200 XP

Use roles to control resource access

10 minutes

Built-in roles for Azure Resources (uses PowerShell)

Azure AD provides several *built-in roles* to cover the most common security scenarios. To understand how the roles work, let's examine three roles that apply to all resource types:

- **Owner**, which has full access to all resources, including the right to delegate access to others.
- **Contributor**, which can create and manage all types of Azure resources but can’t grant access to others.
- **Reader**, which can view existing Azure resources.

Role definitions

Each role is a set of properties defined in a JavaScript Object Notation (JSON) file. This role definition includes a **Name**, **ID**, and **Description**. It also includes the allowable permissions (**Actions**), denied permissions (**NotActions**), and scope (for example, read access) for the role.

For the Owner role, that means all actions, indicated by an asterisk (*); no denied actions; and all scopes, indicated by a forward slash (/).

You can get this information using the PowerShell `Get-AzRmRoleDefinition` cmdlet.

PowerShell

```
Get-AzRoleDefinition
```

This code should produce the following output:

Output

```
Name             : Owner
Id               : 8e3af657-a8ff-443c-a75c-2fe8c4bcb635
IsCustom         : False
Description      : Lets you manage everything, including access to resources.
Actions          : {*}
NotActions       : {}
DataActions      : {}
```

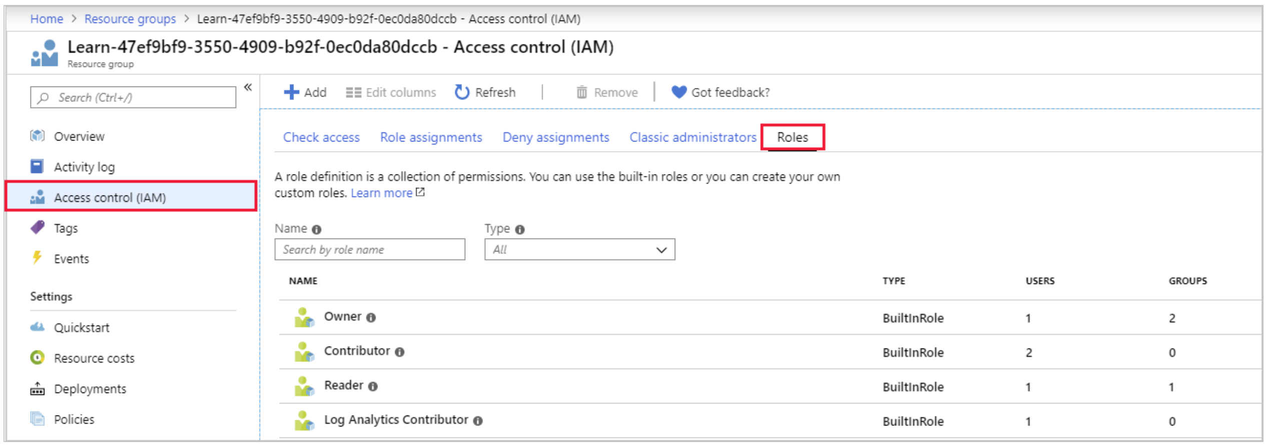
```
NotDataActions   : {}
AssignableScopes : {/}
```

Try the same for the **Contributor** and **Reader** roles to see the actions allowed and denied.

Examine the built-in roles

Next, let's explore some of the other built-in roles.

1. Open the Azure portal .
2. Select **Resource groups** from the left nav bar.
3. Select the resource group.
4. Select the **Access control (IAM)** item from the sidebar menu.
5. Select the **Roles** tab as shown below to see the list of available roles.



A role definition is a collection of permissions. You can use the built-in roles or you can create your own custom roles. Learn more

Name: Search by role name
Type: All

NAME	TYPE	USERS	GROUPS
Owner	BuiltInRole	1	2
Contributor	BuiltInRole	2	0
Reader	BuiltInRole	1	1
Log Analytics Contributor	BuiltInRole	1	0

What's a role definition?

A role definition is a collection of permissions. A role definition lists the operations that can be performed, such as **read, write, and delete**. It can also list the operations that can't be performed or operations related to underlying data.

As previously described, a role definition has the following structure.

Name	Description
`Id`	Unique identifier for the role, assigned by Azure.
`IsCustom`	True if a custom role, False if a built-in role.
`Description`	A readable description of the role.

Name	Description
`Actions []`	Allowed permissions, `*` indicates all.
`NotActions []`	Denied permissions.
`DataActions []`	Specific allowed permissions as applied to data, for example `Microsoft.Storage/storageAccounts/blobServices/containers/blobs/read`
`NotDataActions []`	Specific denied permissions as applied to data.
`AssignableScopes []`	Scopes where this role applies. `/` indicates global, but can reach into a hierarchical tree.

This structure is represented as JSON when used in role-based access control (RBAC) or from the underlying API. For example, here's the **Contributor** role definition in JSON format.

JSON

```json
{
  "Name": "Contributor",
  "Id": "b24988ac-6180-42a0-ab88-20f7382dd24c",
  "IsCustom": false,
  "Description": "Lets you manage everything except access to resources.",
  "Actions": [
    "*"
  ],
  "NotActions": [
    "Microsoft.Authorization/*/Delete",
    "Microsoft.Authorization/*/Write",
    "Microsoft.Authorization/elevateAccess/Action"
  ],
  "DataActions": [],
  "NotDataActions": [],
  "AssignableScopes": [
    "/"
  ]
}
```

Actions and NotActions

You can tailor the `Actions` and `NotActions` properties to grant and deny the exact permissions you need. These properties are always in the format: `{Company}.{ProviderName}/{resourceType}/{action}`.

As an example, here are the actions for the three roles we looked at previously.

Built-in Role	Actions	NotActions
Owner (allow all actions)	`*`	-
Contributor (allow all actions except writing or deleting role assignments)	`*`	`Microsoft.Authorization/*/Delete`, `Microsoft.Authorization/*/Write`, `Microsoft.Authorization/*/elevateAccess/Action`
Reader (allow all read actions)	`*/read`	-

The wildcard (`*`) operation under `Actions` indicates that the principal assigned to this role can perform all actions, or in other words, it can manage everything. Including actions defined in the future, as Azure adds new resource types. With the **Reader** role, only the `read` action is allowed.

The operations under `NotActions` are subtracted from `Actions`. With the **Contributor** role, `NotActions` removes this role's ability to manage access to resources and also assign access to resources.

DataActions and NotDataActions

Data operations are specified in the `DataActions` and `NotDataActions` properties. Data operations can be specified separately from the management operations. This prevents current role assignments with wildcards (`*`) from suddenly having access to data. Here are some data operations that can be specified in `DataActions` and `NotDataActions`:

- Read a list of blobs in a container
- Write a storage blob in a container
- Delete a message in a queue

Only data operations can be added to the `DataActions` and `NotDataActions` properties. Resource providers identify which operations are data operations by setting the `isDataAction` property to `true`. Roles that do not have data operations can omit these properties from the role definition.

These actions work exactly like their management cousins. You specify actions you want to allow (or `*` for all) and then provide a list of specific actions to remove in the `NotDataActions` collection. Here are some examples, you can find the full list of actions and data actions in the resource provider documentation:

Data operation	Description

Data operation	Description
`Microsoft.Storage/storageAccounts/blobServices/containers/blobs/delete`	Delete blob data
`Microsoft.Compute/virtualMachines/login/action`	Log in to a VM as a regular user
`Microsoft.EventHub/namespaces/messages/send/action`	Send messages on an event hub
`Microsoft.Storage/storageAccounts/fileServices/fileshares/files/read`	Return a file/folder or list of files/folders
`Microsoft.Storage/storageAccounts/queueServices/queues/messages/read`	Read a message from a queue

Assignable Scopes

Defining the **Actions** and **NotActions** properties is not enough to fully implement a role. You also need to properly scope your role.

The **AssignableScopes** property of the role specifies the scopes (subscriptions, resource groups, or resources) within which the role is available for assignment. You can make the custom role available for assignment just in the subscriptions or resource groups that need it, thus avoiding cluttering the user experience for the rest of the subscriptions or resource groups.

Here are some examples.

To	Use Scope
Restrict to a subscription.	`"/subscriptions/{sub-id}"`
Restrict to a specific resource group on a specific subscription.	`"/subscriptions/{sub-id}/resourceGroups/{rg-name}"`
Restrict to a specific resource.	`"/subscriptions/{sub-id}/resourceGroups/{rg-name}/{resource-name}"`
Make a role available for assignment in two subscriptions.	`"/subscriptions/{sub-id}", "/subscriptions/{sub-id}"`

Creating roles

Azure AD comes with built-in roles that are likely to cover 99% of what you'll ever want to do. It is preferable to use a built-in role if possible. However, you can create custom roles if you find it necessary.

> ⓘ **Note**
>
> Custom role creation requires Azure AD Premium P1 or P2 and cannot be done in the free tier.

Creating a new role can be done through several mechanisms:

- **Azure portal**. You can use the Azure portal to create a custom role - **Azure Active Directory** > **Roles and administrators** > **New custom role**.
- **Azure PowerShell**. You can use the `New-AzADMSRoleDefinition` cmdlet to define a new role.
- **Azure Graph API**. You can use a REST call to the Graph API to programmatically create a new role.

The summary includes a link to the documentation for all three approaches.

Check your knowledge

1. What information does an `Action` provide in a role definition?

- ○ An `Action` provides the allowed *management* capabilities for the role.
- ○ An `Action` determines what data the role can manipulate.
- ○ An `Action` decides what resource the role is applied to.

2. Which of the following sets the *scope* of a role to be the resource group `myResourceGroup`?

- ○ `/subscriptions/de324015-0284-4582-9d9c-6f1e52a30471`
- ○ `/subscriptions/{ef67bd4f-d0f2-4845-b6dd-6cba225b4f10}/resourceGroups/myResourceGroup/backupvm1`
- ○ `/subscriptions/{ef67bd4f-d0f2-4845-b6dd-6cba225b4f10}/resourceGroups/myResourceGroup`

3. How are `NotActions` used in a role definition?

- ○ `NotActions` are subtracted from the `Actions` to define the list of permissible operations.
- ○ `NotActions` are consulted after `Actions` to deny access to a specific operation.
- ○ `NotActions` allow you to specify a single operation that is not allowed.

Check your answers