

# PLUTORA®

## REST API Primer

April 11, 2017

V1.24

## Table of Contents

Introduction .....	3
Plutora REST API Basics.....	4
Order of Object Creation.....	8
Plutora REST API Setup .....	9
Postman Setup.....	10
Examples.....	13
Environment Creation and Modification .....	13
Modify Environment and Component Fields .....	21
Create an Environment Group with two connected Environments.....	26
Create and Environment Group.....	28
Connect the two Environments.....	28

## Introduction

The REST API for Plutora is a programmatic interface that enables creation, querying, modification and deletion of most Plutora objects such as Releases, Systems, and Environments. This functionality is useful for tool integrations, including:

1. Automating upload of data to Plutora from other tools.
2. Extracting data from Plutora to be supplied to other tools.
3. Triggering actions such as gate approvals.
4. Bulk object creation from spreadsheets or other data sources.

Before attempting to create automation around the Plutora REST API, it is advisable to understand the Plutora object model and the particulars of how the REST API interacts with the Plutora objects. This document sets to provide that background through description and examples.

While automation is typically implemented with a programming or scripting language, this document uses Postman to illustrate the use of the Plutora REST API. The use of Postman has advantages of being language agnostic, giving full access to the REST API, and will automatically generate source code for automation. Furthermore, the reader will be able to fully appreciate and understand the use of the Plutora REST API without being encumbered with the details of its implementation with any particular language. With that said, some basic familiarity with JSON and REST will be helpful.

## Plutora REST API Basics

For additional details, see the Integrations Documentation, [Setting up Plutora API](#).

The Plutora REST API uses three HTTP verbs:

- POST – object creation
- PUT – object modification
- GET – object query
- DELETE – object deletion

Plutora objects are reference by globally unique identifiers (GUIDs), not name. For example, when creating an Environment with the POST verb, it is necessary to identify the System that the Environment is to be based on. That System has to be referenced by its GUID as illustrated in the JSON snippet below:

```
"linkedSystemId": "6d5ac322-33cc-44fe-9324-17a114e94c82"
```

When referring to specific objects with any of the HTTP verbs, the GUID will also appear in the URL. For example, to get the details about the System above, the following REST API call would be called:

```
GET https://<API_host>/systems/6d5ac322-33cc-44fe-9324-17a114e94c82
```

Where `<API_host>` is the FQDN of the REST API host, such as `usapi.plutora.com`.

GUIDs are generated by Plutora, so the REST API user does not create GUIDs. When creating a new object with POST, the `id` field is not to be supplied, instead, the `name` field and any other required fields.

GUIDs can be obtained from the response body of a GET or POST request. For example, to find the GUIDs for all systems, use the following GET query:

```
GET https://<API_host>/systems
```

The response body will be of the JSON form below where the `id` can be found by searching for the desired System `name`:

```
[
  {
    "id": "6d5ac322-33cc-44fe-9324-17a114e94c82",
    "name": "My System"
  },
  {
    "id": "c5d3414f-6bab-4499-925f-26e71e41140d",
    "name": "Another System"
  }
]
```

There are also GUIDs for Organization, Users, and Customizations. These are found using the (abbreviated) REST API calls:

- GET organizations
- GET users
- GET lookupfields/{type}

where the *type* names are obtained using the GET lookupfields API. For example, to create an Environment, a UsageWorkItemId such as “Dev” needed. To find the GUID for “Dev” involves a two-step process:

1. Find the *type* corresponding to UsageWorkItemId in response to GET lookupfields

GET lookupfields

Response:

```
[
  {
    "name": "ActionsByType",
    "description": "Actions - View By Type"
  },
  ***SNIP***
  {
    "name": "UsedForWorkItem",
    "description": "Used for WorkItemName"
  }
]
```

2. Look up “Dev” in the response to GET lookupfields/{type} to obtain the “Dev” UsedForWorkItem GUID:

GET lookupfields/UsedForWorkItem

Response:

```
[
  {
    "id": "fa69ee6d-7218-4002-9fbd-200eadc10e33",
    "value": "Dev",
    "sortOrder": 108,
    "type": "UsedForWorkItem"
  },
  ***SNIP***
  {
    "id": "538dbcf5-0bf5-40b5-8f40-8b0edf797a87",

```

```

    "value": "UAT",
    "sortOrder": 82,
    "type": "UsedForWorkItem"
  }
]

```

Currently, Customizations can only be queried with the Plutora REST API. They cannot be created or modified; this must be done through the UI or by the Plutora support team.

GET and DELETE only take a URL. On the other hand, when using PUT and POST a *body* is to be supplied. The body provides the data necessary used in the modify or creation operation. In this primer, we will use JSON objects to represent this data. Other formats such as XML can be used as well. The data values supplied in the body of a Plutora REST API call can include one of the following datatypes:

Datatype	Sample JSON key-value snippet
String	"description": "Here is a free-form string"
Date	"implementationDate": "2017-12-01T18:00:00"
GUID	"id": "06381564-cc37-4db5-a81e-43a2f291e984"
Enum	"environmentStatus": "Active"
Array	"environmentIDs": [ "ee8c881f-8b1b-450f-9108-2ab68ef34a55", "a6bbd5c3-e42a-414d-992f-b9f6d5dc565f", "fc893d66-2251-441d-b899-ed349a5529d6" ]

When creating an object with the Plutora REST API, there is a minimum set of values that must be supplied and most of the time there will be dependencies on other objects. Thus, there is an order in which objects must be created. The UI or the [Plutora REST API documentation](#) can be used to identify the dependencies and minimum set of values. For example, when creating a Release from the UI, if you supply no entries and attempt to save the Release, you will get the following error message:

- Portfolio Association must be selected from the drop down
- Release ID cannot be blank
- Release Name cannot be blank
- Implementation Date must be set
- Release Type must be selected from the drop down
- Risk Level must be selected from the drop down

[hide](#)

Where the types of these fields can be found in the documentation for [POST releases](#):

Field	Type	Source	Look up REST API(s)
Portfolio Association	GUID	API look up from Organizations	GET organizations
Release ID	String	User supplied	N/A – user supplied value
Release Name	String	User supplied	N/A – user supplied value
Implementation Date	Date	User supplied	N/A – user supplied value
Release Type	GUID	API look up, Customizations	GET lookupfields GET lookupfields/ReleaseType
Risk Level	GUID	API look up, Customizations	GET lookupfields GET lookupfields/ReleaseRiskLevel

## Order of Object Creation

The outline below shows the creation dependency relationship between Plutora objects. Once objects such as Releases and System have been created, further associations can be made such as adding a System to a Release.

1. Organization, Users, and/or Customizations
    - a. Systems
      - i. Environments
        1. Hosts
          - a. Layers
  - b. Releases
    - i. Phases
    - ii. Gates
    - iii. Activities
    - iv. Criteria
  - c. TECRs
  - d. TEBRs
  - e. Changes
2. Environment Groups (Environment can be added after creation)

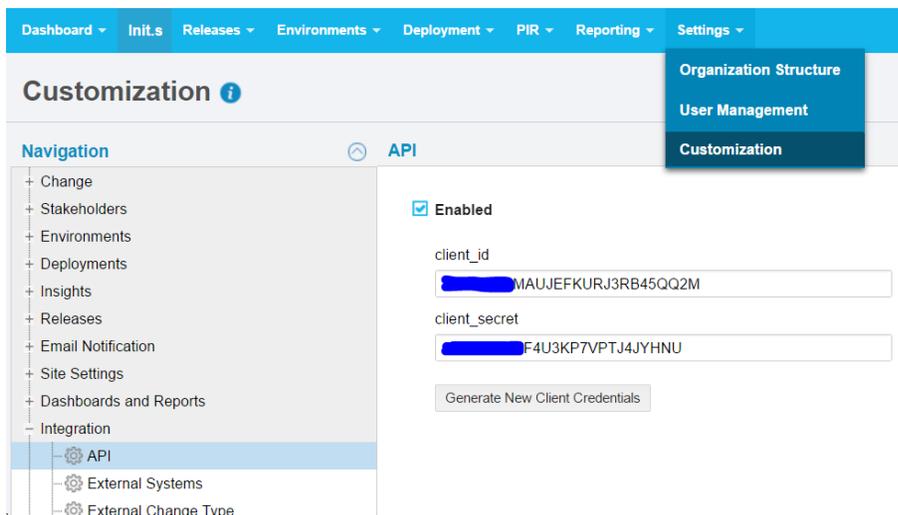
## Plutora REST API Setup

To make Plutora REST API calls you will need:

1. A command or tool capable of issuing REST commands
2. User credentials to a Plutora system which has API access enabled
3. Plutora API keys

There are commands (curl), programming and scripting language bindings (Java, C#, perl, python), and web-based tools (Postman) that can be used to issue REST commands. In this document we will be using Postman to illustrate the use of the Plutora REST API.

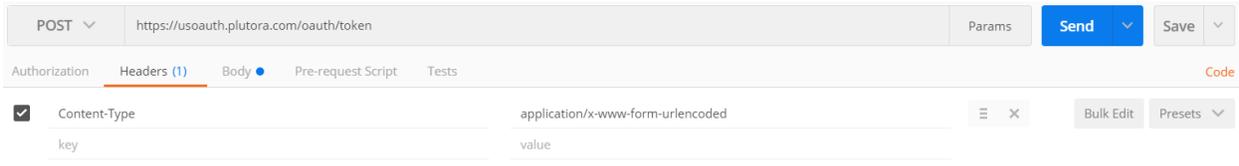
To enable Plutora REST API and access the API keys, navigate to Setting > Customizations > API:



## Postman Setup

Postman can be installed from [www.getpostman.com](http://www.getpostman.com).

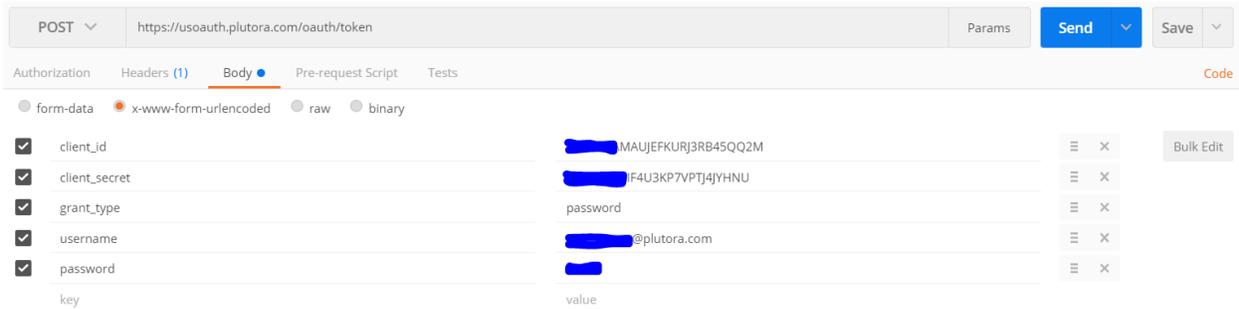
In addition to user credentials, an `access_token` is needed for each call to the Plutora REST API. This token can be generated using the `POST /oauth/token` API. To get the `access_token` in Postman, construct this request:



Make sure to select the POST verb. Your API host may be different than shown here depending on your region:

- US: `https://usapi.plutora.com`
- UK: `https://ukapi.plutora.com`
- AU: `https://auapi.plutora.com`

Add the Header keys "Content-Type" and set its value to "applications/x-www-form-urlencoded".



Select "x-www-form-urlencoded" for the Body format. Create the key-value pairs as shown above using your credentials and API keys. Press Send. The response you get back should look like this:

POST `https://usoauth.plutora.com/oauth/token` Params **Send** Save

Authorization Headers (1) Body Pre-request Script Tests Code

Content-Type `application/x-www-form-urlencoded` Bulk Edit Presets

key value

Body Cookies Headers (14) Tests Status: 200 OK Time: 391 ms

Pretty Raw Preview JSON Save Response

```

1 {
2   "access_token": "8BWEKNSTI_N5gotvW2IOxgJrMky_ifU80ZkSazt8QbrbwQn7Cuxs3j7ATRuiZoAt7vrGRwiLTqLaOoccELPCvsfd0AvvyVI
MzY5MT1N346_gtu74Tc3uKsGLN5HwcL3m0Vuc73EQvwUdTJVSLfmcuHqSqdUBG-G1Cbe
-8MAAeyRRZmYeh_uTcYujLu5qZtLVAVPdwEcr3SziZb6_uZuJ4msc1z1BbZFy8CSVBRkeFEteA45MJQjJzqkysA8meF3UV_oiQTN6XxL6YuMX
SxeYmF5m30rYwwADmkQfQXDRDLy6bquQrb_1G94qEvaJ84b8dUHbcnsZXnKXyQqZwbUYwpQRLkLcWLZ8s4pY-ku88XtNG-uf_B98naC0
-8ET8SEsY_ZBvKNDTDV7M8q_gtBDu3wnQ0SjHTmmhBQn8-TzAwbYGi41brzJPKA0UyLIwFCYvy43HySgQsofzgwV0180-Hq_kEzB9t
-12unq_o0y20MmK1_NmKxPQN3trKbC2xv_B50X6rUFHtcPki004x0Mc08Y",
3   "token_type": "bearer",
4   "expires_in": 1199,
5   "refresh_token": "gzHGk5zH28dTiZmfyt4J3y
-SFgCe2IjTrxz03S4mFvKqz7pTF_9YmGS3bxx0SOPmvuxo7IwR67rNayWvi63A4_HSgopDKpQBaLtSuyiy9SpJ63LYZktm_gkhgjMZJk1wKQ
o8hfPEk7Ev7TivRd1I6Qw7N47_x9GcpCerZ19Qk0GG
-AHCZ5J4RTZNPiIU5vEAwQRpRZBUgTsIOovhoLt193Fp0oftqDhbFfqS5Hm1NAaTIR_wd9uT31-L4
-gcVXR5Rjt4offdIScPzeu28T3YidttG1Gd1ZyXrJQiDYbY3P1Gp15fCAN831wwjdnC7q1m7gVwXN5HP-F0s--bg4A8HC-0PKhsYSDKNUH
-VbnJHXRvNMfDQ--u6od15D6CnziQ6cf-EKjY6
-jXf56ejV06HHgj0p9PTAzJT8_BTmYchj_C95r5Be_8hP2a4k1kM5vARRsM15qPwN_UhxwbIwY4uC5iJY1AIjro0V31aY85AGgU
-YwadiCtquyLvigyjum-Zvd-mJtLjUJe0vJxtjav1zjqqq"
6 }

```

It is the text in quotes after `"access_token"`: that we are interested. Copy this text. Also, you will want to save this POST query because you will be using it a lot; use the *Save* button which is located next to the *Send* button to do this. While you will be able to use your new *access\_token* for multiple API calls, it will eventually expire, resulting in a response like:

Body Cookies Headers (15) Tests Status: 401 Unauthorized Time: 380 ms

Pretty Raw Preview JSON Save Response

```

1 {
2   "message": "Authorization has been denied for this request."
3 }

```

The simplest Plutora REST API call is `GET me` which will return the logged in user's email address. To make this call, set up Postman as follows:

The screenshot displays a REST client interface. At the top, the method is set to GET and the URL is https://usapi.plutora.com/me. The 'Headers' tab is active, showing two headers: 'Authorization' with the value 'bearer 8BWEKNSTI\_N5gotvW2IOxε' and 'Content-Type' with the value 'application/json'. Below the headers, there is a table with columns for 'key' and 'value'. The response status is 200 OK and the time taken is 88 ms. The response body is shown in a 'Pretty' JSON view, displaying a single array element: a redacted email address followed by @plutora.com.

Make sure to select the GET verb. Use the appropriate API host and append /me. Add the two keys, "Authorization" and "Content-Type". "Authorization" should be set to the value "bearer <access\_token>" where <access\_token> is the value copied from the POST oauth/token API call made above. When you press the Send button you should see your email address in the response. Save this query, we can use it as a template for future GET calls.

## Examples

### Environment Creation and Modification

In this example, we will create an Environment model including “Techs & Specs”, that is, Hosts, and Layers. Referring to the Order of Object Creation diagram above, we will realize that before we can create the Environment model, we’ll need a System model. And even before that we’ll need to gather some Organization and Customization GUIDs.

#### *Organization and Customization GUIDs*

To identify all the Organization and Customization GUIDs needed, examine the [Plutora REST APIs](#) for each of the POST operation for each object we’ll be creating:

POST API	Organization & Customization GUIDs	Object GUIDs	REST Calls
Systems	OrganizationId		GET organizations
Environments	UsageWorkItemId EnvironmentStatusId	LinkedSystemId	GET lookupfields/UsedForWorkItem GET lookupfields/EnvironmentStatus GET systems
Hosts		EnvironmentID	GET environments
Layers	StackLayerID	HostID EnvironmentID	GET lookupfields/StackLayer GET hosts GET environments

Recall from above that to find the endpoints for UsageWorkItemId and EnvironmentStatusId we have to use the `GET lookupfields` API. At this point, none of the Object GUIDs exists, so let’s get the four Organization & Customization GUIDs. Then we’ll create each object in turn.

#### *OrganizationId*

Use Postman to issue the `GET organizations` API. The response will look something like this:

```
[
  {
    "id": "9d3879c9-bb43-4243-afd8-0f1ee6273663",
    "name": "ABC Group",
    "type": "Company"
  },
  {
    "id": "4cc6b70b-c075-4e50-b0b2-b9578144122c",
    "name": "Advice Systems",
    "type": "Department"
  },
  ***SNIP***
]
```

We can choose any organization, I will choose the root “ABC Group”, in my Plutora instance.

#### UsageWorkItemId

Use Postman to issue the `GET lookupfields/UsedForWorkItem` API. The response will look something like this:

```
[
  {
    "id": "fa69ee6d-7218-4002-9fbd-200eadc10e33",
    "value": "Dev",
    "sortOrder": 108,
    "type": "UsedForWorkItem"
  },
  ***SNIP***
]
```

Your entries will likely be different. I will choose the “Dev” *UsedForWorkItem*.

#### EnvironmentStatusId

Use Postman to issue the `GET lookupfields/EnvironmentStatus` API. The response will look something like this:

```
[
  {
    "id": "b060ed8b-6e3b-454d-af0f-e90938eb7b47",
    "value": "Active",
  }
]
```

```
"sortOrder": 114,
  "type": "EnvironmentStatus"
},
{
  "id": "f738999c-152b-46d5-8375-e8cc96dfb85c",
  "value": "Decommissioned",
  "sortOrder": 113,
  "type": "EnvironmentStatus"
},
{
  "id": "24ea5c49-9aab-4307-a95e-5267bd6d7d73",
  "value": "Under Build/Config",
  "sortOrder": 88,
  "type": "EnvironmentStatus"
}
]
```

I will choose "Active".

#### StackLayerID

Use Postman to issue the `GET lookupfields/stackLayer` API. The response will look something like this:

```
[
  {
    "id": "891f375c-58a8-4425-a8ec-5459ab04510b",
    "value": "Application",
    "sortOrder": 8490,
    "type": "StackLayer"
  },
  {
    "id": "8991fb6f-b7fa-43e2-812f-f0765c3f0f99",
    "value": "Database",
    "sortOrder": 8491,
    "type": "StackLayer"
  },
  {
    "id": "58ce5fa7-bfc4-4dcc-b014-08d2178d9ded",
    "value": "Middleware",
    "sortOrder": 8492,
    "type": "StackLayer"
  }
]
```

```

},
{
  "id": "42b0329c-6e45-434d-8183-297de5909c92",
  "value": "Network",
  "sortOrder": 8493,
  "type": "StackLayer"
},
{
  "id": "e4524cf4-235b-4ba1-ac94-66f274dd3f9d",
  "value": "Operating System",
  "sortOrder": 8494,
  "type": "StackLayer"
}
]

```

I will choose "Application".

In summary, here are my Organization and Customization GUIDs:

POST API	Organization & Customization GUIDs	GUIDs
Systems	OrganizationId	9d3879c9-bb43-4243-afd8-0f1ee6273663
Environments	UsageWorkItemId EnvironmentStatusId	fa69ee6d-7218-4002-9fbd-200eadc10e33 b060ed8b-6e3b-454d-af0f-e90938eb7b47
Layers	StackLayerID	891f375c-58a8-4425-a8ec-5459ab04510b

### Create System

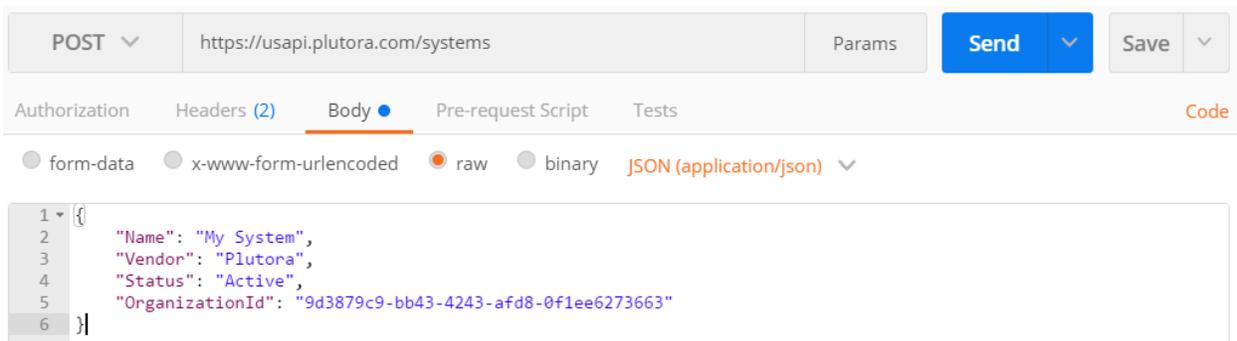
From the `POST systems` API documentation, we see that the following fields are required:

- Name
- Vendor
- Status
- OrganizationId

We'll need a JSON data structure these keys and our desired values. Here's what I'll use:

```
{
  "Name": "My System",
  "Vendor": "Plutora",
  "Status": "Active",
  "OrganizationId": "9d3879c9-bb43-4243-afd8-0f1ee6273663"
}
```

We'll be using the `POST` systems to create this new System. The POST will require the same header as the GET, so you can reuse your GET example from above. Select POST, choose "raw" data format and from the pull down select *JSON (application/json)*.



Again, save this POST as a template for the next few POST calls we will be doing.

The response from this call:

```
{
  "id": "2a8ad43b-81be-4d1a-8fff-0cb85b9905dc",
  "name": "My System",
  "vendor": "Plutora",
  "status": "Active",
  "organizationId": "9d3879c9-bb43-4243-afd8-0f1ee6273663",
  "organization": "ABC Group",
  "description": null,
  "isAllowEdit": false,
  "inMyOrganization": false,
  "additionalInformation": [
    ***SNIP***
  ]
}
```

From this response, we now have a System GUID that we can use to create our Environment.

View/Edit System Details ⌚ 📄 ✕

Details
Additional Information
Dependencies
Approvers

📧 Contact stakeholders

### System Definition

**System Name**

**Alias(es):**  
 Save Alias

**Description**

**Vendor**

**Status**  Active  Inactive

### Organization

**Portfolio Association**  ▼

### Stakeholders

**RACI Matrix**

Expand to see RACI Definitions
⌵

Stakeholder	System Role	R	A	C	I
Stakeholders not added yet					

Add Stakeholder
Remove

Delete System
Save
Save & Close

### Create Environment

From the `POST environments` API documentation, we see that the following fields are required:

- Name
- Vendor
- LinkedSystemId
- UsageWorkItemId
- EnvironmentStatusId
- Color
- IsSharedEnvironment

We'll need a JSON data structure these keys and our desired values. Here's what I'll use:

```
{
  "Name": "My Environment",
  "Vendor": "Plutora",
  "LinkedSystemId": "2a8ad43b-81be-4d1a-8fff-0cb85b9905dc ",
  "UsageWorkItemId": "fa69ee6d-7218-4002-9fbd-200eadc10e33",
  "EnvironmentStatusId": "b060ed8b-6e3b-454d-af0f-e90938eb7b47",
}
```

```
"Color": "#ffffff",
  "IsSharedEnvironment": "true"
}
```

Here's the response:

```
{
  "id": "a99b8e84-bec3-4085-8df1-969ae176d119",
  "name": "My Environment",
  "description": null,
  "url": null,
  "vendor": "Plutora",
  "linkedSystemId": "2a8ad43b-81be-4d1a-8fff-0cb85b9905dc",
  "linkedSystem": null,
  "environmentMgr": null,
  "usageWorkItemId": "fa69ee6d-7218-4002-9fbd-200eadc10e33",
  "usageWorkItem": null,
  "environmentStatusId": "b060ed8b-6e3b-454d-af0f-e90938eb7b47",
  "environmentStatus": null,
  "color": "#ffffff",
  "isSharedEnvironment": true,
  "hosts": null
}
```

Now we have the GUID for our Environment.

#### [Create Host](#)

From the `POST hosts` API documentation, we see that the following fields are required:

- Name
- EnvironmentID

We'll need a JSON data structure these keys and our desired values. Here's what I'll use:

```
{
  "Name": "My Host",
  "EnvironmentID": "a99b8e84-bec3-4085-8df1-969ae176d119"
}
```

Here's the response:

```
{
  "id": "1dd98953-a4fd-40dd-a416-dd7469ea157f",
  "name": "My Host",
  "environmentID": "a99b8e84-bec3-4085-8df1-969ae176d119",
  "layers": null
}
```

Now we have the GUID for our Host.

#### Create Layer

From the `POST layers` API documentation, we see that the following fields are required:

- ComponentName
- HostID
- EnvironmentID
- StackLayerID

We'll need a JSON data structure these keys and our desired values. The Version field is not required, but I'll added it so that we can adjust it in the next section. Here's what I'll use:

```
{
  "ComponentName": "My Application",
  "HostID": "1dd98953-a4fd-40dd-a416-dd7469ea157f",
  "EnvironmentID": "a99b8e84-bec3-4085-8df1-969ae176d119",
  "StackLayerID": "891f375c-58a8-4425-a8ec-5459ab04510b",
  "Version": "1.0"
}
```

Here's the response:

```
{
  "id": "4c7f6709-7258-4f50-ae79-08597b4f1454",
  "componentName": "My Application",
  "version": "1.0",
  "hostID": "1dd98953-a4fd-40dd-a416-dd7469ea157f",
  "environmentID": "a99b8e84-bec3-4085-8df1-969ae176d119",
  "stackLayerID": "891f375c-58a8-4425-a8ec-5459ab04510b",
  "stackLayer": null,
  "stackLayerType": null
}
```

Edit Environment Details
⌚ 📄 ✕

[Details](#)
[Additional Information](#)
[Stakeholders](#)
[Linked Items](#)
[Release Automation](#)
📧 Contact stakeholders

### Environment Definition

Environment Name:

Description:

System:

Integrated with:

URL:

Used for Phase:  Scheduler Display:

Vendor:

This is a Shared Environment  Status

Any booking will be automatically approved

### Technical Specs and Configuration

Import 📄
Add Host
Add Layer
Add Component
Update On Build

Layer	Component Name	Version	Actions
Host:	My Host		✖
	Application		✖
	My Application	1.0	✖

### Environment Allocation Schedule

Click here to hide allocations

## Modify Environment and Component Fields

Field values can be modified using the PUT verb. The steps to modifying fields are:

1. Use the GET <objectname> API to find the GUID of the object of interest
2. Use the GET <objectname>/{id} to fetch the object data structure
3. Copy the response to the *body* of POST <objectname>/{id}
4. Edit the desired values
5. *Send* the POST request

## Modify Environment Fields

GET environments

```
[
***SNIP***
{
  "id": "a99b8e84-bec3-4085-8df1-969ae176d119",
  "name": "My Environment",
  "url": null,
  "linkedSystem": "My System",
  "usageWorkItem": "Dev",
  "status": "Active",
  "color": "#ffffff"
},
]
```

```
***SNIP***
```

```
]
```

```
GET environments/a99b8e84-bec3-4085-8df1-969ae176d119
```

```
{
  "id": "a99b8e84-bec3-4085-8df1-969ae176d119",
  "name": "My Environment",
  "description": null,
  "url": null,
  "vendor": "Plutora",
  "linkedSystemId": "2a8ad43b-81be-4d1a-8fff-0cb85b9905dc",
  "linkedSystem": "My System",
  "environmentMgr": null,
  "usageWorkItemId": "fa69ee6d-7218-4002-9fbd-200eadc10e33",
  "usageWorkItem": "Dev",
  "environmentStatusId": "b060ed8b-6e3b-454d-af0f-e90938eb7b47",
  "environmentStatus": "Active",
  "color": "#ffffff",
  "isSharedEnvironment": false,
  "hosts": [
    {
      "id": "1dd98953-a4fd-40dd-a416-dd7469ea157f",
      "name": "My Host",
      "environmentID": "a99b8e84-bec3-4085-8df1-969ae176d119",
      "layers": [
        {
          "id": "4c7f6709-7258-4f50-ae79-08597b4f1454",
          "componentName": "My Application",
          "version": "1.0",
          "hostID": "1dd98953-a4fd-40dd-a416-dd7469ea157f",
          "environmentID": "a99b8e84-bec3-4085-8df1-969ae176d119",
          "stackLayerID": "891f375c-58a8-4425-a8ec-5459ab04510b",
          "stackLayer": "Application",
          "stackLayerType": "StackLayer"
        }
      ]
    }
  ]
}
```

Note that the Environment data structure includes details of the Hosts and Layers. The Host and Layers values cannot not be modified using the `PUT environments/{id}` API. Instead, the `PUT hosts/{id}` and the `PUT layers/{id}` API must be used. Let's change a description and change the color of the Environment:

```
PUT environments/a99b8e84-bec3-4085-8df1-969ae176d119
```

Using the modified response from above as the body.

```
{
  "id": "a99b8e84-bec3-4085-8df1-969ae176d119",
  "name": "My Environment",
  "description": "Here is the description of My Environment",
  "url": null,
  "vendor": "Plutora",
  "linkedSystemId": "2a8ad43b-81be-4d1a-8fff-0cb85b9905dc",
  "linkedSystem": "My System",
  "environmentMgr": null,
  "usageWorkItemId": "fa69ee6d-7218-4002-9fbd-200eadc10e33",
  "usageWorkItem": "Dev",
  "environmentStatusId": "b060ed8b-6e3b-454d-af0f-e90938eb7b47",
  "environmentStatus": "Active",
  "color": "#ffff00",
  "isSharedEnvironment": false,
  "hosts": [
    {
      "id": "1dd98953-a4fd-40dd-a416-dd7469ea157f",
      "name": "My Host",
      "environmentID": "a99b8e84-bec3-4085-8df1-969ae176d119",
      "layers": [
        {
          "id": "4c7f6709-7258-4f50-ae79-08597b4f1454",
          "componentName": "My Application",
          "version": "1.0",
          "hostID": "1dd98953-a4fd-40dd-a416-dd7469ea157f",
          "environmentID": "a99b8e84-bec3-4085-8df1-969ae176d119",
          "stackLayerID": "891f375c-58a8-4425-a8ec-5459ab04510b",
          "stackLayer": "Application",
          "stackLayerType": "StackLayer"
        }
      ]
    }
  ]
}
```

```
}
]
}
```

Response  
(none!)

**Edit Environment Details**

**Environment Definition**

Environment Name: My Environment

Description: Here is the description of My Environment

System: My System

Integrated with: Select Integrated Environment Group (if applicable)

URL:

Used for Phase: Dev Scheduler Display: #ffff00

Vendor: Plutora

This is a Shared Environment  Status: Active

Any booking will be automatically approved

**Technical Specs and Configuration**

Layer	Component Name	Version	Actions
Host	My Host		-
	Application		-
	My Application	1.0	-

**Environment Allocation Schedule**

Delete Save Save & Close

Note that you can also use a stripped-down version of the body:

```
{
  "id": "a99b8e84-bec3-4085-8df1-969ae176d119",
  "name": "My Environment",
  "description": "Here is the description of My Environment, take two",
  "vendor": "Plutora",
  "linkedSystemId": "2a8ad43b-81be-4d1a-8fff-0cb85b9905dc",
  "usageWorkItemId": "fa69ee6d-7218-4002-9fbd-200eadc10e33",
  "environmentStatusId": "b060ed8b-6e3b-454d-af0f-e90938eb7b47",
  "environmentStatus": "Active",
  "color": "#00ff00",
  "isSharedEnvironment": false
}
```

### Modify Layer Fields

To modify the Version field of the Application Component in our Environment Layer, we will need the Layer GUID. This value was part of the response for the `GET environments/{id}` above, `4c7f6709-7258-4f50-ae79-08597b4f1454`.

`GET layers/4c7f6709-7258-4f50-ae79-08597b4f1454`

Response

```
{
  "id": "4c7f6709-7258-4f50-ae79-08597b4f1454",
  "componentName": "My Application",
  "version": "1.0",
  "hostID": "1dd98953-a4fd-40dd-a416-dd7469ea157f",
  "environmentID": "a99b8e84-bec3-4085-8df1-969ae176d119",
  "stackLayerID": "891f375c-58a8-4425-a8ec-5459ab04510b",
  "stackLayer": "Application",
  "stackLayerType": "StackLayer"
}
```

Use the `PUT layers/{id}` with the modified response to change the *version* value.

`PUT layers/4c7f6709-7258-4f50-ae79-08597b4f1454`

Body:

```

{
  "id": "4c7f6709-7258-4f50-ae79-08597b4f1454",
  "componentName": "My Application",
  "version": "2.0",
  "hostID": "1dd98953-a4fd-40dd-a416-dd7469ea157f",
  "environmentID": "a99b8e84-bec3-4085-8df1-969ae176d119",
  "stackLayerID": "891f375c-58a8-4425-a8ec-5459ab04510b",
  "stackLayer": "Application",
  "stackLayerType": "StackLayer"
}

```

Response:

(none!)

Create an Environment Group with two connected Environments

Create a second System and Environment.

POST systems

```

{
  "name": "My second System",
  "vendor": "Plutora",
  "status": "Active",
  "organizationId": "9d3879c9-bb43-4243-afd8-0f1ee6273663"
}

```

## Response

```
{
  "id": "2bf3e7ab-517c-45ca-b33e-6689775d25b8",
  "name": "My second System",
  "vendor": "Plutora",
  "status": "Active",
  "organizationId": "9d3879c9-bb43-4243-afd8-0f1ee6273663",
  "organization": "ABC Group",
  "description": null,
  "isAllowEdit": false,
  "inMyOrganization": false,
  ***SNIP***
}
```

## POST environments

```
{
  "Name": "My second Environment",
  "Vendor": "Plutora",
  "LinkedSystemId": "2bf3e7ab-517c-45ca-b33e-6689775d25b8",
  "UsageWorkItemId": "fa69ee6d-7218-4002-9fbd-200eadc10e33",
  "EnvironmentStatusId": "b060ed8b-6e3b-454d-af0f-e90938eb7b47",
  "Color": "#ffffff",
  "IsSharedEnvironment": "true"
}
```

## Response

```
{
  "id": "49a77309-2684-4ac6-a4e9-9f4bd8f1ec0a",
  "name": "My second Environment",
  "description": null,
  "url": null,
  "vendor": "Plutora",
  "linkedSystemId": "2bf3e7ab-517c-45ca-b33e-6689775d25b8",
  "linkedSystem": null,
  "environmentMgr": null,
  "usageWorkItemId": "fa69ee6d-7218-4002-9fbd-200eadc10e33",
  "usageWorkItem": null,
  "environmentStatusId": "b060ed8b-6e3b-454d-af0f-e90938eb7b47",
}
```

```
"environmentStatus": null,
"color": "#ffffff",
"isSharedEnvironment": true,
"hosts": null
}
```

### Create and Environment Group

Currently, it is necessary to include the Environments when creating the Environment Groups.

POST environmentgroups

```
{
  "Name": "My Environment Group",
  "Description": "My Environment Group description",
  "Color": "#ffffff",
  "EnvironmentIDs": [
    "a99b8e84-bec3-4085-8df1-969ae176d119",
    "49a77309-2684-4ac6-a4e9-9f4bd8f1ec0a"
  ]
}
```

### Response

```
{
  "id": "ea63d7e9-565f-47c7-ae38-d41dcbb334bd",
  "name": "My Environment Group",
  "description": "My Environment Group description",
  "color": "#ffffff",
  "environmentIDs": [
    "a99b8e84-bec3-4085-8df1-969ae176d119",
    "49a77309-2684-4ac6-a4e9-9f4bd8f1ec0a"
  ]
}
```

### Connect the two Environments

The type of connections available are set up in Customizations. Let's find the GUID for *HTTPS*.

GET lookupfields

```
[
***SNIP***
{
```

```
    "name": "EnvironmentMapConnectivityType",
    "description": "Connectivity Type"
  },
  ***SNIP***
]
```

GET lookupfields/EnvironmentMapConnectivityType

```
[
  {
    "id": "376810cf-898c-4afb-b438-15bc4f1a549e",
    "value": "File",
    "sortOrder": 14460,
    "type": "EnvironmentMapConnectivityType"
  },
  {
    "id": "3090441f-e2eb-43ba-aa4e-eaf22e0bac36",
    "value": "HTTPS",
    "sortOrder": 14462,
    "type": "EnvironmentMapConnectivityType"
  },
  {
    "id": "28cafa45-1352-4f5e-bcfc-b500583a951a",
    "value": "Web Services",
    "sortOrder": 14461,
    "type": "EnvironmentMapConnectivityType"
  }
]
```

POST connectivities

```
{
  "environmentGroup": "ea63d7e9-565f-47c7-ae38-d41dcbb334bd",
  "source": "a99b8e84-bec3-4085-8df1-969ae176d119",
  "target": "49a77309-2684-4ac6-a4e9-9f4bd8f1ec0a",
  "connectivityType": "3090441f-e2eb-43ba-aa4e-eaf22e0bac36",
  "direction": "InAndOut"
}
```

Response

None

**Environment Groups** ✖

**Manage Environment Groups**  Click to collapse

Click on **Details** button to view or edit Add Integrated Environment Group Delete Group

Details	Name	Description	Portfolio Association	Used for Phase	Display
<a href="#">Details</a>	My Environment Group	My Environment Group description			#ffff

---

**Manage Connectivity** Click to collapse

Group Members [Connectivity](#)

**Environment Map Connectivity for My Environment Group**

Source	Direction	Target	Type
<input type="checkbox"/> My Environment	↔	My second Environment	HTTPS

Move Connection/s Add Connection Delete Connection

Save Save & Close