Introduction

How you can integrate software with Tachyon using public facing Application Programming Interfaces (API).

This guide is not aimed at the general reader, it's intended primarily for use by developers who want to create their own consumers, for example using PowerShell or C#.

To get the most from this guide you should have a solid understanding of Tachyon, APIs, HTTP Communication and web API terminology .

The Tachyon API will change as new functionality is introduced and other functionality is either deprecated or removed. For this reason you should integrate with a specific version of the Tachyon Consumer API, preferably the most recent one.

The Consumer API reference shows the minimum version of the API for a given endpoint and any differences between each version.

On this page:

What is the Consumer API?

Tachyon exposes an API called Consumer API and any software accessing this API is a Consumer. Tachyon comes with several preinstalled Consumers, like the Tachyon Explorer interface. If you access the Consumer API using an API testing tool, like Fiddler or Postman, these tools are also considered Consumers. Every Tachyon UI uses the Consumer API. There are no separate, special API endpoints for the Tachyon Explorer or any other built-in Consumer. If the built-in Consumer can do something, your Consumer will be able to do it. You'll have to create a Consumer that you'll use and make sure that it's present in your license.

Please contact support@1e.com to set up a feature in your license that will contain your Consumer.

Accessing the Consumer API

What am I going to need?

Any tool or programming language capable of issuing HTTP requests should be able to use the Tachyon Consumer API . This means you can use your tool of choice.

To help you use the Consumer API we have designed:

- Tachyon .NET Consumer SDK - written in .NET and providing an abstraction layer to make working with Tachyon easier
- Consumer API reference - describing available API endpoints, expected received and returned payloads, permissions
- How to guides - to perform common operations in Tachyon using the Consumer API, c overing basic use-cases with examples in raw JSON payload and Tachyon .NET Consumer SDK.

Tachyon Server versions from 3.3 have documentation you can access with a web browser. Go to **https://<MyTachyonServer>/Consumer/swagger/ui/index#/** (replace <MyTachyonServer> with the address of your Tachyon installation) to view the API documentation .

Use the the built-in swagger documentation as your primary guide regardless of your chosen tool or language and consider Consumer API Reference as back-up source .

How do Consumers work?

Consumers do not have to be purpose-built programs. API testing tools are also considered Consumers. A Python or PowerShell script can also be a Consumer.

Tachyon uses the concept of a Consumer to differentiate the various pieces of software accessing it. It also limits 3rd party software in its use of Tachyon (for example, by defining how many simultaneous instructions a given Consumer can issue) or define where responses to instructions should be offloaded to.

A Consumer is identified by its name and this name has to match between the license, its registration in Tachyon and each API call made. Consumer names are not case sensitive, so "MyConsumer" is considered the same as "MyCoNsUmEr". You can think of a Consumer name as an identifier, a passport if you will, that gains a piece of software access to Tachyon Consumer API. As an example, let's assume that you wish to have your own consumer and you want to call it "ScriptRunner". This means that your license file has to have a Consumer entry with the name="ScriptRunner", you have to register "ScriptRunner" as a Consumer in Tachyon and enable it, and you will have to add X-Tachyon-Consumer header field with the value of "ScriptRunner" to every API call you make using your chosen tool. Don't worry if this sounds complex, we will go through each of these elements on this page.

Many pieces of software can introduce themselves as the same consumer, for instance you can have a set of scripts using the same Consumer. Be aware that all software using the same Consumer shares that Consumer's limits and settings that are set inside Tachyon.

A Consumer must fulfill a number of prerequisites to access Tachyon. Consumers must be:

1. **Licensed** - Tachyon comes with a number of Consumers, like Explorer and Settings, that are already licensed. Any software wishing to access Tachyon must also be present in the license. The Consumer name used in the license must match the name you register with Tachyon using either Settings application or Consumer API directly.

2. **Registered with Tachyon** - a number of system Consumers are registered when installing Tachyon. All other Consumers can be registered after installing Tachyon using Settings web application or using an API call. The name of the Consumer you use here must match the name you have in your license. Note that if a piece of software wishes to register a Consumer, it doesn't have to be a Consumer itself. Permissions, will be checked as usual, read Registering a new Consumer for details.
3. **Enabled** - a setting present in the license entry for the consumer and on a Consumer registered with Tachyon.

When all three prerequisites are met, the Consumer is allowed to access Tachyon. The API call will be subject to permissions assigned using Tachyon Role Based Access Control (RBAC) to the Principal making the call.

Consumers and licensing

Consumers must be present in the license file and be enabled.

Look in your license file for an entry like the one below (where [your consumer name here] is the name of your consumer):

Sample Consumer entry in the license file

```
<Consumer name="[your consumer name here]" enable="on"> </Consumer>
```

If you cannot find this entry, please contact 1E Support and ask for an updated license file.

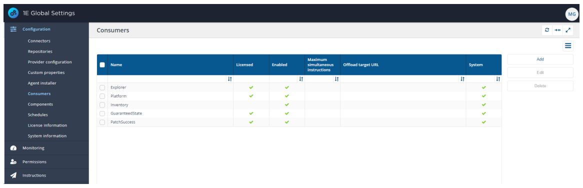
The feature containing your Consumer might look something like this:

Consumer feature in a license file

```
<Feature name="Customer A Consumer">
    <Consumer name="MyConsumer" enable="on"> </Consumer>
</Feature>
```

⚠ The consumer must have enable set to equal "on".

Using Consumers



Consumers

□	Name	Licensed	Enabled	Maximum simultaneous instructions	Offload target URL	System
□	Explorer	✓	✓			✓
□	Platform	✓	✓			✓
□	Inventory		✓			✓
□	GuaranteedState	✓	✓			✓
□	PatchSuccess	✓	✓			✓

Each time an API call is made, Tachyon checks if the:

- Principal making the call has sufficient permissions to do that call
- Consumer they're using fulfills the three prerequisites - licensed, registered with Tachyon and enabled.

The Consumer is checked first, the Principal's permissions are checked second only if the Consumer is successfully verified.

How does software notify Tachyon what Consumer it is?

To define what Consumer is issuing the call, add an X-Tachyon-Consumer field to the header of the HTTP call. Because each HTTP call is independent, each call must include this field in its header.

Some API endpoints do not require or check this header, but most do, so it should be included in each Consumer API call.

Registering a new Consumer

API endpoints relating to Consumers do not check if the software calling them is a Consumer that meets the three prerequisites listed above. This means Consumers can be added or removed by, for instance, their installation or uninstall processes. Although these API endpoints do not check for the X-Tachyon-Consumer HTTP header, they will always check the permissions of the calling Principal.

A Consumer can be registered (added) directly using the API. All the endpoints responsible for Consumer management are located in the Consumer's controller of the API. This is available under **https://<MyTachyonServer>/Consumer/Consumers** URI. Read the Consumer API reference documentation for details.

A Consumer can also be registered through the Tachyon **Settings** web application. Browse to **Configuration** and select the **Consumers** menu item.

To manage Consumers, the Principal making the call must have Write permission on Consumers' securable type. Set this by assigning the Principal to the **Consumer Administrator** role.

Consumers vs Principals

In addition to the concept of a Consumer, Tachyon has a concept of a Principals. A Principal can be a user or a group taken from Active Directory. While Tachyon does come with at least one Principal that represents a machine account (therefore a local account, not an Active Directory one), this Principal is created during installation for the purpose of internal communication between various Tachyon components and you cannot add non Active Directory Principals yourself.

A Consumer is a piece of software. A Principal is the user of that software. A "user" is something or someone using the software, for example a computer account running a service is a user even though it's not a person. Consumers are subject to Consumer check as described above. Principals are subject to RBAC checks.

How do Consumers, Principals and Permissions interact

It is important to understand how Permissions interact with Consumers and Principals.

Consumers cannot be assigned Permissions. They cannot be limited to only certain Principals either. In short, Consumers have no link or interaction whatsoever with Permissions or Principals.

Principals have Permissions through Roles they are assigned to, as per standard RBAC model.

This means that a Principal will have the same set of Permissions regardless of what Consumer they use.

If you were to launch a web browser and open Tachyon Explorer, you would have exactly the same set of Permissions as if you had opened Guaranteed State application, made an API call through Postman or run a Powershell script that contacts Consumer API.

When Princiapls and Permissions are concerned, the Consumer is irrelevant. Any Consumer can be used, as long as it is licensed and registered in Tachyon. It is only a tool that is used to perform an action. For RBAC what matter is **who** performs the action.

What happens when I make a call to Consumer API?

Tachyon performs two independent checks when it receives an API call:

1. A Consumer API check for X-Tachyon-Consumer header is performed for the vast majority of API endpoints, Consumer management and Information endpoints being notable exceptions.
    a. If given endpoint is excluded from Consumer name check, this first check concludes and the second check in performed.
    b. Consumer API will check the header of the incoming request and retrieve the value of X-Tachyon-Consumer.
        i. If this header is missing the call is rejected and an error is returned stating that X-Tachyon-Consumer field was missing in the header.
    c. Consumer API verifies that the Consumer name obtained from X-Tachyon-Consumer represents a Consumer that is known to Tachyon, that said Consumer is licensed and enabled.
        i. If any of these checks fail, the call is rejected and an error is returned that will describe the issue that caused the Consumer to be rejected, be that it not being registered, unlicensed etc.
2. Consumer API checks Permissions on the Principal making the call
    a. If the calling Principal doesn't possess the necessary Permissions to perform the operation requested, the call is rejected and an error is returned.

Excercise in accessing Consumer API

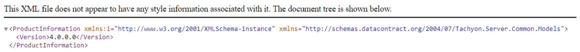
Hello world example

First we'll try to determine if you can access the API by browsing to **https://<MyTachyonServer>/Consumer/Information**.

This endpoint doesn't require the X-Tachyon-Consumer field to be present in the request's header so can be made from a web browser.

If everything works you should see a piece of XML that will state the version of the product.

Performing this step will help you establish that everything it up and running and can be accessed via given URL.



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<ProductInformation xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.datacontract.org/2004/07/Tachyon.Server.Common.Models">
   <Version>4.0.0.0</Version>
 </ProductInformation>
```

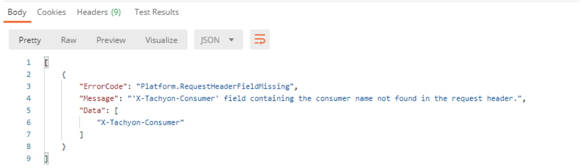
Next, access the API using an API testing tool, like Postman www.getpostman.com or Telerik Fiddler https://www.telerik.com/fiddler. With Postman (shown in the picture) remember that not all versions handle Windows Authentication correctly. Please have a look at API testing tools page which discusses several tools and their configuration.

In the examples below we'll use Postman.

First, issue a GET request to **https://<MyTachyonServer>/Consumer/SystemInformation**

You should see something like the screenshot on the right.

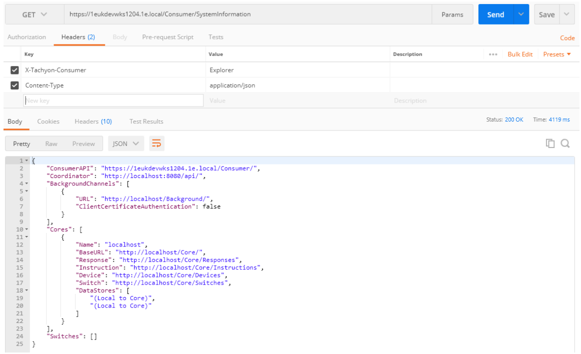
This is because you didn't include **X-Tachyon-Consumer** in the request header.



```
[
    {
        "ErrorCode": "Platform.RequestHeaderFieldMissing",
        "Message": "'X-Tachyon-Consumer' field containing the consumer name not found in the request header.",
        "Data": [
            "X-Tachyon-Consumer"
        ]
    }
]
```

Now let us add this field to the header. The example on the right is using "Explorer" as the Consumer name, but you should use the name of you Consumer.

Now when you make the request you should see something like this:



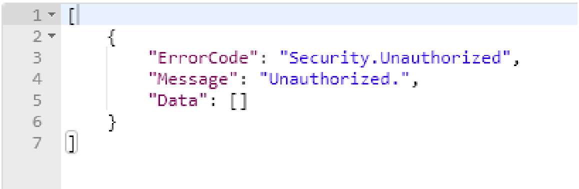
https://1eukdevwks1204.1e.local/Consumer/SystemInformation

	Key	Value	Description
☑	X-Tachyon-Consumer	Explorer	
☑	Content-Type	application/json	
	New key	Value	Description

Status: 200 OK Time: 4119 ms

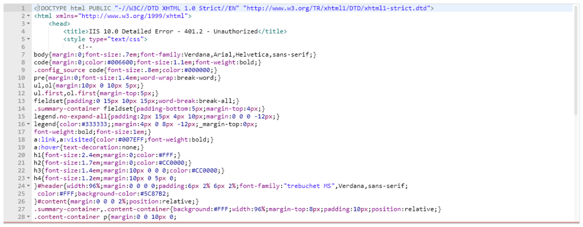
```
{
    "ConsumerAPI": "https://1eukdevwks1204.1e.local/Consumer/",
    "Coordinator": "http://localhost:8080/api/",
    "BackgroundChannels": [
        {
            "URL": "http://localhost/Background/",
            "ClientCertificateAuthentication": false
        }
    ],
    "Cores": [
        {
            "Name": "localhost",
            "BaseURL": "http://localhost/Core/",
            "Response": "http://localhost/Core/Responses",
            "Instruction": "http://localhost/Core/Instructions",
            "Device": "http://localhost/Core/Devices",
            "Switch": "http://localhost/Core/Switches",
            "DataStores": [
                "(Local to Core)",
                "(Local to Core)"
            ]
        }
    ],
    "Switches": []
}
```

You might see something like this:



```
[
    {
        "ErrorCode": "Security.Unauthorized",
        "Message": "Unauthorized.",
        "Data": []
    }
]
```

Or like this:



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>IIS 10.0 Detailed Error - 401.2 - Unauthorized</title>
        <style type="text/css">
            <!--
body{margin:0;font-size:.7em;font-family:Verdana,Arial,Helvetica,sans-serif;}
code{margin:0;color:#006600;font-size:1.1em;font-weight:bold;}
.config_source code{font-size:.8em;color:#000000;}
pre{margin:0;font-size:1.4em;word-wrap:break-word;}
ul,ol{margin:10px 0 10px 5px;}
ul.first,ol.first{margin-top:5px;}
fieldset{padding:0 15px 10px 15px;word-break:break-all;}
.summary-container fieldset{padding-bottom:5px;margin-top:4px;}
legend.no-expand-all{padding:2px 15px 4px 10px;margin:0 0 0 -12px;}
legend{color:#333333;;margin:4px 0 8px -12px;_margin-top:0px;
font-weight:bold;font-size:1em;}
a:link,a:visited{color:#007EFF;font-weight:bold;}
a:hover{text-decoration:none;}
h1{font-size:2.4em;margin:0;color:#FFF;}
h2{font-size:1.7em;margin:0;color:#CC0000;}
h3{font-size:1.4em;margin:10px 0 0 0;color:#CC0000;}
h4{font-size:1.2em;margin:10px 0 5px 0;
}#header{width:96%;margin:0 0 0 0;padding:6px 2% 6px 2%;font-family:"trebuchet MS",Verdana,sans-serif;
 color:#FFF;background-color:#5C87B2;
}#content{margin:0 0 0 2%;position:relative;}
.summary-container,.content-container{background:#FFF;width:96%;margin-top:8px;padding:10px;position:relative;}
.content-container p{margin:0 0 10px 0;
```

In the case of the first error above, you've reached Tachyon and it's telling you the Principal you're using is not authorized. This can mean that it has not been added to Tachyon or does not have permissions to use the endpoint being called. The error you see follows the Tachyon standard error reporting mechanism described in Handling errors returned by Tachyon.

If you get a web page back, like in the second error, it means your request has been rejected by IIS, not Tachyon. This can mean there's a problem with the credentials supplied in the request (incorrect username or password), your IIS settings or the tool you're using. It can also be because your tool does not support Windows Authentication, or because Windows Authentication is disabled. Tachyon uses Windows Authentication for API calls so your chosen tool must support it.

Payload format

Most API endpoints use application JSON data formatting for expected and returned data.

> ⚠ Be aware that some endpoints take multi-part form-data or in a few cases return an octet stream.

Read the Swagger documentation included in the product or Consumer API reference for the details of all endpoints provided by the Consumer API.

Downloading the Tachyon SDK

The Tachyon SDK is available from the 1E Support Portal.

1. Download the version of **TachyonSDK zip** that you require for your version of Tachyon:
    - **v5.1.0.663** from https://1eportal.force.com/s/article/Tachyon51
    - **v5.0.0.592** from https://1eportal.force.com/s/article/Tachyon5
    - **v4.1.0.206** from https://1eportal.force.com/s/article/Tachyon41
2. Extract the files from the download zip to your choice of location
3. Use your choice of tool or programming language, for example PowerShell or C# to create your Consumer.

For details of file versions please refer Tachyon SDK release information.

Tachyon .NET Consumer SDK

The Tachyon .NET Consumer SDK contains an abstraction layer for the Consumer API. The layer is designed to help build Consumers by abstracting away the complexities of HTTP communication behind C# classes and providing models for the data sent to and received from Tachyon. If you're developing a .NET based application to interface with Tachyon, use the SDK library with the Consumer API reference and How to guides.

Read Tachyon Consumer SDK library for more information.

Tachyon Offloading SDK

The Tachyon Offloading SDK provides a managed DLL to use in an ASP.NET application to help create a target for Tachyon offloading. It does not access Tachyon directly, it enables the creation of software Tachyon can access during the offloading process.

Read Tachyon Offloading SDK library for more information.