Aras Innovator 12.0

RESTful API

Document #: 12.0.02019055501

Last Modified: *7/28/2020*

Copyright Information

Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810

Phone: 978-806-9400
Fax: 978-794-9826

E-mail: support@aras.com

Website: https://www.aras.com

Table of Contents

Send Us Your Comments

Aras Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for future revisions.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where and what level of detail?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, indicate the document title, and the chapter, section, and page number (if available).

You can send comments to us in the following ways:

Email:
Support@aras.com
Subject: Aras Innovator Documentation

Or,

Postal service:
Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810
Attention: Aras Innovator Documentation

Or,

FAX:
978-794-9826
Attn: Aras Innovator Documentation

If you would like a reply, provide your name, email address, address, and telephone number.

If you have usage issues with the software, visit https://www.aras.com/support

Document Conventions

The following table highlights the document conventions used in the document:

Table 1: Document Conventions

Convention	Description
Bold	Emphasizes the names of menu items, dialog boxes, dialog box elements, and commands. Example: Click **OK**.
`Code`	Code examples appear in `courier` font. It may represent text you type or data you read.
`Yellow highlight`	Code highlighted in yellow draws attention to the code that is being indicated in the content.
`Yellow highlight with red text`	Red text highlighted in yellow indicates the code parameter that needs to be changed or replaced.
Italics	Reference to other documents.
Note:	Notes contain additional useful information.
Warning	Warnings contain important information. Pay special attention to information highlighted this way.
Successive menu choices	Successive menu choices may appear with a greater than sign (-->) between the items that you will select consecutively. Example: Navigate to **File** --> **Save** --> **OK**.

1 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.

2 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.

Note: In order to work correctly, the translator has to be aware of Aras Innovator's metadata. The OData interface exposes this metadata (item types, methods, etc.) using the $metadata endpoint.

2.1 Request Translation

This section describes the following:

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

2.1.1 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:

```
http://host:port/path/SampleService.svc/Categories(1)/Products?$top=2&$orderby=Name
\_____________________________________/\_____________________/ \__________________/
                  |                                |                     |
          service root URL                 resource path          query options
```

2.1.2 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:

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

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

2.1.3 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
}
```

2.1.4 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.

2.1.5 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
```

2.1.6 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
}
```

2.1.7 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**

2.1.8 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
```

2.1.9 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".

2.1.10 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",
  … remainig Part item properties
}
```

2.1.11 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": "…"
  … remainig Part item properties
}
```

2.1.12 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,
  … remainig Part item properties
}
```

2.2 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"
}
```

2.3 Request options

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

2.3.1 $filter

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

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

The following table describes how each element of the *$filter* option is translated to AML:

Table 2: 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.
Comparison Operators			
Operator	**Description**	**Example**	**AML**
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 3: Logical Operators

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

The $filter expression can contain a call to a built-in function. The following table shows how to translate a built-in function to AML:

Table 4: String Functions

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

2.3.1.1 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
```

2.3.1.2 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
```

2.3.2 $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=relate
d_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.

2.3.3 $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.

2.3.4 $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_numer desc
```

2.3.5 $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
  ]
}
```

2.3.6 $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",…},
  ]
}
```

2.3.7 $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.

2.3.8 $format

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

2.4 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 5: 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 *@aras.action=merge* annotation
POST	create	Uses *@aras.action=create* annotation
PATCH	update	Uses *@aras.action=update* annotation
PATCH	lock	Uses *@aras.action=lock* annotation
PATCH	unlock	Uses *@aras.action=unlock* annotation
DELETE	purge	Uses *@aras.action=purge* 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.

2.4.1 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",
  "method":
  {
    "name": "NewMethod",
    "method type": "C#",
    "method text": "…"
  }
}

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')
```

2.4.2 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.

2.4.3 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
}
```

2.4.4 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.

2.4.5 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.

3 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**

3.1 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 determinate 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", … remainig 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", … remainig Part item properties },
    … 9 remaining Part items
  ]
}
```

3.2 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"
}
```

3.3 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**

4 Response translation

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

4.1 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"
  }
}
```

5 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.

5.1 Begin file upload transaction

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

```
POST http://vault/odata/vault.BeginTransaction


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

Note: You will need to add the following header to the vault.BeginTransaction request:

Header key: `VAULTID Header`
Value: `67BBB9204FE84A8981ED83B049BA066`

5.2 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
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
```

5.3 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
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 the following header to the vault.CommitTransaction request:

Header key: `VAULTID Header`
Value: `67BBB9204FE84A8981ED83B049BA066`

6 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.

6.1 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}",
  "itemtype": "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}",
  "itemtype": "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
}
```

6.2 Filtering by Extended Property Definition

Use the filter function *is_defined* 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
  ]
}
```

6.3 Filtering Items Extended condition *in*

In order to implement the extended condition *in,* the OData Interface has introduced the filter function *in()*. 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,$root/Ite
mType($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’
  ]
}
```

6.4 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 'Bold'
  ]
}
```

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='Bold' or it subclasses
  ]
}
```

7 Using OAuth Tokens from the Authentication Server

The Authentication Server makes it possible for you to request an OAuth token from the server to use as an alternative to basic authentication. The information in this chapter has been adapted from the "Token Authentication Using the REST API" blog.

The Authentication server limits the number of applications that can request tokens by maintaining a list of registered applications that can request tokens. Applications that are not registered cannot get tokens by default. While you can use the default IOMApp Client registry, it is recommended that you create a new client registry to register your application. Use the following procedure:

1. Open the \OAuthServer\OAuth.config in the text editor of your choice.
2. Scroll down to the bottom until you see **<clientRegistry id="IOMApp">**.
3. Copy this node and all of its child nodes.
4. Paste them as a new section just below the original node.
5. Create a new ID for this registry. The ID should match the purpose of your application.
6. Make sure this new registry includes **enabled="true".**

Make note of the new registry ID. You will need to use it in the body of your token request.

7.1 Querying the OAuth Server Location

Aras Innovator allows many of its components to exist on separate servers. This means that the OAuth Server may not be available on the same server as the Instance URL. You can find this information by adding **/Server/OAuthServerDiscovery.aspx** to the end of your Instance, as shown in the following example:

URL: http://localhost/InnovatorServer/Server/OAuthServerDiscovery.aspx

You can make a Get request on the URL with an empty body. The request result is similar to the following:

```
{
  "locations": [
    {
      "uri": "http://localhost/InnovatorServer/oauthserver/"
    }
  ]
}
```

The URL is stored in the uri property of the JSON returned by the request. Use the JSON path `locations.uri` to retrieve the location information.

7.2 Getting the Token Endpoint

Once you have the OAuthServer URL you need to retrieve the exact URL to use to find the token query. This query is based on the OpenID Discovery Specification. Use the OAuthServer URL you created in section 7.1 and append .well-known/openid-configuration to the end of it as shown here:

`http://localhost/InnovatorServer/oauthserver/.well-known/openid-configuration.`

Use a simple Get request to query the URL without passing anything through the body. The request response looks like this:

```
{
  "issuer": "OAuthServer",
  "jwks_uri": "http://localhost/InnovatorServer/oauthserver/.well-
known/openid-configuration/jwks",
  "authorization_endpoint":
"http://localhost/InnovatorServer/oauthserver/connect/authorize",
  "token_endpoint":
"http://localhost/InnovatorServer/oauthserver/connect/token",
  "userinfo_endpoint":
"http://localhost/InnovatorServer/oauthserver/connect/userinfo",
  "end_session_endpoint":
"http://localhost/InnovatorServer/oauthserver/connect/endsession",
...
```

The JSON path is `token_endpoint`.

7.3 Retrieving the Token

The final request to retrieve the token uses the token endpoint URL used in the previous section. This token is linked to the user's credentials, which means that you will need to pass additional information before you can make your request. If you are using Postman, you can use the MultiPart Form to specify the following information in the body of the request:

- **Grant_type = Password or Tokens**
- **Scope: "Innovator"**

  Defines the scope of information accessed by the token. "Innovator" is used for in almost all cases.
- **Client_id: "IOMApp"**

  The ID of the client application that you configured previously.
- **Username: "YOUR_USER_NAME"**

  Any requests made using the returned token share the same permissions as the user.
- **Password: "YOUR_MD5_HASHED_PASSWORD"**
- **Database: "YOUR_DATABASE_NAME"**

  The name of the database that the token can access. "InnovatorSolutions" is the default Aras Innovator database name.

Once you configure the body, send a POST request to the token endpoint URL. The body must contain the properties listed previously. The response contains the OAuth token name and the amount of time you have before the token expires, as shown in the following example:

```
{
  "access_token": "YOUR_TOKEN",
  "expires_in": 3600,
  "token_type": "Bearer"
}
```

7.4 Making a Request Using a Token

Previously, authentication was done by passing the user name and hashed password into the AUTHUSER and AUTHPASSWORD headers. When using tokens, the standard HTTP Authorization header is used. Token Authentication uses the following header format:

Authorization: Bearer {YOUR_TOKEN}

Note: The word Bearer must appear before your token in the header.

If you are using Postman, you should be able to configure authentication differently from other headers, allowing the word “Bearer” to be automatically appended to the header.