Set up Principals, Roles and Permissions

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.

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.

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

All permissions lead 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.

As of version 5.0 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".

Example how to read permissions

Marc (**Principal**) through his Global Questioners (**Role**) has Questioner (**Applicable Operation**) permission on Instructions (**Securable type**).

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.

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:
 	
Return payload
 	
[
    {
        "Id": 11,
        "Name": "1E Client Deployment Administrators",
        "Description": "This role can create, view and cancel 1E Client deployment jobs",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.28Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.28Z",
        "SystemRole": true
    },
    {
        "Id": 12,
        "Name": "1E Client Installer Administrators",
        "Description": "This role can upload, delete and view 1E Client installers",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.307Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.307Z",
        "SystemRole": true
    },
    {
        "Id": 23,
        "Name": "Applications Administrators",
        "Description": "This role can upload and delete Applications",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.693Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.693Z",
        "SystemRole": true
    },
    {
        "Id": 17,
        "Name": "Component Viewers",
        "Description": "This role can view components",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "SystemRole": true
    },
    {
        "Id": 14,
        "Name": "Connector Administrators",
        "Description": "This role can create, update, delete and view connectors; and test them",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "SystemRole": true
    },
    {
        "Id": 10,
        "Name": "Consumer Administrators",
        "Description": "This role can manage Consumers that use the Tachyon platform",
        "CreatedTimestampUtc": "2019-11-07T13:14:55.523Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:55.523Z",
        "SystemRole": true
    },
    {
        "Id": 9,
        "Name": "Custom Properties Administrators",
        "Description": "This role can add, edit or delete custom properties",
        "CreatedTimestampUtc": "2019-11-07T13:14:55.133Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:55.133Z",
        "SystemRole": true
    },
    {
        "Id": 28,
	Use Roles object inside the Tachyon connector instance.
	
	Retrieving all Roles
	
	`roles = connector.Roles.GetAll();`
	
	"roles" object will contain the same data you can see in the JSON response on the left.

```
        "Name": "Event Subscription Administrators",
        "Description": "This role has full control over the Event Subscription
configuration",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.907Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.907Z",
        "SystemRole": true
    },
    {
        "Id": 29,
        "Name": "Event Subscription Viewers",
        "Description": "This role can view the Event Subscription configuration
and reports",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.907Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.907Z",
        "SystemRole": true
    },
    {
        "Id": 7,
        "Name": "Global Actioners",
        "Description": "This role can ask questions, view responses and send
actions for all instruction sets",
        "CreatedTimestampUtc": "2019-11-07T13:14:54.957Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.12Z",
        "SystemRole": true
    },
    {
        "Id": 1,
        "Name": "Global Administrators",
        "Description": "Has the combined rights of all the other system roles",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.757Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:52.757Z",
        "SystemRole": true
    },
    {
        "Id": 5,
        "Name": "Global Approvers",
        "Description": "This role can approve actions for all instruction sets,
for anyone other than self",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.757Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.117Z",
        "SystemRole": true
    },
    {
        "Id": 6,
        "Name": "Global Questioners",
        "Description": "This role can ask questions and view responses for all
instruction sets",
        "CreatedTimestampUtc": "2019-11-07T13:14:54.957Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.12Z",
        "SystemRole": true
    },
    {
        "Id": 8,
        "Name": "Global Viewers",
        "Description": "This role can view instructions and responses for all
instruction sets",
        "CreatedTimestampUtc": "2019-11-07T13:14:54.957Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.12Z",
        "SystemRole": true
    },
    {
        "Id": 24,
        "Name": "Guaranteed State Administrators",
        "Description": "This role has full control over the Guaranteed State
configuration",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.717Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.717Z",
        "SystemRole": true
    },
    {
        "Id": 25,
```

```
        "Name": "Guaranteed State Viewers",
        "Description": "This role can view the Guaranteed State configuration and
reports",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.717Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.717Z",
        "SystemRole": true
    },
    {
        "Id": 4,
        "Name": "Infrastructure Administrators",
        "Description": "This role can view system status instrumentation and
system information",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.757Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.493Z",
        "SystemRole": true
    },
    {
        "Id": 2,
        "Name": "Instruction Set Administrators",
        "Description": "This role can add product packs; add, modify and delete
instruction sets; and delete instruction definitions",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.757Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.117Z",
        "SystemRole": true
    },
    {
        "Id": 19,
        "Name": "Inventory Administrators",
        "Description": "This role can create, update, delete and view inventory
repositories; populate and archive them",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "SystemRole": true
    },
    {
        "Id": 20,
        "Name": "Inventory Viewers",
        "Description": "This role can view inventory repositories",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "SystemRole": true
    },
    {
        "Id": 16,
        "Name": "Log Viewers",
        "Description": "This role can view process, synchronization and
infrastructure logs",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "SystemRole": true
    },
    {
        "Id": 13,
        "Name": "Management Group Administrators",
        "Description": "This role can create, delete and update management groups;
and initiate synchronization of management groups",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.4Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.4Z",
        "SystemRole": true
    },
    {
        "Id": 21,
        "Name": "Management Group Sync Initiators",
        "Description": "This role can view management groups and initiate
synchronization of management groups",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "SystemRole": true
    },
    {
        "Id": 30,
```

```
        "Name": "MySet Viewers",
        "Description": null,
        "CreatedTimestampUtc": "2020-01-02T12:02:12.01Z",
        "ModifiedTimestampUtc": "2020-01-02T12:02:12.01Z",
        "SystemRole": false
    },
    {
        "Id": 26,
        "Name": "Patch Success Viewers",
        "Description": "This role can view the Patch Success dashboards",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.73Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.73Z",
        "SystemRole": true
    },
    {
        "Id": 3,
        "Name": "Permissions Administrators",
        "Description": "This role can add or remove users; view all roles; add,
modify  and delete custom roles; assign roles to any instruction sets and define
their permissions; and view the admin log",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.757Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.117Z",
        "SystemRole": true
    },
    {
        "Id": 22,
        "Name": "Permissions Readers",
        "Description": "This role can view all roles",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "SystemRole": true
    },
    {
        "Id": 18,
        "Name": "Provider Configuration Administrators",
        "Description": "This role can update, delete and view provider
configurations",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "SystemRole": true
    },
    {
        "Id": 15,
        "Name": "Schedule Administrators",
        "Description": "This role can create, update, delete and view schedules;
and can view schedule history",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.67Z",
        "SystemRole": true
    },
    {
        "Id": 27,
        "Name": "ServiceNow ITSM Connect",
        "Description": "This role is used with the ServiceNow 1E ITSM Connect
application. Once the 1E ITSM Connect application is installed in ServiceNow this
role will be used to grant permissions to the specified Service Account user.",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.743Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.743Z",
        "SystemRole": false
    }
]
```


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": 2,
    "Items": [
        {
            "SystemRole": false,
            "AssignedManagementGroupCount": 0,
            "HasAllDevicesManagementGroupAssigned": false,
            "AssignedPrincipalCount": 1,
            "Id": 30,
            "Name": "MySet Viewers",
            "Description": null,
            "CreatedTimestampUtc": "2020-01-02T12:02:12.01Z",
            "ModifiedTimestampUtc": "2020-01-02T12:02:12.01Z"
        },
        {
            "SystemRole": false,
            "AssignedManagementGroupCount": 0,
            "HasAllDevicesManagementGroupAssigned": false,
            "AssignedPrincipalCount": 0,
            "Id": 27,
            "Name": "ServiceNow ITSM Connect",
            "Description": "This role is used with the ServiceNow 1E ITSM
Connect application. Once the 1E ITSM Connect application is installed in
ServiceNow this role will be used to grant permissions to the specified Service
Account user.",
            "CreatedTimestampUtc": "2019-11-07T13:14:56.743Z",
            "ModifiedTimestampUtc": "2019-11-07T13:14:56.743Z"
        }
    ]
}
```

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.

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": "Global Administrators",
    "Description": "Has the combined rights of all the
other system roles",
    "CreatedTimestampUtc": "2019-11-07T13:14:52.757Z",
    "ModifiedTimestampUtc": "2019-11-07T13:14:52.757Z",
    "SystemRole": 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": 2,
        "CreatedTimestampUtc": "2019-11-07T13:14:55.433Z",
        "Role": {
            "SystemRole": true,
            "AssignedManagementGroupCount": 0,
            "HasAllDevicesManagementGroupAssigned": false,
            "AssignedPrincipalCount": 1,
            "Id": 2,
            "Name": "Instruction Set Administrators",
            "Description": "This role can add product packs; add, modify and
delete instruction sets; and delete instruction definitions",
            "CreatedTimestampUtc": "2019-11-07T13:14:52.757Z",
            "ModifiedTimestampUtc": "2019-11-07T13:14:56.117Z"
        },
        "Principal": null
    },
    {
        "PrincipalId": 3,
        "RoleId": 3,
        "CreatedTimestampUtc": "2019-11-07T13:14:54.963Z",
        "Role": {
            "SystemRole": true,
            "AssignedManagementGroupCount": 0,
            "HasAllDevicesManagementGroupAssigned": false,
            "AssignedPrincipalCount": 1,
            "Id": 3,
            "Name": "Permissions Administrators",
            "Description": "This role can add or remove users; view all roles;
add, modify  and delete custom roles; assign roles to any instruction sets and
define their permissions; and view the admin log",
            "CreatedTimestampUtc": "2019-11-07T13:14:52.757Z",
            "ModifiedTimestampUtc": "2019-11-07T13:14:56.117Z"
        },
        "Principal": null
    },
    {
        "PrincipalId": 3,
        "RoleId": 5,
        "CreatedTimestampUtc": "2019-11-07T13:15:01.533Z",
        "Role": {
            "SystemRole": true,
            "AssignedManagementGroupCount": 0,
            "HasAllDevicesManagementGroupAssigned": false,
            "AssignedPrincipalCount": 1,
            "Id": 5,
            "Name": "Global Approvers",
            "Description": "This role can approve actions for all instruction
sets, for anyone other than self",
            "CreatedTimestampUtc": "2019-11-07T13:14:52.757Z",
            "ModifiedTimestampUtc": "2019-11-07T13:14:56.117Z"
        },
        "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

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

Return payload


```
[
    {
        "Id": 1,
        "Name": "InstructionSet",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.75
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.113
Z",
        "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"
            }
        ]
    },
    {
        "Id": 2,
        "Name": "Security",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.75
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:52.75
Z",
        "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"
            }
        ]
    },
```


C# code using Consumer SDK library

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.

```
    {
        "Id": 3,
        "Name": "InstructionSetManagement",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.75
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.113
Z",
        "Operations": [
            {
                "Id": 9,
                "OperationName": "Add",
                "SecurableTypeId": 3,
                "SecurableTypeName":
"InstructionSetManagement"
            },
            {
                "Id": 8,
                "OperationName": "Delete",
                "SecurableTypeId": 3,
                "SecurableTypeName":
"InstructionSetManagement"
            },
            {
                "Id": 10,
                "OperationName": "Read",
                "SecurableTypeId": 3,
                "SecurableTypeName":
"InstructionSetManagement"
            }
        ]
    },
    {
        "Id": 4,
        "Name": "Instrumentation",
        "CreatedTimestampUtc": "2019-11-07T13:14:52.75
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:52.75
Z",
        "Operations": [
            {
                "Id": 11,
                "OperationName": "Read",
                "SecurableTypeId": 4,
                "SecurableTypeName": "Instrumentation"
            }
        ]
    },
    {
        "Id": 5,
        "Name": "CustomProperty",
        "CreatedTimestampUtc": "2019-11-07T13:14:55.13
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:55.13
Z",
        "Operations": [
            {
                "Id": 12,
                "OperationName": "Read",
                "SecurableTypeId": 5,
                "SecurableTypeName": "CustomProperty"
            },
            {
                "Id": 13,
                "OperationName": "Write",
                "SecurableTypeId": 5,
                "SecurableTypeName": "CustomProperty"
            }
        ]
    },
    {
        "Id": 6,
```

```
        "Name": "Consumer",
        "CreatedTimestampUtc": "2019-11-07T13:14:55.523
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:55.523
Z",
        "Operations": [
            {
                "Id": 14,
                "OperationName": "Read",
                "SecurableTypeId": 6,
                "SecurableTypeName": "Consumer"
            },
            {
                "Id": 15,
                "OperationName": "Write",
                "SecurableTypeId": 6,
                "SecurableTypeName": "Consumer"
            }
        ]
    },
    {
        "Id": 7,
        "Name": "AgentDeployment",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.277
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.277
Z",
        "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"
            }
        ]
    },
    {
        "Id": 8,
        "Name": "AgentInstallerManagement",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.307
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.307
Z",
        "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"
            }
        ]
    },
    {
        "Id": 9,
        "Name": "ManagementGroup",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.397
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.397
Z",
        "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"
            }
        ]
    },
    {
        "Id": 10,
        "Name": "Connector",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.663
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.663
Z",
        "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"
            }
        ]
    },
    {
        "Id": 11,
        "Name": "Schedule",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.663
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.663
Z",
        "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"
            }
        ]
    },
    {
        "Id": 12,
        "Name": "ProcessLog",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.663
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.663
Z",
        "Operations": [
            {
                "Id": 33,
                "OperationName": "Read",
                "SecurableTypeId": 12,
                "SecurableTypeName": "ProcessLog"
            }
        ]
    },
    {
        "Id": 13,
        "Name": "SynchronizationLog",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.663
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.663
Z",
        "Operations": [
            {
                "Id": 34,
                "OperationName": "Read",
                "SecurableTypeId": 13,
                "SecurableTypeName":
"SynchronizationLog"
            }
        ]
    },
    {
        "Id": 14,
        "Name": "Component",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.663
Z",
```

```
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.663
Z",
        "Operations": [
            {
                "Id": 35,
                "OperationName": "Read",
                "SecurableTypeId": 14,
                "SecurableTypeName": "Component"
            }
        ]
    },
    {
        "Id": 15,
        "Name": "ProviderConfiguration",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.663
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.663
Z",
        "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"
            }
        ]
    },
    {
        "Id": 16,
        "Name": "InfrastructureLog",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.663
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.663
Z",
        "Operations": [
            {
                "Id": 39,
                "OperationName": "Read",
                "SecurableTypeId": 16,
                "SecurableTypeName": "InfrastructureLog"
            }
        ]
    },
    {
        "Id": 17,
        "Name": "Repository.Inventory",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.663
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.663
Z",
        "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"
            }
        ]
    },
    {
        "Id": 18,
        "Name": "Application",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.693
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.693
Z",
        "Operations": [
            {
                "Id": 46,
                "OperationName": "Delete",
                "SecurableTypeId": 18,
                "SecurableTypeName": "Application"
            },
            {
                "Id": 45,
                "OperationName": "Write",
                "SecurableTypeId": 18,
                "SecurableTypeName": "Application"
            }
        ]
    },
    {
        "Id": 19,
        "Name": "GuaranteedState",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.713
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.713
Z",
        "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"
            }
        ]
    },
    {
        "Id": 20,
        "Name": "Repository.Patch",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.73
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.73
Z",
        "Operations": [
            {
                "Id": 50,
                "OperationName": "Read",
                "SecurableTypeId": 20,
                "SecurableTypeName": "Repository.Patch"
            }
        ]
    },
    {
        "Id": 21,
        "Name": "Repository.BI",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.737
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.737
Z",
        "Operations": [
            {
                "Id": 52,
                "OperationName": "Populate",
                "SecurableTypeId": 21,
                "SecurableTypeName": "Repository.BI"
            },
            {
                "Id": 51,
                "OperationName": "Read",
                "SecurableTypeId": 21,
                "SecurableTypeName": "Repository.BI"
            }
        ]
    },
    {
        "Id": 22,
        "Name": "EventSubscription",
        "CreatedTimestampUtc": "2019-11-07T13:14:56.907
Z",
        "ModifiedTimestampUtc": "2019-11-07T13:14:56.907
Z",
        "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"
            }
        ]
    },
    {
        "Id": 24,
        "Name": "Repository.AppMigration",
        "CreatedTimestampUtc": "2019-11-26T09:34:40.923
Z",
        "ModifiedTimestampUtc": "2019-11-26T09:34:40.923
Z",
        "Operations": [
            {
                "Id": 61,
                "OperationName": "Archive",
                "SecurableTypeId": 24,
                "SecurableTypeName": "Repository.
AppMigration"
            },
            {
                "Id": 60,
                "OperationName": "Execute",
                "SecurableTypeId": 24,
                "SecurableTypeName": "Repository.
AppMigration"
            },
            {
                "Id": 58,
                "OperationName": "Read",
                "SecurableTypeId": 24,
                "SecurableTypeName": "Repository.
AppMigration"
            },
            {
                "Id": 62,
                "OperationName": "Whatever",
                "SecurableTypeId": 24,
                "SecurableTypeName": "Repository.
AppMigration"
            },
            {
                "Id": 59,
                "OperationName": "Write",
                "SecurableTypeId": 24,
                "SecurableTypeName": "Repository.
AppMigration"
            }
        ]
    }
]
```


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",
    "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.

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:	

Return payload

```
[{
    "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"
    }]
},
{
    "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"
    }]
},
{
    "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"
    }]
},
{
    "SecurableId": null,
    "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"
    }]
},
{
    "SecurableId": null,
    "SecurableName": null,
    "SecurableTypeId": 16,
    "SecurableTypeName": "InfrastructureLog",
    "RoleId": 16,
    "RoleName": "Log Viewers",
    "Allowed": true,
    "Operations": [{
        "PermissionId": 63,
```


Use Permissions object inside the Tachyon connector instance.

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.

```
        "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: **Return payload**	Use Permissions object inside the Tachyon connector instance. **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.

```
[
    {
        "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"
            },
            {
                "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"
            }
        ]
    }
]
```

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
 	
[
{
"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.

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
        }
]
```

Tachyon versions 5.0 and newer

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
       }]
   }],
   "ManagementGroupIds": [10,11,12]
}
```

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",
                    "ModifiedTimestampUtc": "2020-01-
```

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 }
            }
        }
    },
    ManagementGroupIds = new List<int> {
10, 11, 12 }
};

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

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

```
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"
                }
            ]
        }
    ],
    "ManagementGroups": [
        {
            "Id": 10,
            "Name": "Management group number 9 from
repository Default Inventory",
            "Description": "This is the description for
Management group number 9 from repository Default
Inventory",
            "Expression": "",
            "Type": 2,
            "Count": 1,
            "UsableId": "9",
            "HashOfMembers":
"4CFD42DAA10FD0FDE0B4E6BE1116155AC6180EE4E9ADF543F39E97D9B
8C51B25",
            "CreatedTimestampUtc": "2019-12-02T12:01:
21.677Z",
            "ModifiedTimestampUtc": "2020-01-08T16:31:
03.343Z"
```

```
        },
        {
            "Id": 11,
            "Name": "Management group number 10 from
repository Default Inventory",
            "Description": "This is the description for
Management group number 10 from repository Default
Inventory",
            "Expression": "",
            "Type": 2,
            "Count": 10,
            "UsableId": "10",
            "HashOfMembers":
"F51DE677220883E4A684DBB7D965110DE74517D714B3459DA4A98C3B5
6C58D1C",
            "CreatedTimestampUtc": "2019-12-02T12:01:21.68
Z",
            "ModifiedTimestampUtc": "2020-01-08T16:31:
03.43Z"
        },
        {
            "Id": 12,
            "Name": "Management group number 11 from
repository Default Inventory",
            "Description": "This is the description for
Management group number 11 from repository Default
Inventory",
            "Expression": "",
            "Type": 2,
            "Count": 27,
            "UsableId": "11",
            "HashOfMembers":
"5D25CC2DEBF1D76B81602884623FDE878FE8EF57568A6F97A9F04242F
EF2655D",
            "CreatedTimestampUtc": "2019-12-02T12:01:21.68
Z",
            "ModifiedTimestampUtc": "2020-01-08T16:31:
03.517Z"
        }
    ],
    "Id": 32,
    "Name": "Complete Role 1",
    "Description": "This is a test role",
    "CreatedTimestampUtc": "2020-01-22T07:18:57.7082854Z",
    "ModifiedTimestampUtc": "2020-01-22T07:18:57.7082854
Z",
    "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});`

Assigning and unassigning Principals to Roles

As we've seen above when we covered retrieving Principals and Roles, the association between Principals and Roles can be examined from either end: which Principals are assigned to a given Role or which Roles given Principal has assigned.

Looking at the issue from the perspective of a Principal, you can assign or unassign Roles(s) to a Principal, or replace any Roles that Principal has with a given set of Roles.

First, let's pick a Principal that we will assign Roles to. In the example below I'll use "SomeDomain\\Jane.Doe", which has the Id of 3 and I'll assign 4 Roles to that Principal: Instruction Set Administrators, Permissions Administrators, Infrastructure Administrators, Global Questioners which have respective Ids of 2, 3, 4 and 6.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request to https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3 with following payload: **Request payload sent to https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3** `[2,3,4,6]` will yield following response: **Return payload**	Use PrincipalRoles object inside the Tachyon connector instance. **Adding multiple Roles to a Principal** `roles = connector. PrincipalRoles. AddRolesToPrincipal(3, new List<int> { 2, 3, 4, 6 });` "roles" object will contain the same data you can see in the JSON response on the left.


```
[
    {
        "PrincipalId": 3,
        "RoleId": 2,
        "CreatedTimestampUtc": "2020-01-23T08:09:43.31Z",
        "Role": {
            "AssignedManagementGroupCount": 0,
            "HasAllDevicesManagementGroupAssigned": false,
            "AssignedPrincipalCount": 2,
            "Id": 2,
            "Name": "Instruction Set Administrators",
            "Description": "This role can add product packs; add, modify
and delete instruction sets; and delete instruction definitions",
            "CreatedTimestampUtc": "2019-11-07T13:14:52.757Z",
            "ModifiedTimestampUtc": "2019-11-07T13:14:56.117Z",
            "SystemRole": true
        },
        "Principal": null
    },
```

```
    {
        "PrincipalId": 3,
        "RoleId": 3,
        "CreatedTimestampUtc": "2020-01-23T08:09:43.31Z",
        "Role": {
            "AssignedManagementGroupCount": 0,
            "HasAllDevicesManagementGroupAssigned": false,
            "AssignedPrincipalCount": 2,
            "Id": 3,
            "Name": "Permissions Administrators",
            "Description": "This role can add or remove users; view all
roles; add, modify  and delete custom roles; assign roles to any
instruction sets and define their permissions; and view the admin log",
            "CreatedTimestampUtc": "2019-11-07T13:14:52.757Z",
            "ModifiedTimestampUtc": "2019-11-07T13:14:56.117Z",
            "SystemRole": true
        },
        "Principal": null
    },
    {
        "PrincipalId": 3,
        "RoleId": 4,
        "CreatedTimestampUtc": "2020-01-23T08:09:43.31Z",
        "Role": {
            "AssignedManagementGroupCount": 0,
            "HasAllDevicesManagementGroupAssigned": false,
            "AssignedPrincipalCount": 1,
            "Id": 4,
            "Name": "Infrastructure Administrators",
            "Description": "This role can view system status
instrumentation and system information",
            "CreatedTimestampUtc": "2019-11-07T13:14:52.757Z",
            "ModifiedTimestampUtc": "2019-11-07T13:14:56.493Z",
            "SystemRole": true
        },
        "Principal": null
    },
    {
        "PrincipalId": 3,
        "RoleId": 6,
        "CreatedTimestampUtc": "2020-01-23T08:09:43.31Z",
        "Role": {
            "AssignedManagementGroupCount": 1,
            "HasAllDevicesManagementGroupAssigned": true,
            "AssignedPrincipalCount": 1,
            "Id": 6,
            "Name": "Global Questioners",
            "Description": "This role can ask questions and view
responses for all instruction sets",
            "CreatedTimestampUtc": "2019-11-07T13:14:54.957Z",
            "ModifiedTimestampUtc": "2019-11-07T13:14:56.12Z",
            "SystemRole": true
        },
        "Principal": null
    }
]
```


Now let's replace the Roles for that Principal with 2 other Roles: Custom Properties Administrators and Consumer Administrators, which have Ids 9 and 10 respectively.

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

Making a PUT request to https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3 with following payload:

Payload sent to https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3

```
[9,10]
```

will yield following response:

Return payload


```
[
    {
        "PrincipalId": 3,
        "RoleId": 9,
        "CreatedTimestampUtc": "2020-01-23T08:24:
25.36Z",
        "Role": {
            "AssignedManagementGroupCount": 0,
            "HasAllDevicesManagementGroupAssigned":
false,
            "AssignedPrincipalCount": 2,
            "Id": 9,
            "Name": "Custom Properties
Administrators",
            "Description": "This role can add, edit
or delete custom properties",
            "CreatedTimestampUtc": "2019-11-07T13:
14:55.133Z",
            "ModifiedTimestampUtc": "2019-11-07T13:
14:55.133Z",
            "SystemRole": true
        },
        "Principal": null
    },
    {
        "PrincipalId": 3,
        "RoleId": 10,
        "CreatedTimestampUtc": "2020-01-23T08:24:
25.36Z",
        "Role": {
            "AssignedManagementGroupCount": 0,
            "HasAllDevicesManagementGroupAssigned":
false,
            "AssignedPrincipalCount": 2,
            "Id": 10,
            "Name": "Consumer Administrators",
            "Description": "This role can manage
Consumers that use the Tachyon platform",
            "CreatedTimestampUtc": "2019-11-07T13:
14:55.523Z",
            "ModifiedTimestampUtc": "2019-11-07T13:
14:55.523Z",
            "SystemRole": true
        },
        "Principal": null
    }
]
```


Use PrincipalRoles object inside the Tachyon connector instance.

Replacing all Roles for a Principal with given Roles

```
roles = connector.PrincipalRoles.
ReplaceRolesForPrincipal(3, new List<int> { 9, 10
});
```

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

Lastly, I'll remove the assiciation to Custom Properties Administrators Role from the Principal

Direct Consumer API call	C# code using Consumer SDK library

Making a DELETE request to https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3 with following payload:

Payload sent to https://my.tachyon.server/Consumer/PrincipalRoles/Principal/3

```
[9]
```

will yield following response:

Response payload

```
[
    {
        "PrincipalId": 3,
        "RoleId": 10,
        "CreatedTimestampUtc": "2020-01-23T08:24:25.36
Z",
        "Role": {
            "AssignedManagementGroupCount": 0,
            "HasAllDevicesManagementGroupAssigned":
false,
            "AssignedPrincipalCount": 2,
            "Id": 10,
            "Name": "Consumer Administrators",
            "Description": "This role can manage
Consumers that use the Tachyon platform",
            "CreatedTimestampUtc": "2019-11-07T13:14:
55.523Z",
            "ModifiedTimestampUtc": "2019-11-07T13:14:
55.523Z",
            "SystemRole": true
        },
        "Principal": null
    }
]
```

Use PrincipalRoles object inside the Tachyon connector instance.

Deleting multiple Roles from a Principal

```
connector.PrincipalRoles.
DeleteRolesFromPrincipal(3, new List<int> { 9
});
```

The method will return a boolean value.

Looking at the issue from the perspective of a Role, you can assign or unassign Principal(s) to a Role, or replace any Principals that Role has with a given set of Principals.

In the examples below we'll work with the Custom Role we've created above. It has the Id of 31 and first we'll assign 3 Principals to it:

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request to https://my.tachyon.server/Consumer/ PrincipalRoles/Role/31

Payload sent to https://my.tachyon.server/Consumer/PrincipalRoles/Role/31

```
[3, 5, 6]
```

will yield following response:

Response payload

```
[
    {
        "PrincipalId": 3,
        "RoleId": 31,
        "CreatedTimestampUtc": "2020-01-23T10:
00:11.63Z",
```

Use PrincipalRoles object inside the Tachyon connector instance.

Adding multiple Principals to a Role

```
principals = connector.PrincipalRoles.
AddPrincipalsToRole(31, new List<int> { 3, 5, 6 });
```

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

```
        "Role": null,
        "Principal": {
            "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
        }
    },
    {
        "PrincipalId": 5,
        "RoleId": 31,
        "CreatedTimestampUtc": "2020-01-23T10:
00:11.63Z",
        "Role": null,
        "Principal": {
            "Id": 1,
            "ExternalId": "S-1-5-21-
1202660629-789336158-1349024092-24850",
            "PrincipalName":
"SomeDomain\\Alastor.Rushal",
            "Email": "Alastor.
Rushal@SomeDomain.com",
            "Enabled": true,
            "CreatedTimestampUtc": "2019-11-
07T13:14:52.777Z",
            "ModifiedTimestampUtc": "2019-11-
07T13:14:52.777Z",
            "SystemPrincipal": false,
            "DisplayName": "Alastor Rushal",
            "IsGroup": false
        }
    },
    {
        "PrincipalId": 6,
        "RoleId": 31,
        "CreatedTimestampUtc": "2020-01-23T10:
00:11.63Z",
        "Role": null,
        "Principal": {
            "Id": 2,
            "ExternalId": "S-1-5-21-
1202660629-789306118-1349024091-24337",
            "PrincipalName":
"SomeDomain\\Marius.Gage",
            "Email": "Marius.Gage@SomeDomain.
com",
            "Enabled": true,
            "CreatedTimestampUtc": "2019-11-
07T13:14:56.687Z",
            "ModifiedTimestampUtc": "2019-11-
07T13:14:56.687Z",
            "SystemPrincipal": false,
            "DisplayName": "Marius Gage",
            "IsGroup": false
        }
    }
]
```

Now we will replace the existing 3 Principals with 2, one of which, "SomeDomain\\Jane.Doe" was already assigned to the Role:

Direct Consumer API call	C# code using Consumer SDK library

Making a PUT request to https://my.tachyon.server/Consumer/ PrincipalRoles/Role/31

Payload sent to https://my.tachyon.server/Consumer/PrincipalRoles/Role/31

```
[3, 4]
```

will yield following response:

Return payload


```
[
    {
        "PrincipalId": 3,
        "RoleId": 31,
        "CreatedTimestampUtc": "2020-01-23T10:
00:11.63Z",
        "Role": null,
        "Principal": {
            "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
        }
    },
    {
        "PrincipalId": 4,
        "RoleId": 31,
        "CreatedTimestampUtc": "2020-01-23T10:
00:11.63Z",
        "Role": null,
        "Principal": {
            "Id": 4,
            "ExternalId": "S-1-5-21-
3276326578-728399001-2836074973-1009",
            "PrincipalName":
"SomeDomain\\John.Doe",
            "Email": "John.Doe@SomeDomain",
            "Enabled": true,
            "CreatedTimestampUtc": "2016-11-
30T15:10:10.73Z",
            "ModifiedTimestampUtc": "2016-11-
30T15:10:10.73Z",
            "SystemPrincipal": false,
            "DisplayName": "John Doe",
            "IsGroup": false
        }
    }
]
```


Use PrincipalRoles object inside the Tachyon connector instance.

Replacing all Principals for a Role with given Principals

```
principals = connector.PrincipalRoles.
ReplacePrincipalsForRole(31, new List<int> { 3, 4 });
```

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

And finally we'll remove all "SomeDomain\\John.Doe" from the Role, leaving just "SomeDomain\\Jane.Doe" assigned to the Role:

Direct Consumer API call	C# code using Consumer SDK library

Making a DELETE request to https://my.tachyon.server/Consumer/ PrincipalRoles/Role/31
 	
Payload sent to https://my.tachyon.server/Consumer /PrincipalRoles/Role/31
 	
`[4]`
 	
will yield following response:
 	
Return payload
 	
`[{ "PrincipalId": 3, "RoleId": 31, "CreatedTimestampUtc": "2020-01-23T10:00: 11.63Z", "Role": null, "Principal": { "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 } }]`
	Use PrincipalRoles object inside the Tachyon connector instance.
	
	Removing multiple Principals from a Role
	
	`connector.PrincipalRoles.DeletePrincipalsFromRole (31, new List<int> { 4 });`
	
	The method will return a boolean value.

As we can see, the Principal and Role assignment, unassignment and replacement endpoints mirror each other covering the approach from either Principal or Role point of view.

Lastly, you can create or remove a link between a single Principal and a single Role.

In this example we'll create a link between Customer Role, which has the id of 31 and "SomeDomain\\John.Doe" Principal, which has the Id of 4.

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request to https://my.tachyon.server/Consumer/PrincipalRoles

Request payload sent to https://my.tachyon.server/Consumer/PrincipalRoles

```
{
    "PrincipalId": 4,
    "RoleId": 31
}
```

will yield following response:

Return payload

```
{
    "PrincipalId": 4,
    "RoleId": 31,
    "CreatedTimestampUtc": "2020-01-23T10:41:24.503
Z",
    "Role": {
        "AssignedManagementGroupCount": 0,
        "HasAllDevicesManagementGroupAssigned":
false,
        "AssignedPrincipalCount": 4,
        "Id": 31,
        "Name": "Custom role",
        "Description": "this is a description",
        "CreatedTimestampUtc": "2020-01-21T13:36:
55.897Z",
        "ModifiedTimestampUtc": "2020-01-21T13:36:
55.897Z",
        "SystemRole": false
    },
    "Principal": {
        "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:
10.73Z",
        "ModifiedTimestampUtc": "2016-11-30T15:10:
10.73Z",
        "SystemPrincipal": false,
        "DisplayName": "John Doe",
        "IsGroup": false
    }
}
```

Use PrincipalRoles object inside the Tachyon connector instance.

Assigning Principal to a Role

```
var payload = new PrincipalRole
{
    PrincipalId = 4,
    RoleId = 31
};
principal = connector.PrincipalRoles.Add
(payload);
```

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

and now we'll remove the link we've just created:

Direct Consumer API call	C# code using Consumer SDK library

Making a DELETE request to https://my.tachyon.server/Consumer/Role/31/Principal/4 will not yield any response payload so you should check the HTTP return code to see if the operation was successful.	Use PrincipalRoles object inside the Tachyon connector instance. **Unassigning Principal from a Role** `connector.PrincipalRoles.Delete(31, 4);` The method will return a boolean value.

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 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 Management Groups

Management Groups can only be assigned to a Role which has Permission to at least one Instruction Set. This prerequisite stems from internal implementation and it has to be taken into account when designing Permissions in your system.

The example below assumes that Role "Custom Role" with the Id of 31 has at least one Instruction Set Permission assigned to it.

You can assign or unassign either a single Management Group or a collection of Management Groups. To assign a single Management Group to a Role you'd issue a POST request to https://my.tachyon.server/Consumer/Roles/[role Id]/ManagementGroups /[management group id], so for instance to assign Management Group wit the Id of 7 to role with the Id of 31 you'd call https://my.tachyon.server/Consumer/Roles/31/ManagementGroups /7.

Making a DELETE request instead of POST will results in the Management Group being unassigned from the Role.

These requests do not require any payload because all information is already present in the URI. Consumer SDK equivalents are Roles.AddPermissionForManagementGroup(...) and Roles.RemovePermissionForManagementGroup(...) respectively, which simply take the Ids of the Role and the Management Group.

To assign or unassign Management Groups in bulk you have to use POST and DELETE respectively to https://my.tachyon.server/Consumer/ Roles/ManagementGroups, and below is an example of the payload that should be sent with the request.

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

To assign multiple Management Groups to a Role in a single call, make a POST request to https://my.tachyon.server/Consumer/Roles/ManagementGroups with following payload:

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

```
{
  "RoleId": 31,
  "ManagementGroupIds": [4,5]
}
```

which will yield following response:

Return payload

```
{
    "Role": {
        "Id": 31,
        "Name": "Custom role",
        "Description": "this is a description",
        "CreatedTimestampUtc": "2020-01-21T13:36:55.897Z",
        "ModifiedTimestampUtc": "2020-01-21T13:36:55.897Z",
        "SystemRole": false
    },
    "ManagementGroups": [
        {
            "Id": 4,
            "Name": "Management group number 3 from repository
Default Inventory",
            "Description": "This is the description for
Management group number 3 from repository Default Inventory",
            "Expression": "",
            "Type": 2,
            "Count": 0,
            "UsableId": "3",
            "HashOfMembers":
"24D13054496E24C95774CD8E983B09C2487D60819C627750B1C16FFDEDB23DF9"
,
            "CreatedTimestampUtc": "2019-12-02T12:01:21.667Z",
            "ModifiedTimestampUtc": "2020-01-08T16:31:02.807Z"
        },
        {
            "Id": 5,
            "Name": "Management group number 4 from repository
Default Inventory",
            "Description": "This is the description for
Management group number 4 from repository Default Inventory",
            "Expression": "",
            "Type": 2,
            "Count": 0,
            "UsableId": "4",
            "HashOfMembers":
"7784ED3BDA9B79F8635E728E1E0F529200D44F3F7AABAE1A26B2ABA6DB957EDB"
,
            "CreatedTimestampUtc": "2019-12-02T12:01:21.67Z",
            "ModifiedTimestampUtc": "2020-01-08T16:31:02.893Z"
        }
    ]
}
```

Use Roles object inside the Tachyon connector instance.

Assigning Management Groups to a Role

```
var payload = new Tachyon.SDK.
Consumer.Models.Send.
ManagementGroupsForRole
{
        RoleId = 31,
        ManagementGroupIds = new
List<int>{ 4, 5 }
};
role = connector.Roles.
AddPermissionForManagementGroups
(payload);
```

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

Unassigning Management Groups from a role is a mirror operation using the DELETE verb on the same web address:

Direct Consumer API call	C# code using Consumer SDK library

To unassign multiple Management Groups from a Role in a single call, make a DELETE request to https://my.tachyon.server/Consumer/Roles/ManagementGroups with following payload:

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

```
{
  "RoleId": 31,
  "ManagementGroupIds": [4]
}
```

This call will not return any data.

Use Roles object inside the Tachyon connector instance.

Unassigning Management Groups from a Role

```
var payload = new Tachyon.SDK.
Consumer.Models.Send.
ManagementGroupsForRole
{
        RoleId = 31,
        ManagementGroupIds = new
List<int> { 4 }
};
connector.Roles.
RemovePermissionForManagementGroups
(payload);
```

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

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
}
```

C# code using Consumer SDK library

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);`