

# How to Use Directory Extension Attributes in Entra ID for Custom Claims, SCIM Provisioning and Dynamic Groups

Learn how to create and use directory extension attributes in Entra ID to extend the schema and add custom attributes to the directory objects for various scenarios.

Posted on February 8, 2024

11 minute read

## Customize your Entra ID Tenant with Directory Extension Attributes and use in Custom SSO Claims, SCIM Provisioning and Dynamic Groups

### Table of Contents

1. [Introduction](#)
2. [Create a Directory Extension Attribute](#)
3. [Using Directory Extension Attributes](#)
4. [Add Directory Extension Attributes in SAML Claims](#)
5. [Add Directory Extension Attributes in OIDC/OAuth JWT Claims](#)
  - [Add Directory Extension Attributes as Optional Claims](#)
6. [Use Directory Extension Attributes in SCIM Provisioning](#)
  - [Add Directory Extension Attributes for SCIM Provisioning using Graph API](#)
7. [Use the Directory Extension Attributes in Dynamic Groups](#)
8. [Conclusion](#)

### Introduction

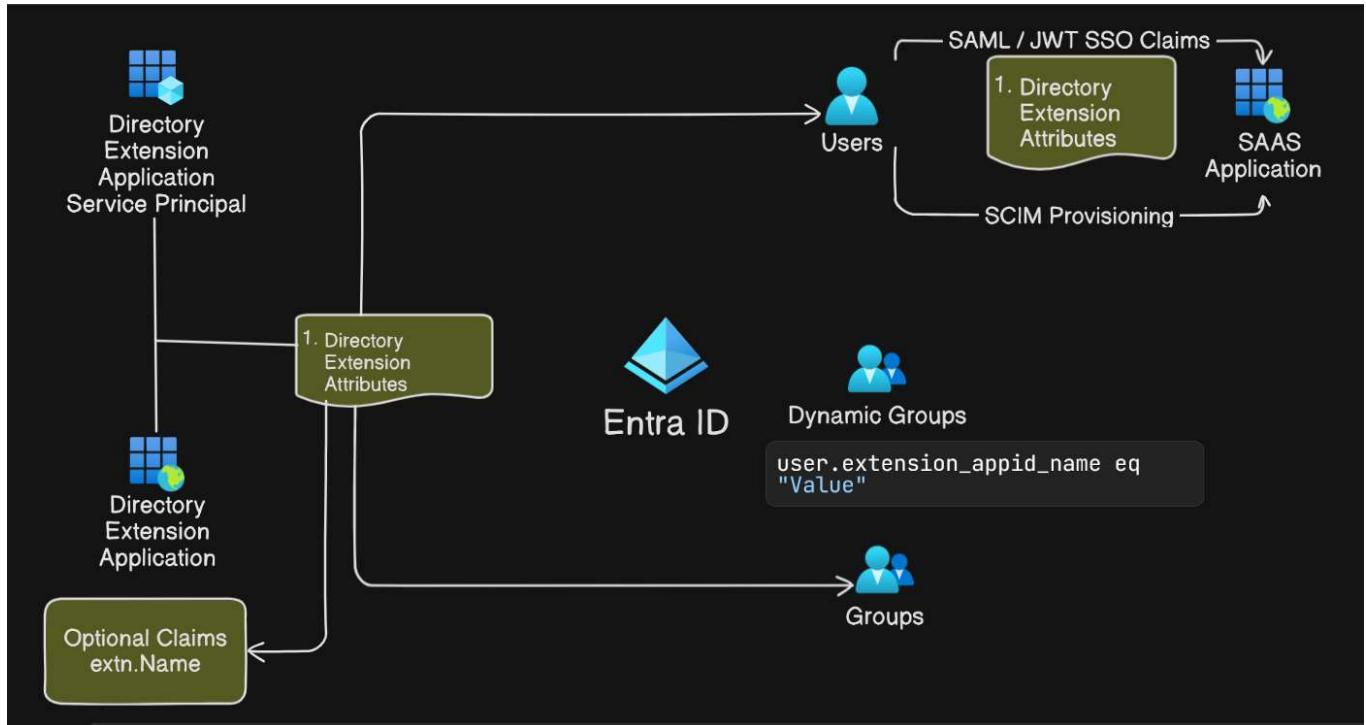
In the realm of identity management, seasoned professionals may find themselves reminiscing about the power and ease of customization offered by Microsoft Identity Manager (MIM). Its ability to extend the default schema and effortlessly add custom attributes to existing or new object types made it a beloved tool for integration. With Microsoft Entra ID, we lack some of the features that we are used to in MIM. However, Microsoft has introduced various ways to extend the schema and add custom attributes to the directory objects. In this article, we will explore how to leverage directory extension attributes in Entra ID for various use cases such as custom claims, SCIM provisioning, and dynamic group membership rules.

1. Extension Attributes 1 - 15
2. Directory Extensions
3. Schema Extensions
4. Open Extensions

These extension attributes can be used to store additional information that is not available in the default set of attributes. In many cases, we often need these attributes in downstream applications for various purposes such as user provisioning, identity synchronization, or attribute mapping. You can see this nice overview of the different ways to extend the schema [here](#). Among these, the most commonly used method for extending the schema is the use of directory extension attributes as it can be used with the following features, not available with the other methods:

1. Create dynamic membership rules using the extension attributes.
2. Use the attributes in SCIM provisioning.
3. Use the attributes in SAML and OIDC claims.

In this article, we will explore how to leverage directory extension attributes in Entra ID for previously mentioned use cases.



## Create a Directory Extension Attribute

At the time of writing this article, there is no GUI option available to create directory extension attributes. However, we can use the Microsoft Graph API to create these attributes. The following is an example of how to create a directory extension attribute using the Microsoft Graph API with PowerShell. One important thing to note is that directory extension attributes are tied to an owner application. It is common to use one app to create and manage all the extension Attributes in a tenant.

```
$ExtensionApp = New-MgApplication -DisplayName "MIM Tenant Schema Extension App"
New-MgServicePrincipal -AppId $ExtensionApp.AppId
```

You can also create it using Graph Explorer or any other Graph API client. The following is an example of how to create a directory extension attribute using Graph Explorer.

```

POST https://graph.microsoft.com/v1.0/applications
Content-Type: application/json

{
  "displayName": "MIM Tenant Schema Extension App",
}

POST https://graph.microsoft.com/v1.0/servicePrincipals
Content-Type: application/json

{
  "appId": "{app-id-of-the-application-created-above}"
}

```

We need to create a service principal for the application to be able to use it as the default PowerShell cmdlets to create a multi-tenant application. After this step, directory extensions become available and consumed for users in the tenant.

we can verify in the Entra ID portal that the application and the service principal have been created.

| 8 applications found |                                 | Application (client) ID               | Created on | Certificates & secrets                     |
|----------------------|---------------------------------|---------------------------------------|------------|--|
|                      | Get Claims from LDAP Directory  | 1ff82508-2c9f-4f40-a67e-bacd72dd660e  | 6/1/2024   | -  |
|                      | jwt.ms                          | ee123c8e-db6a-4f43-b3a4-8e37a15b1be0  | 6/1/2024   | <span style="color: green;">Current</span> |
|                      | LDAPCustomClaimProvider         | 6642cf2d-0365-4a3b-aba7-771e29fb0f020 | 6/1/2024   | -  |
|                      | MIM Tenant Schema Extension App | d7a076d9-1c10-4133-b712-ad6caf3819c9  | 8/2/2024   | -  |
|                      | Mint                            | 754029bc-6a9e-4784-a59a-c7dfcd471361  | 25/1/2024  | -  |
|                      | terraform auth                  | b83f8757-20e4-46cd-a70d-984aeed21ff1  | 28/1/2024  | <span style="color: green;">Current</span> |
|                      | Test Application                | f7b09ed8-2da7-4e1c-9fa0-d2585bb2510c  | 30/1/2024  | <span style="color: green;">Current</span> |
|                      | TestAPI                         | 6e3df5f1-974f-4305-9361-948f43cc43dd  | 30/1/2024  | -  |

| 7 applications found |                                    | Application ID                  | Homepage URL                       | Created on |
|----------------------|------------------------------------|---------------------------------|------------------------------------|------------|
|                      | jwt.ms                             | ee123c8e-db6a-4f43-b3a4-8e3...  | https://account.activedirectory... | 6/1/2024   |
|                      | Mint                               | 754029bc-6a9e-4784-a59a-c7d...  | https://account.activedirectory... | 25/1/2024  |
|                      | Microsoft Graph Command Line Tools | 14d82eec-204b-4c2f-b7e8-296...  | https://docs.microsoft.com/en...   | 19/1/2024  |
|                      | terraform auth                     | b83f8757-20e4-46cd-a70d-984...  | https://docs.microsoft.com/en...   | 28/1/2024  |
|                      | TestAPI                            | 6e3df5f1-974f-4305-9361-948f... | https://developer.microsoft.co...  | 30/1/2024  |
|                      | Graph Explorer                     | de8bc8b5-d9f9-48b1-a8ad-b74...  | https://developer.microsoft.co...  | 19/1/2024  |
|                      | MIM Tenant Schema Extension App    | d7a076d9-1c10-4133-b712-ad...   | https://account.activedirectory... | 8/2/2024   |

Now that we have created our parent application to manage the extension attributes, we can create the extension attribute using the following PowerShell script.

```

$ExtensionProperty1 = New-MgApplicationExtensionProperty -Name "SkillSet" -DataType "String" -TargetObjects "User" -
ApplicationId $ExtensionApp.Id
$ExtensionProperty2 = New-MgApplicationExtensionProperty -Name "supervisoryOrg" -DataType "String" -TargetObjects "User" -
ApplicationId $ExtensionApp.Id

```

You can also create it using Graph Explorer or any other Graph API client. The following is an example of how to create a directory extension attribute using Graph Explorer.

```

POST https://graph.microsoft.com/v1.0/applications/{id}/extensionProperties
Body:
{
  "dataType": "String",
  "name": "SkillSet",
  "targetObjects": [
    "User"
  ]
}

```

```

POST https://graph.microsoft.com/v1.0/applications/{id}/extensionProperties
Body:
{
  "dataType": "String",
  "name": "supervisoryOrg",
  "targetObjects": [
    "User"
  ]
}

```

We cannot verify the extension attribute in the EntraID portal. However, we can use the graph API (or Powershell) to verify that the extension attribute has been created.

```
Get-MgApplicationExtensionProperty -ApplicationId $ExtensionApp.Id | Format-Table Name, DataType, TargetObjects
```

| DeletedDateTime | ID                                   | AppDisplayName | DataType | IsSyncedFromOnPremises | Name  | TargetObjects |
|-----------------|--------------------------------------|----------------|----------|------------------------|---|---------------|
|                 | a23f14ca-3cec-4661-b3eb-929c663d0078 |                | String   | False                  | extension_d7a076d91c104133b712ad6caf3819c9_supervisoryOrg | {user}        |
|                 | d1e6bfc9-21c8-4adc-85aa-29d9afc8c52e |                | String   | False                  | extension_d7a076d91c104133b712ad6caf3819c9_SkillSet       | {user}        |

```
GET https://graph.microsoft.com/v1.0/applications/{id}/extensionProperties
```

```
{
  "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#applications('9c42bcc-e568-45ed-83eb-c99066e29f34')/extensionProperties",
  "@microsoft.graph.tips": "Use $select to choose only the properties your app needs, as this can lead to performance improvements. For example: GET applications('<guid>')/extensionProperties?$select=appDisplayName,dataType",
  "value": [
    {
      "id": "a23f14ca-3cec-4661-b3eb-929c663d0078",
      "deletedDateTime": null,
      "appDisplayName": "",
      "dataType": "String",
      "isMultiValued": false,
      "isSyncedFromOnPremises": false,
      "name": "extension_d7a076d91c104133b712ad6caf3819c9_supervisoryOrg",
      "targetObjects": [
        "User"
      ]
    },
    {
      "id": "d1e6bfc9-21c8-4adc-85aa-29d9afc8c52e",
      "deletedDateTime": null,
      "appDisplayName": "",
      "dataType": "String",
      "isMultiValued": false,
      "isSyncedFromOnPremises": false,
      "name": "extension_d7a076d91c104133b712ad6caf3819c9_SkillSet",
      "targetObjects": [
        "User"
      ]
    }
  ]
}
```

As you can notice, directory extension attributes follow a certain naming convention of the following format:

`extension_{Application (client) Id}_{name}`. The `Application (client) Id` is the application ID of the parent application that owns the extension attribute. The `name` is the name of the extension attribute. We can use this naming convention to reference the extension attribute in the subsequent sections.

Similarly, we can list all the directory extension attributes in the tenant using the following Graph API (or PowerShell) command.

```
Get-MgDirectoryObjectAvailableExtensionProperty
```

```
POST https://graph.microsoft.com/v1.0/directoryObjects/microsoft.graph.getAvailableExtensionProperties
```

## Using Directory Extension Attributes

Similar to the creation of Extension Attributes, at the time of writing this article, there is no GUI option available to use directory extension attributes. However, we can use the Microsoft Graph API to use these attributes. The following is an example of how to use directory extension attributes to enrich the user object.

```
$UserID = "{User Object ID}"
$BodyParam = @{
    extension_d7a076d91c104133b712ad6caf3819c9_supervisoryOrg = "IAM",
    extension_d7a076d91c104133b712ad6caf3819c9_SkillSet = "IAM"
}
Update-MgUser -UserId $UserID -BodyParameter $BodyParam
```

```
PATCH https://graph.microsoft.com/v1.0/users/{id}
Content-Type: application/json

{
    "extension_d7a076d91c104133b712ad6caf3819c9_supervisoryOrg": "IAM"
}
```

We can validate that the extension attribute has been updated using the following Graph API (or PowerShell) script.

```
# Get all Extension Properties - if run on a regular basis these could be cached
$extensions = Get-MgDirectoryObjectAvailableExtensionProperty

# Standard User Properties we want to Fetch
$properties = @("Displayname", "Id", "UserPrincipalName")

# Fetch all users and the Extension Properties
$users = Get-MgUser -All -Property ($properties + $extensions.Name)

$allUsersParsed = [System.Collections.ArrayList]::new()

# Flatten user data by merging extension properties from nested hashtables into a single-level hashtable
# Also filter out unnecessary fields from the full Graph User Schema to leave populated properties
Foreach ($u in $users){
    $userParsed = @{}
    Foreach ($prop in $properties) {
        $userParsed.$prop = $u.$prop
    }
    $userParsed += $u.AdditionalProperties
    $allUsersParsed.Add([pscustomobject]$userParsed) | Out-Null
}

$allUsersParsed | Format-Table ($properties + $extensions.Name)
```

| Displayname | Id                                  | UserPrincipalName             | extension_d7a076d91c104133b712ad6caf3819c9_supervisoryOrg | extension_d7a076d91c104133b712ad6caf3819c9_Skillset |
|-------------|-------------------------------------|-------------------------------|---|---|
| Adele Vance | e1a2bfe0-bd34-4302-9153-4fd793c032e | Adelevv@0323s.onmicrosoft.com | IAM   | IAM   |

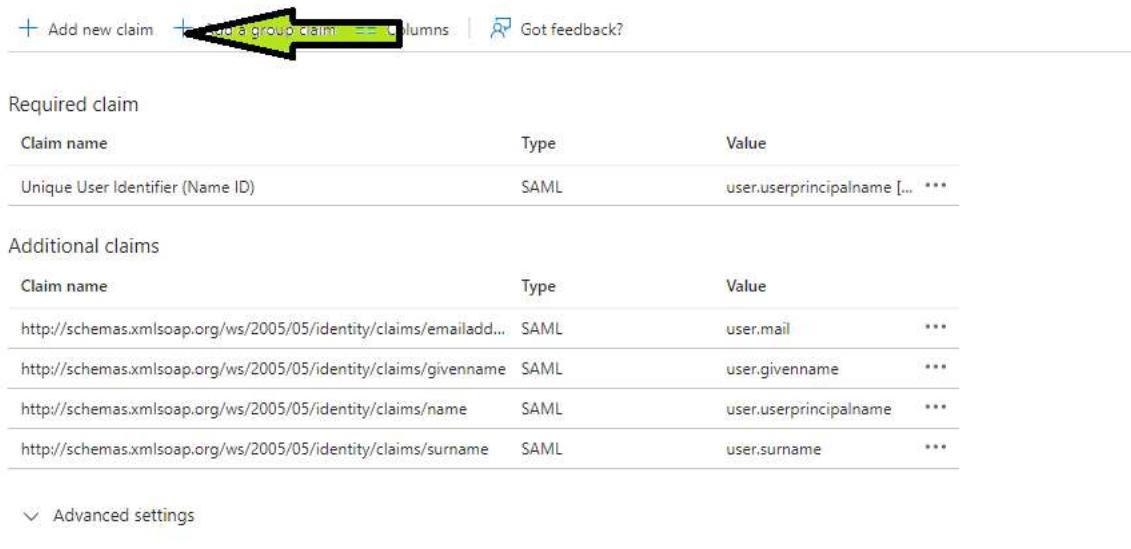
```
GET https://graph.microsoft.com/v1.0/users?  
$select=displayName,id,userPrincipalName,extension_d7a076d91c104133b712ad6caf3819c9_supervisoryOrg,extension_d7a076d91c104133b712ad6caf3819c9_SkillSet
```

```
□ {  
    "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users(id,displayName,userPrincipalName,extension_d7a076d91c104133b712ad6caf3819c9_supervisoryOrg,  
extension_d7a076d91c104133b712ad6caf3819c9_SkillSet)/$entity",  
    "id": "e1a2bfe0-bd34-4302-9153-4fdf793c032e",  
    "displayName": "Adele Vance",  
    "userPrincipalName": "AdeleV@03z3s.onmicrosoft.com",  
    "extension_d7a076d91c104133b712ad6caf3819c9_SkillSet": "IAM",  
    "extension_d7a076d91c104133b712ad6caf3819c9_supervisoryOrg": "IAM"  
}
```

Here is one information, the [Graph Connector of Microsoft Identity Manager](#) supports the use of extension attributes from version [1.1.301.0 August 2020](#). This means that you can use the extension attributes in MIM for user provisioning and synchronization from your source systems.

## Add Directory Extension Attributes in SAML Claims

1. On the Attributes & Claims blade, select Add new claim or edit an existing claim.



The screenshot shows the 'Add new claim' interface in the Azure portal. At the top, there are buttons for 'Add new claim', 'Add a group claim', 'Columns', and 'Got feedback?'. Below this, under 'Required claim', there is a table with one row: 'Unique User Identifier (Name ID)' of type 'SAML' with value 'user.userprincipalname [...] \*\*\*'. Under 'Additional claims', there is another table with four rows corresponding to SAML claims for email, given name, name, and surname, each with the same value mapping. At the bottom, there is a 'Advanced settings' section with a collapse arrow.

| Claim name                       | Type | Value                            |
|----------------------------------|------|----------------------------------|
| Unique User Identifier (Name ID) | SAML | user.userprincipalname [...] *** |

| Claim name  | Type | Value                      |
|---|------|----------------------------|
| http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailadd... | SAML | user.mail ***              |
| http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname   | SAML | user.givenname ***         |
| http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name        | SAML | user.userprincipalname *** |
| http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname     | SAML | user.surname ***           |

2. Select Directory Schema Extension from the Source Checkbox and Select the parent application from the application picker, created in the previous section.

## Manage claim ...

Save Got feedback?

Name \*

Namespace  Enter a namespace URI

Choose name format

Source \*  Attribute  Transformation  Directory schema extension

Schema extension attribute Undefined

Claim conditions

Advanced SAML claims options

## Select Application

Select the application which contains the desired extension attributes

Try changing or adding filters if you don't see what you're looking for.

Search

9 results found

All App registrations

|                          | Name                             | Type              | Details                               |
|--------------------------|----------------------------------|-------------------|---------------------------------------|
| <input type="checkbox"/> | Get Claims from LDAP Directory   | App registrati... | 1ff82508-2c9f-4f40-a67e-bacd72dd668e  |
| <input type="checkbox"/> | jwt.ms                           | App registrati... | ee123c8e-db6a-4f43-b3a4-8e37a15b1be0  |
| <input type="checkbox"/> | LDAPCustomClaimProvider          | App registrati... | 6642cf2d-0365-4a3b-aba7-771e29fb020   |
| <input type="checkbox"/> | MIM Tenant Schema Extension A... | App registrati... | d7a076d9-1c10-4133-b712-ad6caf3819c9  |
| <input type="checkbox"/> | Mint                             | App registrati... | 754029bc-6a9e-4784-a59a-c7dfcd471361  |
| <input type="checkbox"/> | terraform auth                   | App registrati... | b83f8757-20e4-46cd-a70d-984eaed21ff1  |
| <input type="checkbox"/> | Test Application                 | App registrati... | fddcbf84-4d26-4115-ac87-052c5abceefaf |
| <input type="checkbox"/> | Test Application                 | App registrati... | f7b89ed8-2da7-4e1c-9fa0-d2585bb2510c  |
| <input type="checkbox"/> | TestAPI                          | App registrati... | 6e3df5f1-974f-4305-9361-948f43cc43dd  |

## Add Extension Attributes

X

Extension attributes from 'MIM Tenant Schema Extension App' application.

[Select Application](#)

The list of available extension attributes configured against the application: 'MIM Tenant Schema Extension App (d7a076d9-1c10-4133-b712-ad6caf3819c9)'.

 Search for attribute

| Name                | Data Type | Synced From ... |
|---------------------|-----------|-----------------|
| user.supervisoryOrg | String    | false           |
| user.skillSet       | String    | false           |

### Selected Item

user.supervisoryOrg (extension\_d7a076d91c104133b712ad6caf3819c9\_supervisoryOrg)

3. Select Add to add the selection to the claims.

Add

Cancel

- Click Save to add the attribute.

Manage claim ...

- We can verify that the SAML claim has been returned in our test application.

## Add Directory Extension Attributes in OIDC/OAuth JWT Claims

In the scenario we have described above create all the extension attributes in a parent application and make the attribute available for other applications. We need to create a claims mapping policy to emit the extension attributes as claims in the token. The following is an example of how to create a claims mapping policy using the Microsoft Graph API with PowerShell.

```
$claimsMappingPolicy = [ordered]@{
    "ClaimsMappingPolicy" = [ordered]@{
        "Version" = 1
        "IncludeBasicClaimSet" = $true
        "ClaimsSchema" = @(
            [ordered]@{
                "Source" = "user"
                "ExtensionID" = "extension_d7a076d91c104133b712ad6caf3819c9_supervisoryOrg"
                "JwtClaimType" = "superVisoryOrg"
            },
            [ordered]@{
                "Source" = "user"
                "ExtensionID" = "extension_d7a076d91c104133b712ad6caf3819c9_SkillSet"
                "JwtClaimType" = "SkillSet"
            }
        )
    }
}

$definition = @{$claimsMappingPolicy | ConvertTo-Json -Depth 4}
$bodyParam = [ordered]@{
    "displayName" = "Extension Claim Mapping Policy"
    "definition" = $definition
}

$claimspolicy = New-MgPolicyClaimMappingPolicy -Definition $definition -DisplayName "Extension Claim Mapping Policy"
Multiclouds" -Debug
```

Assign the claims mapping policy to the service principal in our tenant. The following is an example of how to assign the claims mapping policy using the Microsoft Graph API with PowerShell.

```

$servicePrincipalId = "{Service Principal Object ID}"
$params = @{
    "@odata.id" = "https://graph.microsoft.com/v1.0/policies/claimsMappingPolicies/$(claimsPolicy.Id)"
}

New-MgServicePrincipalClaimMappingPolicyByRef -ServicePrincipalId $servicePrincipalId -BodyParameter $params

```

```

POST https://graph.microsoft.com/v1.0/servicePrincipals/{service-principal-id}/claimsMappingPolicies/$ref
Content-Type: application/json

{
    "@odata.id": "https://graph.microsoft.com/v1.0/policies/claimsMappingPolicies/{policy-id-from-above}"
}

```

We can verify that the claims mapping policy has been created using the following Graph API (or PowerShell) script.

```
Get-MgPolicyClaimMappingPolicy --ClaimsMappingPolicyId $($claimsPolicy.Id)
```

```

PS [REDACTED] > Get-MgPolicyClaimMappingPolicy -ClaimsMappingPolicyId $($claimsPolicy.Id) | Select -ExpandProperty Definition
@{
    "ClaimsMappingPolicy": {
        "Version": 1,
        "IncludeBasicClaimSet": true,
        "ClaimsSchema": [
            {
                "Source": "user",
                "ExtensionID": "extension_d7a076d91c104133b712ad6caf3819c9_supervisoryOrg",
                "JwtClaimType": "superVisoryOrg"
            },
            {
                "Source": "user",
                "ExtensionID": "extension_d7a076d91c104133b712ad6caf3819c9_SkillSet",
                "JwtClaimType": "SkillSet"
            }
        ]
    }
}

```

```
GET https://graph.microsoft.com/v1.0/policies/claimsMappingPolicies/{policy-id-from-above}
```

```

{
    "@odata.context": "https://graph.microsoft.com/beta/$metadata#policies/claimsMappingPolicies/$entity",
    "@microsoft.graph.tips": "Use $select to choose only the properties your app needs, as this can lead to performance improvements. For example: GET policies/claimsMappingPolicies('<guid>')?$select=definition,isOrganizationDefault",
    "id": "93bd14cd-6fa5-4568-85cd-769a94ca911b",
    "deletedDateTime": null,
    "definition": [
        {
            "ClaimsMappingPolicy": {
                "Version": 1,
                "IncludeBasicClaimSet": true,
                "ClaimsSchema": [
                    {
                        "Source": "user",
                        "ExtensionID": "extension_d7a076d91c104133b712ad6caf3819c9_supervisoryOrg",
                        "JwtClaimType": "superVisoryOrg"
                    },
                    {
                        "Source": "user",
                        "ExtensionID": "extension_d7a076d91c104133b712ad6caf3819c9_SkillSet",
                        "JwtClaimType": "SkillSet"
                    }
                ],
                "displayName": "Extension Claim Mapping Policy Multiclouds",
                "isOrganizationDefault": false
            }
        }
    ]
}

```

It is time to put our configuration to the test. Sign in to the application with a user that has the extension attributes populated. We can verify that the claims have been returned in our test application.

```

rAJg.",
  "sub": "ntixk1j1daBj__rSCAtpLcVCKYeRsfhRbdiDa5atczQ",
  "tid": "b5683b08-cb53-45a8-b4ff-1531a0ed2f38",
  "uti": "3MMiSSwQ5UGH3rj0R2Y7AA",
  "ver": "2.0",
  "superVisoryOrg": "IAM",
  "SkillSet": "IAM"
}

```

We can also use the same method to send the extension attribute in SAML Claims. We just need to update the Claims Mapping Policy to include the SAMLClaim Type attribute in our policy.

```

```powershell
$claimsMappingPolicy = [ordered]@{
  "ClaimsMappingPolicy" = [ordered]@{
    "Version" = 1
    "IncludeBasicClaimSet" = $true
    "ClaimsSchema" = @(
      [ordered]@{
        "Source" = "user"
        "ExtensionID" = "extension_d7a076d91c104133b712ad6caf3819c9_supervisoryOrg"
        "JwtClaimType" = "superVisoryOrg"
        "SamlClaimType" = "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/superVisoryOrg"
      },
      [ordered]@{
        "Source" = "user"
        "ExtensionID" = "extension_d7a076d91c104133b712ad6caf3819c9_SkillSet"
        "JwtClaimType" = "SkillSet"
        "SamlClaimType" = "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/SkillSet"
      }
    )
  }
}

$definition = @($claimsMappingPolicy | ConvertTo-Json -Depth 4)
$BodyParam = [ordered]@{
  "displayName" = "Extension Claim Mapping Policy Multicloud SAML"
  "definition" = $definition
}

$claimspolicy = New-MgPolicyClaimMappingPolicy -BodyParameter $BodyParam -Debug

```

You will notice in the Claims and Attributes tab that SAML Claims are overridden by the policy.

**⚠** This configuration was overwritten by a claim mapping policy created via Graph/PowerShell. Learn more. → 

### Required claim

| Claim name                       | Type | Value                            |
|----------------------------------|------|----------------------------------|
| Unique User Identifier (Name ID) | SAML | user.userprincipalname [... ***] |

### Additional claims

| Claim name                                                        | Type | Value                            |
|-------------------------------------------------------------------|------|----------------------------------|
| http://schemas.microsoft.com/ws/2008/06/identity/claims/groups    | SAML | user.groups [None] ***           |
| http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailadd... | SAML | user.mail ***                    |
| http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname   | SAML | user.givenname ***               |
| http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name        | SAML | user.userprincipalname ***       |
| http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname     | SAML | user.surname ***                 |
| SupervisoryOrg                                                    | SAML | user.supervisoryorg (exte... *** |

 Advanced settings

We can verify that the SAML claims have been returned in our test application.

http://schemas.xmlsoap.org/ws/2005/05/identity/claims/SkillSet

IAM

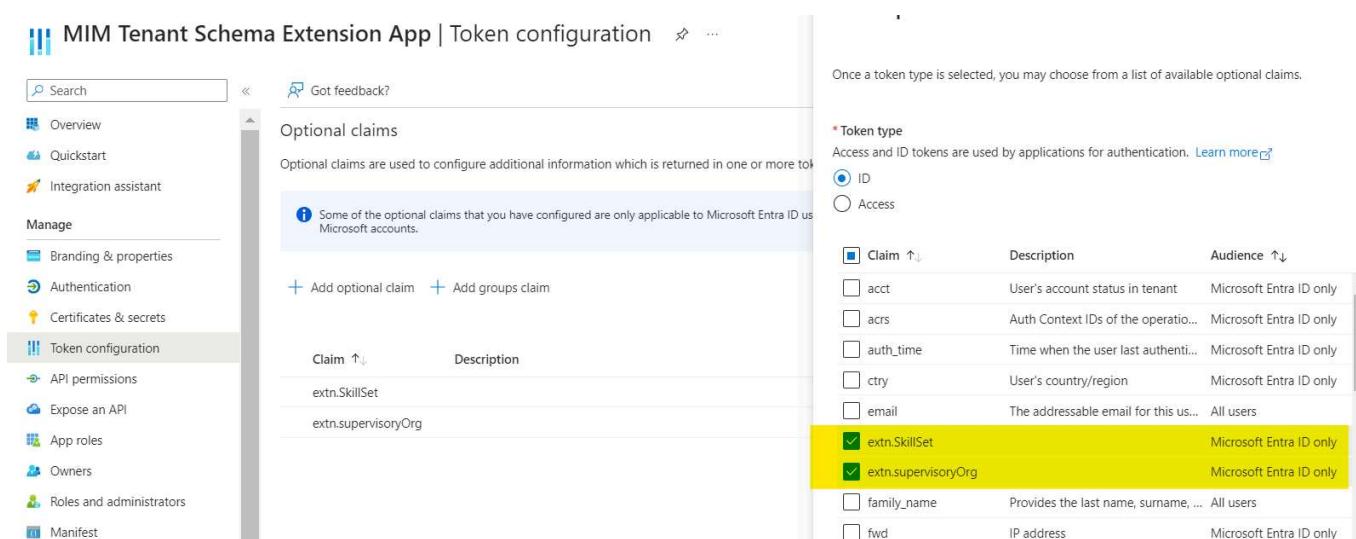
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/supervisoryOrg

IAM

## Add Directory Extension Attributes as Optional Claims

Applications, that require the extension attributes defined in the same application can add the defined application as optional claims or add the extension attributes as optional claims in the application manifest.

Honestly just use the Token Configuration page, because if you mess up in typing, the Token Configuration page will show you the error, which is kind of nice. Here is how you select this extension property.



The screenshot shows the 'Token configuration' section of the MIM Tenant Schema Extension App. On the left, there's a sidebar with links like Overview, Quickstart, Integration assistant, Manage, Branding & properties, Authentication, Certificates & secrets, Token configuration (which is selected), API permissions, Expose an API, App roles, Owners, Roles and administrators, and Manifest. The main area has a search bar and a 'Got feedback?' link. Under 'Optional claims', it says: 'Optional claims are used to configure additional information which is returned in one or more tokens.' A note states: 'Some of the optional claims that you have configured are only applicable to Microsoft Entra ID users.' Below this are buttons for 'Add optional claim' and 'Add groups claim'. A table lists optional claims with columns for Claim, Description, and Audience. Two items are checked: 'extn.SkillSet' (Audience: Microsoft Entra ID only) and 'extn.supervisoryOrg' (Audience: Microsoft Entra ID only). Other items listed include 'acct', 'acrs', 'auth\_time', 'ctry', 'email', 'family\_name', and 'fwd'.

This is the change it made to manifest.xml.

```
"optionalClaims": [
  "idToken": [
    {
      "name": "extension_d7a076d91c104133b712ad6caf3819c9_supervisoryOrg",
      "source": "user",
      "essential": false,
      "additionalProperties": []
    },
    {
      "name": "extension_d7a076d91c104133b712ad6caf3819c9_SkillSet",
      "source": "user",
      "essential": false,
      "additionalProperties": []
    }
  ]
}
```

## **Use Directory Extension Attributes in SCIM Provisioning**

This is probably the least known capability of the directory extension attributes. For example, let's say the knowledge management application in our fictitious company uses the **SkillSet** attribute to assign the right content to the right users. We can use the **SkillSet** extension attribute to provision the user in the knowledge management application.

However, the process of extending the list of source attributes available for provisioning from the Entra ID Directory is not straightforward.

Microsoft provides a special [url](#) to allow modification of the **Entra ID Directory**.

1. Go to the Azure Portal using the special [URL](#).
2. Go to **Microsoft Entra ID** -> **Enterprise applications** -> **Select the Application from the list** -> **Provisioning** -> Click on **Edit Provisioning**.

3. Click on **Mappings** and then select **Provision Microsoft Entra ID Users**.

## Provisioning

Save Discard

Use Microsoft Entra to manage the creation and synchronization of user accounts in Test Application based on user and group assignment.

Admin Credentials

Mappings

### Mappings

Mappings allow you to define how data should flow between Microsoft Entra ID and custom appssso.

| Name                                | Enabled |
|-------------------------------------|---------|
| Provision Microsoft Entra ID Groups | Yes     |
| Provision Microsoft Entra ID Users  | Yes     |

Restore default mappings

Settings

Provisioning Status

On

Off

4. Scroll down to the checkbox and select **Show advanced options**. You will be able to see the option to **Edit attribute list for Microsoft Entra ID**. Click on **Edit attribute list for Microsoft Entra ID**.

5. Add the extension attributes. In our case, we will add `sextension_d7a076d91c104133b712ad6caf3819c9_supervisoryOrg` and `Sextension_d7a076d91c104133b712ad6caf3819c9_SkillSet` to the list of supported attributes.

Save Discard

othermails

String



passwordProfile.password

String



physicalDeliveryOfficeName

String



preferredLanguage

String



proxyAddresses

String



surname

String



telephoneNumber

String



usageLocation

String



userPrincipalName

String



extension\_d7a076d91c104133b712ad6caf3819c9\_supervisoryOrg

String



extension\_d7a076d91c104133b712ad6caf3819c9\_SkillSet

String



6. Click **Save** to save the changes.

Note that directory extension attribute names are case-sensitive, make sure you enter it in the same format as defined in the directory. Provisioning multi-valued directory extension attributes is not supported.

Add the attributes to the application's attribute mapping for User Object.

## Edit Attribute

A mapping lets you define how the attributes in one class of Microsoft Entra object (e.g. Users) should flow to and from this application.

Mapping type ⓘ

Direct

Source attribute \* ⓘ

SkillSet (extension\_d7a076d91c104133b712ad6caf3819c9\_SkillSet)

Default value if null (optional) ⓘ

Target attribute \* ⓘ

nickName

Match objects using this attribute

No

Matching precedence ⓘ

Apply this mapping ⓘ

Always

Ok

We can verify provisioning by checking the logs in the provisioning tab.

|            |           |                               |           |
|------------|-----------|-------------------------------|-----------|
| externalId | SuryenduB | [mailNickname]                | SuryenduB |
| nickName   | IAM       | [extension_d7a076d91c10413... | IAM       |

## Add Directory Extension Attributes for SCIM Provisioning using Graph API

We can also use the Microsoft Graph API to add the extension attributes to the list of supported attributes for SCIM provisioning.

Make a GET Call to retrieve the existing Schema.

```
GET https://graph.microsoft.com/beta/servicePrincipals/{service-principal-id}/jobs/{sync-job-id}/schema
```

1. Copy the existing Schema and modify the Entra ID Directory Section -> User Object -> Attribute list to include the extension attributes.

```

        },
        "defaultValue": "",
        "exportMissingReferences": false,
        "flowBehavior": "FlowWhenChanged",
        "flowType": "Always",
        "matchingPriority": 0,
        "targetAttributeName": "nickName",
        "source": {
            "expression": "[extension_d7a076d91c104133b712ad6caf3819c9_SkillSet]",
            "name": "extension_d7a076d91c104133b712ad6caf3819c9_SkillSet",
            "type": "Attribute",
            "parameters": []
        }
    },

```

2. Modify the Synchronization Rules -> Object Mappings -> Select the User Object(depending on the version it can have different names Provision Microsoft Entra ID Users or Provision Azure AD Users) -> Add the extension attribute to the application's attribute mapping for User Object.

```

{
    "defaultValue": "",
    "exportMissingReferences": false,
    "flowBehavior": "FlowWhenChanged",
    "flowType": "Always",
    "matchingPriority": 0,
    "targetAttributeName": "nickName",
    "source": {
        "expression": "[extension_d7a076d91c104133b712ad6caf3819c9_SkillSet]",
        "name": "extension_d7a076d91c104133b712ad6caf3819c9_SkillSet",
        "type": "Attribute",
        "parameters": []
    }
},

```

3. Copy the modified schema and make a PUT call to update the provisioning job schema. Send the modified schema as the request body.

```
PUT https://graph.microsoft.com/beta/servicePrincipals/{service-principal-id}/jobs/{job-id}/schema
Content-Type: application/json
Body:
{
    "directories": [
        {
            "name": "Microsoft Entra ID",
            "objects": [
                {
                    "name": "User",
                    "attributes": [
                        {
                            "anchor": false,
                            "caseExact": false,
                            "defaultValue": null,
                            "flowNullValues": false,
                            "multivalued": false,
                            "mutability": "ReadWrite",
                            "name": "extension_d7a076d91c104133b712ad6caf3819c9_SkillSet",
                            "required": false,
                            "type": "String",
                            "apiExpressions": [],
                            "metadata": [],
                            "referencedObjects": []
                        }
                    ]
                },
            ],
        },
        {
            "name": "Salesforce",
        }
    ],
    "synchronizationRules": [
        {
            "name": "USERGROUP_OUTBOUND_USERGROUP",
            "sourceDirectoryName": "Microsoft Entra ID",
            "targetDirectoryName": "Knowledge Management Application",
            "objectMappings": [
                {
                    "sourceObjectName": "User",
                    "targetObjectName": "urn:ietf:params:scim:schemas:enterprise:2.0:User",
                    "attributeMappings": [
                        {
                            "defaultValue": "",
                            "exportMissingReferences": false,
                            "flowBehavior": "FlowWhenChanged",
                            "flowType": "Always",
                            "matchingPriority": 0,
                            "targetAttributeName": "nickName",
                            "source": {
                                "expression": "[extension_d7a076d91c104133b712ad6caf3819c9_SkillSet]",
                                "name": "extension_d7a076d91c104133b712ad6caf3819c9_SkillSet",
                                "type": "Attribute",
                                "parameters": []
                            }
                        }
                    ],
                }
            ],
        }
    ]
}
```

# Use the Directory Extension Attributes in Dynamic Groups

Many organizations use dynamic groups to manage access to resources, drive business processes, lifecycle workflows and more. You can use directory extension attributes to create dynamic group membership rules. For example, you can create a dynamic group that includes all users who have a specific skill set or are part of a specific supervisory organization.

1. Create a new group with the **Membership type** set to **Dynamic User**.

New Group ...

Got feedback?

Group type \* ⓘ  
Security

Group name \* ⓘ  
IAM Engineers

Group description ⓘ  
Enter a description for the group

Microsoft Entra roles can be assigned to the group ⓘ  
Yes No

Membership type \* ⓘ  
Dynamic User

Owners  
No owners selected

Dynamic user members \* ⓘ  
Add dynamic query

2. Click **Add dynamic query** and click on **Get custom extension properties**

You can use the rule builder or rule syntax text box to create or edit a dynamic membership rule. ⓘ [Learn more](#)

| And/Or               | Property                                                                                                                |
|----------------------|-------------------------------------------------------------------------------------------------------------------------|
|                      | <Choose a Property>                                                                                                     |
| + Add expression     | + Get custom extension properties ⓘ  |
| Rule syntax          |                                                                                                                         |
| <input type="text"/> |                                                                                                                         |

3. Enter the application ID of the parent application that owns the extension attribute and click on **Refresh Properties**.

### Get custom extension prop...

Enter an application ID to get all the available custom extension properties available for creating a rule. ⓘ

d7a076d9-1c10-4133-b712-ad6caf3819c9 ✓

**Refresh properties**

4. Select the extension attribute from the list and click **Add**. Add the condition and click **Add Expression**.

The screenshot shows a search interface for extension attributes. At the top, there are three dropdown menus: 'Choose a Property', 'Choose an Operator', and 'Add a value'. Below these are two buttons: '+ Add expression' and '+ Get'. A scrollable list of extension attributes is displayed, starting with 'extensionAttribute8' and ending with 'extension\_d7a076d91c104133b712ad6caf3819c9\_SkillSet'. The list includes several specific application IDs like 'extension\_d7a076d91c104133b712ad6caf3819c9\_supervisoryOrg'.

5. Click **Save** to create the dynamic group. Wait for a few minutes for the dynamic membership to be calculated.

The screenshot shows the 'Members' section of a dynamic group configuration. On the left, a sidebar lists 'Overview', 'Diagnose and solve problems', 'Properties', 'Members' (which is selected and highlighted in grey), 'Owners', 'Roles and administrators', and 'Administrative units'. The main area has tabs for 'Direct members' (which is selected) and 'All members'. It features a search bar 'Search by name' and a 'Add filters' button. A table lists the members: 'Adele Vance' (User) and 'FNU LNU' (User). Both entries have a small circular profile picture next to their names.

# Conclusion

Directory extension attributes offer a powerful way to enhance the functionality and customization of Entra ID. By leveraging these attributes, organizations can map and use custom claims in various scenarios, such as SAML claims, dynamic group membership rules, and SCIM provisioning.

I have shown how to add extension attributes to claims mapping policies, configure SAML claims, create dynamic groups based on extension attributes, and customize attribute mappings for SCIM provisioning. It also highlights the importance of using the Token Configuration page and the Azure Portal's special URL for modifying the Entra ID Directory.

Overall, directory extension attributes provide flexibility and extensibility to Entra ID applications, allowing organizations to tailor their identity and access management solutions to meet specific business requirements. By using directory extension attributes, you can leverage the power and flexibility of Entra ID to meet your specific business requirements.

Tags: Entra ID | Directory Extension Attributes | Custom Claims | SCIM Provisioning | Dynamic Groups | Identity Management | SAML | OIDC



[← PREVIOUS POST](#)

[NEXT POST →](#)

## Comments (0)

≡ best



Leave a comment

Sign In



Suryendu Bhattacharyya • 2024 • [MIMAndBeyond](#)

Powered by [Beautiful Jekyll](#)