project** in the displayed toolbar. An account can be associated with more than one project.
2. **Groups** - Groups associated with the account are displayed. To associate the account with a new group, click **Create Group** in the displayed toolbar.
3. **Users** - Users associated with the account are displayed. Use the toggle buttons in the display to enable/disable a specific user, or activate/deactivate MFA for a specific user once enabled for the account. To associate a new user with the account, click **Create User** in the displayed toolbar.
4. **Limits** - Configure resource limitations for the account as described in the following section.

---

✓ **Note:** From the **Identity & Access > Account** view, operations can be per account or per project. To see information or make configurations per project, select a specific project from the list displayed in the project tab. For more information on project-level operations, see [Creating Projects](#). If no project is selected from the list, the operations will be per account.

You can always be sure whether you are in the account or project view by the display on top of the toolbar.

- In the **account** view, the display will be:

**Home > Account > Account\_Name.**

- In the **project** view, the display will be:

**Home > Account > Account\_Name > Project > Project\_Name**

---

## 2.3 Account Limits

Compute, Services and Storage resources can be limited per account and per project within the account. When these limits are set, Zadara Cloud Services tracks the usage of these resources when they are allocated or freed-up, both at the project level and at the account level.

For a given resource, the sum of the limits **set** for each project within an account may exceed the limit set for the account itself. However the total amount of the resource actually **used** may never exceed either the project limit or account limit set for that resource. For example, there may be a 10-image limit set for account A, and a 5-image limit for each of its three member projects - P1, P2, P3. This sets the total limits of the projects at 15, even though the limit of account A is only 10. Users within account A will be allowed to create images until they either individually reach the limit of 5 in their project, or until a total of 10 images have been created in the account across all projects.

---

✓ **Note:** Networking resources can currently be limited only per project.

---

Limits can be imposed on the following account resources:

1. Compute

- Number of cores
- Number of images
- Number of instances
- Number of key-pairs
- RAM

2. Services

- Number of Kubernetes clusters
- Number of database instances
- Number of load balancers
- Number of registries

3. Storage

- Number of snapshots
- Number of volumes
- Volume capacity

---

✓ **Note:** Storage limits are defined and displayed per storage pool, which are then aggregated to an account limit.

---

---

✓ **Note:** Users with roles Member or Tenant Admin can **view** but not **add** or **modify** limits.

---



## PROJECTS

### 3.1 Creating Projects

If you want to divide your virtual resources among numerous users within each account, you may create different projects within the account. A **Tenant Admin** user can create projects only in their own account.

**To create a project:**

1. In the zCompute UI, go to the **Identity & Access > Accounts** view. Select the account for which you wish to create a project.
2. In the lower half of the account view, select the **Projects** tab and click **Create Project**. The **Create Project** dialog is displayed.
3. Enter the following information:
  1. **Project Name** – enter a name for the new project. The name must be unique within the account. Project names are not case-sensitive.
  2. **Project Description** [Optional] – enter a description of the new project.
  3. **Type** - select the type of project:
    - **VPC** - Virtual Private Cloud project type.
    - **DVS** - Distributed Virtual Switch project type.
  4. **IP Pool** (VPC-type projects) - select one of the IP pools. If there is no available IP pool, request an administrator to create a shared edge network available in your Zadara Cloud Services region.
  5. **Grant Permissions** - leave the checkbox selected (default) to grant yourself permissions on the new project.
4. Click **OK** to create the new project. The new project is created and is displayed in the **Projects** tab of the specific account.

### 3.2 Enabling or Disabling Projects

**To enable or disable a project:**

1. In the zCompute UI, go to the **Identity & Access > Accounts** view. Select the account containing the project that you wish to enable or disable.
2. In the lower half of the account view, select the **Projects** tab.
3. From the displayed project list, select the desired project.
4. If the project is currently disabled, click **Enable** in the toolbar to enable it.

5. If the project is currently enabled, click **Disable** in the toolbar to disable it.

## 3.3 Renaming Projects

To rename a project:

1. In the zCompute UI, go to the **Identity & Access > Accounts** view. Select the account containing the project that you wish to rename.
2. In the lower half of the account view, select the **Projects** tab.
3. From the displayed project list, select the desired project.
4. In the toolbar, click **Rename**.
5. In the **Rename Project** dialog, enter the desired changes in the project name and description.
6. Click **OK**.

## 3.4 Deleting Projects

To delete a project:

1. In the zCompute UI, go to the **Identity & Access > Accounts** view. Select the account containing the project that you wish to delete.
2. In the lower half of the account view, select the **Projects** tab.
3. From the displayed project list, select the desired project.
4. In the toolbar, click **Delete**.
5. In the **Delete Project** confirmation dialog, click **Delete**. A message confirming the deletion of the project will pop-up in the upper right-hand corner of the screen.

---

✓ **Note:** You cannot delete the 'default' project in the cloud\_admin account.

---

## 3.5 Assigning a User to a Project

To assign a user to a project:

1. In the zCompute UI, go to the **Identity & Access > Accounts** view. Select the account containing the project for which you wish to assign a user.
2. In the lower half of the account view, select the **Projects** tab.
3. From the displayed project list, select the desired project.
4. In the toolbar, click **Assign User**.
5. In the **Assign User** dialog, enter the following:
  1. **User** - select a user from the drop-down list which will display all users associated with the account.
  2. **Project Roles** - select **Member** or **Tenant Admin**.
  3. **Policies**

4. **AWS API Policies**
5. Click **OK**.
6. A new user will be created with the selected Zadara Cloud Services role and the default Zadara Cloud Services Policy 'FullAccess'.

## 3.6 Project Limits

### 3.6.1 Project Limits Overview

Zadara Cloud Services allows you to set limits on the amount of virtual compute, service, storage and network resources each project can use. It is recommended that each account **Tenant Admin** user set the available resource limits for each of the projects within their account.

### 3.6.2 Project Limits

To view existing project virtual resources:

1. Navigate to the **Identity & Access > Accounts** > view for the specific account containing the project for which the limits should be set.
2. In the lower half of the account view, select the **Projects** tab.
3. From the displayed project list, select the desired project.
4. In the bottom half of the project view, select the **Limits** tab. Existing limits, if any, will be displayed for the following resources:
  1. Compute:
    - Number of cores
    - Number of images
    - Number of instances
    - Number of key-pairs
    - RAM
  2. Services
    - Number of Kubernetes clusters
    - Number of database instances
    - Number of load balancers
    - Number of registries
  3. Storage
    - Number of snapshots
    - Number of volumes
    - Volume capacity

---

✓ **Note:** Storage limits are defined and displayed per storage pool, which are then aggregated to an Account limit.

---

#### 4. Network

- Floating IPs
- Networks
- Routers
- Security Groups
- Security Group Rules
- Subnets

#### To add new project compute, services, storage resource limits:

1. Navigate to the project **Limits** tab as described above.
2. Select the resource category (compute, services, storage) to be limited.
3. Click **Add**.
4. In the **Add Limit** dialog, select the resource to be limited. Note the current usage of the resource will also be displayed.
5. Enter the resource limit to be added. Verify that the limit exceeds the current usage.
6. Click **OK**.

#### To modify existing project compute, services, storage resource limits:

1. Navigate to the project **Limits** tab as described above.
2. Select the resource category (compute, services, storage) to be modified.
3. On the row with the specific limit to be modified, click on the modify icon (pencil).
4. In the **Edit Limit** dialog, note the current resource usage and limit.
5. Enter the new resource limit. Verify that the new limit exceeds the current usage.
6. Click **OK**.

#### To remove existing project compute, services, storage resource limits:

1. Navigate to the project **Limits** tab as described above.
2. Select the resource category (compute, services, storage) with limit to be deleted.
3. On the row with the specific limit to be removed, click on the delete icon. The existing limit will be deleted.



### 3.6.3 Managing Project Virtual Network Resource Limits

To limit project network resources:

1. Navigate to the project **Limits** tab as described above.
2. Click on the **Edit** button on the top-right of the **Network** list.
3. For each virtual network resource, either check the **Unlimited** box, or uncheck it and enter a limit.
4. Click **OK**. The network resource limit list will be updated accordingly.



## USERS

A user can be created within Zadara Cloud Services as a member of a single account. To access and use Zadara Cloud Services, a user must be assigned to at least one project within that account via a Zadara Cloud Services role, and assigned at least one Zadara Cloud Services Policy.

✓ **Note:**

- If a user is assigned to more than one project, the role assigned for each project must be the same.
- To perform these operations, you need to be either the Zadara Admin user who can add users to any account, or a Tenant Admin user who can add users to the tenant account.
- If a user originated from an Identity Provider and not from within Zadara Cloud Services, the only user action which can be performed is the management of their permissions.
- None of these actions can be performed on the system 'admin' user of the cloud\_admin account.

### 4.1 Creating Users

1. Navigate to the **Identity & Access > Accounts** view for the specific account for which you wish to create a user.
2. From the bottom half of the view, select the **Users** tab. This will display the currently defined users.
3. From the tab toolbar, click **Create User**.

4. In the **Create User** dialog, enter the following information:
  1. **Name** – enter a name. All names in an account must be unique. It is not case-sensitive.
  2. **E-mail** – enter the user's email address.

3. **Password** – enter a password. A valid password must satisfy the following currently defined rules:

- Be at least 8 characters long.
- Contain at least 1 number, 1 letter, and 1 special character.

When entering the password, the display on the right will dynamically indicate when each password rule has been satisfied.

4. **Validate Password** – re-enter the password.

5. Click **Next**. The **Permissions** tab displayed.

6. In the **Add Project** field, select a project from the pull-down list. You can select any project in the user's account. You may also select multiple projects, one at a time.

1. For each project added, enter the following information:

- **Roles** - select one of the available roles: Member or Tenant Admin. This is required for the first project. Default role: Member.



**Note:** It is highly recommended not to assign multiple roles for the same user in a single project.

- **Policies** - Select one or more Zadara Cloud Services Policies. The default ZCS policy is FullAccess. This policy grants full access to all of the functionality granted to this role prior to v5.3.4.



**Note:** Effective user and group permissions are determined by the intersection of the user's and group's assigned roles and policies.

For example, if a user is assigned the Tenant Admin role, and one of their Symp policies is ReadOnlyAdministrator, then the effective permission for that user would be read-only permission on all of the account's resources and aspects.

- **AWS Policies** - This field is optional and need only be populated if you want to grant access to AWS APIs.

2. To add a user to another project, click **Add Project** and enter the information above for the new project. To remove a project from the list, click the delete icon near the project name.
3. Click **Finish** to create the new user. The new user is created and is displayed in the **Users** tab of the specific account.

## 4.2 Deleting Users

When deleting a user, this user will no longer have access to Zadara Cloud Services. Deleting a user only removes the user from the system. It does not delete the virtual resources that this user created. These resources will still be accessible to all users with access to the projects for which these resources were created.

**To delete a user:**

1. Navigate to the **Identity & Access > Accounts** view.
2. From the bottom half of the view, select the **Users** tab. This will display the currently defined users.
3. Select the user to be deleted from the displayed list and click **Delete** from the tab toolbar. The **Delete User** confirmation notice appears.
4. Click **OK**. A message confirming the deletion of the user project will pop-up in the upper right-hand corner of the screen.

## 4.3 Managing Users Permissions

After a user is created, you can modify the user's permissions and add/delete projects assigned to the user.

**To manage the permissions of a user:**

1. Navigate to the **Identity & Access > Accounts** view.
2. From the bottom half of the view, select the **Users** tab. This will display the currently defined users.
3. Select the user whose permissions are to be modified from the displayed list and click **Manage Permissions** from the tab toolbar. The **Manage Permissions for User: <name of selected user>** dialog box is displayed.

**Manage Permissions for user: Member-1**

**+ Add Project** *All projects have been added*

**Project: Project-1**

Roles: Member

Policies: FullAccess

AWS API Policies:

**Project: Project-2**

Roles: Member

Policies: StratoReadOnlyAccess, VMFullAccess

AWS API Policies:

**Project: Project-3**

Roles: Member

Policies: No policies assigned

AWS API Policies: MemberFullAccess

Cancel Finish

- In the **Add Project** field, select a project from the pull-down list. The following permissions may be modified.
  - Roles
  - Policies
  - AWS API policies
- To modify permissions for another project, click **Add Project** and enter the information above for the new project. To remove a project from the list, click the delete icon near the project name.
- Click **Finish** to save the modified permissions.

## 4.4 Modifying Users

After creating a user, you can modify its name, email address and password expiration status.

- Navigate to the **Identity & Access > Accounts** view.
- From the bottom half of the view, select the **Users** tab. This will display the currently defined users.
- Select the user to be modified from the displayed list and click **Modify** from the tab toolbar. The **Modify User** dialog box is displayed.
- The following may be modified:
  - Name
  - Email Address

- **Password Expiration** - Enable/Disable

5. Click **Finish**. A message confirming the updating of the user will pop-up in the upper right-hand corner of the screen.

## 4.5 Resetting User Passwords

Once a user is created, the password can be reset by the Tenant Admin user of the user's account.

---

✓ **Note:**

- The password of users in accounts connected to an Identity Provider service, cannot be reset from within Zadara Cloud Services.
- 

To reset a user password:

1. Navigate to the **Identity & Access > Accounts** view.
2. From the bottom half of the view, select the **Users** tab. This will display the currently defined users.
3. Select the user whose password is to be modified from the displayed list and click **Set Password** from the tab toolbar. The **Set Password** dialog box is displayed.
4. Enter the following:
  - **Password**
  - **Validate Password** - re-enter the same password.
5. Click **OK**. A message confirming the successful resetting of the password will pop-up in the upper right-hand corner of the screen.





## CONNECTING AN ACCOUNT TO A MICROSOFT ACTIVE DIRECTORY IDENTITY PROVIDER

zCompute accounts' users can be authenticated using an external LDAP-compatible identity provider, such as a Microsoft Active Directory domain. Once it's set up, users and groups of the identity provider can be granted permissions to zCompute. To enable this, the zCompute account must be connected to Active Directory. Each zCompute account requires independent configuration for the Identity Provider connection.

---

**Important:** As a prerequisite, the zCompute cloud must have a route to the Active Directory domain controllers in order to connect a zCompute account to the Active Directory.

Tenants are advised to consult with their Managed Service Provider.

---

### 5.1 Connecting an Account to an Active Directory Identity Provider

---

✓ **Note:** Before connecting the zCompute account to selected users in Active Directory, it is recommended to first create a dedicated group for them in Active Directory, and add them to that group.

This allows you to use the filters to select only those users that should be connected to the zCompute account.

---

To connect an account to an Active Directory Identity Provider:

1. Navigate to the **Identity & Access > Accounts** view, and highlight the row of the account that you wish to connect to the Identity Provider.

An **Identity Provider** button appears in the toolbar.

2. Click **Identity Provider**.

The **Create Identity Provider** dialog opens at the **Connection** step.

Create Identity Provider
✕

Connection
LDAP Parameters

1
2

Server\*

Secure ☐

Secondary Server

Secure ☐

User ? \*

Password \*

10.11.12.13

Port\*

389

optional

Port

389

zadaraexample@example.com

.....

Cancel

Disconnect

Next

✓ **Note:** An existing LDAP connection can be disconnected via the **Disconnect** button.

1. Enter the Active Directory domain controller (DC) details:

- **Server** - The LDAP server or Active Directory domain controller's IP address.

For example: 10.11.12.13

✓ **Note:** If that server has a public DNS name (FQDN), it is possible to use the DNS name instead of the IP address.

For example: dc3.mydomain.com.

- **Secure** - Check this box if your LDAP server supports the secure LDAP protocol over SSL or TLS.
- **Port** - The reserved port is 389 if the connection is not secure, or 636 if the connection is secure.

- **Secondary Server** (optional) - A backup server if the first one is not available. The Active Directory Domain Controller address, expressed as an IP address or a DNS hostname.

For example: `10.11.12.14` or `dc4.mydomain.com`.

- **Secure** - Check this box if your secondary LDAP server supports the secure LDAP protocol over SSL or TLS.
- **Port** - The reserved port for the secondary server is 389 if the connection is not secure, 636 if the connection is secure.
- **User** - This is the user principal name (UPN) or distinguished name (DN) of a user through which one can gain access to Active Directory Server.

DN example: `cn=zadaraexample,cn=Users,dc=example,dc=com`

UPN example: `zadaraexample@example.com`



**Note:**

- This user should not be an administrator of the domain.
- This user establishes the connection to Active Directory, and is used as a “service account” to sync zCompute with the Active Directory domain.

To guarantee a continuous operation with Active Directory, configure this user’s Active Directory account so that it never expires, and that its password never changes and never expires.

**Zadara Example Properties**

Published Certificates | Member Of | Password Replication | Dial-in | Object  
 Security | Environment | Sessions | Remote control  
 Remote Desktop Services Profile | COM+ | Attribute Editor  
 General | Address | **Account** | Profile | Telephones | Organization

User logon name:  
 zadaraexample @example.com

User logon name (pre-Windows 2000):  
 EXAMPLE\ zadaraexample

Logon Hours... Log On To...

☐ Unlock account

Account options:

- ☐ User must change password at next logon
- ☒ User cannot change password
- ☒ Password never expires
- ☐ Store password using reversible encryption

Account expires:

- ☒ Never
- ☐ End of: Friday, January 24, 2025

OK Cancel Apply Help

- **Password** - The Active Directory password of the user.
- Click **Next**.

After validating the connection to the Active Directory server, the dialog displays the **LDAP Parameters** step.

2. **LDAP Parameters** - All parameters are expressed in the LDAP syntax.

**Create Identity Provider**
✕

**Connection**  
✓

**LDAP Parameters**  
2

Domain \*

User Tree DN \*

User ID Attribute

User Name Attribute

User Object Class

User Filter

Group Tree DN

Group Object Class

Group Filter

**Summary**

Total Users	4983
Filtered Users	500
Total Groups	8
Filtered Groups	5017
Users Under Groups	500

✓ **Note:** A **Summary** of LDAP users and groups appears on the right, and updates dynamically as filters are applied.

- **Domain** - The customer's domain in Active Directory.  
For example: `dc=example,dc=com`
- **User Tree DN** - The location in the Active Directory in which the users will be scanned.  
For example: `OU=Enterprise,DC=example,DC=com`
- **User ID Attribute**: The attribute name in Active Directory that represents the User ID.  
For example: `sAMAccountName`.
- **User Name Attribute**: The attribute name in Active Directory that represents the User Name.  
For example: `sAMAccountName`.
- **User Object Class**: The objectClass property of a user object in Active Directory.  
For example: `person`.
- **User Filter** - Filters the users scanned in the User Tree.  
For example:
  - `(name=example-*)` searches the User Tree for any user beginning with `example-`.

- (memberOf=cn=grp-exampleldap,cn=Users,dc=example,dc=com)

searches for all users who belong to the group `grp-exampleldap` in the Active Directory with domain components matching `dc=example,dc=com`.

---

✓ **Note:** The input syntax for the **User Filter** parameter includes the parentheses.

---

- **Group Tree DN** - The location in the Active Directory in which the groups will be scanned.

For example: `cn=Users,dc=example,dc=com`

**⚠ Caution:** If the **Group Tree DN** is left empty, the UI doesn't display the LDAP users' list. In this case, it is still possible to grant permissions to individual users, but these users' permissions don't display in the UI.

- **Group Object Class** - Active Directory's group object class for the groups. Default: `group`
- **Group Filter** - Filters the groups scanned in the Group Tree.

For example: `(name=grp-*)` searches for all groups with the prefix `grp-`.

---


✓ **Note:** The input syntax for the **Group Filter** parameter includes the parentheses.

---


Click **Finish**.

- The selected account is connected to MS Active Directory.
- The users matching the filters described above will appear as LDAP Users of the selected account, together with their Active Directory passwords and email addresses.
- The groups matching the filters described above will appear as groups of the selected account, containing the users defined in Active Directory.
- The account's top pane **Overview** section displays the **LDAP** badge, and summarizes the number of **LDAP Groups** and **LDAP Users**.


🏠 > Account > tenant



Account



Symp API Policies



AWS API

⚙ Identity Provider
🔒 Enforce MFA
⊖ Block Admin Access

**Info**

ID	3cba14f47453478b97182d62b05d8a2c
Name	tenant
MFA Enforced	<input type="radio"/>
Admin Access Blocked	<input type="radio"/>

**Overview** LDAP

2

Projects

8

Groups

501










Users

## 5.2 Viewing LDAP Groups

To view the LDAP Groups:


In the account's lower pane, click the **Groups** tab.

The LDAP Groups list displays.

Projects	Groups	Users	LDAP Users	Limits
 Create Group				
↑ Name		Projects		
Role-Alpha-Member				
Role-Alpha-TenantAdmin				
Role-Beta-Member				
Role-Beta-TenantAdmin				
Role-defaultproject-Admin				
Role-defaultproject-Member				
Role-defaultproject-nonprod-Admin				
Role-defaultproject-nonprod-Member				

By default, users in a group can't sign on until the group is assigned project permissions. The Projects column for groups without project permissions displays an alert symbol to indicate this.

To configure project permissions for a group, see [Managing an LDAP Group's Permissions](#).

 **Note:** For a zCompute account that is connected to an Active Directory, new users and groups can only be created and managed in the Active Directory.

**Important:** As a best practice, it is recommended to have a single local (non-LDAP) tenant admin user for “break the glass” scenarios, in the event of LDAP connection loss or malfunction.

### 5.2.1 Managing an LDAP Group's Permissions





To assign permissions to an LDAP group and its members:

1. In the account's lower pane, click the **Groups** tab.  
The LDAP Groups list displays.
2. Highlight the row of the group to apply permissions.
3. In the lower pane menu bar, click **Manage Policies**.  
The **Manage Permissions for Group** dialog opens.

The screenshot shows the 'Manage Permissions for group: Role-Alpha-Member' interface. The left sidebar has tabs for 'Projects', 'Groups', 'Users', and 'LDAP Users'. The 'Groups' tab is active, showing a list of roles. The main panel shows two projects, 'proj1-vpc' and 'proj2-vpc', each with 'Member' roles and 'FullAccess' Symp API Policies. The 'Add Project' button is visible at the top.

1. Configure the projects and their permissions for the group
2. Click **Finish**.

The updated projects appear in the Projects column for the group, in the **Groups** tab.

Projects	Groups	Users	LDAP Users	Limits
				
Create Group	Manage Policies	Delete Group	Edit Group	

Name	Projects
Role-defaultproject-Admin	1
Role-defaultproject-Member	1
Role-defaultproject-nonprod-Admin	1
Role-defaultproject-nonprod-Member	1
Role-Alpha-TenantAdmin	1
Role-Alpha-Member	proj1-vpc, proj2-vpc
Role-Beta-TenantAdmin	1
Role-Beta-Member	1


### 5.3 Viewing LDAP Users

To view the LDAP Users:

In the account's lower pane, click the **LDAP Users** tab.

The LDAP Users list displays.



Projects	Groups	Users	LDAP Users	Limits
 Create User				
↑ Name	Email	External ID	Projects	
operator.1			proj1-vpc	
operator.2			!	
operator.3			!	
operator.4			!	
operator.5			!	
operator.6			!	
operator.7			!	
operator.8			!	

**Note:**

- LDAP user passwords and email addresses **cannot** be modified within zCompute.

- **Assigning Project Roles and zCompute Policies**

Before these users will be able to work with zCompute, you must first assign a project to each user, either individually or via the groups of which they are members, together with the group's zCompute role and policies.

Users who are members of groups configured with project permissions have their associated projects listed in the Projects column. For other users, the Projects column displays an alert symbol, indicating that the user cannot sign on until project permissions are assigned. User permissions can be assigned individually or as a member of a group with configured permissions.

## 5.4 Assigning zCompute Roles and permissions to an Active Directory User - UI

The following UI actions are recommended for assigning the Account and Project 'Admin' roles and permissions to a user added to zCompute via Active Directory:

1. Locate the new user added to zCompute via Active Directory.
  1. On the **Identity & Access > Accounts** <account> view for the requested account, click the **LDAP Users** tab.  
The list of users for that account displays.
  2. Click the user to assign zCompute permissions.  
The user's **Project Permissions** tab displays the user's role and policies per project.
2. Assign a project to the user with a project role and zCompute policy.
  1. In the user's upper menu bar, click **Manage Permissions**.
  2. The dialog box **Manage Permissions for User** opens.

3. From the **+ Add Project** dropdown list, select the project.
4. In the section that opens for the project, complete the user's permission settings:

The **Roles**, **AWS API Policies** and **Symp API Policies** have descriptive names that reflect the role or policy scope.

1. **Roles** - From the dropdown, select the role.  
Multiple roles can be applied.
2. **AWS API Policies** - From the dropdown, select the policy.  
Multiple AWS API policies can be applied.
3. **Symp API Policies** - From the dropdown, select the policy.  
Multiple Symp API policies can be applied.
4. Optionally, repeat these steps to assign roles and policies to the user for additional projects.
5. Click **Finish** to save the user's roles and policies per project.

The user's **Project Permissions** tab in the lower pane displays the user's role and policies per project.

## 5.5 Assigning zCompute Roles and permissions to an Active Directory User - CLI

The following CLI commands are recommended for assigning the Account and Project 'Admin' roles and permissions to a user made available to zCompute via Active Directory:

✓ **Note:** In this example the assumption is that the zCompute account, **new\_account**, which is connected to a domain in Active Directory, already contains the project, **new\_project**.

1. Use symp's `user list` command to locate the user.

```
user list -c id -c name -c domain_id
```

id	name	domain_id
c8a63b29558d4765a6cd78760729a2f7	new_user	2d27e2fe6d8a4398b901c4d84c478777
admin	admin	default

Note the user's domain ID.

2. Use symp's `user list` command to verify that the user **new\_user** from the Active Directory list, does not already appear in another zCompute account.

✓ **Note:** A username can appear in more than one zCompute account, but each user's ID is unique even though the username might appear more than once.

```
user list --name new_user -c id -c name -c domain_id
```

id	name	domain_id
c8a63b29558d4765a6cd78760729a2f7	new_user	2d27e2fe6d8a4398b901c4d84c478777

(continues on next page)

(continued from previous page)

```

| name      | new_user      |
| domain_id | 2d27e2fe6d8a4398b901c4d84c478777 |
+-----+-----+

```

3. Use `symp's domain list` command to list the domains with their IDs and names.

```

domain list -c id -c name

+-----+-----+
| id      | name      |
+-----+-----+
| 2d27e2fe6d8a4398b901c4d84c478777 | new_account |
| default | cloud_admin |
+-----+-----+

```

4. Use `symp's project list` command to list the projects with their IDs, names and domains.

```

project list -c id -c name -c domain_name

+-----+-----+-----+
| id      | name      | domain_name |
+-----+-----+-----+
| 4bd79a2fa9574af2a4b9a7a87195f144 | default    | cloud_admin  |
| 1569c28e3a344ee2b3989640499b8eca | new_project | new_account  |
+-----+-----+-----+

```

5. Use `symp's role list` command to list all roles in zCompute.

```

role list

+-----+-----+
| id      | name      |
+-----+-----+
| admin    | admin      |
| tenant_admin | tenant_admin |
| _member_ | member     |
+-----+-----+

```

6. Use `symp's project list-roles-on-project` command to check if user `new_user` has already been assigned a role in the project `new_project`.

Syntax: `project list-roles-on-project <project_id> <user_id>`

```

project list-roles-on-project 1569c28e3a344ee2b3989640499b8eca c8a63b29558d4765a6cd78760729a2f7

+-----+-----+
| value      | tenant_admin |
+-----+-----+

```

If the user is already assigned a role other than `admin` in the project `new_project`, use the `project revoke-role` command to remove the role from `new_user`.

Syntax: `project revoke-role <project_id> <user_id> <role_id>`

```

project revoke-role 1569c28e3a344ee2b3989640499b8eca c8a63b29558d4765a6cd78760729a2f7 tenant_admin

+-----+-----+

```

(continues on next page)

(continued from previous page)

```
| value | Success |
+-----+-----+
```

7. Use symp's `project grant role` command to assign the admin role to user `new_user` in the project `new_project`.

Syntax: `project grant-role <project_id> <user_id> <role_id>`

```
project grant-role 1569c28e3a344ee2b3989640499b8eca c8a63b29558d4765a6cd78760729a2f7 admin

+-----+-----+
| value | Success |
+-----+-----+
```

8. Use symp's `project list-roles-on-project` command to verify that the role of `new_user` in `new_project` is `admin`.


Syntax: `project list-roles-on-project <project_id> <user_id>`

```
project list-roles-on-project 1569c28e3a344ee2b3989640499b8eca c8a63b29558d4765a6cd78760729a2f7

+-----+-----+
| value      | admin      |
+-----+-----+
```

## 5.6 LDAP User Sign-on to zCompute

After LDAP users' project and permission assignments are configured in zCompute, the users can sign on to zCompute using their short username, as registered in zCompute, and listed in the account's **LDAP Users** tab.

Projects	Groups	Users	LDAP Users	Limits
 Create User				
↑ Name	Email	External ID	Projects	
operator.1			proj1-vpc	
operator.2			!	
operator.3			!	

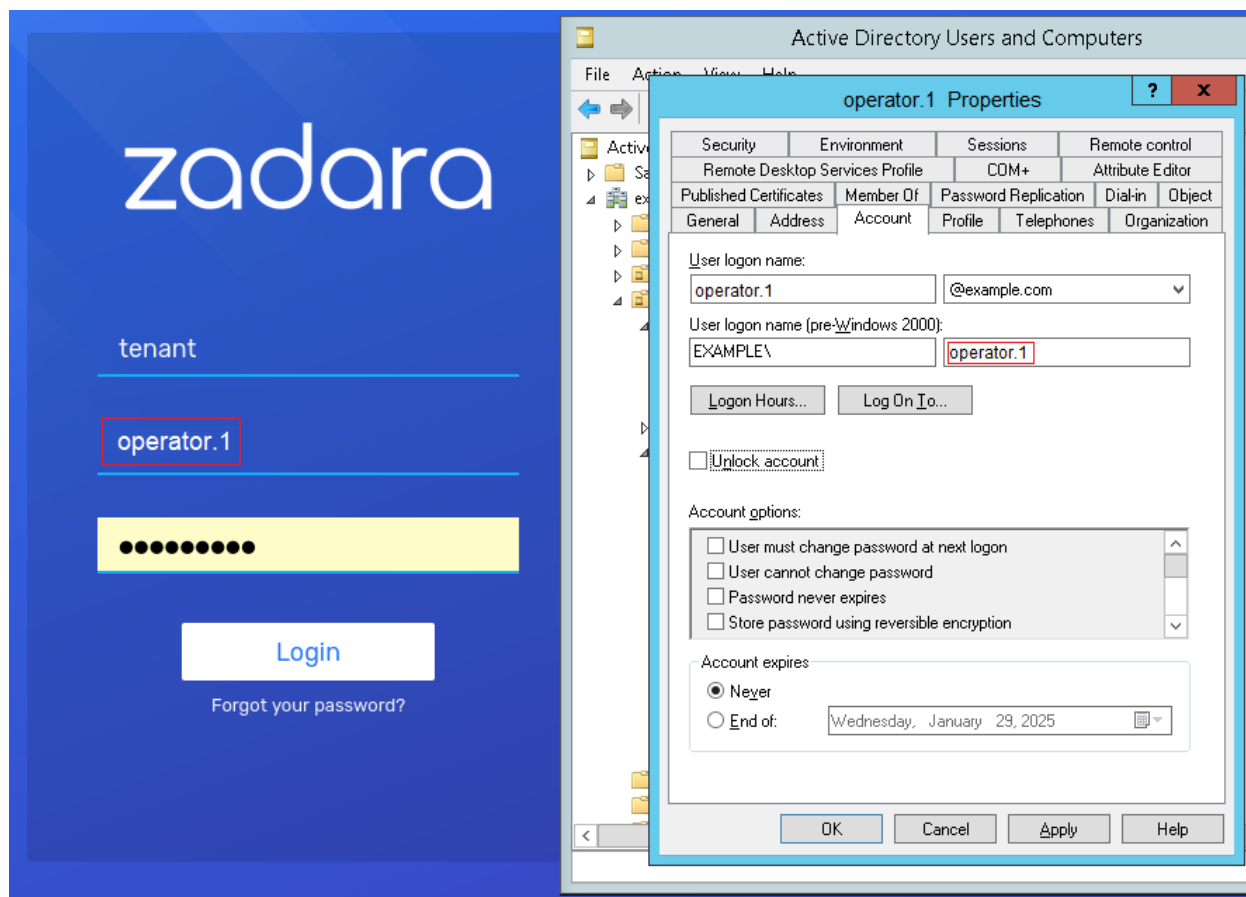
In this example:

- The zCompute account name is `tenant`.

For this example, this is the account where the administrator loaded LDAP users and groups, and assigned them zCompute projects and permissions.

- The LDAP user's username is `operator.1`.

This is the short username of the LDAP user, imported into zCompute based on the mapping of the **LDAP User Name Attribute** `sAMAccountName`. The mapping was configured in the **LDAP Parameters** step of the **Create Identity Provider** dialog.





## AUTHENTICATION USING ZCOMPUTE APIS

Symp CLI commands can be run by authenticating, without requiring a token.

However, it is advisable to authenticate and then generate a token for use in any automated Symp CLI or API process. This will prevent possible `connection reset by peer` errors when rate limits are exceeded.

### 6.1 API Endpoints

The list of a cluster's API endpoints is available in the UI.

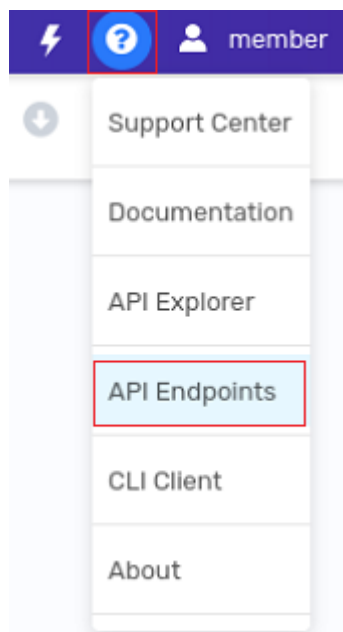
To view the cluster's API endpoints:

1. In the UI, in the upper right corner near the username, click **Help** (the ? icon).

The **Help** dropdown menu opens.

2. In the dropdown menu, select **API Endpoints**.

The **API Endpoints** dialog opens, listing the cluster's API endpoints.



#### API Endpoints

<b>EC2</b>	<code>https://&lt; cluster domain or IP &gt;/api/v2/aws/ec2/</code>
<b>ELB</b>	<code>https://&lt; cluster domain or IP &gt;/api/v2/aws/elbv2/</code>
<b>ASG</b>	<code>https://&lt; cluster domain or IP &gt;/api/v2/aws/autoscaling/</code>
<b>CloudWatch</b>	<code>https://&lt; cluster domain or IP &gt;/api/v2/aws/cloudwatch/</code>
<b>SNS</b>	<code>https://&lt; cluster domain or IP &gt;/api/v2/aws/sns/</code>
<b>IAM</b>	<code>https://&lt; cluster domain or IP &gt;/api/v2/aws/iam/</code>
<b>Route53</b>	<code>https://&lt; cluster domain or IP &gt;/api/v2/aws/route53/</code>
<b>ACM</b>	<code>https://&lt; cluster domain or IP &gt;/api/v2/aws/acm/</code>

## 6.2 API Explorer

zCompute provides an interactive Swagger-based API Explorer.

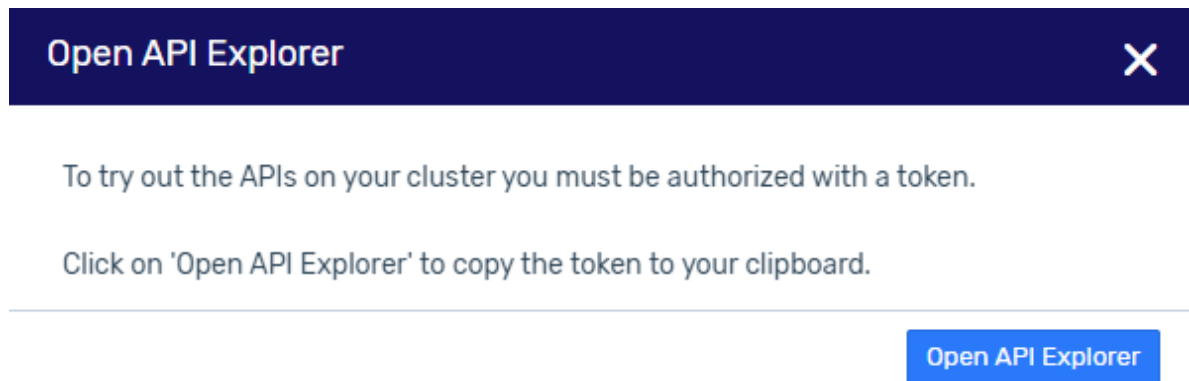
API operations require authorization with a token.

### 6.2.1 Obtaining a token in the UI

To obtain a token in the UI:

1. In the UI, in the upper right corner near the username, click **Help** (the ? icon), and in the dropdown menu, select **API Explorer**.

The **Open API Explorer** dialog opens.



2. Click the **Open API Explorer** button.

- A token is generated and automatically copied to the clipboard.
- The **zCompute Symp API Explorer** opens, listing the zCompute APIs.



---

✓ **Note:** The token in the clipboard can be pasted and used in the API Explorer, or for running API operations using the CLI.

---



## 6.2.2 Running API operations in the API Explorer

1. After Obtaining a token in the UI, click an API in the **zCompute Symp API Explorer** list.  
A Swagger UI opens, displaying the selected API's REST methods.
2. Scroll down the list to locate the desired operation to run.
3. Click the selected REST method or its description, to expand the view to display its parameters.

The screenshot shows the API Explorer interface with the following elements:

- GET /vm-snapshots/{vm\_snapshot\_id}** Retrieve VM snapshot metadata (locked)
- PATCH /vm-snapshots/{vm\_snapshot\_id}** Update the VM snapshot (locked)
- GET /vms List VMs** (selected, highlighted with a red box, and locked)

Below the selected endpoint, the text "List VMs." is displayed.

**Parameters** (highlighted with a red box)

**Try it out** (button, highlighted with a red box)

Name	Description
<b>detailed</b> boolean (query)	Show detailed list <input type="checkbox"/>
<b>status</b> string (query)	Filter by VM status <input type="text" value="status - Filter by VM status"/>

4. Click the **Lock** icon on the right, to submit the authorization token.

✓ **Note:** Alternatively, click the **Authorize** button in the screen header.

The **Available Authorizations** dialog opens.

## Available authorizations



### token (apiKey)

Name: X-Auth-Token

In: header

Value:

JBswAcNv8rgTIK4Tga1WQ=

Authorize

Close

1. In the **token (apiKey)** > **Value** field, paste the token from your clipboard.
2. Click **Authorize**.

✓ **Note:** After the token is authorized and while it is still valid, REST methods for all APIs can be run. It is not necessary to repeat this authorization step.

3. Click **Close** to close the **Available Authorizations** dialog and return to the selected API REST method screen.
5. Click **Try it out**.
6. Enter parameters for the REST method, and click the **Execute** bar.

The **Responses** pane expands, displaying:

- The **Curl** command that was executed.
- The **Request URL**.
- The **Server response**, comprising the return code, response body and response headers.

## Curl

[illegible]

<https://10.16.146.38/api-explorer/api/v2/compute/vms>

## Code

### Details

200

### Response body

```
[
  {
    "instance_profile": "",
    "disable_delete": false,
    "availability_zone": null,
    "networks": [
      {
        "device_index": 0,
        "port_id": "68ce7ec6-c3c3-4b13-bbd6-b092b1823adc",
        "id": "042a06aa-c12e-46f7-afb6-312acac2a257",
        "address": "169.254.64.253"
      }
    ],
    "migration_enabled": "True",
    "id": "26848e5c-43cc-4735-8c2d-f198c682dfb",
    "user_id": "admin",
    "slaProfile": "reserved",
    "hostname": "stratonode0.node.strato",
    "restart_on_failure": true,
    "instanceType": "msit.VPC_DNS",
    "key_pair": null,
    "project_id": "acdc717b53a5421bb3e5eb0f2dc7e58c",
    "launched_at": "2024-01-15T07:22:49Z",
    "metadata": {
      "managing_service": "VPC",
      "managing_resource_id": "062b8cd7-982f-4c3c-afc1-dd95dfbfa7bd",
      "permissions": "internal"
    }
  }
]
```

## 6.3 Generating a token at a bash prompt

Since some commands are limited to the scope of a project, while others have the scope of an entire account (domain), it's necessary to generate a token for the account (domain) scope, and a separate token for the project scope.

Using `curl` on the `/api/v2/identity/auth` endpoint, to generate authentication tokens involves two steps:

First, generating an account (domain) token, and then using the domain token to generate a project token.

1. Generating a domain token:

The `/api/v2/identity/auth` endpoint's authentication token request data structure that defines the **scope** of an entire **account** (domain):

```
{
  "auth": {
    "scope": {
      "domain": {
        "name": "<ACCOUNT_NAME>"
      }
    },
    "identity": {
      "password": {
        "user": {
          "domain": {
            "name": "<ACCOUNT_NAME>"
          },
          "password": "<PASSWORD>",
          "name": "<USER_NAME>"
        }
      },
      "methods": [
        "password"
      ]
    }
  }
}
```

1. The `curl` command at the bash prompt for generating an account (domain) authentication token:

```
curl -D - -H "Content-Type: application/json" -X POST https://<cluster FQDN>/api/v2/identity/auth
↪auth -d \
'{"auth": {"scope": {"domain": {"name": "<account>"}}, "identity": {"password": {"user": {
↪"domain": {"name": "<account>"}, "password": "<password>", "name": "<user>"}}, "methods": [
↪"password"]}}}'
```

For example:

```
curl -D - -H "Content-Type: application/json" -X POST https://10.11.12.13/api/v2/identity/auth
↪-d \
'{"auth": {"scope": {"domain": {"name": "acc1"}}, "identity": {"password": {"user": {"domain":
↪{"name": "acc1"}, "password": "Mypass1!", "name": "user1"}}, "methods": ["password"]}}}'
```

2. The domain token is returned in the `x-subject-token` header, and can be copy/pasted to assign it to an environment variable for later use.

For example:

```
export ZC_DOMAIN_TOKEN=MIISugYJKoZIhvcNAQcCoIISQzCC...<2K+ character string>...
↪ngpXjEqQtXlRGuCEIvo46sGMfedc=
```

- Using the previously generated domain token to generate a project token:

The `/api/v2/identity/auth` endpoint's authentication token request data structure that defines the **scope** as limited to a **project**, and the identity method as a **token**:

```
{
  "auth": {
    "scope": {
      "project": {
        "domain": {
          "name": "<ACCOUNT_NAME>"
        },
        "name": "<PROJECT_NAME>"
      }
    },
    "identity": {
      "token": {
        "id": "'<DOMAIN_TOKEN>'"
      },
      "methods": [
        "token"
      ]
    }
  }
}
```

- The `curl` command at the bash prompt for generating a **project** authentication token:

```
curl -D - -H "Content-Type: application/json" -X POST https:<cluster FQDN>/api/v2/identity/
↪auth -d \
'{"auth": {"scope": {"project": {"domain": {"name": "<account>"}, "name": "<project>"}}},
↪{"identity": {"token": {"id": "'<domain token string>'"}, "methods": ["token"]}}}'
```

For example:

```
curl -D - -H "Content-Type: application/json" -X POST https://10.11.12.13/api/v2/identity/auth
↪-d \
'{"auth": {"scope": {"project": {"domain": {"name": "acc1"}, "name": "vpcproj1"}}, "identity":
↪{"token": {"id": "'$ZC_DOMAIN_TOKEN'"}, "methods": ["token"]}}}'
```

- The project token is returned in the `x-subject-token` header, and can be copy/pasted to assign it to an environment variable for later use.

For example:

```
export ZC_PROJECT_TOKEN=MIIScwYJKoZIhvcNAQcCoIISZDCC...<2K+ character string>...
↪VpADxR3RDrScUbgSwMAaZ8zCSrow=
```

- To use `curl` to call a REST method on an endpoint, provide the token string in the `X-Auth-Token` header:

```
curl -L -X <REST method> "https://<cluster FQDN>/api-explorer/api/v2/<endpoint>/" \
-H "accept: application/json" \
-H "X-Auth-Token: <token string>"
```

For example, to retrieve the project's tags, use the `GET` method on the `/api-explorer/api/v2/tags/`

endpoint, and the project authentication token (assigned in the previous step to the environment variable `$ZC_PROJECT_TOKEN`). This example uses `sed` and `awk` to format the output:

```
curl -L -X GET "https://10.11.12.13/api-explorer/api/v2/tags/" \
-H "accept: application/json" \
-H "X-Auth-Token: $ZC_PROJECT_TOKEN" | \
awk 'BEGIN{FS=",";OFS="\n"} FNR==1{$1=$1;print;exit}' | \
sed '/{/{x;p;x;}'
```

The response:

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload Upload	Total	Spent	Left	Speed
100	116	100	116	0	0	11756	0
100	2302	100	2302	0	0	59137	0

```
[{"description": "Tag2"
"updated-at": "2023-10-31T08:44:01Z"
"created-at": "2023-10-31T08:44:01Z"
"value": ""
"scope": "public"
"project-id": "7ff34832b6754f7d91a918302af3e77f"
"name": "vpctag2"}

{"description": "Tag 1"
"updated-at": "2023-10-31T08:10:49Z"
"created-at": "2023-10-31T08:10:49Z"
"value": ""
"scope": "public"
"project-id": "7ff34832b6754f7d91a918302af3e77f"
"name": "vpctag1"}]
```

### 6.3.1 Time-based One-Time Passcode (TOTP) Multi-Factor Authentication (MFA)

If a user has Time-based One-Time Passcode (TOTP) Multi-Factor Authentication (MFA) enabled, you must also include both of the following:

- A `totp` entry in the `methods` array.
- A `totp` object block with details.

For example:

```
{
"auth": {
  "identity": {
    "methods": [
      "password",
      "totp"
    ],
    "password": {
      "user": {
        "domain": {
          "name": "<ACCOUNT_NAME>"
        },
        "name": "<USERNAME>",
        "password": "<PASSWORD>"
      }
    }
  }
}
```

(continues on next page)

(continued from previous page)

```

    }
  },
  "totp": {
    "user": {
      "domain": {
        "name": "<ACCOUNT_NAME>"
      },
      "name": "<USERNAME>",
      "passcode": "<TOTP_CODE>"
    }
  }
},
"scope": {
  "domain": {
    "name": "<ACCOUNT_NAME>"
  }
}
}
}

```

## 6.4 Generating a token in Symp CLI

1. To generate an authentication token via the Symp CLI shell, use the Symp `login` command:

```
login <user> <password> <domain> <project> -f value
```

For example:

```
login user1 Mypass1! acc1 vpcproj1 -f value

MIISUgYJKoZIhvcNAQcCoIISQzCC...<2K+ character string>...ngpXjEqQtXlRGuCEIvo46sGMfedc=
```

Alternatively, an example of a single command at the bash prompt to generate an authentication token and assign it to an environment variable:

```
export ZC_PROJECT_TOKEN=`symp -k --url https://10.11.12.13 -d acc1 -u user1 --project vpcproj1 -p
↪Mypass1! login user1 Mypass1! acc1 vpcproj1 -f value`

echo $ZC_PROJECT_TOKEN

MIISUgYJKoZIhvcNAQcCoIISQzCC...<2K+ character string>...ngpXjEqQtXlRGuCEIvo46sGMfedc=
```

2. Using the token to sign on to Symp from the bash prompt:

```
symp -k --url https://<cluster IP address> --project-token <token string>
```

For example, generating the token (as in the previous step), and using it to sign on to Symp:

```
export ZC_PROJECT_TOKEN=`symp -k --url https://10.11.12.13 -d acc1 -u user1 --project vpcproj1 -p
↪Mypass1! login user1 Mypass1! acc1 vpcproj1 -f value`

symp -k --url https://10.11.12.13 --project-token $ZC_PROJECT_TOKEN
```

```

Starting new HTTPS connection (1): 10.16.145.114
Connecting in insecure mode!
Starting new HTTPS connection (1): 10.16.145.114
Starting new HTTPS connection (1): 10.16.145.114

d88888b dP dP 8888ba.88ba 888888ba dP dP .88888. 888888ba dP dP
88. "' Y8. .8P 88 `8b `8b 88 `8b 88 88 d8' `8b 88 `8b Y8. .8P
`Y88888b. Y8aa8P 88 88 88 a88aaaa8P' 88aaaaa88a 88 88 88 88 Y8aa8P
`8b 88 88 88 88 88 88 88 88 88 88 88
d8' .8P 88 88 88 88 88 88 88 Y8. .8P 88 88 88
Y88888P dP dP dP dP dP dP dP dP `8888P' dP dP dP

Tap <TAB> twice to get list of available commands.
Type --help to get help with any command
Symphony >

```

## 6.5 Python authentication example

When using a zCompute API, the user is required to pass a valid token in an X-Auth-Token HTTP header.

Users can generate a valid token by logging into the system using the `identity/auth` API.

The following Python script is an example of this process:

```

#!/usr/bin/python
from __future__ import print_function
import copy
import getpass
import os
import sys
import argparse
import requests
try:
    input = raw_input
except NameError:
    pass

IDENTITY_API_SUFFIX = "identity"
VPC_API_SUFFIX = "vpc"
DEFAULT_HEADERS = {
    "Accept": "application/json",
    "Content-Type": "application/json"
}
_debug = False

def _debug_msg(msg, *args, **kwargs):
    global _debug
    if _debug:
        print(msg.format(*args, **kwargs), file=sys.stderr)

def generate_totp_passcode(secret):
    """Generate TOTP passcode.
    :param bytes secret: A base32 encoded secret for the TOTP authentication
    :returns: totp passcode as bytes

```

(continues on next page)



(continued from previous page)

```

"""
try:
    import mintotp
except:
    sys.exit("TOTP code generation library is not present. Please install mintotp using 'pip install
↳ mintotp'")
    return mintotp.totp(secret)

class ZComputeApi(object):

    def __init__(self, api_base_url, account_name, project_name, user, password,
                 insecure=False, totp_code=None, mfa_secret=None, debug=False):
        self._api_base_url = '{} /api/v2'.format(api_base_url)
        self._account_name = account_name
        self._project_name = project_name
        self._user = user
        self._password = password
        self._insecure = insecure
        self._token = None
        self._totp_code = totp_code
        self._mfa_secret = mfa_secret
        self._debug = debug

    def reset_api_params(self, token=None, project_name=None):
        if token:
            self._token = token
        if project_name:
            self._project_name = project_name

    def send_request(self, method='GET', api_path='', api_body=None, headers=None, raise_on_status=True):
        api_url = '{} /{}'.format(self._api_base_url, api_path)
        if headers:
            headers = copy.deepcopy(headers)
            headers.update(DEFAULT_HEADERS)
        else:
            headers = copy.deepcopy(DEFAULT_HEADERS)
        if self._token:
            headers['X-Auth-Token'] = self._token
        try:
            response = requests.request(
                method=method,
                url=api_url,
                headers=headers,
                json=api_body,
                verify=not self._insecure
            )
            if raise_on_status:
                response.raise_for_status()
        except Exception:
            _debug_msg("Failed sending {} {} request".format(method, api_url))
            raise
        return response

    def send_api_request(self, method='GET', api_path='', api_body=None):
        return self.send_request(method=method, api_path=api_path, api_body=api_body).json()

```

(continues on next page)

(continued from previous page)

```

def _get_project_scope(self):
    return {"project": {"name": self._project_name, "domain": {"name": self._account_name}}}

def _get_domain_scope(self):
    return {"domain": {"name": self._account_name}}

def _get_password_auth(self):
    return {
        "methods": [
            "password"
        ],
        "password": {
            "user": {
                "name": self._user,
                "password": self._password,
                "domain": {
                    "name": self._account_name
                }
            }
        }
    }

def _add_totp_code(self, identity_auth, totp_code):
    if identity_auth.get('methods'):
        identity_auth['methods'].append('totp')
    else:
        identity_auth['methods'] = ['totp']
    identity_auth['totp'] = {
        "user": {
            "name": self._user,
            "passcode": totp_code,
            "domain": {
                "name": self._account_name
            }
        }
    }
    return identity_auth

def get_project_token(self):
    identity_auth = self._get_password_auth()
    if self._mfa_secret:
        self._totp_code = generate_totp_passcode(self._mfa_secret)
    if self._totp_code:
        self._add_totp_code(identity_auth, self._totp_code)
    auth_json = {
        "auth": {
            "identity": identity_auth,
            "scope": self._get_project_scope()
        }
    }
    response = self.send_request('POST', '{}/auth'.format(IDENTITY_API_SUFFIX), auth_json, raise_on_
    status=False)
    if response.status_code == requests.codes.UNAUTHORIZED and response.json().get('receipt'):
        os_receipt = response.headers.get('openstack-auth-receipt')
    if self._mfa_secret:
        _debug_msg("Generating MFA secret")
        totp_code = generate_totp_passcode(self._mfa_secret)

```

(continues on next page)

(continued from previous page)

```

        else:
            totp_code = str(input('MFA Code: ')).lower().strip()
            identity_auth = self._add_totp_code({}, totp_code)
            auth_json = {
                "auth": {
                    "identity": identity_auth,
                    "scope": self._get_project_scope()
                }
            }
            response = self.send_request('POST', '{} /auth'.format(IDENTITY_API_SUFFIX),
                                       auth_json,
                                       headers={"openstack-auth-receipt": os_receipt})
        else:
            response.raise_for_status()

        return response.headers['x-subject-token']

def get_account_token(self):
    auth_json = {
        "auth": {
            "identity": self._get_password_auth(),
            "scope": self._get_domain_scope()
        }
    }
    auth_response = self.send_request('POST', '{} /auth'.format(IDENTITY_API_SUFFIX), auth_json)
    return auth_response.headers['x-subject-token']

def get_user_default_project(self):
    details_api_uri = '{} /users/myself/projects'.format(IDENTITY_API_SUFFIX)
    projects = self.send_api_request('GET', details_api_uri)
    if len(projects) > 1:
        sys.exit("There are {} projects: {}\n"
                "please select one using --project flag".format(
                    len(projects), ', '.join([p['name'] for p in projects])))
    return projects[0]['name']

def ask_for_value(parameter, current, hide=False):
    if hide:
        value = getpass.getpass("{} {}: ".format(parameter))
    else:
        value = input("{} {}: ".format(parameter, current or '')).strip()
    if value:
        return value
    return current

def main():
    global _debug
    parser = argparse.ArgumentParser(description="Obtain zCompute API Token using user credentials")
    parser.add_argument("cluster", help="Cluster API endpoint IP or host name")
    parser.add_argument("--interactive",
                        help="Will get login credentials interactively",
                        action='store_true',
                        default=False)
    parser.add_argument("--account",
                        help="Account name (will use environment variable ZCOMPUTE_ACCOUNT if not")

```

(continues on next page)

(continued from previous page)

```

↪provided)",
                default=os.environ.get('ZCOMPUTE_ACCOUNT', None),
                required=False)
    parser.add_argument("--username",
                        help="User name (will use environment variable ZCOMPUTE_USER if not provided)",
                        default=os.environ.get('ZCOMPUTE_USER', None),
                        required=False)
    parser.add_argument("--mfa-secret",
                        help="MFA secret to generate totp code MFA enabled login "
                              "(will use environment variable ZCOMPUTE_MFASECRET if not provided)",
                        default=os.environ.get('ZCOMPUTE_MFASECRET', None),
                        required=False)
    parser.add_argument("--totp-code",
                        help="TOTP code to use for MFA enabled login "
                              "(will use environment variable ZCOMPUTE_TOTP if not provided)",
                        default=os.environ.get('ZCOMPUTE_TOTP', None),
                        required=False)
    parser.add_argument("--project",
                        dest='project_name',
                        help="Project name (will use environment variable ZCOMPUTE_PROJECT if not
↪provided)"
                        " if not provided and only one exists in the account will it",
                        default=os.environ.get('ZCOMPUTE_PROJECT', None),
                        required=False)
    parser.add_argument("--password",
                        help="Password - will use environment variable ZCOMPUTE_PASSWORD if not provided",
                        default=os.environ.get('ZCOMPUTE_PASSWORD', None))
    parser.add_argument("--debug", help="Print debug messages", default=False, action='store_true')
    parser.add_argument("-k", "--insecure", action='store_true', help="Do not verify cluster certificate",
↪default=False)
    args = parser.parse_args()
    if args.debug:
        _debug = True
    if args.interactive:
        args.account = ask_for_value('account', args.account)
        args.project_name = ask_for_value('project name', args.project_name)
        args.username = ask_for_value('User name', args.username)
        args.mfa_secret = ask_for_value('MFA secret', args.mfa_secret, hide=True)
        args.totp_code = ask_for_value('TOTP code', args.totp_code)
        args.password = ask_for_value('Password', args.password, hide=True)

    url = 'https://{0}'.format(args.cluster)
    if args.cluster.startswith('https://'):
        url = args.cluster
    api = ZComputeApi(url, args.account, args.project_name, args.username, args.password,
                     insecure=args.insecure, totp_code=args.totp_code, mfa_secret=args.mfa_secret,
↪debug=args.debug)
    if not args.password:
        sys.exit("Please provide a password")
    if not args.project_name:
        if args.account is None:
            sys.exit("Please provide the account name to login into")
        if args.username is None:
            sys.exit("Please provide the user name to login with")
        api.reset_api_params(token=api.get_account_token())
        project_name = api.get_user_default_project()
    else:

```

(continues on next page)

(continued from previous page)

```
        if args.project_name is None:
            sys.exit("Please provide the project name to login into")
        project_name = args.project_name
        api.reset_api_params(project_name=project_name)
        print(api.get_project_token())

if __name__ == '__main__':
    main()

|api-endpoints-list|
```



## AWS API POLICIES

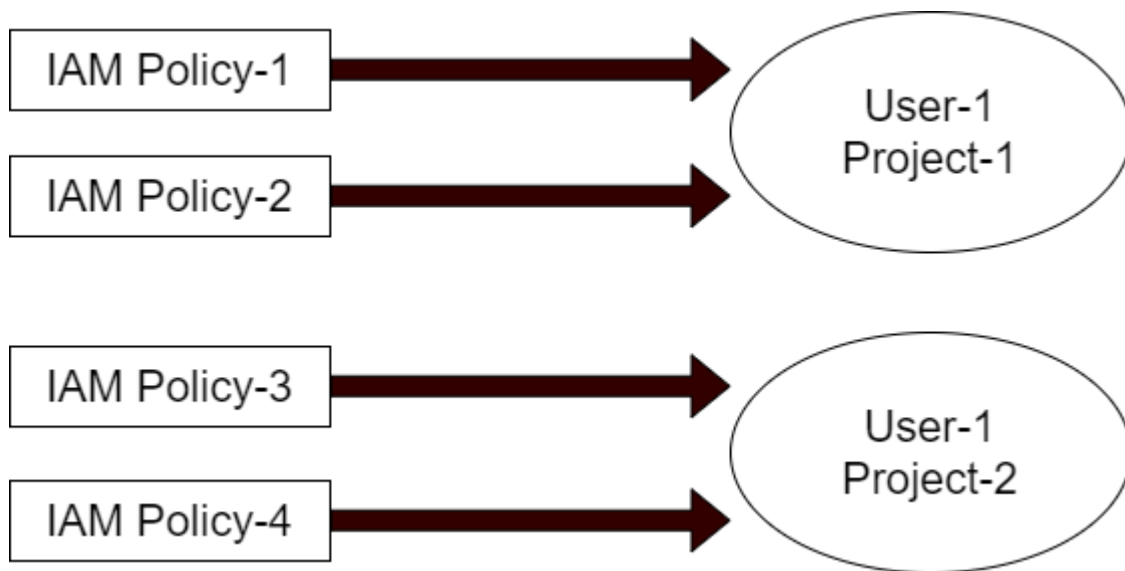
### 7.1 Introduction

Usage of all Zadara Cloud Services-supported AWS services and actions are governed by their corresponding AWS-managed policies. These policies can be assigned per project to users, groups of users, and STS (Security Token Service) Roles. Zadara Cloud Services usage is governed by Zadara IaaS (Infrastructure as a Service) policies together with Zadara Cloud Services roles. Zadara Cloud Services supports both AWS Managed Policies and Zadara Cloud Services Managed Policies.

### 7.2 AWS IAM API Policies and AWS Roles Overview

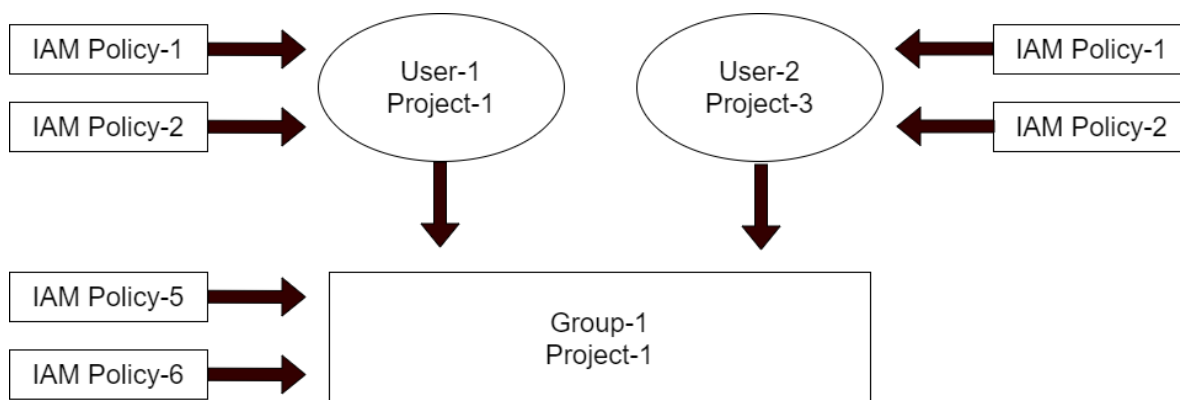
The following section provides different examples in the use of policies and roles.

1. Policies are attached to users, groups or IAM roles only within the context of a project.



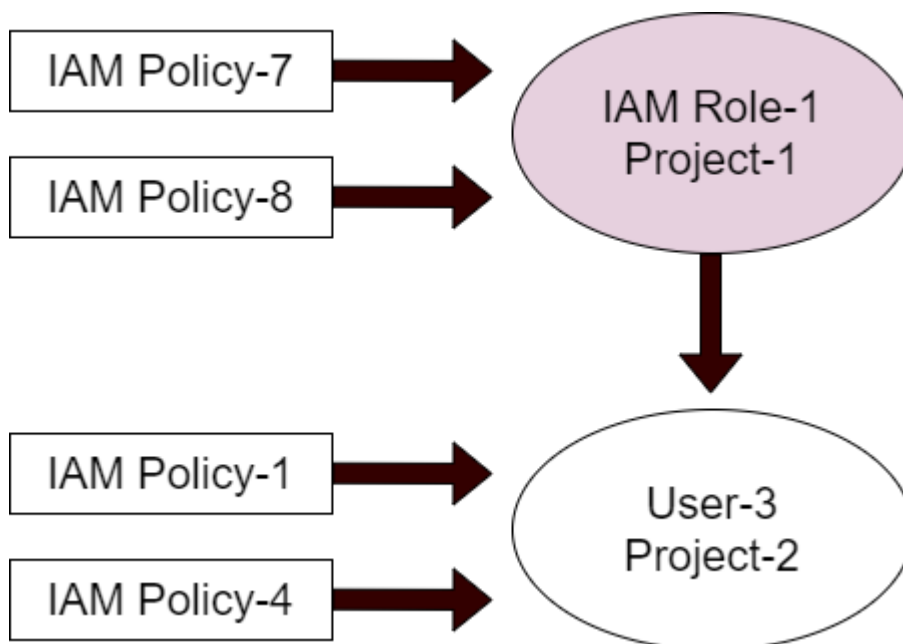
User-1 will have the permissions defined in IAM Policy 1 and IAM Policy-2 only when working within the context of Project-1. When working within the context of Project-2, User-1 will have the permissions defined in IAM Policy-3 and Policy-4.

2. Policies attached to users and groups within the context of the same project are aggregated.



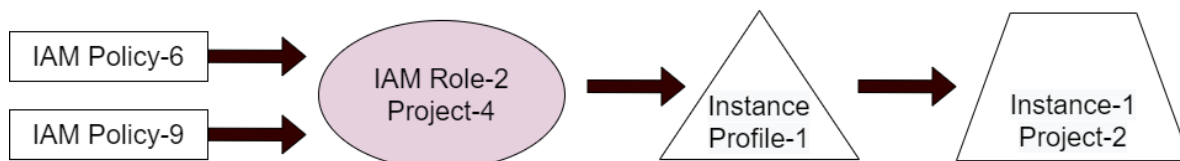
User-1 will have the permissions defined in IAM Policy-1 and IAM Policy-2 in addition to those defined in IAM Policy-5 and IAM Policy-6. On the other hand, User-2 will have the permissions defined in IAM Policy-5 and IAM Policy-6 when working within the context of Project-1. But when working within the context of Project-3, they will both have the permissions defined in IAM Policy-1 and Policy-2.

3. An IAM role with attached IAM policies which is added to a user, grants the user temporary credentials within the context of the IAM role's project, once assumed.



When User-3 assumes IAM Role-1, he will have the temporary permissions defined in IAM Policy-7 and IAM Policy-8, when working within the context of Project-1. When working within the context of Project-2, User-3 will have the permanent permissions defined in IAM Policy-1 and IAM Policy-4.

4. An IAM role with attached IAM policies which is attached to an Instance via an instance profile, grants the instance permanent credentials within the context of the IAM role's project.



Instance-1 will have the permissions defined in Policy-6 and Policy-9, permanently, when working within the context



of Project-4.

## 7.3 Managed AWS API Policies Supported by Zadara-IaaS

### 7.3.1 Zadara Cloud Services-managed AWS IAM API policies

Name	Description
Ama-zonS3BucketManagement	Provides the ability to create buckets and read their data.
EC2AMIDeleteOnly	Provides the ability to delete an EC2 AMI.
EC2AMIDescribeInstances	Provides the ability to describe instances (including their statuses and attributes), and to create and describe images.
EC2AMIFullAccess	Provides full access to all EC2 AMI actions.
EC2AMIReadOnlyAccess	Provides read only access to all EC2 AMI actions.
EC2ManageInstances	Provides read-only access to all EC2 (which includes EC2, VPC, EBS, VM Import/Export) actions, in addition to permission to start and stop EC2 instances.
MemberFullAccess	Provides limited access to IAM policies and full access to all other supported services.
STSAssumeRole	Provides the ability to obtain an IAM role using the “assume-role” action.
STSFULLAccess	Provides full access to all STS actions.

#### ZCS-managed policy definitions

#### JSON policy definition examples

Examples of JSON policy definitions for ZCS managed IAM policies:

#### EC2AMIDeleteOnly

JSON policy definition for the EC2AMIDeleteOnly ZCS managed IAM policy:

```
[
  {
    "Action": [
      "ec2:DeregisterImage"
    ],
    "Resource": [
      "*"
    ],
    "Effect": "Allow"
  }
]
```

### EC2AMIDescribeInstances

JSON policy definition for the EC2AMIDescribeInstances ZCS managed IAM policy:

```
[
  {
    "Action": [
      "ec2:DescribeImageAttribute",
      "ec2:DescribeImages",
      "ec2:CreateImage",
      "ec2:DescribeInstanceAttribute",
      "ec2:DescribeInstanceState",
      "ec2:DescribeInstances"
    ],
    "Resource": [
      "*"
    ],
    "Effect": "Allow"
  }
]
```

### EC2AMIFullAccess

JSON policy definition for the EC2AMIFullAccess Zadara IaaS-managed IAM policy:

```
[
  {
    "Action": [
      "ec2:*Tags",
      "ec2:*Image*"
    ],
    "Resource": [
      "*"
    ],
    "Effect": "Allow"
  }
]
```

### EC2AMIReadOnlyAccess

JSON policy definition for the EC2AMIReadOnlyAccess ZCS managed IAM policy:

```
[
  {
    "Action": [
      "ec2:Describe*Image*",
      "ec2:DescribeTags"
    ],
    "Resource": [
      "*"
    ],
    "Effect": "Allow"
  }
]
```

## EC2ManageInstances

JSON policy definition for the EC2ManageInstances ZCS managed IAM policy:

```
[
  {
    "Action": [
      "ec2:RebootInstances",
      "ec2:Describe*",
      "autoscaling:Describe*",
      "ec2:StartInstances",
      "ec2:DescribeTags",
      "elasticloadbalancing:Describe*",
      "ec2:StopInstances",
      "cloudwatch:GetMetricStatistics",
      "cloudwatch:Describe*",
      "cloudwatch:ListMetrics"
    ],
    "Resource": [
      "*"
    ],
    "Effect": "Allow"
  }
]
```

## MemberFullAccess

JSON policy definition for the MemberFullAccess ZCS managed IAM policy:

```
[
  {
    "NotAction": [
      "iam:*"
    ],
    "Resource": [
      "*"
    ],
    "Effect": "Allow"
  },
  {
    "Action": [
      "iam:ChangePassword",
      "iam:GetUser",
      "iam:CreateAccessKey",
      "iam:ListAccessKeys",
      "iam>DeleteAccessKey"
    ],
    "Resource": [
      "arn:aws:iam::*:user/${aws:username}"
    ],
    "Effect": "Allow"
  },
  {
    "Action": [
      "iam:GetRole",
      "iam:ListAttachedRolePolicies",

```

(continues on next page)

(continued from previous page)

```

        "iam:ListRoles",
        "iam:*InstanceProfile*",
        "iam:GetPolicy",
        "iam:GetPolicyVersion"
    ],
    "Resource": [
        "*"
    ],
    "Effect": "Allow"
},
]

```

## STSAssumeRole

JSON policy definition for the STSAssumeRole ZCS managed IAM policy:

```

[
  {
    "NotAction": [
      "iam:*"
    ],
    "Resource": [
      "*"
    ],
    "Effect": "Allow"
  },
  {
    "Action": [
      "iam:ChangePassword",
      "iam:GetUser",
      "iam:CreateAccessKey",
      "iam:ListAccessKeys",
      "iam>DeleteAccessKey"
    ],
    "Resource": [
      "arn:aws:iam::*:user/${aws:username}"
    ],
    "Effect": "Allow"
  },
  {
    "Action": [
      "iam:GetRole",
      "iam:ListAttachedRolePolicies",
      "iam:ListRoles",
      "iam:*InstanceProfile*",
      "iam:GetPolicy",
      "iam:GetPolicyVersion"
    ],
    "Resource": [
      "*"
    ],
    "Effect": "Allow"
  },
]

```

**STSTFullAccess**

JSON policy definition for the STSTFullAccess ZCS managed IAM policy:

```
[
  {
    "Action": [
      "sts:*"
    ],
    "Resource": [
      "*"
    ],
    "Effect": "Allow"
  }
]
```

### 7.3.2 AWS-managed IAM API policies supported by Zadara Cloud Services, with links to their JSON policy documents

Name	Description
AdministratorAccess	Provides full access to AWS services and resources.
AmazonDynamoDBFullAccess	Provides full access to Amazon DynamoDB via the AWS Management Console.
AmazonDynamoDBFullAccesswithDataPipeline	Provides full access to Amazon DynamoDB including Export/Import.
AmazonDynamoDBReadOnlyAccess	Provides read only access to Amazon DynamoDB via the AWS Management Console.
AmazonEC2ContainerRegistryFullAccess	Provides administrative access to Amazon ECR resources.
AmazonEC2ContainerRegistryPowerUser	Provides full access to Amazon EC2 Container Registry repositories, images, and tags.
AmazonEC2ContainerRegistryReadOnly	Provides read-only access to Amazon EC2 Container Registry repositories, images, and tags.
AmazonEC2FullAccess	Provides full access to Amazon EC2 via the AWS Management Console.
AmazonEC2ReadOnlyAccess	Provides read only access to Amazon EC2 via the AWS Management Console.
AmazonEC2ReportsAccess	Provides full access to all Amazon EC2 reports via the AWS Management Console.
AmazonEC2RoleforAWSCodeDeploy	Provides EC2 access to S3 bucket to download revision. This role is not managed by AWS.
AmazonEKSClusterPolicy	This policy provides Kubernetes the permissions it requires to manage the cluster.
AmazonEKSServicePolicy	This policy allows Amazon Elastic Container Service for Kubernetes to manage the cluster.
AmazonElasticFileSystemFullAccess	Provides full access to Amazon EFS via the AWS Management Console.
AmazonElasticFileSystemReadOnlyAccess	Provides read only access to Amazon EFS via the AWS Management Console.
AmazonElasticMapReduceEditorsRole	Default policy for the Amazon Elastic MapReduce Editors service role.
AmazonElasticMapReduceforAutoScalingRole	Amazon Elastic MapReduce for Auto Scaling. Role to allow Auto Scaling to manage EC2 instances.
AmazonElasticMapReduceforEC2Role	Default policy for the Amazon Elastic MapReduce for EC2 service role.
AmazonElasticMapReduceFullAccess	Provides full access to Amazon Elastic MapReduce and underlying services.
AmazonElasticMapReduceReadOnlyAccess	Provides read only access to Amazon Elastic MapReduce via the AWS Management Console.
AmazonElasticMapReduceRole	Default policy for the Amazon Elastic MapReduce service role.
AmazonEMRCleanupPolicy	Allows the actions that EMR requires to terminate and delete AWS EC2 instances.
AmazonRDSBetaServiceRolePolicy	Allows Amazon RDS to manage AWS resources on your behalf.
AmazonRDSDataFullAccess	Allows full access to use the RDS data APIs, secret store APIs for RDS.
AmazonRDSEnhancedMonitoringRole	Provides access to Cloudwatch for RDS Enhanced Monitoring.
AmazonRDSFullAccess	Provides full access to Amazon RDS via the AWS Management Console.
AmazonRDSPreviewServiceRolePolicy	Amazon RDS Preview Service Role Policy.
AmazonRDSReadOnlyAccess	Provides read only access to Amazon RDS via the AWS Management Console.
AmazonRDSServiceRolePolicy	Allows Amazon RDS to manage AWS resources on your behalf.
AmazonRoute53DomainsFullAccess	Provides full access to all Route53 Domains actions and Create Hosted Zones.

Name	Description
AmazonRoute53DomainsReadOnlyAccess	Provides access to Route53 Domains list and actions.
AmazonRoute53FullAccess	Provides full access to all Amazon Route 53 via the AWS Management Console.
AmazonRoute53ReadOnlyAccess	Provides read only access to all Amazon Route 53 via the AWS Management Console.
AmazonS3FullAccess	Provides full access to all buckets via the AWS Management Console.
AmazonS3ReadOnlyAccess	Provides read only access to all buckets via the AWS Management Console.
AmazonSNSFullAccess	Provides full access to Amazon SNS via the AWS Management Console.
AmazonSNSReadOnlyAccess	Provides read only access to Amazon SNS via the AWS Management Console.
AmazonSNSRole	Default policy for Amazon SNS service role.
AmazonSQSFullAccess	Provides full access to Amazon SQS via the AWS Management Console.
AmazonSQSReadOnlyAccess	Provides read only access to Amazon SQS via the AWS Management Console.
AmazonVPCCrossAccountNetworkInterfaceOperations	Provides access to create network interfaces and attach them to cross-account VPCs.
AmazonVPCFullAccess	Provides full access to Amazon VPC via the AWS Management Console.
AmazonVPCReadOnlyAccess	Provides read only access to Amazon VPC via the AWS Management Console.
AutoScalingConsoleFullAccess	Provides full access to Auto Scaling via the AWS Management Console.
AutoScalingConsoleReadOnlyAccess	Provides read-only access to Auto Scaling via the AWS Management Console.
AutoScalingFullAccess	Provides full access to Auto Scaling.
AutoScalingNotificationAccessRole	Default policy for the AutoScaling Notification Access service role.
AutoScalingReadOnlyAccess	Provides read-only access to Auto Scaling.
AutoScalingServiceRolePolicy	Enables access to AWS Services and Resources used or managed by Auto Scaling.
AWSAutoScalingPlansEC2AutoScalingPolicy	Policy granting permissions to AWS Auto Scaling to periodically forecast EC2 capacity.
AWSCertificateManagerFullAccess	Provides full access to AWS Certificate Manager (ACM).
AWSCertificateManagerReadOnly	Provides read only access to AWS Certificate Manager (ACM).
AWSElasticLoadBalancingServiceRolePolicy	Service Linked Role Policy for AWS Elastic Load Balancing Control Plane.
AWSKeyManagementServiceCustomKeyStoresServiceRolePolicy	Enables access to AWS services and resources required for AWS KMS Custom Key Stores.
AWSKeyManagementServicePowerUser	Provides access to AWS Key Management Service (KMS).
CloudWatchActionsEC2Access	Provides read-only access to CloudWatch alarms and metrics as well as EC2 instances.
CloudWatchFullAccess	Provides full access to CloudWatch.
CloudWatchReadOnlyAccess	Provides read only access to CloudWatch.
DatabaseAdministrator	Grants full access permissions to AWS services and actions required for Amazon RDS.
DynamoDBReplicationServiceRolePolicy	Permissions required by DynamoDB for cross-region data replication.
ElasticLoadBalancingFullAccess	Provides full access to Amazon ElasticLoadBalancing, and limited access to Amazon CloudWatch.
ElasticLoadBalancingReadOnly	Provides read only access to Amazon ElasticLoadBalancing and dependent services.
IAMFullAccess	Provides full access to IAM via the AWS Management Console.
IAMReadOnlyAccess	Provides read only access to IAM via the AWS Management Console.
IAMSelfManageServiceSpecificCredentials	Allows an IAM user to manage their own Service Specific Credentials.
IAMUserChangePassword	Provides the ability for an IAM user to change their own password.
IAMUserSSHKeys	Provides the ability for an IAM user to manage their own SSH keys.
NetworkAdministrator	Grants full access permissions to AWS services and actions required for Amazon VPC.
PowerUserAccess	Provides full access to AWS services and resources, but does not allow access to IAM.
RDSCloudHsmAuthorizationRole	Default policy for the Amazon RDS service role.
ReadOnlyAccess	Provides read-only access to AWS services and resources.
SecretsManagerReadWrite	Provides read/write access to AWS Secrets Manager via the AWS Management Console.
SecurityAudit	The security audit template grants access to read security configurations.
SystemAdministrator	Grants full access permissions necessary for resources required for Amazon VPC.
ViewOnlyAccess	This policy grants permissions to view resources and basic metadata.

## 7.4 Working with Managed AWS API Policies

### 7.4.1 Working with Managed AWS API Policies via the GUI

1. To retrieve the full list of ZCS-supported managed AWS API policies:
  1. Navigate to the **Identity & Access > AWS API Policies** view.
2. To display the JSON policy definition of a specific managed AWS API policy:
  1. In the **Identity & Access > AWS API Policies** view, click on the desired policy and select the **Policy** tab. The policy definition will be displayed.
3. To display all the users, groups and roles assigned to a specific managed AWS API policy:
  1. In the **Identity & Access > AWS API Policies** view, click on the desired policy and select the **Assignments** tab. A list of all of the users, groups and roles assigned to this policy will be displayed.

### 7.4.2 Working with Managed AWS API Policies via the CLI

1. To retrieve the entire list of Zadara Cloud Services-supported managed AWS API policies:

1. Enter the following command:

```
Zadara Cloud Services @ Account-1/Project-1 > policy list
```

The **list** of **all** Zadara Cloud Services-supported managed policies will be displayed, together **with** their IDs.

2. To display the JSON policy definition of a specific policy:

1. First locate the ID of the desired policy from the list of policies with the previous command:

```
Zadara Cloud Services @ Account-1/Project-1 > policy list
```

2. Using the ID of the desired policy, enter the following command to get its policy definition:

```
Zadara Cloud Services @ Account-1/Project-1 > policy get ced7e6aca00340bd84e396c71763c7d8
```

A variety of details about this policy including its policy definition will be displayed.

3. To display all of the users, groups and roles assigned to a specific policy:

1. First locate the ID of the desired policy from the list of policies with the previous command:

```
Zadara Cloud Services @ Account-1/Project-1 > policy list
```

2. Using the ID of the desired policy, enter the following command to get its policy definition:

```
Zadara Cloud Services @ Account-1/Project-1 > policy get -  
→entities ced7e6aca00340bd84e396c71763c7d8
```

All of the users, groups and roles attached to the selected policy, will be displayed.

4. To display all the assigned policies for all of users, groups and roles:

1. Enter the following command:

```
Zadara Cloud Services @ Account-1/Project-1 > policy assignments-list
```

All of the assigned policies for all users, groups and roles, will be displayed.



## AWS IAM ROLES AND INSTANCE PROFILES

### 8.1 AWS IAM Roles

AWS IAM Roles are policy-based tokens with temporary credentials allowing a user temporary access to AWS services and actions which the user is normally not permitted to access. These users may be from different projects or even different accounts. These roles can also be embedded into specific instances allowing these instances access to the necessary actions.

---

**Important:** The AWS IAM roles are independent of the Zadara Cloud Services roles, which together with Zadara Cloud Services policies, grant access to ZCS services and actions.

---

The AWS IAM role consists of the following:

1. Permissions policy which give access to certain ZCS supported AWS services or actions. This is defined in the **Identity & Access > AWS Roles** view, **Attached Policies** tab.
2. Trust policy that defines the relationship between user per project and this role. This is defined in the **Identity & Access > AWS Roles** view, **Trust Policy** tab.
  1. The nature of the relationship may be 'allow' which grants permission to the specified users to assume the role, or 'deny' which prevents these users from assuming the role.
  2. This permission may be granted to multiple users of the same projects, different projects within the same account, or even users of different accounts.
3. The maximum session duration that can be requested when assuming this role.

#### 8.1.1 Viewing the AWS IAM Role List

1. Navigate to the **Identity & Access > AWS Roles** view. The defined AWS IAM Role list will be displayed.
2. Select a specific role in the display list to view the following detailed information:
  1. **Events** - displays events related to the role.
  2. **Attached Policies** - displays AWS API policies attached to this role.
  3. **Trust Policy** - displays entities allowed/denied to assume this role.

### 8.1.2 Creating an AWS IAM Role

1. Navigate to the **Identity & Access > AWS Roles** view.
2. In top toolbar, click **Create**.
3. In the **Create Role** dialog, **Details** tab, enter the following:
  1. **Name** - name of the role AWS IAM Role to be defined.
  2. **Description** - description of role..
  3. **Max. Session Duration** - maximum time (in seconds) that a user can assume this role.
    - **Default:** 3600 seconds (1 hour).
    - **Maximum:** 12 hours (43,200 second).
  4. **Policies** - Select one or more managed AWS policies which define the permissions of this role.
  5. Click **Next**.
4. In the **Allow Assuming** tab, the trust policy is defined for this role. Multiple trust policies can be defined by clicking **Add**. For each trust policy, enter the following information:
  1. **User/Service/Any** - defines for what user or service the role will be applied.
    - If **User** is selected, enter the project name and user name.
    - If **Service** is selected, enter the service type. Currently, the only available service type for the trust policy is **VM**.
    - If **Any** is selected, enter project name. The trust policy will be applied to all users.
  2. Click **Next**.
5. In the **Deny Assuming** tab, users or services who can not assume this role may be defined. This is an optional configuration and by default there are no denied users or services. If defined, the same **user/service/any** options described for the **allow assuming** step above are also available for the **deny assuming** step. **Note:** If the same user and project are defined for both **allow** and **deny**, the 'Deny' prevails.
6. Click **Finish**. The new role will appear in the **Identity & Access > AWS Roles** view.

### 8.1.3 Deleting an AWS IAM Role

---

**Important:** A role cannot be deleted if policies are attached to it.

---

1. Navigate to the **Identity & Access > AWS Roles** view.
2. From the displayed list, select the role to be deleted.
3. From top toolbar, click **Delete**. The **Delete Role** dialog will open.
4. Click **Delete** to confirm. The role will be removed from the **Identity & Access > AWS Roles** list.

### 8.1.4 Modifying an AWS IAM Role

1. Navigate to the **Identity & Access > AWS Roles** view.
2. From the displayed list, select the role to be modified.
3. From top toolbar, click **Modify**. The **Modify Role** dialog will be displayed. The following parameters can be modified:
  1. **Max. Session Duration** - maximum time (in seconds) that a user can assume this role.
  2. **Policies** - managed policies which define the permissions of this role.
  3. **Description**
4. Click **OK** to save the changes.

✓ **Note:** The following can not be modified:

1. The project for which this role is defined.
2. The users who may assume this rule.
3. The users who are prevented from assuming this role.

### 8.1.5 Applying an AWS IAM Role via the CLI

When an AWS IAM role has been created and a policy attached, the role can be applied via the Zadara Cloud Services CLI. This in turn will generate the credentials needed to access the AWS services and actions defined in the role. In order to assume a role you must know its ID.

1. Find the AWS IAM roles you want to use:

1. Run role iam-list:

```
Zadara Cloud Services @ user1/cloudacc1 > role iam-list
```

If you are a Member or Tenant Admin user, this command returns a list of any roles created in the currently logged-in project, together with the ID's of all users who can assume these role(s).

2. To check if the entity\_ids for any of the users is your own, use the following command:

```
Zadara Cloud Services @ user1/cloudacc1 > user get-my-details
```

This will display your user\_id along with your user\_name.

3. Regardless of AWS IAM roles discovered, there may be roles available which are not visible to you. To verify that you have the complete list of roles available to you, it is recommended that a Zadara Admin user does the following:

- List all of the AWS IAM Roles

```
Zadara Cloud Services @ user1/cloudacc1 > role iam-list
```

All AWS IAM roles in the account will be displayed with the ID's of all users to which these roles are available.

- List all of the users

```
Zadara Cloud Services @ user1/cloudacc1 > user list
```

This will display all of the users in the account with their names and ID's. It will now be possible to discover which roles are available for which users.

2. Once you have the Role-ID, you can assume the AWS IAM role via the 'role assume-role **role\_id session\_name**' command, where **role\_id** is supplied in the output above, and **session\_name** is selected by the user.

```
Zadara Cloud Services @ user1/cloudacc1 > role assume-role 565197da-2f13-48dc-b232-bdfffc756b7f9
↩Session-1
```

This returns the following information:

```
=====
**access_key_id**      23515d96a5da4408821783e0b9aa6ff1
**created_at**         2019-03-10T20:55:42Z
**duration_seconds**   3600
**expires_at**         2019-03-10T21:55:42Z
**external_id**        none
**policy_id**          none
**project_id**         fc268815422e471da6756c7918b03d01
**role_assumer**       5120ef807769455b822095497b55ffac
**role_id**            565197da-2f13-48dc-b232-bdfffc756b7f9
**role_name**          Role-2
**secret_access_key**  94d8a61f419b4edbb638abc399e7a420
**session_name**       Session-1
**token**              cd9b0253e0034cdd976c21c317c
=====
```

3. Use the **access\_key\_id**, **secret\_access\_key** and **token** to access the AWS services and actions.

## 8.2 Instance Profiles

An instance profile is a container for an AWS IAM role. It can be used to pass role information to an EC2 instance when the instance starts. When an AWS IAM role, embedded in an instance profile, becomes attached to an instance, its credentials become permanent.

The following actions can be performed with instance profiles:

## 8.2.1 Viewing Instance Profiles

1. In the zCompute UI, go to **Compute > Instance Profiles**.

The instance profiles list is displayed.

2. Click an instance profile in the list to view its details.

The detailed instance profile view is displayed, with the option to modify the instance profile by removing its associated role, and the option to delete the instance profile.

## 8.2.2 Creating an Instance Profile

To create an instance profile:

1. In the zCompute UI, go to **Compute > Instance Profiles**.

The instance profiles list is displayed.

2. In the top menu bar, click **Create**.

The **Create Instance Profile** dialog opens.

3. In the **Create Instance Profile** dialog, enter:

- **Name** - a name to identify the instance profile.
- **Role** - optionally select an AWS IAM role from the dropdown.
- Click **Finish**.

The new instance profile is displayed in the instance profiles list.

## 8.2.3 Modifying an Instance Profile

An instance profile can be modified by changing its AWS IAM role. This involves removing the current AWS IAM role from instance profile, and adding a new role.

### Removing an AWS IAM Role

1. In the zCompute UI, go to **Compute > Instance Profiles**.

The instance profiles list is displayed.

2. To modify an instance profile, click its entry in the list.

The detailed instance profile view is displayed.

3. In the top menu bar, click **Remove Role**.

The **Remove Role from Instance Profile** confirmation dialog opens.

4. Click **Delete** to remove the AWS IAM role from the instance profile.

## Adding an AWS IAM Role

1. In the zCompute UI, go to **Compute > Instance Profiles**.  
The instance profiles list is displayed.
2. To modify an instance profile, click its entry in the list.  
The detailed instance profile view is displayed.
3. In the top menu bar, click **Add Role**.  
The **Add Role to Instance Profile** dialog opens.
4. Select an AWS IAM role from the dropdown, to add to the instance profile.
5. Click **Ok**.

## 8.2.4 Working with Instance Profiles via the CLI

### Adding an AWS IAM Role into an Instance

Using instance profiles, it is possible to grant permissions to an instance to access specific Zadara Cloud Services-supported AWS services. Although, in the context of a user, role permissions are temporary, in the context of an instance, the application is guaranteed to have credentials as long as the instance profile is attached to the instance, and that instance profile has a role embedded in it.

1. From the Zadara Cloud Services GUI, create an AWS IAM role together with its AWS IAM policies and trust policies.

✓ **Note:** This must be performed by a Zadara Admin or a Tenant Admin user.

2. To create an instance profile, use the Zadara Cloud Services CLI command: 'instance-profile create **name**'.

✓ **Note:** This command may be performed by any user.

```
Zadara Cloud Services @ user1/cloudacc1 > instance-profile create instance-profile-1
```

To view basic details about the newly created instance-profile, use command instance-profile -l.

```
=====
**id**          dd56d017-0167-42b7-a130-8706d746493e
**name**        instance-profile-1
**created_at**  2019-03-11T02:58:30Z
**path**        /
**project_id**  4331358dff9b4c29aa53c982e92801f6
**roles**       []
=====
```

3. Find the AWS IAM roles that you can embed in the instance profile as follows:

1. Run role iam-list:

```
Zadara Cloud Services @ user1/cloudacc1 > role iam-list
```

If you are a Member or Tenant Admin user, the above command returns a list of any roles created in the currently logged-in project, together with the ID's of all users who can assume these role(s), as shown below.

✓ **Note:** If you are assigned to more than one project, you must login to each project and run 'role iam-list' to get the complete list of roles available to you.

- To check if the entity\_ids for any of the users is yours run the following command:

```
Zadara Cloud Services @ user1/cloudacc1 > user get-my-details
```

This will display your user\_id along with your user\_name.

- Embed the role in the instance profile with the Zadara Cloud Services CLI command 'instance-profile add-role **instance\_profile-id role\_id**', as follows: (Take the instance\_profile\_id and the role\_id from the steps above.) **Note:** This command can be performed by a Member or Tenant admin if the project of the role is the same as the logged-in project.

```
Zadara Cloud Services @ user1/cloudacc1 > instance-profile add-role **dd56d017-0167-42b7-a130-8706d746493e** **565197da-2f13-48dc-b232-bdffcf756b7f9**
```

- Use the GUI to create an instance.

✓ **Note:** If you create this instance from the CLI, you can add the instance-profile on creation. This will remove the need for steps 6 and 8.

- From the Zadara Cloud Services CLI using the CLI command "vm list" locate the ID of the instance that you just created.
- Since the instance profile operation is very sensitive to network latency and cluster load, the system may time out before finishing the operation. It is therefore recommended to increase the timeout on the metadata service connection and/or allow retries. These can be configured inside the VM that is connected to the instance-profile by setting the following environment vars:

```
AWS_METADATA_SERVICE_TIMEOUT >1
AWS_METADATA_SERVICE_NUM_ATTEMPTS >1
```

- Using the Zadara Cloud Services CLI command 'vm update -instance-profile **INSTANCE\_PROFILE vm\_id**' attach the instance profile to the instance you just created, as follows: (Take the instance\_profile\_id and the vm\_id from the steps above.) **Note:** The role and VM must be defined for the same project. A Member user and Tenant Admin user can perform this command if they are logged in to the same project as that of the role and VM.

```
Zadara Cloud Services @ user1/cloudacc1 > vm update -instance-profile **dd56d017-0167-42b7-a130-8706d746493e d6ca69e7-1533-4740-9881-395d442719f5**
```

## Instance Profiles CLI commands

- To create an instance profile enter the following command:

```
Zadara Cloud Services @ Account-1/Project-1 > instance-profile get <name of instance-profile>
```

A variety of details about this instance profile will be displayed, including it's ID

- To add an AWS IAM role to an instance profile:

- Procure the role-id via the following command

```
Zadara Cloud Services @ Account-1/Project-1 > role iam-list
```

The list of all IAM roles will be displayed, together with their ID's.

2. Using the ID of the desired instance profile, returned from the first command above, enter the following command:

```
Zadara Cloud Services @ Account-1/Project-1 > instance-profile add-role <instance_profile_id>  
↪<role_id>
```

Only one role can be added to an instance profile.

3. To remove an AWS IAM role from an instance profile, enter the following command:

```
Zadara Cloud Services @ Account-1/Project-1 > instance-profile remove-role <instance_profile_id>  
↪<role_id>
```

4. To retrieve the entire list of Instance profiles in the cluster enter the following command:

```
Zadara Cloud Services @ Account-1/Project-1 > instance-profile list
```

The list of all instance profiles will be displayed, together with their ID's.

5. To remove an Instance profile:

Using the ID of the desired instance profile retrieved above, enter the following command:

```
Zadara Cloud Services @ Account-1/Project-1 > instance-profile remove <instance_profile_id>
```

6. To retrieve information about a specific instance profile:

Using the ID of the desired instance profile retrieved above, enter the following command:

```
Zadara Cloud Services @ Account-1/Project-1 > instance-profile get <instance_profile_id>
```

A variety of details about this instance profile including its ID, will be displayed.



## ZADARA CLOUD SERVICES POLICIES

### 9.1 Overview

Zadara Cloud Services Policies define the permissions for accessing Zadara Cloud Services functionality. For each project available to a user, one or more Zadara Cloud Services Policies and a single Zadara Cloud Services Role are assigned. If the role of the user does not match the role type of any of the APIs included in the policy, then the user cannot perform these APIs, even if the policy which includes these APIs was assigned to the user.

Whenever a new user is created via the GUI, a FullAccess policy is the suggested default Zadara Cloud Services policy. When a new user is created via the API/CLI, FullAccess is automatically assigned Zadara Cloud Services policy. This policy provides full access to all of the services that each Zadara Cloud Services role granted in the past, prior to v5.3.4.

---

✓ **Note:** In the ZCS GUI, Zadara Cloud Services policy is shown in the **Identity & Access > Symp API Policies** view.

---

### 9.2 Working with Zadara Cloud Services Policies

Zadara Cloud Services policies can be assigned while creating a user on the Permissions tab of the Create User wizard, or on the Manage Permissions dialog for an existing user. For more information on assigning policies to users, see [Creating Users](#).

---

✓ **Note:** If a Zadara Cloud Services policy is assigned to a user while the user is in the middle of a session, it will not take effect until the user first logs out of the current session and then logs in again to a new session.

---

#### 9.2.1 Assigning Permissions for Creating a VM

As an example of how to use policies to allow certain functionality, the following section explains how to use Zadara Cloud Services policies to grant the permissions needed by a user to create a VM.

1. A Member user with only VMFullAccess rights, grants permission to create a VM and select an Instance Type, but there are no permissions to view the following entities:

1. **Image**
2. **Storage Pool**
3. **Subnet**

2. A Member user with legacy permissions providing full access to all Zadara Cloud Services member functionality, can create a VM. All of the fields in the Create VM dialog, including Image, Storage Pool, and Subnet are available to these users.
3. A Member user assigned the VMFullAccess Zadara Cloud Services policy together with the ZadaraReadOnlyAccess policy can also create a VM.
  1. The VMFullAccess Zadara Cloud Services policy permits the creation of a VM and the selection of any Instant Type.
  2. The ZadaraReadOnlyAccess Zadara Cloud Services policy permits the viewing and selecting of any entity available to the legacy Member user, including the following:
    - **Images**
    - **Storage Pools**
    - **Subnets**
4. A Member user assigned the VMFullAccess Zadara Cloud Services policy together with the following read-only Zadara Cloud Services policies can also create a VM:
  1. The VMFullAccess Zadara Cloud Services policy permits the creation of a VM and the selection of an Instant Type.
  2. The ImagesReadOnlyAccess Zadara Cloud Services policy permits the viewing and selecting of Images.
  3. The StorageReadOnlyAccess Zadara Cloud Services policy permits the viewing and selecting of Storage Pools.
  4. The VPCReadOnlyAccess Zadara Cloud Services policy permits the viewing and selecting of Subnets.

### 9.2.2 Assigning Permissions for Multiple Projects to a Single User

The following section explains how to use Zadara Cloud Services policies to assign multiple projects to a single user. This can be done when creating the user or by managing permissions of an existing user. For example, to assign permissions for only Zadara Cloud Services functionality (but not AWS API functionality) for each of three different projects, the configurations below should be made.

**Projects:**

- Project-1 - Full legacy Member role permissions
- Project-2 - Permissions to create VMs and many other actions concerning VMs
- Project-3 - To be determined

**Manage Permissions for user: Member-1**

+ Add Project *All projects have been added*

**Project: Project-1**

Roles: Member

Policies: FullAccess

**Project: Project-2**

Roles: Member

Policies: StratoReadOnlyAccess, VMFullAccess

**Project: Project-3**

Roles: Member

Policies: No policies assigned

Cancel Finish

In the **Manage Permissions for user** view, the following information should be configured:

1. **Roles** - Verify that the **Member** role has been assigned to each project.

✓ **Note:** It is strongly recommended not to assign different roles to the same user for different projects.

2. **Policies** - Each Project should be assigned the following Zadara Cloud Services policies:

1. Project-1 - Select the **FullAccess** policy to grant Member legacy permissions.
2. Project-2 - Select the **VMFullAccess** and **StratoReadOnlyAccess** to grant permissions to create a VM, including viewing Images, Storage Pools and Subnets, access to which is necessary when creating a VM.
3. Project-3 - Leave empty until User-1's responsibilities for this project have been determined.

✓ **Note:** As long as at least one project has both assigned a Zadara Cloud Services Role and Policy, the user's permissions for other projects need not be completely defined.

3. **AWS API policies** - Verify the **MemberFullAccess** policy is removed from each of the projects.
4. Click **Finish**.

### 9.2.3 List of Managed Zadara Cloud Services Policies

Zadara Cloud Services Policies basically consist of two policies per service, one for **FullAccess** of all the service's APIs, and one for **ReadOnly access** of all the service's APIs.

1. **FullAccess** – a policy which provides full access to all Zadara Cloud Services APIs based on the user's role.
2. **ReadOnlyAccess** – a policy which provides read-only access to all Zadara Cloud Services APIs based on the user's role.

Exceptions to this rule are:

1. The Identity Service which has two additional policies:
  1. **IdentityBasicUsage** – which provides access to those identity entities available to a Member user.
  2. **IdentitySTSAssumeRole** – which provides the ability to obtain an IAM role using the 'assume-role' API.
2. The Snapshot service which has three sets of FullAccess and ReadOnlyAccess policies, one set for each of the following:
  1. Snapshot
  2. RemoteSnapshot
  3. RemoteVMSnapshot
3. The VM service which has two sets of FullAccess and ReadOnlyAccess policies, one set for each of the following:
  1. VM
  2. VMSnapshot

The table below summarizes all Zadara Cloud Services Policies with their descriptions and the roles for which they are enforced. To see the actual content of a policy go to the **Identity & Access > Symp API Policies** view, and click on the requested policy.

	Zadara Cloud Services Policy	Description	Available to:
1.	AlarmsFullAccess	Provides full access to all alarms APIs	All ZCS roles
2.	AlarmsReadOnlyAccess	Provides read-only access to all alarms APIs	All ZCS roles
3.	AutoScalingFullAccess	Provides full access to all autoscaling-group APIs	All ZCS roles
4.	AutoScalingReadOnlyAccess	Provides read-only access to all autoscaling-group APIs	All ZCS roles
5.	CRSFullAccess	Provides full access to all container registry APIs	All ZCS roles
6.	CRSReadOnlyAccess	Provides read-only access to all container registry APIs	All ZCS roles

continues on next page

Table 1 – continued from previous page

	<b>Zadara Cloud Services Policy</b>	<b>Description</b>	<b>Available to:</b>
7.	CertificateManagerFullAccess	Provides full access to all certificate APIs	All ZCS roles
8.	CertificateManagerRead-OnlyAccess	Provides read-only access to all certificate APIs	All ZCS roles
9.	CloudWatchFullAccess	Provides full access to all cloudwatch APIs	All ZCS roles
10.	CloudWatchReadOnlyAccess	Provides read-only access to all cloudwatch APIs	All ZCS roles
11.	ConversionsFullAccess	Provides full access to all conversion APIs	All ZCS roles
12.	ConversionsReadOnlyAccess	Provides read-only access to all conversion APIs	All ZCS roles
13.	DBCFullAccess	Provides full access to all DB cluster APIs	All ZCS roles
14.	DBCReadOnlyAccess	Provides read-only access to all DB cluster APIs	All ZCS roles
15.	DBSFullAccess	Provides full access to all DB APIs	All ZCS roles
16.	DBSReadOnlyAccess	Provides read-only access to all DB APIs	All ZCS roles
17.	EngineFullAccess	Provides full access to all engine APIs	All ZCS roles
18.	EngineReadOnlyAccess	Provides read-only access to all engine APIs	All ZCS roles
19.	ExternalEndpointFullAccess	Provides full access to all external-endpoint APIs	All ZCS roles
20.	ExternalEndpointRead-OnlyAccess	Provides read-only access to all external-endpoint APIs	All ZCS roles
21.	EventsAccess	Provides access to all events APIs	All ZCS roles
22.	FullAccess	Provides full access to all Zadara Cloud Services APIs based on user's scope	All ZCS roles

continues on next page

Table 1 – continued from previous page

	<b>Zadara Cloud Services Policy</b>	<b>Description</b>	<b>Available to:</b>
23.	GuestnetToolFullAccess	Provides full access to all guestnet-admin-tool APIs	All ZCS roles
24.	GuestnetToolReadOnlyAccess	Provides read-only access to all guestnet-admin-tool APIs	All ZCS roles
25.	HotUpgradeFullAccess	Provides full access to all hot-upgrade APIs	Admin
26.	HotUpgradeReadOnlyAccess	Provides read-only access to all hot-upgrade APIs	Admin
27.	IdentityBasicUsage	Provides access to basic identity entities operations.	All ZCS roles
28.	IdentityFullAccess	Provides full access to identity APIs	All ZCS roles
29.	IdentityReadOnlyAccess	Provides read-only access to identity APIs	All ZCS roles
30.	IdentitySTSAssumeRole	Provides the ability to obtain an IAM role using the 'assume-role' API	All ZCS roles
31.	ImagesFullAccess	Provides full access to all image APIs	All ZCS roles
32.	ImagesReadOnlyAccess	Provides read-only access to all image APIs	All ZCS roles
33.	InspectorFullAccess	Provides full access to all inspector APIs	Admin
34.	InspectorReadOnlyAccess	Provides read-only access to all inspector APIs	Admin
35.	KubernetesFullAccess	Provides full access to all Kubernetes APIs	All ZCS roles
36.	KubernetesReadOnlyAccess	Provides read-only access to all Kubernetes APIs	All ZCS roles
37.	LbaasFullAccess	Provides full access to all LBaaS APIs	All ZCS roles
38.	LbaasReadOnlyAccess	Provides read-only access to all LBaaS APIs	All ZCS roles

continues on next page

Table 1 – continued from previous page

	<b>Zadara Cloud Services Policy</b>	<b>Description</b>	<b>Available to:</b>
39.	MapReduceFullAccess	Provides full access to all map-reduce APIs	All ZCS roles
40.	MapReduceReadOnlyAccess	Provides read-only access to all map-reduce APIs	All ZCS roles
41.	MetricsAccess	Provides access to all metrics APIs	All ZCS roles
42.	NFSFullAccess	Provides full access to all NFS APIs	All ZCS roles
43.	NFSReadOnlyAccess	Provides read-only access to all NFS APIs	All ZCS roles
44.	NodesFullAccess	Provides full access to all node APIs	Admin
45.	NodesReadOnlyAccess	Provides read-only access to all node APIs	Admin
46.	ObjectStoresFullAccess	Provides full access to all object-store APIs	All ZCS roles
47.	ObjectStoresReadOnlyAccess	Provides read-only access to all object-store APIs	All ZCS roles
48.	ProtectionFullAccess	Provides full access to all protection APIs	All ZCS roles
49.	ProtectionReadOnlyAccess	Provides read-only access to all protection APIs	All ZCS roles
50.	QuotasFullAccess	Provides full access to all quota APIs	All ZCS roles
51.	QuotasReadOnlyAccess	Provides read-only access to all quota APIs	All ZCS roles
52.	RemoteSnapshotFullAccess	Provides full access to all remote-snapshot APIs	All ZCS roles
53.	RemoteSnapshotReadOnlyAccess	Provides read-only access to all remote-snapshot APIs	All ZCS roles
54.	RemoteVMSnapshotFullAccess	Provides full access to remote VM Snapshot APIs	All ZCS roles

continues on next page

Table 1 – continued from previous page

	<b>Zadara Cloud Services Policy</b>	<b>Description</b>	<b>Available to:</b>
55.	RemoteVMSnapshotRead-OnlyAccess	Provides read-only access to remote VM Snapshot APIs	All ZCS roles
56.	Route53FullAccess	Provides full access to all Route53 APIs	All ZCS roles
57.	Route53ReadOnlyAccess	Provides read-only access to all Route53 APIs	All ZCS roles
58.	SnapshotFullAccess	Provides full access to all local compute-snapshot APIs	All ZCS roles
59.	SnapshotReadOnlyAccess	Provides read-only access to all local compute-snapshot APIs	All ZCS roles
60.	StorageFullAccess	Provides full access to all storage APIs	Admin
61.	StorageReadOnlyAccess	Provides read-only access to all storage APIs	All ZCS roles
62.	StratoReadOnlyAccess	Provides read-only access to all Zadara Cloud Services APIs based on user's scope	All ZCS roles
63.	VMFullAccess	Provides full access to VM APIs	All ZCS roles
64.	VMReadOnlyAccess	Provides read-only access to VM APIs	All ZCS roles
65.	VMSnapshotFullAccess	Provides full access to VM Snapshot APIs	All ZCS roles
66.	VMSnapshotReadOnlyAccess	Provides read-only access to VM Snapshot APIs	All ZCS roles
67.	VPCFullAccess	Provides full access to all VPC APIs	All ZCS roles
68.	VPCReadOnlyAccess	Provides read-only access to all VPC APIs	All ZCS roles
69.	VolumesFullAccess	Provides full access to all volume APIs	All ZCS roles
70.	VolumesReadOnlyAccess	Provides read-only access to all volume APIs	All ZCS roles



**Notes:**

1. It is currently possible to assign a Zadara Cloud Services policy to a user, for whose role, the APIs in this policy will not be permitted. For example, it is possible to assign the NodesFullAccess policy to Member users, even though the APIs in this policy will not be permitted for Members. In a future release, this invalid assignment will be prevented.
2. A Zadara Cloud Services policy which is permitted for a Member or Tenant Admin role, may nevertheless include some APIs/CLIs which are not permitted for that role. For example, the VMFullAccess policy whose APIs/CLIs are permitted for all roles, including the Member role, contains the API, 'vm live-migrate' which migrates VMs from one node to another. This action can be performed only by a Zadara Admin user. To determine the role for which specific APIs of a Zadara Cloud Services policy are permitted, access the API Explorer for the service covered by the policy and open the API, as shown below.

Swagger  
powered by SMARTHEAD

/api-explorer/api/v2/autoscaling-groups/swagger.json Explore

## Autoscaling Groups <sup>1.0.0</sup>

[ Base URL: 10.16.128.10/api-explorer/api/v2/autoscaling-groups ]  
api-explorer/api/v2/autoscaling-groups/swagger.json

1. To gain authorization to try out the APIs on your cluster, click **Authorize** and paste in the auth-token.
2. To view the documentation of a specific action expand it.
3. To try out an action, click **Try it out** and after entering the values of the requested Parameters click **Execute**.
4. To download the documentation, click the link above.

Schemes HTTPS Authorize

**default**

**GET** **/config** List all autoscaling groups configuration values

List all autoscaling groups configuration values.

**Parameters** Try it out

No parameters

**Responses** Response content type: application/json

Code	Description
200	default response
401	Invalid credentials
403	No permissions
500	Internal error

**Extensions**

Field	Value
x-cli	"autoscaling-groups config list"
x-scope	"member"

3. In the Extensions section at the bottom of the window, the x-cli field displays the name of the CLI while the x-scope field displays the role with the least amount of permissions which may use this CLI. (e.g. 'member' has less permissions than 'tenant\_admin'). If x-scope = 'all' permissions are not required to perform the CLI.



## AWS IDENTITY-AND-ACCESS MANAGEMENT (IAM)

AWS API Reference	Ignored Param	Optional Parameters	Required Parameters
AddRoleToInstanceProfile	[]	[]	InstanceProfileName
AddUserToGroup	[]	[]	GroupName UserName
AttachGroupPolicy	[]	[]	GroupName PolicyArn
AttachRolePolicy	[]	[]	PolicyArn RoleName
AttachUserPolicy	[]	[]	PolicyArn UserName
ChangePassword	[]	[]	NewPassword OldPassword
CreateAccessKey	[]	UserName	[]
CreateGroup	[]	Path	GroupName
CreateInstanceProfile	[]	Path	InstanceProfileName
CreateLoginProfile	[]	PasswordResetRequired	UserName Password
CreatePolicy	[]	Description Path	PolicyDocument PolicyName
CreateRole	[]	Description MaxSessionDuration Path	AssumeRolePolicyDocument RoleName
CreateUser	[]	Path	UserName
DeleteAccessKey	[]	UserName	AccessKeyId
DeleteGroup	[]	[]	GroupName
DeleteInstanceProfile	[]	[]	InstanceProfileName
DeleteLoginProfile	[]	[]	UserName
DeletePolicy	[]	[]	PolicyArn
DeleteRole	[]	[]	RoleName
DeleteUser	[]	[]	UserName
DetachGroupPolicy	[]	[]	GroupName PolicyArn
DetachRolePolicy	[]	[]	RoleName PolicyArn
DetachUserPolicy	[]	[]	PolicyArn UserName
GetGroup	[]	Marker MaxItems	GroupName
GetInstanceProfile	[]	[]	InstanceProfileName
GetLoginProfile	[]	[]	UserName
GetPolicy	[]	[]	PolicyArn
GetPolicyVersion	[]	[]	PolicyArn VersionId
GetRole	[]	[]	RoleName
GetUser	[]	UserName	[]
ListAccessKeys	[]	Marker MaxItems UserName	[]
ListAttachedGroupPolicies	[]	Marker MaxItems PathPrefix	GroupName
ListAttachedRolePolicies	[]	Marker MaxItems PathPrefix	RoleName
ListAttachedUserPolicies	[]	Marker MaxItems PathPrefix	UserName
ListEntitiesForPolicy	[]	EntityFilter Marker MaxItems PathPrefix	PolicyArn
ListGroups	[]	Marker MaxItems PathPrefix	[]
ListGroupsForUser	[]	Marker MaxItems	UserName

Table 1 – continued from previous page

AWS API Reference	Ignored Param	Optional Parameters	Required Parameters
ListInstanceProfiles	[]	Marker MaxItems PathPrefix	[]
ListInstanceProfilesForRole	[]	[]	RoleName
ListMFADevices	[]	Marker MaxItems UserName	[]
ListPolicies	[]	Marker MaxItems OnlyAttached PathPrefix Scope	[]
ListRoles	[]	Marker MaxItems PathPrefix	[]
ListUsers	[]	Marker MaxItems PathPrefix	[]
RemoveRoleFromInstanceProfile	[]	[]	InstanceProfileName
RemoveUserFromGroup	[]	[]	GroupName UserName
UpdateAccessKey	[]	UserName	AccessKeyId Status
UpdateAssumeRolePolicy	[]	[]	RoleName PolicyDoc
UpdateGroup	[]	NewGroupName NewPath	GroupName
UpdateLoginProfile	PasswordResetRequired	Password PasswordResetRequired	UserName
UpdateRole	[]	Description MaxSessionDuration	RoleName
UpdateRoleDescription	[]	[]	RoleName Description
UpdateUser	[]	NewPath NewUserName	UserName

## 10.1 AWS-STS

AWS API Reference	Ignored Param	Optional Parameters	Required Parameters	Unsupported Params
AssumeRole	[]	DurationSeconds	RoleArn RoleSession-Name	ExternalId Policy SerialNumber TokenCode
GetCallerIdentity	[]	[]	[]	[]