

Maximo Federated Resource

V1 - Updated: 03/01/2016

Contents

Introduction	2
Overview	3
Getting Started.....	5
JSON Resource Type.....	5
Creating a JSON Resource	8
Example: Federated Resource using 2 Maximo environments	15
Example: Federated Resource - Merge.....	23
Federated Resource Concepts	30
Accessing Federated Resource via the Maximo JSON API	33

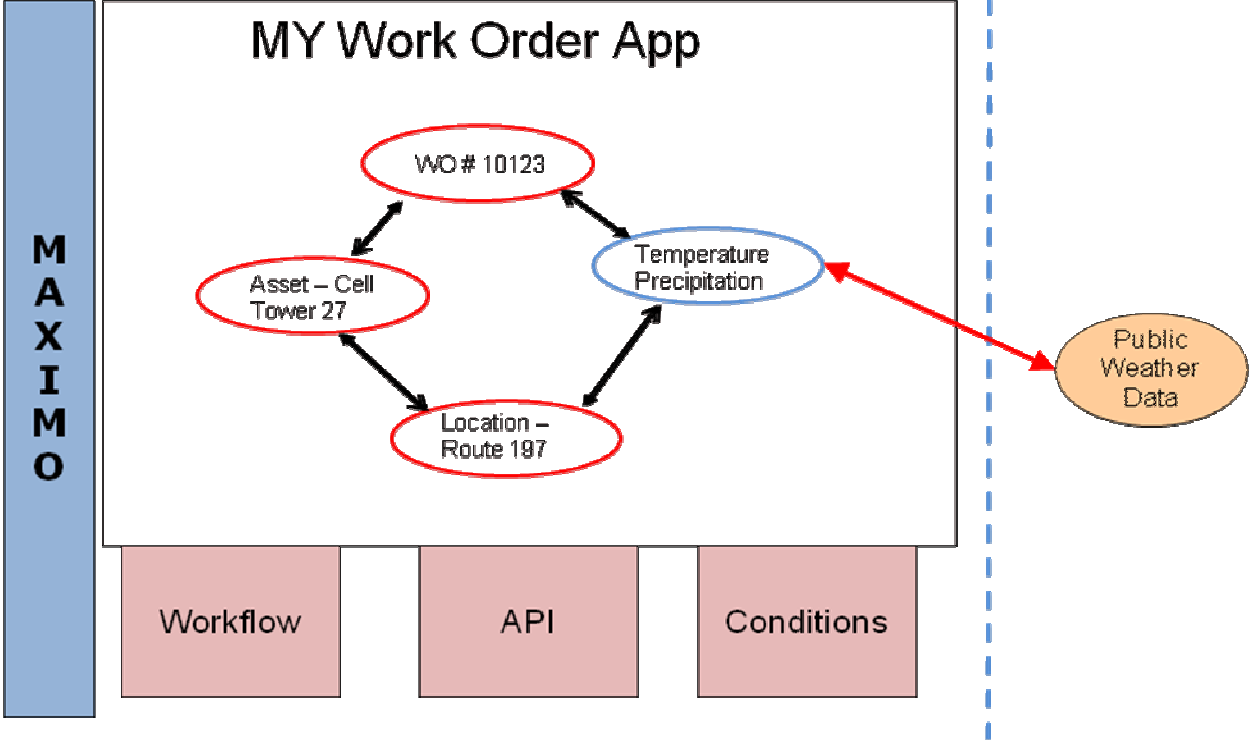
Introduction

Beginning Maximo 7.6.0.3, a new integration featured called Federated Resource (also described as Federated MBO) was introduced and this included a new integration application named JSON Resources. This feature can expose external data as a MBO within Maximo without the need to replicate the data on an on-going basis to keep a 'fresh' copy in Maximo. Being represented as a MBO this data can be used by standard Maximo components such as an application, a CRON task or a Workflow. The implementation of the Federated Resource supports querying and updating of external data (some limitations are noted below).

Overview

The Federated Resource concept, initial release in 7.6.0.3, can expose external data as a MBO within Maximo, providing that the external data is supported through a RESTful JSON API. When supported, the external data can be registered using the JSON Resources application and then the data becomes available for MBO processing within the Maximo Business Object framework. Having this feature provides a simpler approach to sharing data with Maximo rather than requiring the external data being replicated into the Maximo environment. When used, the Federated MBO will reach out to the external application on-demand and retrieve the most up to date version of the data that the external REST API provides. Using the Federated Resource to query external and view within Maximo is the most common scenario. Using the Federated Resource to perform updates to the external data is possible, however this likely will require additional coding in the Processing Class of the Resource Type to enable updates using the external JSON API.

A simple use case could be defining a JSON Resource for external weather data that could be linked to a Maximo Location. When viewing a Work Order for the Asset at that location, the weather data could be displayed using the JSON Resource that supports weather data and that information could be used for work scheduling. Additionally that weather data could be used by other Maximo processes such as a Workflow or part of a Condition. External clients accessing Maximo data using the Maximo JSON API could retrieve that weather data as well.



Getting Started

Before attempting to use the Federated Resource you should determine whether this is the best approach for your integration requirements. Consider these questions:

1. Do you need to have external data available in Maximo as a MBO or available through Maximo's JSON API?
2. Would you prefer to not replicate that external data into Maximo on a recurring basis?
3. Is that external data accessible using a RESTful JSON API?

If you answered Yes to all 3 questions then the Federated Resource should be considered for your integration scenario.

Keep in mind that the use of Federated Resource is intended to support data that is 'complementary' to your Maximo data.

JSON Resource Type

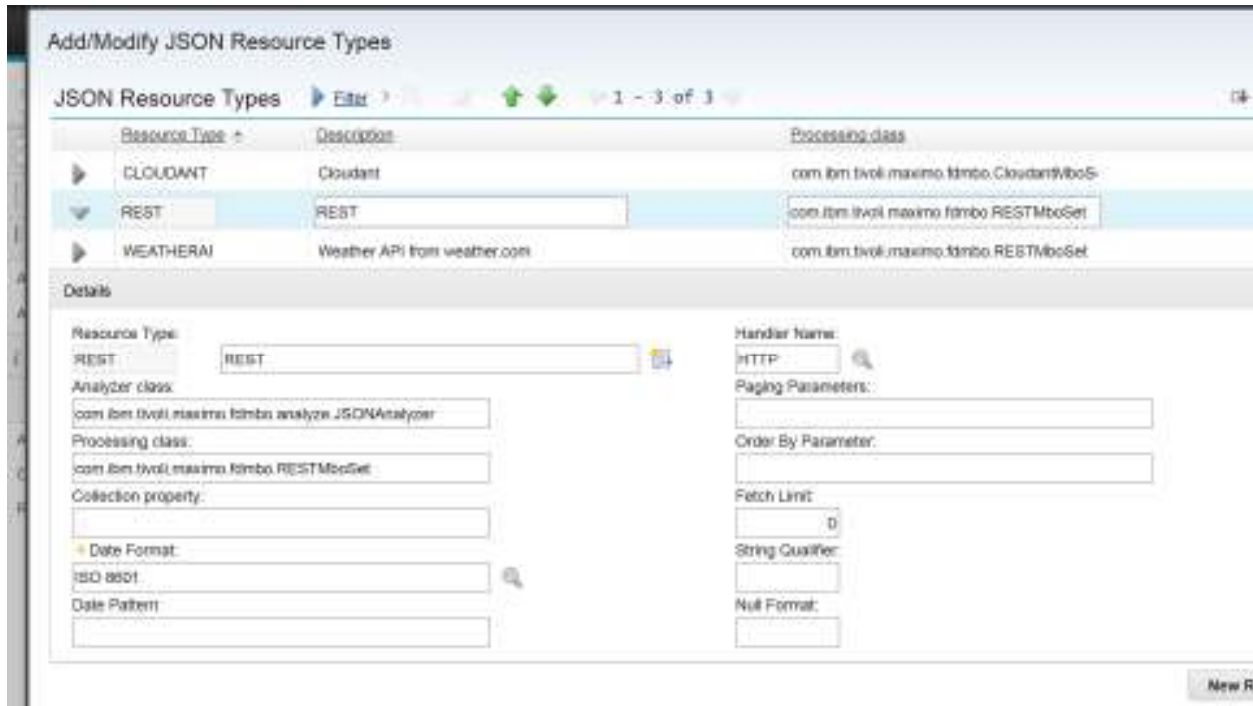
To create a JSON Resource for your external data, you need to have a predefined Resource Type that provides a common set of data and functionality specific to interacting with the external api. It is possible that the Resource Type and its metadata can be re-used with multiple JSON Resources.

Unfortunately, not all RESTful JSON APIs are equal. There are some cases where APIs go beyond REST and/or JSON standards and some additional coding is required in order to use the API with a JSON Resource. The first step in creating a JSON resource is to determine if one of the provided JSON Resource Types will work with your external JSON API or if a new resource type needs to be defined.

Maximo provides 3 resource types out of the box. One is for the CLOUDANT REST API, a second one is provided for the API from weather.com and the third is a generic 'REST' API resource type. In many cases the REST resource type will work with an external JSON api, however if you determine it does not, then you may have to create your own Resource Type which would include providing an Analyzer Class and a Processing Class. These Java classes would have to be provided to handle the 'specialties' of the API that are not handled by the Maximo REST Resource type, similar to what was done to create the CLOUDANT and weather.com resource types.

You should attempt to use the 'out of box' REST resource type provided with Maximo to determine if works with your external JSON API. If it does not, then you will need to create a new Resource Type and provide the necessary Java Classes.

Below is a screen shot of the provided 'REST' Resource type:



The Resource Type attributes are as follows:

Resource Type: Unique Name of the Resource Type - this has a related Description and Long Description.

Analyzer Class - this Java class is used to interpret the sample JSON in order to generate Maximo non-persistent objects (MBOs) that represent the external data. This class must be included as part of the Maximo EAR file. This class would only be used when the Resource Definition has the Usage set to Object (not API). The 'Usage' will be discussed in the section covering the creation of a JSON Resource.

Processing Class - this Java class would do any specific processing of data when resource data is retrieved from the external source. This class must be included as part of the Maximo EAR file. This class would only be used when the Resource Definition has the Usage set to Object (not API). This class is registered for the MBO/Table in the DB Configuration application and could be changed there as needed.

Collection Property - multi-noun identifier within the external JSON data when it contains a collection of data rather than a single instance of data.

Date Format - choices are ISO8601, Milliseconds, Seconds or Custom - identifies the date format in the external JSON data.

Date Pattern - only used when Date Format is Custom - sample date format could be yyyyymmdd.

Handler Name - the value must be an existing integration handler and it will be used to create the End Point (if needed) during the JSON Resource creation. This Handler would support accessing the external JSON data. The HTTP Handler is the most common Handler for JSON Resources.

Paging Parameters - a comma separated list of 2 query parameter names used in the End Point URL when retrieving the external data - the first name identifies the Page starting count while the second parameter identifies the Page size (or limit). Two parameter names must be provided. This value can be overridden in the configuration of a Resource.

Fetch Limit - This is the value of page size (limit) that is assigned to the 2nd parameter name from the Paging Parameters (above). This holds a limit on the number of rows to select for the JSON resource and to maintain in a cache. This is intended to support efficient retrieval (Paging) of external data. A value that is slightly higher than the 'normal' amount of data to be retrieved most often, would be an appropriate value. A value of 0 means no paging is supported. This value can be overridden in the configuration of a Resource.

Order By Parameter - Identifies the name of the query parameter name in the End Point URL that supports the ordering of external JSON data when queried. An example could be: `_orderby`

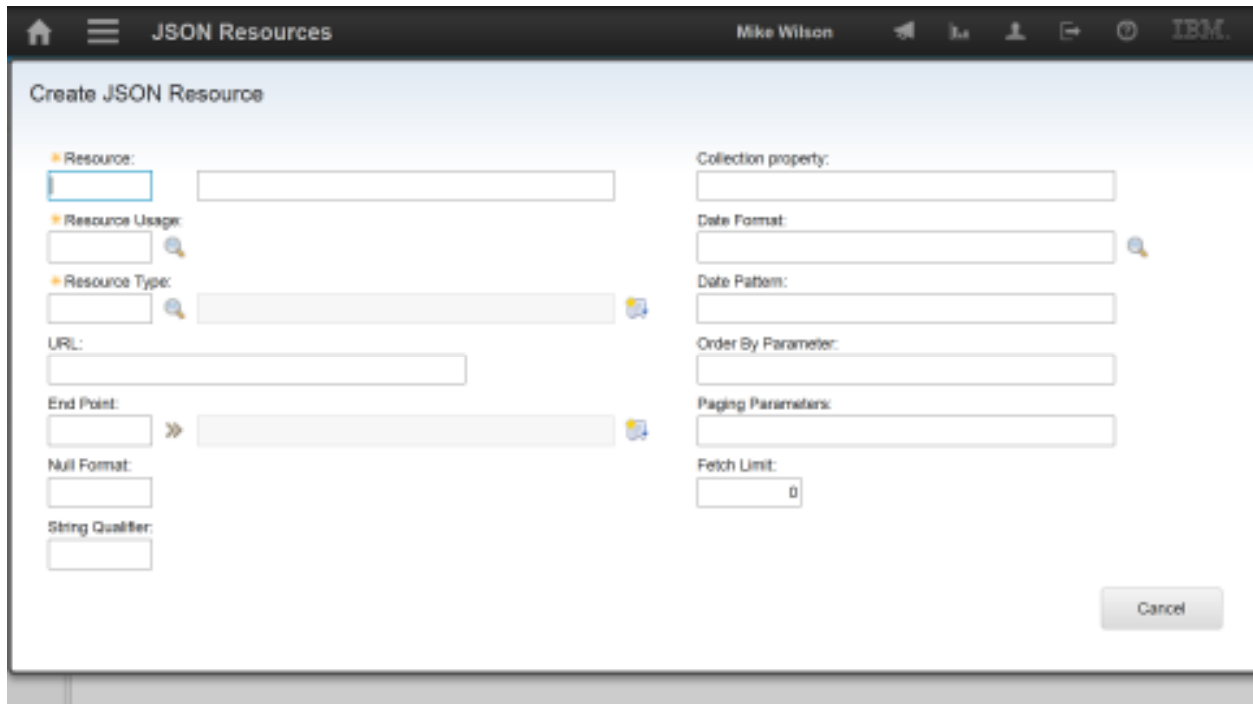
String Qualifier - Some external data sources require that string values provided as query parameters be wrapped in a qualifier, such as a quote (') or double quotes ("). For any query parameter that is set using a Maximo field value (bind variable) and the value is a String, it will be wrapped in the URL with the value provided in this property. This value can be overridden in the configuration of a Resource.

Null Format - Some external data sources require that for any query parameters that do not have a value, they must provide an explicit value (such as +null+) that denotes a null value. For any query parameter that is set using a Maximo field value (attr bind variable) and its value is null, it will be set in the URL with the value provided in this property. This value can be overridden in the configuration of a Resource.

Creating a JSON Resource

To create a new JSON Resource using the JSON Resources application, the user will proceed through a multi-dialog wizard providing the necessary data. Depending upon which Resource Usage is selected will determine how many dialogs will be displayed.

Screen 1:



The screenshot shows a web application interface for creating a JSON resource. The title bar indicates the application is 'JSON Resources' and the user is 'Mike Wilson'. The main heading is 'Create JSON Resource'. The form is organized into two columns. The left column contains fields for 'Resource', 'Resource Usage', 'Resource Type', 'URL', 'End Point', 'Null Format', and 'String Qualifier'. The right column contains fields for 'Collection property', 'Date Format', 'Date Pattern', 'Order By Parameter', 'Paging Parameters', and 'Fetch Limit'. A 'Cancel' button is positioned at the bottom right of the dialog.

The User needs to provide a unique Resource name, select a predefined Resource Type and select a Resource Usage value of either OBJECT or API. The Resource Type may provide default values for the rest of the fields in this dialog, excluding the End Point and URL.

NOTE: if you are creating a resource with a Usage of OBJECT, you will need to provide a sample JSON data in order to generate the corresponding MBOs and their attributes. You should retrieve the sample json prior to starting the creation process.

Resource Usage

When the Usage is set to API, the creation of the resource is completed once this dialog has been processed. When the Usage is set to Object, the creation process will proceed through two additional dialogs (discussed further down in this document) to complete the JSON Resource Definition.

The Usage identifies whether the external data is going to be accessed only as JSON data (using the Maximo JSON API) and does not need to be converted to a Maximo Business Object (MBO). When accessing the external data using the JSON API (no MBO needed), the processing of the external JSON data can be done more efficiently without representing the external data using a MBO.

When using the Maximo JSON API to query external JSON data represented as a MBO in Maximo (through a JSON Resource with a Usage of Object), the processing of the external data would be as follows:

1. Retrieve external JSON data
2. Convert the JSON data to MBO data
3. Convert MBO data to the response format data (JSON)

In the case of accessing the data using the Maximo JSON API, the conversion of data through a MBO is costly in terms of performance and provides no value if the MBO data is not being leveraged for other reasons.

With this in mind, setting JSON Resource with a Usage of API (rather than Object) would result in this processing:

1. Retrieve external JSON data
2. Provide external JSON data in response (*note that the response data would have any related Maximo data in the Maximo JSON format and the federated resource data in the JSON format provided by that external application, not converted to the Maximo JSON format*)

NOTE: The querying of external JSON data in this manner is supported only through the Maximo OSLC/JSON API and is not supported by other MIF integration options such as Web Services, REST api or XML/HTTP. This feature using the Maximo JSON API supports only querying of external JSON data, not updating.

URL and End Point

The URL, entered or provided by the End Point, provides access to the external JSON data. There are 3 scenarios regarding the entry of a URL and End Point.

- When just a URL to the external data is provided, processing will create an End Point using the URL and the Handler specified in selected the Resource Type. If the Resource Type has no Handler defined, then the default HTTP Handler will be used.

- An existing integration End Point can be selected without a URL being entered. The End Point selected will be used to retrieve the external data
- When a URL and End Point are both entered, the URL is saved with the Resource and acts as an override to the End Point URL parameter when the external data is accessed. This allows one End Point to be shared across multiple resources where one, or more resources, have a different URL. This allows the parameters of the End Point, such as login credentials, to be shared across multiple resources.

The remaining fields on this initial creation dialog can have defaults from the Resource Type and can be overridden here for the specific resource. Some of these properties are used in the processing of the external data while others are used to support query parameters in the URL when accessing the external data.

NULL Format

Described under Resource Type

String Qualifier

Described under Resource Type

Collection Property

Described under Resource Type

Data Format and Pattern

Described under Resource Type

Order By Parameter

Described under Resource Type

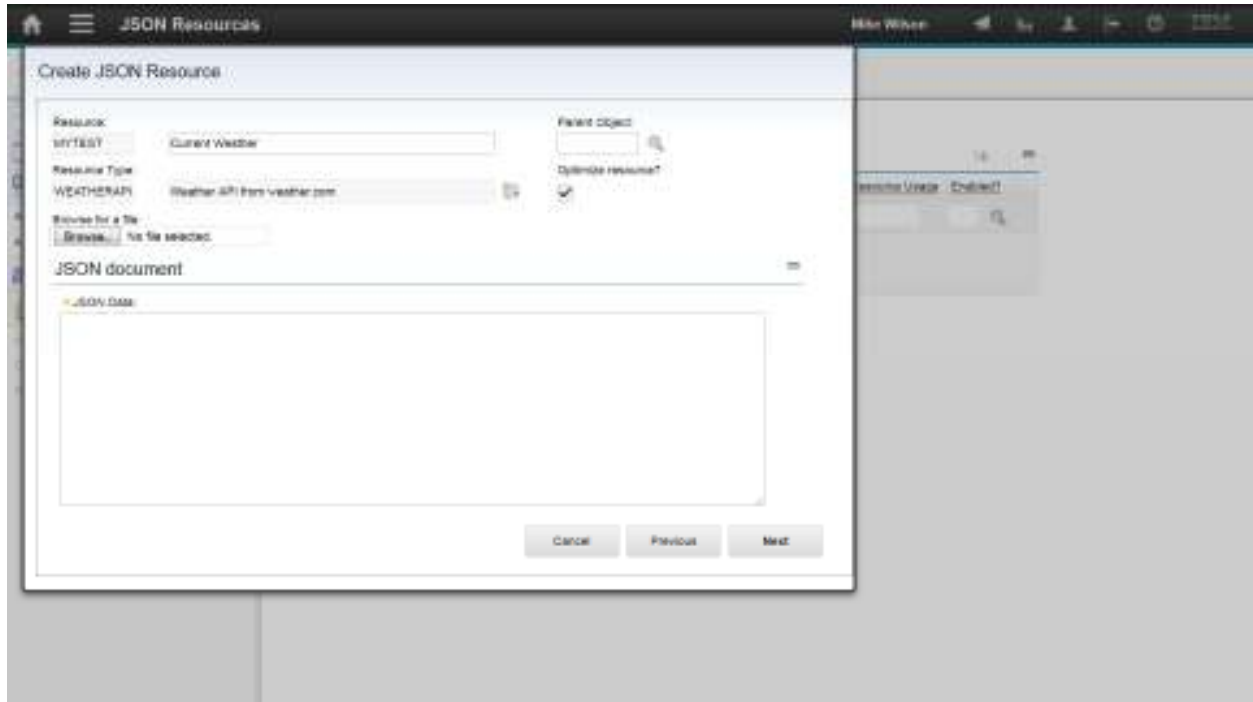
Paging Parameters

Described under Resource Type

Fetch Limit

Described under Resource Type

If the selected Usage is API, then the JSON Resource creation ends with the first dialog. If the Usage is OBJECT, then the second dialog below will appear:



In this dialog you can configure:

- Parent Object - select a parent object (MBO) that will be related to your JSON Resource MBO. For example if you were using a Service Address to provide data to your URL then you could select the SERVICEADDRESS MBO as the Parent Object. This will allow you to use the zip code (for example) in the request for weather data. Processing will create a Maximo Relationship from the selected parent object and the root MBO (non-persistent) of your Federated Resource.
- Optimize Resource - The Optimize Resource checkbox will merge multiple levels of the resource (JSON) data per the rules below:
 1. all JSON objects that have 1:1 relation with the parent object will be merged to the parent object. All property names for that merged object will have to be checked for unique names (update to have unique names if necessary) in the newly defined scope of parent object.
 2. if the root object is a wrapper element (a JSON object that has no simple properties, rather it has only JSON objects and arrays as its properties is called a

wrapper element), and there is a single child which is an array - that child would be merged to the parent.

For these 2 rules - the merge is virtual to the designer - as it will still show the JSON in its true form. The merge will only impact the MBO generation.

- JSON Data - this is the sample data from the external JSON API. This data will be parsed in order to support the creation of non-persistent MBOs that represent this data within Maximo.

Note: when copying and pasting JSON data from a browser, it is recommended that you do View Source/View Page Source in the browser and then copy the data to avoid capturing extraneous characters that can cause errors during parsing of the JSON data.

When you hit Next on the dialog above, the processing will parse the JSON data and present the following dialog which will list the JSON objects and their properties. The information presented is dictated by the structure of the data provided in the sample JSON.

The screenshot shows the 'Create JSON Resource' dialog in IBM Maximo. The resource is named 'MYTEST' with the type 'WEATHERAPI'. The 'Resource Objects for Resource' table lists the following objects and their paths:

Relation Property Name	Property Path
metadata	/metadata
observation	/observation
imperial	/observation/imperial

The 'Properties For metadata' table lists the following properties:

Property Name	Attribute Name	Field Title	Type	Length
status_code	STATUS_CODE	STATUS_CODE	INTEGER	12
expire_time_gmt	EXPIRE_TIME_GMT	EXPIRE_TIME_GMT	INTEGER	12
language	LANGUAGE	LANGUAGE	ALN	50

The 'Resource Objects for Resource' lists the JSON Objects and their path. Below is snippet of the sample JSON used for this resource creation (some data was removed for easier viewing of the sample):

```
{
  "metadata":
  {
    "language": "en-US",
    "expire_time_gmt": 1453478889,
    "status_code": 200
  },
  "observation":
  {
    "class": "observation",
    "obs_time_local": "2016-01-22T10:45:00-0500",
    "sunrise": "2016-01-22T07:06:55-0500",
    "sunset": "2016-01-22T16:45:05-0500",
    "day_ind": "D",
    "dow": "Friday",
    "wdir_cardinal": "NW",
    "sky_cover": "Partly Cloudy",
    "imperial":
    {
      "wspd": 15,
      "gust": null,
      "temp": 25,
      "temp_change_24hour": -2,
      "precip_24hour": 0.00,
    }
  }
}
```

Beyond the root there are 3 JSON objects

- / (root)
- metadata
- observation
- imperial

Using this data and with the Optimize Resource flag checked, only 1 non-persistent MBO, MYTEST, will be created and the properties for all JSON objects will become attributes of the MYTEST MBO (per the Optimization rules mentioned above).

If SERVICEADDRESS was entered as the Parent Object, a relationship would be created to link SERVICEADDRESS to MYTEST.

With the Optimize Resource flag unchecked, 4 non-persistent MBOs will be created, MYTEST, METADATA, OBSERVATION and IMPERIAL. Maximo Relationships will also be created to support this hierarchical structure:

```
MYTEST
  METDATA
  OBSERVATION
  IMPERIAL
```

Use of the Optimize Resource processing is dependent upon the structure of the external JSON data and which objects and properties you need to use in Maximo. If data is needed from an object that is an array then you cannot optimize since there would be a 1 to many relationship from the parent object (which could be the root) and the child object.

The user can control the number of JSON Objects and Properties created by adjusting the content of the JSON sample data. In the dialog above, the default names, types and lengths for the MBO attributes can be changed prior to the creation. Once MBOs are created, attributes can only be updated using the DB Configuration application (as it would be for other Maximo objects).

Example: Federated Resource using 2 Maximo environments

Set up

For the examples below we will use two Maximo environments where one environment (MX A) has just locations the other environment (MX B) has locations with location meters. MX B will act as the 'external' data source and MX A wants visibility to the location meters without replicating that meter information. To do this:

- A. MX B will expose location meters through the Maximo JSON API.
- B. MX A will create a Federated Resource for location meters using MX B's JSON API to access that data on demand.

In MX B, a new Integration object structure, FEDLOCMETER, was created with the Locations and Locationmeter objects. The system property, mxe.oslc.webappurl, was updated to reflect the actual host:port of Maximo to enable access using REST/JSON.

A sample JSON data was captured by retrieving a single location that contained at least 1 meter:

First a request for a collection of Locations:

http://MX B Host:Port/maximo/oslc/os/fedlocmeter

From which a single Location URL was used:

http://MX B Host:Port/maximo/oslc/os/fedlocmeter/_UFQxMDAvQkVERk9SRA--

where *_UFQxMDAvQkVERk9SRA--* is the resource ID of the location. The following is a snippet of the JSON data (not the entire content):

```
{
  "changedate":"2011-02-15T15:19:23-05:00",
  "pluscpmextdate":false,
  "status_description":"Operating",
  "location":"PT100",
  "children":false,
  "type":"OPERATING",
  "autowogen":false,
  "_rowstamp":"[0 0 0 0 -114 124 -113]",
  "haschildren":false,
  "orgid":"EAGLENA",
  "description":"PT100",
  "type_description":"Operating Location",
  "siteid":"BEDFORD",
  "href":"http://host:port/maximo/oslc/os/fedlocmeter/_UFQxMDAvQkVERk9SRA--",
```

```

    "isrepairfacility":false,
    "isdefault":false,
    "locationsid":359,
    "pluscloop":false,
    "changeby":"WILSON",
    "status":"OPERATING",
    "statusdate":"2011-02-15T15:19:23-05:00",

    "locationmeter_collectionref":"http://host:port/maximo/oslc/os/fedlocmeter/_UFQxMDAvQkVERk9SRA--/int_locationmeter",
    "useinpopr":false,
    "hasparent":false,
    "statusiface":false,
    "disabled":false,
    "locationmeter":
  [
  {

    "avgcalcmethod_description":"Average not recalculated",
    "avgcalcmethod":"STATIC",
    "changedate":"2016-01-25T10:10:39-05:00",
    "locationmeterid":202,
    "sincelastrepair":0.0,
    "measureunitid":"HOURS",
    "sincelastoverhaul":0.0,
    "changeby":"WILSON",
    "lifetodate":0.0,
    "sincelastinspect":0.0,
    "readingtype":"DELTA",
    "localref":"http://host:port/maximo/oslc/os/fedlocmeter/_UFQxMDAvQkVERk9SRA--/int_locationmeter/0-202",
    "metername":"FLTHRS",
    "readingtype_description":"Incremental usage",
    "_rowstamp":"[0 0 0 0 0 -114 124 -112]",
    "orgid":"EAGLENA",
    "sinceinstall":0.0,
    "active":true,

```



```

    "href":"http://host:port/maximo/oslc/os/fedlocmeter/_UFQxMDAvQkVERk9SRA--
    #TE9DQVRJT05TL0xPQ0FUSU9OTUVURVivUFQxMDAvRkxUSFJTL0JFREZPUkQ-",
    "average":4.0
  },
],
}

```

This JSON data will be used as input when creating the JSON resource in MX A. What is provided will be the basis for the non-persistent objects and attributes that will be created to represent the external data (from MX B) in Maximo. Before pasting the above sample JSON data into the JSON Resource definition you can choose to remove any content that you have no plans to use. For our example, we will use the following as the sample JSON:

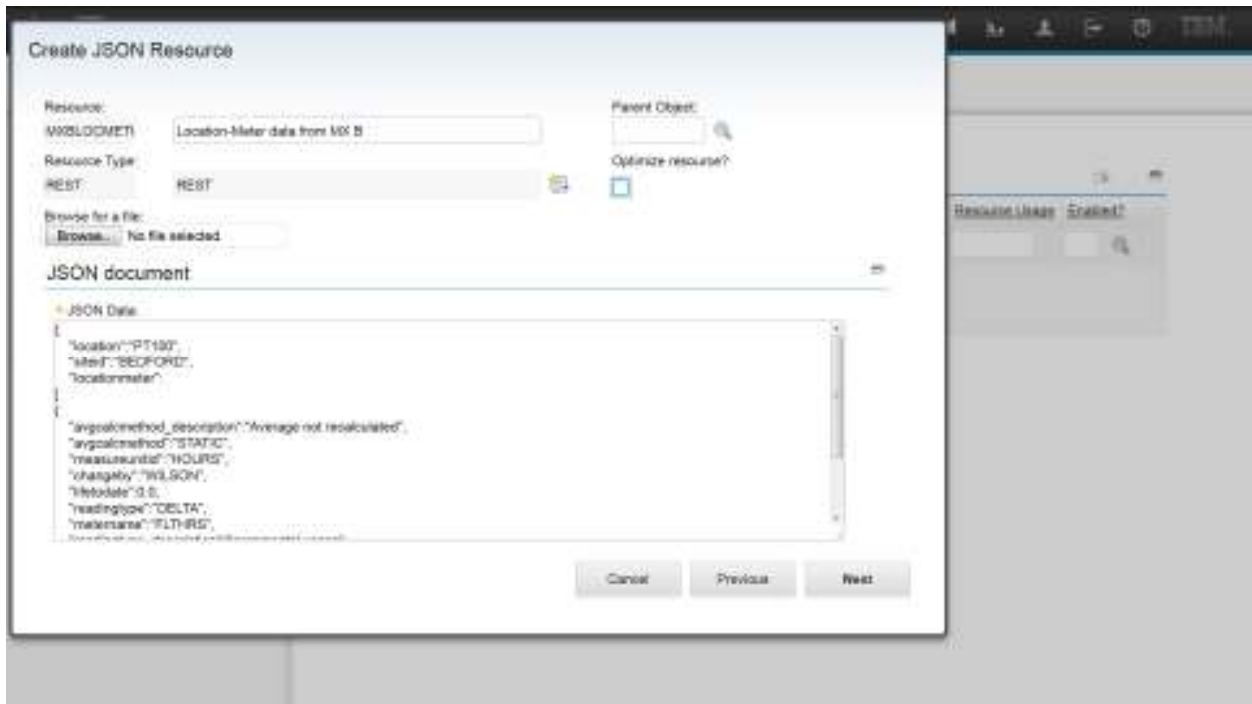
```

{
  "location":"PT100",
  "siteid":"BEDFORD",
  "locationmeter":
  [
  {
    "avgcalcmethod_description":"Average not recalculated",
    "avgcalcmethod":"STATIC",
    "measureunitid":"HOURS",
    "changeby":"WILSON",
    "lifetodate":0.0,
    "readingtype":"DELTA",
    "metername":"FLTHRS",
    "readingtype_description":"Incremental usage",
    "active":true
  },
  ],
}

```

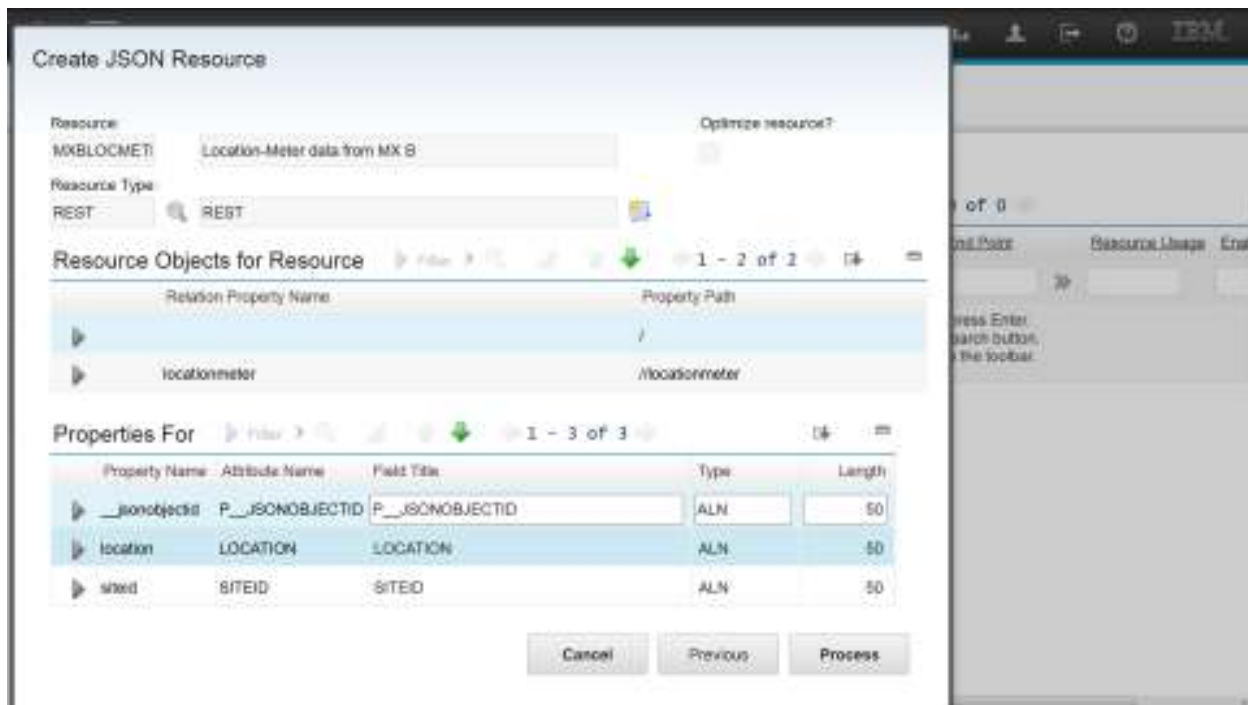
At the Location level, only the location number and site ID is needed along with a handful of properties of the locationmeter data.

This smaller JSON data is pasted in during the JSON Resource creation below:



No Parent object is assigned and the Optimize Resource was left unchecked since the data contains an array, locationmeter[].

The next screen shows the objects and attributes that will be created in Maximo. The root object (Path = /) will take on the Resource Name, MXBLOCMETER, as its object name and the child object name will be LOCATIONMETER. In cases where the object name might already exist in Maximo, the processing will append a sequence number to reach a unique name.



Depending upon how you plan to use the external data, you may need to create a relationship between an existing Maximo object and the new MBO you created for the Federated Resource. If the Federated Resource is a stand-alone object that you plan to create a new application for then a relationship is not required.

For our example we plan to display this data when viewing locations in MX A, to do this requires a relationship from the LOCATIONS object to the MXBLOCMETER object. What this relationship enables is the ability to provide Location data to the external resource URL so that it can be used for retrieving data. For example, when viewing location PT100 in sited BEDFORD in MX A, we want to retrieve the location meters from MX B. To do this requires that the Federated Resource URL include parameters that identify the location and site in order to select the correct data from the external resource (MX B). To enable this capability:

- Provide a Parent Object (LOCATIONS) during the creation process and the application will automatically generate a relationship between the parent and the Federated Resource objects

OR

Manually create a relationship between the objects using the DB Configuration application.

Note that the relationship does not require a 'where' clause since the purpose is to allow the passing of the parent object data to Federated Resource's URL, which was saved as an Integration End Point (using same name as the resource) during the resource creation.

With the relationship created, you may need to revisit the integration end point for the resource if the one entered during the creation process did not include bind variables for the parent object data (assuming its needed).

- In our example, to access the Federated Resource data in MX B while viewing a location in MX A, the URL would be as follows:

```
http://MX B
Host:Port/maximo/oslc/os/fedlocmeter?lean=1&oslc.where=location=attr:location
n and siteid=attr:siteid&oslc.select=*
```

example:

```
http://MX B
Host:Port/maximo/oslc/os/fedlocmeter?lean=1&oslc.where=location=attr:"PT100"
" and siteid=attr:"BEDFORD"&oslc.select=*
```

the **attr:** identifies that the bind variables are attributes of the parent object (there are other 'types' of bind values available - discussed further down in this document). In this case the fields being used from the parent object are **location** and **sited** of the Location that is being viewed in MX A.

In addition to the where clause there is an oslc.select to select all properties of the resource, without this the Maximo JSON API will return just a URL to the resource.

String Qualifier

The Maximo JSON API (being used in MX B) requires that the string variables (PT100 & BEDFORD) passed in the URL be wrapped with double quotes. Given that, the Resource definition (or the Resource Type) should have " (double quote) configured as the String Qualifier value.

Collection Property

If you used the above URL to retrieve Location PT100 in MX B, the resulting JSON will be a collection of locations that contains one location (PT100). The collection is identified by the array called member[] :

```
{
  "member":
  [
    {
      "location": "PT100",
      "siteid": "BEDFORD",
      "locationmeter":
      [
```

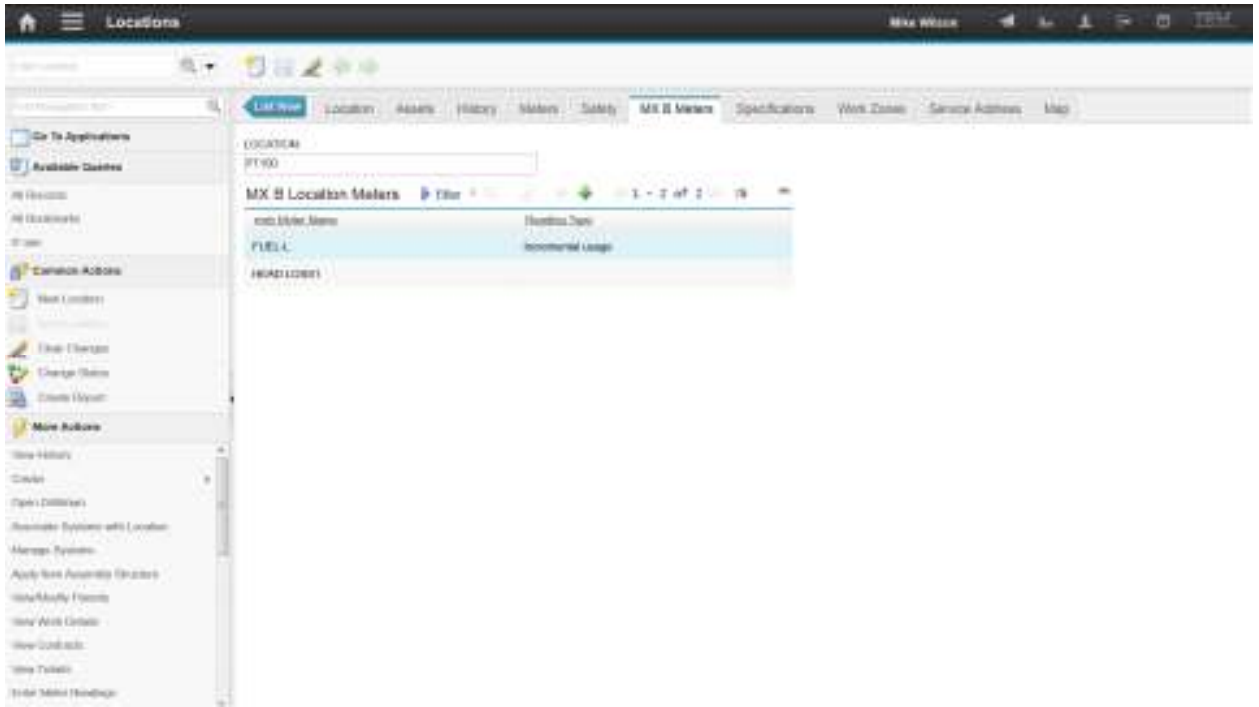
```
{
    "avgcalcmethod_description":"Average not recalculated",
    "avgcalcmethod":"STATIC",
    "measureunitid":"HOURS",
    "changeby":"WILSON",
    "lifetodate":0.0,
    "readingtype":"DELTA",
    "metername":"FLTHRS",
    "readingtype_description":"Incremental usage",
    "active":true
}
]
]
}
```

In order for the processing to parse this data properly, the collection property defined on the JSON Resource should be configured with the value of *member*.

Keep in mind that this Federated Resource/End Point could be used from other parent objects/applications (besides locations) providing that the parent object has the same attributes. For example, a work order contains the location and siteid attributes, therefore this Federated Resource could be used from the Work Order application if a relationship from WORKORDER to MXBLOCMETER is created.

Once the Federated Resource is fully configured it can be used for view location meters. The screen shot below shows a Locations application with a new tab named, MX B Meters, and it displays the location and related meter data that was retrieved using the Federated Resource MBOs (MXBLOCATION & LOCATIONMETER4).

To support the UI changes below, within application designer a new tab was created in the Locations application and the relationship (from LOCATIONS) MXBLOCMETER was specified for the tab to access the Federated Resource data.



Fetch Limit

In the screen above a list of meters from MX B is being displayed. If the table view (configured as part of the application in Application Designer) limits to showing 3 meters at a time, the configuration of the resource should make sure Fetch Limit is at least 3 (or higher) so that it takes only a single request to fill the table.

Example: Federated Resource - Merge

Set up

Similar to the example above, we will use two Maximo environments where one environment (MX A) wants to display data from the other environment (MX B). In this example each Maximo has a common set of locations and these locations have meters however the meters are not the same in each system. In this example, we will use a Federated MBO to retrieve the meters from MX B Locations and merge that list of meters with list of meters that currently exist for the location in MX A. So when the user is viewing a single list of meters for a location the source of those meters will be a consolidated list from MX A and MX B.

To support the Merge feature, the federated resource must be a single level structure (not a hierarchy). In the prior example we created an object structure (in MX B) that contained a hierarchy of:

```
Location
  Locationmeter
```

For the merge feature we cannot use this object structure so will create a new object structure (FEDMETER) with just the locationmeter object within MX B

A sample JSON data was captured by retrieving a single locationmeter:

```
http://MX B Host:Port /maximo/oslc/os/fedmeter/_UFQxMDAvRIVFTC1HL0JFREZPUkQ-?lean=1
```

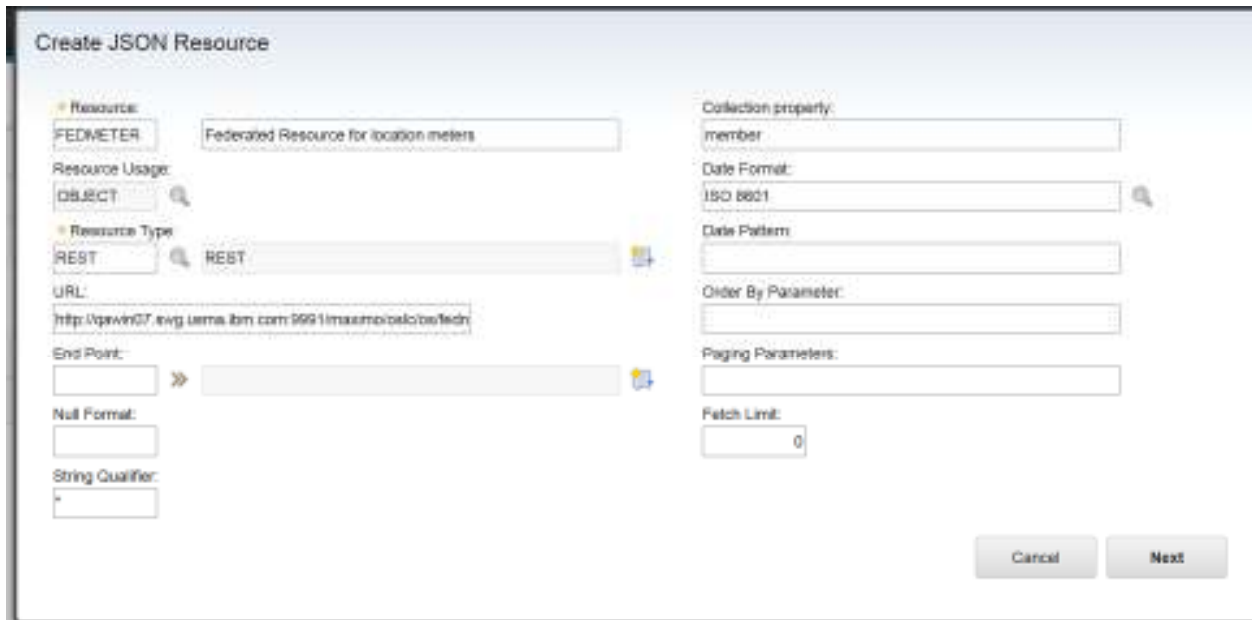
where _UFQxMDAvRIVFTC1HL0JFREZPUkQ- is the resource ID of the locationmeter. The following is the JSON data:

```
{
  "avgcalcmethod_description": "Average not recalculated",
  "avgcalcmethod": "STATIC",
  "changedate": "2016-01-27T11:50:24-05:00",
  "locationmeterid": 215,
  "sincelastrepair": 0.0,
  "measureunitid": "GALS",
  "sincelastoverhaul": 0.0,
```

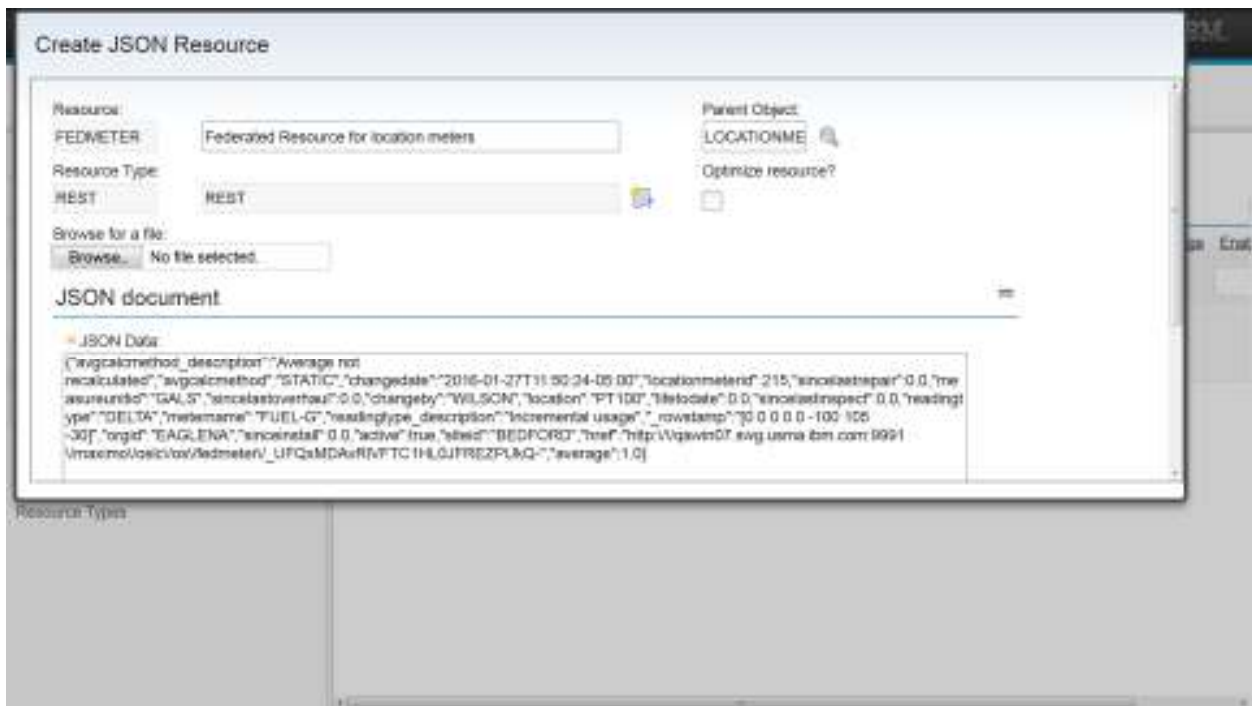
```
"changeby": "WILSON",
"location": "PT100",
"lifetodate": 0.0,
"sincelastinspect": 0.0,
"readingtype": "DELTA",
"metername": "FUEL-G",
"readingtype_description": "Incremental usage",
"_rowstamp": "[0 0 0 0 0 -100 105 -30]",
"orgid": "EAGLENA",
"sinceinstall": 0.0,
"active": true,
"siteid": "BEDFORD",
"href": "http://host:port/maximo/oslc/os/fedmeter/\_UFQxMDAvRIVFTC1HL0JFREZPUkQ-",
"average": 1.0
}
```

This JSON data will be used during the JSON Resource creation. When creating the resource we will use the REST resource type, set the String Qualifier to " and set the Collection Property to *member*, as was done in the prior example. The URL will be set to *http://MX B Host:Port /maximo/oslc/os/fedmeter* and we will update with the bind variables further down.

Below are screen shots of the creation of the JSON resource:



Above shows the collection property and string qualifier being set



Above shows the parent object being LOCATIONMETER, this will drive the creation of a Maximo relationship from LOCATIONMETER to FEDMETER (fed resource mbo).

The next two screens show the object (single object) and the properties information based on the JSON data provided.

Create JSON Resource

Resource: FEDMETER Federated Resource for location meters Optimize resource?

Resource Type: REST

Resource Objects for Resource: 1 - 1 of 1

Relation Property Name	Property Path

Properties For: 1 - 10 of 22

Property Name	Attribute Name	Field Title	Type	Length
▶ _jsonobjectid	P__JSONOBJECTID	P__JSONOBJECTID	ALN	50

Resource Types

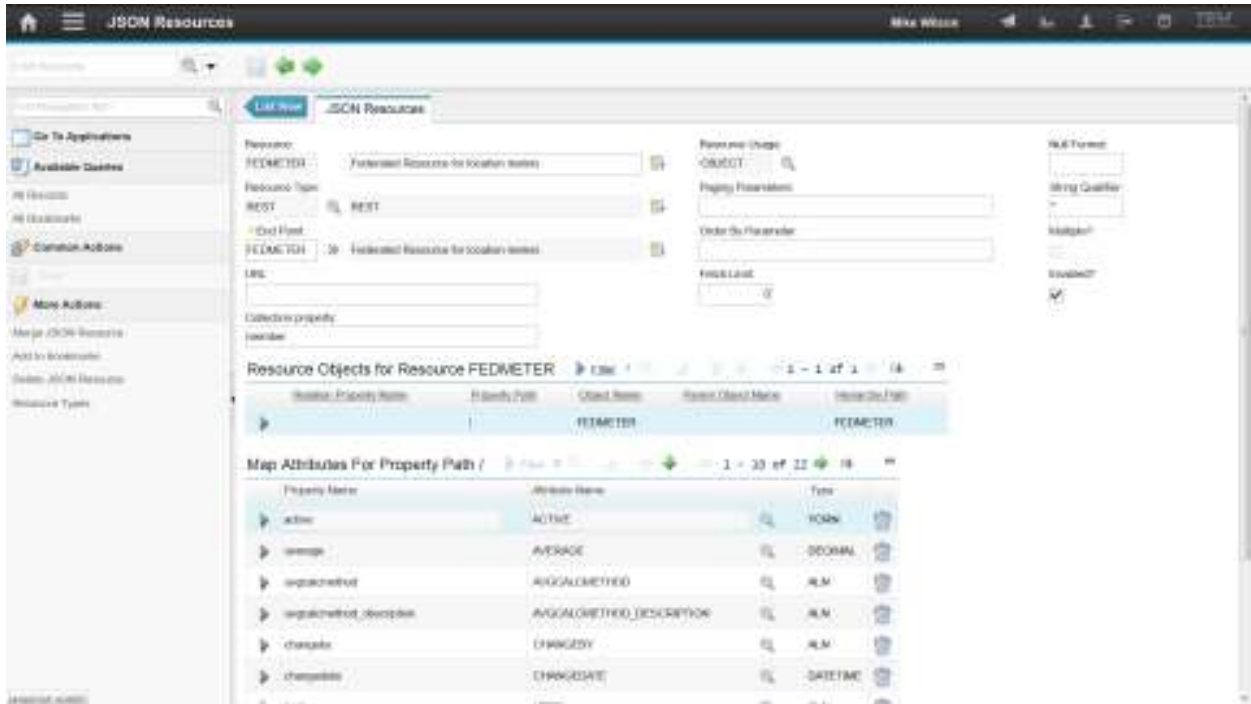
Create JSON Resource

Properties For: 1 - 10 of 22

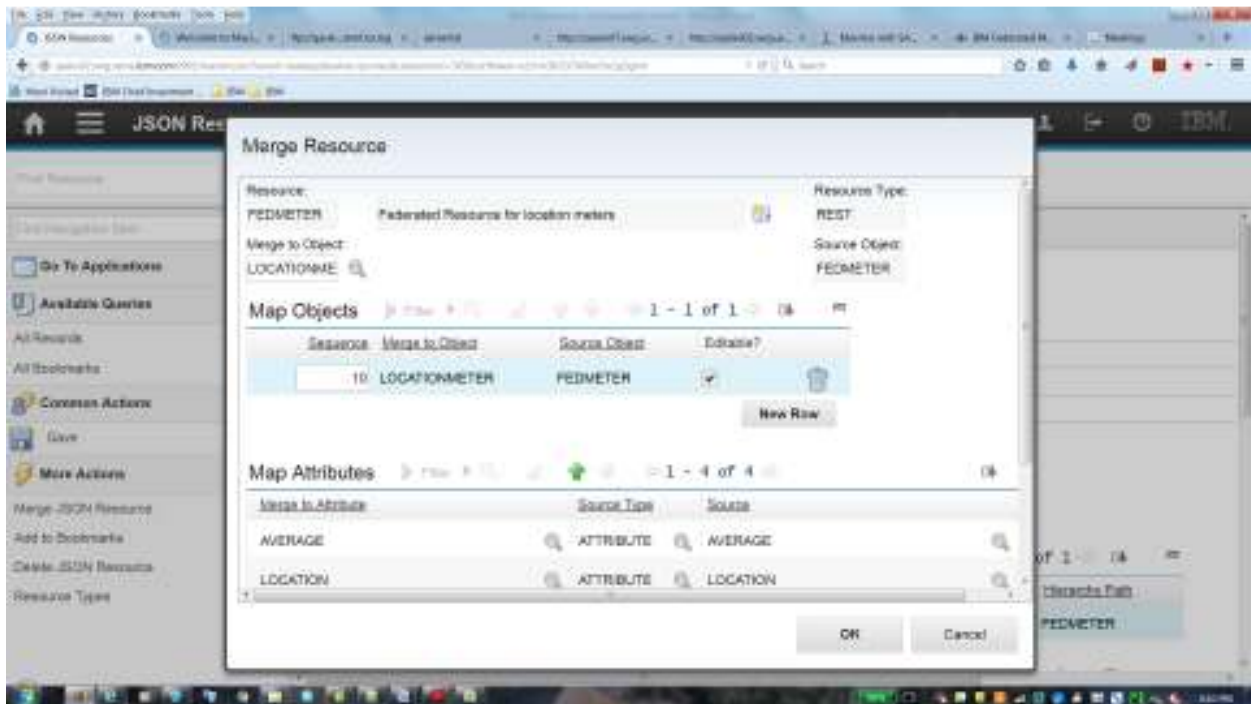
Property Name	Attribute Name	Field Title	Type	Length
▶ _jsonobjectid	P__JSONOBJECTID	P__JSONOBJECTID	ALN	50
▶ avgcalcmethod_description	AVGCALCMETHOD_DESCRIPTION	AVGCALCMETHOD_DESCRIPTION	ALN	50
▶ avgcalcmethod	AVGCALCMETHOD	AVGCALCMETHOD	ALN	50
▶ changedate	CHANGEDATE	CHANGEDATE	DATE TIME	10
▶ locationmeterid	LOCATIONMETERID	LOCATIONMETERID	INTEGER	12
▶ sincelastrepair	SINCELASTREPAIR	SINCELASTREPAIR	DECIMAL	16
▶ measureunitid	MEASUREUNITID	MEASUREUNITID	ALN	50
▶ sincelastoverhaul	SINCELASTOVERHAUL	SINCELASTOVERHAUL	DECIMAL	16
▶

Resource Types

When completed the JSON Resource definition looks like this:



Now that the JSON Resource is created we will select the Merge JSON Resource Action which opens this dialog:



In Map Objects table there is a single row since the federated resource data is from a single object. In the Map attributes section, this is where the properties of the Federated Resource (the Source) are mapped to the attributes of the LOCATIONMETER object (the Merge to Attribute). As an implementer you can choose which source properties map to the merged object (not required to map all).

Note: since this example is using the same underlying table (locationmeter) in two Maximo environments the Source properties and Attributes have the same names by default.

With the Merge action completed, as we did in the prior example we will change the URL on the End Point to include the bind variables to find the correct location using the Federated Resource.

```
http://MX B  
Host:Port/maximo/oslc/os/fedmeter?lean=1&oslc.where=location=attr:location  
and siteid=attr:siteid&oslc.select=*
```

Again we are using the location and siteid as attribute bind variables in the URL with the values coming from the configured parent object, LOCATIONMETER.

In addition to the bind variables there is also an oslc.select clause to select all columns. This is needed as the Maximo JSON API will return on the resource URL by default, so the select is needed to retrieve the columns that being mapped as part of the Merge.

Presentation XML

The plan is to view the Federated Resource data when a user is in the Locations application on the Meters tab, the list of meters displayed will be a combinations of meters stored in LOCATIONMETER of MX A and meters obtained from FEDMETER which are retrieved from MX B. To do this requires a configuration step needed to render the JSON Resource data in an application User Interface (UI) along with the Maximo MBO identified as the 'Merge to Object'. This configuration must be done by manually updating the UI Presentation XML (not available to configure in Application Designer).

Opening the Locations application in the Application Designer application, we exported the presentation XML to a local file. When the presentation XML is opened, we searched for the display of data for the 'Merge to Object' data (LOCATIONMETER). Add a new property called 'federatedresources' where the object is specified.

```
</section>
```

```
<table id="1453816261183" label="Meters" orderby="sequence"  
      relationship="LOCATIONMETER" federatedresources="FEDMETER">  
<tbody displayrowsperpage="16" filterable="true" id="1453816261186">
```

After making the presentation change and importing the presentation xml, when you select a location in the Locations application and navigate to the Meters Tab, the list will be a combination of meters from MX A and MX B.

Federated Resource Concepts

Bind Variables

Earlier in this document, the attribute (attr) bind variable was covered in how it can be used to provide dynamic values to the URL of the Federated Resource. In addition to attr there are 3 other bind variables:

prop - a Maximo system property value

A system property can be used to provide a configurable value to a URL. If you look one of the weather related end points provided with Maximo you will see the apikey parameter populated by a system property (weatherapi.apikey):

<http://api.weather.com/v1/geocode/attr:latitude/attr:longitudex/observations/current.json?apiKey=prop:weatherapi.apikey&language=en-US&units=e>

script - a Maximo script name

A Maximo script, with no launch point, can be provided to derive the value for a URL parameter if a calculation or business logic is required. The output value from the script must be named 'token'.

rel - a Maximo relationship name

The request of data using a Federated Resource would often come through a parent object leveraging a Maximo relationship. To provide flexibility in the assignment of the bind variable value from different fields of a parent object or from different parent objects, the relationship can be used to assign the variable.

For example, let's assume the federated resource was retrieving information based on a GL Account provided by the parent object. Assume the parent object is Asset and it has two GL Account fields, GLCREDITACCT and GLDEBITACCT. In the Maximo Asset application, two tables are created to display Federated GL data for each of these accounts. The URL for the Federated Resource requires a bind variable that contains the GL Account Number. Using the 'attr' bind variable would allow the use of a single field in Asset which would require two federated resources, one for the debit account and one for the credit account.

Using a relationship bind variable will enable one Federated Resource URL make use of two fields from ASSET to provide values. When the UI changes are done in application designer, each table would identify a relationship name, RELCREDIT for the GL Credit Account and

RELDEBIT for the GL Debit Account. As described earlier in this document the table would also identify the federated resource name. The UI Changes would look something like this:

```
<table id="1453816261183" label="GL Account" orderby="sequence"
relationship="RELCREDIT" federatedresources="FEDACCT">
```

The URL for the federated resource would be something similar to this:

```
http://host:port/GLSYSTEM/rel:acct
```

Since the parent object in our example is ASSET, a relationship would be created from ASSET to FEDACCT(federated resource name) with the name RELCREDIT and the 'where clause' would be `acct=glcreditacct`.

From the URL, `rel:` identifies that a relationship bind variable is being used and `'acct'` identifies the variable name that will identify the actual column name (GLCREDITACCT) from the parent object (ASSET). The MBO framework makes use of the relationship name (RELCREDIT) that is provided by the UI presentation layer for the Federated Resource (FEDACCT).

Likewise, for the Debit account, the UI presentation would identify RELDEBIT as the relationship name and it would have a where clause of `acct=gldebitacct`.

This allows one URL:

```
http://host:port/GLSYSTEM/rel:acct
```

to display federated resource data for two GL Account fields in the Asset application.

One additional capability is that if the URL requires multiple bind values, the relationship can specify more than 1 using an `&` separator:

```
acct=glcreditacct&date=changedate&id=assetnum
```

Date Formatting

The Resource (and Resource Type) can identify the date format that an external api will provide, such as ISO8601. Some external REST APIs may not be able to provide a consistent date format for all of the data provided through the API. In cases there are isolated dates that are different than what is defined in on the JSON resource and those values are used in the URL, the format for those can be provided as follows:

```
http://api.weather.com/v1/geocode/attr:latitudey/attr:longitudex/observa
tions/historical.json?apiKey=prop:weatherapi.apikey&units=e&startDat
```

e=attr:&OWNER&.startdate^YYYYMMDD^&endDate=attr:&OWNER&.enddate^YYYYMMDD^

Custom date formats can be assigned during the MBO mapping step as well to convert dates in the response data.

Order By

If there is a need to retrieve the Federated Resource data in a sorted order you can configure the 'Order by' parameter name as part of your resource definition.

For example, if the Federated resource data, represented as a MBO in Maximo, is being displayed in table in a Maximo application, the user may be able to click on a column to direct the MBO to sort the data based on that column. When that occurs, the Federated Resource URL will be re-executed and added to the URL is an 'order by' clause that is formatted using the order by parameter named defined on the JSON Resource definition and the column name selected by the user. If the JSON Resource Order By value is Sort and the column selected is changedate the url parameter would be:

&Sort=changedate asc (or desc)

Where Clause

Following the example used in the Order By (above), if the user is viewing a table of federated resource date and chooses to filter the data by entering a value in a column, the Federated Resource URL will be re-executed and added to the URL will be the oslc.where with the column name and the value entered.

&oslc.where=*fieldname=fieldvalue*

'Out of Box' Resources

Maximo provides 4 'out of the box' JSON Resources that work with weather data from weather.com. A weather-specific resource type is also provided.

- CURWEATHER
- DAILYFORECAST
- HISTWEATHER
- HOURLYFORECAST

These Resources leverage Attribute bind variables that are supported in the SERVICEADDRESS MBO.

Accessing Federated Resource via the Maximo JSON API

In earlier examples we covered cases where a Maximo application, Locations, was viewing data (Location Meters) that came from a Federated Resource. Another use case could be that a new UI application is built outside of the Maximo Framework and it is using the Maximo JSON API to present Maximo data in this application. Along with the Maximo data (locations), this application wants to include external data (location meters) from another application. In this case the same JSON Resource is being used but instead of a Maximo application using that MBO, the data is being access through integration.

There are two options to consider when accessing Federated Resource data via integration, using a Resource with a Usage of API or with a Usage of Object. See the content earlier in this document that covers the Resource Usage during the creation of a Resource to understand the benefits of using either. When the Federated Resource has a Usage of Object then it can be configured as an object in an integration object structure and accessed using the JSON API in the same manner as other Maximo Objects. Then next sections covers how the API can use a Federated Resource with a Usage Type of API.

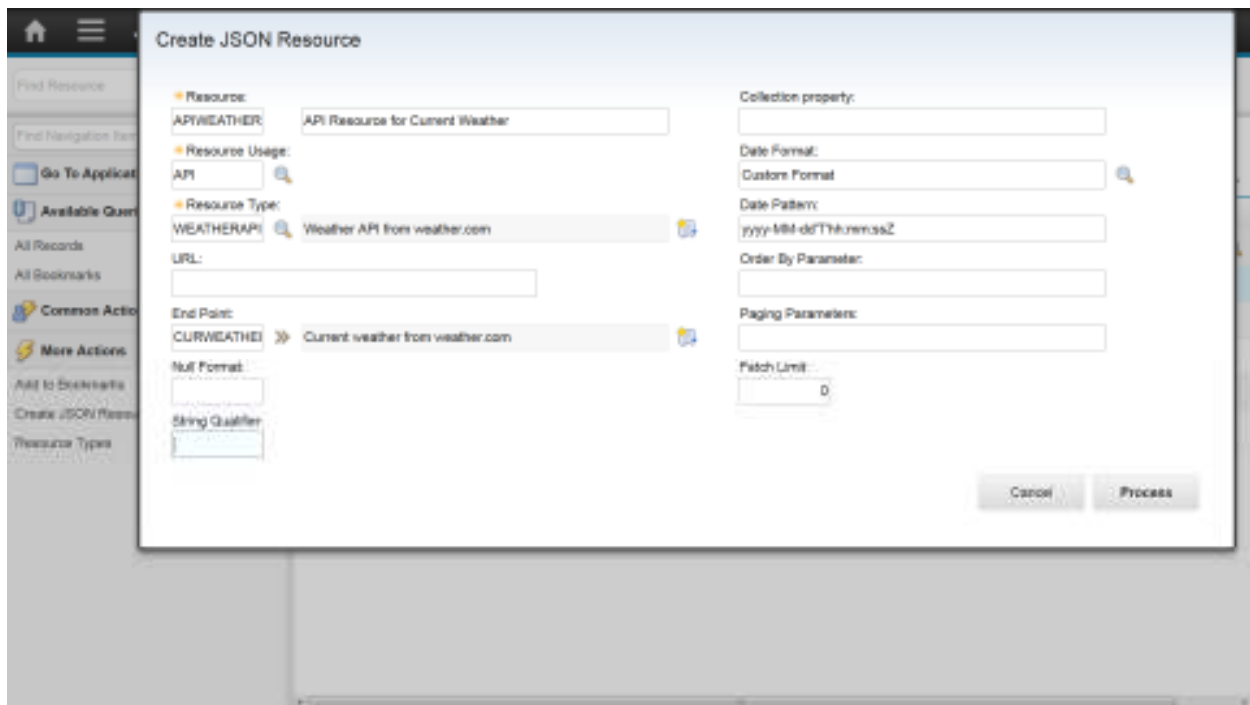
Resource Usage is API

When a JSON Resource is defined with a Usage of API, this means the federated data will not be represented as a MBO and access to that is only supported via the Maximo JSON API. The JSON Resource creation process described earlier in this document is completed after the first dialog is processed. When accessing, the JSON data format is the default format that is provided by the external API.

For the following examples a new Federated Resource, APIWEATHER, was created with a usage of API. The URL from the End Point is as follows:

```
http://api.weather.com/v1/geocode/attr:latitudey/attr:longitudex/observations/current.json?apiKey=prop:weatherapi.apikey&language=en-US&units=e
```

The URL depends on 3 bind variables, two are attributes from the 'parent' object and one is a system property, weatherapi.apikey. The attribute values provide a latitude and longitude for a location from which the api will return the current weather conditions. Below is a screen shot of the resource creation:



Since the resource is not a type Object, there is no MBO representation for APIWEATHER in Maximo so this prevents the use of relationships to identify the parent object. Given that, the parent object and the bind variables it will provide to the Federated Resource will be identified in the JSON API request URL.

In Maximo there is a SERVICEADDRESS MBO that contains latitudey and longitudex attributes thus it can support use of this Federated Resource. Location PT100 has been configured with Service Address 1004 which is located in Boston. This address has a latitude value of 42.364506 and longitude value of -71.038887 which is Boston, MA.

As an external client using the MAXIMO JSON API, there is a need to retrieve location PT100 and display the current weather for the Boston location. To do this requires the following URL:

```
http://host:port/maximo/oslc/os/mxoperloc?lean=1&oslc.select=*,serviceaddress{
city,description,extres. curweather}&oslc.where=location="PT100"
```

The 'out of box' MXOPERLOC object structure was changed to include a new object, SERVICEADDRESS, configured with LOCATIONS as its parent object. Since SERVICEADDRESS is going to provide the values for the bind variables for the URL (.../attr:latitudey/attr:longitudex/...), the reference to the federated resource is done 'within' service address by specifying extres.apiweather within curly brackets { }.

extres. is a reserved word identifying the use of a Federated Resource

apiweather is the name of the Federated Resource

The `oslc.select (&oslc.select=*,serviceaddress{ city,description,extres.curweather})` will retrieve:

- * - all the attributes values for Location PT100 and URLs for the child objects in the `mxoperloc` object structure

`serviceaddress{city,description}` - the city and description values from the `serviceaddress` object related to location PT100

`,extres.curweather}` - all of the values of the `apiweather` federated resource - this data was retrieved using this URL from the **APIWEATHER End Point**

`http://api.weather.com/v1/geocode/attr:latitudey/attr:longitudex/observations/current.json?apiKey=prop:weatherapi.apikey&language=en-US&units=e`

where

`'attr:latitudey'` was replaced with `42.364506` from `SERVICEADDRESS.LATITUDEY`

`'attr:longitudex'` was replaced with `-71.038887` from `SERVICEADDRESS.LONGITUDEX`

`'prop:weatherapi.apikey'` was replaced by `34b54a2413263374bdace07052e0fdf3` from System Property `weatherapi.apikey`

The fully resolved URL:

`http://api.weather.com/v1/geocode/42.364506/-71.038887/observations/current.json?apiKey=34b54a2413263374bdace07052e0fdf3&language=en-US&units=e`

Below is a snippet (not entire response) of the JSON response from the Maximo JSON API

```
{
  "member":
  [
    {
      "changedate": "2016-02-02T13:21:52-05:00",
      "saddresscode": "1004",
      "status_description": "Operating",
      "location": "PT100",
      "serviceaddress":
      [
        {
```

```
"description": "Boston HQ",
"city": "Boston",
"apiweather":
{
  "observation":
  {
    "uv_desc": "Low",
    "phrase_32char": "Fair",
    "imperial":
    {
      "precip_2day": 0.0,
      "temp_min_24hour": 38,
      "snow_24hour": 0.0,
      "precip_ytd": 3.35,
      "snow_ytd": 10.6,
      "snow_3day": 0.0,
      "rh": 32,
      "hi": 49,
      "dewpt": 20,
      "temp_change_24hour": -10,
      "ceiling": null,
      "snow_mtd": 0.0,
      "temp": 49,
    }
  }
}
```

The response data includes values from LOCATIONS, the city and description from SERVICEADDRESS and the values from APIWEATHER. The APIWEATHER data is in the format provided by the external API (not reformatted to the Maximo native format).