Set up Principals, Roles and Permissions, for Tachyon version 8.0 and later

Introduction

This section contains an overview of Tachyon role based access control (RBAC) and its components and how to configure Tachyon RBAC. This is not intended as an in-depth explanation of what RBAC is, but as a demonstration of how it's been implemented in Tachyon and how users can configure it programmatically.

RBAC implementation in Tachyon has had a major rework in version 8.0. This page will highlight differences between 'old' and 'new' implementations.

The C# examples assume you're using Tachyon Consumer SDK and you have an instantiated an instance of Tachyon connector class in an object called connector . All SDK methods return the same class called ApiCallResponse. Inside the object of ApiCallResponse you'll find a property called ReceivedObject. That object is the actual data received from the API.

In the following examples this detail is left out, stating that the returned object contains the data . For example, when we say that XYZ object contains certain data, this means the ReceivedObject property inside said object contains that data .

On this page:

Basics of role based access control (RBAC)

Role-based access control is an access control mechanism defined around roles and privileges. This security model pivots around the concept or a Role. Users (called principals in Tachyon) can be assigned to a Role and it's through a Role they gain permissions to perform actions. Each element of Tachyon's security system leads back to a Role.

In Tachyon 8.0, a third element has been added - a Management Group. How it comes into play you can read further down.

RBAC objects in Tachyon

Tachyon has several types of objects that are part of its RBAC system. This section serves as a brief description of these objects and their purpose.

Principals

Principals are identical to users. They're not called users because the word 'user' is traditionally associated with a person. A principal is, from a technical standpoint, an Active Directory account which may be a user account or a computer account. It can also be an AD group.

Tachyon allows access only to principals authenticated through Windows authentication and known to Tachyon itself. Depending on the request, Tachyon establishes the permissions of the calling principal by looking at the roles that principal is assigned to. By default, a fresh installation of Tachyon has a principal representing the user account installing it and another principal for the "NT AUTHORITY\Network Service", which many of Tachyon's services will be running as.

Roles

A Role is a container for Permissions and all permissions are linked to roles.

Principals must have roles assigned to them before using the system.

Permissions

A permission is an ability to perform an operation on a securable type.

Securable Types

A Securable Type represents a type of an object that can have permissions assigned to it. For example, Instructions can have permissions, as can Consumers and Management Groups. Security itself is an object principals need permissions to, so they can modify it, it will also have a Securable Type. If an element of Tachyon has security defined for it, it must have a securable type.

Instances

An Instance is one, specifc copy of an object of a given Securable Type. This can, for instance, be a single Instruction Set.

While most permissions work on Securable Types as a general thing, where if a Principal is assigned a Permission on a given Securable Type they can work with any object of that type, some Securable Types allow Permissions to be specified on just one given instance of a Securable Type object, giving a more granular control over access to that Securable Type.

At the moment only InstructionSet securable type supports instances.

Operations

An Operation (also called Applicable Operation) represents the type of an action that can be performed on a securable type, like "Read" or "Write".

Management Group

A Management Group is a container for Devices. A Management Group can either define rules, which are then used to decide which Devices belongs to a given group, or can explicitly list Devices that should belong to the Management Group.

While Management Groups are discussed in this article, they are not a focus of it and the nature of their creation and evaluation is covered elsewhere.

A Management Group and contain other Management Groups, with child Management Groups inheriting all rules of the parent and adding new ones, narrowing the device selection further with each level of nesting.

Permissions are inherited throughout the Management Group structure. This means that for RBAC purposes having Permissions on a Management Group means a automatically having the same Permissions on any and all children of that Management Groups.

Example how to read permissions

Marc (**Principal**) through his All Instructions Questioners (**Role**) has Questioner (**Applicable Operation**) permission on Instructions (**Securable type**) assigned on USA-Engineering (**Management Group**).

A Principal will have a specific permission on given securable type through a role the principal belongs to. This is the only way principals can obtain permissions and permissions are always for a specific operation on a given type.

To establish the full permission set of a given principal you have to combine all permissions from all the roles assigned to that principal.

Changes introduced in Tachyon version 8.0

There are two main changes introduced in version 8.0. The first is the very mechanism by which a User is assigned to a Role, and the second is the concept of delegation. There are, however, other, auxiliary concepts and changes that we will cover in this section.

Principals, Roles and Management Groups

In previous versions of Tachyon a Principal was assigned to a Role.

Furthermore, a Role could be linked to a Management Group, providing said Role had an Instruction Set related permission(s).

That way, a Role could be linked to given Management Group(s), and that link would be passed on to every Principal assigned to the Role. That way, a Role could be created with access to a specific Management Group(s) before any Principals have been assigned to the Role.

In version 8.0, Management Groups cannot be linked directly to Roles. Instead, RBAC assignments follow a "Who, What, Where" pattern - "Who" can perform "What" action(s) "Where".

"Who" is a Principal, so a User or a Group that performs various actions in the system.

"What" is the set of Permissions assigned to a Role. We will be talking about a Role, as it is little more than a container for Permissions, but we should remember that it is the Permissions assigned to a Role that defined what the Role's assignee can do.

"Where" is the collection of Devices on which the Principal can excercise their Permissions. These collections are Management Groups.

This means that from version 8.0 onwards, Principals are assigned to a Role on a specific Management Group, so given Principal will be able to excercise Pemrissions of the Role they've been assigned on a given Management Group and not elsewhere.

As an example, when a Principal wished to excercise their "Actioner" Permission they have via 'All Instructions Actioner' Role, they will only be able to do so on a specific Management Group the is defined in the assignment.

Global and Local Permissions

Not all Securable Types are connected to Management Groups. Many types are used to secure parts of the system not connected in any way with Devices, hence they are ignorant of Management Groups.

These Securable Types are called 'global' securable types and are easily identifiable as they have 'IsGlobal' flag set to true.

Because global Securable Types have no link to Devices or Management Groups, it is irrelevant what Management Group is assigned to Role that uses global securable types in its Permissions.

A Securable Type that has the 'IsGlobal' flag set to false is called a Local type and Permissions that use that Securable Type are called Local or Localized Permissions.

This means that the Management Group part of the assignment will define the boundaries of where the Permission can be excercised.

Role types

Because some Permissions are localized and some are global, depending on which Securable Type they use, we now have 3 types of Roles: Global, Local and Hybrid.

- Global Roles - these roles contain only permissions that use SecurableTypes which have 'IsGlobal' flag set tot true.
- Local Roles - these roles contain only permissions that use Securable Types which have 'IsGlobal' flag set to false.
- Hybrid Roles - these roles contain a mixture of permissions that use both global and non-global Securable Types.

A Local Role can be maked as delegatable. Global and Hybrid roles are not delegatable.

Non-delegatable Roles (be that Global, Hybrid or Local) can only be assigned to "All Devices", with the exception of "Group Administrator" Role, which cannot be assigned to All Devices despite being a hybrid Role.

Delegatable Roles, except for "Group Administrator", can be assigned to any Management Group, including "All Devices", subject to caller's Security Permissions.

You also have to remember that only Local Roles can be made delegatable. You can read about delegation here.

Anatomy of an RBAC assignment

An Assignment is a three-way "who, what, where" relationship between a Principal, Role and a Management Group.

A single Assignment is between a single Principal, a single Role and a single Management Group.

If you wish for a Principal to be able to excercise given Role's Permissions on two unrelated (see here) Management Groups, you will need to create two assignments.

You can create any number of assignments as long as they are all for unique combination of Principal, Role and Management Group.

Delegation

Delegation is, in broadest terms, a concept where an administrator would delegate some of their own responsibility to other users while limiting the area where they can excercise that responsibility.

While the concept itself is outside the bounds of this page, we will cover how delegation works internally in Tachyon and what implications it has to API users.

Types of security administrator

Before we delve into how the feature works, we have to establish two terms that we will use later on: global security administrator and local security administrator.

A global security administrator is a Principal that has a Role with a Permission that uses "Security" Securable Type and that Role is assigned to "All Devices" Management Group.

A local security administrator is a Principal that has a Role with a Permission that uses "Security" Securable Type and that Role is assigned on a Management Group other than "All Devices".

It is important to understand that global security administrator and local security administrator are not explicit Roles. Any Role with "Security" Permissions can make a Principal be considered global or local security administrator, depending on what Management Group said Role is assigned on.

One should also remember that the global and local security administrator aren't actually referenced in Tachyon and are not terms used by documentation in general. We define them here because they will be used on this page to descirbe how delegation works.

Who can assign what?

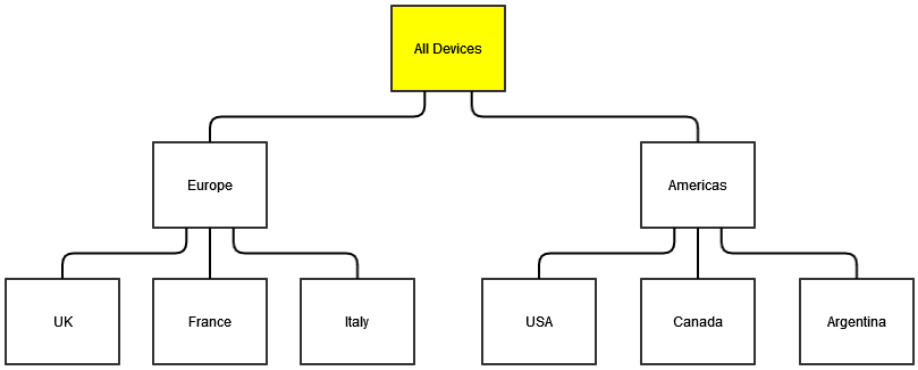
A global security administrator can create and modify Roles. They can also assign any Role in the system to any Principal using any Management Group they see fit (with some exceptions, like the fact that "Group Administrator" Role cannot be assigned to "All Devices" at all).

A local security administrator cannot create or modify Roles. They can assign any Role that is marked as delegatable ("CanBeDelegated" flag is set to true) to a Management Group they have the Security Permission on or any of its children. The exception here is that a local security administrator cannot assign a Role that has "Security" Permission using the same Management Group they themselves have "Security" Permission on. They have to use a child Management Group of the one they themselves have "Security" Permission on.

Effectively, a local security administrator cannot create an assignment that would result in another Principal having the same "Security" Permissions as they do. This limitation does not apply to global security administrator.

An example of how it works

Let's assume we have following Management Group structure:



We also have a Role called 'Security Administrator', which has Permission on Security (Read and Write), and another role called "Actioner" which has Actioner permission on an Instruction Set.

Now let us assume we have 3 Principals. First Principal, called 'John', has the 'Security Administrator' Role assigned on 'All Devices' Management Group.

The second Principal, called 'Jane', has, at the moment, no Roles assigned at all.

The third Principal, called 'Frank'', has no Roles assigned either.

At this point John, as per the rules outlined above, can create and modify Roles and assign any Role to any Principal using any Management Group.

John decides that he is going to delegate some of his responsibilities to Jane by assigning her to 'Security Administrator' on 'Europe' Management Group.

This means Jane is not able to create or edit Roles because, as per rules outlined above, she is a local security administrator. When manging assignments she can also only use "Europe", "UK", "France" and "Italy" Management Groups, because she has Security permission assigned on "Europe" and "UK", "France" and "Italy" are child Management Groups of "Europe". She is also limited to using only Roles that are marked as 'delegatable'.

Jane can assign "Actioner" role to Frank on "Europe" Management Group, because that Role does not have Security Permission.

Jane will not be able to assign "Security Administrator" Role to Frank using "Europe" Management Group, because she herself has Security Permission on that very Management Group and because that Role does have a Security Permission. She can, however, assign "Security Administrator" to Frank using either "UK", "France" or "Italy" Management Groups, because those are children Management Groups or the Management Group she has Security Permission on.

Retrieving RBAC objects

In this section we will look at endpoints that allow you to examine existing RBAC objects.

Getting Principals

The most basic functionality is to retrieve all Principals:

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Principals will yield following response:

Return payload


```
[
    {
        "Id": 1,
        "ExternalId": "S-1-5-21-1202660629-789336158-
1349024091-27850",
        "PrincipalName": "SomeDomain\\Administrator",
        "Email": "Administrator@SomeDomain.com",
        "Enabled": true,
        "CreatedTimestampUtc": "2019-11-07T13:14:
52.777Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:
52.777Z",
        "SystemPrincipal": true,
        "DisplayName": "Administrator",
        "IsGroup": false
    },
    {
        "Id": 2,
        "ExternalId": "S-1-5-20",
        "PrincipalName": "NT AUTHORITY\\Network
Service",
        "Email": null,
        "Enabled": true,
        "CreatedTimestampUtc": "2019-11-07T13:14:
56.687Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:
56.687Z",
        "SystemPrincipal": true,
        "DisplayName": "Network Service",
        "IsGroup": false
    },
    {
        "Id": 3,
        "ExternalId": "S-1-5-21-1202660629-789336058-
1343024091-23842",
        "PrincipalName": "SomeDomain\\Jane.Doe",
        "Email": "Jane.Doe@SomeDomain.com",
        "Enabled": true,
        "CreatedTimestampUtc": "2019-11-07T13:14:
52.777Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:
52.777Z",
        "SystemPrincipal": false,
        "DisplayName": "Jane Doe",
        "IsGroup": false
    },
    {
        "Id": 4,
        "ExternalId": "S-1-5-21-3276326578-728399001-
2836074973-1009",
        "PrincipalName": "SomeDomain\\John.Doe",
        "Email": "John.Doe@SomeDomain",
        "Enabled": true,
        "CreatedTimestampUtc": "2016-11-30T15:10:10.73
Z",
        "ModifiedTimestampUtc": "2016-11-30T15:10:
10.73Z",
        "SystemPrincipal": false,
        "DisplayName": "John Doe",
        "IsGroup": false
    }
]
```


Use Principals object inside the Tachyon connector instance.

Retrieving all principals

```
principals = onnector.Principals.GetAll();
```

"principals" object will contain the same data you can see in the JSON response on the left.

Or just a single Principal by their Id. Here we'll look for "SomeDomain\\Jane.Doe", who in has the Id of 3:

Direct Consumer API call

Making a GET request to https://my.tachyon.server/Consumer/Principals/3 will yield following response:

Return payload


```
{
    "Id": 3,
    "ExternalId": "S-1-5-21-1202660629-789336058-
1343024091-23842",
    "PrincipalName": "SomeDomain\\Jane.Doe",
    "Email": "Jane.Doe@SomeDomain.com",
    "Enabled": true,
    "CreatedTimestampUtc": "2019-11-07T13:14:52.777Z",
    "ModifiedTimestampUtc": "2019-11-07T13:14:52.777Z",
    "SystemPrincipal": false,
    "DisplayName": "Jane Doe",
    "IsGroup": false
}
```


C# code using Consumer SDK library

Use Principals object inside the Tachyon connector instance.

Retrieving specific Principal

```
principal = connector.Principals.Get(3);
```

"principal" object will contain the same data you can see in the JSON response on the left.

You can also retrieve all Principals that have a specific Role assigned to them. Here we'll get all Principals who are "Global Administrators", which has the Id of 1:

Direct Consumer API call

Making a GET request to https://my.tachyon.server/Consumer/Principals/Role/1 will yield following response:

Return payload


```
[
    {
        "PrincipalId": 1,
        "RoleId": 1,
        "CreatedTimestampUtc": "2019-11-07T13:15:01.533
Z",
        "Role": null,
        "Principal": {
            "Id": 1,
            "ExternalId": "S-1-5-21-1202660629-
789336158-1349024091-27850",
            "PrincipalName":
"SomeDomain\\Administrator",
            "Email": "Administrator@SomeDomain.com",
            "Enabled": true,
            "CreatedTimestampUtc": "2019-11-07T13:14:
52.777Z",
            "ModifiedTimestampUtc": "2019-11-07T13:14:
52.777Z",
            "SystemPrincipal": true,
            "DisplayName": "Administrator",
            "IsGroup": false
        }
    }
]
```


C# code using Consumer SDK library

Use Principals object inside the Tachyon connector instance.

Retrieving all principals that have a specific role

```
principals = connector.Principals.GetForRole
(1);
```

"principals" object will contain the same data you can see in the JSON response on the left.

Getting Roles

Much like with Principals, you can retrieve all Roles in the system:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Roles will yield following response	Use Roles object inside the Tachyon connector instance.

Return payload


```
[
    {
        "Id": 27,
        "Name": "1E ITSM Connect Actioner",
        "Description": "The ServiceNow proxy user is added to this role instead
of Global Actioners so that ServiceNow users can only use instructions belonging
to instruction sets assigned to this role",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.51Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
        "SystemRole": false,
        "CanBeDelegated": true,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 7,
        "Name": "All Instructions Actioner",
        "Description": "This role will allow assigned users to execute all
instructions that are defined as Actions.",
        "CreatedTimestampUtc": "2021-10-07T10:51:35.063Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
        "SystemRole": true,
        "CanBeDelegated": true,
        "NumberOfAssignments": 1,
        "HasSecurityPermission": false
    },
    {
        "Id": 5,
        "Name": "All Instructions Approver",
        "Description": "This role will allow assigned users to approve all
instructions that require approval.",
        "CreatedTimestampUtc": "2021-10-07T10:51:34.84Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
        "SystemRole": true,
        "CanBeDelegated": true,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 6,
        "Name": "All Instructions Questioner",
        "Description": "This role will allow assigned users to execute all
instructions that are defined as Questions.",
        "CreatedTimestampUtc": "2021-10-07T10:51:35.06Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
        "SystemRole": true,
        "CanBeDelegated": true,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 8,
        "Name": "All Instructions Viewer",
        "Description": "This role will allow assigned users to view responses to
all instructions.",
        "CreatedTimestampUtc": "2021-10-07T10:51:35.063Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
        "SystemRole": true,
        "CanBeDelegated": true,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
```


Retrieving all Roles

```
roles = connector.
Roles.GetAll();
```

"roles" object will contain the same data you can see in the JSON response on the left.

```
        "Id": 40,
        "Name": "Area Administrator",
        "Description": "The Area Administrator role allows users assigned to
assign users permission that have been defined as delegatable in Tachyon, this
means that they can be restricted to a Management Group.",
        "CreatedTimestampUtc": "2021-10-07T10:51:39.213Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:39.213Z",
        "SystemRole": true,
        "CanBeDelegated": true,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": true
    },
    {
        "Id": 32,
        "Name": "Engagement Administrator",
        "Description": "View, create, update, delete and enable Engagements",
        "CreatedTimestampUtc": "2021-10-07T10:51:38.01Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:38.01Z",
        "SystemRole": true,
        "CanBeDelegated": false,
        "NumberOfAssignments": 1,
        "HasSecurityPermission": false
    },
    {
        "Id": 41,
        "Name": "Experience Administrator",
        "Description": "The Experience Administrator role allows users assigned
to access all areas of Experience, including defining Engagement Surveys and
Announcements.",
        "CreatedTimestampUtc": "2021-10-07T10:51:39.22Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:39.22Z",
        "SystemRole": false,
        "CanBeDelegated": false,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 38,
        "Name": "Experience Engagement Assigner",
        "Description": "This role will allow assigned users the ability to
assign Engagements to any Management Group that is including in their
assignments.",
        "CreatedTimestampUtc": "2021-10-07T10:51:38.97Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:38.97Z",
        "SystemRole": false,
        "CanBeDelegated": true,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 30,
        "Name": "Experience User",
        "Description": "The Experience User role allows users assigned to view
access all areas of Experience.",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.833Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
        "SystemRole": false,
        "CanBeDelegated": false,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 1,
        "Name": "Full Administrator",
        "Description": "The Full Administrator has the combined permissions of
all other roles.",
        "CreatedTimestampUtc": "2021-10-07T10:51:34.837Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
        "SystemRole": true,
        "CanBeDelegated": false,
        "NumberOfAssignments": 1,
```

```
        "HasSecurityPermission": true
    },
    {
        "Id": 24,
        "Name": "Guaranteed State Administrator",
        "Description": "The Guaranteed State Administrator role allows users
assigned to access all areas of Guaranteed State including defining Rules,
Polices and uploading fragments.",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.463Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
        "SystemRole": true,
        "CanBeDelegated": false,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 37,
        "Name": "Guaranteed State Policy Assigner",
        "Description": "This role will allow assigned users the ability to
assign Policies to any Management Group that is including in their assignments.",
        "CreatedTimestampUtc": "2021-10-07T10:51:38.96Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:38.96Z",
        "SystemRole": true,
        "CanBeDelegated": true,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 25,
        "Name": "Guaranteed State User",
        "Description": "The Guaranteed State User role allows users assigned to
view access all areas of Guaranteed State.",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.467Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
        "SystemRole": true,
        "CanBeDelegated": false,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 39,
        "Name": "Installer",
        "Description": "The Installer role has minimum permissions to install
the Platform and Applications, register Consumers, upload Products packs and
manage Instructions sets.",
        "CreatedTimestampUtc": "2021-10-07T10:51:39.203Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:39.203Z",
        "SystemRole": true,
        "CanBeDelegated": false,
        "NumberOfAssignments": 1,
        "HasSecurityPermission": true
    },
    {
        "Id": 13,
        "Name": "Management Group Administrator",
        "Description": "Create, delete, update and initiate synchronization of
Management Groups",
        "CreatedTimestampUtc": "2021-10-07T10:51:36.98Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
        "SystemRole": true,
        "CanBeDelegated": false,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 44,
        "Name": "Nomad Administrators",
        "Description": null,
        "CreatedTimestampUtc": "2021-10-07T13:42:13.523Z",
        "ModifiedTimestampUtc": "2021-10-07T13:42:13.523Z",
        "SystemRole": false,
```

```
        "CanBeDelegated": true,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 42,
        "Name": "Patch Success Administrator",
        "Description": "The Patch Success Administrator role allows users
assigned to access all areas of Patch Success including deploying Patches.",
        "CreatedTimestampUtc": "2021-10-07T10:51:39.223Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:39.223Z",
        "SystemRole": false,
        "CanBeDelegated": false,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 26,
        "Name": "Patch Success User",
        "Description": "The Patch Success User role allows users assigned to
view access all areas of Patch Success and initiate check status and update
status.",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.49Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
        "SystemRole": false,
        "CanBeDelegated": false,
        "NumberOfAssignments": 0,
        "HasSecurityPermission": false
    },
    {
        "Id": 43,
        "Name": "Tachyon System",
        "Description": "The System role allows assigned users to performance
operations required to execute the different cross component activities needed
for Tachyon Platform to run normally.",
        "CreatedTimestampUtc": "2021-10-07T10:51:39.227Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:39.227Z",
        "SystemRole": true,
        "CanBeDelegated": false,
        "NumberOfAssignments": 1,
        "HasSecurityPermission": true
    }
]
```


But unlike the Principals, roles also have an endpoint that supports filtering, sorting and paging. In the example below we'll look for custom roles and sort them by name.

Direct Consumer API call	**C# code using Consumer SDK library**
Making a POST request to https://my.tachyon.server/Consumer/Roles/Search with following payload:	

Request payload

```
{
 "PageSize": 10,
 "Start": 1,
 "Filter": {
 "Attribute": "SystemRole",
 "Operator": "==",
 "Value": "false"
 },
 "Sort": [{
 "Direction": "ASC",
 "Column": "Name"
 }]
}
```

Yields following response:

Return payload


```
{
    "TotalCount": 7,
    "Items": [
        {
            "Id": 27,
            "Name": "1E ITSM Connect Actioner",
            "Description": "The ServiceNow proxy user is added to this role
instead of Global Actioners so that ServiceNow users can only use
instructions belonging to instruction sets assigned to this role",
            "CreatedTimestampUtc": "2021-10-07T10:51:37.51Z",
            "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
            "SystemRole": false,
            "CanBeDelegated": true,
            "NumberOfAssignments": 0,
            "HasSecurityPermission": false
        },
        {
            "Id": 41,
            "Name": "Experience Administrator",
            "Description": "The Experience Administrator role allows users
assigned to access all areas of Experience, including defining Engagement
Surveys and Announcements.",
            "CreatedTimestampUtc": "2021-10-07T10:51:39.22Z",
            "ModifiedTimestampUtc": "2021-10-07T10:51:39.22Z",
            "SystemRole": false,
            "CanBeDelegated": false,
            "NumberOfAssignments": 0,
            "HasSecurityPermission": false
        },
        {
            "Id": 38,
            "Name": "Experience Engagement Assigner",
            "Description": "This role will allow assigned users the ability
to assign Engagements to any Management Group that is including in their
assignments.",
            "CreatedTimestampUtc": "2021-10-07T10:51:38.97Z",
            "ModifiedTimestampUtc": "2021-10-07T10:51:38.97Z",
            "SystemRole": false,
            "CanBeDelegated": true,
            "NumberOfAssignments": 0,
            "HasSecurityPermission": false
        },
        {
            "Id": 30,
            "Name": "Experience User",
            "Description": "The Experience User role allows users assigned to
view access all areas of Experience.",
```


Use Roles object inside the Tachyon connector instance.

Searching for Roles

```
var payload = new Search
{
        PageSize = 10,
        Start = 1,
        Filter = new
ExpressionObject
        {

Attribute =
"SystemRole",

Operator = "==",
                Value =
"false"
        },
        Sort = new
List<SortSpec>() { new
SortSpec
        {

Direction = "ASC",
                Column
= "Name"
        }}
};

searchResults =
connector.Roles.
FindRoles(payload);
```

"searchResults" object will contain the same data you can see in the JSON response on the left.

```
            "CreatedTimestampUtc": "2021-10-07T10:51:37.833Z",
            "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
            "SystemRole": false,
            "CanBeDelegated": false,
            "NumberOfAssignments": 0,
            "HasSecurityPermission": false
        },
        {
            "Id": 44,
            "Name": "Nomad Administrators",
            "Description": null,
            "CreatedTimestampUtc": "2021-10-07T13:42:13.523Z",
            "ModifiedTimestampUtc": "2021-10-07T13:42:13.523Z",
            "SystemRole": false,
            "CanBeDelegated": true,
            "NumberOfAssignments": 0,
            "HasSecurityPermission": false
        },
        {
            "Id": 42,
            "Name": "Patch Success Administrator",
            "Description": "The Patch Success Administrator role allows users
assigned to access all areas of Patch Success including deploying Patches.",
            "CreatedTimestampUtc": "2021-10-07T10:51:39.223Z",
            "ModifiedTimestampUtc": "2021-10-07T10:51:39.223Z",
            "SystemRole": false,
            "CanBeDelegated": false,
            "NumberOfAssignments": 0,
            "HasSecurityPermission": false
        },
        {
            "Id": 26,
            "Name": "Patch Success User",
            "Description": "The Patch Success User role allows users assigned
to view access all areas of Patch Success and initiate check status and
update status.",
            "CreatedTimestampUtc": "2021-10-07T10:51:37.49Z",
            "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
            "SystemRole": false,
            "CanBeDelegated": false,
            "NumberOfAssignments": 0,
            "HasSecurityPermission": false
        }
    ]
}
```


You can also retrieve a singel Role by its Id, here we'll retrieve the Global Administrators role which has the Id of 1.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Roles/1 will yield following response:

Return payload


```
{
    "Id": 1,
    "Name": "Full Administrator",
    "Description": "The Full Administrator has
permissions in the system.",
    "CreatedTimestampUtc": "2021-10-07T10:51:34.837Z",
    "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
    "SystemRole": true,
    "CanBeDelegated": false,
    "NumberOfAssignments": 1,
    "HasSecurityPermission": true
}
```


Use Roles object inside the Tachyon connector instance.

Retrieving a specifc Role by its Id

```
roles = connector.Roles.Get(1);
```

"role" object will contain the same data you can see in the JSON response on the left.

Lastly, we'll look at retrieving all Roles given Principal has assigned to them using that Principal's Id. Here we'll get all the Roles that "SomeDomain\Jane.Doe" has assigned and that Principal's Id is 3:

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Roles/Principal/3 will yield following response:

Return payload


```
[
    {
        "PrincipalId": 3,
        "RoleId": 7,
        "CreatedTimestampUtc": "2021-10-08T13:39:01.5694234Z",
        "Role": {
            "Id": 7,
            "Name": "All Instructions Actioner",
            "Description": "This role will allow assigned users to
execute all instructions that are defined as Actions.",
            "CreatedTimestampUtc": "2021-10-07T10:51:35.063Z",
            "ModifiedTimestampUtc": "2021-10-07T10:51:37.893Z",
            "SystemRole": true,
            "CanBeDelegated": false,
            "NumberOfAssignments": 1,
            "HasSecurityPermission": false
        },
        "Principal": null
    }
]
```


Use Roles object inside the Tachyon connector instance.

Retrieving all Roles fora given Principal

```
roles = connector.Roles.
GetForPrincipal(principalId);
```

"roles" object will contain the same data you can see in the JSON response on the left.

Getting Securable Types

You can retrieve all Securable Types with following call:

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/SecurableTypes will yield following response:

Return payload


```
[
    {
        "Id": 1,
```

```
        "Name": "InstructionSet",
        "CreatedTimestampUtc": "2021-10-07T10:51:34.83Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:36.647Z",
        "Operations": [
            {
                "Id": 2,
                "OperationName": "Actioner",
                "SecurableTypeId": 1,
                "SecurableTypeName": "InstructionSet"
            },
            {
                "Id": 4,
                "OperationName": "Approver",
                "SecurableTypeId": 1,
                "SecurableTypeName": "InstructionSet"
            },
            {
                "Id": 3,
                "OperationName": "Questioner",
                "SecurableTypeId": 1,
                "SecurableTypeName": "InstructionSet"
            },
            {
                "Id": 1,
                "OperationName": "Viewer",
                "SecurableTypeId": 1,
                "SecurableTypeName": "InstructionSet"
            }
        ],
        "IsGlobal": false,
        "Description": "Execute, schedule, cancel, and approve
instructions - view responses"
    },
    {
        "Id": 2,
        "Name": "Security",
        "CreatedTimestampUtc": "2021-10-07T10:51:34.83Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:34.83Z",
        "Operations": [
            {
                "Id": 7,
                "OperationName": "Delete",
                "SecurableTypeId": 2,
                "SecurableTypeName": "Security"
            },
            {
                "Id": 5,
                "OperationName": "Read",
                "SecurableTypeId": 2,
                "SecurableTypeName": "Security"
            },
            {
                "Id": 6,
                "OperationName": "Write",
                "SecurableTypeId": 2,
                "SecurableTypeName": "Security"
            }
        ],
        "IsGlobal": false,
        "Description": "Add and remove Users - view all Roles - add,
modify, and delete Custom roles - assign roles to users - view Audit
information log"
    },
    {
        "Id": 3,
        "Name": "InstructionSetManagement",
        "CreatedTimestampUtc": "2021-10-07T10:51:34.83Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:36.647Z",
        "Operations": [
            {
                "Id": 9,
```


Use SecurableTypes object inside the Tachyon connector instance.

Retrieving all Securable Types

```
secTypes = connector.
SecurableTypes.GetAll();
```

"secTypes" object will contain the same data you can see in the JSON response on the left.

```
                "OperationName": "Add",
                "SecurableTypeId": 3,
                "SecurableTypeName": "InstructionSetManagement"
            },
            {
                "Id": 8,
                "OperationName": "Delete",
                "SecurableTypeId": 3,
                "SecurableTypeName": "InstructionSetManagement"
            },
            {
                "Id": 10,
                "OperationName": "Read",
                "SecurableTypeId": 3,
                "SecurableTypeName": "InstructionSetManagement"
            }
        ],
        "IsGlobal": true,
        "Description": "Upload product packs - add, modify, and delete
instruction sets - delete instruction definitions"
    },
    {
        "Id": 4,
        "Name": "Infrastructure",
        "CreatedTimestampUtc": "2021-10-07T10:51:34.83Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:34.83Z",
        "Operations": [
            {
                "Id": 85,
                "OperationName": "Delete",
                "SecurableTypeId": 4,
                "SecurableTypeName": "Infrastructure"
            },
            {
                "Id": 11,
                "OperationName": "Read",
                "SecurableTypeId": 4,
                "SecurableTypeName": "Infrastructure"
            },
            {
                "Id": 84,
                "OperationName": "Write",
                "SecurableTypeId": 4,
                "SecurableTypeName": "Infrastructure"
            }
        ],
        "IsGlobal": true,
        "Description": "View System health and System information -
view, add, and edit global settings"
    },
    {
        "Id": 5,
        "Name": "CustomProperty",
        "CreatedTimestampUtc": "2021-10-07T10:51:35.33Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:35.33Z",
        "Operations": [
            {
                "Id": 12,
                "OperationName": "Read",
                "SecurableTypeId": 5,
                "SecurableTypeName": "CustomProperty"
            },
            {
                "Id": 13,
                "OperationName": "Write",
                "SecurableTypeId": 5,
                "SecurableTypeName": "CustomProperty"
            }
        ],
        "IsGlobal": true,
        "Description": "View, add, edit, and delete Custom properties"
```

```
    },
    {
        "Id": 6,
        "Name": "Consumer",
        "CreatedTimestampUtc": "2021-10-07T10:51:35.877Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:35.877Z",
        "Operations": [
            {
                "Id": 14,
                "OperationName": "Read",
                "SecurableTypeId": 6,
                "SecurableTypeName": "Consumer"
            },
            {
                "Id": 15,
                "OperationName": "Write",
                "SecurableTypeId": 6,
                "SecurableTypeName": "Consumer"
            }
        ],
        "IsGlobal": true,
        "Description": "View, add, edit, and delete Consumers"
    },
    {
        "Id": 7,
        "Name": "AgentDeployment",
        "CreatedTimestampUtc": "2021-10-07T10:51:36.84Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:36.84Z",
        "Operations": [
            {
                "Id": 18,
                "OperationName": "Approve",
                "SecurableTypeId": 7,
                "SecurableTypeName": "AgentDeployment"
            },
            {
                "Id": 16,
                "OperationName": "Execute",
                "SecurableTypeId": 7,
                "SecurableTypeName": "AgentDeployment"
            },
            {
                "Id": 17,
                "OperationName": "View",
                "SecurableTypeId": 7,
                "SecurableTypeName": "AgentDeployment"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, and cancel 1E Client deployment
jobs"
    },
    {
        "Id": 8,
        "Name": "AgentInstallerManagement",
        "CreatedTimestampUtc": "2021-10-07T10:51:36.88Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:36.88Z",
        "Operations": [
            {
                "Id": 19,
                "OperationName": "Add",
                "SecurableTypeId": 8,
                "SecurableTypeName": "AgentInstallerManagement"
            },
            {
                "Id": 20,
                "OperationName": "Delete",
                "SecurableTypeId": 8,
                "SecurableTypeName": "AgentInstallerManagement"
            },
            {
```

```
                "Id": 21,
                "OperationName": "Read",
                "SecurableTypeId": 8,
                "SecurableTypeName": "AgentInstallerManagement"
            }
        ],
        "IsGlobal": true,
        "Description": "View, upload, and delete 1E Client installers"
    },
    {
        "Id": 9,
        "Name": "ManagementGroup",
        "CreatedTimestampUtc": "2021-10-07T10:51:36.98Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:36.98Z",
        "Operations": [
            {
                "Id": 29,
                "OperationName": "Delete",
                "SecurableTypeId": 9,
                "SecurableTypeName": "ManagementGroup"
            },
            {
                "Id": 22,
                "OperationName": "Read",
                "SecurableTypeId": 9,
                "SecurableTypeName": "ManagementGroup"
            },
            {
                "Id": 24,
                "OperationName": "Synchronize",
                "SecurableTypeId": 9,
                "SecurableTypeName": "ManagementGroup"
            },
            {
                "Id": 23,
                "OperationName": "Write",
                "SecurableTypeId": 9,
                "SecurableTypeName": "ManagementGroup"
            }
        ],
        "IsGlobal": false,
        "Description": "Create, delete, edit, and initiate
synchronization of Management Groups"
    },
    {
        "Id": 10,
        "Name": "Connector",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.38Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.38Z",
        "Operations": [
            {
                "Id": 27,
                "OperationName": "Delete",
                "SecurableTypeId": 10,
                "SecurableTypeName": "Connector"
            },
            {
                "Id": 28,
                "OperationName": "Execute",
                "SecurableTypeId": 10,
                "SecurableTypeName": "Connector"
            },
            {
                "Id": 25,
                "OperationName": "Read",
                "SecurableTypeId": 10,
                "SecurableTypeName": "Connector"
            },
            {
                "Id": 26,
                "OperationName": "Write",
```

```
                "SecurableTypeId": 10,
                "SecurableTypeName": "Connector"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, delete, and test Connectors"
    },
    {
        "Id": 11,
        "Name": "Schedule",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.38Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.38Z",
        "Operations": [
            {
                "Id": 32,
                "OperationName": "Delete",
                "SecurableTypeId": 11,
                "SecurableTypeName": "Schedule"
            },
            {
                "Id": 30,
                "OperationName": "Read",
                "SecurableTypeId": 11,
                "SecurableTypeName": "Schedule"
            },
            {
                "Id": 31,
                "OperationName": "Write",
                "SecurableTypeId": 11,
                "SecurableTypeName": "Schedule"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, and delete Schedules - view
Schedule history"
    },
    {
        "Id": 12,
        "Name": "ProcessLog",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.38Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.38Z",
        "Operations": [
            {
                "Id": 90,
                "OperationName": "Delete",
                "SecurableTypeId": 12,
                "SecurableTypeName": "ProcessLog"
            },
            {
                "Id": 33,
                "OperationName": "Read",
                "SecurableTypeId": 12,
                "SecurableTypeName": "ProcessLog"
            },
            {
                "Id": 89,
                "OperationName": "Write",
                "SecurableTypeId": 12,
                "SecurableTypeName": "ProcessLog"
            }
        ],
        "IsGlobal": true,
        "Description": "View and purge the Process log, Cancel all
actions"
    },
    {
        "Id": 13,
        "Name": "SynchronizationLog",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.38Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.38Z",
        "Operations": [
```

```
            {
                "Id": 34,
                "OperationName": "Read",
                "SecurableTypeId": 13,
                "SecurableTypeName": "SynchronizationLog"
            }
        ],
        "IsGlobal": true,
        "Description": "View Sync log"
    },
    {
        "Id": 14,
        "Name": "Component",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.38Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.38Z",
        "Operations": [
            {
                "Id": 35,
                "OperationName": "Read",
                "SecurableTypeId": 14,
                "SecurableTypeName": "Component"
            },
            {
                "Id": 86,
                "OperationName": "Write",
                "SecurableTypeId": 14,
                "SecurableTypeName": "Component"
            }
        ],
        "IsGlobal": true,
        "Description": "View and configure Components"
    },
    {
        "Id": 15,
        "Name": "ProviderConfiguration",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.38Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.38Z",
        "Operations": [
            {
                "Id": 38,
                "OperationName": "Delete",
                "SecurableTypeId": 15,
                "SecurableTypeName": "ProviderConfiguration"
            },
            {
                "Id": 36,
                "OperationName": "Read",
                "SecurableTypeId": 15,
                "SecurableTypeName": "ProviderConfiguration"
            },
            {
                "Id": 37,
                "OperationName": "Write",
                "SecurableTypeId": 15,
                "SecurableTypeName": "ProviderConfiguration"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, and delete Providers"
    },
    {
        "Id": 16,
        "Name": "InfrastructureLog",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.38Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.38Z",
        "Operations": [
            {
                "Id": 39,
                "OperationName": "Read",
                "SecurableTypeId": 16,
                "SecurableTypeName": "InfrastructureLog"
```

```
            }
        ],
        "IsGlobal": true,
        "Description": "View Infrastructure log"
    },
    {
        "Id": 17,
        "Name": "Repository.Inventory",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.38Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.38Z",
        "Operations": [
            {
                "Id": 43,
                "OperationName": "Archive",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repository.Inventory"
            },
            {
                "Id": 42,
                "OperationName": "Delete",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repository.Inventory"
            },
            {
                "Id": 53,
                "OperationName": "EvaluateManagementGroups",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repository.Inventory"
            },
            {
                "Id": 44,
                "OperationName": "Populate",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repository.Inventory"
            },
            {
                "Id": 40,
                "OperationName": "Read",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repository.Inventory"
            },
            {
                "Id": 41,
                "OperationName": "Write",
                "SecurableTypeId": 17,
                "SecurableTypeName": "Repository.Inventory"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, and delete Inventory
repositories - populate and archive them"
    },
    {
        "Id": 18,
        "Name": "Application",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.433Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.433Z",
        "Operations": [
            {
                "Id": 46,
                "OperationName": "Delete",
                "SecurableTypeId": 18,
                "SecurableTypeName": "Application"
            },
            {
                "Id": 45,
                "OperationName": "Write",
                "SecurableTypeId": 18,
                "SecurableTypeName": "Application"
            }
        ],
```

```
        "IsGlobal": true,
        "Description": "Install and uninstall Tachyon Portal
applications"
    },
    {
        "Id": 19,
        "Name": "GuaranteedState",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.463Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.463Z",
        "Operations": [
            {
                "Id": 49,
                "OperationName": "Delete",
                "SecurableTypeId": 19,
                "SecurableTypeName": "GuaranteedState"
            },
            {
                "Id": 47,
                "OperationName": "Read",
                "SecurableTypeId": 19,
                "SecurableTypeName": "GuaranteedState"
            },
            {
                "Id": 48,
                "OperationName": "Write",
                "SecurableTypeId": 19,
                "SecurableTypeName": "GuaranteedState"
            }
        ],
        "IsGlobal": true,
        "Description": "View, add, edit, and delete Rules, Fragments,
Trigger templates, and Policies - view Guaranteed State dashboards"
    },
    {
        "Id": 20,
        "Name": "Repository.Patch",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.49Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.49Z",
        "Operations": [
            {
                "Id": 50,
                "OperationName": "Read",
                "SecurableTypeId": 20,
                "SecurableTypeName": "Repository.Patch"
            }
        ],
        "IsGlobal": true,
        "Description": "View Patch Success dashboards"
    },
    {
        "Id": 21,
        "Name": "Repository.BI",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.5Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.5Z",
        "Operations": [
            {
                "Id": 52,
                "OperationName": "Populate",
                "SecurableTypeId": 21,
                "SecurableTypeName": "Repository.BI"
            },
            {
                "Id": 51,
                "OperationName": "Read",
                "SecurableTypeId": 21,
                "SecurableTypeName": "Repository.BI"
            }
        ],
        "IsGlobal": true,
        "Description": "View and populate the BI respository"
    },
```

```
    {
        "Id": 22,
        "Name": "EventSubscription",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.74Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.74Z",
        "Operations": [
            {
                "Id": 56,
                "OperationName": "Delete",
                "SecurableTypeId": 22,
                "SecurableTypeName": "EventSubscription"
            },
            {
                "Id": 54,
                "OperationName": "Read",
                "SecurableTypeId": 22,
                "SecurableTypeName": "EventSubscription"
            },
            {
                "Id": 55,
                "OperationName": "Write",
                "SecurableTypeId": 22,
                "SecurableTypeName": "EventSubscription"
            }
        ],
        "IsGlobal": false,
        "Description": "View, create, edit, and delete the
configurations of event subscriptions"
    },
    {
        "Id": 24,
        "Name": "Experience",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.833Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.833Z",
        "Operations": [
            {
                "Id": 58,
                "OperationName": "Read",
                "SecurableTypeId": 24,
                "SecurableTypeName": "Experience"
            }
        ],
        "IsGlobal": true,
        "Description": "View Experience dashboards"
    },
    {
        "Id": 25,
        "Name": "Inventory",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.85Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.85Z",
        "Operations": [
            {
                "Id": 60,
                "OperationName": "Export",
                "SecurableTypeId": 25,
                "SecurableTypeName": "Inventory"
            },
            {
                "Id": 59,
                "OperationName": "Read",
                "SecurableTypeId": 25,
                "SecurableTypeName": "Inventory"
            }
        ],
        "IsGlobal": true,
        "Description": "View Inventory dashboards and export inventory
data"
    },
    {
        "Id": 26,
        "Name": "Inventory.Association",
```

```
        "CreatedTimestampUtc": "2021-10-07T10:51:37.85Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.85Z",
        "Operations": [
            {
                "Id": 64,
                "OperationName": "Delete",
                "SecurableTypeId": 26,
                "SecurableTypeName": "Inventory.Association"
            },
            {
                "Id": 63,
                "OperationName": "Export",
                "SecurableTypeId": 26,
                "SecurableTypeName": "Inventory.Association"
            },
            {
                "Id": 61,
                "OperationName": "Read",
                "SecurableTypeId": 26,
                "SecurableTypeName": "Inventory.Association"
            },
            {
                "Id": 62,
                "OperationName": "Write",
                "SecurableTypeId": 26,
                "SecurableTypeName": "Inventory.Association"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, and delete SCCM Associations
in Inventory"
    },
    {
        "Id": 27,
        "Name": "AppClarity.Entitlement",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.877Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.877Z",
        "Operations": [
            {
                "Id": 68,
                "OperationName": "Delete",
                "SecurableTypeId": 27,
                "SecurableTypeName": "AppClarity.Entitlement"
            },
            {
                "Id": 69,
                "OperationName": "Execute",
                "SecurableTypeId": 27,
                "SecurableTypeName": "AppClarity.Entitlement"
            },
            {
                "Id": 67,
                "OperationName": "Export",
                "SecurableTypeId": 27,
                "SecurableTypeName": "AppClarity.Entitlement"
            },
            {
                "Id": 65,
                "OperationName": "Read",
                "SecurableTypeId": 27,
                "SecurableTypeName": "AppClarity.Entitlement"
            },
            {
                "Id": 66,
                "OperationName": "Write",
                "SecurableTypeId": 27,
                "SecurableTypeName": "AppClarity.Entitlement"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, delete, export, and manage
```

```
AppClarity Entitlement"
    },
    {
        "Id": 28,
        "Name": "AppClarity.Compliance",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.877Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.877Z",
        "Operations": [
            {
                "Id": 73,
                "OperationName": "Delete",
                "SecurableTypeId": 28,
                "SecurableTypeName": "AppClarity.Compliance"
            },
            {
                "Id": 74,
                "OperationName": "Execute",
                "SecurableTypeId": 28,
                "SecurableTypeName": "AppClarity.Compliance"
            },
            {
                "Id": 72,
                "OperationName": "Export",
                "SecurableTypeId": 28,
                "SecurableTypeName": "AppClarity.Compliance"
            },
            {
                "Id": 70,
                "OperationName": "Read",
                "SecurableTypeId": 28,
                "SecurableTypeName": "AppClarity.Compliance"
            },
            {
                "Id": 71,
                "OperationName": "Write",
                "SecurableTypeId": 28,
                "SecurableTypeName": "AppClarity.Compliance"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, delete, export, and manage
AppClarity Compliance and LDC"
    },
    {
        "Id": 29,
        "Name": "AppClarity.Reclaim",
        "CreatedTimestampUtc": "2021-10-07T10:51:37.877Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:37.877Z",
        "Operations": [
            {
                "Id": 78,
                "OperationName": "Delete",
                "SecurableTypeId": 29,
                "SecurableTypeName": "AppClarity.Reclaim"
            },
            {
                "Id": 79,
                "OperationName": "Execute",
                "SecurableTypeId": 29,
                "SecurableTypeName": "AppClarity.Reclaim"
            },
            {
                "Id": 77,
                "OperationName": "Export",
                "SecurableTypeId": 29,
                "SecurableTypeName": "AppClarity.Reclaim"
            },
            {
                "Id": 75,
                "OperationName": "Read",
                "SecurableTypeId": 29,
```

```
                "SecurableTypeName": "AppClarity.Reclaim"
            },
            {
                "Id": 76,
                "OperationName": "Write",
                "SecurableTypeId": 29,
                "SecurableTypeName": "AppClarity.Reclaim"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, delete, export, and manage
AppClarity Reclaim"
    },
    {
        "Id": 30,
        "Name": "Engagements",
        "CreatedTimestampUtc": "2021-10-07T10:51:38.01Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:38.01Z",
        "Operations": [
            {
                "Id": 82,
                "OperationName": "Delete",
                "SecurableTypeId": 30,
                "SecurableTypeName": "Engagements"
            },
            {
                "Id": 83,
                "OperationName": "Execute",
                "SecurableTypeId": 30,
                "SecurableTypeName": "Engagements"
            },
            {
                "Id": 80,
                "OperationName": "Read",
                "SecurableTypeId": 30,
                "SecurableTypeName": "Engagements"
            },
            {
                "Id": 81,
                "OperationName": "Write",
                "SecurableTypeId": 30,
                "SecurableTypeName": "Engagements"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, delete, and enable
Engagements (Surveys and Announcements)"
    },
    {
        "Id": 31,
        "Name": "VDI",
        "CreatedTimestampUtc": "2021-10-07T10:51:38.1Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:38.1Z",
        "Operations": [
            {
                "Id": 87,
                "OperationName": "Read",
                "SecurableTypeId": 31,
                "SecurableTypeName": "VDI"
            },
            {
                "Id": 88,
                "OperationName": "Write",
                "SecurableTypeId": 31,
                "SecurableTypeName": "VDI"
            }
        ],
        "IsGlobal": true,
        "Description": "View, create, edit, and delete application
servers"
    },
```

```
    {
        "Id": 32,
        "Name": "ProviderOperationLog",
        "CreatedTimestampUtc": "2021-10-07T10:51:38.13Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:38.13Z",
        "Operations": [
            {
                "Id": 91,
                "OperationName": "Read",
                "SecurableTypeId": 32,
                "SecurableTypeName": "ProviderOperationLog"
            }
        ],
        "IsGlobal": true,
        "Description": "Update, delete and view provider configurations"
    },
    {
        "Id": 33,
        "Name": "PolicyDeployment",
        "CreatedTimestampUtc": "2021-10-07T10:51:38.15Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:38.15Z",
        "Operations": [
            {
                "Id": 92,
                "OperationName": "Execute",
                "SecurableTypeId": 33,
                "SecurableTypeName": "PolicyDeployment"
            }
        ],
        "IsGlobal": true,
        "Description": "Deploy all types of policies (including metrics,
events, and engagements)"
    },
    {
        "Id": 34,
        "Name": "OffloadingData",
        "CreatedTimestampUtc": "2021-10-07T10:51:38.25Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:38.25Z",
        "Operations": [
            {
                "Id": 93,
                "OperationName": "Offload",
                "SecurableTypeId": 34,
                "SecurableTypeName": "OffloadingData"
            }
        ],
        "IsGlobal": true,
        "Description": "Offload (forward) event data to any Web API
responsible for processing that data"
    },
    {
        "Id": 35,
        "Name": "Nomad",
        "CreatedTimestampUtc": "2021-10-07T10:51:38.287Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:38.287Z",
        "Operations": [
            {
                "Id": 96,
                "OperationName": "Delete",
                "SecurableTypeId": 35,
                "SecurableTypeName": "Nomad"
            },
            {
                "Id": 94,
                "OperationName": "Read",
                "SecurableTypeId": 35,
                "SecurableTypeName": "Nomad"
            },
            {
                "Id": 95,
                "OperationName": "Write",
```

```
                "SecurableTypeId": 35,
                "SecurableTypeName": "Nomad"
            }
        ],
        "IsGlobal": true,
        "Description": "View Nomad dashboards and SSD peer data. View,
add, and delete pre-cache jobs. Pause and resume download activity of
Nomad clients"
    },
    {
        "Id": 36,
        "Name": "Protect",
        "CreatedTimestampUtc": "2021-10-07T10:51:38.317Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:38.317Z",
        "Operations": [
            {
                "Id": 97,
                "OperationName": "Read",
                "SecurableTypeId": 36,
                "SecurableTypeName": "Protect"
            },
            {
                "Id": 98,
                "OperationName": "Write",
                "SecurableTypeId": 36,
                "SecurableTypeName": "Protect"
            }
        ],
        "IsGlobal": true,
        "Description": "View and deploy patches at all endpoints"
    },
    {
        "Id": 37,
        "Name": "PolicyAssignment",
        "CreatedTimestampUtc": "2021-10-07T10:51:38.603Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:38.603Z",
        "Operations": [
            {
                "Id": 99,
                "OperationName": "Assign",
                "SecurableTypeId": 37,
                "SecurableTypeName": "PolicyAssignment"
            }
        ],
        "IsGlobal": false,
        "Description": "Assign Guaranteed State policies to Management
Groups"
    },
    {
        "Id": 38,
        "Name": "EngagementAssignment",
        "CreatedTimestampUtc": "2021-10-07T10:51:38.957Z",
        "ModifiedTimestampUtc": "2021-10-07T10:51:38.957Z",
        "Operations": [
            {
                "Id": 100,
                "OperationName": "Assign",
                "SecurableTypeId": 38,
                "SecurableTypeName": "EngagementAssignment"
            }
        ],
        "IsGlobal": false,
        "Description": "Assign Engagements (Surveys and Announcements)
to Management Groups"
    }
]
```


You can also retrieve a specific Securable Type by either its Name or Id.

Direct Consumer API call	C# code using Consumer SDK library

If you wish to use the Id, make a GET request to https://my.tachyon.server/Consumer/SecurableTypes/1
 	
If you wish to use the Name instead, make a GET request to https://my.tachyon.server/Consumer/SecurableTypes/Name/InstructionSet
 	
Both yield following response:
 	
Return payload
 	
`{ "Id": 1, "Name": "InstructionSet", "CreatedTimestampUtc": "2019-11-07T13:14:52.75Z", "ModifiedTimestampUtc": "2019-11-07T13:14:56.113Z", "IsGlobal": false, "Description": "Execute, schedule, cancel, and approve instructions - view responses" "Operations": [{ "Id": 2, "OperationName": "Actioner", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet" }, { "Id": 4, "OperationName": "Approver", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet" }, { "Id": 3, "OperationName": "Questioner", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet" }, { "Id": 1, "OperationName": "Viewer", "SecurableTypeId": 1, "SecurableTypeName": "InstructionSet" }] }`
	Use SecurableTypes object inside the Tachyon connector instance.
	
	To retrieve a Securable Type using its Id use:
	
	Retrieving a Securable Type by Id
	
	`secType = connector.SecurableTypes.Get (1);`
	
	To retrieve a Securable Type using its Name use:
	
	Retrieving Securable Type by Name
	
	`secType = connector.SecurableTypes.Get ("InstructionSet");`
	
	"secType" object will contain the same data you can see in the JSON response on the left.

Getting Applicable Operations

Applicable Operations work in the context of a Securable Type, which is why Tachyon allows you to retrieve all Applicable Operations for given Securable Type. You can use either Name or Id to specify which Securable Type you want Applicable Operations for.

Direct Consumer API call	C# code using Consumer SDK library

If you wish to use the Id, make a GET request to https://my.tachyon.server/Consumer/ ApplicableOperations/SecurableTypeId/1

If you wish to use the Name instead, make a GET request to https://my.tachyon.server /Consumer/ ApplicableOperations/SecurableType Name/InstructionSet

Both yield the following response:

Return payload


```
[
    {
        "Id": 2,
        "OperationName": "Actioner",
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet"
    },
    {
        "Id": 4,
        "OperationName": "Approver",
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet"
    },
    {
        "Id": 3,
        "OperationName": "Questioner",
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet"
    },
    {
        "Id": 1,
        "OperationName": "Viewer",
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet"
    }
]
```


Use ApplicableOperations object inside the Tachyon connector instance.

To retrieve Applicable Operations for a Securable Type using the type's Id use:

Retrieve Applicable Operations using Securable Type's Id

```
operations = connector.
ApplicableOperations.Get(1);
```

To retrieve Applicable Operations for a Securable Type using the type's Name use:

Retrieve Applicable Operations using Securable Type's Name

```
operations = connector.
ApplicableOperations.Get
("InstructionSet");
```

"operations" object will contain the same data you can see in the JSON response on the left.

Getting Permissions

You can retrieve permissions in several different ways, for example for a Principal, a Role or a Securable Type.

Some APIs perform checks for the calling user by pulling the user information from the HTTP request itself. Other APIs allow you to specify which object you want to get permissions for.

Getting permissions for a Principal

Getting all Permissions

Retrieving a principal's permissions is done using the account name (for example "somedomain\jane.doe") of that principal. When directly using the API, you have to encode that account name into base64 before sending it. C# Consumer API SDK will do the encoding for you.

In general, any GET endpoint requires you to base64 encode the principal name, due to the fact that principal names can contain characters that are not allowed in URIs.

When you request permissions for a specific Principal, you'll get the permissions that Principal has stemming from any of their roles.

Please note that these Permissions will specify which Management Group they have been granted on, which reflect on how the Role containing these Permissions is assigned to the Principal.

If given Principal has the same Permission on two different Management Groups (for instance because those Permissions come from two different Roles, which have been assigned using a different Management Group, or perhaps the same Role assigned twice, each time using a different Management Group), there will be two rows for the same Permission which will differ in the Management Group Id and Name.

Another important thing to remember is that a non-global securable type will yield extra Permission row(s), one per child of the Management Group that the original Permission has been assigned on.

What this means is that in an example we saw earlier, if "Actioner" Role is assigned on "Europe" Management Group to a Principal, asking about that Principal's Permission with yield 4 rows - one for "Europe", one for "UK", one for "France" and one for "Italy". This is because "InstructionSet" is a local Securable Type.

Had "Actioner" has Permisison for, for instance, "InfrastructureLog" Securable Type, which is global, and was assigned to "Europe", there would only be one row, the one containing "Europe".

The following examples use the "somedomain\jane.doe" account.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer /Permissions/Principal /c29tZWRvbWFpblxqYW5lLmRvZQ== will yield following response:	Use Permissions object inside the Tachyon connector instance.

Return payload


```
[{
    "ManagementGroupId": 1,
    "ManagementGroupName": "All Devices",
    "SecurableId": null,
    "SecurableName": null,
    "SecurableTypeId": 4,
    "SecurableTypeName": "Instrumentation",
    "RoleId": 4,
    "RoleName": "Infrastructure Administrators",
    "Allowed": true,
    "Operations": [{
        "PermissionId": 19,
        "OperationId": 11,
        "OperationName": "Read",
        "CreatedTimestampUtc": "2019-06-07T15:12:40.757Z",
        "ModifiedTimestampUtc": "2019-06-07T15:12:40.757Z"
    }]
},
{
    "ManagementGroupId": 1,
    "ManagementGroupName": "All Devices",
    "SecurableId": null,
    "SecurableName": null,
    "SecurableTypeId": 12,
    "SecurableTypeName": "ProcessLog",
    "RoleId": 16,
    "RoleName": "Log Viewers",
    "Allowed": true,
    "Operations": [{
        "PermissionId": 61,
        "OperationId": 33,
        "OperationName": "Read",
        "CreatedTimestampUtc": "2019-06-07T15:12:44.13Z",
        "ModifiedTimestampUtc": "2019-06-07T15:12:44.13Z"
    }]
},
{
    "ManagementGroupId": 1,
    "ManagementGroupName": "All Devices",
    "SecurableId": null,
    "SecurableName": null,
    "SecurableTypeId": 13,
    "SecurableTypeName": "SynchronizationLog",
    "RoleId": 16,
    "RoleName": "Log Viewers",
    "Allowed": true,
    "Operations": [{
        "PermissionId": 62,
        "OperationId": 34,
        "OperationName": "Read",
        "CreatedTimestampUtc": "2019-06-07T15:12:44.13Z",
        "ModifiedTimestampUtc": "2019-06-07T15:12:44.13Z"
    }]
},
{
    "ManagementGroupId": 1,
    "ManagementGroupName": "All Devices",
    "SecurableId": null,
```


Retrieving all Permissions for a specific Principal

```
permissions = connector.Permissions.
GetForPrincipal("somedomain\\jane.doe");
```

"permissions" object will contain the same data you can see in the JSON response on the left.

```
    "SecurableName": null,
    "SecurableTypeId": 14,
    "SecurableTypeName": "Component",
    "RoleId": 17,
    "RoleName": "Component Viewers",
    "Allowed": true,
    "Operations": [{
        "PermissionId": 64,
        "OperationId": 35,
        "OperationName": "Read",
        "CreatedTimestampUtc": "2019-06-07T15:12:44.13Z",
        "ModifiedTimestampUtc": "2019-06-07T15:12:44.13Z"
    }]
},
{
    "ManagementGroupId": 1,
    "ManagementGroupName": "All Devices",
    "SecurableId": null,
    "SecurableName": null,
    "SecurableTypeId": 16,
    "SecurableTypeName": "InfrastructureLog",
    "RoleId": 16,
    "RoleName": "Log Viewers",
    "Allowed": true,
    "Operations": [{
        "PermissionId": 63,
        "OperationId": 39,
        "OperationName": "Read",
        "CreatedTimestampUtc": "2019-06-07T15:12:44.13Z",
        "ModifiedTimestampUtc": "2019-06-07T15:12:44.13Z"
    }]
}]
```


Checking for a specific Permission

You can also retrieve permissions given a Principal has on a particular Securable Type.

In the example below, we'll look at InstructionSet related permissions "somedomain\jane.doe" account has.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer /Permissions /Principal/c29tZWRvbWFpblxqYW5lLmRvZQ==/Type/InstructionSet will yield following response:	Use Permissions object inside the Tachyon connector instance.

Return payload


```
[
    {
        "ManagementGroupId": 1,
        "ManagementGroupName": "All Devices",
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 1,
        "RoleName": "Global Administrators",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 1,
                "OperationId": 1,
                "OperationName": "Viewer",
                "CreatedTimestampUtc": "2019-11-07T13:14:
52.77Z",
                "ModifiedTimestampUtc": "2019-11-07T13:
14:52.77Z"
            },
```


Retrieving InstructionSet related permissions for a Principal

```
permissions = connector.Permissions.
GetForPrincipalAndType("somedomain\\jane.
doe", "InstructionSet");
```

"permissions" object will contain the same data you can see in the JSON response on the left.

```
            {
                "PermissionId": 2,
                "OperationId": 2,
                "OperationName": "Actioner",
                "CreatedTimestampUtc": "2019-11-07T13:14:
52.77Z",
                "ModifiedTimestampUtc": "2019-11-07T13:
14:52.77Z"
            },
            {
                "PermissionId": 3,
                "OperationId": 3,
                "OperationName": "Questioner",
                "CreatedTimestampUtc": "2019-11-07T13:14:
52.77Z",
                "ModifiedTimestampUtc": "2019-11-07T13:
14:52.77Z"
            },
            {
                "PermissionId": 4,
                "OperationId": 4,
                "OperationName": "Approver",
                "CreatedTimestampUtc": "2019-11-07T13:14:
52.77Z",
                "ModifiedTimestampUtc": "2019-11-07T13:
14:52.77Z"
            }
        ]
    },
    {
        "ManagementGroupId": 1,
        "ManagementGroupName": "All Devices",
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 5,
        "RoleName": "Global Approvers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 5,
                "OperationId": 4,
                "OperationName": "Approver",
                "CreatedTimestampUtc": "2019-11-07T13:14:
52.77Z",
                "ModifiedTimestampUtc": "2019-11-07T13:
14:52.77Z"
            }
        ]
    }
]
```


If given Securable Type supports Instances, you can also check a Principal's permissions on a specific instance.

In the example below we'll use a different account called "somedomain\john.doe" and check what permissions it has on Instruction Set with the Id of 1.

Direct Consumer API call	**C# code using Consumer SDK library**

Making a GET request to https://my.tachyon.server/Consumer /Permissions /Principal/c29tZWRvbWFpblxqb2huLmRvZQ==/Type/InstructionSet/1 will yield following response:

Return payload


```
[
    {
        "ManagementGroupId": 1,
        "ManagementGroupName": "All Devices",
        "SecurableId": 1,
        "SecurableName": "MySet",
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 30,
        "RoleName": "MySet Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 137,
                "OperationId": 1,
                "OperationName": "Viewer",
                "CreatedTimestampUtc": "2020-01-
02T12:04:04.963Z",
                "ModifiedTimestampUtc": "2020-01-
02T12:04:04.963Z"
            }
        ]
    }
]
```


Use Permissions object inside the Tachyon connector instance.

Retrieving Principal's permissions to a specific instruction set

```
permissions = connector.Permissions.
GetForPrincipalAndTypeAndInstance
("somedomain\\john.doe", "InstructionSet", 1);
```

"permissions" object will contain the same data you can see in the JSON response on the left.

If the given Principal has no permissions or does not have permissions on given Securable Type and/or instance, an empty collection is returned.

Checking Permissions on a specific Management Group

You can check what Permissions a Principal has on a given Management Group.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Permissions/Principal/ c29tZWRvbWFpblxqb2huLmRvZQ== /ManagementGroup/1 will yield following response:

Return payload


```
[
    {
        "ManagementGroupId": 1,
        "ManagementGroupName": "All Devices",
        "SecurableId": 1,
        "SecurableName": "MySet",
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 30,
        "RoleName": "MySet Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 137,
                "OperationId": 1,
                "OperationName": "Viewer",
                "CreatedTimestampUtc": "2020-01-02T12:04:
04.963Z",
                "ModifiedTimestampUtc": "2020-01-02T12:04:
04.963Z"
            }
        ]
    }
]
```


Use Permissions object inside the Tachyon connector instance.

Retrieving Principal's permissions on a specific Management Group

```
permissions = connector.Permissions.
GetForPrincipalAndManagementGroup
("somedomain\\john.doe", 1);
```

"permissions" object will contain the same data you can see in the JSON response on the left.

You can also check if given Principal has a specific Permission on a specific Management Group

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Permissions/Principal/ c29tZWRvbWFpblxqb2huLmRvZQ== /ManagementGroup/1/Type/InstructionSet will yield following response:

Return payload


```
[
    {
        "ManagementGroupId": 1,
        "ManagementGroupName": "All Devices",
        "SecurableId": 1,
        "SecurableName": "MySet",
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 30,
        "RoleName": "MySet Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 137,
                "OperationId": 1,
                "OperationName": "Viewer",
                "CreatedTimestampUtc": "2020-01-02T12:
04:04.963Z",
                "ModifiedTimestampUtc": "2020-01-02T12:
04:04.963Z"
            }
        ]
    }
]
```


Use Permissions object inside the Tachyon connector instance.

Retrieving a specific type or Permission that a Principal has on a specific Management Group

```
permissions = connector.Permissions.
GetForPrincipalAndTypeAndManagementGroupAndIns
tance("somedomain\\john.doe", 1, 1);
```

"permissions" object will contain the same data you can see in the JSON response on the left.

Getting permissions for a Role

You can retieve Role's permissions by using the role Id.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Permissions/Role/16 will yield following response:

Return payload


```
[
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 12,
        "SecurableTypeName": "ProcessLog",
        "RoleId": 16,
        "RoleName": "Log Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 61,
                "OperationId": 33,
                "OperationName": "Read",
                "CreatedTimestampUtc": "2019-11-07T13:
14:56.673Z",
                "ModifiedTimestampUtc": "2019-11-07T13:
14:56.673Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 13,
        "SecurableTypeName": "SynchronizationLog",
        "RoleId": 16,
        "RoleName": "Log Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 62,
                "OperationId": 34,
                "OperationName": "Read",
                "CreatedTimestampUtc": "2019-11-07T13:
14:56.673Z",
                "ModifiedTimestampUtc": "2019-11-07T13:
14:56.673Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 16,
        "SecurableTypeName": "InfrastructureLog",
        "RoleId": 16,
        "RoleName": "Log Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 63,
                "OperationId": 39,
                "OperationName": "Read",
                "CreatedTimestampUtc": "2019-11-07T13:
14:56.673Z",
                "ModifiedTimestampUtc": "2019-11-07T13:
14:56.673Z"
            }
        ]
    }
]
```


Use Permissions object inside the Tachyon connector instance.

Retrieving all permissions assigned to a Role

```
permissions = connector.Permissions.GetForRole
(16);
```

"permissions" object will contain the same data you can see in the JSON response on the left.

You can also retrieve all permissions given Role has on a specific Securable Type.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Permissions/Role/16 /Type/ProcessLog will yield following response:
 	
Return payload
 	
[
{
"SecurableId": null,
"SecurableName": null,
"SecurableTypeId": 12,
"SecurableTypeName": "ProcessLog",
"RoleId": 16,
"RoleName": "Log Viewers",
"Allowed": true,
"Operations": [
{
"PermissionId": 61,
"OperationId": 33,
"OperationName": "Read",
"CreatedTimestampUtc": "2019-11-07T13:14:56.673Z",
"ModifiedTimestampUtc": "2019-11-07T13:14:56.673Z"
}
]
}
]
	Use Permissions object inside the Tachyon connector instance.
	
	Retrieving permissions that a Role has on a given Securable Type
	
	permissions = connector.Permissions.GetForRoleAndType(16, "ProcessLog");
	
	"permissions" object will contain the same data you can see in the JSON response on the left.

And for Securable Types that support Instances, you can also check Permissions on a specific Instance.

Here we'll get permissions for a custom Role that's been assigned a 'Viewer' permissions on an Instruction Set.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Permissions /Role/30 /Type/InstructionSet/1 will yield following response:

Return payload


```
[
    {
        "SecurableId": 1,
        "SecurableName": "Wszystko",
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 30,
        "RoleName": "MySet Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 137,
                "OperationId": 1,
                "OperationName": "Viewer",
                "CreatedTimestampUtc": "2020-01-
02T12:04:04.963Z",
                "ModifiedTimestampUtc": "2020-01-
02T12:04:04.963Z"
            }
        ]
    }
]
```


Use Permissions object inside the Tachyon connector instance.

Retrieving permissions a Role has on a specific Instance of a Securable Type

```
permissions = connector.Permissions.
GetForRoleAndTypeAndInstance(30,
"InstructionSet", 1);
```

"permissions" object will contain the same data you can see in the JSON response on the left.

Getting permissions for a Securable Type

You can retrieve permissions granted on a Securable Type using the type's Id. This will return a collection of all permissions on a given type assigned to any of the Roles.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Permissions/ Securable/1 will yield following response:

Return payload


```
[
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 1,
        "RoleName": "Global Administrators",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 1,
                "OperationId": 1,
                "OperationName": "Viewer",
                "CreatedTimestampUtc": "2019-11-07T13:14:
52.77Z",
                "ModifiedTimestampUtc": "2019-11-07T13:
14:52.77Z"
            },
            {
                "PermissionId": 2,
                "OperationId": 2,
                "OperationName": "Actioner",
                "CreatedTimestampUtc": "2019-11-07T13:14:
52.77Z",
                "ModifiedTimestampUtc": "2019-11-07T13:
14:52.77Z"
```


Use Permissions object inside the Tachyon connector instance.

Retrieving all Permisisons on a given Securable Type

```
permissions = connector.Permissions.
GetForSecurableType(1);
```

"permissions" object will contain the same data you can see in the JSON response on the left.

```
            },
            {
                "PermissionId": 3,
                "OperationId": 3,
                "OperationName": "Questioner",
                "CreatedTimestampUtc": "2019-11-07T13:14:
52.77Z",
                "ModifiedTimestampUtc": "2019-11-07T13:
14:52.77Z"
            },
            {
                "PermissionId": 4,
                "OperationId": 4,
                "OperationName": "Approver",
                "CreatedTimestampUtc": "2019-11-07T13:14:
52.77Z",
                "ModifiedTimestampUtc": "2019-11-07T13:
14:52.77Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 5,
        "RoleName": "Global Approvers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 5,
                "OperationId": 4,
                "OperationName": "Approver",
                "CreatedTimestampUtc": "2019-11-07T13:14:
52.77Z",
                "ModifiedTimestampUtc": "2019-11-07T13:
14:52.77Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 6,
        "RoleName": "Global Questioners",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 21,
                "OperationId": 3,
                "OperationName": "Questioner",
                "CreatedTimestampUtc": "2019-11-07T13:14:
54.96Z",
                "ModifiedTimestampUtc": "2019-11-07T13:
14:54.96Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 7,
        "RoleName": "Global Actioners",
        "Allowed": true,
        "Operations": [
            {
```

```
                "PermissionId": 22,
                "OperationId": 2,
                "OperationName": "Actioner",
                "CreatedTimestampUtc": "2019-11-07T13:14:
54.96Z",
                "ModifiedTimestampUtc": "2019-11-07T13:
14:54.96Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 8,
        "RoleName": "Global Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 23,
                "OperationId": 1,
                "OperationName": "Viewer",
                "CreatedTimestampUtc": "2019-11-07T13:14:
54.96Z",
                "ModifiedTimestampUtc": "2019-11-07T13:
14:54.96Z"
            }
        ]
    },
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 27,
        "RoleName": "ServiceNow ITSM Connect",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 136,
                "OperationId": 1,
                "OperationName": "Viewer",
                "CreatedTimestampUtc": "2019-11-07T13:18:
05.767Z",
                "ModifiedTimestampUtc": "2019-11-07T13:
18:05.767Z"
            }
        ]
    }
]
```


You can also retrieve permissions granted on a specific Instance of a Securable Type.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Permissions/Securable/1/1 will yield following response:

Return payload

```
[
    {
        "SecurableId": 1,
        "SecurableName": null,
        "SecurableTypeId": 1,
        "SecurableTypeName": "InstructionSet",
        "RoleId": 30,
        "RoleName": "MySet Viewers",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 137,
                "OperationId": 1,
                "OperationName": "Viewer",
                "CreatedTimestampUtc": "2020-01-02T12:04:
04.963Z",
                "ModifiedTimestampUtc": "2020-01-02T12:
04:04.963Z"
            }
        ]
    }
]
```

Use Permissions object inside the Tachyon connector instance.

Retrieving permissions assigned on a specific instance of a Securable Type

```
connector.Permissions.
GetForSecurableTypeAndInstance(1, 1);
```

"permissions" object will contain the same data you can see in the JSON response on the left.

Helper APIs

Consumer API has a number of endpoints that are not directly involved with its RBAC system but instead provide various utility functions around the general principal and permission area.

Who am I?

You can ask the API to return the information about the user who's making the request. The API examines the incoming request, retrieves the name of the user from it, makes sure the user is authenticated, and returns information summary about the user, like user's principal name, SID, display name and email.

This endpoint is useful for systems that cannot assume they're running under a particular account, like a browser or an application running in the context of launching user. In those cases this endpoint can be used to obtain the name of the user, which is then fed to the Permissions endpoint seen above in order to retrieve the permissions of the caller.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/PrincipalSearch/whoami will yield following response:	Use PrincipalSearch object inside the Tachyon connector instance.

Return payload

```
{
    "PrincipalName": "Somedomain\\Jane.Doe",
    "ExternalId": "S-1-5-21-1202660629-789336058-
1343024091-19970",
    "Email": "Jane.Doe@SomeDomain.com",
    "DisplayName": "Jane Doe",
    "Photo": null
}
```

Who am I?

```
user = connector.PrincipalSearch.
GetCurrentlyLoggedInUser();
```

"user" object will contain the same data you can see in the JSON response on the left.

Active Directory search

Because in order to add a Principal to Tachyon you have to know the SID of the user, Tachyon's Consumer API exposes endpoints that help you look up users in Active Directory and retrieve information about them, including their SID.

You can perform a search for Active Directory accounts whose common name or sAMAccountName contain given phrase. Details of what exact search is performed are beyond the scope of this page and depend on type of objects being searched for, like users, groups and computers and whether your Tachyon installation is using GC or LDAP.

In the examples below we'll be looking for accounts that contain the phrase "Administrator". How you perform this search depends on your version of Tachyon.

Tachyon versions up to and including 4.1

You simply provide a base64 encoded search string and issue a GET request.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/PrincipalSearch/ QWRtaW5pc3RyYXRvcg== will yield following response: **Return payload**	Use PrincipalSearch object inside the Tachyon connector instance. **Searching for principals in Active Directory** `results = connector.PrincipalSearch.SearchForPrincipals("Administrator");` "results" object will contain the same data you can see in the JSON response on the left.

```
[
        {
                "PrincipalName":
"SomeDomain\\CSUserAdministrator",
                "ExternalId": "S-1-5-21-1202660629-
789336058-1353024091-8145",
                "Email": null,
                "DisplayName": "CSUserAdministrator",
                "IsGroup": true
        },
        {
                "PrincipalName":
"SomeDomain\\CSVoiceAdministrator",
                "ExternalId": "S-1-5-21-1202660629-
789336058-1343023091-8024",
                "Email": null,
                "DisplayName": "CSVoiceAdministrator",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDomain\\Exchange
Organization Administrators",
                "ExternalId": "S-1-5-21-1202660629-
789336058-1343324091-6881",
                "Email": null,
                "DisplayName": "Exchange Organization
Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDomain\\Exchange
Public Folder Administrators",
                "ExternalId": "S-1-5-21-1202660629-
789336058-1340024091-7747",
                "Email": null,
                "DisplayName": "Exchange Public Folder
Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDomain\\Exchange
Recipient Administrators",
                "ExternalId": "S-1-5-21-1202660629-
789336058-1340024091-6913",
                "Email": null,
                "DisplayName": "Exchange Recipient
Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDomain\\Exchange
View-Only Administrators",
                "ExternalId": "S-1-5-21-1202660629-
789336058-1343024001-6999",
                "Email": null,
                "DisplayName": "Exchange View-Only
Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDomain\\Security
Administrator",
                "ExternalId": "S-1-5-21-1202660629-
789336058-1343021091-58611",
                "Email": null,
                "DisplayName": "Security
Administrator",
                "IsGroup": true
        }
]
```

Here you have greater control over the search. A POST request should be used and in its payload you can specify, apart from the search string, number of results returned, sort column and order.

You can search for "user" and "group" object types and sort in either ascending (using "ASC") or descending order (using "DESC") on following columns: "cn", "mail", "sAMAccountName", "description", "objectSid", "displayName". Column names, object types and sorting direction string are not case sensitive.

You have to provide the SearchText (the text to search for) and at least one object type. Pagesize and sort are optional and if not provided will default to respectively 100 and ascending sorting on display name.

One things that you have to keep in mind when it comes to PageSize property is that, due to internal implementation, it defines the size of the page returned by Active Directory. This means that you can get fewer results than you requested, even if there would have been more to return because Tachyon filters out users already in the system. As an example, if you define the page size to be 10 and the Active Directory has 18 entries that match your search string, it will return 10 entries. Now let us assume that out of those 10 entries 1 user is already in Tachyon. This user will be filtered out of the return data set, so you will receive only 9 entries.

Direct Consumer API call

Making a POST request to https://my.tachyon.server/Consumer/PrincipalSearch with following payload:

Request payload

```
{
        "SearchText": "Administrator",
        "ObjectTypes": ["user", "group"],
        "PageSize": 100,
        "Sort": {
                "Column": "displayname",
                "Direction": "ASC"
        }
}
```

will yield following response:

Return payload

C# code using Consumer SDK library

Use PrincipalSearch object inside the Tachyon connector instance.

Retrieving 100 entries sorted by displayname

```
var searchParams = new
ActiveDirectorySearchModel
{
        SearchText = "Administrator",
        ObjectTypes = new List<string> {
"user", "group" },
        PageSize = 100,
        Sort = new SortSpec
        {
                Column = "displayname",
                Direction = "ASC"
        }
};

results = connector.PrincipalSearch.
SearchForPrincipals(searchParams);
```

"results" object will contain the same data you can see in the JSON response on the left.

```
[
        {
                "PrincipalName":
"SomeDomain\\CSUserAdministrator",
                "ExternalId": "S-1-5-21-1202660629-
789336058-1353024091-8145",
                "Email": null,
                "DisplayName": "CSUserAdministrator",
                "IsGroup": true
        },
        {
                "PrincipalName":
"SomeDomain\\CSVoiceAdministrator",
                "ExternalId": "S-1-5-21-1202660629-
789336058-1343023091-8024",
                "Email": null,
                "DisplayName": "CSVoiceAdministrator",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDomain\\Exchange
Organization Administrators",
                "ExternalId": "S-1-5-21-1202660629-
789336058-1343324091-6881",
                "Email": null,
                "DisplayName": "Exchange Organization
Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDomain\\Exchange
Public Folder Administrators",
                "ExternalId": "S-1-5-21-1202660629-
789336058-1340024091-7747",
                "Email": null,
                "DisplayName": "Exchange Public Folder
Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDomain\\Exchange
Recipient Administrators",
                "ExternalId": "S-1-5-21-1202660629-
789336058-1340024091-6913",
                "Email": null,
                "DisplayName": "Exchange Recipient
Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDomain\\Exchange
View-Only Administrators",
                "ExternalId": "S-1-5-21-1202660629-
789336058-1343024001-6999",
                "Email": null,
                "DisplayName": "Exchange View-Only
Administrators",
                "IsGroup": true
        },
        {
                "PrincipalName": "SomeDomain\\Security
Administrator",
                "ExternalId": "S-1-5-21-1202660629-
789336058-1343021091-58611",
                "Email": null,
                "DisplayName": "Security Administrator",
                "IsGroup": true
        }
]
```

Retrieving Active Directory information about a specific account

You can also retrieve user's information by their principal name, though this endpoint only returns authenticated users who are also Principals in Tachyon (either directly or throught group membership).

This means that in order to obtain information about somedomain\jane.doe using this endpoint, somedomain\jane.doe has to be a domain account that's either a Principal in Tachyon, or belongs to an Active Directory group that is a Principal in Tachyon.

When using the API directly, the principal name has to be base64 encoded. Using the SDK encoding is not necessary because the SDK will perform the encoding internally.

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/PrincipalSearch/user/c29tZWRvbWFpblxqYW5lLmRvZQ== will yield following response: **Return payload** `[ { "PrincipalName": "SomeDomain\\Jane.Doe", "ExternalId": "S-1-5-21-1202660629-789336058-1343024091-23842", "Email": "jane.doe@somedomain.com", "DisplayName": "Jane Doe", "IsGroup": false } ]`	Use PrincipalSearch object inside the Tachyon connector instance. **Retrieve information about a principal from Active Directory** `results = connector.PrincipalSearch.SearchForUser("SomeDomain\\Jane.Doe");` "results" object will contain the same data you can see in the JSON response on the left.

Retrieving members of an Active Directory group

You can ask Tachyon to retrieve members of an Active Directory group. This group does not have to be a principal in Tachyon for this functionality to work.

When using the API directly the group name has to be base64 encoded. Using the SDK encoding is not necessary because the SDK will perform the encoding internally.

In the example below we'll use an AD group called "Tachyon users" and retrieve its members.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/PrincipalSearch/GetMembers/VGFjaHlvbiB1c2Vycw== will yield following response:

Return payload

```
[
    {
        "PrincipalName": "SomeDomain\\Jane.Doe",
        "ExternalId": "S-1-5-21-1202660629-789336058-
1343024091-23842",
        "Email": "jane.doe@somedomain.com",
        "DisplayName": "Jane Doe",
        "IsGroup": false
    },
    {
        "PrincipalName": "SomeDomain\\Meetra.Surnik",
        "ExternalId": "S-1-5-21-1202660629-789336058-
1343024091-8631",
        "Email": "Meetra.Surnik@somedomain.com",
        "DisplayName": "Meetra Surnik",
        "IsGroup": false
    },
    {
        "PrincipalName": "SomeDomain\\Keiran.Halcyon",
        "ExternalId": "S-1-5-21-1202660629-789336058-
1343024091-1138",
        "Email": "Keiran.Halcyon@somedomain.com",
        "DisplayName": "Keiran Halcyon",
        "IsGroup": false
    }
]
```

Use PrincipalSearch object inside the Tachyon connector instance.

Retrieving members of an Acive Directory group

```
results = connector.PrincipalSearch.
GetGroupMembers("Tachyon users");
```

"results" object will contain the same data you can see in the JSON response on the left.

Configuring Tachyon RBAC through the Consumer API

In this section we will look at configuring RBAC emphasizing operations on Principals, Roles and Permissions but also covering Securable Types and Operations later on.

Adding Principals from Active Directory

Adding Principals to Tachyon will be one of the first things you'll do after installing the system. A fresh installation of Tachyon (which in this context means an installation where there was no previous Master database or such database was dropped) will have two Principals - the account used to install the Tachyon Server (and created the database to be specific) and NT AUTHORITY\Network Service account.

These accounts have limited permissions so to properly use the system you should add more Principals. In order to add a principal to Tachyon you will need an External Id, also called SID, for an active directory account that you wish to add as a principal. Although you can obtain account details through various means. In Active Directory search we've seen Tachyon exposes endpoints that return this information and enables you to search for Active Directory accounts.

Here we'll assume you already have the account details and focus on adding an account as a Principal to Tachyon.

In the example below we'll "SomeDomain\Jane.Doe" account and add it to Tachyon.

To add an account as a Principal you'll at the very least need to supply an ExternalId (SID) and Principal Name, though its advised to also supply a Display Name. Also, unless you set the Enabled flag to true, the newly created account will be disabled by default.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request to https://my.tachyon.server/Consumer/Principals with following payload:

Request payload

```
{
    "PrincipalName": "SomeDomain\\Jane.Doe",
    "ExternalId": "S-1-5-21-1202660629-789336058-
1343024091-23842",
    "Email": "Jane.Doe@SomeDomain.com",
    "DisplayName": "Jane Doe",
    "IsGroup": false,
    "Enabled": true
}
```

Will yield following response:

Return payload

```
{
    "Id": 3,
    "ExternalId": "S-1-5-21-1202660629-789336058-
1343024091-23842",
    "PrincipalName": "SomeDomain\\Jane.Doe",
    "Email": "Jane.Doe@SomeDomain.com",
    "Enabled": true,
    "CreatedTimestampUtc": "2019-11-07T13:14:52.777
Z",
    "ModifiedTimestampUtc": "2019-11-07T13:14:52.777
Z",
    "SystemPrincipal": false,
    "DisplayName": "Jane Doe",
    "IsGroup": false
}
```

Use Principals object inside the Tachyon connector instance.

Adding a Principal to Tachyon

```
var payload = new Principal
{
    PrincipalName = "SomeDomain\\Jane.Doe",
    ExternalId = "S-1-5-21-1202660629-789336058-
1343024091-23842",
    Email = "Jane.Doe@SomeDomain.com",
    DisplayName = "Jane Doe",
    IsGroup = false,
    Enabled = true
};

principal = connector.Principals.Add(payload);
```

"principal" object will contain the same data you can see in the JSON response on the left.

You cannot add a Principal with the same PrincipalName or ExternalId as another Principal that's already in the system, because both of those values must be unique.

Updating a Principal

Updating a Principal is very similar to adding one. You have to use a PUT verb instead of a POST if using the API directly, or Update method instead of Add if you're using the SDK. You also have to provide an Id that belongs to an existing Principal. Remember, you cannot modify system Principals.

Again, you have to remember that ExternalId and PrincipalName must remain unique so you cannot change Principal's PrincipalName or ExternalId to one that matches another Principal's.

Configuring Roles

Roles are the pivotal point of an RBAC system.

Tachyon comes with a number of pre-defined roles, which should be sufficient to start with, but in time you will most likely create custom roles, either to restrict access to instruction sets or to cover sets of permission required by specific roles within your organization.

Adding, Editing and Removing a Role

First let's look at creating a Role. This can be done either in isolation, where just the role is created and any permissions must be assigned in subsequent call(s), or as a more complete package, where a Role is created along with a set of Permissions, which can include Management Groups.

Regardless of which approach is used, Principals have to be assigned separately.

To create a Role all you require is a name, and this name must be unique. Description can optionally be provided and you can't create system roles.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request to https://my.tachyon.server/Consumer/Roles with following payload:

Request payload

```
{
    "Name": "Custom role",
    "Description": "this is a description"
}
```

Will yield following response:

Return payload

```
{
    "Id": 31,
    "Name": "Custom role",
    "Description": "this is a description",
    "CreatedTimestampUtc": "2020-01-21T13:36:
55.8956916Z",
    "ModifiedTimestampUtc": "2020-01-21T13:36:
55.8956916Z",
    "SystemRole": false
}
```

Use Roles object inside the Tachyon connector instance.

Creating a Role

```
var payload = new Role
{
    Name = "Custom role",
    Description = "this is a description"
};

role = connector.Roles.Add(payload);
```

"role" object will contain the same data you can see in the JSON response on the left.

Creating a Role with Permissions is possible by joining Role creation we've just seen with creating permissions, which is described a bit later.

This effectively means that we take a regular Role creation payload and add a 'Permissions' property to it. That propery is an array of permissions, which we are yet to look at in greater details.

The example below creates a role, which has "Viewer" and "Approver" permissions on Instruction Set with the Id of 1 and "Actioner", "Questioner", "Viewer" and "Approver" permissions on Instruction Set with the Id of 2. It also has permissions on three Management Groups.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/Roles/Complete with following payload:	

Request payload

```
{
   "Name": "Complete Role 1",
   "Description": "This is a test role",
   "Permissions": [{
       "SecurableId": 2,
       "SecurableTypeId": 1,
       "Allowed": true,
       "Operations": [{
           "OperationId": 2
       },
       {
           "OperationId": 4
       },
       {
           "OperationId": 3
       },
       {
           "OperationId": 1
       }]
   },
   {
       "SecurableId": 1,
       "SecurableTypeId": 1,
       "Allowed": true,
       "Operations": [{
           "OperationId": 1
       },
       {
           "OperationId": 4
       }]
   }]
}
```

Will yield following response:

Return payload

```
{
    "Permissions": [
        {
            "SecurableId": 1,
            "SecurableName": null,
            "SecurableTypeId": 1,
            "SecurableTypeName":
"InstructionSet",
            "RoleId": 32,
            "RoleName": "Complete Role 1",
            "Allowed": true,
            "Operations": [
                {
                    "PermissionId": 142,
                    "OperationId": 1,
                    "OperationName": "Viewer",
                    "CreatedTimestampUtc":
"2020-01-22T07:18:57.73Z",
                    "ModifiedTimestampUtc":
"2020-01-22T07:18:57.73Z"
                },
                {
                    "PermissionId": 143,
                    "OperationId": 4,
                    "OperationName":
"Approver",
                    "CreatedTimestampUtc":
"2020-01-22T07:18:57.73Z",
```

Use Roles object inside the Tachyon connector instance.

Creating a Role with Permissions

```
var payload = new Tachyon.SDK.Consumer.Models.Send.
RoleWithPermissionsAndManagementGroups
{
    Name = "Complete Role 1",
    Description = "This is a test role",
    Permissions = new AggregatedPermission[]
    {
        new AggregatedPermission
        {
            SecurableId = 2,
            Allowed = true,
            SecurableTypeId = 1,
            Operations = new List<PermissionOperation>
            {
                new PermissionOperation { OperationId =
1 },
                new PermissionOperation { OperationId =
2 },
                new PermissionOperation { OperationId =
3 },
                new PermissionOperation { OperationId =
4 }
            }
        },
        new AggregatedPermission
        {
            SecurableId = 1,
            Allowed = true,
            SecurableTypeId = 1,
            Operations = new List<PermissionOperation>
            {
                new PermissionOperation { OperationId =
1 },
                new PermissionOperation { OperationId =
4 }
            }
        }
    }
};

role = connector.Roles.Add(payload);
```

"role" object will contain the same data you can see in the JSON response on the left.

```
                    "ModifiedTimestampUtc":
"2020-01-22T07:18:57.73Z"
                }
            ]
        },
        {
            "SecurableId": 2,
            "SecurableName": null,
            "SecurableTypeId": 1,
            "SecurableTypeName":
"InstructionSet",
            "RoleId": 32,
            "RoleName": "Complete Role 1",
            "Allowed": true,
            "Operations": [
                {
                    "PermissionId": 138,
                    "OperationId": 2,
                    "OperationName":
"Actioner",
                    "CreatedTimestampUtc":
"2020-01-22T07:18:57.73Z",
                    "ModifiedTimestampUtc":
"2020-01-22T07:18:57.73Z"
                },
                {
                    "PermissionId": 139,
                    "OperationId": 4,
                    "OperationName":
"Approver",
                    "CreatedTimestampUtc":
"2020-01-22T07:18:57.73Z",
                    "ModifiedTimestampUtc":
"2020-01-22T07:18:57.73Z"
                },
                {
                    "PermissionId": 140,
                    "OperationId": 3,
                    "OperationName":
"Questioner",
                    "CreatedTimestampUtc":
"2020-01-22T07:18:57.73Z",
                    "ModifiedTimestampUtc":
"2020-01-22T07:18:57.73Z"
                },
                {
                    "PermissionId": 141,
                    "OperationId": 1,
                    "OperationName": "Viewer",
                    "CreatedTimestampUtc":
"2020-01-22T07:18:57.73Z",
                    "ModifiedTimestampUtc":
"2020-01-22T07:18:57.73Z"
                }
            ]
        }
    ]
    "Id": 32,
    "Name": "Complete Role 1",
    "Description": "This is a test role",
    "CreatedTimestampUtc": "2020-01-22T07:18:
57.7082854Z",
    "ModifiedTimestampUtc": "2020-01-22T07:18:
57.7082854Z",
    "SystemRole": false
}
```

To update a Role you should use PUT versions of the two endpoints seen above and provide one additional property in the JSON payload - the Id of the Role being modified. If you're using the Consumer SDK, you should call Update method instead of Add and you must provide an Id in the payload object sent to the API.

Since the names of Roles have to be unique, you won't be able to change the name of a Role to one that is the same as another Role.

Changing the Role details with https://my.tachyon.server/Consumer/Roles (or using the Update method in Consumer SDK that takes Role object type) will change just the details of the role will not change the permissions in any way. It will not change which Mangement Groups or Principals are assigned to the Role either.

Changing the Role via https://my.tachyon.server/Consumer/Roles/Complete will not change which Principals are assigned to the Role, but it will change the Permissions and Management Groups. When using this endpoint with PUT verb (or using the Update method in Consumer SDK that takes RoleWithPermissionsAndManagementGroups object type), all of the Permissions for given Role will be replaced with Permissions provided in the payload and all Management Groups will be replaced with Management Groups provided in the payload. This means that if you do not provide Permissions (omit the property, provide a null value or an empty array), all Permissions will be removed from the Role. Likewise, if you do not provide ManagementGroups all Management Groups will be unassigned from the Role.

Deleting a role is straightforward and you just need its Id. You can also delete multiple Roles in one call by providing a collection of Role Ids. In either case, you cannot delete a system Role.

Direct Consumer API call	C# code using Consumer SDK library

To delete a single Role make a DELETE request to https://my.tachyon.server/Consumer/Roles/31 , which will delete the Role with the Id of 31.
 	
To delete multiple Roles, make a DELETE request and send an array of Ids to https://my.tachyon.server/Consumer/Roles. To delete Roles with Ids 31, 32, 33 and 34 you would send this payload:
 	
Request payload
`[31,32,33,34]`
	Use Roles object inside the Tachyon connector instance.
	To delete a single Role:
	
	Deleting the Role with Id 31
	`connector.Roles.Delete(31);`
	
	To delete multiple Roles:code
	
	Deleting Roles with Ids 31,32,33 and 34.
	`connector.Roles.DeleteMultiple (new List<int> {31, 32, 33, 34});`

Adding and removing Permissions to a Role

The main method of assigning Permissions is to send two collections, one with Permissions that need to be changed or updated and another one with permissions that should be removed.

Crafting the correct payload should be done carefully as omitting an entry can result in deletion of an existing Permission.

PermissionsToSaveOrUpdate collection should have permissions you either want to add or modify. You only need to provide Ids, names are ignored. It is important to understand when the system considers given operation an 'updated' and when a 'creation' as it will behave slightly differently.

Because a Permission is, effectively, a link between a Securable Type and a Role, with possible addition of Securable Id, those three properties define it. Operations are considered a property of a Permission and they are always considered together - i.e. a single Permission can Allow or Deny a set of Operations, but it cannot mix and match.

When you request a Permission to be added or updated, Consumer API will look at the SecurableId, SecurableTypeId and RoleId properties provided and check for any permissions already existing for this combination of values. If none are found, a new permission is added with the operations specified in the payload. If an entry is found, then Operations are examined next. If the existing Permission has Operations that aren't present in the Permission sent to the API, they are removed. Any operations present in the Permission sent to the API that do not exist in the Permission that is already in the system are added.

As you might have noticed, PermissionsToSaveOrUpdate can be used to remove Permissions by omitting Operations you want to remove. Furthermore, if you provide a Permission without any Operations, the entire Permission is removed.

Lets look at an example that should help us better understand how this functionality works.

Let us assume that we have a Permission on Instruction Set "My set" assigned to Role "Custom Role" with just one Operation "View". "My Set" has the Id of 4, "Custom Role" has the Id of 31, "Instruction Set" Securable Type has the Id of 1 and "Viewer" has the Id of 1. This means that the existing Permission will look as follows:

Existing Permission

```
{
    "Allowed": true,
    "SecurableTypeId": 1,
    "SecurableId": 4,
    "RoleId": 31,
    "Operations": [
                {
                "OperationId": 1
        }
        ]
}
```

Now let us assume that a request comes into the Consumer API that looks as follows:

Request payload

```
{
    "PermissionsToSaveOrUpdate": [
        {
            "Allowed": true,
            "SecurableTypeId": 1,
            "SecurableId": 4,
            "RoleId": 31,
            "Operations": [
                                {
                                "OperationId": 1
                        },
                                {
                                "OperationId": 3
                        }
                        ]
        }
    ],
    "PermissionsToDelete": []
}
```

This request matches a Permission we already have, because it has the same SecurableId, SecurableTypeId and RoleId, but the Operations are slightly different - Id of 3 has been added, which represents 'Questioner' Operation. Following the rules outlined above, the API will match this to the existing Permission and see that Operation with the Id of 1 already exists, so no action is needed but the Operation with the Id of 3 doesn't exist, so it needs to be added. Afterwards we will end up with a permission that looks like this:

Existing Permission

```
{
    "Allowed": true,
    "SecurableTypeId": 1,
    "SecurableId": 4,
    "RoleId": 31,
    "Operations": [
                {
                "OperationId": 1
        },
                {
                "OperationId": 3
        }
        ]
}
```

If now we were to make another request with with following payload:

Request payload

```
{
    "PermissionsToSaveOrUpdate": [
        {
            "Allowed": true,
            "SecurableTypeId": 1,
            "SecurableId": 4,
            "RoleId": 31,
            "Operations": [
                                {
                                "OperationId": 2
                        },
                                {
                                "OperationId": 4
                        }
                        ]
        }
    ],
    "PermissionsToDelete": []
}
```

The Consumer API will again recognise that a Permission already exists that matches the one in the payload and will examine the Operations. Because neither operation 1 nor 3, which do exist in the Permission already in the system are present in the payload, they will be removed. Operations 2 and 4 (Actioner and Approve respectively) do not exist in the Permission so they will be added and changed made will result in a Permission that looks like this:

Existing Permission

```
{
    "Allowed": true,
    "SecurableTypeId": 1,
    "SecurableId": 4,
    "RoleId": 31,
    "Operations": [
                {
                "OperationId": 2
        },
                {
                "OperationId": 4
        }
        ]
}
```

As a last example let's look at what will happen when we send this payload to the API:

Request payload

```
{
    "PermissionsToSaveOrUpdate": [
        {
            "Allowed": true,
            "SecurableTypeId": 1,
            "SecurableId": 4,
            "RoleId": 31,
            "Operations": []
        }
    ],
    "PermissionsToDelete": []
}
```

Again, the system will recognise this as an existing Permission and will remove Operations 2 and 4 because they do not exist in the request payload. But since there are no Operations to add, the entire Permission will be removed.

Lastly, let's look at how an example API call would look like:

Direct Consumer API call

Making a POST request to https://my.tachyon.server/Consumer/Permissions with following payload:

Payload sent to https://my.tachyon.server/Consumer/Permissions

```
{
    "PermissionsToSaveOrUpdate": [
        {
            "Allowed": true,
            "SecurableTypeId": 5,
            "SecurableId": null,
            "RoleId": "31",
            "Operations": [
                {
                    "OperationId": 12
                },
                {
                    "OperationId": 13
                }
            ]
        }
    ],
    "PermissionsToDelete": []
}
```

will yield following response:

Response payload

```
[
    {
        "SecurableId": null,
        "SecurableName": null,
        "SecurableTypeId": 5,
        "SecurableTypeName": "CustomProperty",
        "RoleId": 31,
        "RoleName": "Custom Role",
        "Allowed": true,
        "Operations": [
            {
                "PermissionId": 191,
                "OperationId": 12,
                "OperationName": "Read",
                "CreatedTimestampUtc": "2020-01-
23T14:47:13.023Z",
                "ModifiedTimestampUtc": "2020-01-
23T14:47:13.023Z"
            },
            {
                "PermissionId": 192,
                "OperationId": 13,
                "OperationName": "Write",
                "CreatedTimestampUtc": "2020-01-
23T14:47:13.023Z",
                "ModifiedTimestampUtc": "2020-01-
23T14:47:13.023Z"
            }
        ]
    }
]
```

C# code using Consumer SDK library

Use Permissions object inside the Tachyon connector instance.

Creating Permissions

```
var payload = new PermissionsSaveContainer
{
    PermissionsToSaveOrUpdate = new
List<AggregatedPermission>
    {
        new AggregatedPermission
        {
            RoleId = 31,
            SecurableTypeId = 5,
            Operations = new
List<PermissionOperation>
            {
                new PermissionOperation
                {
                    OperationId = 12
                },
                new PermissionOperation
                {
                    OperationId = 13
                }
            }
        }
    }
};
permissions = connector.Permissions.AddOrUpdate
(payload);
```

"permissions" object will contain the same data you can see in the JSON response on the left.

Configuring Assignments

An assignment is a link between a Principal, a Role and a Management Group and we have already briefly explored how they work.

Before we look at how you can use the API to manage assignments, let us go over basic rules and limitations:

- A single assignment is between one Principal, one Role and one Management Group. If you, for instance, wish to assign the same principal to the same role but for multiple Management Groups, you will have to create multiple assignments
- You cannot change assignments of a System Principal.
- If you are a global security administrator, you can use any Role in an assignments. If you are a local security administrator, you can only use delegatable Roles in your assignments. Please see here for details.
- You can manipulate any assignments which uses a Management Group which you have "Security" "Write" Permission on. You will be able to see other assignments but they are marked as "Inaccessible" (see their "AccessType" property).
- You can only assign "Full Administrator" Role to "All Devices" Management Group.
- You cannot assign "Group Administrator" Role to "All Devices" Management Group.
- You cannot assign any Role that has a Permission which uses "Security" securable type to a Management Group which you have "ReadOnly" access to (see Management Group's "AccessType" property"), unless that Management Group is "All Devices".

Now that's out of the way, let us look at two way in which we can create assignments.

The first one is to use the bulk add functionality by issuing a POST request to https://my.tachyon.server/Consumer/PrincipalRoleManagementGroups API.

This API takes a collection of assignment objects and will attemp to create them. Any assignments that already exist are ommited. Any assignments that are invalid will cause the entire operation to fail and return an error.

Here's an example of how you'd create assignments using that API:

Direct Consumer API call	**C# code using Consumer SDK library**

Making a POST request to https://my.tachyon.server/Consumer/PrincipalRoleManagementGroups with following payload:

Payload sent to https://my.tachyon.server/Consumer/Roles/ManagementGroups

```
[
    {
        "PrincipalId": 3,
        "RoleId": 40,
        "ManagementGroupId": 2
    },
    {
        "PrincipalId": 4,
        "RoleId": 37,
        "ManagementGroupId": 10
    },
    {
        "PrincipalId": 4,
        "RoleId": 37,
        "ManagementGroupId": 11
    }
]
```

which will yield a response similar to this one:

Return payload

```
[
    {
        "PrincipalId": 3,
        "RoleId": 40,
        "ManagementGroupId": 2,
        "CreatedTimestampUtc": "2021-10-19T15:10:17.927Z",
        "Principal": {
            "Id": 3,
            "ExternalId": "S-1-5-21-3276326578-728399001-2836074973-1010",
            "PrincipalName": "SomeDomain\\Jane.Doe",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11-30T15:03:21.677Z",
            "ModifiedTimestampUtc": "2016-11-30T15:03:21.677Z",
```

```
            "SystemPrincipal": false,
            "DisplayName": "Stranger",
            "IsGroup": false,
            "NumberOfAssignments": 1
        },
        "Role": {
            "Id": 40,
            "Name": "Group Administrator",
            "Description": "The Group Administrator role allows users
assigned to assign users permission that have been defined as delegatable
in Tachyon, this means that they can be restricted to a Management Group.",
            "CreatedTimestampUtc": "2021-10-18T11:40:29.79Z",
            "ModifiedTimestampUtc": "2021-10-18T11:40:29.79Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 1,
            "HasSecurityPermission": false,
            "AccessType": null
        },
        "ManagementGroup": {
            "Id": 2,
            "Name": "UK",
            "Description": "All computers in UK",
            "Expression": null,
            "TachyonManagementGroupType": 1,
            "TachyonDeviceCount": 10,
            "UsableId": "UK",
            "HashOfMembers": "9670A13F-C5C6-4819-AE77-4D7D8983180C",
            "CreatedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ModifiedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ParentUsableId": "global",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    },
    {
        "PrincipalId": 4,
        "RoleId": 37,
        "ManagementGroupId": 10,
        "CreatedTimestampUtc": "2021-10-19T15:10:17.927Z",
        "Principal": {
            "Id": 4,
            "ExternalId": "S-1-5-21-3276326578-728399001-2836074973-1009",
            "PrincipalName": "SomeDomain\\John.Doe",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11-30T15:10:10.73Z",
            "ModifiedTimestampUtc": "2016-11-30T15:10:10.73Z",
            "SystemPrincipal": false,
            "DisplayName": "TestUser",
            "IsGroup": false,
            "NumberOfAssignments": 2
        },
        "Role": {
            "Id": 37,
            "Name": "Guaranteed State Policy Assigner",
            "Description": "This role will allow assigned users the ability
to assign Policies to any Management Group that is including in their
assignments.",
            "CreatedTimestampUtc": "2021-10-18T11:40:29.507Z",
            "ModifiedTimestampUtc": "2021-10-18T11:40:29.507Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 2,
            "HasSecurityPermission": false,
            "AccessType": null
        },
        "ManagementGroup": {
            "Id": 10,
            "Name": "USWestServers",
            "Description": "All servers in West US",
```

Use PrincipalRoleManagementGroups object inside the Tachyon connector instance.

Assigning Management Groups to a Role

```
var payload = new
List<PrincipalRoleManageme
ntGroup>
{
        new
PrincipalRoleManagementGro
up
        {

PrincipalId = 3,
                RoleId =
40,

ManagementGroupId = 2
        },
        new
PrincipalRoleManagementGro
up
        {

PrincipalId = 4,
                RoleId =
37,

ManagementGroupId = 10
        },
        new
PrincipalRoleManagementGro
up
        {

PrincipalId = 4,
                RoleId =
37,

ManagementGroupId = 11
        }
};

var assignments =
connector.
PrincipalRoleManagementGro
ups.AddAssignments
(payload);
```

"assignments" object will contain the same data you can see in the JSON response on the left.

```
            "Expression": null,
            "TachyonManagementGroupType": 1,
            "TachyonDeviceCount": 10,
            "UsableId": "ServersWestUS",
            "HashOfMembers": "DD46B223-02DE-4839-BFC6-1CA3F9E2EFBA",
            "CreatedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ModifiedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ParentUsableId": "WestUS",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    },
    {
        "PrincipalId": 4,
        "RoleId": 37,
        "ManagementGroupId": 11,
        "CreatedTimestampUtc": "2021-10-19T15:10:17.927Z",
        "Principal": {
            "Id": 4,
            "ExternalId": "S-1-5-21-3276326578-728399001-2836074973-1009",
            "PrincipalName": "SomeDomain\\John.Doe",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11-30T15:10:10.73Z",
            "ModifiedTimestampUtc": "2016-11-30T15:10:10.73Z",
            "SystemPrincipal": false,
            "DisplayName": "TestUser",
            "IsGroup": false,
            "NumberOfAssignments": 2
        },
        "Role": {
            "Id": 37,
            "Name": "Guaranteed State Policy Assigner",
            "Description": "This role will allow assigned users the ability
to assign Policies to any Management Group that is including in their
assignments.",
            "CreatedTimestampUtc": "2021-10-18T11:40:29.507Z",
            "ModifiedTimestampUtc": "2021-10-18T11:40:29.507Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 2,
            "HasSecurityPermission": false,
            "AccessType": null
        },
        "ManagementGroup": {
            "Id": 11,
            "Name": "USWestDesktops",
            "Description": "All desktops in West US",
            "Expression": null,
            "TachyonManagementGroupType": 1,
            "TachyonDeviceCount": 10,
            "UsableId": "DesktopsWestUS",
            "HashOfMembers": "1DECDC63-5387-49AC-BEC0-1CC22AE3B4A6",
            "CreatedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ModifiedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ParentUsableId": "WestUS",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    }
]
```


While this is useful to create a lot of assignments in bulk, but in real life operations you will usually deal with Assignments for a single instance of a particular type of entity, so either a Principal, a Role or a Management Group.

There are separate APIs that allow you to replace all Assignments for either a Principal, Role or a Management Group. Each of those APIs will check if you're not violating rules outlined above and that all assignments are valid.

What is important is to remember that you should provide the API with the Assignments that are supposed to exist AFTER the call has finished, not merely the ones you want to add. These API endpoints will make a diff between what you provided and what exists in the DB and decide what to add and what to remove.

Because of this, if you load Assignments for a Role and you do not have Permission to some of those Assignments, you can still add more Assignments for that Role or remove those Assignments you do have Permission on, as long as list of assignment you send to the API contains all assignments that should exist, that should be kept, so including the ones you do not have Permission to.

That way the API will recognise those entries as already existing and will not attempt to delete them.

As an example, imagine that Role with Id 3 has following assignments:

- Principal Id 1, Management Group 1
- Principal Id 3, Management Group 2
- Principal Id 7, Management Group 1

and you want to have Principal Id 3 have assignment to this Role on Management Group 4, not 2. You have Permission to manipulate assignment for Management Groups 2, 3 and also 4, but do not have Permission to manupulate assignments for Management Group 1.

To do that you'd send following assignments to appropriate endpoint:

- Principal Id 1, Management Group 1
- Principal Id 3, Management Group 4
- Principal Id 7, Management Group 1

Because you have not provided the assignment for Principal Id 3 and Management Group 2, this Assignment will be deleted, which is ok because you have Permission to do it.

And becuase you provided an Assignment for Principapl Id 3 and Management Group 4 and it doesn't exist, it will be created because you have Permission to do it.

Now, if you were to provide only this assignment to the API:

- Principal Id 3, Management Group 4

the system would determine that Assignments for Principal Id 1 / Mamagement Group 1 and Principal Id 7 / Management Group 1 are missing from the list you provided so you wish to delete them. but since you do not have Permission to perform such an operation, you would get an error.

Let us now look at the replacement API endpoints.

First, we have an endpoints that allows us to replace all Assignments for a specific Principal:

Direct Consumer API call	**C# code using Consumer SDK library**
You can identify the Principal either by Id by making a PUT request to https://my.tachyon.server/Consumer/PrincipalRoleManagementGroups/Principal/Id/3 or by Name by making a PUT request to https://my.tachyon.server/Consumer/PrincipalRoleManagementGroups/Principal/Name/U29tZURvbWFpblxKYW5lLkRvZQ== . The name must be base64 encoded. In either case, the payload is the same: **Request payload** `[ { "RoleId": 40, "ManagementGroupId": 2 }, { "RoleId": 37, "ManagementGroupId": 10 }, { "RoleId": 37, "ManagementGroupId": 11 } ]` Either call will yield following result:	

Return payload


```
[
    {
        "PrincipalId": 3,
        "RoleId": 37,
        "ManagementGroupId": 10,
        "CreatedTimestampUtc": "2021-10-20T15:41:54.93Z",
        "Principal": {
            "Id": 3,
            "ExternalId": "S-1-5-21-3276326578-728399001-
2836074973-1010",
            "PrincipalName": "SomeDomain\\Jane.Doe",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11-30T15:03:21.677Z",
            "ModifiedTimestampUtc": "2016-11-30T15:03:21.677Z",
            "SystemPrincipal": false,
            "DisplayName": "Stranger",
            "IsGroup": false,
            "NumberOfAssignments": 3
        },
        "Role": {
            "Id": 37,
            "Name": "Guaranteed State Policy Assigner",
            "Description": "This role will allow assigned users
the ability to assign Policies to any Management Group that is
including in their assignments.",
            "CreatedTimestampUtc": "2021-10-18T11:40:29.507Z",
            "ModifiedTimestampUtc": "2021-10-18T11:40:29.507Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 4,
            "HasSecurityPermission": false,
            "AccessType": null
        },
        "ManagementGroup": {
            "Id": 10,
            "Name": "USWestServers",
            "Description": "All servers in West US",
            "Expression": null,
            "TachyonManagementGroupType": 1,
            "TachyonDeviceCount": 10,
            "UsableId": "ServersWestUS",
            "HashOfMembers": "DD46B223-02DE-4839-BFC6-
1CA3F9E2EFBA",
            "CreatedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ModifiedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ParentUsableId": "WestUS",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    },
    {
        "PrincipalId": 3,
        "RoleId": 37,
        "ManagementGroupId": 11,
        "CreatedTimestampUtc": "2021-10-20T15:41:54.93Z",
        "Principal": {
            "Id": 3,
            "ExternalId": "S-1-5-21-3276326578-728399001-
2836074973-1010",
            "PrincipalName": "SomeDomain\\Jane.Doe",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11-30T15:03:21.677Z",
            "ModifiedTimestampUtc": "2016-11-30T15:03:21.677Z",
            "SystemPrincipal": false,
            "DisplayName": "Stranger",
            "IsGroup": false,
```


Use PrincipalRoleManagementGroups object inside the Tachyon connector instance.

Replacing Assignments for a Principal

```
IEnumerable<PrincipalRoleManagement
Group> assignments;
var payload = new
List<PrincipalRoleManagementGroup>
{
        new
PrincipalRoleManagementGroup
        {
                RoleId = 40,
                ManagementGroupId
= 2
        },
        new
PrincipalRoleManagementGroup
        {
                RoleId = 37,
                ManagementGroupId
= 10
        },
        new
PrincipalRoleManagementGroup
        {
                RoleId = 37,
                ManagementGroupId
= 11
        }
};

// To replace assignments for a
Principal using Principal's Id:
assignments = connector.
PrincipalRoleManagementGroups.
ReplaceAssignmentsForPrincipal(3,
payload);

// to replace assignments for a
Principal using Principal's Name:
assignments = connector.
PrincipalRoleManagementGroups.
ReplaceAssignmentsForPrincipal
("SomeDomain\\Jane.Doe", payload);
```

"assignments" object will contain the same data you can see in the JSON response on the left.

```
            "NumberOfAssignments": 3
        },
        "Role": {
            "Id": 37,
            "Name": "Guaranteed State Policy Assigner",
            "Description": "This role will allow assigned users
the ability to assign Policies to any Management Group that is
including in their assignments.",
            "CreatedTimestampUtc": "2021-10-18T11:40:29.507Z",
            "ModifiedTimestampUtc": "2021-10-18T11:40:29.507Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 4,
            "HasSecurityPermission": false,
            "AccessType": null
        },
        "ManagementGroup": {
            "Id": 11,
            "Name": "USWestDesktops",
            "Description": "All desktops in West US",
            "Expression": null,
            "TachyonManagementGroupType": 1,
            "TachyonDeviceCount": 10,
            "UsableId": "DesktopsWestUS",
            "HashOfMembers": "1DECDC63-5387-49AC-BEC0-
1CC22AE3B4A6",
            "CreatedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ModifiedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ParentUsableId": "WestUS",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    },
    {
        "PrincipalId": 3,
        "RoleId": 40,
        "ManagementGroupId": 2,
        "CreatedTimestampUtc": "2021-10-19T15:10:17.927Z",
        "Principal": {
            "Id": 3,
            "ExternalId": "S-1-5-21-3276326578-728399001-
2836074973-1010",
            "PrincipalName": "SomeDomain\\Jane.Doe",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11-30T15:03:21.677Z",
            "ModifiedTimestampUtc": "2016-11-30T15:03:21.677Z",
            "SystemPrincipal": false,
            "DisplayName": "Stranger",
            "IsGroup": false,
            "NumberOfAssignments": 3
        },
        "Role": {
            "Id": 40,
            "Name": "Group Administrator",
            "Description": "The Group Administrator role allows
users assigned to assign users permission that have been defined
as delegatable in Tachyon, this means that they can be restricted
to a Management Group.",
            "CreatedTimestampUtc": "2021-10-18T11:40:29.79Z",
            "ModifiedTimestampUtc": "2021-10-18T11:40:29.79Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 1,
            "HasSecurityPermission": true,
            "AccessType": null
        },
        "ManagementGroup": {
            "Id": 2,
            "Name": "UK",
            "Description": "All computers in UK",
```

```
            "Expression": null,
            "TachyonManagementGroupType": 1,
            "TachyonDeviceCount": 10,
            "UsableId": "UK",
            "HashOfMembers": "9670A13F-C5C6-4819-AE77-
4D7D8983180C",
            "CreatedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ModifiedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ParentUsableId": "global",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    }
]
```

Next, we have endpoints that allow you to replace all Assignments for a specific Role

Direct Consumer API call	C# code using Consumer SDK library

You can identify the Role either by Id by making a PUT request to https://my.tachyon.server/Consumer/PrincipalRoleManagementGroups/Role/Id/37
 	
or by Name by making a PUT request to https://my.tachyon.server/Consumer/Principal RoleManagementGroups/Role/Name /R3VhcmFudGVlZCBTdGF0ZSBQb2xpY3kgQXNzaWduZXI= . The name must be base64 encoded.
 	
In either case, the payload is the same:
 	
Request payload
 	
`[ { "PrincipalId": 3, "ManagementGroupId": 10 }, { "PrincipalId": 3, "ManagementGroupId": 11 } ]`
 	
Either call will yield following result:
 	
Return payload
 	
`[{ "PrincipalId": 3, "RoleId": 37, "ManagementGroupId": 10, "CreatedTimestampUtc": "2021-10-20T15:41:54.93Z", "Principal": { "Id": 3, "ExternalId": "S-1-5-21-3276326578-728399001-2836074973-1010", "PrincipalName": "SomeDomain\\Jane.Doe", "Email": null, "Enabled": true, "CreatedTimestampUtc": "2016-11-30T15:03:21.677Z", "ModifiedTimestampUtc": "2016-11-30T15:03:21.677Z", "SystemPrincipal": false, "DisplayName": "Stranger", "IsGroup": false,`
	Use PrincipalRoleManagementGroups object inside the Tachyon connector instance.
	
	Replacing Assignments for a Role
	
	`IEnumerable<PrincipalRoleManagementGroup> assignments; var payload = new List<PrincipalRoleManagementGroup> { new PrincipalRoleManagementGroup { PrincipalId = 3, ManagementGroupId = 11 }, new PrincipalRoleManagementGroup { PrincipalId = 3, ManagementGroupId = 10 } }; // To replace assignments for a Role using Role's ID: assignments = connector. PrincipalRoleManagementGroups. ReplaceAssignmentsForRole(37, payload); // To replace assignments for a Role using Role's Name; assignments = connector. PrincipalRoleManagementGroups. ReplaceAssignmentsForRole("Guaranteed State Policy Assigner", payload);`
	
	"assignments" object will contain the same data you can see in the JSON response on the left.

```
            "NumberOfAssignments": 3
        },
        "Role": {
            "Id": 37,
            "Name": "Guaranteed State Policy Assigner",
            "Description": "This role will allow assigned
users the ability to assign Policies to any Management
Group that is including in their assignments.",
            "CreatedTimestampUtc": "2021-10-18T11:40:
29.507Z",
            "ModifiedTimestampUtc": "2021-10-18T11:40:
29.507Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 4,
            "HasSecurityPermission": false,
            "AccessType": null
        },
        "ManagementGroup": {
            "Id": 10,
            "Name": "USWestServers",
            "Description": "All servers in West US",
            "Expression": null,
            "TachyonManagementGroupType": 1,
            "TachyonDeviceCount": 10,
            "UsableId": "ServersWestUS",
            "HashOfMembers": "DD46B223-02DE-4839-BFC6-
1CA3F9E2EFBA",
            "CreatedTimestampUtc": "2021-10-19T15:07:51.11
Z",
            "ModifiedTimestampUtc": "2021-10-19T15:07:
51.11Z",
            "ParentUsableId": "WestUS",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    },
    {
        "PrincipalId": 3,
        "RoleId": 37,
        "ManagementGroupId": 11,
        "CreatedTimestampUtc": "2021-10-20T15:41:54.93Z",
        "Principal": {
            "Id": 3,
            "ExternalId": "S-1-5-21-3276326578-728399001-
2836074973-1010",
            "PrincipalName": "SomeDomain\\Jane.Doe",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11-30T15:03:
21.677Z",
            "ModifiedTimestampUtc": "2016-11-30T15:03:
21.677Z",
            "SystemPrincipal": false,
            "DisplayName": "Stranger",
            "IsGroup": false,
            "NumberOfAssignments": 3
        },
        "Role": {
            "Id": 37,
            "Name": "Guaranteed State Policy Assigner",
            "Description": "This role will allow assigned
users the ability to assign Policies to any Management
Group that is including in their assignments.",
            "CreatedTimestampUtc": "2021-10-18T11:40:
29.507Z",
            "ModifiedTimestampUtc": "2021-10-18T11:40:
29.507Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 4,
```

```
            "HasSecurityPermission": false,
            "AccessType": null
        },
        "ManagementGroup": {
            "Id": 11,
            "Name": "USWestDesktops",
            "Description": "All desktops in West US",
            "Expression": null,
            "TachyonManagementGroupType": 1,
            "TachyonDeviceCount": 10,
            "UsableId": "DesktopsWestUS",
            "HashOfMembers": "1DECDC63-5387-49AC-BEC0-
1CC22AE3B4A6",
            "CreatedTimestampUtc": "2021-10-19T15:07:51.11
Z",
            "ModifiedTimestampUtc": "2021-10-19T15:07:
51.11Z",
            "ParentUsableId": "WestUS",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    }
]
```


Finally, we have endpoints that allow you to replace all Assignments for a specific Management Group.

Direct Consumer API call	C# code using Consumer SDK library

You can identify the Role either by Id by making a PUT request to https://my.tachyon.server/Consumer/Principal RoleManagementGroups/ManagementGroup/Id/9

or by Name by making a PUT request to https://my.tachyon.server/Consumer/Principal RoleManagementGroups/ManagementGroup/UsableId/USEastDesktops

In either case, the payload is the same:

Request payload

```
[
        {
                "PrincipalId": 3,
                "RoleId": 40
        },
        {
                "PrincipalId": 4,
                "RoleId": 37
        },
        {
                "PrincipalId": 3,
                "RoleId": 37
        }
]
```

Either call will yield the same result:

Return payload


```
[
    {
        "IsInherited": false,
        "PrincipalId": 3,
        "RoleId": 37,
        "ManagementGroupId": 9,
        "CreatedTimestampUtc": "2021-10-21T12:29:51.523Z",
        "Principal": {
            "Id": 3,
```

```
            "ExternalId": "S-1-5-21-3276326578-728399001-
2836074973-1010",
            "PrincipalName": "SomeDomain\\Jane.Doe",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11-30T15:03:21.677Z",
            "ModifiedTimestampUtc": "2016-11-30T15:03:21.677Z",
            "SystemPrincipal": false,
            "DisplayName": "Stranger",
            "IsGroup": false,
            "NumberOfAssignments": 5
        },
        "Role": {
            "Id": 37,
            "Name": "Guaranteed State Policy Assigner",
            "Description": "This role will allow assigned users
the ability to assign Policies to any Management Group that is
including in their assignments.",
            "CreatedTimestampUtc": "2021-10-18T11:40:29.507Z",
            "ModifiedTimestampUtc": "2021-10-18T11:40:29.507Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 4,
            "HasSecurityPermission": false,
            "AccessType": "ReadWrite"
        },
        "ManagementGroup": {
            "Id": 9,
            "Name": "USEastDesktops",
            "Description": "All desktops in East US",
            "Expression": null,
            "TachyonManagementGroupType": 1,
            "TachyonDeviceCount": 10,
            "UsableId": "DesktopsEastUS",
            "HashOfMembers": "89CABF45-7C93-4DA6-8C9A-
8234557B44E4",
            "CreatedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ModifiedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ParentUsableId": "EastUS",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    },
    {
        "IsInherited": false,
        "PrincipalId": 3,
        "RoleId": 40,
        "ManagementGroupId": 9,
        "CreatedTimestampUtc": "2021-10-21T12:29:51.53Z",
        "Principal": {
            "Id": 3,
            "ExternalId": "S-1-5-21-3276326578-728399001-
2836074973-1010",
            "PrincipalName": "SomeDomain\\Jane.Doe",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11-30T15:03:21.677Z",
            "ModifiedTimestampUtc": "2016-11-30T15:03:21.677Z",
            "SystemPrincipal": false,
            "DisplayName": "Stranger",
            "IsGroup": false,
            "NumberOfAssignments": 5
        },
        "Role": {
            "Id": 40,
            "Name": "Group Administrator",
            "Description": "The Group Administrator role allows
users assigned to assign users permission that have been
defined as delegatable in Tachyon, this means that they can be
restricted to a Management Group.",
            "CreatedTimestampUtc": "2021-10-18T11:40:29.79Z",
```


Use PrincipalRoleManagementGroups object inside the Tachyon connector instance.

Replacing all Assignments for a Management Group

```
IEnumerable<PrincipalRoleManagementGro
up> assignments;
var payload = new
List<PrincipalRoleManagementGroup>
{
        new
PrincipalRoleManagementGroup
        {
                PrincipalId = 3,
                RoleId = 40
        },
        new
PrincipalRoleManagementGroup
        {
                PrincipalId = 4,
                RoleId = 37
        },
        new
PrincipalRoleManagementGroup
        {
                PrincipalId = 3,
                RoleId = 37
        }
};

// To replace assignments for a
Management Group using Management
Group Id:
assignments = connector.
PrincipalRoleManagementGroups.
ReplaceAssignmentsForManagementGroup
(9, payload).ReceivedObject;

// To replace assignments for a
Management Groups using Management
Group's UsableId:
assignments = connector.
PrincipalRoleManagementGroups.
ReplaceAssignmentsForManagementGroup
("USEastDesktops", payload).
ReceivedObject;
```

"assignments" object will contain the same data you can see in the JSON response on the left.

```
            "ModifiedTimestampUtc": "2021-10-18T11:40:29.79Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 2,
            "HasSecurityPermission": true,
            "AccessType": "ReadWrite"
        },
        "ManagementGroup": {
            "Id": 9,
            "Name": "USEastDesktops",
            "Description": "All desktops in East US",
            "Expression": null,
            "TachyonManagementGroupType": 1,
            "TachyonDeviceCount": 10,
            "UsableId": "DesktopsEastUS",
            "HashOfMembers": "89CABF45-7C93-4DA6-8C9A-
8234557B44E4",
            "CreatedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ModifiedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ParentUsableId": "EastUS",
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    },
    {
        "IsInherited": false,
        "PrincipalId": 4,
        "RoleId": 37,
        "ManagementGroupId": 9,
        "CreatedTimestampUtc": "2021-10-21T12:29:51.53Z",
        "Principal": {
            "Id": 4,
            "ExternalId": "S-1-5-21-3276326578-728399001-
2836074973-1009",
            "PrincipalName": "SomeDomain\\John.Doe",
            "Email": null,
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11-30T15:10:10.73Z",
            "ModifiedTimestampUtc": "2016-11-30T15:10:10.73Z",
            "SystemPrincipal": false,
            "DisplayName": "TestUser",
            "IsGroup": false,
            "NumberOfAssignments": 1
        },
        "Role": {
            "Id": 37,
            "Name": "Guaranteed State Policy Assigner",
            "Description": "This role will allow assigned users
the ability to assign Policies to any Management Group that is
including in their assignments.",
            "CreatedTimestampUtc": "2021-10-18T11:40:29.507Z",
            "ModifiedTimestampUtc": "2021-10-18T11:40:29.507Z",
            "SystemRole": true,
            "CanBeDelegated": true,
            "NumberOfAssignments": 4,
            "HasSecurityPermission": false,
            "AccessType": "ReadWrite"
        },
        "ManagementGroup": {
            "Id": 9,
            "Name": "USEastDesktops",
            "Description": "All desktops in East US",
            "Expression": null,
            "TachyonManagementGroupType": 1,
            "TachyonDeviceCount": 10,
            "UsableId": "DesktopsEastUS",
            "HashOfMembers": "89CABF45-7C93-4DA6-8C9A-
8234557B44E4",
            "CreatedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ModifiedTimestampUtc": "2021-10-19T15:07:51.11Z",
            "ParentUsableId": "EastUS",
```

```
            "AccessType": "Unknown"
        },
        "AccessType": "ReadWrite"
    }
]
```

Please note that the replacement APIs take the Id, Name or UsableId used to identify the object from the URI, not from the payload.

This means that if you were to, for instance, replace Assignments for a Principapal and in the payload provide principal Ids, they will be ignored. Likewise for Role Id when replacing Assignments for a Role and Management Group Id when replacing Assignment for a Management Group.

Deleting Assignments

Assignments can also be deleted, providing you have the Permission to do so, either in bulk:

Direct Consumer API call	C# code using Consumer SDK library
Make a DELETE request to https://my.tachyon.server/Consumer /PrincipalRoleManagementGroups with following payload: **Request payload** `[ { "PrincipalId": 3, "RoleId": 40, "ManagementGroupId": 2 }, { "PrincipalId": 4, "RoleId": 37, "ManagementGroupId": 10 }, { "PrincipalId": 4, "RoleId": 37, "ManagementGroupId": 11 } ]` No data is returned by this call.	Use PrincipalRoleManagementGroups object inside the Tachyon connector instance. **Deleting Assignments in bulk** `var payload = new List<PrincipalRoleManagementGroup> { new PrincipalRoleManagementGroup { PrincipalId = 3, RoleId = 40, ManagementGroupId = 2 }, new PrincipalRoleManagementGroup { PrincipalId = 4, RoleId = 37, ManagementGroupId = 10 }, new PrincipalRoleManagementGroup { PrincipalId = 4, RoleId = 37, ManagementGroupId = 11 } }; connector.PrincipalRoleManagementGroups. DeleteAssignments(payload);`

or individually:

Direct Consumer API call	C# code using Consumer SDK library
Make a DELETE request to https://my.tachyon.server/Consumer /PrincipalRoleManagementGroups/PrincipalId/3/RoleId/40/ManagementGroupId/2 without any payload. No data is returned by this call.	Use PrincipalRoleManagementGroups object inside the Tachyon connector instance. **Deleting a single assignment** `connector. PrincipalRoleManagementGroups. DeleteAssignment(3, 40, 2);`

Configuring Securable Types and Applicable Operations

We have already seen how to retrieve Securable Types and now we'll look at how we can create and modify them.

All you need to create a Securable Type is a name, which must be unique.

In order to delete a Securable Type, you have to first delete all Permissions that use that Securable Types and all Applicable Operations linked to the Securable Type.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request to https://my.tachyon.server/Consumer/SecurableTypes
 	
Paylaod sent to https://my.tachyon.server/Consumer/SecurableTypes
 	
`{`
`  "Name": "Custom securable type"`
`}`
 	
will yield following response:
 	
Return payload
 	
`{`
`  "Id": 26,`
`  "Name": "Custom securable type",`
`  "CreatedTimestampUtc": "2020-01-24T14:36:23.8756464Z",`
`  "ModifiedTimestampUtc": "2020-01-24T14:36:23.8756464Z",`
`  "Operations": null`
`}`
	Use SecurableTypes object inside the Tachyon connector instance.
	
	Creating a new Securable Type
	
	`var payload = new SecurableType`
	`{`
	`  Name = "Custom securable type"`
	`};`
	`return connector.SecurableTypes.Add(payload);`
	
	"securableType" object will contain the same data you can see in the JSON response on the left.

If you wish to update a securable type you'll have to provide its Id:

Direct Consumer API call	C# code using Consumer SDK library

Making a PUT request to https://my.tachyon.server/Consumer/SecurableTypes
 	
Paylaod sent to https://my.tachyon.server/Consumer/SecurableTypes
 	
`{ "Id": 26, "Name": "Modified securable type" }`
 	
will yield following response:
 	
Return payload
 	
`{ "Id": 26, "Name": "Modified securable type", "CreatedTimestampUtc": "2020-01-24T14:36:23.8756464Z", "ModifiedTimestampUtc": "2020-01-24T14:38:24.8756464Z", "Operations": null }`
	Use SecurableTypes object inside the Tachyon connector instance.
	
	Modifying an existing Securable Type
	
	`var payload = new SecurableType { Id = 26, Name = "Modified securable type" }; return connector.SecurableTypes.Update(payload);`
	
	"securableType" object will contain the same data you can see in the JSON response on the left.

and to delete this Sedurable Type:

Direct Consumer API call	C# code using Consumer SDK library
Make a DELETE request to https://my.tachyon.server/Consumer/SecurableTypes/26 This call will not yield any response.	Use SecurableTypes object inside the Tachyon connector instance. **Deleting a Securable Type** `connector.SecurableTypes.Delete(26);`

It is worth noting that Securable Types are created "empty" and that Applicable Operation have to be created separately.

Dealing with Operations

Applicable Operations exist only in the context of a Securable Type. While Applicable Operations must be unique within a single Securable Type, they do not have to be unique between Securable Types.

To create an Operation you have to provide either Id or Name (but not both) of Securable Type for which the Operation should be created. In the example below we'll use the Securable Type we've just created, which has the Id of 26.

Direct Consumer API call	C# code using Consumer SDK library

Make a POST request to https://my.tachyon.server/Consumer/ ApplicableOperations to use Securable Type Id

Payload sent to https://my.tachyon.server/Consumer/ApplicableOperations

```
{
  "OperationName": "View",
  "SecurableTypeId": 26
}
```

or with this payload to use Securable Type Name:

Payload sent to https://my.tachyon.server/Consumer/ApplicableOperations

```
{
  "OperationName": "View",
  "SecurableTypeName": "Modified securable type"
}
```

Both will yield following response:

Return payload

```
{
    "Id": 63,
    "OperationName": "View",
    "SecurableTypeId": 26,
    "SecurableTypeName": "Modified securable type"
}
```

Use ApplicableOperations object inside the Tachyon connector instance.

To use Securable Type Id:

Creating a new Applicable Operation

```
var payload = new ApplicableOperation
{
    SecurableTypeId = 26,
    OperationName = "View"
};
operation = connector.ApplicableOperations.Add
(payload);
```

To use Securable Type Name:

Creating a new Applicable Operation

```
var payload = new ApplicableOperation
{
        SecurableTypeName = "Modified
securable type",
        OperationName = "View"
};
operation = connector.ApplicableOperations.Add
(payload);
```

"operation" object will contain the same data you can see in the JSON response on the left.

To delete an Operation you just need its Id.

Direct Consumer API call	C# code using Consumer SDK library
Make a DELETE request to https://my.tachyon.server/Consumer/ApplicableOperations/63	Use ApplicableOperations object inside the Tachyon connector instance. **Deleting an Applicable Operation** `connector.ApplicableOperations.Delete(63);`