Configure how users consent to applications - Microsoft Entra ID

By omondiatieno

Archived: 2026-04-05 13:12:40 UTC

In this article, you learn how to configure user consent settings in Microsoft Entra ID to control when and how users grant permissions to applications. This guidance helps IT admins reduce security risks by restricting or disabling user consent.

Before an application can access your organization's data, a user must grant the application permissions to do so. Different permissions allow different levels of access. By default, all users are allowed to consent to applications for permissions that don't require administrator consent. For example, by default, a user can consent to allow an app to access their mailbox but can't consent to allow an app unfettered access to read and write to all files in your organization.

To reduce the risk of malicious applications attempting to trick users into granting them access to your organization's data, we recommend that you allow user consent only for applications that have been published by a verified publisher.

Note

Applications that require users to be assigned to the application must have their permissions consented by an administrator, even if the user consent policies for your directory would otherwise allow a user to consent on behalf of themselves.

Prerequisites

To configure user consent, you need:

- A user account. If you don't already have one, you can create an account for free.
- A Privileged Role Administrator role.
- A Global Administrator role is only required when using the Microsoft Entra admin center.

Configure user consent settings

You can configure user consent settings in Microsoft Entra ID using either the Microsoft Entra admin center, Microsoft Graph PowerShell, or Microsoft Graph API. The settings you configure apply to all users in your organization.

Configure user consent in Microsoft Entra admin center

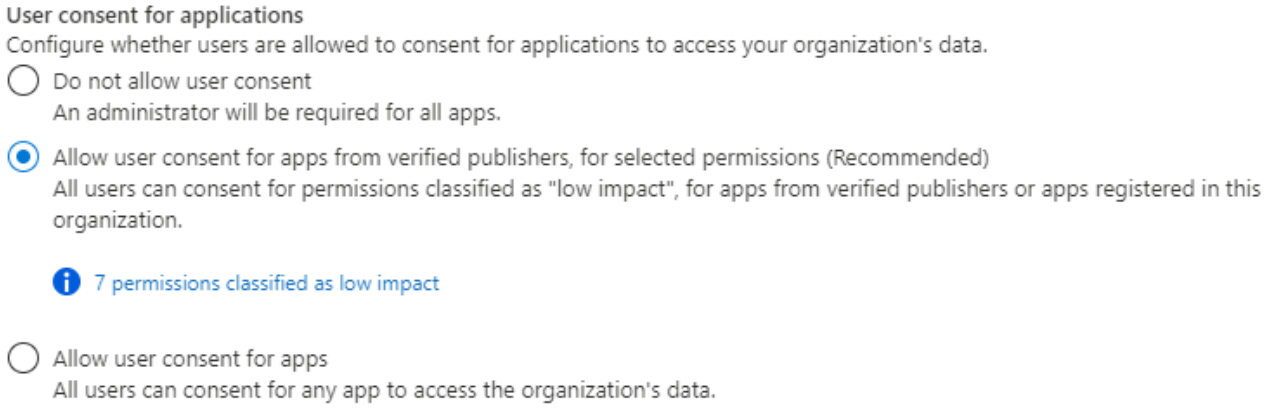
To configure user consent settings through the Microsoft Entra admin center:

1. Sign in to the Microsoft Entra admin center as a Global Administrator.

2. Browse to **Identity** > **Applications** > **Enterprise apps** > **Consent and permissions** > **User consent settings**.

3. Under **User consent for applications**, select which consent setting you want to configure for all users.

4. Select **Save** to save your settings.



Understand authorization and permission grant policies in Microsoft Graph PowerShell

To configure user consent settings programmatically using Microsoft Graph PowerShell, it's important to understand the distinction between the tenant-wide **authorization policy** and individual **permission grant policies**. The `authorizationPolicy`, retrieved using Update-MgPolicyAuthorizationPolicy governs global settings such as whether users can consent to apps and which permission grant policies are assigned to the default user role. For example, you can disable user consent while still allowing developers to manage permissions for the apps they own by assigning only `ManagePermissionGrantsForOwnedResource.DeveloperConsent` in the `permissionGrantPoliciesAssigned` collection.

On the other hand, the permissionGrantPolicies endpoint lists your current permission grant policies. These policies determine what permissions can be granted to applications and under what circumstances. Each policy 'includes' certain conditions, but 'excludes' others. When a user tries to consent to an application, the system checks the permission grant policies to see if any of them apply to the user's request. For example, the low-risk policy would allow users to consent to those permissions configured as 'low risk'. It includes these low-risk policies (as a GUID). In another scenario, if a user tries to consent in a context that matches the 'AdminOnly' policy, they're unable to consent.

Note

Before updating consent settings with a `Update-MgPolicyPermissionGrantPolicy` command, always retrieve the current `authorizationPolicy` to identify which permission grant policies are already assigned. This ensures you preserve necessary permissions—such as those enabling developers to manage consent for apps they own—and avoid unintentionally removing existing functionality.

To choose which app consent policy governs user consent for applications, use the Microsoft Graph PowerShell module. The cmdlets used here are included in the Microsoft.Graph.Identity.SignIns module.

Connect to Microsoft Graph PowerShell using the least-privilege permission needed. For reading the current user consent settings, use *Policy.Read.All*. For reading and changing the user consent settings, use *Policy.ReadWrite.Authorization*. You need to sign in as a Privileged Role Administrator.

```
Connect-MgGraph -Scopes "Policy.ReadWrite.Authorization"
```

Disable user consent using Microsoft Graph PowerShell

To disable user consent, ensure that the consent policies (`PermissionGrantPoliciesAssigned`) include other current `ManagePermissionGrantsForOwnedResource.*` policies if any while updating the collection. This way, you can maintain your current configuration for user consent settings and other resource consent settings.

```
 # only exclude user consent policy
$body = @{
    "permissionGrantPolicyIdsAssignedToDefaultUserRole" = @(
        "managePermissionGrantsForOwnedResource.{other-current-policies}"
    )
}
Update-MgPolicyAuthorizationPolicy -BodyParameter $body
```

Allow user consent subject to an app consent policy using PowerShell

To allow user consent, choose which app consent policy should govern users' authorization to grant consent to apps. Ensure that the consent policies (`PermissionGrantPoliciesAssigned`) include other current `ManagePermissionGrantsForOwnedResource.*` policies if any while updating the collection. This way, you can maintain your current configuration for user consent settings and other resource consent settings.

```
 $body = @{
    "permissionGrantPolicyIdsAssignedToDefaultUserRole" = @(
        "managePermissionGrantsForSelf.{consent-policy-id}",
        "managePermissionGrantsForOwnedResource.{other-current-policies}"
    )
}
Update-MgPolicyAuthorizationPolicy -BodyParameter $body
```

Replace `{consent-policy-id}` with the ID of the policy you want to apply. You can choose a custom app consent policy that you've created, or you can choose from the following built-in policies:

ID	Description
microsoft-user-default-low	**Allow user consent for apps from verified publishers, for selected permissions** Allow limited user consent only for apps from verified publishers and apps that are registered in your tenant, and only for permissions that you classify as *low impact*.

ID	Description
	(Remember to classify permissions to select which permissions users are allowed to consent to.)
microsoft-user-default-legacy	**Allow user consent for apps** This option allows all users to consent to any permission that doesn't require admin consent, for any application

For example, to enable user consent subject to the built-in policy `microsoft-user-default-low`, run the following commands:

```
$body = @{
   "permissionGrantPolicyIdsAssignedToDefaultUserRole" = @(
       "managePermissionGrantsForSelf.managePermissionGrantsForSelf.microsoft-user-default-low",
       "managePermissionGrantsForOwnedResource.{other-current-policies}"
   )
}
```

Understand authorization and permission grant policies in Microsoft Graph

To configure user consent settings programmatically using Microsoft Graph, it's important to understand the distinction between the tenant-wide **authorization policy** and individual **permission grant policies**. The `authorizationPolicy` (retrieved using `GET https://graph.microsoft.com/v1.0/policies/authorizationPolicy/authorizationPolicy`) governs global settings such as whether users can consent to apps and which permission grant policies are assigned to the default user role. For example, you can disable user consent while still allowing developers to manage permissions for the apps they own by assigning only `ManagePermissionGrantsForOwnedResource.DeveloperConsent` in the `permissionGrantPoliciesAssigned` collection.

On the other hand, the `permissionGrantPolicies` endpoint (`GET https://graph.microsoft.com/v1.0/policies/permissionGrantPolicies`) lists your current permission grant policies. These policies determine what permissions can be granted to applications and under what circumstances. Each policy 'includes' certain conditions, but 'excludes' others. When a user tries to consent to an application, the system checks the permission grant policies to see if any of them apply to the user's request. For example, the low-risk policy would allow users to consent to those permissions configured as 'low risk'. It includes these low-risk policies (as a GUID). In another scenario, if a user tries to consent in a context that matches the 'AdminOnly' policy, they're unable to consent.

Note

Before updating consent settings with a `PATCH` request, always retrieve the current `authorizationPolicy` to identify which permission grant policies are already assigned. This ensures you preserve necessary permissions—such as those enabling developers to manage consent for apps they own—and avoid unintentionally removing existing functionality.

Use the Graph Explorer to choose which app consent policy governs user consent for applications. You need to sign in as a Privileged Role Administrator.

Disable user consent using Microsoft Graph

To disable user consent, ensure that the consent policies (`PermissionGrantPoliciesAssigned`) include other current `ManagePermissionGrantsForOwnedResource.*` policies if any while updating the collection. This way, you can maintain your current configuration for user consent settings and other resource consent settings.

```
 PATCH https://graph.microsoft.com/v1.0/policies/authorizationPolicy
{
   "defaultUserRolePermissions": {
       "permissionGrantPoliciesAssigned": [
           "managePermissionGrantsForOwnedResource.{other-current-policies}"
        ]
    }
}
```

Allow user consent subject to an app consent policy using Microsoft Graph

To allow user consent, choose which app consent policy should govern users' authorization to grant consent to apps. Ensure that the consent policies (`PermissionGrantPoliciesAssigned`) include other current `ManagePermissionGrantsForOwnedResource.*` policies if any while updating the collection. This way, you can maintain your current configuration for user consent settings and other resource consent settings.

```
 PATCH https://graph.microsoft.com/v1.0/policies/authorizationPolicy

{
    "defaultUserRolePermissions": {
        "managePermissionGrantsForSelf.{consent-policy-id}",
        "managePermissionGrantsForOwnedResource.{other-current-policies}"
   }
}
```

Replace `{consent-policy-id}` with the ID of the policy you want to apply. You can choose a custom app consent policy that you've created, or you can choose from the following built-in policies:

ID	Description
microsoft-user-default-low	**Allow user consent for apps from verified publishers, for selected permissions** Allow limited user consent only for apps from verified publishers and apps that are registered in your tenant, and only for permissions that you classify as *low impact*. (Remember to classify permissions to select which permissions users are allowed to consent to.)

ID	Description
microsoft-user-default-legacy	**Allow user consent for apps** This option allows all users to consent to any permission that doesn't require admin consent, for any application

For example, to enable user consent subject to the built-in policy `microsoft-user-default-low`, use the following PATCH command:

```
 PATCH https://graph.microsoft.com/v1.0/policies/authorizationPolicy

{
    "defaultUserRolePermissions": {
        "permissionGrantPoliciesAssigned": [
            "managePermissionGrantsForSelf.microsoft-user-default-low",
            "managePermissionGrantsForOwnedResource.{other-current-policies}"
        ]
    }
}
```

Any updates to user consent settings only affect future consent operations for applications. Existing consent grants remain unchanged, and users continue to have access based on the permissions previously granted. To learn how to revoke existing consent grants, see Review permissions granted to enterprise applications.

Tip

To allow users to request an administrator's review and approval of an application that the user isn't allowed to consent to, enable the admin consent workflow. For example, you might do this when user consent has been disabled or when an application is requesting permissions that the user isn't allowed to grant.

Next steps

- Manage app consent policies
- Configure the admin consent workflow
