



Aras Innovator 2024

Release

RESTful API

Document #: D-008126

Last Modified: 5/20/2024

Copyright Information

Copyright © 2023 Aras Corporation. All Rights Reserved.

Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810
Phone: 978-691-8900

E-mail: support@aras.com

Website: <https://www.aras.com/>

Notice of Rights

Copyright © 2023 by Aras Corporation and/or its affiliates. All rights reserved.

This document is protected by U.S. and international copyright laws and conventions. No copyright may be obscured or removed from this document. This document may not be modified or altered, or reproduced or transmitted in any form, without the explicit permission of 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.

Notice of Liability

THIS DOCUMENT IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY, AND THE CONTENTS HEREOF ARE SUBJECT TO CHANGE WITHOUT NOTICE. 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.

Table of Contents

Introduction	5
1 Aras Innovator OData Interface.....	6
1.1 Request Translation	6
1.1.1 OData Request URL Format	6
1.1.2 Entities and Properties	6
1.1.3 Item properties.....	7
1.1.4 Relationships	9
1.1.5 File Properties	10
1.1.6 PolyItems	11
1.1.7 Extended Properties	11
1.1.8 Get.....	11
1.1.9 Define operations	11
1.1.10 Update operation	12
1.1.11 Change permission operation	12
1.1.12 Restricted properties	12
1.2 Server methods.....	13
1.3 Request options	13
1.3.1 \$filter	13
1.3.2 \$expand.....	16
1.3.3 \$select	17
1.3.4 \$orderby.....	17
1.3.5 \$top.....	17
1.3.6 \$skip	18
1.3.7 \$count.....	18
1.3.8 \$format.....	19
1.4 Operations.....	20
1.4.1 Create	21
1.4.2 Update	23
1.4.3 Upsert	24
1.4.4 Delete	24
1.4.5 File operations.....	25
2 Aras Innovator specific extensions	26
2.1 Paging	26
2.2 Multilingual strings.....	27
2.3 Unsupported Features	27
3 Response translation	28
3.1 Error response	28
4 Vault OData interface	29
4.1 Begin file upload transaction	29

4.2	Upload file chunk.....	29
4.3	Commit file upload transaction.....	30
5	AML Enhancements in OData	32
5.1	Retrieving extended property values and attributes.....	32
5.2	Filtering by Extended Property Definition.....	33
5.3	Filtering Items Extended condition <i>in</i>	35
5.4	Filtering items by an Extended Class or its Descendants.....	36
6	Using OAuth 2.0 Tokens from the Authentication Server.....	38
6.1	Getting the Access Token.....	38
6.2	Confirming Token Authentication for Aras Innovator	41

Introduction

Aras Innovator's RESTful API uses the Open Data Protocol (OData) to create and consume a RESTful API. OData is an ISO/IEC approved OASIS standard that defines best practices for building and consuming RESTful APIs.

JSON enables the Aras Innovator client to communicate with the Aras Innovator server more efficiently through the use of a more compact data format and faster processing. It also allows for communication with other systems that support the OData protocol.

1 Aras Innovator OData Interface

The OData interface receives OData requests and translates them into AML requests. Each corresponding AML request is translated back into an OData response before being returned to the requestor.

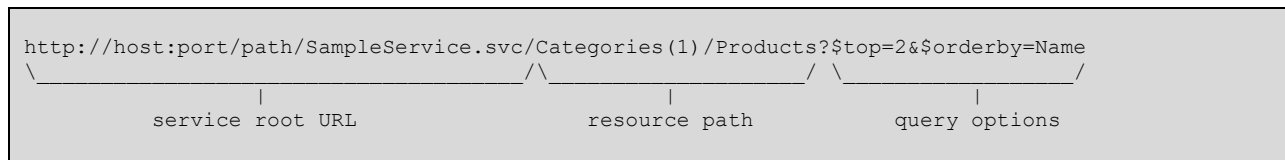
1.1 Request Translation

This section describes the following:

- OData Request URL format
- Entities and properties
- Item properties

1.1.1 OData Request URL Format

The OData Request URL points to the requested entity/entity collection. The URL path starts from the entity set of an action/function call. The following example shows the components of the OData URL:



1.1.2 Entities and Properties

Entity sets contain predefined collections of entities. In Aras Innovator each item type name represents an entity set. For example, the following request returns all items of type Part:

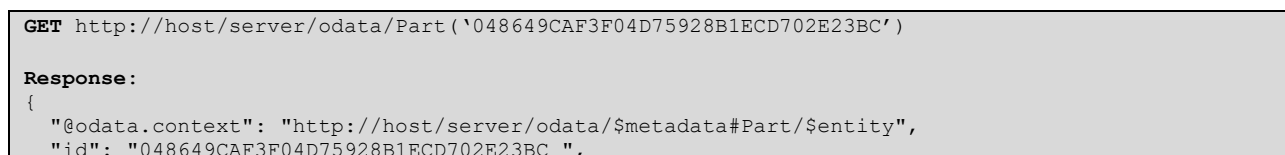
GET: http://host/server/odata/Part

Response:

```

{
  "@odata.context": "http://host/server/odata/$metadata#Part",
  "value": [
    { "id": "...", "item_number": "P-01", ... remainig Part item properties },
    ... remaining Part items
  ]
}
  
```

The client can request a specific item of type Part by passing the item ID in parentheses, as shown in the following example:



```
"item_number": "P-01",
... other Part item properties
}
```

The client can use the *\$filter* attribute in the query for Part items to return a result subset:

```
GET http://host/server/odata/Part?$filter=item_number eq 'P-01'

Response:
{
  "@odata.context": "http://host/server/odata/$metadata#Part",
  "value": [
    { "id": "...", "item_number": "P-01", ... remaining Part item properties }
  ]
}
```

For requests that return a collection, the client can append *\$count* to the request to get the number of elements contained in the collection. Use the AML attributes *pagesize="1"* and *page="1"* to limit the number of items returned in the response. The number of resulting items is stored in the response's *itemmax* attribute:

```
GET http://host/server/odata/Part/$count

Response (value of itemmax attribute of the returned <Item> unless "no items" Fault is returned):
12
```

Append the property name to the resource path to request a property such as *item_number*, as shown here:

```
GET http://host/server/odata/Part('048649CAF3F04D75928B1ECD702E23BC')/item_number

Response:
{
  "@odata.context": "http://host/server/odata/$metadata#Part/$entity/item_number",
  "value": "P-01"
}
```

The OData service only returns the *item_number* property in the response. The request returns the property in the appropriate format (e.g. JSON). The client can add the *\$value* attribute to the request to get the raw property value:

```
GET http://host/server/odata/Part('048649CAF3F04D75928B1ECD702E23BC')/item_number/$value

Response:
P-01
```

This request returns the raw value of the *item_number* property.

1.1.3 Item properties

An Item property contains the ID of the referenced item. When the client requests an item with an Item property, the property does not appear in the response unless its name is specified in either a *\$select* or *\$expand* list.

If the Item property is specified in a *\$select* list, the OData service returns the property value referencing the corresponding item. All additional attributes of the corresponding AML element appear in the response with the annotation *@aras.**.

```
GET http://host/server/odata/Part('048649CAF3F04D75928B1ECD702E23BC')?$select=created_by_id
```

Response:

```
{
  "@odata.context": "http://host/server/odata/$metadata#Part(created_by_id)/$entity",
  "id": "048649CAF3F04D75928B1ECD702E23BC",

  "created_by_id@odata.navigationLink": "User('0CB01F8A2A93430B9A713F7196A85B20')",
  "created_by_id@aras.keyed_name": "Innovator Admin",
}
```

If the client specifies an Item property in an *\$expand* list, the OData service returns the value of the property as a full item:

```
GET http://host/server/odata/Part('048649CAF3F04D75928B1ECD702E23BC')?$expand=created_by_id
```

Response:

```
{
  "@odata.context": "http://host/server/odata/$metadata#Part/$entity",
  "id": "048649CAF3F04D75928B1ECD702E23BC",

  "created_by_id":
  {
    "id": "0CB01F8A2A93430B9A713F7196A85B20",
    ... other User item properties
  },
  ... other Part item properties
}
```

The *\$expand* and *\$select* options can contain more than one property and can be combined in one request, as shown here:


```

GET
http://host/server/odata/Part('048649CAF3F04D75928B1ECD702E23BC')?$expand=created_by_id,Part_CAD
&$select=item_number,owned_by_id

Response:
{
  "@odata.context": "http://host/server/odata/$metadata#Part(item_number,
owned_by_id,created_by_id,Part_CAD)/$entity",
  "id": "048649CAF3F04D75928B1ECD702E23BC",
  "item_number": "P-01",

  "created_by_id":
  {
    "id": "0CB01F8A2A93430B9A713F7196A85B20",
    ... other User item properties
  },

  "owned_by_id@odata.navigationLink": "User('A035BBBCCC8D40BEBED49D71E3129E6F')",
  "owned_by_id@aras.keyed_name": "Innovator User",

  "Part_CAD": [
    { "id": "...", ... remainig Part_CAD item properties },
    ... remaining Part_CAD items
  ]
}

```

If the client requests the Item property itself, the OData Interface returns all of the properties associated with the referenced item:

```

GET http://host/server/odata/Part('048649CAF3F04D75928B1ECD702E23BC')/created_by_id

Response:
{
  "@odata.context": "http://host/server/odata/$metadata#User/$entity",
  "id": "0CB01F8A2A93430B9A713F7196A85B20",
  ... other User item properties
}

```

1.1.4 Relationships

The client appends the relationship name to the resource path in the request:

```

GET http://host/server/odata/Part('048649CAF3F04D75928B1ECD702E23BC')/Part CAD

Response:
{
  "@odata.context": "http://host/server/odata/$metadata#Part CAD",
  "value": [
    { "id": "...", ... remainig Part CAD item properties },
    ... remaining Part CAD items
  ]
}

```

This request returns the collection of Part CAD relationship items.

Note: Aras Innovator doesn't allow properties to start with a capital letter. For this reason, the translator identifies property names starting with capital letters as relationship types. Property names that begin with a lower-case letter are identified as regular properties.

In order to request a related item, the client should request the `related_id` property for a specific relationship item:

```
GET http://host/server/odata/Part('0486...23BC')/Part CAD('01AA...FF3D')/related_id

Response:
{
  "@odata.context": "http://host/server/odata/server/$metadata#CAD/$entity",
  "id": "B9D77528B11B4516BB244F75BE2E606F",
  "item_number": "CAD-01",
  ... other CAD item properties
}
```

The OData service only returns the `related_id` property.

1.1.5 File Properties

A Request for a file property returns the File item along with the corresponding `odata.media*` annotations describing the file stream:

```
GET http://host/server/odata/Document('A035...9E6F')/Document_File('536E...3114')/related_id

Response:
{
  "@odata.context": "http://host/server/odata/server/$metadata#File/$entity",
  "@odata.mediaContentType": "application/pdf",
  "@odata.mediaReadLink": "File('4792...89B0')/$value",
  "id": "4792...89B0",
  "filename": "PartDescription.pdf",
  ... other File item properties
}
```

The OData service only returns the received File item.

In order to get the file content, the client adds the `$value` attribute to the resource path. In this case the OData service constructs a file URL and redirects the client to the Vault server. The Vault server is selected based on the Vault priority assigned to the current user. For example, if the user accessing the Vault server is located in China and the server is also located in China, then the server is assigned a priority of 1.

```
GET http://host/server/odata/Document('A035...9E6F')/Document_File('536E...3114')/related_id/$value
```

1.1.6 PolyItems

When the client requests either an Item property or a collection with the PolyItem type, the OData service adds an `@odata.type` annotation in the response that describes the item type for each item returned by the service:

```
GET http://host/server/odata/Affected_Item('0486...23BC')/affected_id

Response:
{
  "@odata.context": "http://host/server/odata/$metadata#Affected_Item/$entity/affected_id",
  "@odata.type": "#Part",
  "id": "47921523824141E084EA4EE4C06A89B0",
  ... other Part item properties
}
```

1.1.7 Extended Properties

Aras Innovator supports Extended Properties (xProperties). You need to assign an xProperty to a specific item type before you can use it. Once you assign the xproperty, it can be used by items of the same ItemType. You can also create xClasses. xProperties assigned to Parent xClasses are inherited by the child xClasses.

Users with the correct permissions can create and maintain xProperty Definitions and Classification Trees either programmatically or through the UI. End users can assign multiple xProperties to one item. They can also:

- Classify items
- Use xProperty values to search for items.

The OData Interface enables you to perform the following operations:

- Get
- Define
- Update
- Set permissions

1.1.8 Get

You can use the Get operation to retrieve x-properties by using the `$select` option as shown in the following example:

```
GET http://host/server/odata/Part('AF1A....8A88')?$select=xp-prop1,xp-prop2
```

1.1.9 Define operations

Before you can use an explicitly defined xProperty on an item, you must define it. Add an "explicit" value to the "set" attribute and add an "explicit" attribute with the value "1".

To set an xProperty to an undefined state, add the "explicit" value to the "set" attribute and add an "explicit" attribute with the value "0".

1.1.10 Update operation

If an xProperty with a value is included in the JSON body, the "value" is automatically added to the set attribute.

```
PATCH http://host/server/odata/Part('048649CAF3F04D75928B1ECD702E23BC')
{
  "xp-prop": "xp-prop-value"
}
```

Response:

```
HTTP/1.1 200 OK
{
  "@odata.context": "http://host/server/odata/$metadata#Part/$entity",
  "id": "...",
  "xp-prop": "xp-prop-value",
  ... remaining Part item properties
}
```

1.1.11 Change permission operation

Users can change the permission xProperty as shown in the following example.

```
PATCH http://host/server/odata/Part('048649CAF3F04D75928B1ECD702E23BC')
{
  "xp-prop@aras.permission_id": "..."
}
```

Response:

```
HTTP/1.1 200 OK
{
  "@odata.context": "http://host/server/odata/$metadata#Part/$entity",
  "id": "..."
  ... remaining Part item properties
}
```

1.1.12 Restricted properties

Item properties without "Get" permissions are returned with a null value and the "is_null" attribute is equal to "0". To provide information about these properties, "@aras.restricted" metadata is returned in the item, but will not be returned in the response. The same rules apply to xProperties.

Get:

```
http://host/server/odata/Part('048649CAF3F04D75928B1ECD702E23BC')?$select=*
```

Response:

```
{
  "@odata.context": "http://host/server/odata/$metadata#Part/$entity",
  "id": "...",
  "xp-prop@aras.restricted": true,
  "team_id@aras.restricted": true,
  ... remaining Part item properties
}
```

1.2 Server methods

Aras Innovator's server methods are exposed as OData actions. OData functions are not used. The Method call has the method namespace prepended to it to avoid conflicts with item types and property names. The client sends a POST request to the method's URI to call it:

```
POST http://host/server/odata/method.CalculateCost
```

The Translator uses the *Method* item type if the request body is empty or doesn't specify the payload type. The client specifies the payload type using the *@odata.type* annotation:

```
POST http://host/server/odata/method.DoSomething
{
  "@odata.type": "http://host/server/odata/$metadata#Person",
  "name": "John Doe",
  "age": 36
}
```

The client can also append a method to a collection or item URL. In this case the item type and ID (if specified) is taken from the URI.

```
POST http://host/server/odata/Part('01AA...FF3D')/method.DoSomething
{
  "item_number": "P-001",
  "description": "Very important part"
}
```

1.3 Request options

The following OData query options are used to apply additional conditions on a returned data set.

1.3.1 \$filter

The *\$filter* option applies conditions to the returned data set. The *\$filter* expression is translated to AML using the following AML features:

- The *condition* attribute of property element
- The *<or>* and *<and>* elements
- The *where* attribute of an *Item* element.

Table 2 describes how each element of the *\$filter* option is translated to AML.

Table 1: Comparison Operators

Comparison Operators			
Operator	Description	Example	AML
eq	Equal	name eq "Part Name"	condition="eq" Note: OData equal is case sensitive.
ne	Not equal	name ne "Part Name"	condition="ne" Note: OData not equal is case sensitive.
gt	Greater than	price gt 20	condition="gt"
ge	Greater than or equal	price ge 10	condition="ge"
lt	Less than	price lt 20	condition="lt"
le	Less than or equal	price le 20	condition="le"

Table 2: Logical Operators

Logical Operators			
Operator	Description	Example	AML element
and	Logical and	price le 200 and price gt 3.5	<and>
or	Logical or	price le 3.5 or price gt 200	<or>
not	Logical negation	not endswith(description, 'milk')	<not>

The `$filter` expression can contain a call to a built-in function. Table 4 shows how to translate a built-in function to AML.

Table 3: String Functions

String Functions		
Function	Example	AML
contains	<code>contains(company_name, 'freds')</code>	<code><company_name condition="like"> %freds% </company_name></code>
endswith	<code>endswith(company_name, 'freds')</code>	<code><company_name condition="like"> %freds </company_name></code>
startswith	<code>startswith(company_name, 'Alfr')</code>	<code><company_name condition="like"> Alfr% </company_name></code>

1.3.1.1 Null values

OData uses the special value null to indicate that a property doesn't have value. Equality to null is translated using the condition "is null."

GET `http://host/server/odata/Part?$filter=name eq null`

Inequality is translated using the condition "is not null":

GET `http://host/server/odata/Part?$filter=name ne null`

You can use the previous condition with negation:

GET `http://host/server/odata/Part?$filter=not name eq null`

1.3.1.2 Filtering by Related Item Properties

The OData interface supports filtering using a property associated with the related item:

GET `http://host/server/odata/Part?$filter=created_by_id/name eq 'Admin'`

In Aras Innovator, item type names may contain spaces. This conflicts with the filter expression syntax where a space character is used as a delimiter. In order to resolve the conflict, names containing a space character should be enclosed by square brackets in filter expressions, as shown:

GET `http://host/server/odata/Part?$filter=[Part BOM]/quantity eq 2`

1.3.2 \$expand

The \$expand option specifies which Item property or relationship must be included in a response. If you do not specify the \$expand option, Item properties are included in the response as an Item with a single id property. Relationships are not included at all.

Expanding Item property:

```
GET http://host/server/odata/Part?$expand=created_by_id
```

Expanding relationship:

```
GET http://host/server/odata/Part('048649CAF3F04D75928B1ECD702E23BC')?$expand=Part CAD
```

Response:

```
{
  "@odata.context": "http://host/server/odata/$metadata#Part",
  "id": "...",

  "Part_CAD":
  [
    { "id": "...", ... remaining Part CAD item propertires },
    { "id": "...", ... remaining Part CAD item propertires },
    ... remainig Part CAD items
  ],
  ... remaining Part item properties
}
```

Expanding property of a related item:

```
GET http://host/server/odata/Part('048649CAF3F04D75928B1ECD702E23BC')?$expand=Part CAD($expand=related_id)
```

Response:

```
{
  "@odata.context": "http://host/server/odata/$metadata#Part",
  "id": "048649CAF3F04D75928B1ECD702E23BC",

  "Part_CAD":
  [
    {
      "related_id":
      {
        "item_number": "CAD-1",
        ... remaining CAD item properties
      },
      ... remainig Part CAD item properties
    },
    ... remainig Part CAD items
  ],
  ... remaining Part item properties
}
```

Properties specified using the \$expand option are always implicitly included in the \$select list.

1.3.3 \$select

The \$select option defines which properties should be included in a result set. The ID property must always be included:

```
GET http://host/server/odata/Part?$select=item_number

Response:
{
  "@odata.context": "http://host/server/odata/$metadata#Part(item_number)",
  "value":
  [
    {"id": "...", "item_number": "P-01"},
    {"id": "...", "item_number": "P-02"},
    ... remaining Part items
  ]
}
```

You can also apply the \$select option to related items:

```
GET http://host/server/odata/Part?$expand=Part CAD($select=related_id)

Response:
{
  "@odata.context": "http://host/server/odata/$metadata#Part",
  "value":
  [
    {
      "id": "...",
      "item_number": "P-01",
      "Part_CAD":
      [
        {
          "id": "...",
          "related_id@odata.navigationLink": "CAD('CCF205347C814DD1AF056875E0A880AC')",
          "related_id@aras.keyed_name": "CAD-01",
        },
        ... remaining Part CAD items
      ],
      ... remaining Part item properties
    },
    ... remaining Part items
  ]
}
```

If you omit the \$select option, the OData service will not return properties that have a *null* value. In order to return all properties the client must use the star "*" value for the \$select query option in the request.

1.3.4 \$orderby

The \$orderby option specifies the order of items in a result set. The Translator converts the sorting expression to AML syntax and sets the *orderBy* attribute of the *Item* element:

```
GET http://host/server/odata/Part?$orderby=item_number desc
```

1.3.5 \$top

The \$top option limits the result set to the first \$top items. In the following example, the query specifies that Part items 01-10 be returned. The Translator adds the *fetch* attribute to the AML request and specifies the *GetItemWithPropertyConditions* method as the action:

```
GET http://host/server/odata/Part?$top=10

Response:
{
  "@odata.context": "serviceRoot/$metadata#Part",
  "value":
  [
    {"id": "...", "item_number": "P-01",...},
    {"id": "...", "item_number": "P-02",...},
    ... remaining 8 Part items
  ]
}
```

1.3.6 \$skip

The [\\$skip](#) option allows a specified number of items in the resulting collection to be skipped. The following example specifies that parts 01-10 be excluded from the collection.

The Translator adds the *offset* attribute to the AML request and specifies the *GetItemWithPropertyConditions* method as the action:

```
GET http://host/server/odata/Part?$skip=10

Response:
{
  "@odata.context": "serviceRoot/$metadata#Part",
  "value":
  [
    ... Parts starting from 11th item remaining 8 Part items
    {"id": "...", "item_number": "P-01",...},
    {"id": "...", "item_number": "P-02",...},
  ]
}
```

1.3.7 \$count

If the [\\$count](#) option is present and set to true, the translator adds the `@odata.count` property to the resulting data set with a value equal to the number of items to be included in the response:

```
GET http://host/server/odata/Part?$count=true

Response:
{
  "@odata.context": "serviceRoot/$metadata#Part",
  "@odata.count": 10
  "value":
  [
    {"id": "...", "item number": "P-01",...},
    {"id": "...", "item_number": "P-02",...},
    ...
  ]
}
```

If the result does not return a data set [\\$count](#) is ignored.

1.3.8 \$format

The \$format option only supports the JSON format for communication.

1.4 Operations

In OData, operations are specified as HTTP methods. Any Aras Innovator action that cannot be specified as an HTTP method is passed using the custom `@aras.action` annotation in the request body. Actions are prepended with the action namespace to avoid conflicts with item types and property names. The actions use the POST HTTP method.

Table 4: HTTP Methods

HTTP method	Aras Innovator action	Notes
GET	get	
POST	add	Can be sent only to collection.
PATCH	edit	Can be sent only to specific entity
PUT	edit	Update simple property or single navigation property
DELETE	delete	Can be sent only to specific entity
PATCH	merge	Uses the <code>@aras.action=merge</code> annotation
POST	create	Uses the <code>@aras.action=create</code> annotation
PATCH	update	Uses the <code>@aras.action=update</code> annotation
PATCH	lock	Uses the <code>@aras.action=lock</code> annotation
PATCH	unlock	Uses the <code>@aras.action=unlock</code> annotation
DELETE	purge	Uses the <code>@aras.action=purge</code> annotation

The following example shows how the action can be used:

```

PATCH http://host/server/odata/Part('0486...23BC')
Prefer: return=minimal
{
  "@aras.action": "edit",
  "item_number": "P-025"
}

Response:
HTTP/1.1 204 No Content
Location: http://host/server/odata/Part('0486...23BC')

```

If the action is successful, all data modification operations except DELETE return a modified item representation if the client specified *return=representation* in *Prefer* header. In a case where the client specified *return=minimal*, the OData service returns the *204 No Content* status code.

The OData specification doesn't define service behavior in case the *return* preference is omitted. In order to be consistent with Aras Innovator, the OData service uses *return=representation* as the default.

The response's *Location* header must point to either a created or modified item.

1.4.1 Create

The client must send a POST request to the collection URL to create an item. The client can also call a create or merge action. The Request body must contain a single item containing all the required properties. If the HTTP Prefer header is set to *return=minimal* in the request, the OData service returns a 204 No Content status code. Otherwise the service returns a 201 Created status code along with the item's properties:

```
POST http://host/server/odata/Part
{
  "item_number": "P-01"
}

Response:
HTTP/1.1 201 Created
Location: http://host/server/odata/Part('0486...23BC')
{
  "@odata.context": "http://host/server/odata/$metadata#Part/$entity",
  "id": "0486...23BC",
  "item_number": "P-01",
  ... other Part item properties
}
```

The client can create an item referenced by an item property in a single request (in OData this is referred to as a [deep insert](#)):

```
POST http://host/server/odata/Action
Prefer: return=minimal
{
  "name": "New Action",
  "type": "Item",
  "location": "Client",
  "target": "Main",
  "method":
    {
      "name": "NewMethod",
      "method_type": "C#",
      "method_text": "test"
    }
}

Response:
HTTP/1.1 204 No Content
Location: http://host/server/odata/Action('01AA...FF3D')
```

The client can also create relationships between items:

```

POST http://host/server/odata/Part
Prefer: return=minimal
{
  "item_number": "P-01",
  "Part_CAD": [{
    "related_id":
      {
        "item_number": "CAD-01"
      }
  }]
}

Response:
HTTP/1.1 204 No Content
Location: http://host/server/odata/Part('0486...23BC')

```

The client can use [@odata.bind annotation](#) to add a reference to an existing item:

```

POST http://host/server/odata/Part
Prefer: return=minimal
{
  "item_number": "P-01",
  "Part_CAD": [{
    "related_id@odata.bind": "CAD('01AA...FF3D')"
  }]
}

Response:
HTTP/1.1 204 No Content
Location: http://host/server/odata/Part('0486...23BC')

```

The client can send a POST request to a relationship property to add a new relationship item to an existing item:

```

POST http://host/server/odata/Part('C542F...EF6B')/Part_CAD
Prefer: return=minimal
{
  "related_id@odata.bind": "CAD('01AA...FF3D')"
}

Response:
HTTP/1.1 204 No Content
Location: http://host/server/odata/Part('C542F...EF6B')

```

If the Part item type is versionable, the POST request creates a new item version.

The client can also create a new relationship along with a new related item in “Innovator style”. In this case a new item version is not created:

```

POST http://host/server/odata/Part_CAD
Prefer: return=minimal
{
  "source_id@odata.bind": "Part('01AA...FF3D')"
  "related_id":
    {
      "item_number": "CAD-01"
    }
}

Response:
HTTP/1.1 204 No Content
Location: http://host/server/odata/Part_CAD('C542F...EF6B')

```

1.4.2 Update

In order to [update an existing item](#) the client sends a PATCH request to the specific item URL. The request body must contain a single item with updatable properties:

```
PATCH http://host/server/odata/Part('01AA...FF3D')
{
  "item_number": "P-100",
}

Response:
HTTP/1.1 200 OK
Location: http://host/server/odata/Part('01AA...FF3D')
{
  "@odata.context": "http://host/server/odata/$metadata#Part/$entity",
  "id": "01AA...FF3D",
  "item_number": "P-01",
  ... other Part item properties
}
```

The client can also include *edit* or *update* actions in the PATCH request for data modification:

```
POST http://host/server/odata/Part('01AA112937924D5383FB4A101B2AFF3D')
Prefer: return=minimal
{
  "@aras.action": "update",
  "item_number": "P-100"
}

Response:
HTTP/1.1 204 No Content
Location: http://host/server/odata/Part('01AA...FF3D')
```

In order to specify a value for a simple property, the client must send a PUT request to the corresponding property URL:

```
PUT http://host/server/odata/Part('01AA112937924D5383FB4A101B2AFF3D')/item_number
Prefer: return=minimal
{
  "value": "P-001"
}

Response:
HTTP/1.1 204 No Content
Location: http://host/server/odata/Part('01AA...FF3D')/item_number
```

The client can also send a PUT request to the \$value endpoint of a simple property where the body of the request contains a raw property value:

```
PUT http://host/server/odata/Part('01AA112937924D5383FB4A101B2AFF3D')/item_number/$value
Prefer: return=minimal

P-001

Response:
HTTP/1.1 204 No Content
Location: http://host/server/odata/Part('01AA...FF3D')/item_number/$value
```

In order to set a value for an Item property, the client must send a PUT request to the Item property reference:

```
PUT http://host/server/odata/Part('01AA112937924D5383FB4A101B2AFF3D')/owned_by_id/$ref
Prefer: return=minimal
{
  "@odata.id": "http://host/server/odata/User('C542FC153AE647A59CD6F6967295EF6B') "
}

Response:
HTTP/1.1 204 No Content
Location: http://host/server/odata/Part('01AA...FF3D')
```

Update actions can only modify one item at time.

1.4.3 Upsert

In order to perform an [upsert](#) operation, the client should send a PATCH request to a specific entry and include an HTTP header *If-Match* with the value "*" in the request. It will be translated to AML using the *merge* action:

```
PATCH http://host/server/odata/Part('01AA...FF3D')
If-Match: *
{
  "item_number": "P-100",
}

Response:
HTTP/1.1 200 OK
Location: http://host/server/odata/Part('01AA...FF3D')
{
  "@odata.context": "http://host/server/odata/$metadata#Part/$entity",
  "id": "01AA...FF3D",
  "item_number": "P-01",
  ... other Part item properties
}
```

1.4.4 Delete

In order to [delete an existing item](#) the client must send a DELETE request to the specific item URL:

```
DELETE http://host/server/odata/Part('01AA112937924D5383FB4A101B2AFF3D')

Response:
HTTP/1.1 204 No Content
```

The client uses the `@aras.action=purge` annotation within the body of the query to delete only one version of the item:

```
DELETE http://host/server/odata/Part('01AA112937924D5383FB4A101B2AFF3D')
Prefer: return=minimal
{
  "@aras.action": "purge"
}

Response:
HTTP/1.1 204 No Content
```


To remove a relationship, the client sends a DELETE request to the *\$ref* URL of the corresponding relationship property, passing the *\$id* query string option:

```
DELETE http://host/server/odata/Part('01AA...FF3D')/Part_CAD/$ref?$id=Part_CAD('B9D7...606F')
Response:
HTTP/1.1 204 No Content
```

To clear an item property, the client sends a DELETE request to the *\$ref* URL for the corresponding property. The Purge action cannot be used for this request:

```
DELETE http://host/server/odata/Part('01AA112937924D5383FB4A101B2AFF3D')/owned_by_id/$ref
Response:
HTTP/1.1 204 No Content
```

A Delete action can only remove one item at time.

Note: Because Aras Innovator always sends a success response for a delete operation even if the item being deleted cannot be found, the OData service also always sends a success response.

1.4.5 File operations

The Aras Innovator OData interface only supports get and delete operations for File items. You must use the [Vault OData interface](#) to perform a Create operation. The Update operation is not supported.

2 Aras Innovator specific extensions

This section describes the extensions that can be used in Aras Innovator to:

- Configure server-driven paging
- Retrieve the language associated with multi-lingual string types

2.1 Paging

Use the offset and fetch AML attributes to configure Server-driven paging. When the client specifies the `odata.maxpagesize` value in the `Prefer` request header, the OData Interface translates the value using a combination of the offset and fetch attributes while taking into account any `$skip` and `$top` options specified in the request. The `@odata.nextLink` annotation value includes a `$skiptoken` option which specifies the starting position of the next page within a requested data set. All query options from the original request appear in the `@odata.nextLink` URL.

The last page cannot contain the `@odata.nextLink` annotation. In order to determine whether `@odata.nextLink` must be present in the response, the translator produces AML that requests one additional item. If all requested items are received, `@odata.nextLink` must be included in the response. If the number of returned items is less than requested, `@odata.nextLink` must be omitted.

```
GET http://host/server/odata/Part?$skip=3&$top=30
Prefer: odata.maxpagesize=10

Response:
{
  "@odata.context": "http://host/server/odata/$metadata#Part",
  "@odata.nextLink": "http://host/server/odata/Part?$skip=3&$top=30&$skiptoken=10",
  "value": [
    { "id": "...", "item_number": "P-01", ... remaining Part item properties },
    ... 9 remaining Part items
  ]
}
```

The request for the next page is translated as follows:

```
GET http://host/server/odata/Part?$skip=3&$top=30&$skiptoken=10
Prefer: odata.maxpagesize=10

Response:
{
  "@odata.context": "http://host/server/odata/$metadata#Part",
  "@odata.nextLink": "http://host/server/odata/Part?$skip=3&$top=30&$skiptoken=20",
  "value": [
    { "id": "...", "item_number": "P-10", ... remaining Part item properties },
    ... 9 remaining Part items
  ]
}
```

2.2 Multilingual strings

The properties of the multilingual string type are always returned using the *lang* attribute to specify the language used by the string. The *lang* attribute is translated to *@aras.lang* annotation for this property:

```
GET http://host/server/odata/ItemType('4F1A...8A88')?$select=label

Response:
{
  "@odata.context": "http://host/server/odata/$metadata#ItemType(label)/$entity",
  "id": "4F1A...8A88",
  "label": "Part",
  "label@aras.lang": "en"
}
```

The client can also ask for the property value specifying a language other than the language being used in the session or in all the languages defined in the database. In AML the *language* attribute serves this purpose. It may contain a comma-separated list of languages or star "*" specifying all the languages in the database. The OData service introduces the *lang* query option with the same meaning and *#aras.lang.** annotation qualifier that is used to return the property or property attribute value in the requested languages:

```
GET http://host/server/odata/ItemType('4F1A...8A88')?$select=label&lang=*

Response:
{
  "@odata.context": "http://host/server/odata/$metadata#ItemType(label)/$entity",
  "id": "4F1A...8A88",
  "label": "Part",
  "label@aras.lang": "en",
  "label@aras.custom_attr": "PPP",
  "label#@aras.lang.ru": "Деталь"
  "label@aras.custom_attr#@aras.lang.ru": "ДДД"
}
```

2.3 Unsupported Features

Currently, the OData interface does not support the following terms. It will respond with a "501 Not Implemented" HTTP code if you use one of them:

- *\$all* resource path component
- *\$crossjoin* resource path component
- [POST request to reference \(\\$ref\) of collection navigation property](#)
- [Asynchronous requests](#)
- Using number of related items as sorting criteria:
http://host/server/odata/Part?\$orderby=Part_CAD/\$count
- Asynchronous operations
- Delta responses

3 Response translation

This section describes the response types supported by the OData server and how they translate to AML.

3.1 Error response

An [Error response](#) reports OData service-specific errors and [unsupported functionality](#) as well as AML error responses:

```
Response:  
HTTP/1.1 400 Bad Request  
{  
  "code": "400",  
  "message": "Bad Request",  
  "target": "AML",  
  "details":  
  {  
    "code": "SOAP-ENV:Server",  
    "message": "addRelshipPermInfo is not specified or its sourceID is empty",  
    "target": "System.ArgumentException"  
  }  
}
```

4 Vault OData interface

The content Vault servers store files. When there is more than one Vault server, each server may be in a different location. All Vault servers have an assigned priority based on the user's location. For example, if a user located in China tries to access a Vault Server also located in China, then that Vault server has priority 1 because of its proximity to the user.

The content Vault server has its own OData interface for uploading file content. This interface exposes the three methods described in the following sections.

4.1 Begin file upload transaction

The *.BeginTransaction* Method starts the file upload transaction and returns a transaction ID:

```
POST http://vault/odata/vault.BeginTransaction
VAULTID: {{vault_id}}

Response:
HTTP/1.1 200 OK
Content-Type: application/json
{
  "transactionId": "8970e933322d1328537f645d3683214c"
}
```

Note: You will need to add VAULTID header to the vault.BeginTransaction request. To receive {{vault_id}} value you need to send the following request:

```
GET http://host/server/odata/User('{{current_user_id}}')?$select=default_vault

Response:
{
  "@odata.context": "http://host/server/odata/$metadata#User(default_vault)/$entity",
  "@odata.id": "User('{{current_user_id}})",
  "default_vault@odata.associationLink": "User('{{current_user_id}})/default_vault/$ref",
  "default_vault@odata.navigationLink": "User('{{current_user_id}})/default_vault",
  "default_vault@aras.keyed_name": "Default",
  "default_vault@aras.id": "{{vault_id}}"
}
```

4.2 Upload file chunk

The *vault.UploadFile* Method uploads the file chunk within a given transaction. The File ID is GUID generated on the client and passed to the vault server using the *fileId* query option. The upload must use Transfer-encoding: chunked and containing a Content-range header where the range is the range of bytes within the chunk of the file being transferred and the size is the size of the file. A Content-disposition header must be included containing the name of the file being uploaded.

```

POST http://vault/odata/vault.UploadFile?fileId=2778F0A17FDF4095A497A9A27B594376
VAULTID: {{vault_id}}
Content-Disposition:attachment; filename*=utf-8''file_0.txt
Content-Length:53
Content-Range:bytes 0-52/53
Content-Type:application/octet-stream
Aras-Content-Range-Checksum:2535782373
Aras-Content-Range-Checksum-Type:xxHashAsUInt32AsDecimalString
transactionid:{{transaction_id}}

... file chunk content

Response:
HTTP/1.1 200 OK

```

Note: You will need to add a VAULTID header to the vault.UploadFile request. To receive {{vault_id}} value you need to send the request as shown in the following example.

```

GET http://host/server/odata/User('{{current_user_id}}')?$select=default_vault

Response:
{
  "@odata.context": "http://host/server/odata/$metadata#User(default_vault)/$entity",
  "@odata.id": "User('{{current_user_id}})",
  "default_vault@odata.associationLink": "User('{{current_user_id}}')/default_vault/$ref",
  "default_vault@odata.navigationLink": "User('{{current_user_id}}')/default_vault",
  "default_vault@aras.keyed_name": "Default",
  "default_vault@aras.id": "{{vault_id}}"
}

```

4.3 Commit file upload transaction

The *vault.CommitTransaction* method finalizes a file upload and submits the OData representation of the item using file(s) uploaded by the *vault.UploadFile* method. At a minimum, this includes the file ID, the filename, file size, and the Located item that points to the current user's vault. For example:

```

{
  'id': [id passed to the vault.UploadFile method],
  'filename': [the name of the file being uploaded],
  'file_size': [the size of the file in bytes],
  'located':
    [
      {
        'file_version': [the version of the file],
        'related_id': [the ID of the user's vault]
      }
    ]
}

```

Once Aras Innovator responds that the transaction is successful, the Vault server commits the transaction on its side.

```

POST http://vault/odata/vault.CommitTransaction
VAULTID: {{vault_id}}
Prefer: return=minimal
OData-Version: 4.0
transactionid: 8970e933322d1328537f645d3683214c
Content-Type: multipart/mixed; boundary=batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Length: ###

--batch_36522ad7-fc75-4b56-8c71-56071383e77b
Content-Type: application/http

POST /odata/Document HTTP/1.1
Host: host
Content-Type: application/json

{
  "@odata.context": "http://host/server/odata/$metadata#Document/$entity",
  "@odata.type": "#Document",
  "id": "FBCC...74FC",
  "item_number": "DOC-01",
  "Document File": [{
    "id": "40F1...6369",
    "related_id":
    {
      "id": "2778F0A17FDF4095A497A9A27B594376",
      "filename": "6d039e86857376a761b87c4d559953e2.jpg"
    }
  }]
}

--batch_36522ad7-fc75-4b56-8c71-56071383e77b--

Response:
HTTP/1.1 204 No Content
Location: http://host/server/odata/Document('FBCC...74FC')

```

Note: You will need to add VAULTID header to the vault.CommitTransaction request. To receive {{vault_id}} value you need to send the request shown here.

```

GET http://host/server/odata/User('{{current_user_id}}')?$select=default_vault

Response:
{
  "@odata.context": "http://host/server/odata/$metadata#User(default_vault)/$entity",
  "@odata.id": "User('{{current_user_id}})",
  "default_vault@odata.associationLink": "User('{{current_user_id}})/default_vault/$ref",
  "default_vault@odata.navigationLink": "User('{{current_user_id}})/default_vault",
  "default_vault@aras.keyed_name": "Default",
  "default_vault@aras.id": "{{vault_id}}"
}

```

5 AML Enhancements in OData

This section describes enhancements to AML in the OData interface that enable you to retrieve information about extended properties and classes. You can also filter items by extended class or extended property definition. These enhancements have been introduced as part of the Extended Classification feature.

5.1 Retrieving extended property values and attributes

In order to request the value of a specific extended property you need to include the xproperty name in a `$select` request:

```
GET http://host/server/odata/Part('048...3BC')?$select=xp-cost

AML:
<Item type="Part" action="get" id="048...3BC" select="xp-cost($value)" />

Response:
HTTP/1.1 200 OK

{
  "@odata.context": "http://host/server/odata/$metadata#Part(xp-cost)/$entity",
  "@odata.id": "Part('0486...23BC')",
  "xp-cost": "{defined xp-cost value}",
  "itimetype": "4F1AC04A2B484F3ABA4E20DB63808A88"
}
```

Use OData annotations to request an attribute associated with an extended property as shown in the following example:

```
GET http://host/server/odata/Part('048...3BC')?$select=xp-cost@aras.permission_id

AML:
<Item type="Part" action="get" id="048...3BC" select="xp-cost(@permission_id)" />

Response:
HTTP/1.1 200 OK

{
  "@odata.context": "http://host/server/odata/$metadata#Part(xp-*@aras.permission_id)/$entity",
  "@odata.id": "Part('0486...23BC')",
  "xp-text@aras.permission_id": "{Permission_PropertyValue ID}",
  "itimetype": "4F1AC04A2B484F3ABA4E20DB63808A88"
}
```

If you request both a value and/or more than one attribute associated with the same property, you must combine them into one selection list element in the resulting AML:


```

GET http://host/server/odata/Part('048...3BC')?$select=xp-cost, xp-cost@aras.permission_id

AML:
<Item type="Part" action="get" id="048...3BC" select="xp-cost($value,@permission_id)" />

Response:
HTTP/1.1 200 OK

{
  "@odata.context": "http://host/server/odata/$metadata#Part(xp-*@aras.permission_id)/$entity",
  "@odata.id": "Part('0486...23BC')",
  "xp-cost": "{defined xp-cost value}",
  "xp-text@aras.permission_id": "{Permission_PropertyValue ID}",
  "itemtype": "4F1AC04A2B484F3ABA4E20DB63808A88"
}

```

Use `xp-` as the property name to request all of the extended properties that are defined for an item:

```

GET http://host/server/odata/Part('048...3BC')?$select=xp-*

AML:
<Item type="Part" action="get" id="048...3BC" select="xp-*( $value)" />

Response:
HTTP/1.1 200 OK

{
  "@odata.context": "http://host/server/odata/$metadata#Part(xp-cost)/$entity",
  "@odata.id": "Part('0486...23BC')",
  "xp-cost": "{defined xp-cost value}",
  "itemtype": "4F1AC04A2B484F3ABA4E20DB63808A88"

  ... other Part item extended properties
}

```

5.2 Filtering by Extended Property Definition

Use `is_defined` the filter function to filter request results using the extended property definition and how the property is defined (either explicit or through extended class):

```

GET http://host/server/odata/Part?$filter=is_defined(xp-cost)

AML:
<Item type="Part" action="get">
  <xp-cost condition="is defined"/>
</Item>

Response:
HTTP/1.1 200 OK

{
  "@odata.context": "http://host/server/odata/$metadata#Part",
  "value": [
    {
      "id": "048...3BC ",
      "item_number": "Part_with_Defined_xp_cost",
      "itemtype": "4F1AC04A2B484F3ABA4E20DB63808A88"

      ... other Part item properties
    }
  ]
}

```

```

    }
    ... other Part items with defined xp-cost
  ]
}

```

You can request a property that is defined in a specific way:

```
GET http://host/server/odata/Part?$filter=is_defined(xp-cost,explicit)
```

AML:

```

<Item type="Part" action="get">
  <xp-cost condition="is defined" defined_as="explicit"/>
</Item>

```

Response:

HTTP/1.1 200 OK

```

{
  "@odata.context": "http://host/server/odata/$metadata#Part",
  "value": [
    {
      "id": "048...3BC ",
      "item_number": "Part_with_Defined_xp_cost",
      "itemtype": "4F1AC04A2B484F3ABA4E20DB63808A88"

      ... other Part item properties
    }
    ... other Part items with explicitly defined xp-cost
  ]
}

```

Use the Negate operation to request items that do not have a specific extended property defined:

```
GET http://host/server/odata/Part?$filter=not is_defined(xp-cost,class)
```

AML:

```

<Item type="Part" action="get">
  <xp-cost condition="is not defined" defined_as="class"/>
</Item>

```

Response:

HTTP/1.1 200 OK

```

{
  "@odata.context": "http://host/server/odata/$metadata#Part",
  "value": [
    {
      "id": "048...3BC ",
      "item_number": "Part_with_xp_cost_not_Defined_in_Class",
      "itemtype": "4F1AC04A2B484F3ABA4E20DB63808A88"

      ... other Part item properties
    }
    ... other Part items with xp-cost not defined via xClass
  ]
}

```

5.3 Filtering Items Extended condition *in*

In order to implement the `in` extended condition, the OData Interface has introduced the `in()` filter function. This function enables you to request specific information related to an item. In the following example, the first argument is the property name, the second argument is the collection to look in, and the third argument is the property name of the collection element:

```
GET
http://host/server/odata/xPropertyContainerItem?$select=id,itemtype&$filter=in(itemtype,$root/Itemtype($select=id; $filter=name eq 'Part'),'id')

AML:
<Item type="xPropertyContainerItem" action="get" select="name,itemtype">
  <itemtype condition="in" by="id">
    <Item type="ItemType" select="id">
      <Name condition="eq">Part</Name>
    </Item>
  </itemtype>
</Item>

Response:
HTTP/1.1 200 OK

{
  "@odata.context": "http://host/server/odata/$metadata#xPropertyContainerItem(id,itemtype)",
  "value": [
    {
      "@odata.type": "#Part",
      "id": "048...3BC ",
      "itemtype": "4F1AC04A2B484F3ABA4E20DB63808A88"
    }
    ... other xPropertyContainerItem items with Part data_source
  ]
}
```

You can use the `in()` function with other item types as well:

```
GET http://host/server/odata/Part?$filter=in(created_by_id,$root/User($select=id;$filter=login_name eq 'jsmith'),'id')

AML:
<Item type="Part" action="get">
  <created_by_id condition="in" by="id">
    <Item type="User" select="id">
      <logon>jsmith</logon>
    </Item>
  </created_by_id>
</Item>

Response:
HTTP/1.1 200 OK

{
  "@odata.context": "http://host/server/odata/$metadata#Part",
  "value": [
    {
      "id": "048...3BC ",
      "item_number": "Part-1",
      "itemtype": "4F1AC04A2B484F3ABA4E20DB63808A88"
    }
    ... other Part item properties
  ]
  ... other Part items created by User 'jsmith'
}
```

5.4 Filtering items by an Extended Class or its Descendants

Use the `isof_xclass('<class-name-or-id>')` function to filter an item collection by its assigned `xClass`:

```
GET http://host/server/odata/Part?$filter=isof_xclass('Bolt')
```

AML:

```
<Item type="Part" action="get">
  <Relationships>
    <Item type="Part_xClass" action="get">
      <related_id>
        <Item type="xClass" action="get">
          <name>Bolt</name>
        </Item>
      </related_id>
    </Item>
  </Relationships>
</Item>
```

Response:
HTTP/1.1 200 OK

```
{
  "@odata.context": "http://host/server/odata/$metadata#Part",
  "value": [
    {
      "id": "048...3BC ",
      "item_number": "Bolt",
      "itemtype": "4F1AC04A2B484F3ABA4E20DB63808A88"

      ... other Part item properties
    }
    ... other Part items that classified by class 'Bolt'
  ]
}
```

If you pass the id string as a function argument, the class id is returned instead of the class name:

```
GET http://host/server/odata/Part?$filter=isof_xclass('F48...3BC')
```

AML:

```
<Item type="Part" action="get">
  <Relationships>
    <Item type="Part_xClass" action="get">
      <related_id>
        <Item type="xClass" action="get">
          <id>048...3BC</id>
        </Item>
      </related_id>
    </Item>
  </Relationships>
</Item>
```

Response:
HTTP/1.1 200 OK

```
{
  "@odata.context": "http://host/server/odata/$metadata#Part",
  "value": [
    {
      "id": "048...3BC ",
      "item_number": "Bolt",
      "itemtype": "4F1AC04A2B484F3ABA4E20DB63808A88"
    }
  ]
}
```

```

    ... other Part item properties
  }
  ... other Part items that classified by class (with ID='F48...3BC')
]
}

```

Use the `isof_xclassordescendants('<class-name-or-id>')` function to filter an item collection by its assigned `xClass` or descendants:

GET `http://host/server/odata/Part?$filter=isof_xclassordescendants('Bolt')`

AML:

```

<Item type="Part" action="get">
  <Relationships>
    <Item type="Part_xClass" action="get">
      <related_id>
        <Item type="xClass" action="getXClassAndAllDescendants">
          <name>Bolt</name>
        </Item>
      </related_id>
    </Item>
  </Relationships>
</Item>

```

Response:

HTTP/1.1 200 OK

```

{
  "@odata.context": "http://host/server/odata/$metadata#Part",
  "value": [
    {
      "id": "048...3BC ",
      "item_number": "Bolt",
      "itemtype": "4F1AC04A2B484F3ABA4E20DB63808A88"

      ... other Part item properties
    }
    ... other Part items that classified by class class='Bolt' or it subclasses
  ]
}

```

6 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.

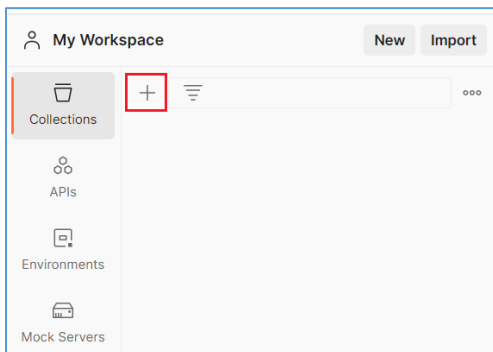
From Aras Innovator 14+ versions, OAuth 2.0 tokens is used for token authentication.

In this document, Postman is used, although the same request can be sent using any API platform.

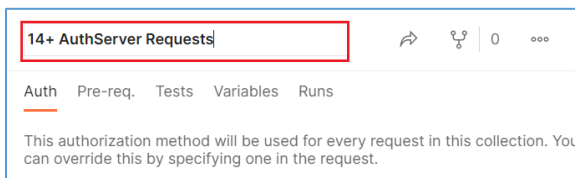
6.1 Getting the Access Token

The following steps outline the process of getting the access token:

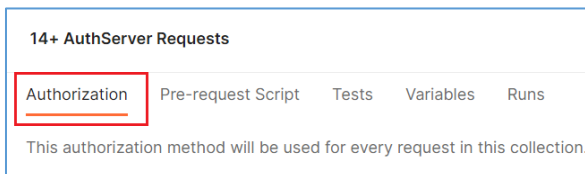
1. Download the [Postman](#) application on the local machine.
2. Under **My Workspace**, select **Collections** and click **Create New Collection**.



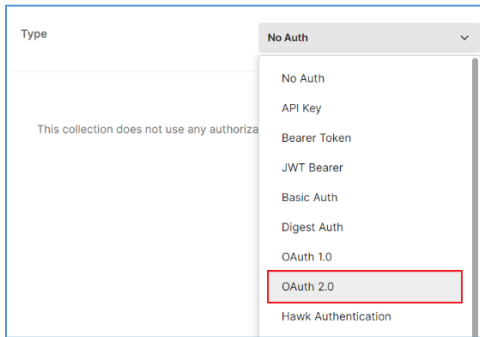
3. Provide a name to the new Workspace. For example, 14+ AuthServer Requests.



4. Select the **Authorization** tab.



- For the **Type** field, select **OAuth 2.0** from the dropdown.



- Scroll down to **Configure New Token** section and set the configuration options as follows:

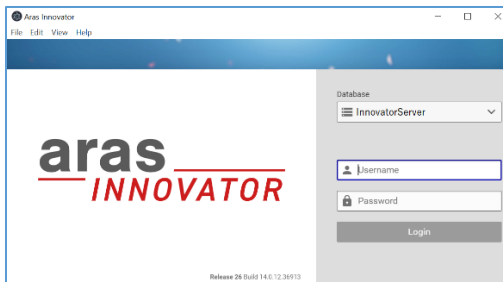
The image shows the 'Configure New Token' configuration form. It has two tabs: 'Configuration Options' (selected) and 'Advanced Options'. The form contains the following fields and values:

- Token Name: aras
- Grant Type: Authorization Code
- Callback URL: http://localhost/14SP12/Client/OAuth...
- Auth URL: http://localhost/14SP12/Client/oauth...
- Access Token URL: http://localhost/14SP12/Client/oauth...
- Client ID: InnovatorClient
- Client Secret: Client Secret
- Scope: openid Innovator offline_access ...
- State: State
- Client Authentication: Send as Basic Auth header

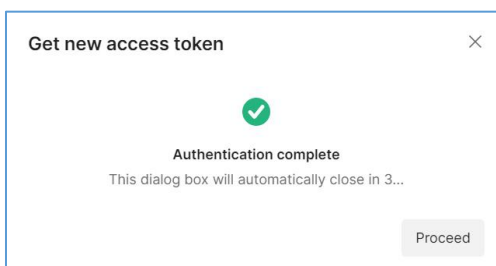
At the bottom of the form, there is a 'Clear cookies' button and a 'Get New Access Token' button.

Field	Value
Token Name	Name of the Token
Grant Type	Authorization Code (With PKCE)
Callback URL	http://<servername>/<web alias>/Client/OAuth/PopupCallback
Auth URL	http://<servername>/<web alias>/oauthserver/connect/authorize
Access Token URL	http://<servername>/<web alias>/oauthserver/connect/token
Client ID	InnovatorClient
Scope	openid Innovator offline_access

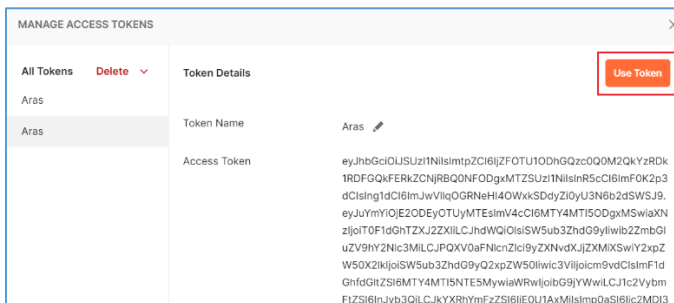
- Click **Get New Access Token** button.
- Log into the popup Aras Innovator window with your credentials.



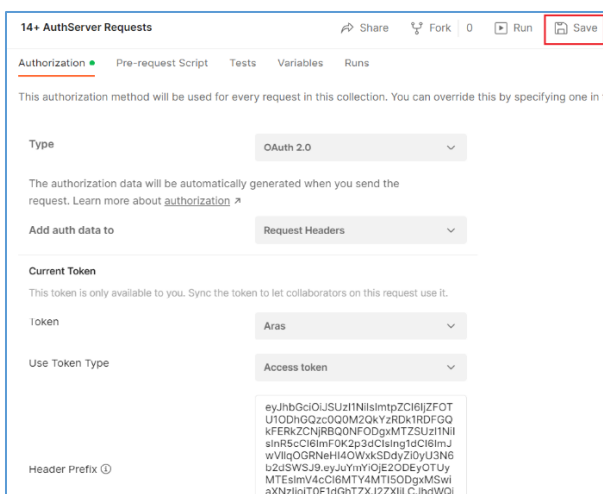
- On the **Get new access token** dialog box, click **Proceed**.



- On **Manage Access Token** dialog box, click **Use Token**.



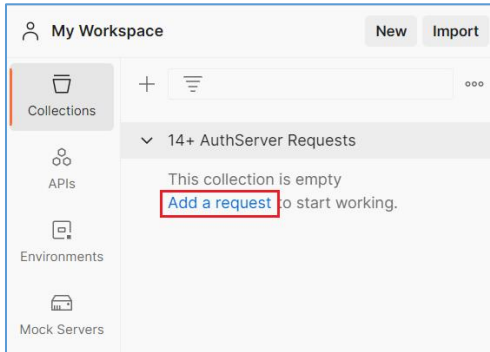
- Click **Save** to save the changes.



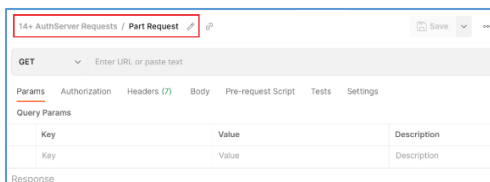
6.2 Confirming Token Authentication for Aras Innovator

The following steps outline the process of confirming the Token Authentication:

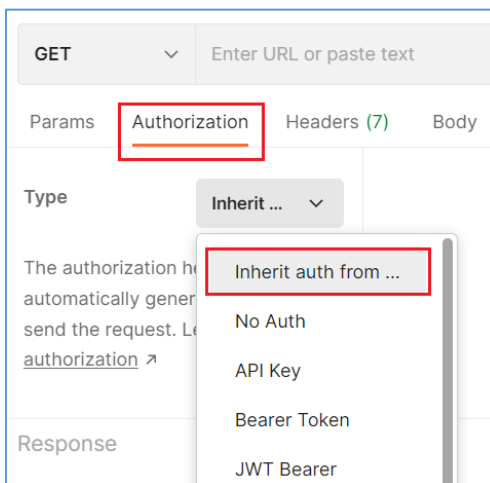
1. On the newly created workspace, click **Add a request**.



2. Enter the name of the request. For example, Part Request.

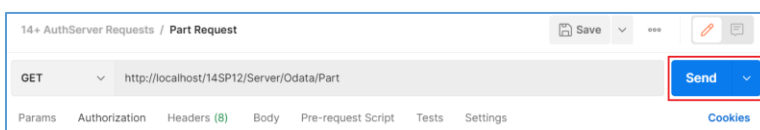


3. Select the **Authorization** tab and in the **Type** drop-down select **Inherit auth from parent**.

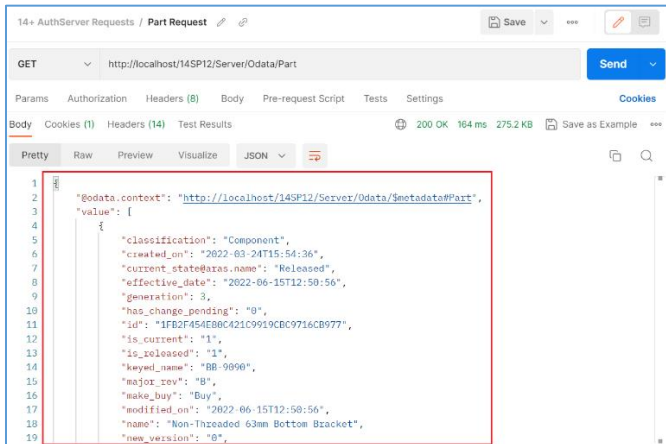


In order to confirm token authentication, run a simple query to get all the Part data from the Aras Innovator.

4. Select the **GET** request and enter the following request URL.
URL: `http://<servername>/<web alias>/Server/Odata/Part`
5. Click **Send** to run the request.



The request returns the JSON with all the parts within the database as follows:



```
1  {"@odata.context": "http://localhost:145P12/Server/Odata/$metadata#Part",
2  "value": [
3    {
4      "classification": "Component",
5      "created_on": "2022-03-24T15:54:36",
6      "current_state@aras.name": "Released",
7      "effective_date": "2022-06-15T12:50:56",
8      "generation": 3,
9      "has_change_pending": "B",
10     "id": "1F827454E986421C9919C0C9716CB977",
11     "is_current": "1",
12     "is_released": "1",
13     "keyed_name": "BB-9090",
14     "major_rev": "B",
15     "make_buy": "Buy",
16     "modified_on": "2022-06-15T12:50:56",
17     "name": "Non-Threaded 63mm Bottom Bracket",
18     "new_version": "B",
19   }
20 ]
```

This confirms that the token authentication and Innovator server connection was successful.