# Integrating SailPoint with Azure Active Directory Connector

Version: 8.1 Patch 4

## Copyright and Trademark Notices

# Contents

# Integrating SailPoint with Azure Active Directory

SailPoint Azure Active Directory manages the users and groups in Azure Active Directory. Azure Active Directory is the directory for all cloud based organizational Microsoft Directory services including Microsoft Office 365.

SailPoint Azure Active Directory Connector:

- can also be used to provision users into a federated domain in Azure Active Directory.
- uses **Retry-After** value to retry API request. It uses exponential back-off if **Retry-After** value is not returned by API.
- uses **Microsoft Graph API**'s to manage users, groups and licenses.

Azure is Microsoft’s cloud solution platform which provides plenty of cloud services like IaaS , PaaS , SaaS and so on. Azure uses Azure Active Directory as it authentication source to provide access on different services to users. Azure has management container objects which are used to group resources and manage access to them.

Azure Active Directory connector provides support for access management of Azure Management Objects along with managing of the Azure Active Directory Management Objects.

## Supported Features

The Azure Active Directory Connector supports the following features:

- Account Management for User, User in Federated Domain, and Guest User (B2B)
- Account Management for Local User (B2C)
- OAuth 2.0 Authentication
- Azure Endpoint Configuration
- Account - Group Management
- Group Management for Azure Management Objects
- Microsoft Teams
- Service Principal Management
- Channel Management
- Exchange Online Mailbox Management
- Risky User Alert Feature
- Azure Privileged Identity Management (PIM)

### Account Management for User, User in Federated Domain, and Guest User (B2B)

This section describes the supported features for User, User in federated domain, and Guest User (B2B).

| **Operations** | **User** | **User in federated domain** | **Guest User (B2B)** |
|---|---|---|---|
| Aggregation | Yes | Yes | Yes |
| Delta Aggregation | Yes | Yes | Yes |
| Partitioning Aggregation | Yes | Yes | Yes |
| Get Account | Yes | Yes | Yes |
| Pass Through Authentication | Yes | No | Yes |
| Create | Yes | Yes | Yes |
| Update | Yes | No | Yes |
| Delete | Yes | No | Yes |
| Enable/Disable Users | Yes | No | Yes |
| Set Password | Yes | No | Yes |
| Add/Remove Entitlements<br>• Add/Remove Shared Mailbox<br>• Add/Remove individual license plans<br>• Add/Remove license packs<br>• Add/Remove Roles<br>• Add/Remove user's group membership<br>• Add/Remove ServicePrincipal Names<br>• Add/Remove Channels | Yes | Yes | Yes<br>*Not applicable for Add/RemoveShared Mailbox from users* |
| Aggregate Exchange Mailbox Attribute | Yes | Yes | No |
| Update Exchange Mailbox Attribute | | | |
| Aggregate Shared Mailbox as an Entitlement | | | |

## Account Management for Local User (B2C)

Azure Active Directory B2C is primarily for businesses and developers that create customer-facing applications. With Azure Active Directory B2C, users can use Azure Active Directory as the full-featured identity system for their application, while letting customers sign in with an identity they already have established (like Facebook or Gmail).

B2C is intended for inviting customers of your mobile and web applications, whether individuals, institutional or organizational customers into your Azure Active Directory.

Supported identities are Consumer users with local application accounts (any email address or user name) or any supported social identity with direct federation. A new/separate Azure Active Directory application must be created to manage B2C tenant.

If **Manage B2C Tenant** is selected, entitlements are not fetched by default. To fetch entitlements set **fetchB2CMembership** attribute to true in as mentioned in "Additional Configuration Parameters" section.

| Operations | Manage B2C Tenant | |
|---|---|---|
| | Local User | Social User |
| Aggregation | Yes | Yes |
| Partitioning Aggregation | Yes | Yes |
| Create | Yes | No |
| Update | Yes | Yes |
| Delete | Yes | Yes |
| Enable/Disable Users | Yes | Yes |
| Add/Remove Entitlements<br>Add/Remove Roles<br>Add/Remove user's group membership<br>Add/Remove ServicePrincipal Names | Yes | Yes |

Delta Aggregation is not applicable if '**Manage B2C Tenant**' is selected in "Configuration Parameters".

## OAuth 2.0 Authentication

Azure Active Directory connector supports OAuth2.0 authentication with the following grant types:

- Client Credentials
- SAML Bearer Assertion
- Refresh Token/Auth Code
- JWT Certificate Credentials

### *Client Credentials*

For the default grant type client credentials based authentication, following are the required configurations:

- Obtain **Client ID**, **Client Secret** from Azure Active Directory by registering the application.
- Assign the required Microsoft Graph API permission to application. For more information, see Assigning Application Permission from Portal.
- The following permissions do not allow the connector to manage users with administrative roles. To manage users with administrative roles, the application created on Azure must have **User Account Administrator** or **Global Administrator** role assigned using Windows Azure Active Directory Module for Windows PowerShell. For more information, see Assigning Application Role using PowerShell.

| Permission | Type | Purpose |
|---|---|---|
| Directory.ReadWriteAll | Application | Read, Update, Delete Group, Add Membership<br>Read, Update, Change Password, Delete User |
| User.Invite.All | Application | Invite B2B Users |

## SAML Bearer Assertion

The SAML Bearer Assertion grant type authentication involves password-based user authentication with ADFS environment. SAML assertion issued by ADFS after authentication is used to obtain access token from Azure Active Directory.

The SAML Bearer Assertion authentication requires the following additional configurations to be performed:

- AAD Connect configured with Azure Active Directory along with ADFS.
- Obtain Client ID, Client Secret from Azure Active Directory by registering the application.
- Assign required permissions to application:
-

| Permission | Type | Purpose |
|---|---|---|
| Directory.ReadWriteAll | Delegated | Read, Update, Delete Group, Add Membership<br>Read User, Update User |
| User.Invite.All | Delegated | Invite B2B Users |
| Directory.AccessAsUser | Delegated | Change Password, Delete User |

Assign granular level application permission for each operation if you do not want to assign full directory level permission. For more information, see Required Permissions.

- The authentication user must be synchronized in Azure Active Directory.
- Assign the user with **User Administrator** permission Role in Azure Active Directory.

  To manage users with administrative roles assign user with **Global Administrator** role.

- The ADFS endpoint required to authenticate user must be enabled.

- The ADFS service communication certificate must be installed on IdentityIQ machine.
- The ADFS machine time zone must be in sync with Azure time zone, that is, UTC

### *Refresh Token/Auth Code or JWT Certificate Credentials*

- **Refresh Token/Auth Code**: Refresh Token/Auth Code grant type is a client credentials based authentication protocol. In addition to client credentials, it uses Refresh Token to perform authentication.
- **JWT Certificate Credentials**: JWT Certificate Credentials supports Authentication based on JWT assertion prepared from Certificate and private key.

For the Refresh Token/Auth Code or JWT Certificate Credentials grant type client credentials based authentication, following are the required configurations:

- Obtain **Client ID**, **Client Secret** from Azure Active Directory by registering the application.
- Assign the required Microsoft Graph API permission to application. For more information, see Assigning Application Permission from Portal.
- The following permissions do not allow the connector to manage users with administrative roles. To manage users with administrative roles, the application created on Azure must have **User Account Administrator** or **Global Administrator** role assigned using Windows Azure Active Directory Module for Windows PowerShell. For more information, see Assigning Application Role using PowerShell.

| Permission | Type | Purpose |
|---|---|---|
| Directory.ReadWriteAll | Delegated | Read, Update, Delete Group, Add Membership<br>Read User, Update User |
| User.Invite.All | Delegated | Invite B2B Users |
| Directory.AccessAsUser | Delegated | Change Password, Delete User |

> Assign granular level application permission for each operation if you do not want to assign full directory level permission.

- (*For Refresh Token/Auth Code only*) Generate Refresh Token. For more information, see Generating Refresh Token.
- (*For JWT Certificate Credentials only*) The Certificate (self-signed or CA signed) must be uploaded. It must be of type X.509 Certificate and Private Key must be encrypted with RSA and registered at the Azure AD portal. Perform the following steps to register the certificate with the Microsoft identity platform:
  a. Login to **Azure Portal**.
  b. In the Azure app registrations for client application, select the client application.
  c. Select the Certificates & secrets.
  d. Click the **Upload certificate** and select the certificate file that is to be uploaded.
  e. Click **Add**.
  f. Obtain values for the following configurations:

- Private Key: Obtain the private key text file.
- Private Key Password
- Certificate: Obtain the text file of the same certificate which was uploaded on the Azure portal.

## Azure Endpoint Configuration

To meet security and compliance needs of federal agencies, state and local governments, Microsoft provides separate instance of the Microsoft Azure service. Few example of such instances are Azure Government, Azure China. As these instances are separated from general Azure services, endpoint (host address) for such instances might also be different.

With endpoint configuration functionality, Azure Active Directory Connector can be configured to communicate with these instances.

Following attributes must be configured in application debug page:

### *msGraphResourceBase*

Base resource URL to be used for Microsoft Graph API rest calls.

Following example points base to default Microsoft Graph resource base:

```
<entry key="msGraphResourceBase" value="https://graph.microsoft.us"/>
```

### *msGraphTokenBase*

Base token URL to be used to get access token for Microsoft Graph API rest calls

Following example points base to default Microsoft Graph token base:

```
<entry key="msGraphTokenBase" value="https://login.microsoftonline.us"/>
```

> Actual values of endpoints can be found in Microsoft documentation for specific instance. For connector operation, values for the above attributes must be provided.

## Account - Group Management

Azure Active Directory connector supports managing following group types as an entitlement and as separate group objects. These different type of groups are created for different purpose which are described as follows:

- Microsoft 365 groups (formerly Office 365 groups) are used for collaboration between internal and external users of the company.
- Distribution List (Distribution groups) are used for sending notifications to a group of people.
- Security groups are used for granting access to resources such as SharePoint sites.
- Mail-enabled security groups are used for granting access to resources such as SharePoint, and emailing notifications to those users.

Following table describes different conditions/ configurations depending on which connector will aggregate / able to provision different types of groups:

| **Group Types** | **Filter condition to support Group Aggregation/Provisioning** | | |
|---|---|---|---|
| | **Default Aggregation** | **skipMailEnabledGroup=true in application debug page** | **‘Manage Office 365 Groups’selected in Configuration Parameters** |
| Security | Yes | Yes | Yes |
| Mail Enabled Security | Yes | No | Yes |
| Office 365 | No | No | Yes |
| Distribution List | No | No | No |

## Group Management for Azure Management Objects

Azure Active Directory connector now provides support for access management of the following Azure Management Objects:

- Management Groups
- Subscriptions
- Resource Groups
- Role Assignment (RBAC role assignments. This is a custom group object)

The details of the feature are:

Following attributes can be configured in application debug page as per requirement:

### *managementGroup-api-version*

API version to be used for management group API. Type: String

Default value: 2020-02-01

### *subscription-api-version*

API version to be used for subscription API's. Type: String

Default value: 2020-01-01

### *resourceGroup-api-version*

API version to be used for resource group API. Type: String

Default value: 2020-06-01

### *role-assignment-api-version*

API version to be used for Role Assignments API. Type: String

Default value: 2018-07-01

### *azure-management-resource-base*

Azure management api resource base in case of Azure Gov or other private instance. Type: String

Default value: https://management.azure.com

### *fetch-azure-roleassignment-getobject*

Specify if roles assignments needs to be fetched during Get Account call. Type: boolean

Default value: False

## *Prerequisites*

- Active Directory connector supports the following grant types for OAuth2 authentication:
    - Client Credentials
    - Auth Code / Refresh Token
    - Certificate Credentials

  Ensure that the appropriate permissions are granted as mentioned in the following 'Administrator Permissions' section
- Existing clients must be modified for supporting **management.azure.com** as the scope.

## *Administrator Permissions*

Based on the supported operations (Aggregation and Add/ Remove Entitlements), following are the required respective permissions:

### *Object Aggregation*

**Permission**: Microsoft.Management/managementGroups/read

*Or*

**Role**: Reader

**Scope**: Management Group

### *Role Assignment Aggregation*

**Permission**: Microsoft.Authorization/roleAssignments/read

*Or*

**Role**: Reader

**Scope**: Management Group / Subscription

### *Role Definition Aggregation*

**Permission**: Microsoft.Authorization/roleDefinitions/read

*Or*

**Role**: Reader

**Scope**: Management Group / Subscription

### *Add Role Assignment*

**Permission**: Microsoft.Authorization/roleAssignments/write

*Or*

**Role**: User Access Administrator

**Scope**: Management Group / Subscription

### *Remove Role Assignment*

**Permission**: Microsoft.Authorization/roleAssignments/delete

*Or*

**Role**: User Access Administrator

**Scope**: Management Group / Subscription

**API Permissions**

| OAuth2.0 Authentication | Type | API | Permission |
|---|---|---|---|
| Client Credentials | Delegated | Azure Service Management | user_impersonation |
| | Application | Microsoft Graph | Directory.ReadWriteAll |
| Refresh Token / AuthCode | Delegated | Azure Service Management | user_impersonation |
| JWT Certificate Credentials | Delegated | Azure Service Management | user_impersonation |

### *Supported Schema Attributes*

To manage the following Azure Management Objects, ensure that the attributes present in the respective sections are present in the group schema:

- Management Groups
- Subscriptions
- Resource Group
- Azure Role Assignment

Schema for above mentioned Group objects is not available out of the box.

## Microsoft Teams

Microsoft Teams, also referred to simply as Teams, is a unified communication and collaboration platform that combines persistent workplace chat, video meetings, file storage (including collaboration on files), and application integration.

Teams are built on the top Microsoft 365 groups. The Microsoft 365 group forms the directory object representing Teams. All identity and access controls for Teams are performed on the Microsoft 365 group.

As Azure Active Directory Connector deals with directory objects, Microsoft Teams are represented as Microsoft 365 Groups.

The **Manage Office 365 Groups** checkbox must be selected as it is aggregated as Microsoft 365 Group. Following operations are supported:

- Creation of Teams as Microsoft 365 Group
- Add\Remove Owners and Members

> To delete Teams, respective Microsoft Office 365 Group must be deleted.

The **Manage Office 365 Groups** checkbox must be selected because it is aggregated as a Microsoft 365 group.

| Attribute Name | Action | Group Schema | Create Group Policy |
|---|---|---|---|
| teamsEnabled | Distinguish Teams group | Type: Boolean | NA |
| | Enabling Teams creation | NA | Type: Boolean<br><br>Review Required: True |
| | Enabling Teams creation for existing Microsoft Office 365 Groups | NA | Type: Boolean<br><br>Review Required: True |
| Owners | NA | Property: Multi-Valued<br><br>Data Type: string<br><br>Description: Owners of the group | Read-Only: False |

## Service Principal Management

Azure Service Principal is a security identity used by user-created applications, services, and automation tools to access specific Azure resources. Service Principal is a **user identity** (login and password or certificate) with a specific role, and controlled permissions to access your resources. Security is improved if minimum permissions level are granted to perform management tasks.

Following operations are supported for ServicePrincipal object type:

- Aggregation
- View details of **ServicePrincipal** (like object properties, members and so on)
- Provision and Revoke access request for **ServicePrincipal**

For more information on the service principal schema attributes, see Service Principal.

> To improve the aggregation performance, delete the following if you do not want to aggregate Service Principal as an entitlement and as a separate group object:
>
> - **servicePrincipals** attribute from Account Schema
> - **servicePrincipal** object type Schema

## Channel Management

Microsoft Teams, also referred to as simply Teams, is a unified communication and collaboration platform that combines persistent workplace chat, video meetings, file storage (including collaboration on files), and application integration.

Channels are dedicated sections within a team to keep conversations organized as per organization requirement.

Basic criteria for any user to be able to be part of channel is to be part of the team. If user is not part of the team, user can not be added to channel. Access to the channel varies on the following types of channel:

- **Standard Channel**: Any user who is member of the Team gets membership of the standard channel by default.
- **Private Channel**: Owner/ Admin of the channel needs to add team member to channel. By default channel is not accessible to anyone other than owner/ admin

As addition of user to team gives access to standard channel, removal of user from team leads to automatic removal from all channels of team including private channels to which user has access to.

This feature provides support for the following operations:

- Aggregation of channels as separate group object
- Aggregation of user membership to channels.
- Create/ Update/ Delete Channels
- (*Applicable only for private channels*) Add/ Remove user to/ from Channels

  - Before enabling Teams Governance, ensure that the **Manage Office 365 Groups** flag is set true.
  - Enabling Teams Governance feature can have impact on aggregation performance due to API limitations to fetch channel memberships.

### *Administrator Permissions*

| Permission | Permission Type | Purpose |
|---|---|---|
| Channel.Create | Application | Create Channel |
| Channel.Delete.All | Application | Delete Channel |
| Channel.ReadBasic.All | Application | Aggregate Channels |
| ChannelMember.ReadWrite.All | Application | Add/ Remove Channel Members |
| ChannelMember.ReadWrite.All | Delegated | Add/ Remove Channel Members |

### *Supported Schema Attributes*

To manage the channel management objects, ensure that the attributes present in Channel are present in the group schema.

## Exchange Online Mailbox Management

Azure Active Directory can be used to manage exchange online mailboxes . Azure Active Directory Connector uses Exchange Online Powershell Module through IQService to support this feature. Following operations are supported:

- Aggregation of Exchange Online Mailbox attributes for users
- Aggregation of Shared Mailbox as an entitlement for users
- Modification of Exchange Online Mailbox attributes
- Add/Remove Shared Mailbox from users

> Delta Aggregation does not capture changes in Exchange Online Attributes and Shared Mailbox assignments.

### *Prerequisites*

- Configure IQService
- Install Exchange Online Powershell Module V2 on same IQService machine.
- Create a user in Azure Active Directory with **Exchange Administrator** role.
- Select **Manage Exchange Online** on Application Configuration page.
- Provide username and password of user created in step 3

> Due to limitation on Powershell sessions SailPoint recommends to use separate IQService instance and separate exchange admin user for different Azure Active Directory applications which are defined to manage exchange online.

> Connector uses PowerShell sessions to manage Exchange Online Mailboxes. Due to restrictions on number of concurrent PowerShell sessions allowed by Microsoft, there may be delay or occasional failures when connector processes Exchange Online requests.

### *Schema Configurations*

Exchange attributes which must be aggregated must be added to account schema with **EXO_** as prefix. For example, to aggregate **EmailAddresses** attribute, it must be added in schema as follows:

- Name: EXO_EmailAddresses
- Type: String
- Property: Multivalued
- To aggregate shared mailbox attributes as an Entitlement add **sharedMailbox** as account attribute.
- Example Name: sharedMailbox
- Type: String
- Property: Multivalued, Entitlement, Managed

### *Provisioning Policy Changes*

To update Exchange Mailbox attribute value, attribute must be added to provisioning policy with **EXO_** as prefix.

For example to update **Alias** attribute, it must be added in provisioning policy as follows:

- Name: EXO_EmailAddresses
- Type: String
- Type Setting: Multivalued: true, Review Required: true

> It is important to add with proper attribute type and property, which meets exchange attribute definition for successful update operation.

> Attributes to be added in provisioning policy must be present in account schema.

### *Shared Mailbox As Entitlement*

Exchange user can be a member of Shared Mailbox. Along with membership user obtains permission on the mailbox. This permission has to be selected while adding user to the shared mailbox. The following permissions are categorized as Recipient and Mailbox:

- Recipient: SendAs
- Mailbox: ChangeOwner, ChangePermission, DeleteItem, ExternalAccount, FullAccess, ReadPermission

In order to provide flexibility to select the permission to be assigned, mailbox entitlements would be created one per permission per mailbox as explained in the following example:

```
User Mark Taylor has SendAs, FullAccess and ReadPermission permissions on shared
mailbox called O365Support. After account aggregation following entitlements
would be created:
```

```
O365Support: SendAs
```

```
O365Support: FullAccess
```

```
O365Support: ReadPermission
```

## Risky User Alert Feature

With the security reports in the Azure Active Directory, you can gauge the probability of compromised user accounts in your environment. A user flagged for risk is an indicator for a user account that might have been compromised. The risky user represents the probability that a given identity or account is compromised. These risks are calculated offline using Microsoft's internal and external threat intelligence sources including security researchers, law enforcement professionals, security teams at Microsoft, and other trusted sources.

This feature supports the following operations:

- Account Aggregation
- Get Object

By default this feature is enabled for new connectors. If your Azure Active Directory instance doesn't require or support the Risky User Alert feature, you must disable the feature by removing attributes from the schema.

### *Prerequisite*

An Azure Active Directory Premium P2 license

### *Administrator Permissions*

To fetch risky user details using MS Graph APIs, the following API permissions must be assigned:

| OAuth2.0 Authentication | Type | Permission | Purpose |
|---|---|---|---|
| Client Credentials | Application | IdentityRiskEvent.Read.All<br>IdentityRiskyUser.Read.All | Aggregate or Get Risky user related information |
| SAML Bearer Assertion<br>Refresh Token / AuthCode<br>JWT Certificate Credentials | Delegated | IdentityRiskEvent.Read.All<br>IdentityRiskyUser.Read.All | Aggregate or Get Risky user related information |

For SAML Bearer Assertion the authenticated user must have the Security Operator role.

### *Supported Schema Attributes*

To manage the risky user alert feature, ensure that the following attributes are present in the account schema:

#### *riskLevel*

Level of the detected risky user.

#### *riskState*

State of the user's risk.

#### *riskDetail*

Details of the detected risk.

#### *riskLastUpdatedDateTime*

The date and time that the risky user was last updated.

Only the default schema attributes will be supported. The Account schema can not be extended for other risk related attributes.

## Azure Privileged Identity Management (PIM)

The Azure Active Directory source now supports the Privileged Identity Management (PIM) as a service in Azure Active Directory that enables you to manage, control, and monitor access to important resources in your organization.

These resources include resources in Azure AD, Azure, and other Microsoft Online Services such as Microsoft 365 or Microsoft Intune.

PIM provides time-based and approval-based role activation to mitigate the risks of excessive, unnecessary, or misused access permissions on resources. It reduces the chance of a malicious actor getting access to the resource or an authorized user accidentally impacting a sensitive resource.

**Supported Features**

- Aggregation of PIM Role Assignment objects for Azure and Azure AD during Account aggregation
- Aggregation of Azure and Azure AD PIM roles during Entitlement Aggregation
- Provisioning of eligible role assignment on user for Azure/Azure AD PIM role
- Provisioning of Active role assignment on user for Azure/Azure AD PIM role.

| Operations | Group Objects |
|---|---|
| Entitlement Aggregation | Azure Eligible Roles, AzureAD Eligible Roles, Azure Active Roles, and Azure AD Active Roles (These are custom group objects.) |
| Account Aggregation | Azure Eligible Roles, Azure AD Eligible Roles, Azure Active Roles, and Azure AD Active Roles - these four are aggregated as entitlements |
| Add / Remove Entitlement | Azure Eligible Roles, Azure AD Eligible Roles , Azure Active Roles, Azure AD Active Roles |

**Permissions**

### Azure AD roles

To communicate with the PIM Graph API for Azure AD roles, you must have at least one of the following permissions:

- `RoleManagement.ReadWrite.Directory`
- `RoleManagement.Read.Directory`

### Azure Resource Roles

The PIM API for Azure resource roles is developed on top of the Azure Resource Manager framework. You will need to give consent to Azure Resource Management but won't need any Graph API permission. You must ensure that the user or the service principal communicating with the API has at least the Owner or User Access Administrator role on the resource.

## Prerequisites

### Azure Management

Ensure that a client application has been registered on your Azure Management portal as a web application or web API, and you have access to the Client ID and Client Secret for this application.

To use Graph API, a client application must be registered on the Azure management portal. This application is responsible for calling Web APIs on behalf of the connector. The application's client ID and client secret key are required while configuring the application.

To register an application on Azure, perform the following:

1. User can use any of the following Azure management portal to do the configuration:
   **https://portal.azure.com**
   Or
   **https://aad.portal.azure.com**
2. Select **Azure Active Directory** in the left pane.
3. Click on **App registrations**.
4. Click **New registration**.
5. On the **Register an application** page, in the Name field, enter the name of the application that you want to set up. For example, SailPointAzureADManagement.
6. In the Supported account types, set up accounts on the basis of users, eligible to avail that application or the API.
7. (*Optional*) Set up the URL in **Redirect URL**, to have the successful response after authentication. You can use the following format: **http://***domainName***/GraphWebapp**

   > Azure Active Directory connector does not use the URL mentioned above, the above example is just a place-holder and does not impact functionality.

8. Click **Register**. An Application is created. On the Application page the **Application (client) ID**, and other details are displayed. Note down this ID.
9. On the left-hand panel, select **Certificates & secrets**. On the Certificates & secrets page, in the **Client secrets** section, click **New client secret**.
10. On the **Add a client secret** page, enter the **Description** to generate a secret, choose the validity duration in the **Expires** list. Click **Add**. Note down the value of the secret that you have just created.

    > If Azure Active Directory Connector is behind proxy server, see the “Special Java Considerations” section of the *SailPoint IdentityIQ Installation Guide*.

### Azure Active Directory

The required host values (**https://&lt;host&gt;/**) by the Azure Active Directory Connector to interact with the managed system are as follows:

- https://graph.microsoft.com
- https://login.microsoftonline.com

## Required Permissions

Following permissions must be granted to the client application created in Azure:

- Read Directory Data
- Read and Write Directory Data

To grant permissions to the client application:

- Click **API permissions** in Azure Active Directory console. Click **Add a permission**.
- On the **Request API permissions** page, you will see a list of supported APIs, click **Microsoft Graph API**.
- Choose **Application permissions** under **What type of permissions does your application require?**
- Under **Select permissions**, choose permissions mentioned in the following permission table. Click **Add permissions**.
- In Grant consent, click **Grant admin consent** for your configuration and directory. On the pop-up dialog box, click **Yes**.

**Granular Level Application Permission**

| Permission | Type | Purpose |
|---|---|---|
| User.Invite.All | Application | Creating / Inviting B2B User |
| User.Read.All | Application | Account Aggregation, Account Delta, Get Object, Roles and Groups Membership Aggregation |
| User.ReadWrite.All | Application | Create User, Update User Properties (Non Entitlement), Add / Remove License Pack and Plan, Enable/ Disable User Account, Delete User |
| Organization.Read.All | Application | Aggregate License Pack and Plan Details of tenant |
| RoleManagement.ReadWrite.Directory | Application | Add / Remove Directory Roles |
| User.Read | Application | Passthrough Authentication |
| Group.Read.All | Application | Group Aggregation |
| Group.ReadWrite.All | Application | Create Group, Update Group, Delete Group |
| Application.Read.All | Application | Service Principal Aggregation |

| Permission | Type | Purpose |
|---|---|---|
| AppRoleAssignment.ReadWrite.All | Application | Add / Remove users from Service Principal |
| *Applicable for **SAML Bearer Assertion**, **Refresh Token / AuthCode** and **JWT Certificate Credentials** Grant Types* | | |
| Directory.AccessAsUser | Delegated | Change Password, Delete User |

To perform Set Password and Delete user operations, application created on Azure must have **User Account Administrator** role assigned using **Windows Azure Active Directory Module for Windows PowerShell**.

To manage users with administrative roles, application created on Azure must have **User Account Administrator** or **Global Administrator** role assigned using **Windows Azure Active Directory Module for Windows PowerShell**. Following are the prerequisite for executing the PowerShell commands.

> These prerequisites are not required for the connector to function and can be installed on any system for temporary use to provide required role to the application on Azure.

- Microsoft Online Services Sign-In Assistant for IT Professionals RTW
- Azure Active Directory Module for Windows PowerShell

After installing the prerequisites, open **Azure Active Directory Module for Windows PowerShell** console and execute the following commands:

- `Connect-msolservice`, press enter, provide Azure administrator credentials.
- Execute `Get-MsolServicePrincipal -All | ft DisplayName, AppPrincipalId -Autosize`
- Locate your application name from the list and copy the **AppPrincipalId** value.
- Execute `$ClientObjID = '<copied objectId of the application in the previous step>'`
- Execute `$webApp = Get-MsolServicePrincipal -AppPrincipalId $ClientObjID`
- Execute `Add-MsolRoleMember -RoleName "Global Administrator " -RoleMemberType ServicePrincipal -RoleMemberObjectId $webapp.ObjectId`

## Upgrade Considerations

- To manage Office 365 type of groups after upgrading IdentityIQ, add the following:

| Attribute Name | Group Schema | Create Group Policy | Update Group Policy |
|---|---|---|---|
| groupTypes | Property: Multi-Valued<br>Data Type: string<br>Description: Type of the group | Review Required: true<br>Required: true<br>Type: String<br>Allowed Values: Security, Office 365 | Multi-Valued: true<br>Type: String<br>Read Only: True |
| securityEnabled | NA | NA | Type: Boolean<br>Read Only: True |

- For creating and enabling Teams after upgrading IdentityIQ, add the following:

| Attribute Name | Action | Group Schema | Create Group Policy |
|---|---|---|---|
| teamsEnabled | Distinguish Teams group | Type: Boolean | NA |
|  | Enabling Teams creation | NA | Type: Boolean<br>Review Required: True |
|  | Enabling Teams creation for existing Microsoft Office 365 Groups | NA | Type: Boolean<br>Review Required: True |
| Owners | NA | Property: Multi-Valued<br>Data Type: string<br>Description: Owners of the group | Read-Only: False |

- To manage channel type of account object type after upgrading IdentityIQ, add the following:

| Attribute Name | Account Schema | Description |
|---|---|---|
| channels | Property: Multi-Valued, Entitlement, Managed | List of channels to which user belongs. |

Add the create channel provisioning policy attributes as described in the Create Channel Policy section.

# Connecting SailPoint and Azure Active Directory

To connect SailPoint and Azure Active Directory, perform the following tasks:

- Configure the Connector in SailPoint
- Configuration Parameters
- Azure PIM Configuration

## Configuration Parameters

This section contains the information that this connector uses to connect and interact with the application. Each application type requires different information to create and maintain a connection.

The Azure Active Directory connector uses the following connection parameters:

### Connector Credentials

#### *Authentication Method*

Authentication method supported by the managed system. Default is **OAuth 2.0**.

#### *Grant Type*

Grant type to be used for the authentication.

#### *Client Credentials*

SAML Bearer Assertion.

#### *Refresh Token / Auth Code*

JWT Certificate Credentials.

#### *Azure AD application client ID**

Client ID of the application created on the Azure Active Directory for using Graph REST API.

#### *Azure AD application client secret key**

Client secret of the Azure Active Directory application.

#### *Azure AD domain name**

Name of the Azure Active Directory domain to be managed. For example, `contoso.onmicrosoft.com`.

**Applicable only if *Grant Type* is selected as *SAML Bearer Assertion***

#### *Authorization Endpoint**

Endpoint URL of authorization server.

#### *Username**

Username for authorization.

#### *Password**

Password of the user for authorization.

#### *Request Body**

Request body for SAML assertion.

For more information on SAML Request Body, see SAML Request Body.

**Applicable only if *Grant Type* is selected as *Refresh Token/Auth Code***

### *Refresh Token**

Enter valid refresh token.

**Applicable only if *Grant Type* is selected as *JWT Certificate Credentials***

### *Certificate**

The unique alpha-numeric value of certificate used to sign the JWT assertion.

### *Private Key**

Private Key text used for encrypting the JWT assertion.

### *Private Key Password**

Password for decrypting private key.

## Additional Configuration

### *Page Size*

Number of records per page. Default: 500.

### *IQService Host*

FQDN/IP of the system where IQService is installed.

### *IQService Port*

The TCP/IP port on which IQService is listening for requests.

f you enable **Use TLS**, configure the corresponding IQService TLS port.

> To enable exchange online mailbox management, you must configure the IQService Host and IQService Port parameters.

### *IQService User*

User registered with IQService for Client Authentication.

### *IQService Password*

Password of registered user for Client Authentication.

### *Use TLS for IQService*

Indicates whether this is a TLS communication between IdentityIQ and IQService.

If you enable **Use TLS for IQService**, **IQService User** and **IQService Password** attributes are mandatory

### *Manage Office 365 Groups*

Enables aggregation and provisioning of Office 365 groups.

### *Enable Teams Governance*

Enables aggregation and provisioning of Channels.

### *Manage B2C Tenant*

Indicates whether application is used to manage B2C tenant.

### *Manage Exchange Online*

Enables Aggregation and Provisioning of Exchange mailbox attributes.

**Applicable only if*Manage Exchange Online*is selected**

### *Username*

UserPrincipalName of User having Exchange Administrator role.

### *Password*

Password of user having Exchange Administrator Role.

Attributes marked with * sign are the mandatory attributes.

To enable native before/after script execution for provisioning requests, configure **IQService Host** and **IQService Port** parameters. For more information on enabling the Client Authentication and TLS communication, see IQService.

## Additional Configuration Parameters

### *performGetObjectForEXOAttributes*

To enable / disable reading of exchange online attributes from exchange system (real time exchange online attribute values) during explicit get object call for specified user, set the value of **performGetObjectForEXOAttributes** attribute to true as follows (Default: false):

```
<entry key="performGetObjectForEXOAttributes">
  <value>
    <Boolean>true</Boolean>
  </value>
</entry>
```

This configuration parameter is applicable only for get object call. Enabling this configuration parameter will degrade performance of get object calls.

### *createAccountTimelag*

Time in seconds to wait after create account and before calling get account. Default: 20 seconds.

For example, `<entry key="createAccountTimelag" value="20"/>`

### *maxReadTimeout*

Time in seconds to wait for getting response from the REST call before the read operation times out. Default: 180 seconds.

For example, `<entry key="maxReadTimeout" value="200"/>`

### *maxRetryCount*

Indicates the number of time read operation must be performed on the errors that appear. Default: 5.

For example, `<entry key="maxRetryCount" value="6"/>`

### *retryableErrorsOnAgg*

List of errors which must be retried if occurred during aggregation or get operation. Type: List of strings.

### *userPartitions*

List of filters to be used for creating partitions during partitioned aggregation.

For more information, see Partitioning Aggregation.

### *userFilters*

Filter that defines the scoping condition for Accounts to be applied during account aggregation to limit set of data. The Azure Active Directory Connector now supports advanced query filters like `endsWith`, `NOT` and `NE` during aggregation.

Following is an example for **userFilters** configuration attribute:

```
<entry key="connector_userFilters" value="(startswith(displayName,'A')and
accountEnabled eq true)"/>
```

For more information on filters, see the **Supported query options** section of the Azure AD Graph API Concepts document.

### *groupFilters*

Filter that defines the scoping condition for Groups to be applied during account-group aggregation to limit set of data.

Following is an example for **groupFilters** configuration attribute:

```
<entry key="groupFilters" value="(startswith(displayName,'A') or
startswith(displayName,'B'))"/>
```

### *partitionHost*

(*Applicable only for partitioned account delta aggregation*) The **partitionHost** is the host on which all delta partitions must be executed. Following is an example for **partitionHost** configuration attribute:

```
<entry key="partitionHost" value= "myhost"/>
```

### *skipMailEnabledGroup*

If set to true, Mail Enabled Groups is skipped during aggregation. Default value is false.

For example, `<entry key="skipMailEnabledGroup" value="true"/>`

### *aggregateAllGroups*

If set to true, all types of groups (that are, Security, Office 365 (Unified), Distribution List, Mail Enabled Security) are aggregated. Default value is false.

For example, `<entry key="aggregateAllGroups" value="true"/>`

### *fetchB2CMembership*

(*Applicable for B2C tenant only*) If set to true B2C user memberships is fetched for all users. Default value is false.

For example, `<entry key="fetchB2CMemberships" value="true"/>`

### *teamsConfiguration*

(*Applicable only when **teamsEnabled** is selected for Group creation*) Teams configuration with which all teams are created. Default settings are provided in **Additional Configuration Parameters**.

### *createGroupTimelag*

Time in seconds to wait after create group and before calling get object. Default: 5 seconds.

For example, `<entry key="createGroupTimelag" value="10"/>`

### *useMSGraphAPI*

Enables use of Microsoft Graph API for all connector operations.

For example, `entry key="useMSGraphAPI" value="true"`

### *msgraph-api-version*

The Microsoft Graph API version is configurable. Add the following entry key in the application debug page:

```
<entry key="msgraph-api-version" value="v1.0"/>
```

The value of the attribute mentioned must be a valid version stated by Microsoft for MS Graph APIs.

**Default Settings for Teams Configuration**

Teams can be created by selecting **Group Type** as **Office 365** and selecting **teamsEnabled** in Group Creation. For Teams creation owner having **Teams** license must be provided. By default, Teams are created with the following settings:

```
"memberSettings": {
        "allowCreateUpdateChannels": true,
        "allowDeleteChannels": true,
        "allowAddRemoveApps": true,
        "allowCreateUpdateRemoveTabs": true,
        "allowCreateUpdateRemoveConnectors": true
    },
    "guestSettings": {
        "allowCreateUpdateChannels": false,
        "allowDeleteChannels": false
    },
    "messagingSettings": {
        "allowUserEditMessages": true,
        "allowUserDeleteMessages": true,
        "allowOwnerDeleteMessages": true,
        "allowTeamMentions": true,
        "allowChannelMentions": true
    },
    "funSettings": {
        "allowGiphy": true,
        "giphyContentRating": "strict",
        "allowStickersAndMemes": true,
        "allowCustomMemes": true
    }
}
```

To create Teams with custom settings, add the following entry key in the application debug page:

```
<entry key="teamsConfiguration" value="{"memberSettings":
{"allowCreateUpdateChannels": false, "allowDeleteChannels": true,
"allowAddRemoveApps": false, "allowCreateUpdateRemoveTabs": false,
"allowCreateUpdateRemoveConnectors": true}}" />
```

The custom settings must be in valid JSON format and HTML escaping must be performed before adding to XML, that is, double quotes must be replaced with **&quot;**.

Team creation within 15 minutes of Microsoft group creation can fail with a 404 error code due to replication delays. If group creation fails, retry with a 10 second delay between calls. SailPoint Azure Active Directory Connector follows this recommended pattern.

## Azure PIM Configuration

| Attribute | Details |
|---|---|
| enablePIM | It enables PIM for Azure AD and Azure Roles. You can enable PIM by selecting the **Enable Privileged Identity Management** checkbox or by using the following: `<entry key=”enablePIM”> <value> <Boolean>true</Boolean> </value> </entry>` |
| eligibleRoleExpiresAfter | It specifies the default duration for which Azure and Azure AD eligible roles must be assigned to user. Values must be in the ISO_8601 duration format. For example, if eligible role needs to be assigned for 180 Days, you can use `<entry key=”eligibleRoleExpiresAfter” value=”P180D” />` |
| activeRoleExpiresAfter | It specifies the default duration for which Azure and Azure AD Active roles must be assigned to user. Values must be in the ISO_8601 format. For example, if eligible role must be assigned for 10 Hours, you can use `<entry key=”activeRoleExpiresAfter” value=”PT10H” />` |

# Configuring Azure Active Directory for Integration

This section describes the various configurations to be performed to support the following features:

- SAML Request Body
- Assigning ServicePrincipal
- Assigning Application Permissions
- Service Plan Management (Managing Licenses)
- Generating Refresh Token
- IQService Before/ After Script Attributes
- Support for Provisioning Operations of 'manager' Attribute
- Delta Aggregation
- Partitioning Aggregation

## Microsoft Graph API

Microsoft recommends the use of Microsoft Graph API instead of the Azure Active Directory Graph API. Microsoft has stopped addition of new feature in the Azure Active Directory API. The Azure Active Directory connector has been enhanced to use the Microsoft Graph API completely.

- To enable using the latest attributes and features supported by Microsoft Graph API, set the value of **useMSGraphAPI** attribute to true.
- The default Microsoft Graph API version is 1.0. To use another version, specify the verison in the in msgraph-api-version attribute in the source .

For more information, see Additional Configuration Parameters.

## SAML Request Body

The SAML Request Body must consists of the following:

- SAML Request must have MicrosoftOnline as the intended audience
- Username in request must be replaced by the **$username$** place holder
- Password in request must be replaced by **$password$** place holder

Following is an example of a sample SAML Request Body:

**Authorization Endpoint**: https://<YOUR_DOMAIN>/adfs/services/trust/2005/usernamemixed

**SAML Request**:

```
<s:Envelope xmlns:s='http://www.w3.org/2003/05/soap-envelope'
xmlns:a='http://www.w3.org/2005/08/addressing'
xmlns:u='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
```

```
1.0.xsd'>
  <s:Header><a:Action s:mustUnder-
stand='1'>http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue
  </a:Action>
<a:ReplyTo><a:Ad-
dress>http://www.w3.org/2005/08/addressing/anonymous</a:Address></a:ReplyTo>
        <a:To s:mustUnderstand='1'>https://<YOUR_DOMAIN>/ad-
fs/services/trust/2005/usernamemixed</a:To>
          <o:Security s:mustUnderstand='1'
          xmlns:o='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd'>
            <o:UsernameToken u:Id='uuid-<user's object id>'>
        <o:Username>$username$</o:Username>
        <o:Password>$password$</o:Password></o:UsernameToken>
    </o:Security>
    </s:Header>
    <s:Body>
            <trust:RequestSecurityToken xmlns:trust-
='http://schemas.xmlsoap.org/ws/2005/02/trust'>
            <wsp:AppliesTo xmlns:wsp='http://schemas.xmlsoap.org/ws/2004/09/policy'>
            <a:EndpointReference>
            <a:Address>urn:federation:MicrosoftOnline</a:Address>
            </a:EndpointReference>
            </wsp:AppliesTo>
            <trust:KeyType>http://schemas.xmlsoap.org/ws/2005/05/identity/NoProofKey</tru-
st:KeyType>
            <trust:RequestType>http://schemas.xmlsoap.org/ws/2005/02/trust/Issue</trust:R-
equestType>
            </trust:RequestSecurityToken>
  </s:Body>
</s:Envelope>
```

## Assigning ServicePrincipal

While providing role id from policy, if user has configured policy for specifying role id, then this role id would be used for provisioning of all **ServicePrincipals** in a request. Hence, user must provision one **ServicePrincipal** in one request.

While providing an access to a **ServicePrincipal**, it is mandatory to provide an access role for the User. These access roles may differ depending upon the **ServicePrincipal** type. IdentityIQ would use one of the access role it receives for each **ServicePrincipal** during the aggregation of **ServicePrincipals**.

> The **servicePrincipalRoleId** attribute must be added in **Update Account Policy** while assigning ServicePrincipals from Request Access.

## Assigning Application Permissions

This section provides details about assigning application permissions from Portal and using PowerShell.

### Assigning Application Permission from Portal

- Click **API permissions** section registered App in Azure Active Directory console. Click **Add a permission**.
- On the **Request API permissions** page a list of supported APIs is listed.
    - **Microsoft Graph** can be found under Commonly used Microsoft API
- Select the type of permission as **Delegated** or **Application** under **What type of permissions does your application require?**
- Under **Select permissions**, select the required permissions.
- Click **Add permissions**.

In Grant consent, click **Grant** Admin Consent for your configuration and directory. On the pop-up dialog box, click **Yes**.

### Assigning Application Role using PowerShell

Following are the prerequisite for the PowerShell commands.

- Microsoft Online Services Sign-In Assistant for IT Professionals RTW
- Azure Active Directory Module for Windows PowerShell

> These prerequisites are not required for the Azure Active Directory Connector to function. These can be installed on any system for temporary use to provide required role to the application on Azure.

After installing the prerequisites, open **Azure Active Directory Module for Windows PowerShell** console and execute the following commands:

- `Connect-msolservice`, press enter, provide Azure administrator credentials.
- Execute `Get-MsolServicePrincipal | ft DisplayName, AppPrincipalId -Autosize`
- Locate your application name and copy the **ObjectId** value.
- Execute `$ClientObjID = '<copied objectId of the application in the previous step>'`
- Execute `$webApp = Get-MsolServicePrincipal -AppPrincipalId $ClientObjID`
- Execute `Add-MsolRoleMember -RoleName "Global Administrator " -RoleMemberType ServicePrincipal -RoleMemberObjectId $webapp.ObjectId`

## Service Plan Management (Managing Licenses)

Azure Active Directory connector supports assigning different Azure services licenses to the users. Connector provides options to assign license either by individual plan or as a whole license pack.

- **Assigning license plan**: Office 365 license pack consist of licenses for individual services. For example, Exchange Online, SharePoint Online and so on.
- The connector models **assignedPlans** attribute from account schema as an entitlement. It can be requested as an entitlement during **Create** or **Update** operations for Identities.

- **Assigning license pack**: To assign license pack, set **assignedLicenses** attribute from account schema as **Managed**, **Entitlement**, **Multi-Valued**, So that it request-able as an entitlement.

    - It is recommended that **assignedPlans** or **assignedLicenses** must be promoted as an entitlement to avoid conflicts.
    - To provision the licenses or plans to user, set the user's **usageLocation** property correctly.

Service Plan now has separate group schema attributes. For a detailed list of the attributes, see Service Plan.

Following operations are supported for Service Plan object type:

- Aggregation
- View details of Service Plan (like object properties, members and so on)
- Assign and Revoke access request for Service Plan (entitlement attribute - **assignedPlans** or **assignedLicenses**)

## Generating Refresh Token

Perform the following steps to generate the Refresh Token:

1. Obtain the authorization code using the following request. This process involves user consent.

```
client_id=<clinetId>
&response_type=code
&redirect_uri=<redirect_uri>
&response_mode=query
&scope=offline_access%20https://graph.microsoft.com/.default
&state=<randomnumber>
```

Parameters details:

### *client_id*

Client ID of the registered application.

### *response_type*

For authorization code the value must be **code**.

### *redirect_uri*

This should be redirect uri configured while registering the application. This should be pointed to localhost and any unused port.

### *response_mode*

To get the code to in string parameter on your redirect URI, it should be **query**.

### *scope*

Here multiple scopes can be added for defining the required API.

The default added scope is **https://graph.microsoft.com/.default**.

The refreshToken parameter is also required for configuration. The **offline_access** scope must be added.

### *state*

This can be any random number. Same value will be returned along with authorization code.

The Response in browser address bar would be displayed as follows:

```
https://localhost:44320/?code=OAQABAAIAAAAm-
06blBE1TpVMil8KPQ41U9-...&state=1234566&session_state=9557b1f1-0fd8-4e12-
a39f-213cfcd12153
```

Copy the code present between **code=** and **&state**.

> The received Authorization Code can be used only once and is valid for a short duration.

2. Obtain the Refresh Token as follows using the Auth Code obtained in the above step:

The Auth Code received from the previous steps must be passed in the following request along with other parameters in following request:

```
    POST https://login.microsoftonline.com/{{Your Domain name}}/oauth2/v2.0/token
     Content-Type: application/x-www-form-urlencoded
client_id=<clinet id >
&scope=offline_access%20https://graph.microsoft.com/.default
    &code=OAAABAAAAiL9Kn2Z27UubvWFPbm0gLWQJVzCTE9UkP3pSx1aXxUjq3n8b2JRLk4OxVXr...
    &redirect_uri=http%3A%2F%2Flocalhost%2Fmyapp%2F
    &grant_type=authorization_code
    &client_secret=JqQX2PNo9bpM0uEihUPzyrh
```

Following are the newly added parameters:

### *code*

Authorization code obtained in previous steps.

### *grant_type*

As we intend to redeem the Auth Code it should be **authorization_code**.

### *client_secret*

Client Secret of the registered application.

**Response**:

```
{
    "token_type": "Bearer",
    "scope": "https://graph.windows.net/Directory.Read.All
    https://graph.windows.net/User.Read https://graph.windows.net/.default",
    "expires_in": 7199,
    "ext_expires_in": 7199,
    "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IlNzWnNCT
    mhaY0YzUTlTNHRycFFCVEJ5TlJSSSIsImtpZCI6IlNzWnNCTmhaY0YzUTlTNHRycFFCVEJ5TlJSSSJ9
    .eyJhdWQiOiJodHRwczovL2dyYXBoLndpbmRvd3MubmV0Iiwia...",
    "refresh_token": "OAQABAAAAAAAm-
06blBE1TpVMil8KPQ410LOG6EPVxGfgH8rHUXApUs5fPFtel9FsKTXo2oN8Rw
    _ngEOpKNt1hfufYyJJnG39XxfscpcW...."
}
```

Copy the **Refresh Token** received in the response.

## IQService Before/ After Script Attributes

The other feature supported by Azure Active Directory is executing native before/after scripts for provisioning requests.

Attributes to be used only for customization (for example, IQService Before/After Script), must be passed as Attributes map of the **AccountRequest** instead of AttributeRequest. For more information, see the example of **AccountRequest** xml provided in the IQService Before/After Scripts.

## Support for Provisioning Operations of 'manager' Attribute

The Azure Active Directory supports the **manager** attribute for the following:

- **Account Aggregation** (Full and Delta): On Account Aggregation, manager attribute for user will populate for non null (blank) value on Managed System. Add the **manager** attribute in the account schema with the string type. Refer to How do I add an attribute to a source schema in IdentityNow?
- **Provisioning**: The **manager** attribute is supported for the Add, Remove, and Set operations where the value of the manager can be set as either **objectId** or the **userPrincipalName**. Add the **manager** attribute in the Provisioning policy. The blank value in **Set** operation is applicable to remove the **manager** value at the Azure Active Directory managed system.

## Delta Aggregation

Delta Aggregation is supported for SailPoint Azure Active Directory connector. On Full Aggregation, the respective delta link of account and group aggregation are stored in the Application object which are used by Delta Aggregation to retrieve the changed data into IdentityIQ. Same values are updated after each respective delta aggregation.

- Account Delta Aggregation does not capture role assignment changes.
- Delta Aggregation triggered without first full aggregation would trigger respective full aggregation by default.
- Delta Aggregation does not capture changes in Exchange Online attributes and Shared Mailbox assignments.
- Delta Aggregation does not capture changes related to channel attributes and channel memberships.

## Partitioning Aggregation

Azure Active Directory connector supports partitioning aggregation based on search filters. To use partitioning feature perform the following:

1. Enable Partitioning on the aggregation task definition page by selecting the **Enable Partitioning** check box.
2. Add the following application configuration attribute:

   ```
   <entry key="userPartitions">
   ```

   The **userPartitions** configuration attribute is a multi-valued attribute. It’s value consists of different search filters for the attributes which are filterable like accountEnabled, city, displayName, mail, usageLocation and so on. For example:

   ```
   <entry key="userPartitions">
       <value>
           <List>
                <String>startswith(displayName,'J')</String>
                <String>startswith(givenName,'Smith')</String>
                <String>accountEnabled eq true</String>
                <String>userPrincipalName eq 'Paul@contoso.onmicrosoft.com'</String>
           </List>
       </value>
   </entry>
   ```

For large environments, for faster delta aggregation of the accounts, the connector supports partition delta aggregation.

Due to PowerShell limitations, partitioning aggregation does not aggregate Exchange Online attribute details and Shared Mailbox as entitlement.

## Supported operators are

- Logical operators: **and**, **or**
- Comparison operators: **'eq'(equal to)**, **ge' (greater than or equal to)** and **'le'(Less than or equal to)**
- **startswith**
- **any** is supported while querying multi valued properties

For example,

- `proxyAddresses/any(c:c eq 'smtp:Mary@contoso.com')`
- `proxyAddresses/any(c:startswith(c,'smtp:Mary@contoso.com'))`

# Schema Attributes

This section describes the different schema attributes. This connector currently supports the following types of objects:

- **Account:** Account objects are used when building identities Link objects.
- **Group**: The group schema is used when building Account-Group objects that are used to hold entitlements shared across identities.
- **Service Principal**: The Service Principal schema is used to represent Azure Service Principal which are identities created for use with applications, hosted services, and automated tools to access Azure resources.
- **Service Plan**: The Service Plan schema is used to represent Azure Service Plan (License Plans).
- **Management Group**: The Management group schema is used to represent Azure Management Groups which are created to allow Azure customers to organize their subscriptions and apply governance controls such as Role Based Access Control and Policies.
- **Subscription**: The Subscription schema is used to represent Azure Subscription objects which represent a signed agreement between a supplier and customer that the customer would receive and provide payment for regular products or services.
- **Resource Group**: The Resource group schema is used to represent Azure Resource Groups which are container objects that hold related resources for an Azure solution.
- **Azure Role Assignment**: The Azure Role Assignment schema is used to represent custom group objects that consist of the following elements:
    - Security Principle (user, groups, service principle, managed identity)
    - Role Definition
    - Access Scope

## Using Multiple Group Entitlements with pre-existing source

For the pre-existing Azure Active Directory sources perform the following steps to start using the Multiple Group Entitlements with your current source.

1. Use createSchema API to create new group schema for your source. Refer to the following respective sections for examples of API body content for adding to existing source:
    - servicePlan
    - Service Principal
2. Update Account Schema with the following steps:
    a. Get account schema using getSchema API.
    b. Copy the schema to a file and search for attribute corresponding to the group schema that is added in step 1.
    c. Add/update below two properties for the attribute found in previous step (2.b.)

```
"isGroup": true,
"schema": {"type": "CONNECTOR_SCHEMA","id": "<Schema_ID_From_Step1>","name":
"<Name_Of_Schema_Created_In_Step1>"}
```

d. Add the schema modified in step 2.c. to the payload to update account schema in source using replaceS-chema API.

## Account Attributes

### *accountEnabled*

True if the user is enabled; otherwise, false.

### *assignedLicenses*

List of the licenses that are assigned to the user.

### *assignedPlans*

Plans that are assigned to the user (Entitlement).

### *city*

City in which the user is located.

### *country*

Country/region in which the user is located.

### *department*

Name for the department the user belongs to.

### *dirSyncEnabled*

Indicates whether this user was synced from the on-premises directory.

### *disabledPlans*

Plans that are not assigned to the user.

### *displayName*

Name displayed in the address book for the user.

### *facsimileTelephoneNumber*

Telephone number of the user's business fax machine.

### *givenName*

First name of an user.

### *groups*

Groups assigned to an user (Entitlement).

### *servicePrincipals*

Enterprise Applications assigned to a user.

### *immutableId*

Property used to associate an on-premises Active Directory user account to their Azure Active Directory user account.

### *jobTitle*

User’s job title.

### *lastDirSyncTime*

Indicates the last time at which the user was synchronized with the on-premises directory.

### *mail*

The SMTP address for the user. For example, `john@contoso.onmicrosoft.com`

### *mailNickname*

Mail alias for the user.

### *signInNames*

Specifies the collection of sign-in names for a local account in an Azure Active Directory B2C tenant.

### *userIdentities*

Specifies the collection of userIdentities for a social user account in an Azure Active Directory B2C tenant.

### *creationType*

Indicates whether the user account is a local account for an Azure Active Directory B2C tenant.

### *manager*

Manager of the user. (Type: String).

Azure Active Directory Connector now provides support for provisioning of the ‘manager’ attribute. For more information, see Support for Provisioning Operations of ‘manager’ Attribute.

### *mobile*

Primary cellular telephone number for the user.

### *objectId*

Unique identifier for the user.

### *onPremisesSecurityIdentifier*

Contains the on-premises security identifier (SID) for the user that was synchronized from on-premises to the cloud.

### *otherMails*

A list of additional email addresses for the user.

### *passwordPolicies*

Specifies password policies for the user.

### *physicalDeliveryOfficeName*

Office location in the user's place of business.

### *postalCode*

ZIP OR postal code for the user's postal address.

### *preferredLanguage*

Preferred written or spoken language for a person.

### *proxyAddresses*

Proxy addresses.

For example, `["SMTP: bob@contoso.com", "smtp: bob@sales.contoso.com"]`

### *roles*

Administrator Role assigned to user (Entitlement).

### *sipProxyAddress*

Specifies the voice over IP (VOIP) session initiation protocol (SIP) address for the user.

### *state*

State or province in the user's address.

### *streetAddress*

Street address of the user's place of business.

### *surname*

Last name of the user.

### *telephoneNumber*

Primary telephone number of the user's place of business.

### *usageLocation*

A two letter country code indicating usage location.

### *userPrincipalName*

User principal name (UPN) of the user.

### *userType*

Type of the user.

### *channels*

List of channel membership of a user.

Type: Managed, Entitlement, Multi-Valued

## Group Attributes

### *description*

Description for the group.

### *dirSyncEnabled*

Indicates whether this group was synced from the on-premises directory.

### *displayName*

Display name for the group.

### *lastDirSyncTime*

Indicates the last time at which the group was synced with the on-premises directory.

### *mail*

SMTP address for the group.

### *mailEnabled*

Specifies whether the group is mail-enabled

### *mailNickname*

Mail alias for the group.

### *objectId*

Group ID.

### *onPremisesSecurityIdentifier*

Contains the on-premises security identifier (SID) for the group that was synchronized from on-premises to the cloud.

### *owners*

Owner of the group.

### *proxyAddresses*

Proxy addresses of the group.

### *securityEnabled*

Specifies whether the group is a security group.

### *teamsEnabled*

Specifies whether teams has been enabled for the group.

### *groupTypes*

Type of the group. Blank for Security and /or **Unified** for Office 365 type of groups.

For example, Unified for Office 365 group

## Service Principal

### *displayName*

ServicePrincipal name.

### *servicePrincipalType*

ServicePrincipal type.

### *objectId*

ServicePrincipal ID.

### *owners*

ServicePrincipal Owners. Type: String, Multi-Valued

### *appRoles*

ServicePrincipal Roles. Type: String, Multi-Valued

## Service Plan

### *objectId*

Unique identifier for the service plan which is a combination of skuId and servicePlanId separated by colon (:).

### *skuId*

Unique identifier (GUID) for the License Pack.

### *skuPartNumber*

The SKU part number of the License Pack. For example, AAD_PREMIUM.

### *appliesTo*

Object types to which the license pack can be assigned. For example, User, Company.

### *displayName*

Display name of service plan which is a combination of skuPartNumber and servicePlanName separated by colon (:).

### *servicePlanId*

Unique identifier of the service plan.

### *servicePlanName*

The name of the service plan.

### *provisioningStatus*

The provisioning status of the service plan.

## Channel

To manage the channel management objects, ensure that the following attributes are present in the group schema.

### *description*

Optional textual description for the channel.

### *displayName*

Channel name as it will appear to the user in Microsoft Teams.

Combination of team display name: channel display name.

### *id*

The channel's unique identifier.

Combination of team id: channel id

### *isFavoriteByDefault*

Indicates whether the channel must automatically be marked **favorite** for all members of the team. Default: false.

### *email*

The email address for sending messages to the channel. Read-only.

### *webUrl*

A hyperlink that will go to the channel in Microsoft Teams.

### *membershipType*

The type of the channel. Possible values are:

- standard: Channel inherits the list of members of the parent team.
- private: Channel can have members that are a subset of all the members on the parent team.

### *createdDateTime*

Read only. Timestamp at which the channel was created.

### *teamEmail*

The email address of the parent team.

***owners***

The list of owners of the private channel.

Properties: Multi-Valued

## Azure PIM Attributes

Add the following attributes to source schema.

| Name | Type | Details |
|---|---|---|
| azureEligibleRoles | String : Multivalued : Entitlement | List of Azure Roles for which user is eligible |
| azureADEligibleRoles | String : Multivalued : Entitlement | List of Azure AD Roles for which user is eligible |
| azureActiveRoles | String : Multivalued : Entitlement | List of Azure Roles assigned to user |
| azureADActiveRoles | String : Multivalued : Entitlement | List of Azure AD Roles assigned to user |

### Membership Aggregation

Membership Aggregation is part of account aggregation. PIM Membership aggregation will be added to existing membership aggregation flow and will be triggered only if flag is ON and required attributes are present in account schema Memberships will be aggregated if flag is true and account attributes are present in schema.

For example:

- If the enablePIM attribute is true and only Azure AD Roles : Active is present in schema, only Azure AD Roles membership for active assignments are aggregated.
- PIM Memberships are aggregated per page. For example, membership is fetched for users aggregated in a page.
- PIM Memberships for Azure resources are aggregated if Azure Management is enabled and respective attributes are present in the schema.

### Group Schema Objects

The following new group schema objects are supported.

### Azure AD Eligible Role

Object Type is `azureADEligibleRole`, Identity Attribute is `id`, and the Display Attribute is `displayName`.

***displayName***

Eligible role display name.

***id***

Azure AD eligible role ID.

***isBuiltIn***

Indicates whether role is custom or built-in.

***roleName***

Name of the role.

***scope***

Scope at which role can be assigned.

### Azure AD Active Role

Object Type is `azureADActiveRole`, Identity Attribute is `id`, and the Display Attribute is `displayName`.

***displayName***

Active AD role display name.

***id***

Azure AD active role ID.

***isBuiltIn***

Indicates whether role is custom or built-in.

***roleName***

Name of the role.

***scope***

Scope at which role can be assigned.

### Azure Active Role

Object Type is `azureActiveRole`, Identity Attribute is `id`, and the Display Attribute is `displayName`.

***displayName***

Active role display name.

***id***

Azure active role ID.

***isBuiltIn***

Indicates whether role is custom or built-in.

***roleName***

Name of the role.

***scope***

Scope at which role can be assigned.

### Azure Eligible Role

Object Type is `azureEligibleRole`, Identity Attribute is `id` , and the Display Attribute is `displayName`.

***displayName***

Eligible role display name.

***id***

Azure eligible role ID.

***isBuiltIn***

Indicates whether role is custom or built-in.

***roleName***

Name of the role.

***scope***

Scope at which role can be assigned.

## Management Groups

### *id*

Fully qualified ID for the management group.

### *displayName*

Name of the management group.

### *type*

Objecttype.

### *name*

Name of the management group.

## Subscriptions

### *id*

Subscription identifier with scope.

### *authorizationSource*

Authorization Source for the subscription. Default: RoleBased

### *managedByTenants*

List of tenants managing subscription.

Type: String, Multi-Valued

### *subscriptionId*

Unique identifier for the subscription.

### *tenantId*

Tenant Id with which subscription has trust relationship.

### *displayName*

Display name of the subscription.

### *state*

State of subscription, that is, it is enabled or disabled.

## Resource Group

### *id*

ID of the resource group.

### *name*

Name of the resource group.

### *type*

Type of the resource group.

### *location*

Location of the resource group.

### *provisioningState*

Provisioning state of the resource group.

## Azure Role Assignment

### *id*

Azure Role Assignment ID.

### *displayName*

Display name.

### *resource*

Display name of the resource on which role can be assigned.

### *roleName*

Display name of the role which can be assigned on resource.

# Provisioning Policy Attributes

This section lists different policy attributes for Azure Active Directory Connector.

> The attributes with * are required attributes.

## Create Account Policy

Following are the attributes in the create account policy:

### *accountType**

Default is User

To create user, set this value to **User**.

### *userPrincipalName**

User principal name (UPN) of the user.

For example, `jeff@contoso.onmicrosoft.com`

### *password**

Password for the new account.

### *displayName**

Display name of the account.

### *mailNickname**

Mail alias for the account.

### *accountEnabled*

Set it to false to create disabled account. Default: True

### *forceChangePasswordNextLogin*

If true, asks user to change password on next login. Default: True

### *department*

Department in which the user works.

### *jobTitle*

User's job title.

### *isFederatedDomain*

Set it true to create federated domain user. If this is checked and **immutableId** is not set then random **immutableId** value will be used.

### *immutableId*

This property is used to associate an on-premises Active Directory user account to their Azure AD user account; Populate this attribute with objectGUID of account from on-premises Active Directory to create federated user synchronized with on –premises Active Directory user.

### *passwordPolicies*

Specifies password policies for the user.

For example: DisablePasswordExpiration, DisableStrongPassword

### *otherMails*

Additional email addresses for the user.

### *givenName*

First name of the user.

### *surname*

Surname of the user.

### *usageLocation*

A two letter country code (ISO standard 3166). Required for users that will be assigned licenses.

### *country*

Country/region in which the user is located. For example, **US** or **UK**

### *state*

State or province in the user's address.

### *city*

City in which the user is located.

### *streetAddress*

Street address of the user's place of business.

### *postalCode*

Postal code for the user's postal address.

### *physicalDeliveryOfficeName*

Office location in the user's place of business.

### *preferredLanguage*

Preferred language for the user. Should follow ISO 639-1 Code.

For example, `en-US`

### *telephoneNumber*

Primary telephone number of the user's place of business.

### *mobile*

Primary cellular telephone number for the user.

### *facsimileTelephoneNumber*

Telephone number of the user's business fax machine.

### *servicePrincipalRoleId*

ServicePrincipal Role Id.

By default not present in the schema. It is required if you want to assign ServicePrincipal during account creation. For more information, see Assigning ServicePrincipal.

## Guest User (B2B) Support

Azure Active Directory Connector supports creation of Guest User (B2B) by sending invitations. Creation of Guest User (B2B) varies from normal user creation in terms of attributes provided during creation.

## Create Guest User (B2B) Account Policy

### *accountType**

Default is User

To create Guest User (B2B), set this value to **Guest User B2B**.

### *invitedUserEmailAddress**

Email address of the user.

### *inviteRedirectUrl**

The URL that the user will be redirected to after redemption.

### *sendInvitationMessage**

Set it to **False** if invitation email need not to be sent to the user. Default is True

### *invitedUserDisplayName*

The display name of the user being invited.

### *invitedUserUsageLocation*

A two letter country code indicating usage location (ISO standard 3166).

### *servicePrincipalRoleId*

ServicePrincipal Role Id.

By default not present in the schema. It is required if you want to assign ServicePrincipal during account creation. For more information, see Assigning ServicePrincipal.

## Local User (B2C) Support

Azure Active Directory Connector supports creation of Local Users. Creation of Local User account varies from normal user in terms of attributes provided during creation. Account creation also supports custom attribute.

## Create Local User (B2C) Account Policy

### *accountType**

Default is User

To create Local User (B2C), set this value to **Local User B2C**.

### *signInNameType**

Sign-in type for user in your Azure directory

### *signInNameValue**

Sign-in name for user.

### *localAccountDisplayName**

Display name of account.

### *b2cPassword**

Password for the new account.

### *enableLocalAccount*

Set it to false to create disabled account. Default: True

### *b2cForceChangePasswordNextLogin*

If true, asks user to change password on next login. Default: True

### *servicePrincipalRoleId*

ServicePrincipal Role Id.

By default not present in the schema. It is required if you want to assign ServicePrincipal during account creation. For more information, see Assigning ServicePrincipal.

> Custom user attribute can be added in B2C create account policy by appending suffix "_C" to the attribute.

## Create Group Policy

### *displayName**

Display name of the group.

### *owners*

Owner of the group.

### *mailNickname**

The mail alias for the group.

### *groupTypes*

Type of the group to be created, that is, Security or Office 365

### *teamsEnabled*

Specifies whether teams has been enabled for the group.

### *addOwnerAsMember*

Specifies if Owners must be added as Members in group.

> Native Identity (ObjectID) of the newly created group can be fetched from the result object using the following syntax:
> `result.getObject().getIdentity()`

## Update Group Policy

### *description*

Description of the group.

### *owners*

Owner of the group.

### *mailEnabled*

True if it is Office 365 or mail enabled security group. Read only.

### *securityEnabled*

True if it is security group or Office 365 type of group. Read only.

### *groupTypes*

Type of the group, that is, Blank for Security and /or Unified for Office 365 type of groups. Read only.

### *teamsEnabled*

Specifies whether teams has been enabled for the group.

For Teams creation owner having **Teams** license must be provided.

Teams can be enabled for existing Microsoft 365 group by selecting **Teams Enabled**.

For SAML Bearer Authentication method, Authorization User is the default owner of a Team but must be added as owner to enable Teams for existing Microsoft 365 Group.

## Create Channel Policy

Following are the attributes in the create channel policy:

### *teamEmail*

The email address of the parent team.

### *description*

Optional textual description for the channel.

### *displayName*

Channel name as it will appear to the user in Microsoft Teams.

Combination of team display name: channel display name.

### *membershipType*

The type of the channel. Possible values are:

- standard: Channel inherits the list of members of the parent team.
- private: Channel can have members that are a subset of all the members on the parent team.

### *owners*

Required only if membershipType=Private.

## Azure PIM Provisioning Policy

There is no requirement of addition/removal of the existing Create Profile (Provisioning Policy). However, while assigning/removing Azure /Azure AD Eligible/Active Role assignment, additional request details can be provided through additional attributes such as the following:

### *duration*

Specifies the duration for which role needs to be assigned to user from current time.

**Details**: For example, if role needs to be assigned for 6 months duration should be provided as P180D.

Default Values -- For Eligible Role: P365D,

For Active Role: P180D

**Example/Format**: ISO 8601 duration format must be used. For example,

6 months: P180D

1 week: P7D

8 hours: PT8H

### *startDateTime*

**Details**:It specifies the start date time from which role needs to be assigned to user. It is optional when duration is provided but required when endDatetime is provided.

**Example/Format**: The timestamp type represents date and time information using ISO 8601 format and is always in UTC time.

For example,

2021-11-19T09:40:27.91Z

### *endDateTime*

**Details**: It specifies the end date time for role assignment of the user. startDateTime must be provided along with this attribute.

**Example/Format**:The timestamp type represents date and time information using ISO 8601 format and is always in UTC time. For example, 2022-11-19T09:40:27.91Z

### *justification*

**Details:** It specifies the reason for role assignment/removal operation. Default Value: `Admin Action for user id <user id>`

#### Sample Provisioning Requests

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE ProvisioningPlan PUBLIC "sailpoint.dtd" "sailpoint.dtd">

<ProvisioningPlan nativeIdentity="8cb133a4-0cea-XXXX-8c02-XXXXXXXX">

  <AccountRequest application="Azure PIM" nativeIdentity="8cb133a4-0cea-XXXX-8c02-
XXXXXXXX" op="Modify">

    <AttributeRequest name="azureActiveRoles" op="Add">

        <Value>

            <List>

                <String>/subscriptions/3XXXXXXX8-c792-1212-9b4a-8cXXXXXXX4:c2f4ef07-c644-
48eb-af81-4b1b4947fb11</String>

                <String>/subscriptions/3XXXXXXX8-c792-1212-9b4a-8cXXXXXXX4:a2138dac-4907-
4679-a376-736901ed8ad8</String>

            </List>

           </Value>

        <Attributes>

            <Map>

```

```
                <entry key="duration" value="P10D"/>

                <entry key="justification" value="Role Required to Manage Subscription"/>

            </Map>

        </Attributes>

    </AttributeRequest>

    <AttributeRequest name="azureEligibleRoles" op="Add">

        <Value>

            <List>

                <String>/subscriptions/3XXXXXXX8-c792-1212-9b4a-8cXXXXXXX4/re-
sourceGroups/DEV-ENV-RG:5e467623-bb1f-42f4-a55d-6e525e11384b</String>

                <String>/subscriptions/3XXXXXXX8-c792-1212-9b4a-8cXXXXXXX4/re-
sourceGroups/DEV-ENV-RG:4f8fab4f-1852-4a58-a46a-8eaf358af14a</String>

                <entry key="justification" value="Role Required to Manage Dev Resource
Group"/>

            </List>

        </Value>

        <Attributes>

            <Map>

                <entry key="startDateTime" value="2021-11-19T09:40:27.91Z"/>

                <entry key="endDateTime" value="2021-12-19T09:40:27.91Z"/>

            </Map>

        </Attributes>

    </AttributeRequest>

  </AccountRequest>

</ProvisioningPlan>
```

# Troubleshooting

### *Error: Add entitlement request fails*

While adding a Service Principal Name to a user, the add entitlement request fails with the following error message:

```
Response Code - 400 Error - Permission being assigned was not found on
application
```

**Resolution**: Verify if the user already has the Service Principal Name added in entitlements.

### *Error: Test Connection fails*

Test Connection fails with the following error message when Azure Active Directory application is checked for 'SAML Bearer Assertion' Grant Type:

```
OAuth2Exception [toString()=connector.common.oauth2.OAuth2Exception: Unable to
generate access token. Response returned:
```

```
{"error":"invalid_grant","error_description":"AADSTS50008: The SAML token is
invalid.\r\nTrace ID: a74df376-3ede-4c17-ba34-b352079e3300\r\nCorrelation ID:
f8c370ec-4ef6-48a4-a393-0297a5ce3b20\r\nTimestamp: 2020-05-04 05:58:16Z","error_
codes":[50008],"timestamp":"2020-05-04 05:58:16Z","trace_id":"a74df376-3ede-4c17-
ba34-b352079e3300","correlation_id":"f8c370ec-4ef6-48a4-a393-
0297a5ce3b20","error_uri":"https://login.microsoftonline.com/error?code=50008"}
```

```
]
```

**Resolution**: Verify if the time zone of ADFS machine is in synchronize with Azure time zone, that is, UTC. If not, change the ADFS machine time and re-start the ADFS services.

### *Error: Test Connection /Account Aggregation fails*

Test Connection/ Account Aggregation fails with the following error message when Azure Active Directory application is checked for **SAML Bearer Assertion** Grant Type:

```
Error - invalid_grant: AADSTS5000811: Unable to verify token signature. The
signing key identifier does not match any valid registered keys.
```

**Resolution**: Microsoft recommends to execute the following command from PowerShell running on ADFS server to manually renew token signing certificates:

```
Update-MSOLFederatedDomain -DomainName <domain>
```

### *Error: Provisioning of Exchange attribute fails*

Provisioning of Exchange attribute fails if account aggregation is in progress. Exchange online module supports maximum three sessions per user hence limiting the parallel operations in execution.

**Resolution**: Connector uses all the three sessions while performing aggregation. Hence to improve the aggregation performance no other operation must be performed using the configured user.

### *Error: Account Aggregation fails*

When new attributes are added in account schema, Get Object / Account Aggregation fails with the following error message:

```
Error - 501 This operation target is not yet supported
```

**Resolution**: Microsoft Graph API supports some of the following attributes only for retrieving single user:

*Attributes*: aboutMe, birthday, hireDate, interests, mySite, pastProjects, preferredName, responsibilities, schools, skills, mailboxSettings

Addition of such attributes while retrieving collection of users leads to aggregation failure. Hence verify if the newly added attributes are in the list of these attributes and remove them from the schema.

## *Error: Update account fails*

Update account fails for new attribute additions in the provisioning policy.

**Resolution**: Due to API limitations following are some of the attributes that are not applicable for updating user properties, fetching the attributes for a user through Account Aggregation or Get Account operation:

*Attributes*: birthday, mySite, interests, pastProjects, preferredName, responsibilities, schools, skills and mail-boxSettings

## *Error: Unexpected errors*

```
An unexpected 'PrimitiveValue' node was found when reading from the JSON reader.
A 'StartArray' node was expected.
```

Or

```
Invalid value for the Property
```

**Resolution**: Ensure that the data type of the attribute is added correctly to the provisioning policy. For more information, see Microsoft Documentation.

> For update account operation, the **businessPhones** attribute accepts only single value with the data type as String collection.

## *Error: Error occurred while authenticating User: Xyz*

```
Unable to use PTA with SASL
```

**Resolution:**Check the UPN name being used.

Kerberos Authentication requires userPrincipalName as the primary requirement, and due to differences in the domain for userPrincipalName, the PTA authentication is failing. You must use the correct domain.

For example if you have two domains:

- Abc.com
- xyz. abc.com

and you want to perform PTA for users present at `xyz.abc.com`, the format should be `user-name@xyz.abc.com.`