Deal with connected devices running Tachyon Agent (now part of 1E Client)

Introduction

This section covers how to get information about computers connected to Tachyon. Tachyon keeps information about computers that are connected to it (through the Tachyon agent\client) or that connected to it within last 90 days (this is configurable) .

Although from Tachyon 4.1 the Tachyon Agent is now called the 1E Client this section still refers to it as the Tachyon Agent . This is because we're dealing with the Tachyon part of the 1E Client.

This section helps you understand how to check what impact an instruction has on your estate.

In this section:

- The C# examples assume you're using the Tachyon Consumer SDK
- You have an instantiated instance of Tachyon connector class in an object called connector
- A computer is a device running the Tachyon Agent service
- 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 contains that data, since that's always true .

On this page:

Find all computers

The most basic use case is finding every computer Tachyon knows about:

Direct Consumer API call	C# code using Consumer SDK library
Making a GET request to https://my.tachyon.server/Consumer/Devices will return this response:	Use Devices object inside the Tachyon connector instance and call GetDevices without any parameters.

Response received from https://my.tachyon.server/Consumer/Devices


```
[
    {
        "Id": 1,
        "Name": "Somemachine120",
        "Fqdn": "Somemachine120.somedomain.com",
        "Status": 0,
        "OsType": "Windows",
        "OsVerNum": 1688863374311424,
        "OsVerTxt": "Microsoft Windows 7
Enterprise",
        "AgentVersion": 281483566841860,
        "Manufacturer": "Dell Inc.",
        "ChassisType": 7,
        "DeviceType": "Desktop",
        "CpuType": "Intel(R) Xeon(R) CPU E5-2630
0 @ 2.30GHz",
        "CpuArchitecture": "x64",
        "OsArchitecture": "x64",
        "RamMB": 32693,
        "SMBiosGuid": "55f144f2-c52b-4349-add4-
1467dc34449e",
        "TachyonGuid": "eadad09d-1dd0-4f72-966a-
f53d1302875f",
        "LastBootUTC": "2019-02-04T00:00:00Z",
```


Retrieving devices

```
var devices = connector.Devices.GetDevices();
```

The devices object contains the same data as the JSON response on the left.

```
        "LastConnUtc": "2019-03-07T08:14:35.93Z",
        "CreatedUtc": "2019-02-03T00:00:00Z",
        "VrPlatform": "",
        "TimeZone": 0,
        "CertType": "Domain",
        "CertExpiryUtc": "2019-10-09T04:53:24Z",
        "PushToken": null,
        "Model": "Precision T5600",
        "Domain": "SomeDomain",
        "Tags": "||",
        "ConnectionState": [],
        "LocalIpAddress": null,
        "TimeZoneId": null,
        "SerialNumber": null,
        "Criticality": 1,
        "CoverageTags": {}
    },
    {
        "Id": 2,
        "Name": "Somemachine121",
        "Fqdn": "Somemachine121.somedomain.com",
        "Status": 0,
        "OsType": "Windows",
        "OsVerNum": 1688863374311424,
        "OsVerTxt": "Microsoft Windows 7
Enterprise",
        "AgentVersion": 281483566841860,
        "Manufacturer": "Dell Inc.",
        "ChassisType": 7,
        "DeviceType": "Desktop",
        "CpuType": "Intel(R) Xeon(R) CPU E5-2630
0 @ 2.30GHz",
        "CpuArchitecture": "x64",
        "OsArchitecture": "x64",
        "RamMB": 32693,
        "SMBiosGuid": "81725c90-63b8-4e82-ac79-
8ff19ffb54a7",
        "TachyonGuid": "5c76c6fd-cda4-4690-ac1a-
d394cb2104f0",
        "LastBootUTC": "2019-02-04T00:00:00Z",
        "LastConnUtc": "2019-03-07T08:14:35.93Z",
        "CreatedUtc": "2019-02-03T00:00:00Z",
        "VrPlatform": "",
        "TimeZone": 0,
        "CertType": "Domain",
        "CertExpiryUtc": "2019-10-09T04:53:24Z",
        "PushToken": null,
        "Model": "Precision T5600",
        "Domain": "SomeDomain",
        "Tags": "||",
        "ConnectionState": [],
        "LocalIpAddress": null,
        "TimeZoneId": null,
        "SerialNumber": null,
        "Criticality": 1,
        "CoverageTags": {}
    },
    {
        "Id": 3,
        "Name": "Somemachine122",
        "Fqdn": "Somemachine122.somedomain.com",
        "Status": 0,
        "OsType": "Windows",
        "OsVerNum": 1688863374311424,
        "OsVerTxt": "Microsoft Windows 7
Enterprise",
        "AgentVersion": 281483566841860,
        "Manufacturer": "Dell Inc.",
        "ChassisType": 7,
        "DeviceType": "Desktop",
        "CpuType": "Intel(R) Xeon(R) CPU E5-2630
```

```
0 @ 2.30GHz",
        "CpuArchitecture": "x64",
        "OsArchitecture": "x64",
        "RamMB": 32693,
        "SMBiosGuid": "b30efd3e-39e3-4c52-bf5e-
cfc096d0545f",
        "TachyonGuid": "eff2792d-83f3-411f-9e04-
684f66ac50ea",
        "LastBootUTC": "2019-02-04T00:00:00Z",
        "LastConnUtc": "2019-03-07T08:14:35.93Z",
        "CreatedUtc": "2019-02-03T00:00:00Z",
        "VrPlatform": "",
        "TimeZone": 0,
        "CertType": "Domain",
        "CertExpiryUtc": "2019-10-09T04:53:24Z",
        "PushToken": null,
        "Model": "Precision T5600",
        "Domain": "SomeDomain",
        "Tags": "||",
        "ConnectionState": [],
        "LocalIpAddress": null,
        "TimeZoneId": null,
        "SerialNumber": null,
        "Criticality": 1,
        "CoverageTags": {}
    }
]
```

Finding a specific computer or computers

To find a specific computer or computers matching specific criteria, you should use a filter expression.

Filtering and sorting with pagination

The device's API supports standard filtering, sorting and pagination methods seen throughout the Tachyon Consumer API to find computers matching specific parameters. See API reference for a list of the columns you can sort and filter.

Direct Consumer API call	C# code using Consumer SDK library
Making a POST request with the following payload:	Use devices object inside the Tachyon connector instance and call GetDevices, while passing in an object containing filtering and sorting specification.

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

```
{
  "Start": 1,
  "PageSize": 3,
  "Filter": {
    "Attribute": "OsType",
    "Operator": "==",
    "Value": "Linux"
  }
}
```

Returns this response:

Return payload

```
{
    "TotalCount": 10,
    "Items": [
        {
            "Id": 41,
            "Name": "Somemachine1240",
```

Retrieving filtered, sorted and paginated devices

```
var searchSettings = new Search
{
    Start = 1,
    PageSize = 3,
    Filter = new ExpressionObject
    {
        Attribute = "OsType",
        Operator = "==",
        Value = "Linux"
    }
}
var devices = connector.Devices.GetDevices(searchSettings);
```

devices object contains the same data as the JSON response on the left.

```
            "Fqdn": "Somemachine1240.
somedomain.com",
            "Status": 0,
            "OsType": "Linux",
            "OsVerNum":
1688863374311424,
            "OsVerTxt": "Ubuntu 14.1",
            "AgentVersion":
281483566841860,
            "Manufacturer": "DELL",
            "ChassisType": 7,
            "DeviceType": "Server",
            "CpuType": "Intel(R) Core
i7 6700K 4.0GHz",
            "CpuArchitecture": "x64",
            "OsArchitecture": "x64",
            "RamMB": 32693,
            "SMBiosGuid": "ff2e3a5d-
0395-423a-90bd-4d5c8f7b6a76",
            "TachyonGuid": "1a59b12e-
a2cb-4ea4-926e-7d88c7e25a80",
            "LastBootUTC": "2019-02-
04T00:00:00Z",
            "LastConnUtc": "2019-03-
07T08:14:35.93Z",
            "CreatedUtc": "2019-02-
03T00:00:00Z",
            "VrPlatform": "",
            "TimeZone": 0,
            "CertType": "Domain",
            "CertExpiryUtc": "2019-10-
09T04:53:24Z",
            "PushToken": null,
            "Model": "Poweredge",
            "Domain": "SomeDomain",
            "Tags": "||",
            "ConnectionState": [],
            "LocalIpAddress": null,
            "TimeZoneId": null,
            "SerialNumber": null,
            "Criticality": 1,
            "CoverageTags": {}
        },
        {
            "Id": 42,
            "Name": "Somemachine1241",
            "Fqdn": "Somemachine1241.
somedomain.com",
            "Status": 0,
            "OsType": "Linux",
            "OsVerNum":
1688863374311424,
            "OsVerTxt": "Ubuntu 14.1",
            "AgentVersion":
281483566841860,
            "Manufacturer": "DELL",
            "ChassisType": 7,
            "DeviceType": "Server",
            "CpuType": "Intel(R) Core
i7 6700K 4.0GHz",
            "CpuArchitecture": "x64",
            "OsArchitecture": "x64",
            "RamMB": 32693,
            "SMBiosGuid": "837cdff5-
a3bb-4875-8158-da69751f607a",
            "TachyonGuid": "89044184-
2263-40e4-8fe8-83bb2215568e",
            "LastBootUTC": "2019-02-
04T00:00:00Z",
            "LastConnUtc": "2019-03-
07T08:14:35.93Z",
```

```
            "CreatedUtc": "2019-02-
03T00:00:00Z",
            "VrPlatform": "",
            "TimeZone": 0,
            "CertType": "Domain",
            "CertExpiryUtc": "2019-10-
09T04:53:24Z",
            "PushToken": null,
            "Model": "Poweredge",
            "Domain": "SomeDomain",
            "Tags": "||",
            "ConnectionState": [],
            "LocalIpAddress": null,
            "TimeZoneId": null,
            "SerialNumber": null,
            "Criticality": 1,
            "CoverageTags": {}
        },
        {
            "Id": 43,
            "Name": "Somemachine1242",
            "Fqdn": "Somemachine1242.
somedomain.com",
            "Status": 0,
            "OsType": "Linux",
            "OsVerNum":
1688863374311424,
            "OsVerTxt": "Ubuntu 14.1",
            "AgentVersion":
281483566841860,
            "Manufacturer": "DELL",
            "ChassisType": 7,
            "DeviceType": "Server",
            "CpuType": "Intel(R) Core
i7 6700K 4.0GHz",
            "CpuArchitecture": "x64",
            "OsArchitecture": "x64",
            "RamMB": 32693,
            "SMBiosGuid": "f911c21f-
c764-46d2-8a0d-d07666937475",
            "TachyonGuid": "1436299e-
77ff-46a9-8b9c-665351ed6da3",
            "LastBootUTC": "2019-02-
04T00:00:00Z",
            "LastConnUtc": "2019-03-
07T08:14:35.93Z",
            "CreatedUtc": "2019-02-
03T00:00:00Z",
            "VrPlatform": "",
            "TimeZone": 0,
            "CertType": "Domain",
            "CertExpiryUtc": "2019-10-
09T04:53:24Z",
            "PushToken": null,
            "Model": "Poweredge",
            "Domain": "SomeDomain",
            "Tags": "||",
            "ConnectionState": [],
            "LocalIpAddress": null,
            "TimeZoneId": null,
            "SerialNumber": null,
            "Criticality": 1,
            "CoverageTags": {}
        }
    ]
}
```

Finding specific computers

A computer has two properties that on their own should be enough to identify it:

- Fully Qualified Domain Name (FQDN) - each FQDN in an organization should be unique
- Tachyon GUID - created by the Tachyon Agent to uniquely identify a computer.

You can use either FQDN or Tachyon GUID to locate a specific computer.

In edge cases, a computer might be replaced (for example, during an upgrade) and the new computer given the same FQDN . Although the FQDN will be identical, the Tachyon GUID will be different as it's created during the Tachyon Agent install on a new computer.

Finding a Device by its Fully Qualified Domain Name (FQDN)

Because machine's FQDN might contain characters that are not URI-friendly, FQDN sent to the Consumer API has to be Base64 encoded.

If I wanted to look for a computer whose FQDN is "Somemachine.somedomain.com", I would have to base64 encode it and use the encoded string as parameter in the API call.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Devices/fqdn/U29tZW1hY2hpbmUuc29tZWRvbWFpbi5jb20= will yield following response:
 	
Response received from https://my.tachyon.server/Consumer/Devices/fqdn/U29tZW1hY2hpbmUuc29tZWRvbWFpbi5jb20=
 	
{
"Id": 1,
"Name": "Somemachine",
"Fqdn": "Somemachine.somedomain.com",
"Status": 0,
"OsType": "Windows",
"OsVerNum": 1688863374311424,
"OsVerTxt": "Microsoft Windows 7 Enterprise",
"AgentVersion": 281483566841860,
"Manufacturer": "Dell Inc.",
"ChassisType": 7,
"DeviceType": "Desktop",
"CpuType": "Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30GHz",
"CpuArchitecture": "x64",
"OsArchitecture": "x64",
"RamMB": 32693,
"SMBiosGuid": "55f144f2-c52b-4349-add4-1467dc34449e",
"TachyonGuid": "eadad09d-1dd0-4f72-966a-f53d1302875f",
"LastBootUTC": "2019-02-04T00:00:00Z",
"LastConnUtc": "2019-03-07T08:14:35.93Z",
"CreatedUtc": "2019-02-03T00:00:00Z",
"VrPlatform": "",
"TimeZone": 0,
"CertType": "Domain",
"CertExpiryUtc": "2019-10-09T04:53:24Z",
"PushToken": null,
"Model": "Precision T5600",
"Domain": "Somedomain",
"Tags": "||",
"ConnectionState": [],
"LocalIpAddress": null,
"TimeZoneId": null,
"SerialNumber": null,
"Criticality": 1,
"CoverageTags": {}
}
	Here FQDN doesn't have to be Base64 encoded because the SDK performs the encoding internally.
	
	Use Devices object inside the Tachyon connector instance.
	
	Retrieving device by its FQDN
	
	`var device = connector.Devices.GetDeviceByFqdn("Somemachine.somedomain.com");`
	
	Device object contains the same data as the JSON response on the left.

Finding a Device by its Tachyon GUID

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Devices/tachyonguid/1D8863621D374BBB9E39CABA431593AD

Response received from https://my.tachyon.server/Consumer/Devices/tachyonguid/1D886362-1D37-4BBB-9E39-CABA431593AD


```
{
    "Id": 50,
    "Name": "Somemachine",
    "Fqdn": "Somemachine.somedomain.com",
    "Status": 0,
    "OsType": "Linux",
    "OsVerNum": 1688863374311424,
    "OsVerTxt": "Ubuntu 14.1",
    "AgentVersion": 281483566841860,
    "Manufacturer": "DELL",
    "ChassisType": 7,
    "DeviceType": "Server",
    "CpuType": "Intel(R) Core i7 6700K 4.0GHz",
    "CpuArchitecture": "x64",
    "OsArchitecture": "x64",
    "RamMB": 32693,
    "SMBiosGuid": "e474c20c-2aac-46ad-9bf7-
0bda1c936e13",
    "TachyonGuid": "1d886362-1d37-4bbb-9e39-
caba431593ad",
    "LastBootUTC": "2019-02-04T00:00:00Z",
    "LastConnUtc": "2019-03-07T08:14:35.93Z",
    "CreatedUtc": "2019-02-03T00:00:00Z",
    "VrPlatform": "",
    "TimeZone": 0,
    "CertType": "Domain",
    "CertExpiryUtc": "2019-10-09T04:53:24Z",
    "PushToken": null,
    "Model": "Poweredge",
    "Domain": "SomeDomain",
    "Tags": "||",
    "ConnectionState": [],
    "LocalIpAddress": null,
    "TimeZoneId": null,
    "SerialNumber": null,
    "Criticality": 1,
    "CoverageTags": {}
}
```


Use devices object inside the Tachyon connector instance.

Retrieving device by its Tachyon GUID

```
var device = connector.Devices.GetDeviceByGuid
("1D8863621D374BBB9E39CABA431593AD");
```

device object contains the same data as the JSON response on the left.

Finding management groups for computers

Tachyon supports organizing computers into management groups. A single computer can belong to many groups and at least belong to the All Devices group. All computers belong to the All Devices group by default and cannot be removed from it.

Management groups affect how instructions are targeted , so it's helpful to find out what groups computers belong to . You can do this by looking a computer up by its FQDN or Tachyon GUID as described in finding a specific computer.

By Fully Qualified Domain Name (FQDN)

Just as before, we have to use base64 encoded FQDN.

Direct Consumer API call	C# code using Consumer SDK library

Making a GET request to https://my.tachyon.server/Consumer/Devices/fqdn/U29tZW1hY2hpbmUuc29tZWRvbWFpbi5jb20=/ManagementGroups

Reponse received from https://my.tachyon.server/Consumer/Devices/fqdn/U29tZW1hY2hpbmUuc29tZWRvbWFpbi5jb20=/ManagementGroups


```
[
    {
        "Id": 1,
        "Name": "All Devices",
        "Description": "All devices are members of this
ManagementGroup",
        "Expression": null,
        "Type": 0,
        "Count": -1,
        "UsableId": "global",
        "HashOfMembers": null,
        "CreatedTimestampUtc": "2019-03-06T11:48:53.58
Z",
        "ModifiedTimestampUtc": "2019-03-06T11:48:53.58
Z"
    }
]
```


Use Devices object inside the Tachyon connector instance.

Finding out which Management Groups device belong to using its FQDN

```
var device = connector.Devices.
GetManagementGroupsForDeviceFqdn("Somemachine.
somedomain.com");
```

Device object contains the same data as the JSON response on the left.

By Tachyon GUID

Direct Consumer API call

Making a GET request to https://my.tachyon.server/Consumer/Devices/tachyonguid/1D8863621D374BBB9E39CABA431593AD/ManagementGroups

Response received from https://my.tachyon.server/Consumer/Devices/tachyonguid/1D8863621D374BBB9E39CABA431593AD/ManagementGroups


```
[
    {
        "Id": 1,
        "Name": "All Devices",
        "Description": "All devices are members of this
ManagementGroup",
        "Expression": null,
        "Type": 0,
        "Count": -1,
        "UsableId": "global",
        "HashOfMembers": null,
        "CreatedTimestampUtc": "2019-03-06T11:48:53.58Z",
        "ModifiedTimestampUtc": "2019-03-06T11:48:53.58Z"
    }
]
```


C# code using Consumer SDK library

Use Devices object inside the Tachyon connector instance.

Finding out which Management Groups device belong to using its Tachyon GUID

```
var guid = Guid.Parse
("1D8863621D374BBB9E39CABA431593AD");
var device = connector.Devices.
GetManagementGroupsForDeviceTachyonGuid
(guid);
```

Device object contains the same data as the JSON response on the left.

Find computers matching scope

Using scope allows you to limit which computers receive an instruction. Read defining the scope to find out more about how scope works, and narrowing the scope for examples.

Although using scope can seem like filtering, there are some differences, for example scope:

- Supports tags and management groups while regular filtering does not
- Does not support sorting or pagination so all results are returned in one go.

Some differences relate to the intended use of both features, for example:

- Filtering's design allows you to view the estate
- Scope's design allows you to narrow the target device audience of an instruction.

Scope expressions are standard Tachyon filter expressions. To find out more, read using scope and filter expressions and defining a filter.

Direct Consumer API call

Making a POST request with following payload:

Payload sent to https://my.tachyon.server/Consumer/Devices/scope

```
{
        "Operator": "AND",
        "Operands": [{
                "Attribute": "OsType",
                "Operator": "==",
                "Value": "Linux"
        },
        {
                "Attribute": "model",
                "Operator": "==",
                "Value": "Cisco UCS"
        }]
}
```

Returns this response:

Return payload

```
[
    {
        "Id": 49,
        "Name": "Somemachine1248",
        "Fqdn": "Somemachine1248.somedomain.
com",
        "Status": 0,
        "OsType": "Linux",
        "OsVerNum": 1688863374311424,
        "OsVerTxt": "Ubuntu 14.1",
        "AgentVersion": 281483566841860,
        "Manufacturer": "DELL",
        "ChassisType": 7,
        "DeviceType": "Server",
        "CpuType": "Intel(R) Core i7 6700K 4.0
GHz",
        "CpuArchitecture": "x64",
        "OsArchitecture": "x64",
        "RamMB": 32693,
        "SMBiosGuid": "25a0acf0-dddd-4b0e-8d71-
7ac5869be4b5",
        "TachyonGuid": "5a3ea68c-76f7-4b81-
8428-c0896755105c",
        "LastBootUTC": "2019-02-04T00:00:00Z",
        "LastConnUtc": "2019-03-07T08:14:35.93
Z",
        "CreatedUtc": "2019-02-03T00:00:00Z",
        "VrPlatform": "",
        "TimeZone": 0,
        "CertType": "Domain",
        "CertExpiryUtc": "2019-10-09T04:53:
24Z",
        "PushToken": null,
        "Model": "Cisco UCS",
        "Domain": "SomeDomain",
        "Tags": "||",
        "ConnectionState": [],
        "LocalIpAddress": null,
```

C# code using Consumer SDK library

Use Device object inside the Tachyon connector instance.

Retrieving devices matching scope

```
var expression = new ExpressionObject
{
    Operator = "AND",
    Operands = new List<ExpressionObject>
    {
        new ExpressionObject
        {
            Attribute = "OsType",
            Operator = "==",
            Value = "Linux"
        },
        new ExpressionObject
        {
            Attribute = "model",
            Operator = "==",
            Value = "Cisco UCS"
        }
    }
};

var statistics = connector.Devices.
GetDevicesMatchingScope(expression);
```

Devices object contains the same data as the JSON response on the left.

```
        "TimeZoneId": null,
        "SerialNumber": null,
        "Criticality": 1,
        "CoverageTags": {}
    },
    {
        "Id": 50,
        "Name": "Somemachine1249",
        "Fqdn": "Somemachine1249.somedomain.
com",
        "Status": 0,
        "OsType": "Linux",
        "OsVerNum": 1688863374311424,
        "OsVerTxt": "Ubuntu 14.1",
        "AgentVersion": 281483566841860,
        "Manufacturer": "DELL",
        "ChassisType": 7,
        "DeviceType": "Server",
        "CpuType": "Intel(R) Core i7 6700K 4.0
GHz",
        "CpuArchitecture": "x64",
        "OsArchitecture": "x64",
        "RamMB": 32693,
        "SMBiosGuid": "e474c20c-2aac-46ad-9bf7-
0bda1c936e13",
        "TachyonGuid": "1d886362-1d37-4bbb-
9e39-caba431593ad",
        "LastBootUTC": "2019-02-04T00:00:00Z",
        "LastConnUtc": "2019-03-07T08:14:35.93
Z",
        "CreatedUtc": "2019-02-03T00:00:00Z",
        "VrPlatform": "",
        "TimeZone": 0,
        "CertType": "Domain",
        "CertExpiryUtc": "2019-10-09T04:53:
24Z",
        "PushToken": null,
        "Model": "Cisco UCS",
        "Domain": "SomeDomain",
        "Tags": "||",
        "ConnectionState": [],
        "LocalIpAddress": null,
        "TimeZoneId": null,
        "SerialNumber": null,
        "Criticality": 1,
        "CoverageTags": {}
    }
]
```


Find an approximate target for an instruction

When issuing instructions think about your permissions, as they limit which management groups you can target. For more information read how do Management Groups work.

An approximate target:

- Is calculated in the context of a specific instruction definition
- Accepts an optional scope expression
- Gives you an overview on how the instruction you're about to issue will affect your estate.

Returned information is aggregated by computer and operating system type. You cannot configure this aggregation.

The following example is using the "1E-Explorer-TachyonCore-InstalledOS" instruction definition. This instruction definition has an Id of 124 on my Tachyon installation, so if I would like to know what part of the estate would be affected if I sent this instruction, I should use this Id when making the reques.

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

Making a POST request to https://my.tachyon.server/Consumer/Devices/approxtarget/124 without any payload returns this response:

Return payload


```
{
    "TotalDevicesOnline": 0,
    "TotalDevicesOffline": 50,
    "ByDeviceType": [
        {
            "AggregateValue": "Desktop",
            "CountOffline": 40,
            "CountOnline": 0
        },
        {
            "AggregateValue": "Server",
            "CountOffline": 10,
            "CountOnline": 0
        }
    ],
    "ByOsType": [
        {
            "AggregateValue": "Linux",
            "CountOffline": 10,
            "CountOnline": 0
        },
        {
            "AggregateValue": "Windows",
            "CountOffline": 40,
            "CountOnline": 0
        }
    ]
}
```


Use Devices object inside the Tachyon connector instance.

Retrieving information about affected devices

```
var statistics = connector.Devices.
GetApproximateTarget(null, 124);
```

Statistics object contains the same data as the JSON response on the left.

To check the effect the same instruction has on my estate when issued with a scope I should add the scope to the request's payload:

Direct Consumer API call	C# code using Consumer SDK library

Making a POST request with following payload:

Payload sent to https://my.tachyon.server/Consumer/Devices/approxtarget/124

```
{
        "Operator": "AND",
        "Operands":[{
            "Attribute": "OsType",
            "Operator": "==",
            "Value": "Linux"
        },
        {
            "Attribute": "model",
            "Operator": "==",
            "Value": "Cisco UCS"
        }]
}
```

Returns following response:

Return payload

```
{
    "TotalDevicesOnline": 0,
    "TotalDevicesOffline": 2,
    "ByDeviceType": [
        {
            "AggregateValue": "Server",
            "CountOffline": 2,
            "CountOnline": 0
        }
    ],
    "ByOsType": [
        {
            "AggregateValue": "Linux",
            "CountOffline": 2,
            "CountOnline": 0
        }
    ]
}
```

Use Devices object inside the Tachyon connector instance.

Retrieving information about affected devices, narrowed by scope

```
var expression = new ExpressionObject
{
    Operator = "AND",
    Operands = new List<ExpressionObject>
    {
        new ExpressionObject
        {
            Attribute = "OsType",
            Operator = "==",
            Value = "Linux"
        },
        new ExpressionObject
        {
            Attribute = "model",
            Operator = "==",
            Value = "Cisco UCS"
        }
    }
};

var statistics = connector.Devices.GetApproximateTarget
(expression, 124);
```

Statistics object contains the same data as the JSON response on the left.