

ACE 2024

STUDENT TRAINING GUIDE

# Implementing RESTful Services



Copyright © 2024 by Aras Corporation. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, V1.0 or later (the latest version is presently available at <http://www.opencontent.org/openpub/>).

Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.

Distribution of the work or derivative of the work in any standard (paper) book form for a commercial purpose is prohibited unless prior permission is obtained from the copyright holder.

Aras Innovator, Aras, and the Aras Corp "A" logo are registered trademarks of Aras Corporation in the United States and other countries.

All other trademarks referenced herein are the property of their respective owners.

Microsoft, Office, SQL Server, IIS, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

### **Notice of Liability**

The information contained in this document is distributed on an "As Is" basis, without warranty of any kind, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose or a warranty of non-infringement. Aras shall have no liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this document or by the software or hardware products described herein.

Revision MARCH 2024

# Implementing RESTful Services

## Overview

In this session, we will discuss how to manipulate items using the Aras Innovator RESTful API that allows you to retrieve, add, update, and delete items, as well as execute server methods, stored in the Aras Innovator database. The API uses the Open Data Protocol (OData) which is an International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) approved Open Access Same-Time Information Standard (OASIS). Data elements that are exchanged with the server are defined using the JavaScript Object Notation (JSON) format. This allows any clients that support this protocol to communicate easily with the Aras Innovator server.

## Objectives

- Discuss the RESTful architecture.
- Review the Open Data Protocol (OData).
- Retrieve Items and Properties.
- Use Request options to filter data.
- Add new Items.
- Edit existing Items.
- Delete Items.
- Execute server methods.

## Defining RESTful Architecture

---

- Web service style (Representational State Transfer)
- Provides interoperability between computers using the internet
- Communicate using HTTP Methods/ URL's
- Guiding constraints:
  - Uniform interface
  - Client-server architecture
  - Stateless
  - Cacheable
  - Support Layered Systems
  - Code on Demand (Optional)



### Defining RESTful Architecture

REST stands for “Representational State Transfer” and it is a common architecture style for designing loosely coupled applications that interact using HTTP. REST does not enforce any rules regarding how it should be implemented at a lower level; it just defines high-level design guidelines, which are implemented by web services developers.

The Aras RESTful API has been created with these goals in mind and follows six architectural constraints, as outlined in the following recommended reading for this topic, included in detail in Roy T. Fielding’s year 2000 dissertation presented to the University of California, Irvine and found online at the following URL: [https://ics.uci.edu/~fielding/pubs/dissertation/fielding\\_dissertation.pdf](https://ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf).

## **Architectural Constraints**

The architectural constraints suggested in the technical literature above are briefly described below with a very simple approach.

### **Uniform Interface**

All resources should be accessible via a URI using a common approach (such as HTTP GET, POST, PATCH, DELETE) and follow a standardized set of syntax rules.

### **Client-Server architecture**

The development of the client and the server must be able to evolve independently.

### **Stateless**

Each client-to-server request must contain all the information necessary to understand the request and cannot take advantage of any stored context on the server. The session state is therefore kept entirely on the client.

### **Cacheable**

If a response is cacheable, then a client cache is given the right to reuse that response dataset for later equivalent requests.

### **Layered Systems**

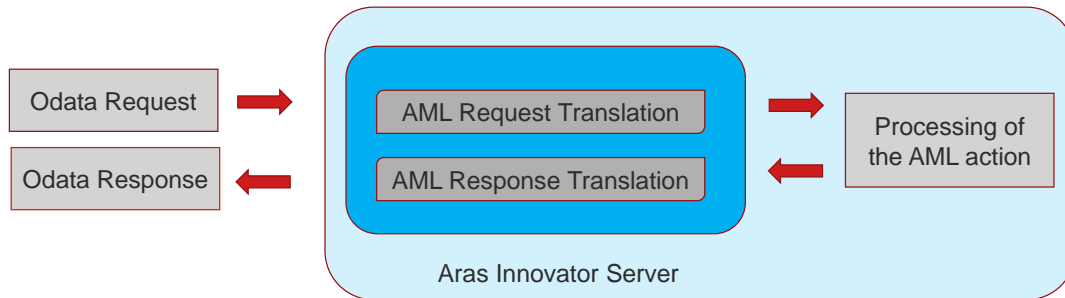
The architecture must support systems built in hierarchical layers so that different servers can work together to service a request.

### **Code on Demand (optional)**

Allows the client to download and execute code in the form of applets or scripts.

## Understanding Odata

- Data access protocol using REST architecture
- Built on the AtomPub protocol for retrieving and updating resources (OASIS standard)
- Aras Innovator Odata Interface:



## Understanding OData

The Open Data Protocol (OData) is a data access protocol for the web. OData provides a uniform mechanism to query and manipulate datasets through CRUD operations (create, read, update, and delete).

OData is built on the AtomPub protocol for retrieving and sending data and follows OASIS recommendations. It uses standardized technologies such as HTTP, XML, and JSON to send and receive data in a uniform manner, as well as define the data model being used.

The Aras implementation of RESTful Services uses OData to receive an OData Request at the server and subsequently translate that request into a corresponding AML request. The resulting dataset provided by the server execution of the received request is then translated back as an OData response and sent to the requesting client.

Understanding the Aras AML command language is extremely helpful when constructing and debugging OData requests to be consumed by the server.

Simply put, Aras Innovator RESTful API uses OData protocol to handle its RESTful API. On the other hand, JSON allows a more compact data format, hence enabling faster processing and communication with other systems that also support the OData protocol.

## Reviewing Odata URL

### ▪ Odata Request URL Format

`https://training.aras-cloud.com/sit-1-0-202111233/server/odata/Part?$top=2&$orderby=Name`

Service Root URL

Resource Path Query Options

#### Service Root URL

- `.../server/oData`
- Resource Path
  - ItemType or Method
- Query Options
  - Item Request Options



## Reviewing OData URL

The simple OData URL structure allows one to easily identify the elements involved in the client-server communication using HTTP messages. Those messages may include the following components:

- **Service Root URL**  
Contains the base URL for the requested service. It normally presents the name of the server and associated port in use, if needed.
- **Resource Path**  
Exposes the object accessible to HTTP elements, using its standard methods, such as GET, POST, PUT, PATCH and DELETE. For Aras, this element typically represents an ItemType, or server method involved with the request.
- **Query Options (optional)**  
Following the query string indicator (?), these parameters are passed to the service and support attributes to the queries to be applied on the requested resource. Multiple query strings can be included, and they are delimited by the ampersand character (&). The Aras API options include filtering, paging, and selecting data.

## Implementing HTTP Methods as Aras Actions

HTTP Method	Aras AML Action	Notes
GET	get	
POST	add	default for POST
POST	create	@aras.action: create
PUT	edit	update single property on an Item
PATCH	edit	default for PATCH
PATCH	Update	@aras.action: update
PATCH	Lock	@aras.action: lock
PATCH	Unlock	@aras.action: unlock
PATCH	Merge	@aras.action: merge
DELETE	Delete	default for DELETE
DELETE	Purge	@aras.action: purge



### Implementing HTTP Methods as Aras Actions

Every OData request relates to an HTTP method which is then used by the Aras API to determine its purpose and build the appropriate AML command to be presented to the server.





### **Insomnia API Testing Tool**

Several 3<sup>rd</sup> party tools are available to help you create and test HTTP requests and review the responses.

In this session, we will be using Insomnia, which is downloadable from <https://insomnia.rest/download>.

As an API testing tool, Insomnia will:

- Provide an HTTP Client to be used as a testbed for Web Services.
- Execute Aras RESTful API requests.
- Use local, cloud or Git based storage.
- Offer advanced scripting capabilities.
- Handle environmental variables.
- Incorporate secure authentication.

## Authorizing Requests Using OAuth Token

Our test implementation will explore the Insomnia capability of sharing an authentication token with all requests within a collection. To get the authentication token, proper system configuration parameters are entered onto the authorization form available in Insomnia. For further details on how to fill out the authorization form, see instructions on this configuration on the Aras Innovator RESTful API documentation that goes with each Aras Innovator release documentation. Specific reference to configure Postman properly is found on Chapter 7 – Using OAuth 2.0 Tokens from the Authentication Server, where the setup parameters are discussed in detail.

**Aras Innovator 29**  
RESTful API

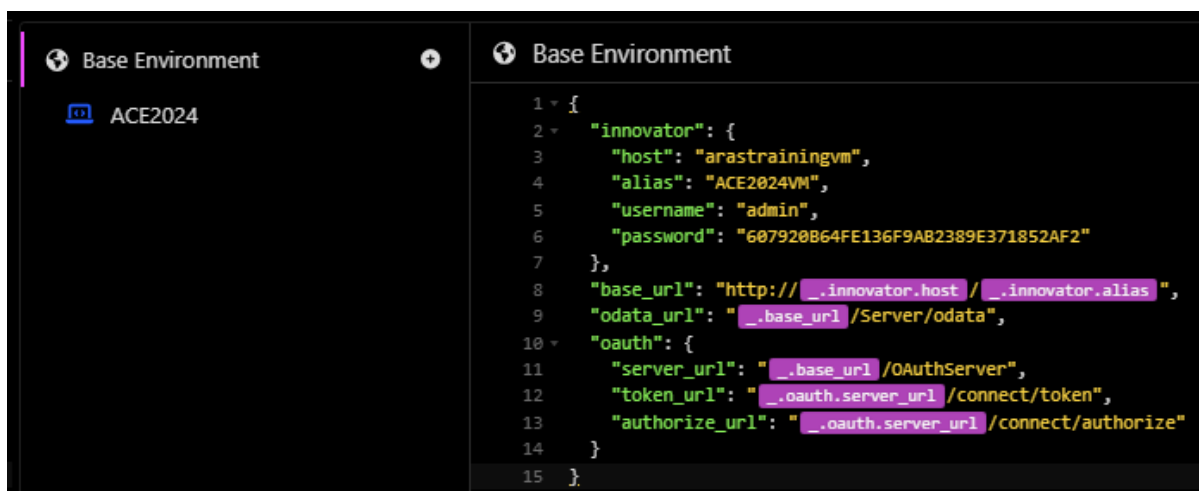
## 7 Using OAuth 2.0 Tokens from the Authentication Server

The Authentication Server enables to request an OAuth 2.0 token from the server to use as basic authentication. The information in this chapter has been adapted from the ["Authenticating in OAuth 2.0 with Aras RESTful API"](#) blog.

**Note:** The Aras Innovator version 12 and below require different authentication steps. Refer to [Token Authentication using the REST API](#) blog for more information.

From Aras Innovator 14+ versions, OAuth 2.0 tokens is used for token authentication.

The Aras Innovator API requires each request to be authorized to access the server using a set of credentials. An access token may be first obtained by providing a set of parameters to the Aras OAuthServer.



```

1 {
2   "innovator": {
3     "host": "arastrainingvm",
4     "alias": "ACE2024VM",
5     "username": "admin",
6     "password": "607920B64FE136F9AB2389E371852AF2"
7   },
8   "base_url": "http://_.innovator.host /_.innovator.alias ",
9   "odata_url": "_.base_url /Server/odata",
10  "oauth": {
11    "server_url": "_.base_url /OAuthServer",
12    "token_url": "_.oauth.server_url /connect/token",
13    "authorize_url": "_.oauth.server_url /connect/authorize"
14  }
15 }

```



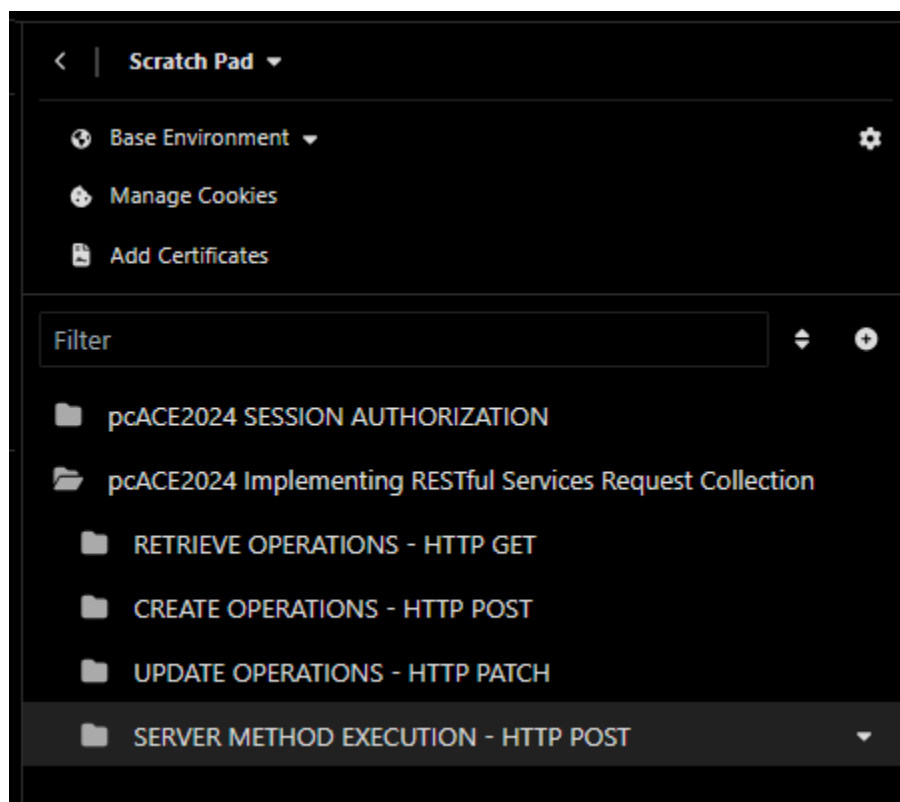
## Hands-on Practice

At this point of our discussion, we have established the technical foundation on which the handshake of a client requester system and the server response provider system communicate. The following examples will allow us to exercise and solidify this understanding using some hands-on exercises to experience the exchange of different types of queries.

Our sample queries were grouped into similar requests within each HTML method group, with some sort of progressive level of complexity, allowing us to provide examples of the manipulation of ItemTypes (datasets) on its entirety, or a more detailed approach at lower levels to manipulate single and specific items, or collection of items, within the corresponding dataset, with the application of arguments.

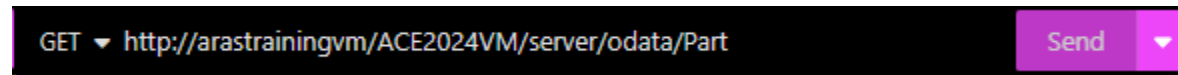
The additional arguments in the request string correspond to AML elements used to:

- Filter on Item properties.
- Select Item properties to be returned.
- Apply conditional selection of returned items based on property values and ranges.
- Include relationship properties to the result set.
- Include related items to the result set.
- Manipulate extended properties.
- Determine items to be affected by an update request.
- Invoke the execution of a server-side method.



## Get All Parts

The HTTP GET method is used to retrieve items from the database. The resource section of the OData URL specifies the ItemType (database table) the server operation will use to fulfill the request.



Try it:

### To retrieve all parts

1. To retrieve all items of a specific ItemType use the HTTP GET method and provide the ItemType name in the resource section of the OData URL.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen.

A screenshot of a REST client interface showing a successful GET response. The top bar displays '200 OK', '264 ms', and '290.4 KB'. The response is shown in a dark-themed editor with a 'Preview' tab selected. The JSON data is as follows:

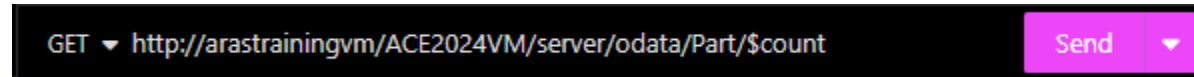
```

1 {
2   "@odata.context": "http://arastrainingvm/ACE2024VM/server/odata/$metadata#Part",
3   "value": [
4     {
5       "classification": "Component",
6       "control_type": "None",
7       "cost": 93.0100,
8       "cost_basis": "Actual",
9       "created_on": "2024-02-15T16:23:48",
10      "current_state@aras.name": "Preliminary",
11      "description": "Test part created for RESTful Services hands-on exercise during ACE2024
12      by Paulo Costa. DO NOT DELETE.",
13      "generation": 5,
14      "has_change_pending": "0",
15      "id": "126C99CDC1944ECE853A70F7A64EE859",
16      "is_current": "1",
17      "is_released": "0",
18      "keyed_name": "ACE-9301-PC",
19      "major_rev": "A",
20      "make_buy": "Make",
21      "modified_on": "2024-02-27T15:36:32",
22      "name": "HTTP PATCH TEST",
23      "new_version": "0",
24      "not_lockable": "0",
25      "state": "Preliminary",
26      "unit": "EA",
27      "item_number": "ACE-9301-PC",
28      "itemtype": "4F1AC04A2B484F3ABA4E20DB63808A88"
29    },
30    {
31      "created_on": "2024-01-11T16:46:07",
32      "current_state@aras.name": "Preliminary",
33      "generation": 1,
34      "has_change_pending": "0",
35      "id": "62A15ECA07234A94870899B31A37D330",
36      "is_current": "1",
37      "is_released": "0",
38      "keyed_name": "ap001-100",

```

## Get All Parts count

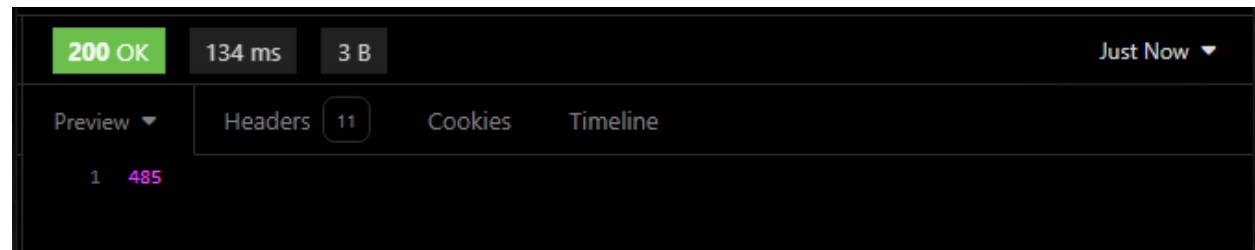
The HTTP GET method is used to retrieve items from the database. The addition of optional parameters will change the logic of the request, and consequently alter the presented result set.



### Try it:

#### To retrieve the total count of items in a table

1. To retrieve the total count of items of a specific ItemType, use the HTTP GET method, provide the ItemType name in the resource section of the OData URL, and append the option "\$count" at the end of the request.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen.



## Get All Parts Top 10

The HTTP GET method is used to retrieve items from the database. The addition of optional parameters will change the logic of the request, and consequently alter the presented result set.

GET <http://arastrainingvm/ACE2024VM/server/odata/Part>

Send

Try it:

### To retrieve the top 10 items of ItemType Part

1. To retrieve the top 10 items of ItemType Part, use the HTTP GET method, provide the ItemType name in the resource section of the OData URL, and append the option “?\$top=10” at the end of the request.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen.

```

200 OK 135 ms 5.4 KB Just Now
Preview Headers 11 Cookies Timeline
1 {
2   "@odata.context": "http://arastrainingvm/ACE2024VM/server/odata/$metadata#Part",
3   "value": [
4     {
5       "classification": "Component",
6       "control_type": "None",
7       "cost": 93.0100,
8       "cost_basis": "Actual",
9       "created_on": "2024-02-15T16:23:48",
10      "current_state@aras.name": "Preliminary",
11      "description": "Test part created for RESTful Services hands-on exercise
during ACE2024 by Paulo Costa. DO NOT DELETE.",
12      "generation": 5,
13      "has_change_pending": "0",
14      "id": "126C99CDC1944ECE853A70F7A64EE859",
15      "is_current": "1",
16      "is_released": "0",
17      "keyed_name": "ACE-9301-PC",
18      "major_rev": "A",
19      "make_buy": "Make",
20      "modified_on": "2024-02-27T15:36:32",
21      "name": "HTTP PATCH TEST",
22      "new_version": "0",
23      "not_lockable": "0",
24      "state": "Preliminary",
25      "unit": "EA",
26      "item_number": "ACE-9301-PC",
27      "itemtype": "4F1AC04A2B484F3ABA4E20DB63808A88"
28    },
29    {
30      "created_on": "2024-01-11T16:46:07",
31      "current_state@aras.name": "Preliminary",
32      "generation": 1,

```

## Get All Parts OrderBy

The HTTP GET method is used to retrieve items from the database. The addition of optional parameters will change the logic of the request, and consequently alter the presented result set.

GET <http://arastrainingvm/ACE2024VM/server/odata/Part>

Send

Try it:

### To retrieve items ordered by a specific property

1. To retrieve items ordered by a specific property, use the HTTP GET method, provide the ItemType name in the resource section of the OData URL, and append the option “?\$orderby=item\_number” at the end of the request.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen.

```

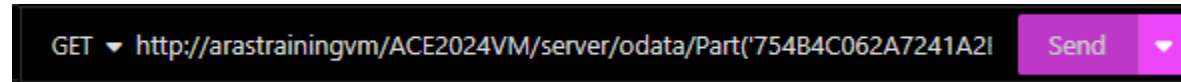
200 OK 324 ms 290.4 KB Just Now
Preview Headers 11 Cookies Timeline
1 {
2   "@odata.context":
   "http://arastrainingvm/ACE2024VM/server/odata/$metadata#Part",
3   "value": [
4     {
5       "classification": "Component",
6       "control_type": "None",
7       "cost": 93.0100,
8       "cost_basis": "Actual",
9       "created_on": "2024-02-15T16:23:48",
10      "current_state@aras.name": "Preliminary",
11      "description": "Test part created for RESTful Services hands-on exercise
during ACE2024 by Paulo Costa. DO NOT DELETE.",
12      "generation": 5,
13      "has_change_pending": "0",
14      "id": "126C99CDC1944ECE853A70F7A64EE859",
15      "is_current": "1",
16      "is_released": "0",
17      "keyed_name": "ACE-9301-PC",
18      "major_rev": "A",
19      "make_buy": "Make",
20      "modified_on": "2024-02-27T15:36:32",
21      "name": "HTTP PATCH TEST",
22      "new_version": "0",
23      "not_lockable": "0",
24      "state": "Preliminary",
25      "unit": "EA",
26      "item_number": "ACE-9301-PC",
27      "itemtype": "4F1AC04A2B484F3ABA4E20DB63808A88"
28    },
29   {
30     "created_on": "2024-01-11T16:46:07",
31     "current_state@aras.name": "Preliminary",

```



## Get Single Part by Id

The HTTP GET method is used to retrieve items from the database. The addition of optional parameters will change the logic of the request, and consequently alter the presented result set.



Try it:

### To retrieve a single item by its Id

1. To retrieve a single item by its identification, use the HTTP GET method, provide the ItemType name in the resource section of the OData URL, and append the item "id" as shown above.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen.

A screenshot of a REST client interface showing the response of a GET request. The top bar displays the status "200 OK", response time "165 ms", and size "818 B". Below the top bar are tabs for "Preview", "Headers", "Cookies", and "Timeline". The "Preview" tab is selected, showing a JSON response with the following fields:

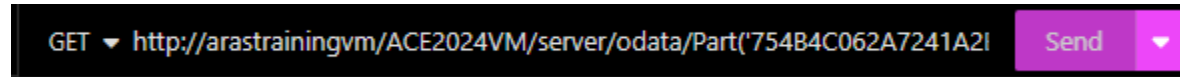
```

1 {
2   "@odata.context":
3     "http://arastrainingvm/ACE2024VM/server/odata/$metadata#Part/$entity",
4   "classification": "Component",
5   "control_type": "Serial",
6   "cost": 0.0200,
7   "cost_basis": "Actual",
8   "created_on": "2017-10-16T10:08:00",
9   "current_state@aras.name": "Released",
10  "description": "",
11  "effective_date": "2017-11-07T13:37:00",
12  "external_owner": "SofTech.Mechanical.SolidWorks",
13  "generation": 1,
14  "has_change_pending": "0",
15  "id": "754B4C062A7241A2B512BB0B96276F8E",
16  "is_current": "1",
17  "is_released": "1",
18  "keyed_name": "MP0190",
19  "major_rev": "A",
20  "make_buy": "Make",
21  "modified_on": "2017-11-07T13:37:00",
22  "name": "M3 Washer",
23  "new_version": "0",
24  "not_lockable": "0",
25  "release_date": "2017-11-07T13:37:00",
26  "state": "Released",
27  "thumbnail": "vault:///fileId=EF01183252BB408CB12F3AE4F3D816DA",
28  "unit": "EA",
29  "item_number": "MP0190",
30  "itemtype": "4F1AC04A2B484F3ABA4E20DB63808A88"
31 }

```

## Get Single Property of a Single Part

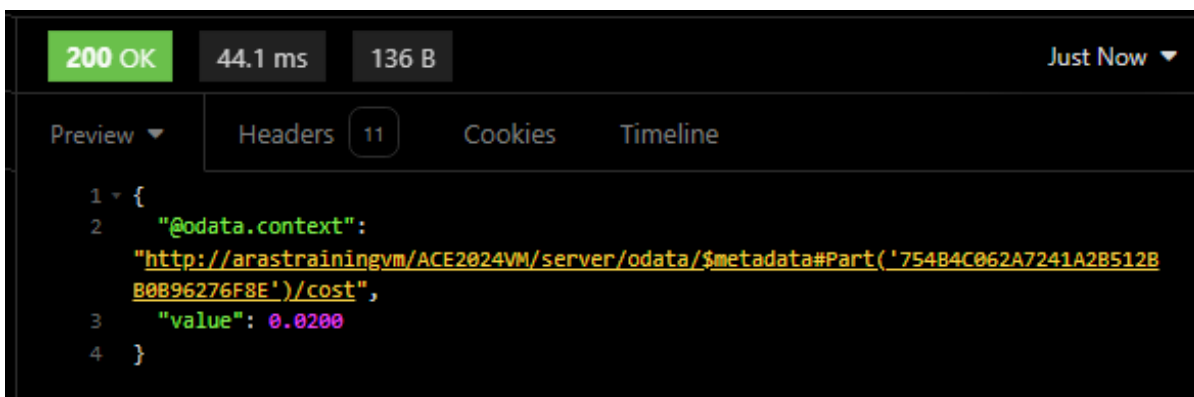
The HTTP GET method is used to retrieve items from the database. The addition of optional parameters will change the logic of the request, and consequently alter the presented result set.



Try it:

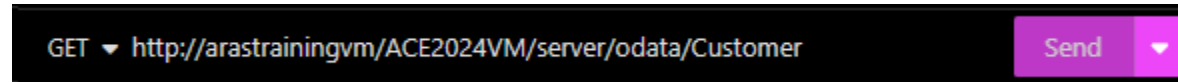
### To retrieve a single property of a single item

1. To retrieve a single property of a single item, use the HTTP GET method, provide the ItemType name in the resource section of the OData URL, and append both the item id and the option that indicates the name of the desired property to be retrieved as shown above.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen.



## Get Single Property of All Items and Total Count

The HTTP GET method is used to retrieve items from the database. The addition of optional parameters will change the logic of the request, and consequently alter the presented result set.



Try it:

### To retrieve a single property of all items and the total count

1. To retrieve a single property of a single item, use the HTTP GET method, provide the ItemType name in the resource section of the OData URL, append the options for both total count and select the desired properties to be returned as shown above.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen.

A screenshot of a REST client interface showing the response of the GET request. The top bar displays '200 OK', '6.14 s', and '1041 B'. Below the bar are tabs for 'Preview', 'Headers', 'Cookies', and 'Timeline'. The 'Preview' tab is active, showing a JSON response with 41 lines of code. The response is a JSON object with an '@odata.context' property, a '@odata.count' property, and a 'value' array containing six customer objects. Each object has '@odata.id', 'name', and 'itemtype' properties.

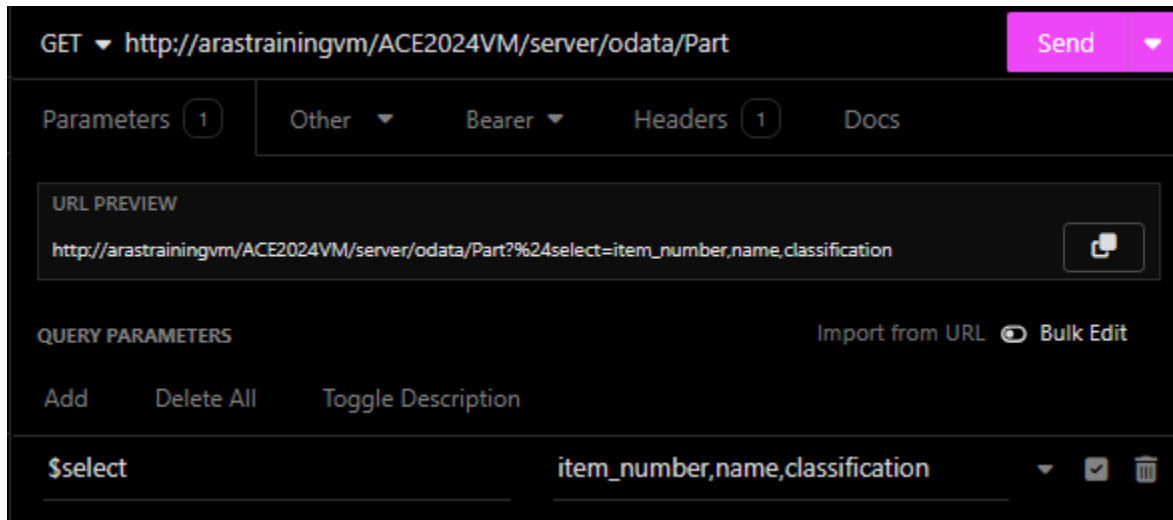
```

1 - {
2   "@odata.context":
3   "http://arastrainingvm/ACE2024VM/server/odata/$metadata#Customer(name)",
4   "@odata.count": 7,
5   "value": [
6     {
7       "@odata.id": "Customer('F787E0F27612445EB1FFAA2D2D01E086')",
8       "name": "DaimlerChrysler",
9       "itemtype": "E4847F4A706B4285A395C0228A867E0F"
10    },
11   {
12     "@odata.id": "Customer('CA2769A1B85B4AC38484E1034D260808')",
13     "name": "Delphi Corporation",
14     "itemtype": "E4847F4A706B4285A395C0228A867E0F"
15   },
16   {
17     "@odata.id": "Customer('E0078E62C53348D7A4E1839A36C99D94')",
18     "name": "Ford Motor Company",
19     "itemtype": "E4847F4A706B4285A395C0228A867E0F"
20   },
21   {
22     "@odata.id": "Customer('FCBBDAC13D5E4EA29E1975552284D27D')",
23     "name": "General Motors",
24     "itemtype": "E4847F4A706B4285A395C0228A867E0F"
25   },
26   {
27     "@odata.id": "Customer('36CC5562C8FB4DA08D9DC4F1A07EF970')",
28     "name": "Honda of America",
29     "itemtype": "E4847F4A706B4285A395C0228A867E0F"
30   },
31   {
32     "@odata.id": "Customer('634059758D0148D083527D7C9E482242')",
33     "name": "Lear Corporation",
34     "itemtype": "E4847F4A706B4285A395C0228A867E0F"
35   },
36   {
37     "@odata.id": "Customer('C6B487B638304F1EBB5A5CC36B59633B')",
38     "name": "Wal-Mart",
39     "itemtype": "E4847F4A706B4285A395C0228A867E0F"
40   }
41  ]

```

## Get Selected Properties of All Items

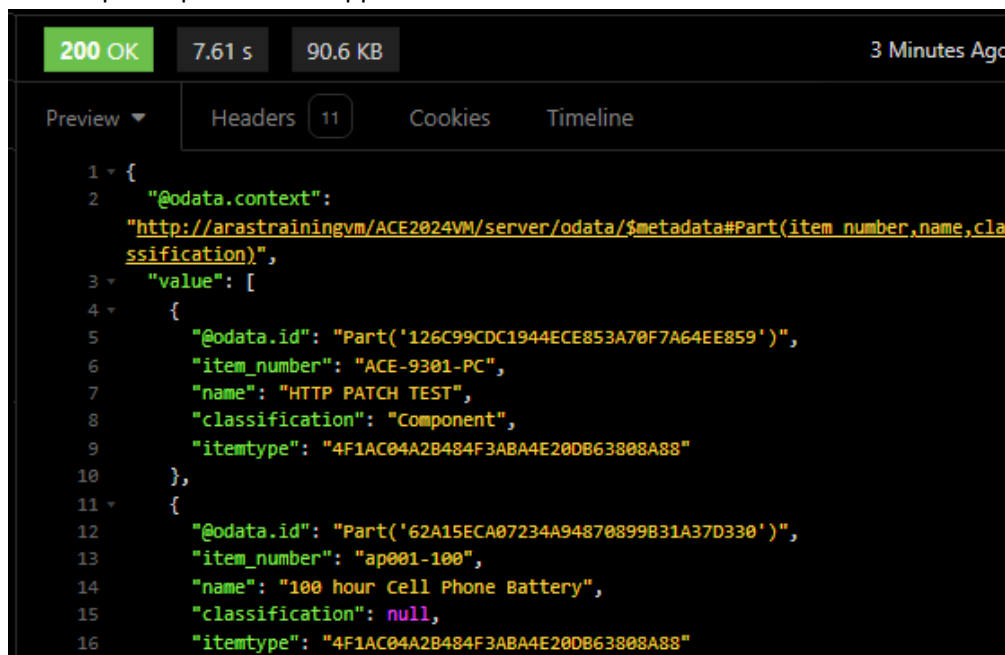
The HTTP GET method is used to retrieve items from the database. The addition of optional parameters will change the logic of the request, and consequently alter the presented result set.



Try it:

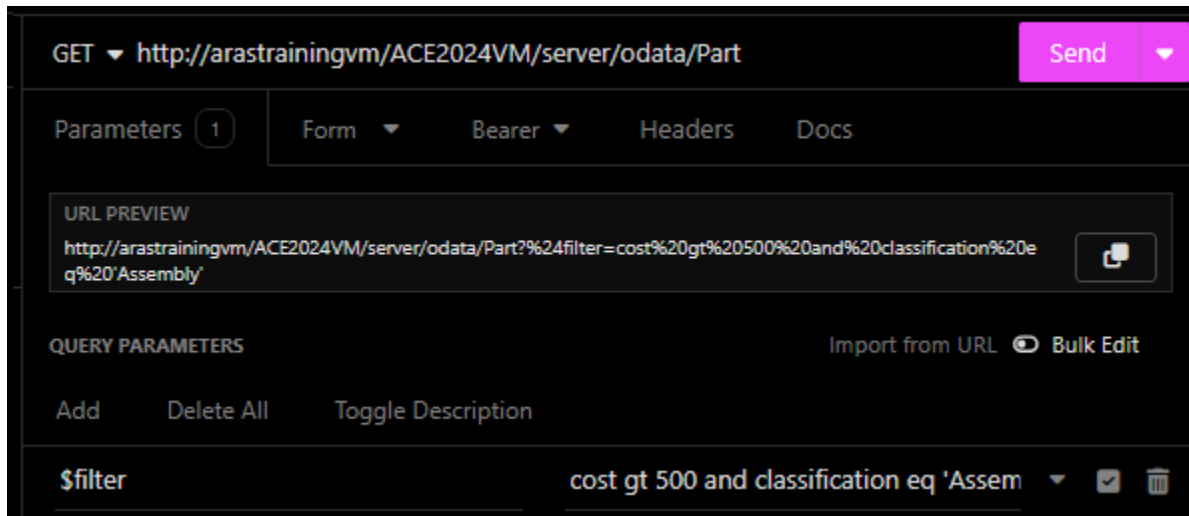
### To retrieve selected properties of all items

1. To retrieve selected properties of all items, use the HTTP GET method, provide the ItemType name in the resource section of the OData URL, and append the option to select the desired properties to be returned as shown above.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen.



## Get All Items with Matching Selected Properties

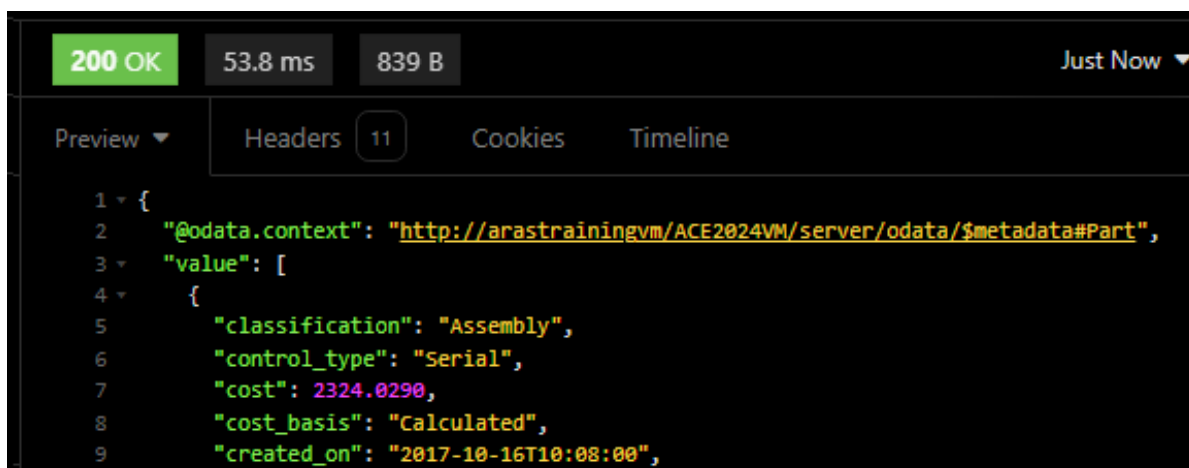
The HTTP GET method is used to retrieve items from the database. The addition of optional parameters will change the logic of the request, and consequently alter the presented result set.



Try it:

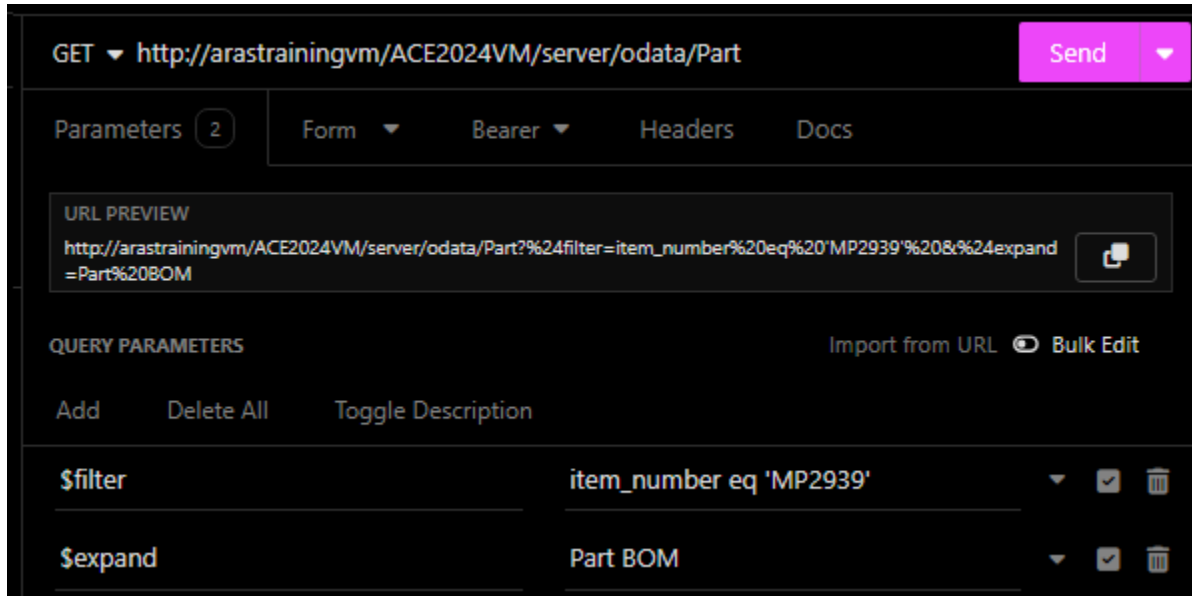
### To retrieve all items with matching selected properties

1. To retrieve selected properties of all items, use the HTTP GET method, provide the ItemType name in the resource section of the OData URL, and append both the option to filter on select the desired properties and conditions to be returned as shown above.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen.



## Get Item and Expand on Relationships

The HTTP GET method is used to retrieve items from the database. The addition of optional parameters will change the logic of the request, and consequently alter the presented result set.



**Try it:**

### To retrieve an item and expand its relationships

1. To retrieve selected properties of all items, use the HTTP GET method, provide the ItemType name in the resource section of the OData URL, and append both options to filter on the selected item and expand the desired relationship ItemType as shown above.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen.

#### **NOTE:**

See screenshot of partial result set on next page.

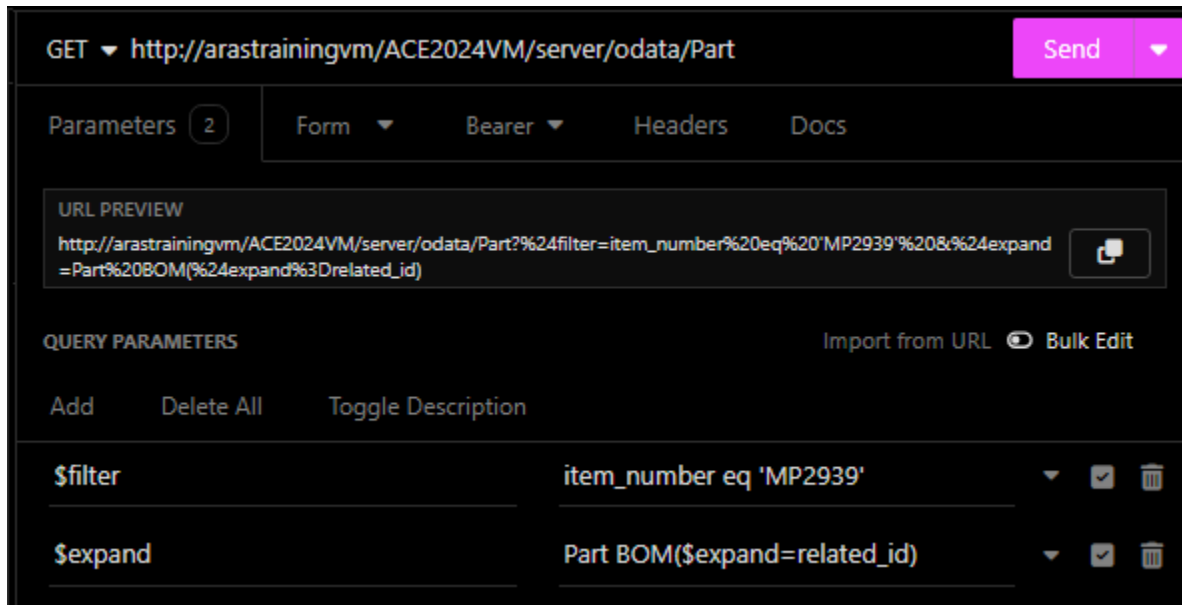
```

200 OK 482 ms 1711 B Just Now
Preview Headers 11 Cookies Timeline
1 {
2   "@odata.context": "http://arastrainingvm/ACE2024VM/server/odata/$metadata#Part",
3   "value": [
4     {
5       "classification": "Assembly",
6       "control_type": "Serial",
7       "cost": 9.9000,
8       "cost_basis": "Calculated",
9       "created_on": "2017-10-16T10:08:00",
10      "current_state@aras.name": "Released",
11      "description": "",
12      "effective_date": "2017-11-07T13:37:00",
13      "external_owner": "SofTech.Mechanical.SolidWorks",
14      "generation": 1,
15      "has_change_pending": "0",
16      "id": "9EF3CD4874C6451E8166FFF26058FB71",
17      "is_current": "1",
18      "is_released": "1",
19      "keyed_name": "MP2939",
20      "major_rev": "A",
21      "make_buy": "Make",
22      "modified_on": "2017-11-07T13:37:00",
23      "name": "Body Fan Assembly",
24      "new_version": "0",
25      "not_lockable": "0",
26      "release_date": "2017-11-07T13:37:00",
27      "state": "Released",
28      "thumbnail": "vault:///}?fileId=89FF05DF6ED94DBCBE2549A051C58501",
29      "unit": "EA",
30      "item_number": "MP2939",
31      "Part BOM": [
32        {
33          "behavior": "fixed",
34          "created_on": "2017-10-24T15:09:00",
35          "external_owner": "SofTech.Mechanical.SolidWorks",
36          "generation": 1,
37          "id": "A2F6477F1FA64E36A36FB7CA76CD7D88",
38          "is_current": "1",
39          "is_released": "0",
40          "keyed_name": "A2F6477F1FA64E36A36FB7CA76CD7D88",
41          "major_rev": "A",
42          "modified_on": "2017-10-24T15:09:00",
43          "new_version": "1",
44          "not_lockable": "0",
45          "quantity": 1,
46          "reference_designator": "",
47          "sort_order": 5,
48          "itemtype": "5E9C5A12CC58413A8670CF4003C57848"
49        },

```

## Get Item and Expand on Relationships and Related Items

The HTTP GET method is used to retrieve items from the database. The addition of optional parameters will change the logic of the request, and consequently alter the presented result set.



**Try it:**

### To retrieve an item and expand its relationships and related items

1. To retrieve selected properties of all items, use the HTTP GET method, provide the ItemType name in the resource section of the OData URL, append both query parameters to filter on the selected item and expand the desired relationship ItemType and related items as shown above.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen.

#### NOTE:

See screenshot of partial result set on next page.



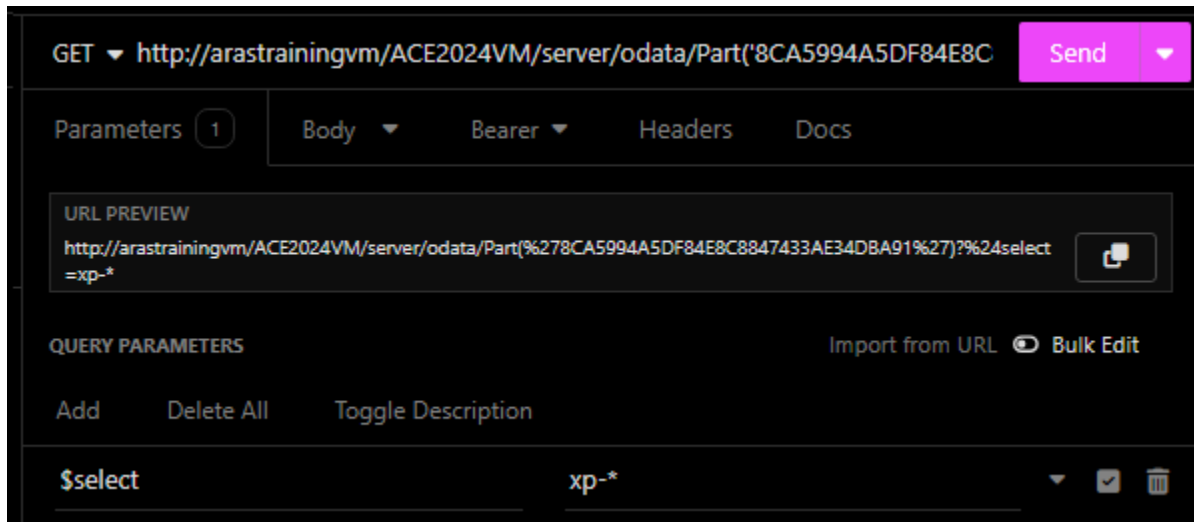
```

200 OK    71.6 ms    3.1 KB    Just Now
Preview ▾ Headers 11 Cookies Timeline
1 {
2   "@odata.context": "http://arastrainingvm/ACE2024VM/server/odata/$metadata#Part",
3   "value": [
4     {
5       "classification": "Assembly",
6       "control_type": "Serial",
7       "cost": 9.9000,
8       "cost_basis": "Calculated",
9       "created_on": "2017-10-16T10:08:00",
10      "current_state@aras.name": "Released",
11      "description": "",
12      "effective_date": "2017-11-07T13:37:00",
13      "external_owner": "SofTech.Mechanical.Solidworks",
14      "generation": 1,
15      "has_change_pending": "0",
16      "id": "9EF3CD4874C6451E8166FFF2605BFB71",
17      "is_current": "1",
18      "is_released": "1",
19      "keyed_name": "MP2939",
20      "major_rev": "A",
21      "make_buy": "Make",
22      "modified_on": "2017-11-07T13:37:00",
23      "name": "Body Fan Assembly",
24      "new_version": "0",
25      "not_lockable": "0",
26      "release_date": "2017-11-07T13:37:00",
27      "state": "Released",
28      "thumbnail": "vault:///?fileId=89FF05DF6ED94DBCBE2549A051C5B501",
29      "unit": "EA",
30      "item_number": "MP2939",
31      "Part BOM": [
32        {
33          "behavior": "fixed",
34          "created_on": "2017-10-24T15:09:00",
35          "external_owner": "SofTech.Mechanical.Solidworks",
36          "generation": 1,
37          "id": "A2F6477F1FA64E36A36FB7CA76CD7D88",
38          "is_current": "1",
39          "is_released": "0",
40          "keyed_name": "A2F6477F1FA64E36A36FB7CA76CD7D88",
41          "major_rev": "A",
42          "modified_on": "2017-10-24T15:09:00",
43          "new_version": "1",
44          "not_lockable": "0",
45          "quantity": 1,
46          "reference_designator": "",
47          "related_id": {
48            "classification": "Component",
49            "control_type": "Serial",
50            "cost": 9.0000,
51            "cost_basis": "Actual",

```

## Get All Extended Properties of an Items

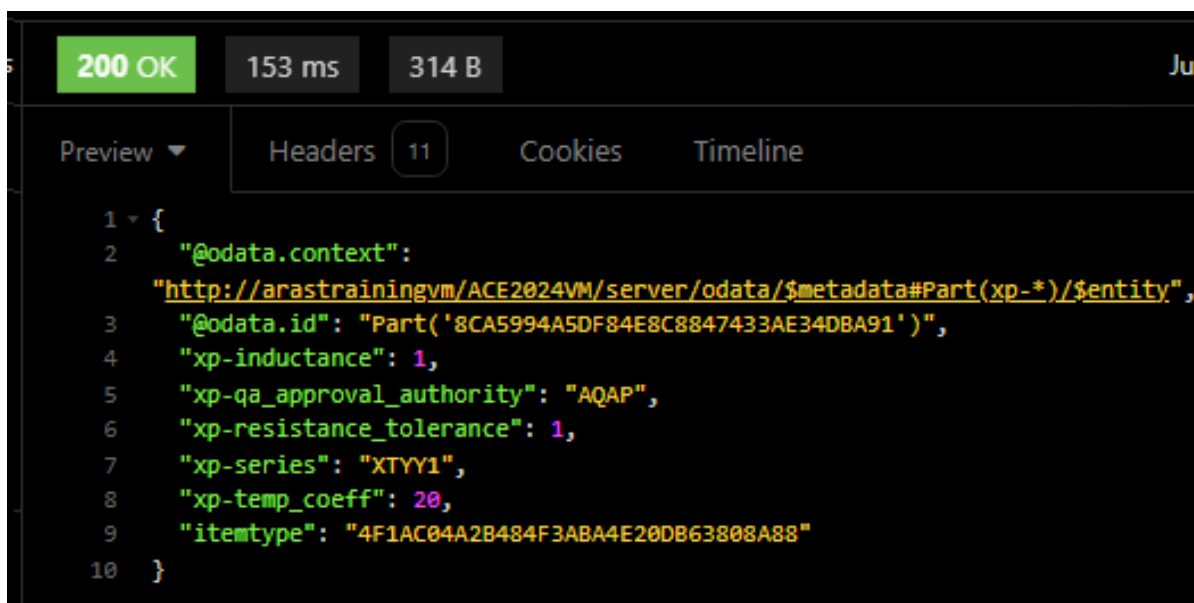
The HTTP GET method is used to retrieve items from the database. The addition of optional parameters will change the logic of the request, and consequently alter the presented result set.



Try it:

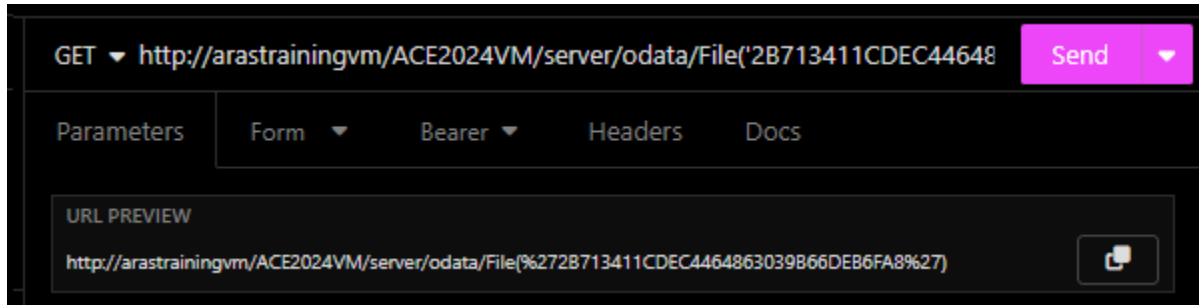
### To retrieve all extended properties of an item

1. To retrieve selected properties of all items, use the HTTP GET method, provide the ItemType name in the resource section of the OData URL, and append all query parameters to filter on the selected item and desired multiple “xp-“ properties as shown above.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen.



## Get All Properties of a File Item

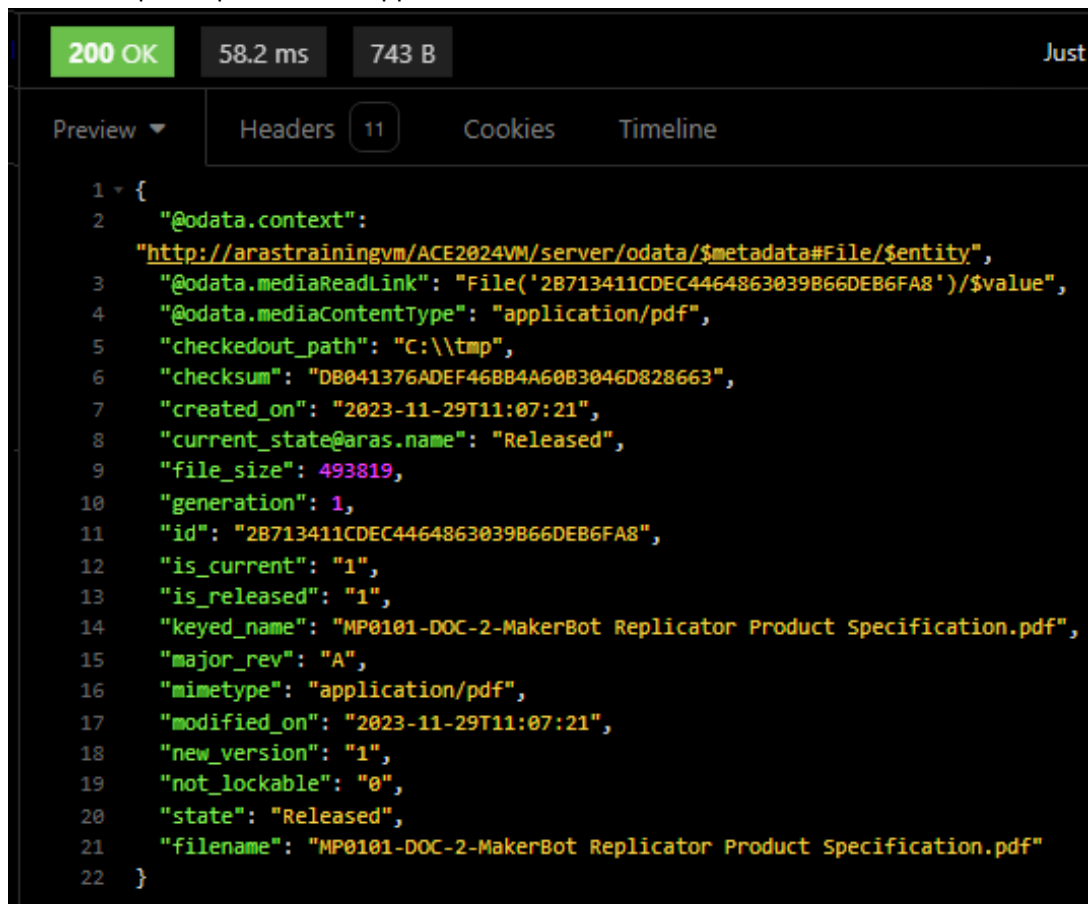
The HTTP GET method is used to retrieve items from the database. The addition of optional parameters will change the logic of the request, and consequently alter the presented result set.



Try it:

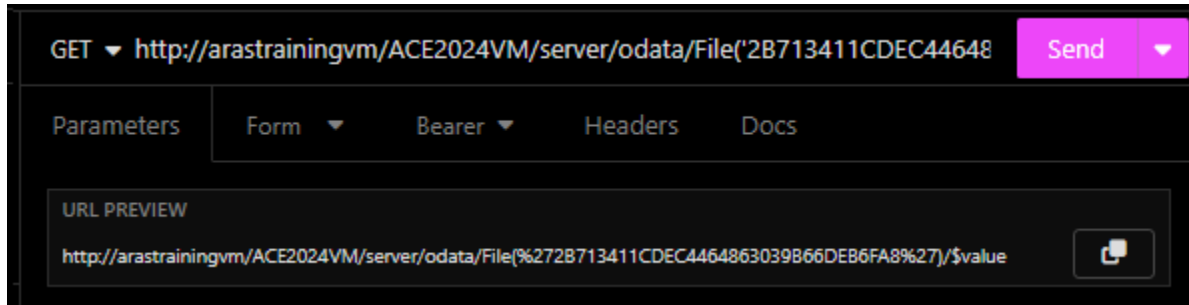
### To retrieve all properties of a File Item

1. To retrieve all properties of a File item, use the HTTP GET method and provide the unique Id of the desired file item to explore as shown above.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen.



## Get the physical File item

The HTTP GET method is used to retrieve items from the database. The addition of optional parameters will change the logic of the request, and consequently alter the presented result set.



Try it:

### To retrieve the physical File item

1. To retrieve the physical file item itself use the HTTP GET method and in addition provide the unique Id of the desired file item to explore, as well as the option "\$value" as shown above.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen.


#### NOTE:

See screenshot of partial result set on next page.


200 OK
3.61 s
482.2 KB
Just Now ▾

Preview ▾
Headers 12
Cookies
Timeline


☰
1 / 11
-
+
📄
🔄
⬇️
🖨️
⋮




**1**



**2**



**3**



**4**

MakerBot INDUSTRIES

DOCUMENT NUMBER: MP1017-DOC-2 Page 1 of 11

TITLE: MakerBot Replicator Product Specification

REVISION: 0.0

AUTHOR: Dave Perault

## MakerBot Replicator Product Specification

**Related Issues**

Issue Number	Description
MP1017-001	Creating MakerBot Replicator 3D printer

**Change History**

Date	Author(s)	Comments
2010-01-01	Dave Perault	Initial version
2010-01-01	Dave Perault	Added sections 3 and 4
2010-01-01	Dave Perault	Update for redesign due to 3D card interference

Morceaux placés à la fin du préambule, et avant votre table des matières

MakerBot INDUSTRIES

DOCUMENT NUMBER: MP1017-DOC-2 Page 2 of 11

TITLE: MakerBot Replicator Product Specification

REVISION: 0.0

AUTHOR: Dave Perault

### Contents

- 1 Overview ..... 3
- 1.1 Purpose ..... 3
- 1.2 Audience ..... 3
- 1.3 Assumptions ..... 3
- 1.3.1 Lignes à suivre pour et avant, connecteur adaptateur etc.
- 1.3.2 In ou à la fin de l'assemblage, broches à souder, matériel etc.
- 1.4 Dépendances ..... 3
- 1.4.1 Outils et matériel requis avant usage
- 1.5 Terminologie ..... 3
- 2 Business Requirements ..... 4
- 2.1 Requirements ..... 4
- 2.1.1 Process et état des réglages, paramètres et état in, fin de vie
- 2.1.2 Matériaux placés à la fin du préambule
- 2.2 Use Cases ..... 5
- 2.2.1 Procédures à suivre et à éviter
- 3 Pratiques ..... 6
- 3.1 Nulla et du libendum ..... 6
- 3.2 Maecenas placerat felis et du pretium
- 3.3 Etiam pellentesque sem hendrerit
- 3.4 Nulla et du libendum ..... 7
- 3.5 Maecenas placerat felis et du pretium
- 3.6 Etiam pellentesque sem hendrerit
- 3.7 Nulla et du libendum ..... 7
- 3.8 Maecenas placerat felis et du pretium
- 3.9 Vestibulum eu nunc vitae
- 3.10 Nulla et du libendum ..... 8
- 3.11 Maecenas placerat felis et du pretium
- 4 Product Readiness ..... 11
- 4.1 Testing ..... 11
- 4.2 Performance ..... 11
- 4.3 Documentation ..... 11
- 4.4 Installation ..... 11
- 4.5 Upgrade ..... 11

Morceaux placés à la fin du préambule, et avant votre table des matières

MakerBot INDUSTRIES

DOCUMENT NUMBER: MP1017-DOC-2 Page 3 of 11

TITLE: MakerBot Replicator Product Specification


REVISION: 0.0

AUTHOR: Dave Perault

### 1 Overview

## Add New Item

The HTTP POST method is used to add items to the database. The addition of optional parameters and supplemental elements will change the logic of the request, and consequently alter the behavior to be adopted by the server.



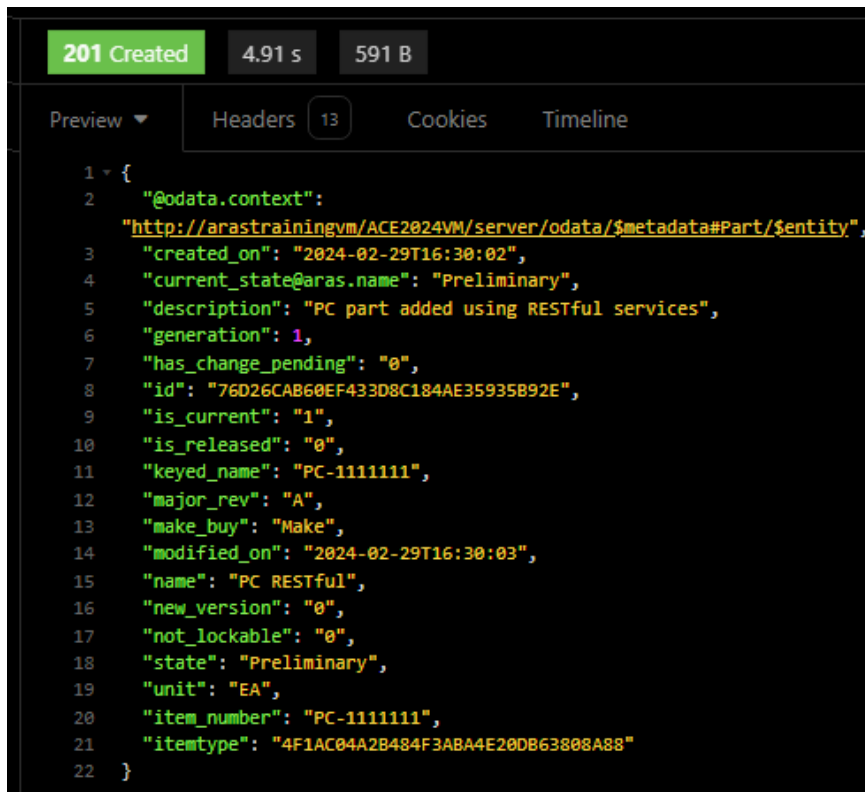
```

POST http://arastrainingvm/ACE2024VM/server/odata/Part
Parameters JSON Bearer Headers 1 Docs
1 {
2   "item_number": "PC-1111111",
3   "name": "PC RESTful",
4   "description": "PC part added using RESTful services"
5 }
  
```

Try it:

### To add a new item

1. To add a new item to the database, use the HTTP POST method, provide the ItemType, and include elements to associate values to the item properties, as seen on the Body tab of the client interface.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen.



```

201 Created 4.91 s 591 B
Preview Headers 13 Cookies Timeline
1 {
2   "@odata.context":
3     "http://arastrainingvm/ACE2024VM/server/odata/$metadata#Part/$entity",
4   "created_on": "2024-02-29T16:30:02",
5   "current_state@aras.name": "Preliminary",
6   "description": "PC part added using RESTful services",
7   "generation": 1,
8   "has_change_pending": "0",
9   "id": "76D26CAB60EF433D8C184AE35935B92E",
10  "is_current": "1",
11  "is_released": "0",
12  "keyed_name": "PC-1111111",
13  "major_rev": "A",
14  "make_buy": "Make",
15  "modified_on": "2024-02-29T16:30:03",
16  "name": "PC RESTful",
17  "new_version": "0",
18  "not_lockable": "0",
19  "state": "Preliminary",
20  "unit": "EA",
21  "item_number": "PC-1111111",
22  "itemtype": "4F1AC04A2B484F3ABA4E20DB63808A88"
  }
  
```

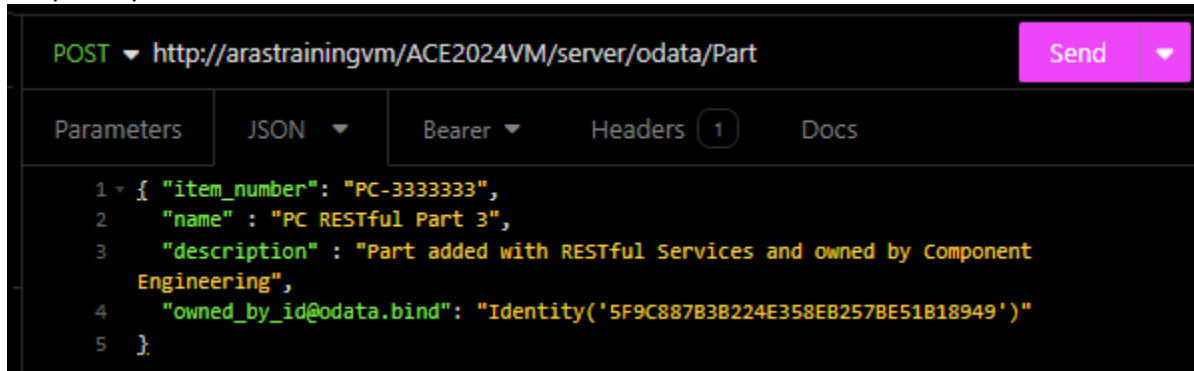
Additional verification may be performed directly on the Innovator client with a new search on Parts.

The screenshot shows the Aras Innovator web interface. At the top, there is a search bar with the text 'aras INNOVATOR' and a search icon. Below the search bar, there is a breadcrumb trail 'Parts' with a close icon. The main navigation area includes a 'Parts' menu item with a gear icon and a star icon. Below this, there are search controls: a 'Search' button, a 'Clear' button, a 'Simple' dropdown menu, a 'Current' dropdown menu, a date field set to 'Today', and a 'Default \*' dropdown menu. The main content area displays a table with the following columns: Part Number ↑, Revi..., Name, Type, State, Cost, Chan..., and Indexed On [...]. The table contains one row with the following data: PC\*, A, PC RESTful, Preliminary, and an unchecked checkbox. The row is highlighted in yellow.

Part Number ↑	Revi...	Name	Type	State	Cost	Chan...	Indexed On [...]
PC*							
PC-1111111	A	PC RESTful		Preliminary			<input type="checkbox"/>

## Add New Item with Reference to Another Item

The HTTP POST method is used to add items to the database. The addition of optional parameters and supplemental elements will change the logic of the request, and consequently alter the behavior to be adopted by the server.



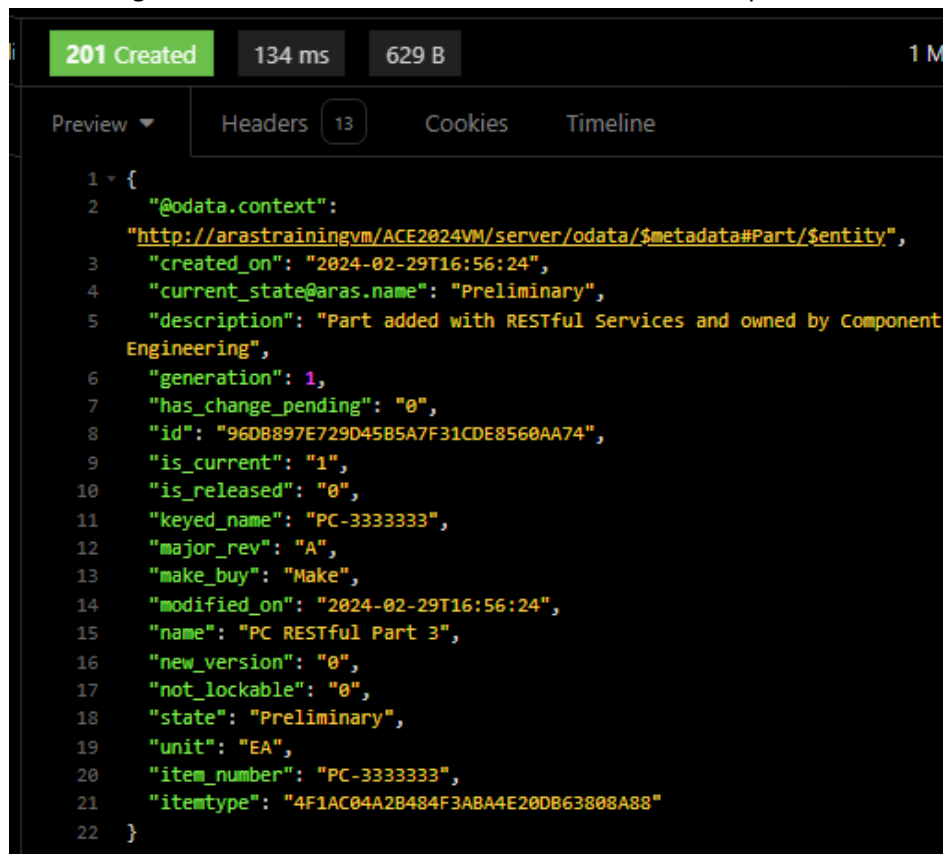
```

POST http://arastrainingvm/ACE2024VM/server/odata/Part
Parameters JSON Bearer Headers 1 Docs
1 { "item_number": "PC-3333333",
2   "name" : "PC RESTful Part 3",
3   "description" : "Part added with RESTful Services and owned by Component
   Engineering",
4   "owned_by_id@odata.bind": "Identity('5F9C887B3B224E358EB257BE51B18949')",
5 }
  
```

Try it:

### To add a new item with reference to another item

1. To add a new item to the database and establish a reference to an existing item in the system, use the HTTP POST method, provide the ItemType, and include elements to bind properties to already existing items in Innovator, as seen on the Body tab of the client interface.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen. Additional verification may be directly performed using the Aras Innovator web browser-based client component.



```

201 Created 134 ms 629 B 1 M
Preview Headers 13 Cookies Timeline
1 {
2   "@odata.context":
3   "http://arastrainingvm/ACE2024VM/server/odata/$metadata#Part/$entity",
4   "created_on": "2024-02-29T16:56:24",
5   "current_state@aras.name": "Preliminary",
6   "description": "Part added with RESTful Services and owned by Component
   Engineering",
7   "generation": 1,
8   "has_change_pending": "0",
9   "id": "96DB897E729D45B5A7F31CDE8560AA74",
10  "is_current": "1",
11  "is_released": "0",
12  "keyed_name": "PC-3333333",
13  "major_rev": "A",
14  "make_buy": "Make",
15  "modified_on": "2024-02-29T16:56:24",
16  "name": "PC RESTful Part 3",
17  "new_version": "0",
18  "not_lockable": "0",
19  "state": "Preliminary",
20  "unit": "EA",
21  "item_number": "PC-3333333",
22  "itemtype": "4F1AC04A2B484F3ABA4E20DB63808A88"
}
  
```



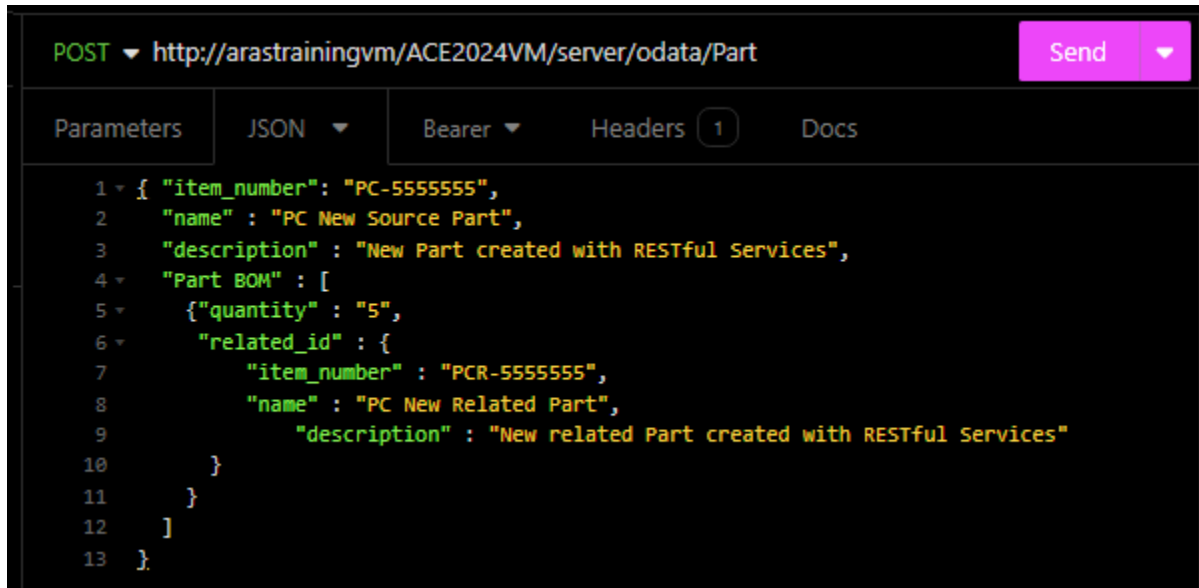
Note the newly created item on the corresponding Aras Innovator Client Item View form and verify the allocation of the Assigned Creator to the Component Engineering identity.

The screenshot displays the Aras Innovator interface for a Part record. The header shows the Aras INNOVATOR logo and a search bar. Below the header, there are tabs for 'Parts' and 'PC-3333333'. The main content area shows the details for the Part 'PC-3333333'. The record includes the following fields:

Part Number	Revision	State	Assigned Creator	
PC-3333333	A	Preliminary	<a href="#">Component Engineering</a>	
Name	Designated User			
PC RESTful Part 3				
Type	Unit	Make / Buy	Cost	Effective Date
	EA	Make		
Long Description	Part added with RESTful Services and owned by Component Engineering			

## Add New Item Related to Another New Item

The HTTP POST method is used to add items to the database. The addition of optional parameters and supplemental elements will change the logic of the request, and consequently alter the behavior to be adopted by the server.



```

1  {
2    "item_number": "PC-5555555",
3    "name" : "PC New Source Part",
4    "description" : "New Part created with RESTful Services",
5    "Part BOM" : [
6      {
7        "quantity" : "5",
8        "related_id" : {
9          "item_number" : "PCR-5555555",
10         "name" : "PC New Related Part",
11         "description" : "New related Part created with RESTful Services"
12       }
13     ]
14   }

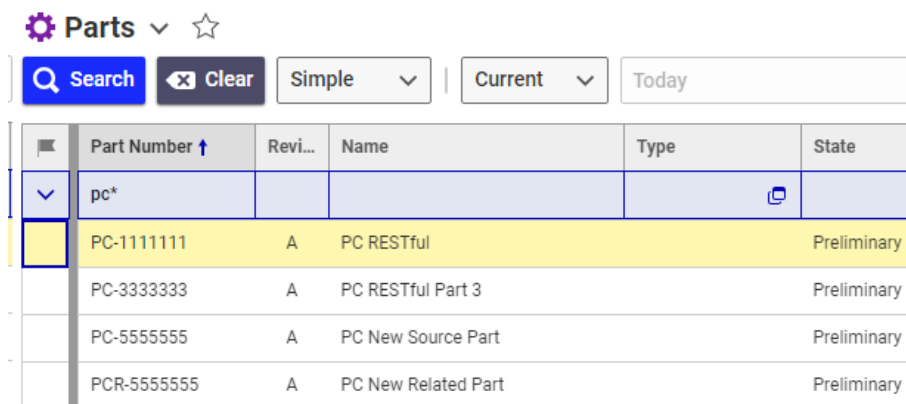
```

Note the JSON constructor built in the Body tab to indicate both the source and related items in this new relationship.

### Try it:

#### To add a new item related to another new item

1. To add two new items to the database and establish a relationship between the two, use the HTTP POST method, provide the ItemType, and include elements to bind them in a new relationship, as seen above on the Body tab of the requester client interface.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen. Additional verification may be directly performed using the Aras Innovator web browser-based client component.



Parts

Search Clear Simple Current Today

Part Number ↑	Revi...	Name	Type	State
pc*				
PC-1111111	A	PC RESTful		Preliminary
PC-3333333	A	PC RESTful Part 3		Preliminary
PC-5555555	A	PC New Source Part		Preliminary
PCR-5555555	A	PC New Related Part		Preliminary

**NOTE:** See screenshot of partial result set on next page.

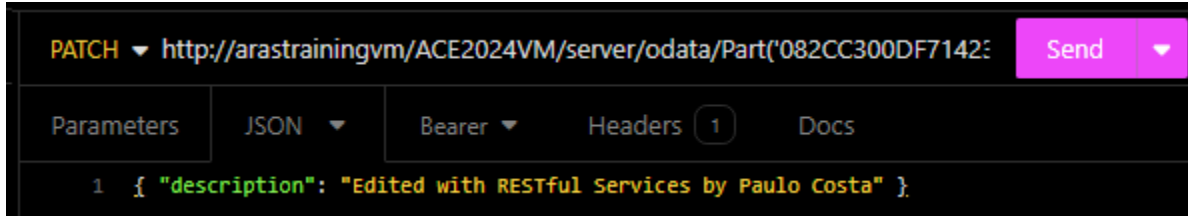
```

201 Created 515 ms 1512 B
Preview ▾ Headers 13 Cookies Timeline
1 {
2   "@odata.context":
3   "http://arastrainingvm/ACE2024VM/server/odata/\$metadata#Part/\$entity",
4   "created_on": "2024-02-29T17:11:30",
5   "current_state@aras.name": "Preliminary",
6   "description": "New Part created with RESTful Services",
7   "generation": 1,
8   "has_change_pending": "0",
9   "id": "C3BB007D74254E50B54ED12DC0764F9B",
10  "is_current": "1",
11  "is_released": "0",
12  "keyed_name": "PC-555555",
13  "major_rev": "A",
14  "make_buy": "Make",
15  "modified_on": "2024-02-29T17:11:30",
16  "name": "PC New Source Part",
17  "new_version": "0",
18  "not_lockable": "0",
19  "state": "Preliminary",
20  "unit": "EA",
21  "item_number": "PC-555555",
22  "Part BOM": [
23    {
24      "behavior": "float",
25      "created_on": "2024-02-29T17:11:30",
26      "generation": 1,
27      "id": "9959F6DC46494E03874E519B184ED194",
28      "is_current": "1",
29      "is_released": "0",
30      "keyed_name": "9959F6DC46494E03874E519B184ED194",
31      "major_rev": "A",
32      "modified_on": "2024-02-29T17:11:30",
33      "new_version": "1",
34      "not_lockable": "0",
35      "quantity": 5,
36      "related_id": {
37        "created_on": "2024-02-29T17:11:30",
38        "current_state@aras.name": "Preliminary",
39        "description": "New related Part created with RESTful Services",
40        "generation": 1,
41        "has_change_pending": "0",
42        "id": "AB3EF72795724283803E00E77A89FE5E",
43        "is_current": "1",
44        "is_released": "0",
45        "keyed_name": "PCR-555555",
46        "major_rev": "A",
47        "make_buy": "Make",
48        "modified_on": "2024-02-29T17:11:30",
49        "name": "PC New Related Part",
50        "new_version": "0",

```

## Edit an Item Property

The HTTP PATCH method is used to edit items in the database. The addition of optional parameters and supplemental elements will change the logic of the request, and consequently alter the behavior to be adopted by the server.



Note the constructor built in the Body tab to indicate both the item property and associated value to be edited on the selected item.

Try it:

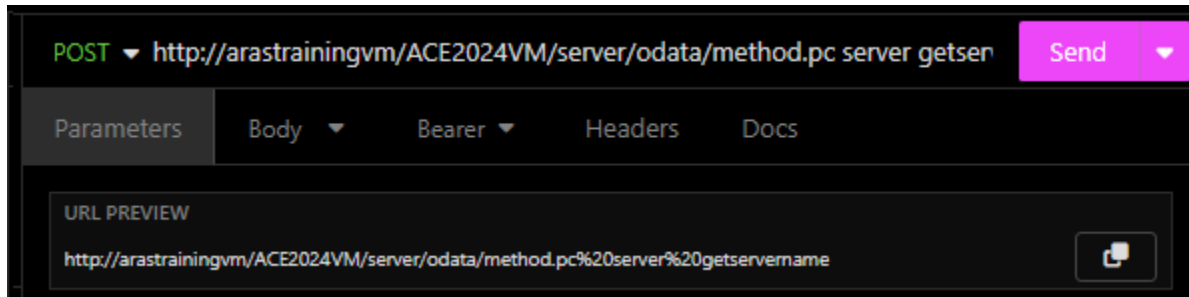
### To edit an item property

1. To edit an item property, use the HTTP PATCH method, provide the ItemType, and define the unique item to be edited. Include elements to the Body tab to indicate both the property name and associated value to be edited on the selected item, as seen on the Body tab of the client interface.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen. Additional verification may be directly performed using the Aras Innovator web browser-based client component.

Part				
Part Number	Revision	State	Assigned Creator	
ACE-9301-PC	A	Preliminary	<a href="#">Innovator Admin</a>	
Name	Designated User			
HTTP PATCH TEST	<a href="#">Innovator Admin</a>			
Type	Unit	Make / Buy	Cost (Actual)	Effective Date
Component	EA	Make	93.0100	
Long Description	Edited with RESTful Services by Paulo Costa			

## Running an Aras Innovator server-side method

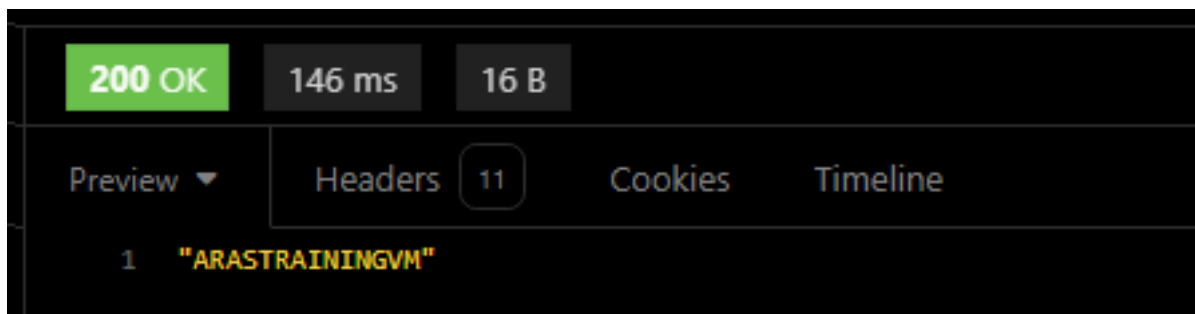
The HTTP POST method is used to instruct the server to execute server-side methods already present in the Aras Innovator database. Note the use of dot (.) notation to link the requested resource to the desired server-side method name, as saved in the Aras Innovator database. The addition of optional parameters and supplemental elements will change the logic of the request, and consequently alter the behavior to be adopted by the server.



Try it:

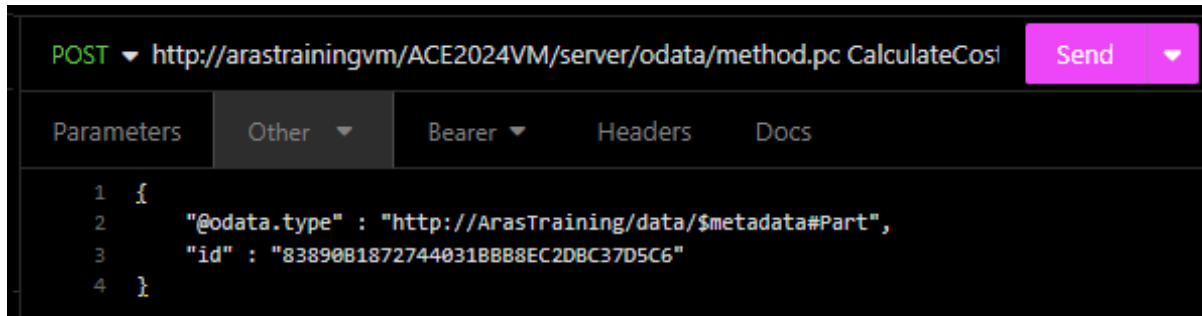
### To run an Aras Innovator server-side method

1. To run an Aras Innovator server-side method, use the HTTP POST method, provide the resource to be utilized on the request, and include the name of the selected server-side method to be executed, using the dot (.) notation to link the elements of your request.
2. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen.



## Running an Aras Innovator server-side method on an Item

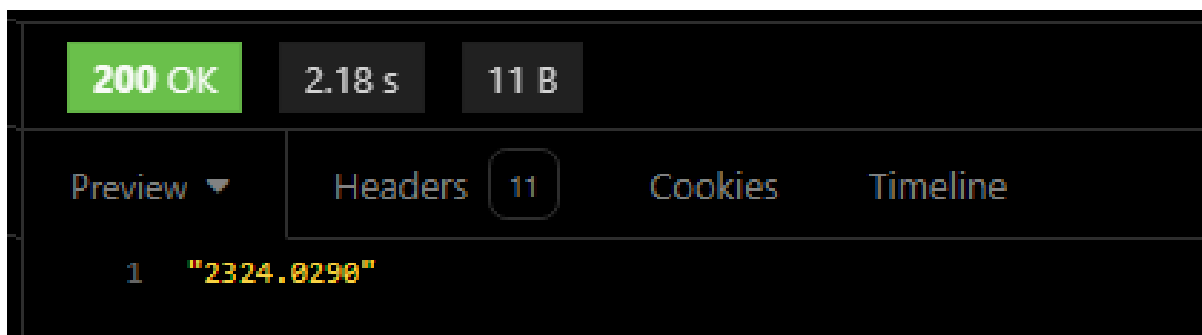
The HTTP POST method is used to instruct the server to execute server-side methods already present in the Aras Innovator database. Note the use of dot (.) notation to link the requested resource to the desired server-side method name, as saved in the Aras Innovator database. The addition of optional parameters and supplemental elements will change the logic of the request, and consequently alter the behavior to be adopted by the server.



Try it:

### To run an Aras Innovator server-side method

1. To run an Aras Innovator server-side method, use the HTTP POST method and in addition to providing the resource to be utilized on the request, include the name of the selected server-side method to be executed, using the dot (.) notation to link the elements of your request.
2. Add the reference of the item to be processed by the selected server-side method to be executed by the Aras Innovator server with the construct included in the Body tab, as seen above.
3. Click the Send button to process the request. The response (items or errors) will appear in the server response panel of the application screen.



## Summary

In this session, we discussed several aspects of the Aras RESTful Services API utilization. As a result, you should be able to:

- Understand the RESTful architecture using OData.
- Use HTTP methods in OData formatted instructions to:
  1. Retrieve Items and properties values.
  2. Add new Items.
  3. Edit existing Items.
  4. Execute Aras Innovator server-side methods.