**Section 1: Basic create and update operations**

1. Create a new contact, named Peter Cambel.

**Request**

HTTPCopy

POST https://[Organization URI]/api/data/v9.0/contacts HTTP/1.1

Content-Type: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

{

"firstname": "Peter",

"lastname": "Cambel"

}

**Response**

HTTPCopy

HTTP/1.1 204 No Content

OData-Version: 4.0

OData-EntityId: https://[Organization URI]/api/data/v9.0/contacts(60f77a42-5f0e-e611-80e0-00155da84c03)

1. Update the contact with values for annual income ($80,000) and job title (Junior Developer).

**Request**

HTTPCopy

PATCH https://[Organization URI]/api/data/v9.0/contacts(60f77a42-5f0e-e611-80e0-00155da84c03) HTTP/1.1

Content-Type: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

{

"annualincome": 80000,

"jobtitle": "Junior Developer" }

**Response**

HTTPCopy

HTTP/1.1 204 No Content

1. Retrieve the contact with its set of explicitly initialized properties. The fullname is a read-only property that is calculated from the firstname and lastname properties, which were explicitly initialized when the instance was created. In contrast, the description property was not explicitly initialized, so it retains its default value, a null string.

Note that the response, in addition to the requested values and typical headers, also automatically returns the following types of additional information:

* + The primary ID for the current entity type, here contactid.
  + An *ETag* value, denoted by the @odata.etag key, which identifies the specific version of the resource requested. For more information, see [Perform conditional operations using the Web API](https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/webapi/perform-conditional-operations-using-web-api).
  + The metadata context, denoted by the @odata.context key, provides a way to compare query results to determine if they came from the same query.
  + A \_transactioncurrencyid\_value that indicates the local currency of the monetary transaction.

**Request**

HTTPCopy

GET https://[Organization URI]/api/data/v9.0/contacts(60f77a42-5f0e-e611-80e0-00155da84c03)?$select=fullname,annualincome,jobtitle,description HTTP/1.1

Accept: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

**Response**

HTTPCopy

HTTP/1.1 200 OK

Content-Type: application/json; odata.metadata=minimal

OData-Version: 4.0

{

"@odata.context":"https://[Organization URI]/api/data/v9.0/$metadata#contacts(fullname,annualincome,jobtitle,description)/$entity",

"@odata.etag":"W/\"628883\"",

"fullname":"Peter Cambel",

"annualincome":80000.0000,

"jobtitle":"Junior Developer",

"description":null,

"\_transactioncurrencyid\_value":"0d4ed62e-95f7-e511-80d1-00155da84c03",

"contactid":"60f77a42-5f0e-e611-80e0-00155da84c03"

}

1. Update the contact entity instance by supplying new values to these same properties.

**Request**

HTTPCopy

PATCH https://[Organization URI]/api/data/v9.0/contacts(60f77a42-5f0e-e611-80e0-00155da84c03) HTTP/1.1

Content-Type: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

{

"jobtitle": "Senior Developer",

"annualincome": 95000,

"description": "Assignment to-be-determined"

}

**Response**

HTTPCopy

HTTP/1.1 204 No Content

1. Explicitly set a single property, the primary phone number. Note this is a PUT request and that the JSON key named value is used when performing operations on individual properties.

**Request**

HTTPCopy

PUT https://[Organization URI]/api/data/v9.0/contacts(60f77a42-5f0e-e611-80e0-00155da84c03)/telephone1 HTTP/1.1

Content-Type: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

{

"value": "555-0105"

}

**Response**

HTTPCopy

HTTP/1.1 204 No Content

**Request**

HTTPCopy

GET https://[Organization URI]/api/data/v9.0/contacts(60f77a42-5f0e-e611-80e0-00155da84c03)/telephone1 HTTP/1.1

Accept: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

**Response**

HTTPCopy

HTTP/1.1 200 OK

Content-Type: application/json; odata.metadata=minimal

OData-Version: 4.0

{

"@odata.context":"https://[Organization URI]/api/data/v9.0/$metadata#contacts(60f77a42-5f0e-e611-80e0-00155da84c03)/telephone1",

"value":"555-0105"

}

**Request**

HTTPCopy

POST https://[Organization URI]/api/data/v9.0/contacts?$select=fullname,annualincome,jobtitle,contactid HTTP/1.1

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: return=representation

{

"firstname": "Peter\_Alt",

"lastname": "Cambel",

"jobtitle": "Junior Developer",

"annualincome": 80000,

"telephone1": "555-0110"

}

**Response**

HTTPCopy

HTTP/1.1 201 Created

Content-Type: application/json; odata.metadata=minimal

Preference-Applied: return=representation

OData-Version: 4.0

{

"@odata.context":"https://[Organization URI]/api/data/v9.0/$metadata#contacts/$entity","@odata.etag":"W/\"758870\"","\_transactioncurrencyid\_value":"0d4ed62e-95f7-e511-80d1-00155da84c03","annualincome":80000.0000,"contactid":"199250b7-6cbe-e611-80f7-00155da84c08","jobtitle":"Junior Developer","fullname":"Peter\_Alt Cambel"

}

1. Update this similar contact and also return instance information in the same operation. Again, this capability is enabled by the Prefer: return=representation header.

**Request**

HTTPCopy

POST https://[Organization URI]/api/data/v9.0/contacts?$select=fullname,annualincome,jobtitle,contactid HTTP/1.1

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: return=representation

{

"firstname": "Peter\_Alt",

"lastname": "Cambel",

"jobtitle": "Junior Developer",

"annualincome": 80000,

"telephone1": "555-0110"

}

**Response**

HTTPCopy

HTTP/1.1 201 Created

Content-Type: application/json; odata.metadata=minimal

Preference-Applied: return=representation

OData-Version: 4.0

{

"@odata.context":"https://[Organization URI]/api/data/v9.0/$metadata#contacts/$entity","@odata.etag":"W/\"758870\"","\_transactioncurrencyid\_value":"0d4ed62e-95f7-e511-80d1-00155da84c03","annualincome":80000.0000,"contactid":"199250b7-6cbe-e611-80f7-00155da84c08","jobtitle":"Junior Developer","fullname":"Peter\_Alt Cambel"

}

**Section 2: Create with association**

This section creates a new account instance named Contoso, Ltd. and associates it to an existing contact, Peter Cambel, which was created in [Section 1](https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/webapi/web-api-basic-operations-sample#bkmk_section1). This creation and association is performed in a single POST operation.

1. Create the Contoso, Ltd. account and set its primary contact attribute to the existing contact Peter Cambel. The @odata.bind annotation indicates that an association is being created, here binding the primarycontactid single-valued navigation property to an existing contact, Peter Cambel.

**Request**

HTTPCopy

POST https://[Organization URI]/api/data/v9.0/accounts HTTP/1.1

Content-Type: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

{

"name": "Contoso Inc",

"telephone1": "555-5555",

"primarycontactid@odata.bind": "https://[Organization URI]/api/data/v9.0/contacts(60f77a42-5f0e-e611-80e0-00155da84c03)"

}

**Response**

HTTPCopy

HTTP/1.1 204 No Content

OData-Version: 4.0

OData-EntityId: https://[Organization URI]/api/data/v9.0/accounts(65f77a42-5f0e-e611-80e0-00155da84c03)

1. Retrieve the primary contact for the account Contoso, Ltd., again using $expand with the primarycontactid single-valued navigation property to access the associated [contact EntityType](https://docs.microsoft.com/en-us/dynamics365/customer-engagement/web-api/contact) record.

**Request**

HTTPCopy

GET https://[Organization URI]/api/data/v9.0/accounts(65f77a42-5f0e-e611-80e0-00155da84c03)?$select=name,&$expand=primarycontactid($select=fullname,jobtitle,annualincome) HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

**Response**

HTTPCopy

HTTP/1.1 200 OK

Content-Type: application/json; odata.metadata=minimal

OData-Version: 4.0

{

"@odata.context":"https://[Organization URI]/api/data/v9.0/$metadata#accounts(name,primarycontactid,primarycontactid(fullname,jobtitle,annualincome))/$entity",

"@odata.etag":"W/\"628886\"",

"name":"Contoso Inc",

"accountid":"65f77a42-5f0e-e611-80e0-00155da84c03",

"primarycontactid":{

"@odata.etag":"W/\"628885\"",

"fullname":"Peter Cambel",

"jobtitle":"Senior Developer",

"annualincome":95000.0000,

"\_transactioncurrencyid\_value":"0d4ed62e-95f7-e511-80d1-00155da84c03",

"contactid":"60f77a42-5f0e-e611-80e0-00155da84c03"

}

}

**Section 3: Create related entities (deep insert)**

This section demonstrates how to create an entity instance and related entity instances, in a single POST request. Using this method, all instances are newly created; there are no existing instances to associate with. This approach has two advantages. It is more efficient, replacing multiple simpler creation and association operations with one combined operation. Also, it is [atomic](https://msdn.microsoft.com/library/aa719484(v=vs.71).aspx), where either the entire operation succeeds and all the related objects are created, or the operation fails and none are created.

This section creates an account, its primary contact, and a set of tasks for that contact in one request.

1. Create the account Fourth Coffee and its primary contact Susie Curtis and her three related tasks in one operation. Note the use of the single-valued primarycontactid navigation property and the collection-valued navigation property Contact\_Tasks to define these relationships, respectively. Single-valued navigational properties take an object value, whereas collection-valued navigation properties take an array value.

**Request**

HTTPCopy

POST https://[Organization URI]/api/data/v9.0/accounts HTTP/1.1

Content-Type: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

{

"name": "Fourth Coffee",

"primarycontactid": {

"firstname": "Susie",

"lastname": "Curtis",

"jobtitle": "Coffee Master",

"annualincome": 48000,

"Contact\_Tasks": [

{

"subject": "Sign invoice",

"description": "Invoice #12321",

"scheduledend": "2016-04-19T00:00:00-07:00"

},

{

"subject": "Setup new display",

"description": "Theme is - Spring is in the air",

"scheduledstart": "2016-04-20T00:00:00-07:00"

},

{

"subject": "Conduct training",

"description": "Train team on making our new blended coffee",

"scheduledstart": "2016-06-01T00:00:00-07:00"

}

]

}

}

**Response**

HTTPCopy

HTTP/1.1 204 No Content

OData-Version: 4.0

OData-EntityId: https://[Organization URI]/api/data/v9.0/accounts(6af77a42-5f0e-e611-80e0-00155da84c03)

1. Selectively retrieve the newly created Fourth Coffee account and its primary contact. An expansion is performed on the single-valued navigation property primarycontactid.

**Request**

HTTPCopy

GET https://[Organization URI]/api/data/v9.0/accounts(6af77a42-5f0e-e611-80e0-00155da84c03)?$select=name,&$expand=primarycontactid($select=fullname,jobtitle,annualincome) HTTP/1.1

Accept: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

**Response**

HTTPCopy

HTTP/1.1 200 OK

Content-Type: application/json; odata.metadata=minimal

OData-Version: 4.0

{

"@odata.context":"https://[Organization URI]/api/data/v9.0/$metadata#accounts(name,primarycontactid,primarycontactid(fullname,jobtitle,annualincome))/$entity",

"@odata.etag":"W/\"628902\"",

"name":"Fourth Coffee",

"accountid":"6af77a42-5f0e-e611-80e0-00155da84c03",

"primarycontactid":{

"@odata.etag":"W/\"628892\"",

"fullname":"Susie Curtis",

"jobtitle":"Coffee Master",

"annualincome":48000.0000,

"\_transactioncurrencyid\_value":"0d4ed62e-95f7-e511-80d1-00155da84c03",

"contactid":"6bf77a42-5f0e-e611-80e0-00155da84c03"

}

}

1. Selectively retrieve the tasks associated with the primary contact retrieved in the previous operation. An expansion is performed on the collection-valued navigation property Contact\_Tasks.

**Request**

HTTPCopy

GET https://[Organization URI]/api/data/v9.0/contacts(6bf77a42-5f0e-e611-80e0-00155da84c03)?$select=fullname,&$expand=Contact\_Tasks($select=subject,description,scheduledstart,scheduledend) HTTP/1.1

Accept: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

**Response**

HTTPCopy

HTTP/1.1 200 OK

Content-Type: application/json; odata.metadata=minimal

OData-Version: 4.0

{

"@odata.context":"https://[Organization URI]/api/data/v9.0/$metadata#contacts(fullname,Contact\_Tasks,Contact\_Tasks(subject,description,scheduledstart,scheduledend))/$entity",

"@odata.etag":"W/\"628892\"",

"fullname":"Susie Curtis",

"contactid":"6bf77a42-5f0e-e611-80e0-00155da84c03",

"Contact\_Tasks":[

{

"@odata.etag":"W/\"628903\"",

"subject":"Sign invoice",

"description":"Invoice #12321",

"scheduledstart":"2016-04-19T00:00:00Z",

"scheduledend":"2016-04-19T00:00:00Z",

"activityid":"6cf77a42-5f0e-e611-80e0-00155da84c03"

},

{

"@odata.etag":"W/\"628905\"",

"subject":"Setup new display",

"description":"Theme is - Spring is in the air",

"scheduledstart":"2016-04-20T00:00:00Z",

"scheduledend":"2016-04-20T00:00:00Z",

"activityid":"6df77a42-5f0e-e611-80e0-00155da84c03"

},

{

"@odata.etag":"W/\"628907\"",

"subject":"Conduct training",

"description":"Train team on making our new blended coffee",

"scheduledstart":"2016-06-01T00:00:00Z",

"scheduledend":"2016-06-01T00:00:00Z",

"activityid":"6ef77a42-5f0e-e611-80e0-00155da84c03"

}

]

}

**Section 4: Associate and disassociate existing entities**

This section demonstrates how to associate and disassociate existing entity instances. Forming an association requires the use of a reference URI and relationship object, which are then sent in a POST request. Disassociating requires sending a DELETE request to the reference URI for that association. First a one-to-many association is formed between a contact and an account. Then a many-to-many association is formed between a competitor and one or more opportunities.

1. Add Peter Cambel as a contact to the account Fourth Coffee using the contact\_customer\_accounts collection-valued navigation property. Note the use of the special key @odata.id to specify the associated record.

**Request**

HTTPCopy

POST https://[Organization URI]/api/data/v9.0/accounts(6af77a42-5f0e-e611-80e0-00155da84c03)/contact\_customer\_accounts/$ref HTTP/1.1

Content-Type: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

{

"@odata.id": "https://[Organization URI]/api/data/v9.0/contacts(60f77a42-5f0e-e611-80e0-00155da84c03)"

}

**Response**

HTTPCopy

HTTP/1.1 204 No Content

1. Confirm the previous operation by retrieving the collection of contacts for the account Fourth Coffee. The response contains the array with a single element, the recently assigned contact Peter Cambel.

**Request**

HTTPCopy

GET https://[Organization URI]/api/data/v9.0/accounts(6af77a42-5f0e-e611-80e0-00155da84c03)/contact\_customer\_accounts?$select=fullname,jobtitle HTTP/1.1

Accept: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

**Response**

HTTPCopy

HTTP/1.1 200 OK

Content-Type: application/json; odata.metadata=minimal

OData-Version: 4.0

{

"@odata.context":"https://[Organization URI]/api/data/v9.0/$metadata#contacts(fullname,jobtitle)","value":[

{

"@odata.etag":"W/\"632481\"","fullname":"Peter Cambel","jobtitle":"Senior Developer","contactid":"00b6e0e2-b010-e611-80e1-00155da84c03"

}

]

}

**Request**

HTTPCopy

DELETE https://[Organization URI]/api/data/v9.0/accounts(6af77a42-5f0e-e611-80e0-00155da84c03)/contact\_customer\_accounts/$ref?$id=https://[Organization URI]/api/data/v9.0/contacts(60f77a42-5f0e-e611-80e0-00155da84c03) HTTP/1.1

Content-Type: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

**Response**

HTTPCopy

HTTP/1.1 204 No Content

1. Create a competitor named Adventure Works.

**Request**

HTTPCopy

POST https://[Organization URI]/api/data/v9.0/competitors HTTP/1.1

Content-Type: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

{

"name": "Adventure Works",

"strengths": "Strong promoter of private tours for multi-day outdoor adventures"

}

**Response**

HTTPCopy

HTTP/1.1 204 No Content

OData-Version: 4.0

OData-EntityId: https://[Organization URI]/api/data/v9.0/accounts(77f77a42-5f0e-e611-80e0-00155da84c03)

1. Create an opportunity named River rafting adventure.

**Request**

HTTPCopy

POST https://[Organization URI]/api/data/v9.0/opportunities HTTP/1.1

Content-Type: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

{

"name": "River rafting adventure",

"description": "Sales team on a river-rafting offsite and team building"

}

**Response**

HTTPCopy

HTTP/1.1 204 No Content

OData-Version: 4.0

OData-EntityId: https://[Organization URI]/api/data/v9.0/opportunities(7cf77a42-5f0e-e611-80e0-00155da84c03)

1. Associate this new opportunity to this new competitor. Note that the same general syntax is used in this many-to-many association as was used in the previous one-to-many association.

**Request**

HTTPCopy

POST https://[Organization URI]/api/data/v9.0/opportunities(7cf77a42-5f0e-e611-80e0-00155da84c03)/opportunitycompetitors\_association/$ref HTTP/1.1

Content-Type: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

{

"@odata.id": "https://[Organization URI]/api/data/v9.0/competitors(77f77a42-5f0e-e611-80e0-00155da84c03)"

}

**Response**

HTTPCopy

HTTP/1.1 204 No Content

1. Selectively retrieve all the opportunities associated with the competitor Adventure Works. An array is returned containing a single opportunity.

**Request**

HTTPCopy

GET https://[Organization URI]/api/data/v9.0/competitors(77f77a42-5f0e-e611-80e0-00155da84c03)?$select=name,&$expand=opportunitycompetitors\_association($select=name,description) HTTP/1.1

Accept: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

**Response**

HTTPCopy

HTTP/1.1 200 OK

{

"@odata.context":"https://[Organization URI]/api/data/v9.0/$metadata#competitors(name,opportunitycompetitors\_association,opportunitycompetitors\_association(name,description))/$entity",

"@odata.etag":"W/\"628913\"",

"name":"Adventure Works",

"competitorid":"77f77a42-5f0e-e611-80e0-00155da84c03",

"opportunitycompetitors\_association":[

{

"@odata.etag":"W/\"628917\"",

"name":"River rafting adventure",

"description":"Sales team on a river-rafting offsite and team building",

"opportunityid":"7cf77a42-5f0e-e611-80e0-00155da84c03"

}

]

}

1. Dissociate the opportunity from the competitor. Note again, that this has the same general syntax used to remove a one-to-many association.

**Request**

HTTPCopy

DELETE https://[Organization URI]/api/data/v9.0/opportunities(7cf77a42-5f0e-e611-80e0-00155da84c03)/opportunitycompetitors\_association/$ref?$id=https://[Token-CRM-Org-Name]/Contoso/api/data/v8.1/competitors(77f77a42-5f0e-e611-80e0-00155da84c03) HTTP/1.1

Content-Type: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

**Response**

HTTPCopy

HTTP/1.1 204 No Content

**Section 5: Delete entities (sample cleanup)**

1. Each element of the collection of entity URLs is deleted. The first is a contact record for Peter Cambel.

**Request**

HTTPCopy

DELETE https://[Organization URI]/api/data/v9.0/contacts(60f77a42-5f0e-e611-80e0-00155da84c03) HTTP/1.1

Content-Type: application/json

OData-MaxVersion: 4.0

OData-Version: 4.0

**Response**

HTTPCopy

HTTP/1.1 204 No Content

1. Subsequent iterations through the collection delete the remaining records.

**Request**

HTTPCopy

DELETE https://[Organization URI]/api/data/v9.0/accounts(65f77a42-5f0e-e611-80e0-00155da84c03) HTTP/1.1

. . .

DELETE https://[Organization URI]/api/data/v9.0/accounts(6af77a42-5f0e-e611-80e0-00155da84c03) HTTP/1.1

. . .

DELETE https://[Organization URI]/api/data/v9.0/contacts(6bf77a42-5f0e-e611-80e0-00155da84c03) HTTP/1.1

. . .

## Selecting specific properties

GET https://[Organization URI]/api/data/v9.0/contacts(b848fdee-c143-e611-80d5-00155da84802)?$select=fullname,jobtitle,annualincome HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=10, odata.include-annotations=OData.Community.Display.V1.FormattedValue

### Standard query functions

GET https://[Organization URI]/api/data/v9.0/contacts?$select=fullname,jobtitle,annualincome&$filter=contains(fullname,'(sample)') HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=10, odata.include-annotations=OData.Community.Display.V1.FormattedValue

### Dataverse query functions

GET https://[Organization URI]/api/data/v9.0/contacts?$select=fullname,jobtitle,annualincome&$filter=Microsoft.Dynamics.CRM.LastXHours(PropertyName='createdon',PropertyValue='1') HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=10, odata.include-annotations=OData.Community.Display.V1.FormattedValue

## Using operators

GET https://[Organization URI]/api/data/v9.0/contacts?$select=fullname,jobtitle,annualincome&$filter=contains(fullname,'(sample)')%20and%20annualincome%20gt%2055000 HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=10, odata.include-annotations=OData.Community.Display.V1.FormattedValue

## Setting precedence

You will use parentheses to establish the order in which your conditions are evaluated.

In this example, we are requesting a list of all contacts with fullname containing (sample), jobtitle containing either senior or specialist, and annualincome greater than 55000. To get the results we want, parentheses are used to group the jobtitle filters together. Since all operators have the same precedence, omitting the parentheses will give the or operator the same precedence as the and operators. Filters are applied from left to right. The order in which these statements appear in the filter can affect the results. This is what the query in this example looks like: $filter=contains(fullname,'(sample)') and (contains(jobtitle,'senior') or contains(jobtitle,'specialist')) and annualincome gt 55000.

GET https://[Organization URI]/api/data/v9.0/contacts?$select=fullname,jobtitle,annualincome&$filter=contains(fullname,'(sample)')%20and%20(contains(jobtitle,'senior')%20or%20contains(jobtitle,'specialist'))%20and%20annualincome%20gt%2055000 HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=10, odata.include-annotations=OData.Community.Display.V1.FormattedValue

## Ordering results

GET https://[Organization URI]/api/data/v9.0/contacts?$select=fullname,jobtitle,annualincome&$filter=contains(fullname,'(sample)')%20&$orderby=jobtitle%20asc,%20annualincome%20desc HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=10, odata.include-annotations=OData.Community.Display.V1.FormattedValue

## Parameter alias

GET https://[Organization URI]/api/data/v9.0/contacts?$select=fullname,jobtitle,annualincome&$filter=contains(@p1,'(sample)')%20&$orderby=@p2%20asc,%20@p3%20desc&@p1=fullname&@p2=jobtitle&@p3=annualincome HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=10, odata.include-annotations=OData.Community.Display.V1.FormattedValue

## Limit results

Returning more data than you need is bad for performance. The server will return a maximum of 5000 entities per request. You can limit the number of results returned using the $top query option or by adding odata.maxpagesize in the request header. The $top query option only returns the top number of entities from the result set and ignores the rest. The odata.maxpagesize request header specifies the number of entities return per page with an @odata.nextLink property to get results of the next page. For more information about odata.maxpagesize, see the section on [Pagination](https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/webapi/web-api-query-data-sample#bkmk_filterPagination) and see also [Limits on number of entities returned](https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/webapi/query-data-web-api#bkmk_limits).

### Top results

GET https://[Organization URI]/api/data/v9.0/contacts?$select=fullname,jobtitle,annualincome&$filter=contains(fullname,'(sample)')&$top=5 HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=10, odata.include-annotations=OData.Community.Display.V1.FormattedValue

### Result count

GET https://[Organization URI]/api/data/v9.0/contacts?$select=fullname,jobtitle,annualincome&$filter=contains(jobtitle,'senior')%20or%20contains(jobtitle,%20'manager')&$count=true HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=10, odata.include-annotations=OData.Community.Display.V1.FormattedValue

### Pagination

GET https://[Organization URI]/api/data/v9.0/contacts?$select=fullname,jobtitle,annualincome&$filter=contains(fullname,'(sample)')&$count=true HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=4, odata.include-annotations=OData.Community.Display.V1.FormattedValue

To retrieve page 2, use a GET request with the value of the @odata.nextLink property.

GET https://[Organization URI]/api/data/v9.0/contacts?$select=fullname,jobtitle,annualincome&$filter=contains(fullname,'(sample)')&$count=true&$skiptoken=%3Ccookie%20pagenumber=%222%22%20pagingcookie=%22%253ccookie%2520page%253d%25221%2522%253e%253ccontactid%2520last%253d%2522%257bC748FDEE-C143-E611-80D5-00155DA84802%257d%2522%2520first%253d%2522%257bB848FDEE-C143-E611-80D5-00155DA84802%257d%2522%2520%252f%253e%253c%252fcookie%253e%22%20istracking=%22False%22%20/%3E HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=4, odata.include-annotations=OData.Community.Display.V1.FormattedValue

## Expanding results

A Single-valued navigation property represents a many-to-one relationships. In our sample data, the account has a relationship with a contact via the primarycontactid attribute. In this relationship, the account can only have one primary contact. Using the [account EntityType](https://docs.microsoft.com/en-us/dynamics365/customer-engagement/web-api/account), we can create a query to get information about the account and expanded information about its primary contact.

GET https://[Organization URI]/api/data/v9.0/accounts(b2546951-c543-e611-80d5-00155da84802)?$select=name&$expand=primarycontactid($select=fullname,jobtitle,annualincome) HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=10, odata.include-annotations=OData.Community.Display.V1.FormattedValue

### Expand on partner property

Each navigation property has a corresponding “partner” property. Once an association is made, we can retrieve information through this association. Which attribute we use depends on the base entity that the query is against. For example, in the previous operation, we created a query against the [account EntityType](https://docs.microsoft.com/en-us/dynamics365/customer-engagement/web-api/account) and we wanted to get additional information about its primary contact. We did that via the primarycontactid attribute. If we look up the [account EntityType](https://docs.microsoft.com/en-us/dynamics365/customer-engagement/web-api/account), under the [Single-valued navigation properties](https://docs.microsoft.com/en-us/dynamics365/customer-engagement/web-api/account?view=dynamics-ce-odata-9#Single-valued_navigation_properties) section, we can see that the partner property that corresponds to primarycontactid is account\_primary\_contact collection-valued navigation property found on the [contact EntityType](https://docs.microsoft.com/en-us/dynamics365/customer-engagement/web-api/contact).

Writing a query against a contact, you can expand on the account\_primary\_contact attribute to get information about accounts where this contact is the primary contact. In the sample data, Yvonne McKay (sample) is the primary contact person for only one account. However, she can potentially be assigned to other accounts as primary contact. Because the account\_primary\_contact property has a many-to-one relationship the result is returned as an array of account entities.

GET https://[Organization URI]/api/data/v9.0/contacts(b3546951-c543-e611-80d5-00155da84802)?$select=fullname,jobtitle,annualincome&$expand=account\_primary\_contact($select=name) HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=10, odata.include-annotations=OData.Community.Display.V1.FormattedValue

### Expand on collection-valued navigation property

Collection-valued navigation properties support one-to-many or many-to-many relationships. For example, in our sample data, the account has a relationship with many contacts via the contact\_customer\_accounts attribute.

Using the [account EntityType](https://docs.microsoft.com/en-us/dynamics365/customer-engagement/web-api/account), we can create a query to get information about the account and expand information about its contacts. In this case, the Contoso, Ltd. (sample) is associated to nine other contacts via the contact\_customer\_accounts collection-valued navigation property.

GET https://[Organization URI]/api/data/v9.0/accounts(86546951-c543-e611-80d5-00155da84802)?$select=name&$expand=contact\_customer\_accounts($select=fullname,jobtitle,annualincome) HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=10, odata.include-annotations=OData.Community.Display.V1.FormattedValue

### Expand on multiple navigation properties

You can expand on as many navigation properties as the query requires. However, the $expand option can only go one level deep.

This example expands the primarycontactid, contact\_customer\_accounts, and Account\_Tasks navigation properties of the [account EntityType](https://docs.microsoft.com/en-us/dynamics365/customer-engagement/web-api/account). This query returns a response containing information about the account and two collections: a contacts collection and a tasks collection. The sample code will process these collections separately.

GET https://[Organization URI]/api/data/v9.0/accounts(86546951-c543-e611-80d5-00155da84802)?$select=name&$expand=primarycontactid($select=fullname,jobtitle,annualincome),contact\_customer\_accounts($select=fullname,jobtitle,annualincome),Account\_Tasks($select=subject,description) HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=10, odata.include-annotations=OData.Community.Display.V1.FormattedValue

## FetchXML queries

GET https://[Organization URI]/api/data/v9.0/contacts?fetchXml=%253Cfetch%2520mapping%253D%2522logical%2522%2520output-format%253D%2522xml-platform%2522%2520version%253D%25221.0%2522%2520distinct%253D%2522false%2522%253E%2520%2520%2520%253Centity%2520name%253D%2522contact%2522%253E%2520%2520%2520%2520%2520%253Cattribute%2520name%253D%2522fullname%2522%2520%252F%253E%2520%2520%2520%2520%2520%253Cattribute%2520name%253D%2522jobtitle%2522%2520%252F%253E%2520%2520%2520%2520%2520%253Cattribute%2520name%253D%2522annualincome%2522%2520%252F%253E%2520%2520%2520%2520%2520%253Corder%2520descending%253D%2522true%2522%2520attribute%253D%2522fullname%2522%2520%252F%253E%2520%2520%2520%2520%2520%253Cfilter%2520type%253D%2522and%2522%253E%2520%2520%2520%2520%2520%2520%2520%253Ccondition%2520value%253D%2522%2525(sample)%2525%2522%2520attribute%253D%2522fullname%2522%2520operator%253D%2522like%2522%2520%252F%253E%2520%2520%2520%2520%2520%253C%252Ffilter%253E%2520%2520%2520%253C%252Fentity%253E%2520%253C%252Ffetch%253E%2520 HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=10, odata.include-annotations=OData.Community.Display.V1.FormattedValue

### FetchXML pagination

<fetch mapping="logical"

output-format="xml-platform"

version="1.0"

distinct="false"

page="3"

count="4">

<entity name="contact">

<attribute name="fullname" />

<attribute name="jobtitle" />

<attribute name="annualincome" />

<order descending="true"

attribute="fullname" />

<filter type="and">

<condition value="%(sample)%"

attribute="fullname"

operator="like" />

</filter>

</entity>

</fetch>

GET https://[Organization URI]/api/data/v9.0/contacts?fetchXml=%253Cfetch%2520mapping%253D%2522logical%2522%2520output-format%253D%2522xml-platform%2522%2520version%253D%25221.0%2522%2520distinct%253D%2522false%2522%2520page%253D%25223%2522%2520count%253D%25224%2522%253E%2520%2520%2520%253Centity%2520name%253D%2522contact%2522%253E%2520%2520%2520%2520%2520%253Cattribute%2520name%253D%2522fullname%2522%2520%252F%253E%2520%2520%2520%2520%2520%253Cattribute%2520name%253D%2522jobtitle%2522%2520%252F%253E%2520%2520%2520%2520%2520%253Cattribute%2520name%253D%2522annualincome%2522%2520%252F%253E%2520%2520%2520%2520%2520%253Corder%2520descending%253D%2522true%2522%2520attribute%253D%2522fullname%2522%2520%252F%253E%2520%2520%2520%2520%2520%253Cfilter%2520type%253D%2522and%2522%253E%2520%2520%2520%2520%2520%2520%2520%253Ccondition%2520value%253D%2522%2525(sample)%2525%2522%2520attribute%253D%2522fullname%2522%2520operator%253D%2522like%2522%2520%252F%253E%2520%2520%2520%2520%2520%253C%252Ffilter%253E%2520%2520%2520%253C%252Fentity%253E%2520%253C%252Ffetch%253E%2520 HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=10, odata.include-annotations=OData.Community.Display.V1.FormattedValue

## Predefined queries

### Saved query

In this operation, we will make a request for the savedqueryid GUID of the saved query named **Active Accounts**. Then using the GUID and the savedQuery parameter, we will query for a list of active accounts.

GET https://[Organization URI]/api/data/v9.0/savedqueries?$select=name,savedqueryid&$filter=name%20eq%20'Active%20Accounts' HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=10, odata.include-annotations=OData.Community.Display.V1.FormattedValue

Referer: <https://localhost:1469/WebAPIQuery.html>

Getting the saved query's content using the savedQuery parameter

GET https://[Organization URI]/api/data/v9.0/accounts?savedQuery=00000000-0000-0000-00aa-000010001002 HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Prefer: odata.maxpagesize=10, odata.include-annotations=OData.Community.Display.V1.FormattedValue

### User query

This sample creates a user query, executes it, then deletes it from the system. This user query is asking for any contacts whose fullname contains (sample), jobtitle contains manager, and annualincome greater than 55000. Our sample data has two contacts matching this query.

GET https://[Organization URI]/api/data/v9.0/userqueries?$select=name,userqueryid,&$filter=name%20eq%20'My%20User%20Query' HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Referer: <https://localhost:1469/WebAPIQuery.html>

## Conditional GET

1. Attempt to retrieve the account Contoso Ltd. only if it does *not* match the current version, identified by the initial ETag value that was returned when the account record was created. This condition is represented by the If-None-Match header.

GET https://[Organization URI]/api/data/v9.0/accounts(14e151db-9b4f-e611-80e0-00155da84c08)?$select=name,revenue,telephone1,description HTTP/1.1

If-None-Match: W/"628448"

OData-MaxVersion: 4.0

OData-Version: 4.0

Accept: application/json

1. Update the account by modifying its primary telephone number property.

PUT https://[Organization URI]/api/data/v9.0/accounts(14e151db-9b4f-e611-80e0-00155da84c08)/telephone1 HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Accept: application/json

Content-Type: application/json

{

"value": "555-0001"

}

1. Re-attempt the same conditional GET operation, again using the original ETag value. This time the operation returns the requested data because the version on the server is different (and newer) than the version identified in the request. As in all record retrievals, the response includes an ETag header that identifies the current version.

GET https://[Organization URI]/api/data/v9.0/accounts(14e151db-9b4f-e611-80e0-00155da84c08)?$select=name,revenue,telephone1,description HTTP/1.1

If-None-Match: W/"628448"

OData-MaxVersion: 4.0

OData-Version: 4.0

Accept: application/json

## Optimistic concurrency on delete and update

This section of the program demonstrates how to perform conditional delete and update operations. The most common use for such operations is in implementing an optimistic concurrency approach to record processing in a multi-user environment. More information:[Apply optimistic concurrency](https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/webapi/perform-conditional-operations-using-web-api#bkmk_Applyoptimisticconcurrency)

1. Attempt to delete original account if and only if it matches the original version (ETag value). This condition is represented by the If-Match header. This operation fails because the account record was updated in the previous section, so as a result, its version was updated on the server.

**Request**

HTTPCopy

DELETE https://[Organization URI]/api/data/v9.0/accounts(14e151db-9b4f-e611-80e0-00155da84c08) HTTP/1.1

If-Match: W/"628448"

OData-MaxVersion: 4.0

OData-Version: 4.0

Accept: application/json

**Response**

HTTPCopy

HTTP/1.1 412 Precondition Failed

Content-Type: application/json; odata.metadata=minimal

OData-Version: 4.0

{

"error":{

"code":"","message":"The version of the existing record doesn't match the RowVersion property provided.", . . .

}

}

**Console output**

Copy

Expected Error: The version of the existing record doesn't match the property provided.

Account not deleted using ETag 'W/"628448"', status code: '412'.

1. Attempt to update the account if and only if it matches the original ETag value. Again, this condition is represented by the If-Match header and the operation fails for the same reason.

**Request**

HTTPCopy

PATCH https://[Organization URI]/api/data/v9.0/accounts(14e151db-9b4f-e611-80e0-00155da84c08) HTTP/1.1

If-Match: W/"628448"

OData-MaxVersion: 4.0

OData-Version: 4.0

Accept: application/json

Content-Type: application/json; charset=utf-8

{

"telephone1": "555-0002",

"revenue": 6000000

}

**Response**

HTTPCopy

HTTP/1.1 412 Precondition Failed

Content-Type: application/json; odata.metadata=minimal

OData-Version: 4.0

{

"error":{

"code":"","message":"The version of the existing record doesn't match the RowVersion property provided.", . . .

}

}

**Console output**

Copy

Expected Error: The version of the existing record doesn't match the property provided.

Account not updated using ETag 'W/"628448"', status code: '412'.

1. Re-attempt an update, but instead use the current ETag value obtained from the last record retrieval in the previous section.

**Request**

HTTPCopy

PATCH https://[Organization URI]/api/data/v9.0/accounts(14e151db-9b4f-e611-80e0-00155da84c08) HTTP/1.1

If-Match: W/"628460"

OData-MaxVersion: 4.0

OData-Version: 4.0

Accept: application/json

{

"telephone1": "555-0003",

"revenue": 6000000

}

**Response**

HTTPCopy

HTTP/1.1 204 No Content

**Console output**

Copy

Account successfully updated using ETag: W/"628460", status code: '204'.

1. Confirm the update succeeded by retrieving and outputting the current account state. This uses a basic GET request.

**Request**

HTTPCopy

GET https://[Organization URI]/api/data/v9.0/accounts(14e151db-9b4f-e611-80e0-00155da84c08)?$select=name,revenue,telephone1,description HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Accept: application/json

**Response**

HTTPCopy

HTTP/1.1 200 OK

Content-Type: application/json; odata.metadata=minimal

ETag: W/"628461"

OData-Version: 4.0

{

"@odata.context":"https://[Organization URI]/api/data/v9.0/$metadata#accounts(name,revenue,telephone1,description)/$entity",

"@odata.etag":"W/\"628461\"",

"name":"Contoso Ltd",

"revenue":6000000.0000,

"telephone1":"555-0003",

"description":"Parent company of Contoso Pharmaceuticals, etc.",

"accountid":"14e151db-9b4f-e611-80e0-00155da84c08",

"\_transactioncurrencyid\_value":"0d4ed62e-95f7-e511-80d1-00155da84c03"

}

**Console output**

Copy

{

"@odata.context": "https://[Organization URI]/api/data/v9.0/$metadata#accounts(name,revenue,telephone1,description)/$entity",

"@odata.etag": "W/\"628461\"",

"name": "Contoso Ltd",

"revenue": 6000000.0,

"telephone1": "555-0003",

"description": "Parent company of Contoso Pharmaceuticals, etc.",

"accountid": "14e151db-9b4f-e611-80e0-00155da84c08",

"\_transactioncurrencyid\_value": "0d4ed62e-95f7-e511-80d1-00155da84c03"

}

## Controlling upsert operations

This section of the program demonstrates how to perform conditional PATCH operations, limiting upsert operations to perform as either update-only or insert-only operations. More information:[Limit upsert operations](https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/webapi/perform-conditional-operations-using-web-api#bkmk_limitUpsertOperations)

1. Attempt to insert, without updating, the primary telephone and revenue properties for this account. The If-None-Match header with the value of \* represents this upsert condition. This operation fails because this account record still exists on the server (unless it was concurrently deleted by another user or process).

**Request**

HTTPCopy

PATCH https://[Organization URI]/api/data/v9.0/accounts(14e151db-9b4f-e611-80e0-00155da84c08) HTTP/1.1

If-None-Match: \*

OData-MaxVersion: 4.0

OData-Version: 4.0

Accept: application/json

Content-Type: application/json; charset=utf-8

{

"telephone1": "555-0004",

"revenue": 7500000

}

**Response**

HTTPCopy

HTTP/1.1 412 Precondition Failed

Content-Type: application/json; odata.metadata=minimal

OData-Version: 4.0

{

"error":{

"code":"","message":"A record with matching key values already exists.", . . .

}

}

**Console output**

Copy

Expected Error: A record with matching key values already exists.

Account not updated using ETag 'W/"628448", status code: '412'.

1. Attempt to perform the same update operation without creation. To accomplish this, the conditional If-Match header is used with a value of \*. This operation succeeds because the record exists on the server.

**Request**

HTTPCopy

PATCH https://[Organization URI]/api/data/v9.0/accounts(14e151db-9b4f-e611-80e0-00155da84c08) HTTP/1.1

If-Match: \*

OData-MaxVersion: 4.0

OData-Version: 4.0

Accept: application/json

Content-Type: application/json; charset=utf-8

{

"telephone1": "555-0005",

"revenue": 7500000

}

**Response**

HTTPCopy

HTTP/1.1 204 No Content

**Console output**

Copy

Account updated using If-Match '\*'

1. Retrieve and output the current account state with a basic GET request. Note that the returned ETag value has changed to reflect the new, updated version of the account record.

**Request**

HTTPCopy

GET https://[Organization URI]/api/data/v9.0/accounts(14e151db-9b4f-e611-80e0-00155da84c08)?$select=name,revenue,telephone1,description HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Accept: application/json

**Response**

HTTPCopy

HTTP/1.1 200 OK

Content-Type: application/json; odata.metadata=minimal

ETag: W/"628463"

OData-Version: 4.0

{

"@odata.context":"https://[Organization URI]/api/data/v9.0/$metadata#accounts(name,revenue,telephone1,description)/$entity",

"@odata.etag":"W/\"628463\"",

"name":"Contoso Ltd","revenue":7500000.0000,

"telephone1":"555-0005",

"description":"Parent company of Contoso Pharmaceuticals, etc.",

"accountid":"14e151db-9b4f-e611-80e0-00155da84c08",

"\_transactioncurrencyid\_value":"0d4ed62e-95f7-e511-80d1-00155da84c03"

}

**Console output**

HTTPCopy

{

"@odata.context": "https://[Organization URI]/api/data/v9.0/$metadata#accounts(name,revenue,telephone1,description)/$entity",

"@odata.etag": "W/\"628463\"",

"name": "Contoso Ltd",

"revenue": 7500000.0,

"telephone1": "555-0005",

"description": "Parent company of Contoso Pharmaceuticals, etc.",

"accountid": "14e151db-9b4f-e611-80e0-00155da84c08",

"\_transactioncurrencyid\_value": "0d4ed62e-95f7-e511-80d1-00155da84c03"

}

1. Delete the account with a basic DELETE.

**Request**

HTTPCopy

DELETE https://[Organization URI]/api/data/v9.0/accounts(14e151db-9b4f-e611-80e0-00155da84c08) HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Accept: application/json

**Response**

HTTPCopy

HTTP/1.1 204 No Content

**Console output**

Copy

Account was deleted.

1. Just as in step 2, attempt to update the account if it exists. Again, this condition is represented by the If-Match header with a value of \*. This operation fails because this record was just deleted. However, if this If-Match header was absent, then the resulting basic upsert operation should successfully create a new record.

**Request**

HTTPCopy

PATCH https://[Organization URI]/api/data/v9.0/accounts(14e151db-9b4f-e611-80e0-00155da84c08) HTTP/1.1

If-Match: \*

OData-MaxVersion: 4.0

OData-Version: 4.0

Accept: application/json

Content-Type: application/json; charset=utf-8

{

"telephone1": "555-0006",

"revenue": 7500000

}

**Response**

HTTPCopy

HTTP/1.1 404 Not Found

Content-Type: application/json; odata.metadata=minimal

OData-Version: 4.0

{

"error":{

"code":"","message":"account With Id = 14e151db-9b4f-e611-80e0-00155da84c08 Does Not Exist", . . .

}

}

**Console output**

Copy

Expected Error: Account with Id = 14e151db-9b4f-e611-80e0-00155da84c08 does not exist.

Account not updated because it does not exist, status code: '404'.

There is no need to cleanup sample data because the one account record was already deleted in step 4.

### Using bound action with parameters

Use a bound action that takes parameters. This operation adds a letter to the current user's queue. To accomplish this, we use the [WhoAmI Function](https://docs.microsoft.com/en-us/dynamics365/customer-engagement/web-api/whoami) and the [systemuser EntityType](https://docs.microsoft.com/en-us/dynamics365/customer-engagement/web-api/systemuser) to get a reference to the current user's queue. We also need reference to the [letter EntityType](https://docs.microsoft.com/en-us/dynamics365/customer-engagement/web-api/letter). This letter was created as sample data earlier in the program. Then the bound [AddToQueue Action](https://docs.microsoft.com/en-us/dynamics365/customer-engagement/web-api/addtoqueue) is called to add the letter to the current user's queue. More information:[Bound actions](https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/webapi/use-web-api-actions#bkmk_boundActions)

**Request**

HTTPCopy

POST https://[Organization URI]/api/data/v9.0/queues(1f7bcc50-d3f6-e511-80d0-00155da84802)/Microsoft.Dynamics.CRM.AddToQueue HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Content-Length: 110

{

"Target":{

"activityid":"4c920da5-fb4a-e611-80d5-00155da84802",

"@odata.type":"Microsoft.Dynamics.CRM.letter"

}

}

**Response**

HTTPCopy

HTTP/1.1 200 OK

Content-Type: application/json; odata.metadata=minimal

OData-Version: 4.0

Content-Length: 170

{

"@odata.context":"https://[Organization URI]/api/data/v9.0/$metadata#Microsoft.Dynamics.CRM.AddToQueueResponse",

"QueueItemId":"67bdfabd-fc4a-e611-80d5-00155da84802"

}

**Console output**

Copy

Bound Action: AddToQueue

QueueItemId returned from AddToQueue Action: 67bdfabd-fc4a-e611-80d5-00155da84802

## Working with custom actions

If you define custom actions for your solution, you can call them using the Dataverse Web API. Regardless of whether the operations included in your custom action have side effects, they can potentially modify data and therefore are considered actions rather than functions. There is no way to create a custom function. More information:[Use a custom action](https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/webapi/use-web-api-actions#bkmk_customActions).

This sample comes with two custom actions. They both require parameters but one is bound and the other is unbound.

* sample\_AddNoteToContact: A bound custom action that takes two parameters. One is a NoteTitle and the other is a NoteText. This custom action adds a note to a [contact EntityType](https://docs.microsoft.com/en-us/dynamics365/customer-engagement/web-api/contact). Below is a screen shot of the **Information** page for this custom action.
* sample\_CreateCustomer: An unbound custom action that require different parameters depending on what type of customer is being created. For example, when the AccountType is "account" then it only requires AccountName parameter. When the AccountType is "contact", a ContactFirstName and ContactLastName parameters are required. Below is a screen shot of the **Information** page for this custom action.

### Using bound custom action with parameters

This example calls the sample\_AddNoteToContact custom action which is bound to the contact entity with the required parameters. This custom action adds a note to an existing contact. This action returns an entity with an annotationid property. To show that the note was added, the annotationid is used to request information about the note.

The request and response of the action.

**Request**

HTTPCopy

POST https://[Organization URI]/api/data/v9.0/contacts(4d920da5-fb4a-e611-80d5-00155da84802)/Microsoft.Dynamics.CRM.sample\_AddNoteToContact HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Content-Length: 80

{

"NoteTitle":"The Title of the Note",

"NoteText":"The text content of the note."

}

**Response**

HTTPCopy

HTTP/1.1 200 OK

Content-Type: application/json; odata.metadata=minimal

OData-Version: 4.0

Content-Length: 149

{

"@odata.context":"https://[Organization URI]/api/data/v9.0/$metadata#annotations/$entity",

"annotationid":"ba146d0b-fd4a-e611-80d5-00155da84802"

}

The request and response of the annotation.

**Request**

HTTPCopy

GET https://[Organization URI]/api/data/v9.0/annotations(ba146d0b-fd4a-e611-80d5-00155da84802)?$select=subject,notetext&$expand=objectid\_contact($select=fullname) HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

**Response**

HTTPCopy

HTTP/1.1 200 OK

OData-Version: 4.0

Content-Length: 450

{

"@odata.context":"https://[Organization URI]/api/data/v9.0/$metadata#annotations(subject,notetext,objectid\_contact,objectid\_contact(fullname))/$entity",

"@odata.etag":"W/\"622978\"",

"subject":"The Title of the Note",

"notetext":"The text content of the note.",

"annotationid":"ba146d0b-fd4a-e611-80d5-00155da84802",

"objectid\_contact":{

"@odata.etag":"W/\"622968\"",

"fullname":"Jon Fogg",

"contactid":"4d920da5-fb4a-e611-80d5-00155da84802"

}

}

**Console output**

HTTPCopy

Custom action: sample\_AddNoteToContact

A note with the title 'The Title of the Note' and the content 'The text content of the note.' was created and associated with the contact Jon Fogg.

### Using unbound custom action with parameters

This operation calls the sample\_CreateCustomer custom action to create an "account" customer. Required parameters are passed in for a CustomerType of account.

**Request**

HTTPCopy

POST https://[Organization URI]/api/data/v9.0/sample\_CreateCustomer HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Content-Length: 103

{

"CustomerType":"account",

"AccountName":"Account Customer Created in WebAPIFunctionsAndActions sample"

}

**Response**

HTTPCopy

HTTP/1.1 204 No Content

OData-Version: 4.0

### Handling custom action exceptions

This example shows that custom actions can return custom error messages. You handle custom exceptions the same way you handle standard exceptions. To get the custom error message from the sample\_CreateCustomer custom action , this example creates a "contact" customer. However, we intentionally pass in the wrong parameters for this CustomerType parameter. This operation then catches the exception and displays the error message, then continues with the sample program.

**Request**

HTTPCopy

POST https://[Organization URI]/api/data/v9.0/sample\_CreateCustomer HTTP/1.1

OData-MaxVersion: 4.0

OData-Version: 4.0

Content-Type: application/json; charset=utf-8

Content-Length: 103

{

"CustomerType":"contact",

"AccountName":"Account Customer Created in WebAPIFunctionsAndActions sample"

}

**Response**

HTTPCopy

HTTP/1.1 500 Internal Server Error

Content-Type: application/json; odata.metadata=minimal

OData-Version: 4.0

Content-Length: 2760

{

"error":{

"code":"",

"message":"ContactFirstName and ContactLastName are required when CustomerType is contact."

}

}

**Console output**

Copy

Expected custom error: ContactFirstName and ContactLastName are required when CustomerType is contact.

<html>

<head>

<title>Microsoft CRM Web API Basic Operations Example</title>

<meta charset="utf-8" />

<meta http-equiv="X-UA-Compatible" content="IE=Edge" />

<script src="../ClientGlobalContext.js.aspx" type="text/javascript"></script>

<script src="scripts/es6promise.js" type="text/javascript"></script>

<script src="scripts/WebAPIBasicOperations.js" type="text/javascript"></script>

<style type="text/css">

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

}

#preferences {

border: inset;

padding: 10px 10px;

}

#output\_area {

border: inset;

background-color: gainsboro;

padding: 10px 10px;

}

</style>

</head>

<body>

<h1>Microsoft CRM Web API Basic Operations Example</h1>

<p>This page demonstrates the CRM Web API's basic operations using JavaScript.</p>

<h2>Instructions</h2>

<p>

Choose your preferences and run the JavaScript code.

Use your browser's developer tools to view the output written to the console (e.g.: in IE 11 or Microsoft Edge,

press F12 to load the Developer Tools).

</p>

<p>

Remove sample data (Choose whether you want to delete sample data created during this execution):

<br />

<input name="removesampledata" type="radio" value="yes" checked />

Yes

<input name="removesampledata" type="radio" value="no" />

No

</p>

<input type="button" name="start\_sample" value="Start Sample" onclick="Sdk.startSample()" />

</body>

</html>

### WebAPIBasicOperations.js

"use strict";

var Sdk = window.Sdk || {};

/\*\*

\* @function getClientUrl

\* @description Get the client URL.

\* @returns {string} The client URL.

\*/

Sdk.getClientUrl = function () {

var context;

// GetGlobalContext defined by including reference to

// ClientGlobalContext.js.aspx in the HTML page.

if (typeof GetGlobalContext != "undefined") {

context = GetGlobalContext();

} else {

if (typeof Xrm != "undefined") {

// Xrm.Page.context defined within the Xrm.Page object model for form scripts.

context = Xrm.Page.context;

} else {

throw new Error("Context is not available.");

}

}

return context.getClientUrl();

};

/\*\*

\* An object instantiated to manage detecting the

\* Web API version in conjunction with the

\* Sdk.retrieveVersion function

\*/

Sdk.versionManager = new function () {

//Start with base version

var \_webAPIMajorVersion = 8;

var \_webAPIMinorVersion = 0;

//Use properties to increment version and provide WebAPIPath string used by Sdk.request;

Object.defineProperties(this, {

"WebAPIMajorVersion": {

get: function () {

return \_webAPIMajorVersion;

},

set: function (value) {

if (typeof value != "number") {

throw new Error("Sdk.versionManager.WebAPIMajorVersion property must be a number.")

}

\_webAPIMajorVersion = parseInt(value, 10);

}

},

"WebAPIMinorVersion": {

get: function () {

return \_webAPIMinorVersion;

},

set: function (value) {

if (isNaN(value)) {

throw new Error("Sdk.versionManager.\_webAPIMinorVersion property must be a number.")

}

\_webAPIMinorVersion = parseInt(value, 10);

}

},

"WebAPIPath": {

get: function () {

return "/api/data/v" + \_webAPIMajorVersion + "." + \_webAPIMinorVersion;

}

}

})

}

//Setting variables specific to this sample within a container so they won't be

// overwritten by another scripts code

Sdk.SampleVariables = {

entitiesToDelete: [], // Entity URIs to be deleted later (if user so chooses)

deleteData: true, // Controls whether sample data are deleted at the end of sample run

contact1Uri: null, // e.g.: Peter Cambel

contactAltUri: null, // e.g.: Peter\_Alt Cambel

account1Uri: null, // e.g.: Contoso, Ltd

account2Uri: null, // e.g.: Fourth Coffee

contact2Uri: null, // e.g.: Susie Curtis

opportunity1Uri: null, // e.g.: Adventure Works

competitor1Uri: null

}

/\*\*

\* @function request

\* @description Generic helper function to handle basic XMLHttpRequest calls.

\* @param {string} action - The request action. String is case-sensitive.

\* @param {string} uri - An absolute or relative URI. Relative URI starts with a "/".

\* @param {object} data - An object representing an entity. Required for create and update actions.

\* @param {object} addHeader - An object with header and value properties to add to the request

\* @returns {Promise} - A Promise that returns either the request object or an error object.

\*/

Sdk.request = function (action, uri, data, addHeader) {

if (!RegExp(action, "g").test("POST PATCH PUT GET DELETE")) { // Expected action verbs.

throw new Error("Sdk.request: action parameter must be one of the following: " +

"POST, PATCH, PUT, GET, or DELETE.");

}

if (!typeof uri === "string") {

throw new Error("Sdk.request: uri parameter must be a string.");

}

if ((RegExp(action, "g").test("POST PATCH PUT")) && (!data)) {

throw new Error("Sdk.request: data parameter must not be null for operations that create or modify data.");

}

if (addHeader) {

if (typeof addHeader.header != "string" || typeof addHeader.value != "string") {

throw new Error("Sdk.request: addHeader parameter must have header and value properties that are strings.");

}

}

// Construct a fully qualified URI if a relative URI is passed in.

if (uri.charAt(0) === "/") {

//This sample will try to use the latest version of the web API as detected by the

// Sdk.retrieveVersion function.

uri = Sdk.getClientUrl() + Sdk.versionManager.WebAPIPath + uri;

}

return new Promise(function (resolve, reject) {

var request = new XMLHttpRequest();

request.open(action, encodeURI(uri), true);

request.setRequestHeader("OData-MaxVersion", "4.0");

request.setRequestHeader("OData-Version", "4.0");

request.setRequestHeader("Accept", "application/json");

request.setRequestHeader("Content-Type", "application/json; charset=utf-8");

if (addHeader) {

request.setRequestHeader(addHeader.header, addHeader.value);

}

request.onreadystatechange = function () {

if (this.readyState === 4) {

request.onreadystatechange = null;

switch (this.status) {

case 200: // Operation success with content returned in response body.

case 201: // Create success.

case 204: // Operation success with no content returned in response body.

resolve(this);

break;

default: // All other statuses are unexpected so are treated like errors.

var error;

try {

error = JSON.parse(request.response).error;

} catch (e) {

error = new Error("Unexpected Error");

}

reject(error);

break;

}

}

};

request.send(JSON.stringify(data));

});

};

/\*\*

\* @function startSample

\* @description Runs the sample.

\* This sample demonstrates basic CRUD+ operations.

\* Results are sent to the debugger's console window.

\*/

Sdk.startSample = function () {

// Initializing.

Sdk.SampleVariables.deleteData = document.getElementsByName("removesampledata")[0].checked;

Sdk.SampleVariables.entitiesToDelete = []; // Reset the array.

Sdk.SampleVariables.contact1Uri = "";

Sdk.SampleVariables.account1Uri = "";

Sdk.SampleVariables.account2Uri = "";

Sdk.SampleVariables.contact2Uri = "";

Sdk.SampleVariables.opportunity1Uri = "";

Sdk.SampleVariables.competitor1Uri = "";

/\*\*

\* Behavior of this sample varies by version

\* So starting by retrieving the version;

\*/

Sdk.retrieveVersion()

.then(function () {

return Sdk.basicCreateAndUpdatesAsync()

})

.then(function () {

return Sdk.createWithAssociationAsync()

})

.then(function () {

return Sdk.createRelatedAsync()

})

.then(function () {

return Sdk.associateExistingAsync()

})

.then(function () {

return Sdk.deleteSampleData()

})

.catch(function (err) {

console.log("ERROR: " + err.message);

});

}

Sdk.retrieveVersion = function () {

return new Promise(function (resolve, reject) {

Sdk.request("GET", "/RetrieveVersion")

.then(function (request) {

try {

var RetrieveVersionResponse = JSON.parse(request.response);

var fullVersion = RetrieveVersionResponse.Version;

var versionData = fullVersion.split(".");

Sdk.versionManager.WebAPIMajorVersion = parseInt(versionData[0], 10);

Sdk.versionManager.WebAPIMinorVersion = parseInt(versionData[1], 10);

resolve();

} catch (err) {

reject(new Error("Error processing version: " + err.message))

}

})

.catch(function (err) {

reject(new Error("Error retrieving version: " + err.message))

})

});

};

Sdk.basicCreateAndUpdatesAsync = function () {

return new Promise(function (resolve, reject) {

// Section 1.

//

// Create the contact using POST request.

// A new entry will be added regardless if a contact with this info already exists in the system or not.

console.log("--Section 1 started--");

var contact = {};

contact.firstname = "Peter";

contact.lastname = "Cambel";

var entitySetName = "/contacts";

Sdk.request("POST", entitySetName, contact)

.then(function (request) {

// Process response from previous request.

Sdk.SampleVariables.contact1Uri = request.getResponseHeader("OData-EntityId");

Sdk.SampleVariables.entitiesToDelete.push(Sdk.SampleVariables.contact1Uri); // To delete later

console.log("Contact 'Peter Cambel' created with URI: %s", Sdk.SampleVariables.contact1Uri);

// Setup for next request.

//

// Update contact.

// Add property values to a specific contact using PATCH request.

var contact = {};

contact.annualincome = 80000.00;

contact.jobtitle = "Junior Developer";

return Sdk.request("PATCH", Sdk.SampleVariables.contact1Uri, contact)

})

.then(function () {

// Process response from previous request.

console.log("Contact 'Peter Cambel' updated with job title and annual income.");

// Setup for next request.

//

// Retrieve selected properties of a Contact entity using GET request.

// NOTE: It is performance best practice to select only the properties you need.

// Retrieved contact properties.

var properties = [

"fullname",

"annualincome",

"jobtitle",

"description"].join();

// NOTE: For performance best practices, use $select to limit the properties you want to return

// See also: https://msdn.microsoft.com/library/gg334767.aspx#bkmk\_requestProperties

var query = "?$select=" + properties;

return Sdk.request("GET", Sdk.SampleVariables.contact1Uri + query, null);

})

.then(function (request) {

// Process response from previous request.

var contact1 = JSON.parse(request.response);

var successMsg = "Contact '%s' retrieved:\n"

+ "\tAnnual income: %s \n"

+ "\tJob title: %s \n"

+ "\tDescription: %s";

console.log(successMsg,

contact1.fullname, // This property is read-only. Calculated from firstname and lastname.

contact1.annualincome,

contact1.jobtitle,

contact1.description); // Description will be "null" because it has not been set yet.

// Setup for next request.

//

// Update properties.

// Set new values for some of the properties and apply the values to the server via PATCH request.

// Notice that we are updating the jobtitle and annualincome properties and adding value to the

// description property in the same request.

var contact = {};

contact.jobtitle = "Senior Developer";

contact.annualincome = 95000.00;

contact.description = "Assignment to-be-determined. ";

return Sdk.request("PATCH", Sdk.SampleVariables.contact1Uri, contact);

})

.then(function () {

// Process response from previous request.

console.log("Contact 'Peter Cambel' updated:\n"

+ "\tJob title: Senior Developer, \n"

+ "\tAnnual income: 95000, \n"

+ "\tDescription: Assignment to-be-determined.");

// Setup for next request.

//

// Set value for a single property using PUT request.

// In this case, we are setting the telephone1 property to "555-0105".

var value = { value: "555-0105" };

return Sdk.request("PUT", Sdk.SampleVariables.contact1Uri + "/telephone1", value);

})

.then(function () {

// Process response from previous request.

console.log("Contact 'Peter Cambel' phone number updated.");

// Setup for next request.

//

// Retrieve single value property.

// Get a value of a single property using GET request.

// In this case, telephone1 is retrieved. We should get back "555-0105".

return Sdk.request("GET", Sdk.SampleVariables.contact1Uri + "/telephone1", null);

})

.then(function (request) {

// Process response from previous request.

var phoneNumber = JSON.parse(request.response);

console.log("Contact's phone number is: %s", phoneNumber.value);

})

.then(function () {

// Setup for next request.

//The following operations require version 8.2 or higher

if (Sdk.versionManager.WebAPIMajorVersion > 8 ||

(Sdk.versionManager.WebAPIMajorVersion == 8 && Sdk.versionManager.WebAPIMinorVersion >= 2)

) {

// Starting with December 2016 update (v8.2), a contact instance can be

// created and its properties returned in one operation by using a

//'Prefer: return=representation' header.

var contactAlt = {};

contactAlt.firstname = "Peter\_Alt";

contactAlt.lastname = "Cambel";

contactAlt.jobtitle = "Junior Developer";

contactAlt.annualincome = 80000;

contactAlt.telephone1 = "555-0110";

var properties = [

"fullname",

"annualincome",

"jobtitle"].join();

var query = "?$select=" + properties;

// Create contact and return its state (in the body).

var retRepHeader = { header: "Prefer", value: "return=representation" };

Sdk.request("POST", entitySetName + query, contactAlt, retRepHeader)

.then(function (request) {

var contactA = JSON.parse(request.response);

//Because 'OData-EntityId' header not returned in a 201 response, you must instead

// construct the URI.

Sdk.SampleVariables.contactAltUri = Sdk.getClientUrl() +

Sdk.versionManager.WebAPIPath +

"/contacts(" +

contactA.contactid +

")";

Sdk.SampleVariables.entitiesToDelete.push(Sdk.SampleVariables.contactAltUri);

var successMsg = "Contact '%s' created:\n"

+ "\tAnnual income: %s \n"

+ "\tJob title: %s \n";

console.log(successMsg,

contactA.fullname,

contactA.annualincome,

contactA.jobtitle);

console.log("Contact URI: %s", Sdk.SampleVariables.contactAltUri);

})

.then(function () {

// Setup for next request.

//Similarly, the December 2016 update (v8.2) also enables returning selected properties

//after an update operation (PATCH), with the 'Prefer: return=representation' header.

var contactAlt = {};

contactAlt.jobtitle = "Senior Developer";

contactAlt.annualincome = 95000;

contactAlt.description = "MS Azure and Dataverse Specialist";

var properties = [

"fullname",

"annualincome",

"jobtitle",

"description"].join();

var query = "?$select=" + properties;

// Update contact and return its state (in the body).

var retRepHeader = { header: "Prefer", value: "return=representation" };

return Sdk.request("PATCH", Sdk.SampleVariables.contactAltUri + query, contactAlt, retRepHeader);

})

.then(function (request) {

// Process response from previous request.

var contactA = JSON.parse(request.response);

var successMsg = "Contact '%s' updated:\n"

+ "\tAnnual income: %s \n"

+ "\tJob title: %s \n";

console.log(successMsg,

contactA.fullname,

contactA.annualincome,

contactA.jobtitle);

//End this series of operations:

resolve();

})

.catch(function (err) {

reject(err);

});

}

else {

resolve();

}

})

.catch(function (err) {

reject(err);

});

});

};

Sdk.createWithAssociationAsync = function () {

return new Promise(function (resolve, reject) {

// Section 2.

//

// Create a new account entity and associate it with an existing contact using POST request.

console.log("\n--Section 2 started--");

var account = {};

account.name = "Contoso, Ltd.";

account.telephone1 = "555-5555";

account["primarycontactid@odata.bind"] = Sdk.SampleVariables.contact1Uri; //relative URI ok. E.g.: "/contacts(###)".

var entitySetName = "/accounts";

Sdk.request("POST", entitySetName, account)

.then(function (request) {

// Process response from previous request.

Sdk.SampleVariables.account1Uri = request.getResponseHeader("OData-EntityId");

Sdk.SampleVariables.entitiesToDelete.push(Sdk.SampleVariables.account1Uri);

console.log("Account 'Contoso, Ltd.' created.");

// Setup for next request.

//

// Retrieve account's primary contact with selected properties using GET request and 'expand' query.

var contactProperties = [

"fullname",

"jobtitle",

"annualincome"

].join();

var query = "?$select=name,telephone1&$expand=primarycontactid($select=" + contactProperties + ")";

return Sdk.request("GET", Sdk.SampleVariables.account1Uri + query, null);

})

.then(function (request) {

// Process response from previous request.

var account1 = JSON.parse(request.response);

var successMsg = "Account '%s' has primary contact '%s': \n"

+ "\tJob title: %s \n"

+ "\tAnnual income: %s ";

console.log(successMsg,

account1.name,

account1.primarycontactid.fullname,

account1.primarycontactid.jobtitle,

account1.primarycontactid.annualincome);

//End this series of operations:

resolve();

})

.catch(function (err) {

reject(err);

});

});

};

Sdk.createRelatedAsync = function () {

return new Promise(function (resolve, reject) {

// Section 3.

//

// Create related entities (deep insert).

// Create the following entities in one operation using deep insert technique:

// account

// |--- contact

// |--- tasks

// Then retrieve properties of these entities

//

// Constructing the entity relationship.

console.log("\n--Section 3 started--");

var account = {};

account.name = "Fourth Coffee";

account.primarycontactid = {

firstname: "Susie",

lastname: "Curtis",

jobtitle: "Coffee Master",

annualincome: 48000.00,

Contact\_Tasks: [

{

subject: "Sign invoice",

description: "Invoice #12321",

scheduledend: new Date("April 19th, 2016")

},

{

subject: "Setup new display",

description: "Theme is - Spring is in the air",

scheduledstart: new Date("4/20/2016")

},

{

subject: "Conduct training",

description: "Train team on making our new blended coffee",

scheduledstart: new Date("6/1/2016")

}

]

};

var entitySetName = "/accounts";

Sdk.request("POST", entitySetName, account)

.then(function (request) {

// Process response from previous request.

Sdk.SampleVariables.account2Uri = request.getResponseHeader("OData-EntityId");

Sdk.SampleVariables.entitiesToDelete.push(Sdk.SampleVariables.account2Uri);

console.log("Account 'Fourth Coffee' created.");

// Setup for next request.

//

// Retrieve account entity info using GET request and 'expand' query.

var contactProperties = [

"fullname",

"jobtitle",

"annualincome"].join();

// Expand on primarycontactid to select some of contact's properties.

// NOTE: With $expand, the CRM server will return values for the selected properties.

// The CRM Web API only supports expansions one level deep.

// See also: https://msdn.microsoft.com/library/mt607871.aspx#bkmk\_expandRelated

var query = "?$select=name&$expand=primarycontactid($select=" + contactProperties + ")";

return Sdk.request("GET", Sdk.SampleVariables.account2Uri + query, null);

})

.then(function (request) {

// Process response from previous request.

var account2 = JSON.parse(request.response);

var successMsg = "Account '%s' has primary contact '%s':\n"

+ "\tJob title: %s \n"

+ "\tAnnual income: %s";

console.log(successMsg,

account2.name,

account2.primarycontactid.fullname,

account2.primarycontactid.jobtitle,

account2.primarycontactid.annualincome);

// Setup for next request.

//

// Retrieve contact entity and expanding on its tasks using GET request.

Sdk.SampleVariables.contact2Uri = Sdk.getClientUrl() + Sdk.versionManager.WebAPIPath + "/contacts(" + account2.primarycontactid.contactid + ")"; //Full URI.

Sdk.SampleVariables.entitiesToDelete.push(Sdk.SampleVariables.contact2Uri); // For Susie Curtis

var contactProperties = ["fullname", "jobtitle"].join();

var contactTaskProperties = ["subject", "description", "scheduledstart", "scheduledend"].join();

// Expand on contact\_tasks to select some of its properties for each task.

var query = "?$select=" + contactProperties +

"&$expand=Contact\_Tasks($select=" + contactTaskProperties + ")";

return Sdk.request("GET", Sdk.SampleVariables.contact2Uri + query, null);

})

.then(function (request) {

// Process response from previous request.

var contact2 = JSON.parse(request.response);

console.log("Contact '%s' has the following assigned tasks:", contact2.fullname);

// construct the output string.

var successMsg = "Subject: %s \n"

+ "\tDescription: %s \n"

+ "\tStart: %s \n"

+ "\tEnd: %s \n";

for (var i = 0; i < contact2.Contact\_Tasks.length; i++) {

console.log(successMsg,

contact2.Contact\_Tasks[i].subject,

contact2.Contact\_Tasks[i].description,

contact2.Contact\_Tasks[i].scheduledstart,

contact2.Contact\_Tasks[i].scheduledend

);

}

//End this series of operations:

resolve();

})

.catch(function (err) {

reject(err);

});

});

};

Sdk.associateExistingAsync = function () {

return new Promise(function (resolve, reject) {

// Section 4

//

// Entity associations:

// Associate to existing entities via the different relationship types:

// 1) 1:N relationship - Associate an existing contact to an existing account

// (e.g.: contact - Peter Cambel to account - Fourth Coffee).

// 2) N:N relationship - Associate an competitor to opportunity.

console.log("\n--Section 4 started--");

var contact = {};

contact["@odata.id"] = Sdk.SampleVariables.contact1Uri;

Sdk.request("POST", Sdk.SampleVariables.account2Uri + "/contact\_customer\_accounts/$ref", contact)

.then(function () {

// Process response from previous request.

console.log("Contact 'Peter Cambel' associated to account 'Fourth Coffee'.");

// Setup for next request.

//

// Verify that the reference was made as expected.

var contactProperties = ["fullname", "jobtitle"].join();

// This returns a collection of all associated contacts...in a "value" array.

var query = "/contact\_customer\_accounts?$select=" + contactProperties;

return Sdk.request("GET", Sdk.SampleVariables.account2Uri + query, null);

})

.then(function (request) {

// Process response from previous request.

var relatedContacts = JSON.parse(request.response).value; //collection is in the "value" array.

var successMsg = "\tName: %s, "

+ "Job title: %s ";

console.log("Contact list for account 'Fourth Coffee': ");

for (var i = 0; i < relatedContacts.length; i++) {

console.log(successMsg,

relatedContacts[i].fullname,

relatedContacts[i].jobtitle

);

}

// Setup for next request.

//

// Disassociate a contact from an account.

return Sdk.request("DELETE", Sdk.SampleVariables.account2Uri + "/contact\_customer\_accounts/$ref?$id=" + Sdk.SampleVariables.contact1Uri, null);

})

.then(function () {

// Process response from previous request.

console.log("Contact 'Peter Cambel' disassociated from account 'Fourth Coffee'.");

// Setup for next request.

//

// N:N relationship:

// Associate a competitor to an opportunity.

var competitor = {};

competitor.name = "Adventure Works";

competitor.strengths = "Strong promoter of private tours for multi-day outdoor adventures.";

var entitySetName = "/competitors";

return Sdk.request("POST", entitySetName, competitor);

})

.then(function (request) {

// Process response from previous request.

Sdk.SampleVariables.competitor1Uri = request.getResponseHeader("OData-EntityId");

Sdk.SampleVariables.entitiesToDelete.push(Sdk.SampleVariables.competitor1Uri);

console.log("Competitor 'Adventure Works' created.");

// Setup for next request.

//

// Create a new opportunity...

var opportunity = {};

opportunity.name = "River rafting adventure";

opportunity.description = "Sales team on a river-rafting offsite and team building.";

var entitySetName = "/opportunities";

return Sdk.request("POST", entitySetName, opportunity);

})

.then(function (request) {

// Process response from previous request.

Sdk.SampleVariables.opportunity1Uri = request.getResponseHeader("OData-EntityId");

Sdk.SampleVariables.entitiesToDelete.push(Sdk.SampleVariables.opportunity1Uri);

console.log("Opportunity 'River rafting adventure' created.");

// Setup for next request.

//

// Associate competitor to opportunity.

var competitor = {};

competitor["@odata.id"] = Sdk.SampleVariables.competitor1Uri;

return Sdk.request("POST", Sdk.SampleVariables.opportunity1Uri + "/opportunitycompetitors\_association/$ref", competitor);

})

.then(function () {

// Process response from previous request.

console.log("Opportunity 'River rafting adventure' associated with competitor 'Adventure Works'.");

// Setup for next request.

//

// Retrieve competitor entity and expanding on its opportunitycompetitors\_association

// for all opportunities, using GET request.

var opportunityProperties = ["name", "description"].join();

var competitorProperties = ["name"].join();

var query = "?$select=" + competitorProperties +

"&$expand=opportunitycompetitors\_association($select=" + opportunityProperties + ")";

return Sdk.request("GET", Sdk.SampleVariables.competitor1Uri + query, null);

})

.then(function (request) {

// Process response from previous request.

var competitor1 = JSON.parse(request.response);

console.log("Competitor '%s' has the following opportunities:", competitor1.name);

var successMsg = "\tName: %s, \n"

+ "\tDescription: %s";

for (var i = 0; i < competitor1.opportunitycompetitors\_association.length; i++) {

console.log(successMsg,

competitor1.opportunitycompetitors\_association[i].name,

competitor1.opportunitycompetitors\_association[i].description

);

}

// Setup for next request.

//

// Disassociate competitor from opportunity.

return Sdk.request("DELETE", Sdk.SampleVariables.opportunity1Uri +

"/opportunitycompetitors\_association/$ref?$id=" + Sdk.SampleVariables.competitor1Uri, null);

})

.then(function () {

// Process response from previous request.

console.log("Opportunity 'River rafting adventure' disassociated with competitor 'Adventure Works'");

//End this series of operations:

resolve();

})

.catch(function (err) {

reject(err);

});

});

};

Sdk.deleteSampleData = function () {

return new Promise(function (resolve, reject) {

// House cleaning - deleting sample data

// NOTE: If instances have a parent-child relationship, then deleting the parent will,

// by default, automatically cascade delete child instances. In this program,

// tasks related using the Contact\_Tasks relationship have contact as their parent.

// Other relationships may behave differently.

// See also: https://msdn.microsoft.com/library/gg309412.aspx#BKMK\_CascadingBehavior

console.log("\n--Section 5 started--");

if (Sdk.SampleVariables.deleteData) {

for (var i = 0; i < Sdk.SampleVariables.entitiesToDelete.length; i++) {

console.log("Deleting entity: " + Sdk.SampleVariables.entitiesToDelete[i]);

Sdk.request("DELETE", Sdk.SampleVariables.entitiesToDelete[i], null)

.catch(function (err) {

reject(new Error("ERROR: Delete failed --Reason: \n\t" + err.message))

});

}

resolve();

} else {

console.log("Sample data not deleted.");

resolve();

}

});

};

<!DOCTYPE html>

<html>

<head>

<title>Microsoft CRM Web API Query Example</title>

<meta charset="utf-8" />

<script src="../ClientGlobalContext.js.aspx" type="text/javascript"></script>

<script src="scripts/es6promise.js"></script>

<script src="scripts/WebAPIQuery.js"></script>

<style type="text/css">

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

}

#preferences {

border: inset;

padding: 10px 10px;

}

#output\_area {

border: inset;

background-color: gainsboro;

padding: 10px 10px;

}

</style>

</head>

<body>

<h1>Microsoft CRM Web API Query Example</h1>

<p>This page demonstrates the CRM Web API's Query operations using JavaScript.</p>

<h2>Instructions</h2>

<p>Choose your preferences and run the JavaScript code.

Use your browser's developer tools to view the output written to the console (e.g.: in IE 11 or Microsoft Edge,

press F12 to load the Developer Tools).</p>

<form id="preferences">

<p>

Remove sample data (Choose whether you want to delete sample data created during this execution):

<br />

<input name="removesampledata" type="radio" value="yes" checked /> Yes

<input name="removesampledata" type="radio" value="no" /> No

</p>

<input type="button" name="start\_samples" value="Start Sample" onclick="Sdk.startSample()" />

</form>

</body>

</html>

# Query Data

"use strict";

var Sdk = window.Sdk || {};

/\*\*

\* @function getClientUrl

\* @description Get the client URL.

\* @returns {string} The client URL.

\*/

Sdk.getClientUrl = function () {

var context;

// GetGlobalContext defined by including reference to

// ClientGlobalContext.js.aspx in the HTML page.

if (typeof GetGlobalContext != "undefined")

{ context = GetGlobalContext(); }

else

{

if (typeof Xrm != "undefined") {

// Xrm.Page.context defined within the Xrm.Page object model for form scripts.

context = Xrm.Page.context;

}

else { throw new Error("Context is not available."); }

}

return context.getClientUrl();

}

// Global variables.

var entitiesToDelete = []; // Entity URIs to be deleted (if user chooses to delete sample data)

var deleteData = true; // Delete data by default unless user chooses not to delete.

var clientUrl = Sdk.getClientUrl(); // e.g.: https://org.crm.dynamics.com

var webAPIPath = "/api/data/v8.1"; // Path to the web API.

var account1Uri; // e.g.: Contoso Inc (sample)

var contact1Uri; // e.g.: Yvonne McKey (sample)

var page2Uri; // URI of next page in pagination sample.

// Entity properties to select in a request.

var contactProperties = ["fullname", "jobtitle", "annualincome"];

var accountProperties = ["name"];

var taskProperties = ["subject", "description"];

/\*\*

\* @function request

\* @description Generic helper function to handle basic XMLHttpRequest calls.

\* @param {string} action - The request action. String is case-sensitive.

\* @param {string} uri - An absolute or relative URI. Relative URI starts with a "/".

\* @param {object} data - An object representing an entity. Required for create and update action.

\* @param {boolean} formattedValue - If "true" then include formatted value; "false" otherwise.

\* For more info on formatted value, see:

\* https://msdn.microsoft.com/library/gg334767.aspx#bkmk\_includeFormattedValues

\* @param {number} maxPageSize - Indicate the page size. Default is 10 if not defined.

\* @returns {Promise} - A Promise that returns either the request object or an error object.

\*/

Sdk.request = function (action, uri, data, formattedValue, maxPageSize) {

if (!RegExp(action, "g").test("POST PATCH PUT GET DELETE")) { // Expected action verbs.

throw new Error("Sdk.request: action parameter must be one of the following: " +

"POST, PATCH, PUT, GET, or DELETE.");

}

if (!typeof uri === "string") {

throw new Error("Sdk.request: uri parameter must be a string.");

}

if ((RegExp(action, "g").test("POST PATCH PUT")) && (data === null || data === undefined)) {

throw new Error("Sdk.request: data parameter must not be null for operations that create or modify data.");

}

if (maxPageSize === null || maxPageSize === undefined) {

maxPageSize = 10; // Default limit is 10 entities per page.

}

// Construct a fully qualified URI if a relative URI is passed in.

if (uri.charAt(0) === "/") {

uri = clientUrl + webAPIPath + uri;

}

return new Promise(function (resolve, reject) {

var request = new XMLHttpRequest();

request.open(action, encodeURI(uri), true);

request.setRequestHeader("OData-MaxVersion", "4.0");

request.setRequestHeader("OData-Version", "4.0");

request.setRequestHeader("Accept", "application/json");

request.setRequestHeader("Content-Type", "application/json; charset=utf-8");

request.setRequestHeader("Prefer", "odata.maxpagesize=" + maxPageSize);

if (formattedValue) {

request.setRequestHeader("Prefer",

"odata.include-annotations=OData.Community.Display.V1.FormattedValue");

}

request.onreadystatechange = function () {

if (this.readyState === 4) {

request.onreadystatechange = null;

switch (this.status) {

case 200: // Success with content returned in response body.

case 204: // Success with no content returned in response body.

resolve(this);

break;

default: // All other statuses are unexpected so are treated like errors.

var error;

try {

error = JSON.parse(request.response).error;

} catch (e) {

error = new Error("Unexpected Error");

}

reject(error);

break;

}

}

};

request.send(JSON.stringify(data));

});

};

/\*\*

\* @funnction output

\* @description Generic helper function to output data to console.

\* @param {array} collection - Array of entities.

\* @param {string} label - Text label for what the collection contains.

\* @param {array} properties - Array of properties appropriate for the collection.

\*/

Sdk.output = function (collection, label, properties) {

console.log(label);

collection.forEach(function (row, i) {

var prop = [];

properties.forEach(function (p) {

var f = p + "@OData.Community.Display.V1.FormattedValue";

prop.push((row[f] ? row[f] : row[p])); // Get formatted value if one exists for this property.

})

console.log("\t%s) %s", i + 1, prop.join(", "));

});

}

/\*\*

\* @function startSample

\* @description Runs the sample.

\* This sample demonstrates basic query operations.

\* Results are sent to the debugger's console window.

\*/

Sdk.startSample = function () {

// Initializing...

deleteData = document.getElementsByName("removesampledata")[0].checked;

entitiesToDelete = []; //Reset the array.

account1Uri = "";

contact1Uri = "";

page2Uri = "";

console.log("-- Sample started --");

console.log("Create sample data:");

// Add some data to the CRM server so we can query against it.

// Using Deep Insert, we create all the sample data in one request.

// Data structure:

// Accounts

// |--- primarycontactid

// |--- Contact\_Tasks (3 tasks)

// |--- Account\_Tasks (3 tasks)

// |--- contact\_customer\_accounts (9 child contacts, each with 3 tasks)

// |--- Contacts

// |--- Contact\_Tasks

//

var sampleData = {

"name": "Contoso, Ltd. (sample)",

"primarycontactid": {

"firstname": "Yvonne", "lastname": "McKay (sample)", "jobtitle": "Coffee Master",

"annualincome": 45000, "Contact\_Tasks": [

{ "subject": "Task 1", "description": "Task 1 description" },

{ "subject": "Task 2", "description": "Task 2 description" },

{ "subject": "Task 3", "description": "Task 3 description" }

]

}, "Account\_Tasks": [

{ "subject": "Task 1", "description": "Task 1 description" },

{ "subject": "Task 2", "description": "Task 2 description" },

{ "subject": "Task 3", "description": "Task 3 description" }

],

"contact\_customer\_accounts": [

{

"firstname": "Susanna", "lastname": "Stubberod (sample)", "jobtitle": "Senior Purchaser",

"annualincome": 52000, "Contact\_Tasks": [

{ "subject": "Task 1", "description": "Task 1 description" },

{ "subject": "Task 2", "description": "Task 2 description" },

{ "subject": "Task 3", "description": "Task 3 description" }

]

},

{

"firstname": "Nancy", "lastname": "Anderson (sample)", "jobtitle": "Activities Manager",

"annualincome": 55500, "Contact\_Tasks": [

{ "subject": "Task 1", "description": "Task 1 description" },

{ "subject": "Task 2", "description": "Task 2 description" },

{ "subject": "Task 3", "description": "Task 3 description" }

]

},

{

"firstname": "Maria", "lastname": "Cambell (sample)", "jobtitle": "Accounts Manager",

"annualincome": 31000, "Contact\_Tasks": [

{ "subject": "Task 1", "description": "Task 1 description" },

{ "subject": "Task 2", "description": "Task 2 description" },

{ "subject": "Task 3", "description": "Task 3 description" }

]

},

{

"firstname": "Nancy", "lastname": "Anderson (sample)", "jobtitle": "Logistics Specialist",

"annualincome": 63500, "Contact\_Tasks": [

{ "subject": "Task 1", "description": "Task 1 description" },

{ "subject": "Task 2", "description": "Task 2 description" },

{ "subject": "Task 3", "description": "Task 3 description" }

]

},

{

"firstname": "Scott", "lastname": "Konersmann (sample)", "jobtitle": "Accounts Manager",

"annualincome": 38000, "Contact\_Tasks": [

{ "subject": "Task 1", "description": "Task 1 description" },

{ "subject": "Task 2", "description": "Task 2 description" },

{ "subject": "Task 3", "description": "Task 3 description" }

]

},

{

"firstname": "Robert", "lastname": "Lyon (sample)", "jobtitle": "Senior Technician",

"annualincome": 78000, "Contact\_Tasks": [

{ "subject": "Task 1", "description": "Task 1 description" },

{ "subject": "Task 2", "description": "Task 2 description" },

{ "subject": "Task 3", "description": "Task 3 description" }

]

},

{

"firstname": "Paul", "lastname": "Cannon (sample)", "jobtitle": "Ski Instructor",

"annualincome": 68500, "Contact\_Tasks": [

{ "subject": "Task 1", "description": "Task 1 description" },

{ "subject": "Task 2", "description": "Task 2 description" },

{ "subject": "Task 3", "description": "Task 3 description" }

]

},

{

"firstname": "Rene", "lastname": "Valdes (sample)", "jobtitle": "Data Analyst III",

"annualincome": 86000, "Contact\_Tasks": [

{ "subject": "Task 1", "description": "Task 1 description" },

{ "subject": "Task 2", "description": "Task 2 description" },

{ "subject": "Task 3", "description": "Task 3 description" }

]

},

{

"firstname": "Jim", "lastname": "Glynn (sample)", "jobtitle": "Senior International Sales Manager",

"annualincome": 81400, "Contact\_Tasks": [

{ "subject": "Task 1", "description": "Task 1 description" },

{ "subject": "Task 2", "description": "Task 2 description" },

{ "subject": "Task 3", "description": "Task 3 description" }

]

}

]

};

var uri = "/accounts"; // A relative URI to the account entity.

Sdk.request("POST", uri, sampleData) // Adding sample data so we can query against it.

.then(function (request) {

// Process request.

account1Uri = request.getResponseHeader("OData-EntityId");

entitiesToDelete.push(account1Uri); // To delete later.

console.log("Account 'Contoso, Ltd. (sample)' created with 1 primary contact and 9 associated contacts.");

// Get primary contact info.

// Most queries are done using this contact.

var uri = account1Uri + "/primarycontactid/$ref"; // Request for the URI only.

return Sdk.request("GET", uri);

})

.then(function (request) {

contact1Uri = JSON.parse(request.response)["@odata.id"];

entitiesToDelete.push(contact1Uri); // To delete later.

console.log("Has primary contact 'Yvonne McKay (sample)' with URI: %s\n", contact1Uri);

// Basic query:

// Query using $select option against a contact entity to get the properties you want.

// For performance best practice, always use $select otherwise all properties are returned.

console.log("-- Basic Query --");

var query = "?$select=" + contactProperties.join(); // Array defined in the global scope.

return Sdk.request("GET", contact1Uri + query, null, true);

})

.then(function (request) {

var contact1 = JSON.parse(request.response);

console.log("Contact basic info:\n\tFullname: '%s'\n\tJobtitle: '%s'\n\tAnnualincome: '%s' (unformatted)",

contact1.fullname, contact1.jobtitle, contact1.annualincome);

console.log("\tAnnualincome: %s (formatted)\n",

contact1["annualincome@OData.Community.Display.V1.FormattedValue"]);

// Filter criteria:

// Applying filters to get targeted data.

// 1) Using standard query functions (e.g.: contains, endswith, startswith)

// 2) Using CRM query functions (e.g.: LastXhours, Last7Days, Today, Between, In, ...)

// 3) Using filter operators and logical operators (e.g.: eq, ne, gt, and, or, etc…)

// 4) Set precedence using parenthesis (e.g.: ((criteria1) and (criteria2)) or (criteria3)

// For more info, see: https://msdn.microsoft.com/library/gg334767.aspx#bkmk\_filter

console.log("-- Filter Criteria --");

// Filter 1: Using standard query functions to filter results.

// In this operation, we will query for all contacts with fullname containing the string "(sample)".

var filter = "&$filter=contains(fullname,'(sample)')";

var query = "?$select=" + contactProperties.join() + filter;

return Sdk.request("GET", "/contacts" + query, null, true);

})

.then(function (request) {

var collection = JSON.parse(request.response).value;

Sdk.output(collection, "Contacts filtered by fullname containing '(sample)':", contactProperties);

// Filter 2: Using CRM query functions to filter results.

// In this operation, we will query for all contacts that was created in the last hour.

// For complete list of CRM query functions, see:

// https://msdn.microsoft.com/library/mt607843.aspx

var filter = "&$filter=Microsoft.Dynamics.CRM.LastXHours(PropertyName='createdon',PropertyValue='1')";

var query = "?$select=" + contactProperties.join() + filter;

return Sdk.request("GET", "/contacts" + query, null, true); // Remember page size limit is set to 10.

})

.then(function(request){

var collection = JSON.parse(request.response).value;

Sdk.output(collection, "Contacts that were created within the last 1hr:", contactProperties);

// Filter 3: Using operators

// Building on the previous operation, we will further limit the results by the contact's income.

// For more info on standard filter operators, see:

// https://msdn.microsoft.com/library/gg334767.aspx#bkmk\_filter

var filter = "&$filter=contains(fullname,'(sample)') and annualincome gt 55000";

var query = "?$select=" + contactProperties.join() + filter;

return Sdk.request("GET", "/contacts" + query, null, true);

})

.then(function (request) {

var collection = JSON.parse(request.response).value;

Sdk.output(collection, "Contacts filtered by fullname and annualincome (<$55,000):", contactProperties);

// Filter 4: Set precedence using parenthesis.

// Continue building on the previous operation, we will further limit results by job title.

// Parenthesis and the order of filter statements can impact results returned.

var filter = "&$filter=contains(fullname,'(sample)') " +

"and (contains(jobtitle,'senior') or contains(jobtitle,'specialist')) and annualincome gt 55000";

var query = "?$select=" + contactProperties.join() + filter;

return Sdk.request("GET", "/contacts" + query, null, true);

})

.then(function (request) {

var collection = JSON.parse(request.response).value;

Sdk.output(collection, "Contacts filtered by fullname, annualincome and jobtitle (Senior or Specialist):",

contactProperties);

// Order results:

// Filtered results can be order in descending or ascending order.

console.log("\n-- Order Results --");

var filter = "&$filter=contains(fullname,'(sample)') " +

"&$orderby=jobtitle asc, annualincome desc";

var query = "?$select=" + contactProperties.join() + filter;

return Sdk.request("GET", "/contacts" + query, null, true);

})

.then(function (request) {

var collection = JSON.parse(request.response).value;

Sdk.output(collection, "Contacts ordered by jobtitle (Ascending) and annualincome (descending):",

contactProperties);

// Parameterized Aliases.

// Aliases can be used as parameters in a query. These parameters can be used in $filter and $orderby options.

// Using the previous operation as basis, parameterizing the query will give us the same results.

// For more info, see: https://msdn.microsoft.com/library/gg309638.aspx#bkmk\_passParametersToFunctions

console.log("\n-- Parameterized Aliases --");

var filter = "&$filter=contains(@p1,'(sample)') " +

"&$orderby=@p2 asc, @p3 desc&@p1=fullname&@p2=jobtitle&@p3=annualincome";

var query = "?$select=" + contactProperties.join() + filter;

return Sdk.request("GET", "/contacts" + query, null, true);

})

.then(function (request) {

var collection = JSON.parse(request.response).value;

Sdk.output(collection, "Contacts list using parameterized aliases:", contactProperties);

// Limit records returned.

// To further limit the records returned, use the $top query option.

// Specifying a limit number for $top will return at most that number of results per request.

// Extra results are ignored.

console.log("\n-- Top Results --");

var filter = "&$filter=contains(fullname,'(sample)')&$top=5";

var query = "?$select=" + contactProperties.join() + filter;

return Sdk.request("GET", "/contacts" + query, null, true);

})

.then(function (request) {

var collection = JSON.parse(request.response).value;

Sdk.output(collection, "Contacts top 5 results:", contactProperties);

// Result count.

// Count the number of results matching the filter criteria.

// 1) Get a count of a collection without the data.

// 2) Get a count along with the data.

// HINT: Use count together with the "odata.maxpagesize" to calculate the number of pages in the query.

// NOTE: CRM has a max record limit of 5000 records per response.

console.log("\n-- Result Count --");

return Sdk.request("GET", "/contacts/$count"); // Count is returned in response body.

})

.then(function (request) {

console.log("The contacts collection has %s contacts.", request.response); // Count maximum is 5000.

// 2) Get filtered result with a count

var filter = "&$filter=contains(jobtitle,'senior') or contains(jobtitle, 'manager')&$count=true";

var query = "?$select=" + contactProperties.join() + filter;

return Sdk.request("GET", "/contacts" + query, null, true);

})

.then(function (request) {

var count = JSON.parse(request.response)["@odata.count"];

console.log("%s contacts have either 'Manager' or 'Senior' designation in their jobtitle.", count);

var collection = JSON.parse(request.response).value;

Sdk.output(collection, "Manager or Senior:", contactProperties);

// Pagination:

// For large data sets, you can limit the number of records returned per page.

// Then offer a "next page" and "previous page" links for users to browse through all the data.

// NOTE: This is why you should not use $top with maxpagesize. $top will limit results returned

// preventing you from accessing all possible results in the query.

// For example: If your query has 10 entities in the result and you limit your result to $top=5

// then, you can't get to the remaining 5 results; but with "maxpagesize" (without $top), you can.

// HINT: Save the URI of the current page so users can go "next" and "previous".

console.log("\n-- Pagination --");

var filter = "&$filter=contains(fullname,'(sample)')&$count=true";

var query = "?$select=" + contactProperties.join() + filter;

return Sdk.request("GET", "/contacts" + query, null, true, 4); // 4 records per page.

})

.then(function (request) {

var count = JSON.parse(request.response)["@odata.count"];

var maxpages = Math.ceil(count / 4);

console.log("Contacts total: %s \tContacts per page: %s.\tOutputting first 2 pages.", count, 4);

var collection = JSON.parse(request.response).value;

Sdk.output(collection, "Page 1 of " + maxpages + ":", contactProperties);

// Getting the next page.

page2Uri = JSON.parse(request.response)["@odata.nextLink"]; // This URI is already encoded.

return Sdk.request("GET", decodeURI(page2Uri), null, true, 4); // URI re-encoded in the request function.

})

.then(function (request) {

var count = JSON.parse(request.response)["@odata.count"];

var maxpages = Math.ceil(count / 4);

var collection = JSON.parse(request.response).value;

Sdk.output(collection, "Page 2 of " + maxpages + ":", contactProperties);

// Using expand option to retrieve additional information.

// It is common for entities to have associations with other entities in the system and you might want

// to also retrieve this information in the same request. To retrieve information on associated entities,

// use the $expand query option on navigation properties.

// 1) Expand using single-valued navigation properties (e.g.: via the 'primarycontactid')

// 2) Expand using partner property (e.g.: from contact to account via the 'account\_primary\_contact')

// 3) Expand using collection-valued navigation properties (e.g.: via the 'contact\_customer\_accounts')

// 4) Expand using multiple navigation property types in a single request.

// NOTE: Expansions can only go 1 level deep.

// For performance best practice, always use $select statement in an expand option.

console.log("\n-- Expanding Results --");

// 1) Expand using single-valued navigation properties (e.g.: via the 'primarycontactid')

var expand = "&$expand=primarycontactid($select=" + contactProperties.join() + ")";

var query = "?$select=" + accountProperties.join() + expand;

return Sdk.request("GET", account1Uri + query, null, true);

})

.then(function (request) {

var account = JSON.parse(request.response);

var str = "Account '%s' has the following primary contact person:\n\t" +

"Fullname: '%s' \n\tJobtitle: '%s' \n\tAnnualincome: '%s'";

console.log(str, account.name,

account.primarycontactid.fullname,

account.primarycontactid.jobtitle,

account.primarycontactid.annualincome);

// 2) Expand using partner property (e.g.: from contact to account via the 'account\_primary\_contact')

var expand = "&$expand=account\_primary\_contact($select=" + accountProperties.join() + ")";

var query = "?$select=" + contactProperties.join() + expand;

return Sdk.request("GET", contact1Uri + query, null, true);

})

.then(function (request) {

var contact = JSON.parse(request.response);

var label = "Contact '" + contact.fullname + "' is the primary contact for the following accounts:";

Sdk.output(contact.account\_primary\_contact, label, accountProperties);

// 3) Expand using collection-valued navigation properties (e.g.: via the 'contact\_customer\_accounts')

var expand = "&$expand=contact\_customer\_accounts($select=" + contactProperties.join() + ")"

var query = "?$select=" + accountProperties.join() + expand;

return Sdk.request("GET", account1Uri + query, null, true);

})

.then(function (request) {

var account = JSON.parse(request.response);

var label = "Account '" + account.name + "' has the following contact customers:";

var collection = account.contact\_customer\_accounts;

Sdk.output(collection, label, contactProperties);

// 4) Expand using multiple navigation property types in a single request.

// For example: expanding on primiarycontactid, contact\_customer\_accounts, and Account\_Tasks.

console.log("\n-- Expanding multiple property types in one request -- ");

var expand = "&$expand=primarycontactid($select=" + contactProperties.join() + ")," +

"contact\_customer\_accounts($select=" + contactProperties.join() + ")," +

"Account\_Tasks($select=" + taskProperties.join() + ")";

var query = "?$select=" + accountProperties.join() + expand;

return Sdk.request("GET", account1Uri + query, null, true);

})

.then(function (request) {

var account = JSON.parse(request.response);

var label = "Account '%s' has the following primary contact person:\n\t" +

"Fullname: '%s' \n\tJobtitle: '%s' \n\tAnnualincome: '%s'";

console.log(label, account.name,

account.primarycontactid.fullname,

account.primarycontactid.jobtitle,

account.primarycontactid.annualincome);

// Handling each collection separately.

label = "Account '" + account.name + "' has the following related contacts:";

var collection = account.contact\_customer\_accounts;

Sdk.output(collection, label, contactProperties);

label = "Account '" + account.name + "' has the following tasks:";

collection = account.Account\_Tasks;

Sdk.output(collection, label, taskProperties);

// FetchXML

// Using FetchXML to query for all contacts whose fullname contains '(sample)'.

// NOTE: XML string must be URI encoded.

// For more information, see: https://msdn.microsoft.com/library/gg328117.aspx

console.log("\n-- FetchXML -- ");

var fetchXML = "<fetch mapping=\"logical\" output-format=\"xml-platform\" version=\"1.0\" distinct=\"false\"> \

<entity name=\"contact\"> \

<attribute name=\"fullname\" /> \

<attribute name=\"jobtitle\" /> \

<attribute name=\"annualincome\" /> \

<order descending=\"true\" attribute=\"fullname\" /> \

<filter type=\"and\"> \

<condition value=\"%(sample)%\" attribute=\"fullname\" operator=\"like\" /> \

</filter> \

</entity> \

</fetch> ";

return Sdk.request("GET", "/contacts?fetchXml=" + encodeURIComponent(fetchXML), null, true);

})

.then(function(request){

var collection = JSON.parse(request.response).value;

Sdk.output(collection, "Contacts Fetched by fullname containing '(sample)':", contactProperties);

// FetchXML pagination.

// Noticed the attribute "page=3" and "count=4" in this XML.

// We want to retrieve entities in page 3 but limit results to only 4 entities.

// If the result return zero records for the page, that means we have reached the end of the result set.

// For more info, see: https://msdn.microsoft.com/library/mt607533.aspx#bkmk\_useFetchXML

var fetchXML = "<fetch mapping=\"logical\" output-format=\"xml-platform\" version=\"1.0\" \

distinct=\"false\" page=\"3\" count=\"4\"> \

<entity name=\"contact\"> \

<attribute name=\"fullname\" /> \

<attribute name=\"jobtitle\" /> \

<attribute name=\"annualincome\" /> \

<order descending=\"true\" attribute=\"fullname\" /> \

<filter type=\"and\"> \

<condition value=\"%(sample)%\" attribute=\"fullname\" operator=\"like\" /> \

</filter> \

</entity> \

</fetch> ";

return Sdk.request("GET", "/contacts?fetchXml=" + encodeURIComponent(fetchXML), null, true);

})

.then(function(request){

var collection = JSON.parse(request.response).value;

if (collection.length == 0) {

console.log("There are no records on this page."); // We have reached the end of our query result set.

} else {

Sdk.output(collection, "Contacts Fetched by fullname containing '(sample)' - Page 3:", contactProperties);

}

// Using predefined queries.

// 1) Saved query

// 2) User query

// For more info, see:

// https://msdn.microsoft.com/library/mt607533.aspx

// Saved Query

// Get the Saved Query "Active Accounts" and display results to output.

console.log("\n-- Saved Query -- ");

var filter = "&$filter=name eq 'Active Accounts'";

var query = "?$select=name,savedqueryid" + filter;

return Sdk.request("GET", "/savedqueries" + query, null, true); // Requesting for saved query GUID.

})

.then(function(request){

// Get the savedqueryid GUID and then use it to request for the entities in that query.

var activeAccount = JSON.parse(request.response).value[0]; // Get the first matched.

var savedqueryid = activeAccount.savedqueryid;

// Request for the saved query results

return Sdk.request("GET", "/accounts?savedQuery=" + savedqueryid, null, true);

})

.then (function (request){

var collection = JSON.parse(request.response).value;

Sdk.output(collection, "Saved Query (Active Accounts):", accountProperties);

// User Query

// Create a user query then get it from the server and execute that query for results.

// For more info, see: https://msdn.microsoft.com/library/gg509053.aspx

console.log("\n-- User Query -- ");

var userquery = {

"name": "My User Query",

"description": "User query to display contact info.",

"querytype": 0,

"returnedtypecode": "contact",

"fetchxml": "<fetch mapping=\"logical\" output-format=\"xml-platform\" version=\"1.0\" distinct=\"false\"> \

<entity name=\"contact\"> \

<attribute name=\"fullname\" /> \

<attribute name=\"contactid\" /> \

<attribute name=\"jobtitle\" /> \

<attribute name=\"annualincome\" /> \

<order descending=\"false\" attribute=\"fullname\" /> \

<filter type=\"and\"> \

<condition value=\"%(sample)%\" attribute=\"fullname\" operator=\"like\" /> \

<condition value=\"%Manager%\" attribute=\"jobtitle\" operator=\"like\" /> \

<condition value=\"55000\" attribute=\"annualincome\" operator=\"gt\" /> \

</filter> \

</entity> \

</fetch> "

};

return Sdk.request("POST", "/userqueries", userquery, true); // Create the user query.

})

.then(function (request){

// Look up the user query we just created

// then use it to request for the entities in that query.

var filter = "&$filter=name eq 'My User Query'";

var query = "?$select=name,userqueryid," + filter;

return Sdk.request("GET", "/userqueries" + query, null, true);

})

.then(function (request) {

var userQuery = JSON.parse(request.response).value[0]; // Get the first matched.

var userqueryid = userQuery.userqueryid;

entitiesToDelete.push(clientUrl + webAPIPath + "/userqueries(" + userqueryid + ")");

// Request for the user query results

return Sdk.request("GET", "/contacts?userQuery=" + userqueryid, null, true);

})

.then(function (request) {

var collection = JSON.parse(request.response).value;

Sdk.output(collection, "Saved User Query:", contactProperties);

// House cleaning - deleting sample data

// For more info on cascading delete, see:

// https://msdn.microsoft.com/library/gg309412.aspx#BKMK\_CascadingBehavior

console.log("\n-- Deleting Sample Data --");

if (deleteData) {

for (var i = 0; i < entitiesToDelete.length; i++) {

console.log("Deleting entity: " + entitiesToDelete[i]);

Sdk.request("DELETE", entitiesToDelete[i], null)

.catch(function (err) {

console.log("ERROR: Delete failed --Reason: \n\t" + err.message);

});

}

} else {

console.log("Sample data not deleted.");

}

})

.catch(function (error) {

console.log(error.message);

});

}

# Conditional Operations

"use strict";

var Sdk = window.Sdk || {};

/\*\*

\* @function getClientUrl

\* @description Get the client URL.

\* @return {string} The client URL.

\*/

Sdk.getClientUrl = function () {

var context;

// GetGlobalContext defined by including reference to

// ClientGlobalContext.js.aspx in the HTML page.

if (typeof GetGlobalContext != "undefined")

{ context = GetGlobalContext(); }

else

{

if (typeof Xrm != "undefined") {

// Xrm.Page.context defined within the Xrm.Page object model for form scripts.

context = Xrm.Page.context;

}

else { throw new Error("Context is not available."); }

}

return context.getClientUrl();

}

// Global variables.

var clientUrl = Sdk.getClientUrl(); // e.g.: https://org.crm.dynamics.com

var webAPIPath = "/api/data/v8.1"; // Path to the web API.

var account1Uri; // e.g.: Contoso Ltd (sample)

var initialAcctETagVal; // The initial ETag value of the account created

var updatedAcctETagVal; // The ETag value of the account after it is updated

// Entity properties to select in a request.

var contactProperties = ["fullname", "jobtitle", "annualincome"];

var accountProperties = ["name"];

var taskProperties = ["subject", "description"];

/\*\*

\* @function request

\* @description Generic helper function to handle basic XMLHttpRequest calls.

\* @param {string} action - The request action. String is case-sensitive.

\* @param {string} uri - An absolute or relative URI. Relative URI starts with a "/".

\* @param {object} data - An object representing an entity. Required for create and update actions.

\* @param {object} addHeader - An object with header and value properties to add to the request

\* @returns {Promise} - A Promise that returns either the request object or an error object.

\*/

Sdk.request = function (action, uri, data, addHeader) {

if (!RegExp(action, "g").test("POST PATCH PUT GET DELETE")) { // Expected action verbs.

throw new Error("Sdk.request: action parameter must be one of the following: " +

"POST, PATCH, PUT, GET, or DELETE.");

}

if (!typeof uri === "string") {

throw new Error("Sdk.request: uri parameter must be a string.");

}

if ((RegExp(action, "g").test("POST PATCH PUT")) && (!data)) {

throw new Error("Sdk.request: data parameter must not be null for operations that create or modify data.");

}

if (addHeader) {

if (typeof addHeader.header != "string" || typeof addHeader.value != "string") {

throw new Error("Sdk.request: addHeader parameter must have header and value properties that are strings.");

}

}

// Construct a fully qualified URI if a relative URI is passed in.

if (uri.charAt(0) === "/") {

uri = clientUrl + webAPIPath + uri;

}

return new Promise(function (resolve, reject) {

var request = new XMLHttpRequest();

request.open(action, encodeURI(uri), true);

request.setRequestHeader("OData-MaxVersion", "4.0");

request.setRequestHeader("OData-Version", "4.0");

request.setRequestHeader("Accept", "application/json");

request.setRequestHeader("Content-Type", "application/json; charset=utf-8");

if (addHeader) {

request.setRequestHeader(addHeader.header, addHeader.value);

}

request.onreadystatechange = function () {

if (this.readyState === 4) {

request.onreadystatechange = null;

switch (this.status) {

case 200: // Success with content returned in response body.

case 204: // Success with no content returned in response body.

case 304: // Success with Not Modified.

resolve(this);

break;

default: // All other statuses are error cases.

var error;

try {

error = JSON.parse(request.response).error;

} catch (e) {

error = new Error("Unexpected Error");

}

reject(error);

break;

}

}

};

request.send(JSON.stringify(data));

});

};

/\*\*

\* @function startSample

\* @description Runs the sample.

\* This sample demonstrates conditional operations using CRM Web API.

\* Results are sent to the debugger's console window.

\*/

Sdk.startSample = function () {

// Initializing...

console.log("-- Sample started --");

// Create the CRM account instance.

var account = {

name: "Contoso, Ltd",

telephone1: "555-0000",// Phone number value will increment with each update attempt.

revenue: 5000000,

description: "Parent company of Contoso Pharmaceuticals, etc."

};

var uri = "/accounts"; // A relative URi to the account entity.

Sdk.request("POST", uri, account)

.then( function (request) {

console.log("Account entity created.");

// Assign the Uri to the created account to a global variable.

account1Uri = request.getResponseHeader("OData-EntityId");

// Retrieve the created account entity.

return Sdk.request("GET", account1Uri + "?$select=name,revenue,telephone1,description");

})

.then( function (request) {

// Show the current entity properties.

var account = JSON.parse(request.response);

console.log(JSON.stringify(account, null, 2));

initialAcctETagVal = account["@odata.etag"]; // Save the current ETag value.

// Conditional Get START.

// Attempt to retrieve using conditional GET with current ETag value.

// Expecting nothing in the response because entity was not modified.

console.log("-- Conditional GET section started --");

var ifNoneMatchETag = { header: "If-None-Match", value: initialAcctETagVal };

return Sdk.request("GET", account1Uri + "?$select=name,revenue,telephone1,description", null, ifNoneMatchETag);

})

.then( function (request) {

console.log("Instance retrieved using ETag: %s", initialAcctETagVal);

if (request.status == 304) {

//Expected:

console.log("\tEntity was not modified so nothing was returned.")

console.log(request.response); //Nothing

}

else {

//Not Expected:

console.log(JSON.stringify(JSON.parse(request.response), null, 2));

}

// Modify the account instance by updating telephone1.

// This request operation will also update the ETag value.

return Sdk.request("PUT", account1Uri + "/telephone1", { value: "555-0001" })

} )

.then( function (request) {

console.log("Account telephone number updated.");

// Re-attempt conditional GET with original ETag value.

var ifNoneMatchETag = { header: "If-None-Match", value: initialAcctETagVal };

return Sdk.request("GET", account1Uri + "?$select=name,revenue,telephone1,description", null, ifNoneMatchETag);

} )

.then( function (request) {

if (request.status == 200) {

// Expected.

console.log("Instance retrieved using ETag: %s", initialAcctETagVal);

var account = JSON.parse(request.response);

updatedAcctETagVal = account["@odata.etag"]; //Capture updated ETag.

console.log(JSON.stringify(account, null, 2));

}

else {

// Not Expected.

console.log("Unexpected status: %s", request.status)

}

// Conditional Get END.

// Optimistic concurrency on delete and update START.

console.log("-- Optimistic concurrency section started --");

// Attempt to delete original account (only if matches original ETag value).

var ifMatchETag = { header: "If-Match", value: initialAcctETagVal };

return Sdk.request("DELETE", account1Uri, null, ifMatchETag);

} )

.then( function (request) {

// Success not expected.

console.log("Unexpected status: %s", request.status)

},

// Catch error.

function (error) {

// DELETE: Precondition failed error expected.

console.log("Expected Error: %s", error.message);

console.log("\tAccount not deleted using ETag '%s', status code: '%s'.", initialAcctETagVal, 412)

// Attempt to update account (if matches original ETag value).

var accountUpdate = {

telephone1: "555-0002",

revenue: 6000000

};

var ifMatchETag = { header: "If-Match", value: initialAcctETagVal };

return Sdk.request("PATCH", account1Uri, accountUpdate, ifMatchETag);

})

.then( function (request) {

// Success not expected.

console.log("Unexpected status: %s", request.status);

},

// Catch error.

function (error) {

// UPDATE: Precondition failed error expected.

console.log("Expected Error: %s", error.message);

console.log("\tAccount not updated using ETag '%s', status code: '%s'.", initialAcctETagVal, 412)

// Re-attempt update if matches current ETag value.

var accountUpdate = {

telephone1: "555-0003",

revenue: 6000000

};

var ifMatchETag = { header: "If-Match", value: updatedAcctETagVal };

return Sdk.request("PATCH", account1Uri, accountUpdate, ifMatchETag);

} )

.then( function (request) {

if (request.status == 204) //No Content

{

// Expected.

console.log("Account successfully updated using ETag '%s', status code: '%s'.",

updatedAcctETagVal,

request.status)

}

else {

// Not Expected.

console.log("Unexpected status: %s", request.status)

}

// Retrieve and output current account state.

return Sdk.request("GET", account1Uri + "?$select=name,revenue,telephone1,description");

} )

.then( function (request) {

var account = JSON.parse(request.response);

updatedAcctETagVal = account["@odata.etag"]; // Capture updated ETag.

console.log(JSON.stringify(account, null, 2));

// Optimistic concurrency on delete and update END.

// Controlling upsert operations START.

console.log("-- Controlling upsert operations section started --");

// Attempt to insert (without update) some properties for this account.

var accountUpsert = {

telephone1: "555-0004",

revenue: 7500000

};

var ifNoneMatchResource = { header: "If-None-Match", value: "\*" };

return Sdk.request("PATCH", account1Uri, accountUpsert, ifNoneMatchResource);

} )

.then( function (request) {

// Success not expected.

console.log("Unexpected status: %s", request.status);

},

// Catch error.

function (error) {

// Precondition failed error expected.

console.log("Expected Error: %s", error.message);

console.log("\tAccount not updated using ETag '%s', status code: '%s'.", initialAcctETagVal, 412)

// Attempt to perform same update without creation.

var accountUpsert = {

telephone1: "555-0005",

revenue: 7500000

};

// Perform operation only if matching resource exists.

var ifMatchResource = { header: "If-Match", value: "\*" };

return Sdk.request("PATCH", account1Uri, accountUpsert, ifMatchResource);

} )

.then( function (request) {

if (request.status == 204) // No Content.

{

// Expected.

console.log("Account updated using If-Match '\*'")

}

else {

// Not Expected.

console.log("Unexpected status: %s", request.status)

}

// Retrieve and output current account state.

return Sdk.request("GET", account1Uri + "?$select=name,revenue,telephone1,description");

})

.then( function (request) {

var account = JSON.parse(request.response);

updatedAcctETagVal = account["@odata.etag"]; // Capture updated ETag.

console.log(JSON.stringify(account, null, 2));

// Controlling upsert operations END.

// Prevent update of deleted entity START.

// Delete the account.

return Sdk.request("DELETE", account1Uri);

}

)

.then(

function (request) {

if (request.status == 204) {

console.log("Account was deleted");

// Attempt to update it.

var accountUpsert = {

telephone1: "555-0005",

revenue: 7500000

};

// Perform operation only if matching resource exists.

var ifMatchResource = { header: "If-Match", value: "\*" };

return Sdk.request("PATCH", account1Uri, accountUpsert, ifMatchResource);

}

} )

.then( function (request) {

// Success not expected.

// Without the If-Match header while using PATCH a new entity would have been created with the

// same ID as the deleted entity.

console.log("Unexpected status: %s", request.status);

},

// Catch error.

function (error) {

// Not found error expected.

console.log("Expected Error: %s", error.message);

console.log("\tAccount not updated because it doesn't exist.");

}

)

.catch(function (error) {

console.log(error.message);

});

}

# Functions and Actions

"use strict";

var Sdk = window.Sdk || {};

/\*\*

\* @function getClientUrl

\* @description Get the client URL.

\* @return {string} The client URL.

\*/

Sdk.getClientUrl = function () {

var context;

// GetGlobalContext defined by including reference to

// ClientGlobalContext.js.aspx in the HTML page.

if (typeof GetGlobalContext != "undefined") {

context = GetGlobalContext();

} else {

if (typeof Xrm != "undefined") {

// Xrm.Page.context defined within the Xrm.Page object model for form scripts.

context = Xrm.Page.context;

} else {

throw new Error("Context is not available.");

}

}

return context.getClientUrl();

};

// Global variables

var entitiesToDelete = []; // Entity URIs to be deleted later

// (if user chooses to delete sample data).

var deleteData = true; // Controls whether sample data are deleted at the end of this sample run.

var clientUrl = Sdk.getClientUrl(); // ie.: https://org.crm.dynamics.com

var webAPIPath = "/api/data/v8.1"; // Path to the web API.

var incidentUri; // Incident created with three closed tasks.

var opportunityUri; // Closed opportunity to re-open before deleting.

var letterUri; // Letter to add to contact's queue.

var myQueueUri; // The contact's queue uri.

var contactUri; // Add a note to this contact.

var CUSTOMERACCOUNTNAME = "Account Customer Created in WebAPIFunctionsAndActions sample"; // For custom action.

/\*\*

\* @function getWebAPIPath

\* @description Get the full path to the Web API.

\* @return {string} The full URL of the Web API.

\*/

Sdk.getWebAPIPath = function () {

return Sdk.getClientUrl() + webAPIPath;

}

/\*\*

\* @function request

\* @description Generic helper function to handle basic XMLHttpRequest calls.

\* @param {string} action - The request action. String is case-sensitive.

\* @param {string} uri - An absolute or relative URI. Relative URI starts with a "/".

\* @param {object} data - An object representing an entity. Required for create and update actions.

\* @param {object} addHeader - An object with header and value properties to add to the request

\* @returns {Promise} - A Promise that returns either the request object or an error object.

\*/

Sdk.request = function (action, uri, data, addHeader) {

if (!RegExp(action, "g").test("POST PATCH PUT GET DELETE")) { // Expected action verbs.

throw new Error("Sdk.request: action parameter must be one of the following: " +

"POST, PATCH, PUT, GET, or DELETE.");

}

if (!typeof uri === "string") {

throw new Error("Sdk.request: uri parameter must be a string.");

}

if ((RegExp(action, "g").test("POST PATCH PUT")) && (!data)) {

throw new Error("Sdk.request: data parameter must not be null for operations that create or modify data.");

}

if (addHeader) {

if (typeof addHeader.header != "string" || typeof addHeader.value != "string") {

throw new Error("Sdk.request: addHeader parameter must have header and value properties that are strings.");

}

}

// Construct a fully qualified URI if a relative URI is passed in.

if (uri.charAt(0) === "/") {

uri = clientUrl + webAPIPath + uri;

}

return new Promise(function (resolve, reject) {

var request = new XMLHttpRequest();

request.open(action, encodeURI(uri), true);

request.setRequestHeader("OData-MaxVersion", "4.0");

request.setRequestHeader("OData-Version", "4.0");

request.setRequestHeader("Accept", "application/json");

request.setRequestHeader("Content-Type", "application/json; charset=utf-8");

if (addHeader) {

request.setRequestHeader(addHeader.header, addHeader.value);

}

request.onreadystatechange = function () {

if (this.readyState === 4) {

request.onreadystatechange = null;

switch (this.status) {

case 200: // Success with content returned in response body.

case 204: // Success with no content returned in response body.

case 304: // Success with Not Modified

resolve(this);

break;

default: // All other statuses are error cases.

var error;

try {

error = JSON.parse(request.response).error;

} catch (e) {

error = new Error("Unexpected Error");

}

reject(error);

break;

}

}

};

request.send(JSON.stringify(data));

});

};

/\*\*

\* @function Sdk.startSample

\* @description Initiates a chain of promises to show use of Functions and Actions with the Web API.

\* Functions and actions represent re-usable operations you can perform using the Web API.

\* For more info, see https://msdn.microsoft.com/library/mt607990.aspx#bkmk\_actions

\* The following standard CRM Web API functions and actions are invoked:

\* - WhoAmI, a basic unbound function

\* - GetTimeZoneCodeByLocalizedName, an unbound function that requires parameters

\* - CalculateTotalTimeIncident, a bound function

\* - WinOpportunity, an unbound action that takes parameters

\* - AddToQueue, a bound action that takes parameters

\* - In addition, a custom bound and an unbound action contained within the solution are invoked.

\*/

Sdk.startSample = function () {

// Initializing.

deleteData = document.getElementsByName("removesampledata")[0].checked;

entitiesToDelete = []; // Reset the array.

console.log("-- Sample started --");

// Create the CRM entry intances used by this sample program.

Sdk.createRequiredRecords()

.then(function () {

console.log("-- Working with functions --");

// Bound and Unbound functions

// See https://msdn.microsoft.com/library/gg309638.aspx#bkmk\_boundAndUnboundFunctions

console.log("Using functions to look up your full name.");

// Calling a basic unbound function without parameters.

// Retrieves the user's full name using a series of function requests.

// - Call WhoAmI via the Sdk.getUsersFullName function.

// For more info on the WhoAmI function, see https://msdn.microsoft.com/library/mt607925.aspx

return Sdk.getUsersFullName();

})

.then(function (fullName) {

console.log("\tYour full name is: %s\n", fullName);

console.log("Unbound function: GetTimeZoneCodeByLocalizedName");

// Calling a basic unbound function with no parameters.

// Retrieves the time zone code for the specified time zone.

// - Pass parameters to an unbound function by calling the GetTimeZoneCodeByLocalizedName Function.

// For more info, see https://msdn.microsoft.com/library/mt607644.aspx

var localizedStandardName = 'Pacific Standard Time';

var localeId = 1033;

// Demonstrates best practice of passing parameters.

var uri = ["/GetTimeZoneCodeByLocalizedName",

"(LocalizedStandardName=@p1,LocaleId=@p2)",

"?@p1='" + localizedStandardName + "'&@p2=" + localeId];

/\* This would also work:

var uri = ["/GetTimeZoneCodeByLocalizedName",

"(LocalizedStandardName='" + localizedStandardName + "',LocaleId=" + localeId + ")"];

\*/

return Sdk.request("GET", uri.join("")) // Send request.

})

.then(function (request) {

// Returns GetTimeZoneCodeByLocalizedNameResponse ComplexType.

// For more info, see https://msdn.microsoft.com/library/mt607889.aspx

var localizedStandardName = 'Pacific Standard Time';

var timeZoneCode = JSON.parse(request.response).TimeZoneCode;

console.log("\tFunction returned time zone %s, with code '%s'.", localizedStandardName, timeZoneCode);

console.log("Bound function: CalculateTotalTimeIncident");

// Calling a basic bound function that requires parameters.

// Retrieve the total time, in minutes, spent on all tasks associated with this incident.

// - Use CalculateTotalTimeIncident to get the total duration of all closed activities.

// For more info, see https://msdn.microsoft.com/library/mt593054.aspx

// Note that in a bound function the full function name includes the

// namespace Microsoft.Dynamics.CRM. Functions that aren’t bound must not use the full name.

return Sdk.request("GET", incidentUri + "/Microsoft.Dynamics.CRM.CalculateTotalTimeIncident()")

})

.then(function (request) {

// Returns CalculateTotalTimeIncidentResponse ComplexType.

// For more info, see https://msdn.microsoft.com/library/mt607924.aspx

var totalTime = JSON.parse(request.response).TotalTime; //returns 90

console.log("\tFunction returned %s minutes - total duration of tasks associated with the incident.\n",

totalTime);

console.log("-- Working with Actions --");

// For more info about Action, see https://msdn.microsoft.com/library/mt607600.aspx

console.log("Unbound Action: WinOpportunity");

// Calling an unbound action that requires parameters.

// Closes an opportunity and markt it as won.

// - Update the WinOpportunity (created by Sdk.createRequiredRecords()) by closing it as won.

// Use WinOpportunity Action (https://msdn.microsoft.com/library/mt607971.aspx)

// This action does not return a value

var parameters = {

"Status": 3,

"OpportunityClose": {

"subject": "Won Opportunity",

"opportunityid@odata.bind": opportunityUri

}

}

return Sdk.request("POST", "/WinOpportunity", parameters)

})

.then(function () {

console.log("\tOpportunity won.");

console.log("Bound Action: AddToQueue");

// Calling a bound action that requires parameters.

// Adds a new letter tracking activity to the current user's queue.

// The letter was created as part of the Sdk.createRequiredRecords().

// - Get a reference to the current user.

// - Get a reference to the letter activity.

// - Add letter to current user's queue via the bound action AddToQueue.

// For more info on AddToQueue, see https://msdn.microsoft.com/library/mt607880.aspx

return Sdk.request("GET", "/WhoAmI");

})

.then(function (request) {

var whoAmIResponse = JSON.parse(request.response);

var myId = whoAmIResponse.UserId;

// Get a reference to the current user.

return Sdk.request("GET", Sdk.getWebAPIPath() + "/systemusers(" + myId + ")/queueid/$ref")

})

.then(function (request) {

myQueueUri = JSON.parse(request.response)["@odata.id"];

// Get a reference to the letter activity.

return Sdk.request("GET", letterUri + "?$select=activityid")

})

.then(function (request) {

var letterActivityId = JSON.parse(request.response).activityid

var parameters = {

Target: {

activityid: letterActivityId,

"@odata.type": "Microsoft.Dynamics.CRM.letter"

}

}

//Adding the letter to the user's default queue.

return Sdk.request("POST", myQueueUri + "/Microsoft.Dynamics.CRM.AddToQueue", parameters);

})

.then(function (request) {

var queueItemId = JSON.parse(request.response).QueueItemId;

console.log("\tQueueItemId returned from AddToQueue Action: %s\n", queueItemId);

console.log("-- Working with custom actions --");

console.log("Custom action: sample\_AddNoteToContact");

// Add a note to an existing contact.

// This operation calls a custom action named sample\_AddNoteToContact.

// This custom action is installed when you install this sample's solution to your CRM server.

// - Add a note to an existing contact (e.g.: contactUri)

// - Get the note info and the contact's full name.

// For more info, see https://msdn.microsoft.com/library/mt607600.aspx#bkmk\_customActions

//sample\_AddNoteToContact custom action parameters

var parameters = {

NoteTitle: "The Title of the Note",

NoteText: "The text content of the note."

}

return Sdk.request("POST", contactUri + "/Microsoft.Dynamics.CRM.sample\_AddNoteToContact", parameters)

})

.then(function (request) {

var annotationid = JSON.parse(request.response).annotationid;

var annotationUri = Sdk.getWebAPIPath() + "/annotations(" + annotationid + ")";

// The annotation will be deleted with the contact when it is deleted.

return Sdk.request("GET", annotationUri + "?$select=subject,notetext&$expand=objectid\_contact($select=fullname)")

})

.then(function (request) {

var annotation = JSON.parse(request.response);

console.log("\tA note with the title '%s' and the content '%s' was created and associated with the contact %s.\n",

annotation.subject, annotation.notetext, annotation.objectid\_contact.fullname);

console.log("Custom action: sample\_CreateCustomer");

// Create a customer of a specified type using the custom action sample\_CreateCustomer.

// - Shows how create a valid customer of type "account".

// - Shows how to handle exception from a custom action.

var parameters = {

CustomerType: "account",

AccountName: CUSTOMERACCOUNTNAME

}

// Create the account. This is a valid request

return Sdk.request("POST", "/sample\_CreateCustomer", parameters)

})

.then(function (request) {

// Retrieve the account we just created

return Sdk.request("GET", "/accounts?$select=name&$filter=name eq '" + CUSTOMERACCOUNTNAME + "'");

})

.then(function (request) {

var customerAccount = JSON.parse(request.response).value[0];

var customerAccountId = customerAccount.accountid;

var customerAccountIdUri = Sdk.getWebAPIPath() + "/accounts(" + customerAccountId + ")";

entitiesToDelete.push(customerAccountIdUri);

console.log("\tAccount customer created with the name '%s'", customerAccount.name);

// Create a contact but uses invalid parameters

// - Throws an error intentionally

return new Promise(function (resolve, reject) {

var parameters = {

CustomerType: "contact",

AccountName: CUSTOMERACCOUNTNAME //not valid for contact

// e.g.: ContactFirstName and ContactLastName are required when CustomerType is "contact".

}

Sdk.request("POST", "/sample\_CreateCustomer", parameters) // This request is expected to fail.

.then(function () {

console.log("Not expected.")

reject(new Error("Call to sample\_CreateCustomer not expected to succeed."))

})

.catch(function (err) {

//Expected error

console.log("\tExpected custom error: " + err.message); // Custom action can return custom error messages.

resolve(); // Show the error but resolve the thread so sample can continue.

});

});

})

.then(function () {

// House cleaning.

console.log("\n-- Deleting sample data --");

if (deleteData) {

return Sdk.deleteEntities();

}

else {

console.log("Sample data not deleted.");

}

})

.catch(function (err) {

console.log("ERROR: " + err.message);

});

}

/\*\*

\* @function Sdk.deleteEntities

\* @description Deletes the entities created by this sample

\*/

Sdk.deleteEntities = function () {

return new Promise(function (resolve, reject) {

entitiesToDelete.unshift(opportunityUri) // Adding to the beginning so it will get deleted before the parent account.

// Re-open the created opportunity so it can be deleted.

Sdk.request("PATCH", opportunityUri, { statecode: 0, statuscode: 2 })

.then(function () {

// Get the opportunityclose URI so it can be deleted

return Sdk.request("GET", opportunityUri + "/Opportunity\_OpportunityClose/$ref")

})

.then(function (request) {

var opportunityCloseUri = JSON.parse(request.response).value[0]["@odata.id"];

// Adding to the opportunityclose URI it will get deleted before the opportunity.

entitiesToDelete.unshift(opportunityCloseUri)

/\*

These deletions have to be done consecutively in a specific order to avoid a Generic SQL error

which can occur because of relationship behavior actions for the delete event.

\*/

return Sdk.request("DELETE", entitiesToDelete[0]) //opportunityclose

})

.then(function () {

console.log(entitiesToDelete[0] + " Deleted");

return Sdk.request("DELETE", entitiesToDelete[1]) //opportunity

})

.then(function () {

console.log(entitiesToDelete[1] + " Deleted");

return Sdk.request("DELETE", entitiesToDelete[2])//account

})

.then(function () {

console.log(entitiesToDelete[2] + " Deleted");

return Sdk.request("DELETE", entitiesToDelete[3]) //Fourth Coffee account

})

.then(function () {

console.log(entitiesToDelete[3] + " Deleted");

return Sdk.request("DELETE", entitiesToDelete[4]) //Letter

})

.then(function () {

console.log(entitiesToDelete[4] + " Deleted");

return Sdk.request("DELETE", entitiesToDelete[5]) //Contact

})

.then(function () {

console.log(entitiesToDelete[5] + " Deleted");

return Sdk.request("DELETE", entitiesToDelete[6]) //AccountCustomer

})

.then(function () {

console.log(entitiesToDelete[6] + " Deleted");

resolve();

})

.catch(function (err) {

reject(new Error("Error from Sdk.deleteEntities: " + err.message));

});

});

};

/\*\*

\* @function Sdk.getUsersFullName

\* @description Retrieves the current user's full name.

\* @returns {Promise} - A Promise that returns the full name of the user

\*/

Sdk.getUsersFullName = function () {

return new Promise(function (resolve, reject) {

//Use WhoAmI Function (https://msdn.microsoft.com/library/mt607925.aspx)

Sdk.request("GET", "/WhoAmI")

.then(function (request) {

//Returns WhoAmIResponse ComplexType (https://msdn.microsoft.com/library/mt607982.aspx)

var myId = JSON.parse(request.response).UserId;

//Retrieve the systemuser Entity fullname property (https://msdn.microsoft.com/library/mt608065.aspx)

return Sdk.request("GET", "/systemusers(" + myId + ")?$select=fullname")

})

.then(function (request) {

//Return the users full name

resolve(JSON.parse(request.response).fullname);

})

.catch(function (err) {

reject("Error in Sdk.getUsersFullName function: " + err.message);

});

});

};

/\*\*

\* @function Sdk.createRequiredRecords

\* @description Creates data required by this sample program.

\* - Create an account with three 30 minute tasks.

\* - Create another account associated with an opportunity.

\* - Create a letter.

\* - Create a contact.

\* @returns {Promise} - resolve the promise if all goes well; reject otherwise.

\*/

Sdk.createRequiredRecords = function () {

console.log("-- Creating sample data --");

// Create a parent account, an associated incident with three

// associated tasks(required for CalculateTotalTimeIncident).

return new Promise(function (resolve, reject) {

Sdk.createAccountWithIncidentAndThree30MinuteClosedTasks()

.then(function (iUri) {

incidentUri = iUri;

//Create another account and associated opportunity (required for CloseOpportunityAsWon).

return Sdk.createAccountWithOpportunityToWin();

})

.then(function (oUri) {

opportunityUri = oUri;

// Create a letter to use with AddToQueue action.

var letter = {

description: "Example letter"

}

return Sdk.request("POST", "/letters", letter)

})

.then(function (request) {

letterUri = request.getResponseHeader("OData-EntityId");

entitiesToDelete.push(letterUri);

// Create a contact to use with custom action sample\_AddNoteToContact

var contact = {

firstname: "Jon",

lastname: "Fogg"

}

return Sdk.request("POST", "/contacts", contact)

})

.then(function (request) {

contactUri = request.getResponseHeader("OData-EntityId");

entitiesToDelete.push(contactUri);

resolve()

})

.catch(function (err) {

reject("Error in Sdk.createRequiredRecords function: " + err.message);

});

});

}

/\*\*

\* @function Sdk.createAccountwithIncidentAndThree30MinuteClosedTasks

\* @description Create an account and associate three 30 minute tasks. Close the tasks.

\* @returns {Promise} - A Promise that returns the uri of an incident created.

\*/

Sdk.createAccountWithIncidentAndThree30MinuteClosedTasks = function () {

return new Promise(function (resolve, reject) {

var iUri; // incidentUri

// Create a parent account for the incident.

Sdk.request("POST", "/accounts", { name: "Fourth Coffee" })

.then(function (request) {

// Capture the URI of the created account so it can be deleted later.

var accountUri = request.getResponseHeader("OData-EntityId");

entitiesToDelete.push(accountUri);

// Define an incident associated with the account with three related tasks.

// Each task has a 30 minute duration.

var incident = {

title: "Sample Case",

"customerid\_account@odata.bind": accountUri,

Incident\_Tasks: [

{

subject: "Task 1",

actualdurationminutes: 30

},

{

subject: "Task 2",

actualdurationminutes: 30

},

{

subject: "Task 3",

actualdurationminutes: 30

}

]

};

// Create the incident and related tasks.

return Sdk.request("POST", "/incidents", incident)

})

.then(function (request) {

iUri = request.getResponseHeader("OData-EntityId");

// Retrieve references to the tasks created.

return Sdk.request("GET", iUri + "/Incident\_Tasks/$ref")

})

.then(function (request) {

// Capture the URL for the three tasks in this array.

var taskReferences = [];

JSON.parse(request.response).value.forEach(function (tr) {

taskReferences.push(tr["@odata.id"]);

});

// An array to hold a set of promises.

var promises = [];

// The data to use to update the tasks so that they are closed.

var update = {

statecode: 1, //Completed

statuscode: 5 //Completed

}

// Fill the array with promises

taskReferences.forEach(function (tr) {

promises.push(Sdk.request("PATCH", tr, update))

})

// When all the promises resolve, return a promise.

return Promise.all(promises);

})

.then(function () {

// Return the incident URI to the calling code.

resolve(iUri);

})

.catch(function (err) {

// Differentiate the message for any error returned by this function.

reject(new Error("ERROR in Sdk.createAccountwithIncidentAndThree30MinuteClosedTasks function: " + err.message))

});

});

}

/\*\*

\* @function Sdk.createAccountwithOpportunityToWin

\* @description Create an account and an associated opportunity.

\* @returns {Promise} - A Promise that returns the uri of an opportunity.

\*/

Sdk.createAccountWithOpportunityToWin = function () {

return new Promise(function (resolve, reject) {

var accountUri;

var account = {

name: "Sample Account for WebAPIFunctionsAndActions sample",

opportunity\_customer\_accounts: [{

name: "Opportunity to win"

}]

};

Sdk.request("POST", "/accounts", account) // Create the account.

.then(function (request) {

accountUri = request.getResponseHeader("OData-EntityId");

entitiesToDelete.push(accountUri);

// Retrieve the opportunity's reference.

return Sdk.request("GET", accountUri + "/opportunity\_customer\_accounts/$ref")

})

.then(function (request) {

var oUri = JSON.parse(request.response).value[0]["@odata.id"];

resolve(oUri); // Return the opportunity's uri.

})

.catch(function (err) {

reject(new Error("Error in Sdk.createAccountwithOpportunityToWin: " + err.message));

});

});

};