Aras Innovator OData Interface

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

Request Translation

This section describes the following:

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

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
```

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", … remaining 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.

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 complete 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_CA
"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": "…", … remaining 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
}
```

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": "…", … remaining Part CAD item properties },
… remaining Part CAD items
]
}
```

This request returns the collection of `Part CAD` relationship items.

> **Important**
>
> 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 lowercase letter are identified as regular properties.

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.

File Properties

A Request for a File property returns the File item and 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.

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 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
```

Poly Items

When the client requests either an Item property or a collection with the Poly Item 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
}
```

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 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

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

Define operations

Before using 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".

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
}
```

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
}
```

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
}
```

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) are taken from the URI.

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

Request options

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

$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 the 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.

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"`

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.

String Functions

String Functions

Function	Example	AML

`contains`	`contains(company_name,'freds')`	`<company_name condition=“like”> %freds% </company_name>`
`endswith`	`endswith(company_name, 'freds')`	`<company_name condition=“like”> %freds </company_name>`
`startswith`	`startswith(company_name,'Alfr')`	`<company_name condition=“like”> Alfr% </company_name>`

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
```

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. 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
```

$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 properties },
{ "id": "…", … remaining Part CAD item properties },
… remaining 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
},
… remaining Part CAD item properties
},
… remaining Part CAD items
],
… remaining Part item properties
}
```

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

$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 *$select* option, the OData service will not return properties with a null value. To return all properties, the client must use the star "*" value for the *$select* query option in the request.

$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
```

$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
]
}
```

$skip

The $skip option allows a specified number of items in the resulting collection to be skipped. The following example excludes parts 01-10 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",…},
]
}
```

$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.

$format

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

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.

HTTP Methods

HTTP method	Aras Innovator action	Notes
`GET`	`get`	
`POST`	`add`	It can be sent only to a collection.
`PATCH`	`edit`	Can be sent only to a specific entity
`PUT`	`edit`	Update simple property or single navigation property
`DELETE`	`delete`	Can be sent only to a specific entity
`PATCH`	`merge`	Uses the `@aras.action=merge` annotation
`POST`	`create`	Uses the `@aras.action=create` annotation
`PATCH`	`update`	Uses the `@aras.action=update` annotation
`PATCH`	`lock`	Uses the `@aras.action=lock` annotation
`PATCH`	`unlock`	Uses the `@aras.action=unlock` annotation
`DELETE`	`purge`	Uses the `@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 the *Prefer* header. When the client specified *return=minimal*, the OData service returns the *204 No Content* status code.

The OData specification doesn't define service behavior if the *return* preference is omitted. To be consistent with Aras Innovator, the OData service uses *return=representation* as the default.

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

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 will create a new version of the item.

The client can also create a new relationship and 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')
```

Update

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')
```

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
```

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 a time.

Upsert

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
}
```

Delete

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 a time.

Important

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.

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.

Batch Requests

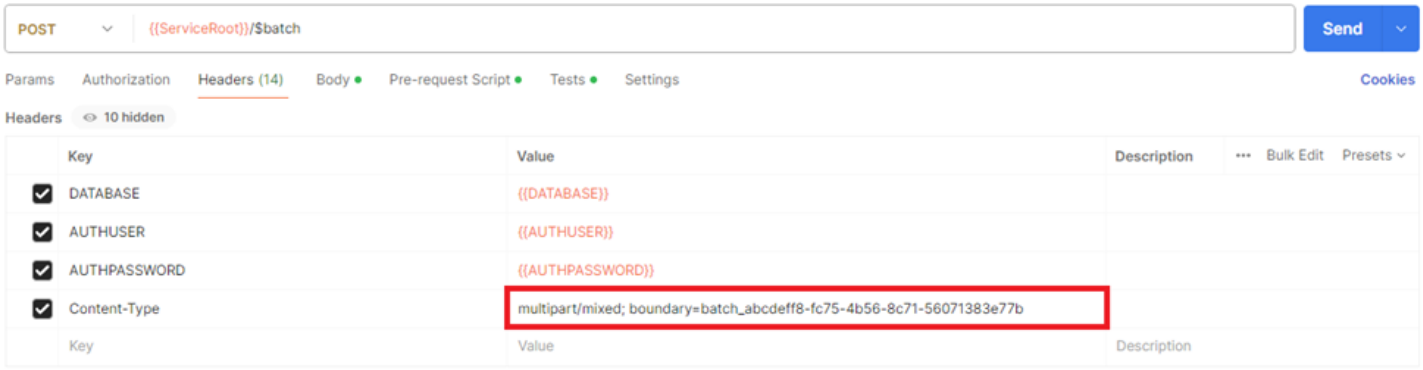
In OData, batching enables developers to bundle multiple requests into a single call to the *$batch* endpoint (see OData documentation). A batch request is represented as a Multipart MIME message, a standard format allowing the representation of multiple parts, each of which may have a different content type, within a single request.

Batch requests are submitted as a single **HTTP POST** request to the $batch endpoint of a Web Service. Any requests nested in a batch are executed sequentially and synchronously.

Batch Request Headers

This section describes the headers for an HTTP POST request to the $batch endpoint.

Content-Type: Required. This header must have the multipart/mixed value and a boundary specification as defined in the Batch Request Body section (see an example below).



- **OData-Version**: Optional. If using this header, set the value to *4.0*.
- **Prefer**: Optional. By default, the CWS engine stops processing a batch request when any of the nested requests return an error. To continue processing the batch in case of an error in a nested request, include the Prefer header with the value *odata.continue-on-error*.

Batch Request Body

The body of a batch request comprises a series of individual requests and Change Sets , each represented as a distinct MIME part (i.e., separated by the boundary defined in the *Content-Type* header). An individual request in the context of a batch request can be

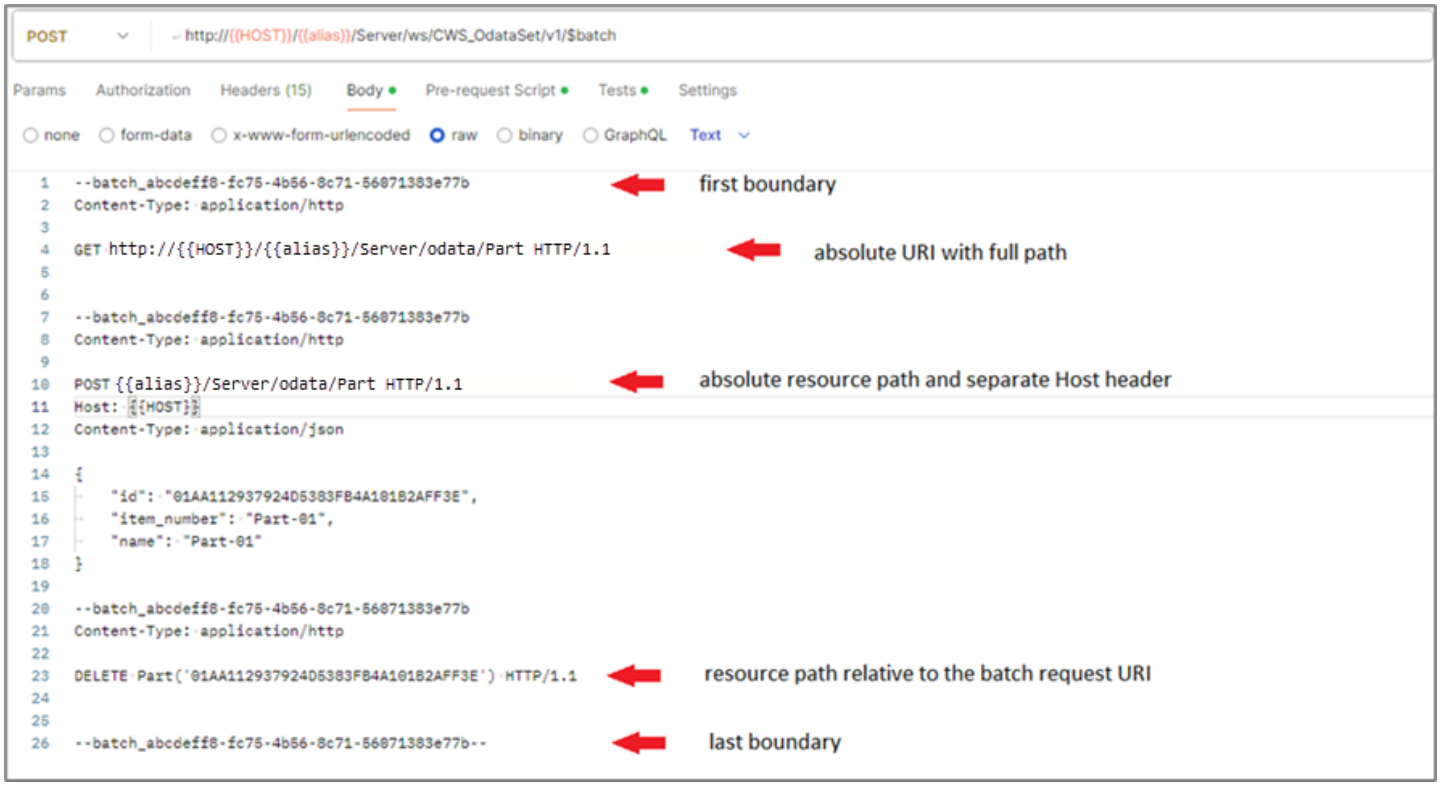
- Data request ;

- Data Modification request;
- Action invocation request.

A MIME part representing an individual request must include a *Content-Type* header with the value *application/http* and should contain a *Content-Transfer-Encoding* header with the value *binary*. CWS supports three Request-URI formats of HTTP requests serialized within MIME part bodies:

- Absolute URI with schema, host, port, and absolute resource path
  - The absolute URI of each request in a batch must be the same as the URI of the source batch request.
- Absolute resource path and separate Host header
- Resource path relative to the batch request URI

The following sample batch request demonstrates each of these formats.



Important

The formatting of the batch request body is essential. An empty line must separate each part. Add an extra empty line after requests without a body (see line #4 above).

Change Sets

A **Change Set** is an atomic unit of work consisting of an unordered group of one or more Data Modification requests or Action invocation requests.

All Change Sets must meet the following requirements:

- Change Sets may not contain any GET requests or other Change Sets.
- The order of requests within a Change Set is significant; the requests within a Change Set are processed sequentially.
- If one request within a Change Set fails, then all requests in the Change Set are rolled back.
- Each Change Set must be a multipart MIME document with one part for each operation that makes up the Change Set.
- Each part representing an operation in the Change Set must include the same headers (Content-Type and Content-Transfer-Encoding) and associated values previously described for operations (see below).

```
--batch_abcdeff8-fc75-4b56-8c71-56071383e77b
Content-Type: application/http

GET Part HTTP/1.1


--batch_abcdeff8-fc75-4b56-8c71-56071383e77b
Content-Type: multipart/mixed; boundary=changeset_77162fcd-b8da-41ac-a9f8-9357efbbd

--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd
Content-Type: application/http
Content-Transfer-Encoding: binary
Content-ID: 1

POST Part HTTP/1.1
Content-Type: application/json

{
    "id": "01AA112937924D5383FB4A101B2AFF3E",
    "item_number": "Part-01",
    "name": "Part-01"
}

--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd
Content-Type: application/http
Content-ID: 2

POST Part HTTP/1.1
Content-Type: application/json

{
    "id": "00BB112937924D5383FB4A101B2AFF56",
    "item_number": "Part-00",
    "name": "Part-00",
    "Part BOM": [
        {
            "id": "00E01E12F3004B3283F3C88B9349E4A0",
            "related_id@odata.bind":"$1"
        }
    ]
}

--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd
Content-Type: application/http
Content-ID: 3

DELETE $2 HTTP/1.1


--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd--

--batch_abcdeff8-fc75-4b56-8c71-56071383e77b
Content-Type: application/http

DELETE Part('01AA112937924D5383FB4A101B2AFF3E') HTTP/1.1


--batch_abcdeff8-fc75-4b56-8c71-56071383e77b--
```

change set boundary

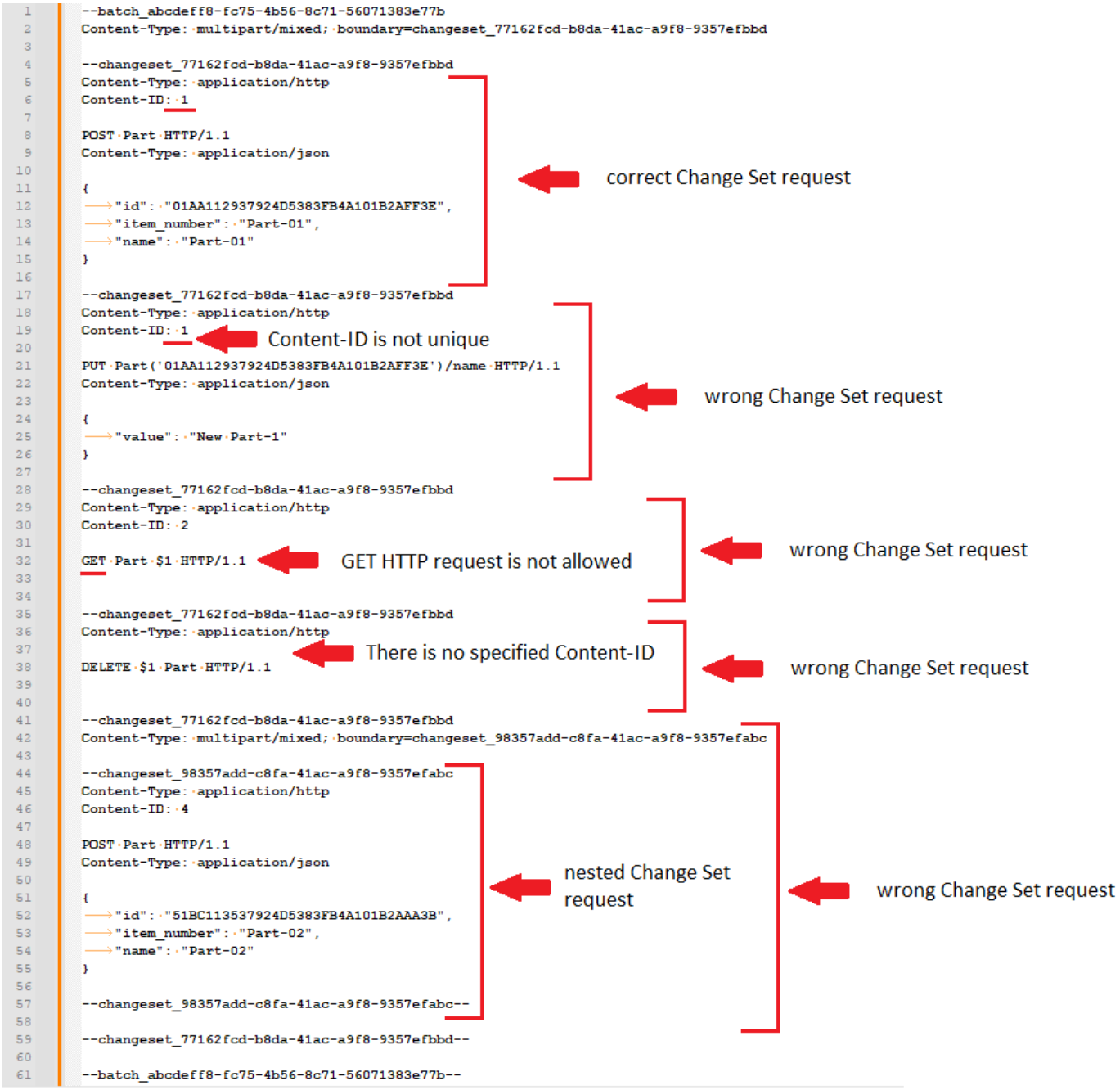
first change set request

second change set request

third change set request

last boundary (end of change set)

- Each request within a Change Set must specify a *Content-ID* header with a value unique within the batch request.
- Entities created by an request within a Change Set can be referenced by subsequent requests within the same Change Set in places where a resource path to an existing entity can be specified. The temporary resource path for a newly inserted entity is the value of the Content-ID header prefixed with a $ character. Examples of correct and wrong Change Set requests are below.




```
--batch_abcdeff8-fc75-4b56-8c71-56071383e77b
Content-Type: multipart/mixed; boundary=changeset_77162fcd-b8da-41ac-a9f8-9357efbbd

--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd
Content-Type: application/http
Content-ID: 1

POST Part HTTP/1.1
Content-Type: application/json

{
    "id": "01AA112937924D5383FB4A101B2AFF3E",
    "item_number": "Part-01",
    "name": "Part-01"
}

--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd
Content-Type: application/http
Content-ID: 1

PUT Part('01AA112937924D5383FB4A101B2AFF3E')/name HTTP/1.1
Content-Type: application/json

{
    "value": "New Part-1"
}

--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd
Content-Type: application/http
Content-ID: 2

GET Part $1 HTTP/1.1

--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd
Content-Type: application/http

DELETE $1 Part HTTP/1.1

--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd
Content-Type: multipart/mixed; boundary=changeset_98357add-c8fa-41ac-a9f8-9357efabc

--changeset_98357add-c8fa-41ac-a9f8-9357efabc
Content-Type: application/http
Content-ID: 4

POST Part HTTP/1.1
Content-Type: application/json

{
    "id": "51BC113537924D5383FB4A101B2AAA3B",
    "item_number": "Part-02",
    "name": "Part-02"
}

--changeset_98357add-c8fa-41ac-a9f8-9357efabc--

--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd--

--batch_abcdeff8-fc75-4b56-8c71-56071383e77b--
```

Responding to a Batch Request

Requests within a batch are evaluated according to the same semantics used when processing individual requests. The sample batch request above generated the following response.

```
--batch_abcdeff8-fc75-4b56-8c71-56071383e77b
Content-Type: application/http
HTTP/1.1 200 OK
OData-Version:4.0
Content-Type:application/json;odata.metadata=minimal;IEEE754Compatible=false
{
"@odata.context": "http://{{host}}/{{alias}}/Server/odata/$metadata#Part",
"value": []
}
--batch_abcdeff8-fc75-4b56-8c71-56071383e77b
Content-Type: multipart/mixed; boundary=changeset_77162fcd-b8da-41ac-a9f8-9357efbbd
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd
Content-Type: application/http
Content-ID: 1
HTTP/1.1 201 Created
Preference-Applied:return=representation
Location:http://{{host}}/{{alias}}/Server/odata/Part('01AA112937924D5383FB4A101B2AFF3E')
OData-Version:4.0
Content-Type:application/json;odata.metadata=minimal;IEEE754Compatible=false
{
"@odata.context": "http://{{host}}/{{alias}}/Server/odata/$metadata#Part/$entity",
"id": "01AA112937924D5383FB4A101B2AFF3E",
"major_rev": "A",
"name": "Part-01",
"item_number": "Part-01"
}
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd
Content-Type: application/http
Content-ID: 2
HTTP/1.1 201 Created
Preference-Applied:return=representation
Location:http://{{host}}/{{alias}}/Server/odata/Part('00BB112937924D5383FB4A101B2AFF56')
OData-Version:4.0
Content-Type:application/json;odata.metadata=minimal;IEEE754Compatible=false
{
"@odata.context": "http://{{host}}/{{alias}}/Server/odata/$metadata#Part/$entity",
"id": "00BB112937924D5383FB4A101B2AFF56",
"major_rev": "A",
"name": "Part-00",
"item_number": "Part-00",
"Part_BOM": [
{
"id": "00E01E12F3004B3283F3C88B9349E4A0",
"sort_order": 128
}
]
}
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd
Content-Type: application/http
Content-ID: 3
HTTP/1.1 204 NoContent
OData-Version:4.0
Content-Type:text/plain
--changeset_77162fcd-b8da-41ac-a9f8-9357efbbd--
--batch_abcdeff8-fc75-4b56-8c71-56071383e77b
Content-Type: application/http
```

```
HTTP/1.1 204 NoContent
OData-Version:4.0
Content-Type:text/plain
--batch_abcdeff8-fc75-4b56-8c71-56071383e77b--
```