Using JSON in the DataMapper

JSON is everywhere. The data format is almost identical to XML, but it is less verbose and therefore more suited for HTTP communications. Yet if you want to extract data from it, you always have to convert it to XML first because the DataMapper doesn't know how to handle JSON natively. Until Version 2020.2, that is...

The JSON format

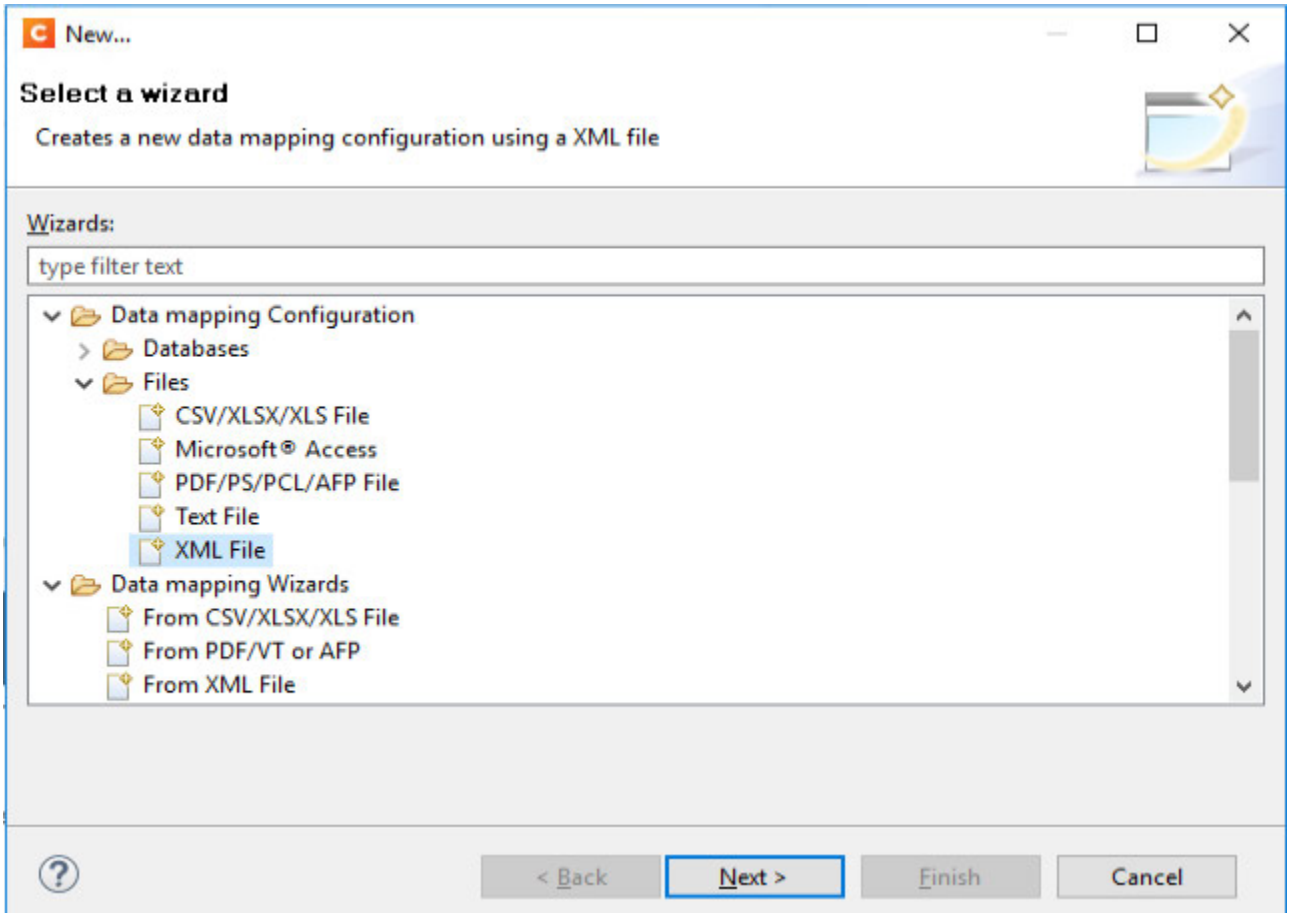
JSON is just a text representation of data as key/value pairs, just like XML is. But since it doesn't require end-tags as XML does, it requires less memory and has therefore become the format of choice for exchanging data between systems via the web, where sending large packets of information can impact overall performance of a system.

With HTTP communications quickly becoming one of the most popular methods for sending data to Connect, the DataMapper needed to have the ability to parse the content of JSON files without you having to convert the file to XML first through some external process (usually, using Workflow's XML/JSON conversion task). That meant each time you received a new JSON file, you had to go through the conversion process before you could import the XML file into the DataMapper in order to start designing your DM Config.

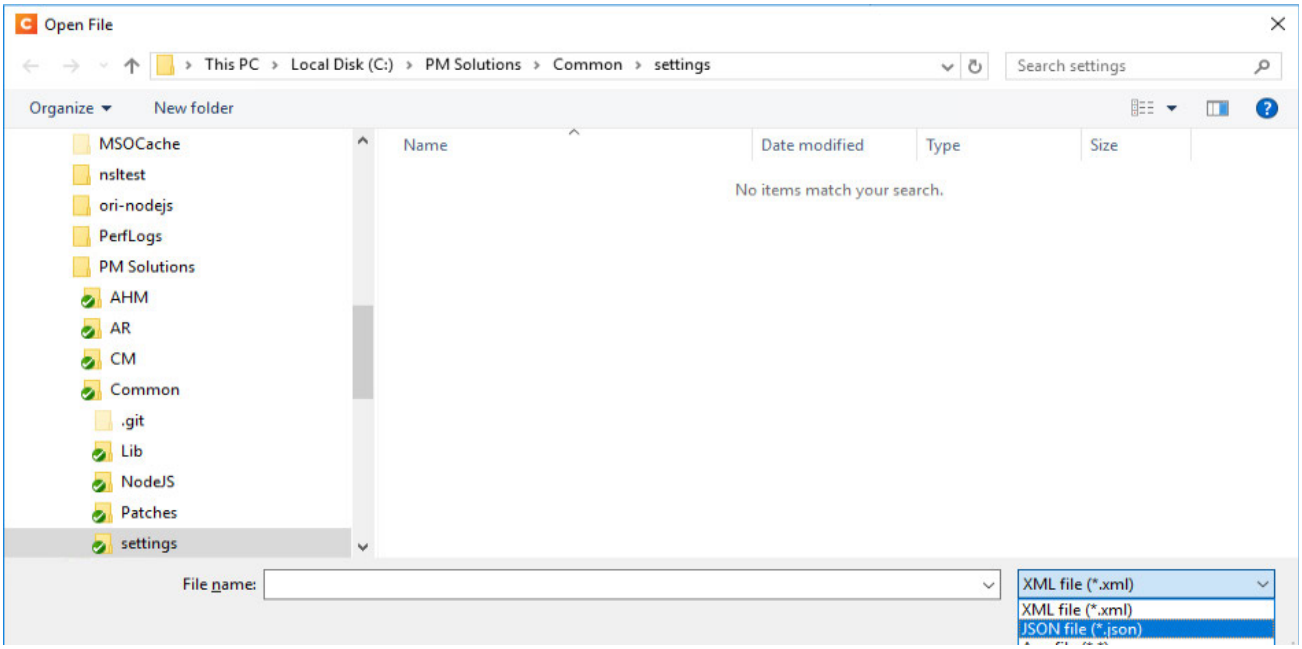
And then there was 2020.2...

With Version 2020.2, JSON is treated as *another form of XML*. In other words, you are still designing an XML data mapping config, but using JSON as the source data instead of an actual XML file. The DataMapper internally converts the JSON file to XML and presents you with the same User Interface as the one displayed with XML files.

So the first step is to create a new XML data mapping config:



In the file type drop-down list, select the new JSON option:



The file gets converted on the fly and is displayed as if it were a native XML file. Here is what it looks like in the DataMapper UI:

Converted JSON file

Compare that with the original JSON file:

```
[{
        "OrderNumber": "INV1867748",
        "CustomerPO": "411350",
        "CustomerNumber": "CU63047838",
        "SalesRep": "Branden Croot",
        "Date": "2017-06-10",
        "ContactFirstName": "Dionysus",
        "ContactLastName": "Moiser",
        "ContactPhone": "1-(150)873-0623",
        "ContactEmail": "dionysus.moiser@example.com",
        "Company": "Twitterlist",
        "StreetAddress": "188 Drewry Street",
        "City": "Raymond",
        "State": "AB",
        "Country": "Canada",
        "ZipCode": "L5A 5D2",
        "DeliveryDate": "2017-06-14",
        "DeliveryTeam": {
            "Name": "Charmine Garter",
            "ID": "732",
            "DateTime": "2017-06-14",
            "Minutes": "0"
        },
        "COTGUser": "Charmine.Garter",
        "Products": [{
                "Number": "404947",
                "Description": "Tubbs\u00c2\u00a0Wilderness Snowshoes",
                "UnitPrice": "149.96",
                "Ordered": "4",
                "Shipped": "4",
                "BackOrder": "0",
                "Total": "599.84"
            }, {
                "Number": "389316",
                "Description": "Hestra\u00c2\u00a0RSL Comp Vertical Cut Mens Ski Racing Gloves XL",
                "UnitPrice": "119.77",
                "Ordered": "1",
                "Shipped": "1",
                "BackOrder": "0",
                "Total": "119.77"
            }, {
```


Original JSON file

You'll notice there are some differences between the two displays. That's because even though JSON and XML are very similar, they are not strictly identical and can therefore not be converted exactly as is. For instance, JSON doesn't require named arrays or objects, whereas XML does.

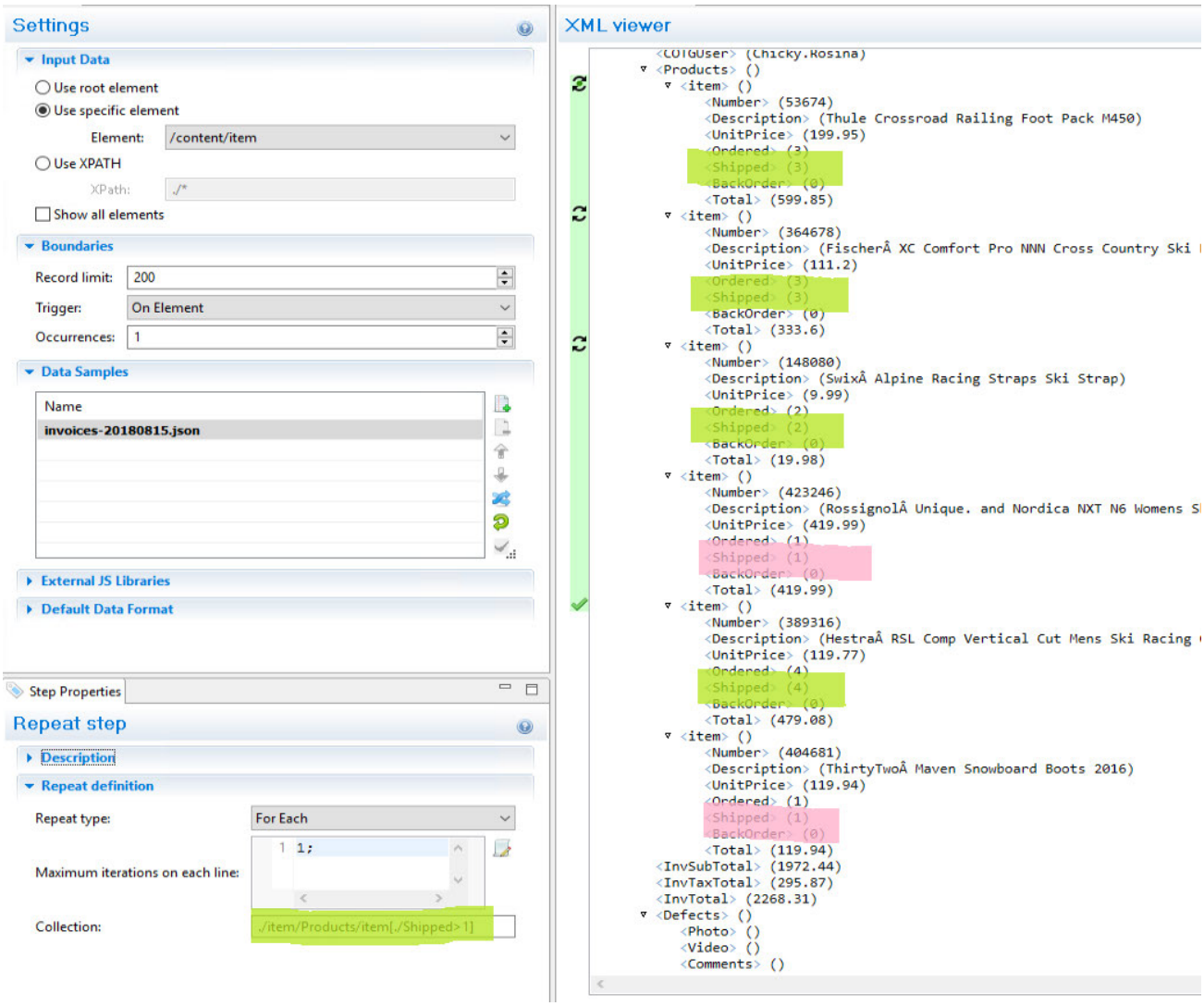
The conversion process takes care of generating valid XML out of the JSON by inserting appropriate, generic tag names whenever they're not provided in the JSON file. So just like with any other data format, as long as the JSON file's hierarchical structure doesn't change, the conversion process will always produce the same results, allowing you to re-use the same DM config for other JSON files of the same type.

Note: at the time of this writing, the JSON conversion feature was still being tested and tweaked so the results you see with the Beta or with the Official release of 2020.2 may be slightly different.

A familiar environment

One advantage of having the JSON file automatically converted to XML is that it allows you to work in an environment you're familiar with. You work with JSON just like you do with XML. And that includes the use of the XPATH language, which allows you to perform tasks that would otherwise be difficult with a pure JSON file.

For instance, in the following example, a Repeat step has been created to loop on all invoice line items whose *Shipped* value is greater than one, skipping the others. This is achieved through the use of the XPATH filtering command `./item/Products/item[./Shipped>1]`:



Using XPATH in this manner makes it far easier to filter out elements than having to go through the entire list and inserting conditions to see if elements meet certain criteria.

Future improvements

This feature is only the beginning for supporting JSON inside the DataMapper. Our R&D teams are already working on the next steps, which will allow for native support without any conversion. Since we still want to benefit from the XPATH syntax, we are looking into implemen ng JSONPath, which is a trimmed-down version of XPATH. But whatever we do, we will s ll keep the current implementa on because of its ease of use and XPATH support.