# Maximo JSON API Comparison

V1 - Updated: 04/26/2016

## Contents

## *Introduction*

In Maximo 7.6.0.2, a new JSON (Javascript Object Notation) API has evolved from the OSLC REST API (released 7.5.0.3), providing similar capabilities along with some Usability improvements. This API can operate on existing INTEGRATION (and OSLC) object structures and does not require the configuration of an OSLC Resource. The API does not support the use of OSLC namespaces and common properties, thus providing clean JSON data format. The API does support some of the OSLC standards around querying, such as the query parameters oslc.select and oslc.where.

## *Overview*

The history of the MIF's support for a RESTful API began in release 7.1.x with the REST api and then the OSLC API was added in 7.5.x. Now in 7.6.x the MIF's support for REST has evolved to the JSON API.

At this time, you should expect that any REST API improvements/enhancements that are provided in future versions of Maximo will be in the JSON API, not in the REST or OSLC APIs.

With this progression, this document will hi-light the differences between the JSON API and the two other APIs.

## *Maximo JSON API and Maximo REST API*

The Maximo REST API was the first version of a RESTful API provided in 7.1.x. The API supported both XML and JSON in the response content and supported Form data or query parameters in order to create or update resources. The API did not provide support for JSON data in the request to create or update resources. The API has undergone limited enhancements since its initial release and it lacks many of the features that were provided in the OSLC and JSON APIs.

Below is a list that hi-lights the primary differences between the JSON and REST APIs:

1. The REST api does support both Object Structures and MBOs as Resources while the JSON API supports Object Structures

2. The JSON API supports the concept of using a Saved Query to retrieve resource data, the REST API does not

3. The JSON API supports JSON data in a request to create or update a resource, the REST API does not

4. The JSON API supports JSON data in the response while the REST api supports either JSON or XML data

5. The JSON API provides metadata for the apis and supports a JSON schema for each resource, the REST API has no support for either.

6. The JSON API provides support for 'bulk' processing that handles multiple resources (which could have different actions) in a single transaction, the REST does not support this.

## *Maximo JSON API and OSLC REST API*

The OSLC API was provided in 7.5.x and was part of the work to support Maximo OSLC enablement for both the Provider and Consumer roles. As an OSLC Provider, Maximo enabled the OSLC REST API to 'provide' Maximo resource data to any OSLC Consumer application. One part of that API enablement was the support of common namespaces and properties that are used within OSLC. To provide this support required the introduction of the OSLC Resources application where an implementer could select an object structure and assign OSLC-specific content.

One drawback of this API was the need to do additional configuration in Maximo of an OSLC Object Structure and the OSLC Resource definition. Additionally, some users who did not want to leverage the OSLC namespaces and properties were forced to use them. These and other reasons led Maximo to provide a new RESTful JSON API that was based on the OSLC API, however it removed the need to use namespaces and properties and provided a clean JSON data format. As well, it is enabled to work with 'out of box' INTEGRATION object structures, such as MXASSET and MXWO, eliminating the need to configure additional integration components.

Below is a list that hi-lights the primary differences between the two APIs:

1. The JSON API is based on INTEGRATION Object structures and does not require OSLC resource definition. Although it will work with an OSLC Resource, that configuration is unnecessary and does not add any value.

2. The JSON API does not promote the use of the OSLC namespace and common property names.

3. The OSLC API provided a 'map' to reference linked attributes, this is replaced in the JSON API by 'dot notation attributes' and related objects. An example is shown below

    i. */oslc/os/mxasset?oslc.select=assetnum,**location.description,rel.openwo{wonum,description}***

    ii. *You can link other MBOs (like location and workorders) from the asset resource - dynamically in the URL - without predefining a map of this data*

4. The OSLC API supported the concept of search terms which allowed users to configure "searchable" attributes at design time and then provide search

tokens at request time. This process has been replaced in favor of providing the "searchable" attributes at request time along with the search tokens - sample below

iii. */oslc/os/mxasset?oslc.searchTerms="hello","hi"&searchAttributes=description,assetnum*

5. In the JSON API, Attachments (doclinks) are support at any level in the object structure, not just the root object as it is for the OSLC API.

6. The JSON API maintained support of query parameters of the OSLC standard such as oslc.where, oslc.select and oslc.orderBy

7. The JSON API provides support for 'bulk' processing that handles multiple resources (which could have different actions) in a single transaction, the OSLC API does not support this.

Please send any corrections or suggestions to Tom Sarasin at tsarasin@us.ibm.com