SAP Graph Workshop

Johannes Rupieper, Chaim Bendelac, SAP Graph

V1.0 – April 2022

The purpose of this workshop is to explore a small programming exercise involving the use of SAP Graph in an interaction with SAP data from multiple data sources. The exercise illustrates the simplicity, powerful expressiveness, and ease of use of the SAP Graph API.

The use case

Your company sells many products to its corporate customers. Your boss is interested in understanding how his sales team positions baskets of the most popular products (by quantity) at your largest customers. She asks you to quickly create a small app to explore this relationship.

What you will do:

Create a small app that

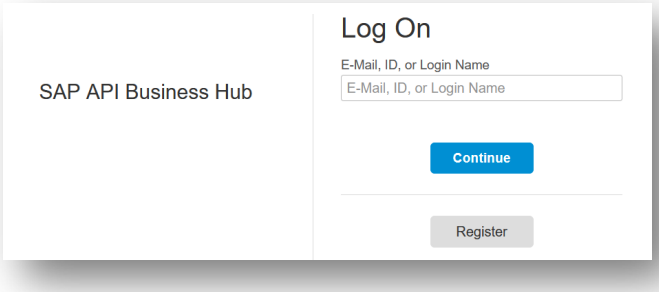
- lists the most significant quotes, generated by your sales team, showing euro-amount and customer-name
- provides deep-dive analysis of the product quantities offered to a selected customer
- allows to look up the details of a specific customer

The app will navigate between sales data, customer (master) data, and product details. In complex landscapes, these data sets may reside in different data sources, such as SAP Sales Cloud and SAP S/4HANA.

Prerequisites

As a developer, you will start development from a small boiler-plate web application, which you will download, unzip and run directly from your workstation (PC or Mac). Here are the specific prerequisites:

1. ***Mandatory***: Ability to open a downloaded zip file. ***Optional***: use of Git.
2. ***Mandatory***: modern browser like Chrome. ***Recommended***: installed JSON formatter such as "JSONView".
3. ***Required***: some familiarity with the OData protocol which we will use during the workshop. We recommend reading this short blog: https://blogs.sap.com/2021/07/28/part-4-the-sap-graph-data-protocol/
4. ***Advantage***: experience with JSON structures and some familiarity with JavaScript.
5. **Mandatory**: you will need your own "API Key" from SAP's API Business Hub. Login, or register if this is your first visit, to obtain an API Key, in advance of the workshop:



Preparations and explanations (10-15 minutes)

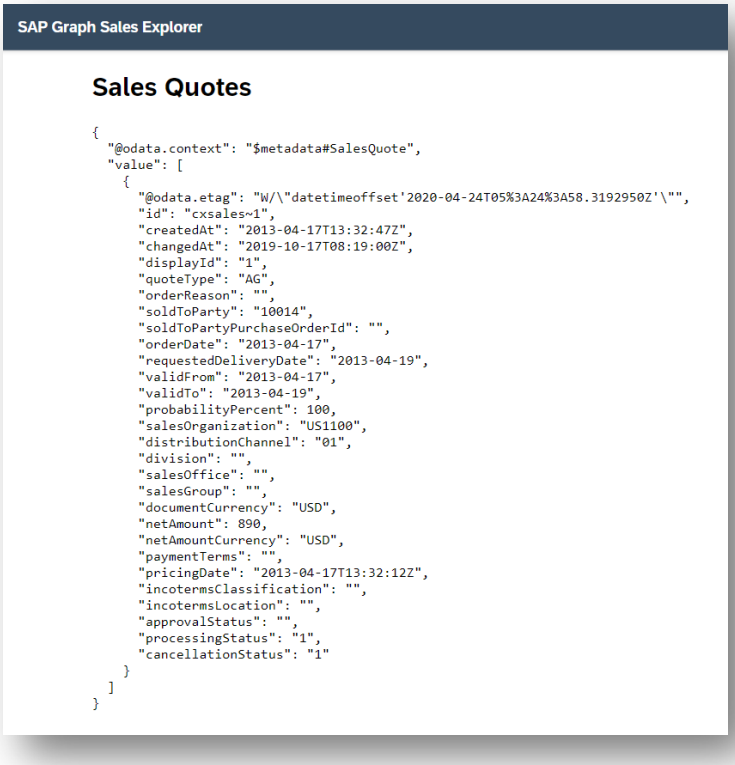
1. Save the downloaded zip-file to a folder on your workstation.

2. Navigate to this folder, and edit the file **`main.js`**
   This file contains the main "boilerplate" function that we will use throughout the workshop, called `getFromGraph()`.
   Take a minute to familiarize with this function: it simply takes one parameter, the SAP Graph query string, to fetch data from SAP Graph. For demo purposes, we use an "API key" for authentication with a "sandbox" landscape. In "a productive" applications, the code of this function is a little bit more complex, in order to handle oAuth-based authentication.

3. Carefully replace the string
   ```
   insert your API Key here
   ```
   With your own API key.

4. Save the file.

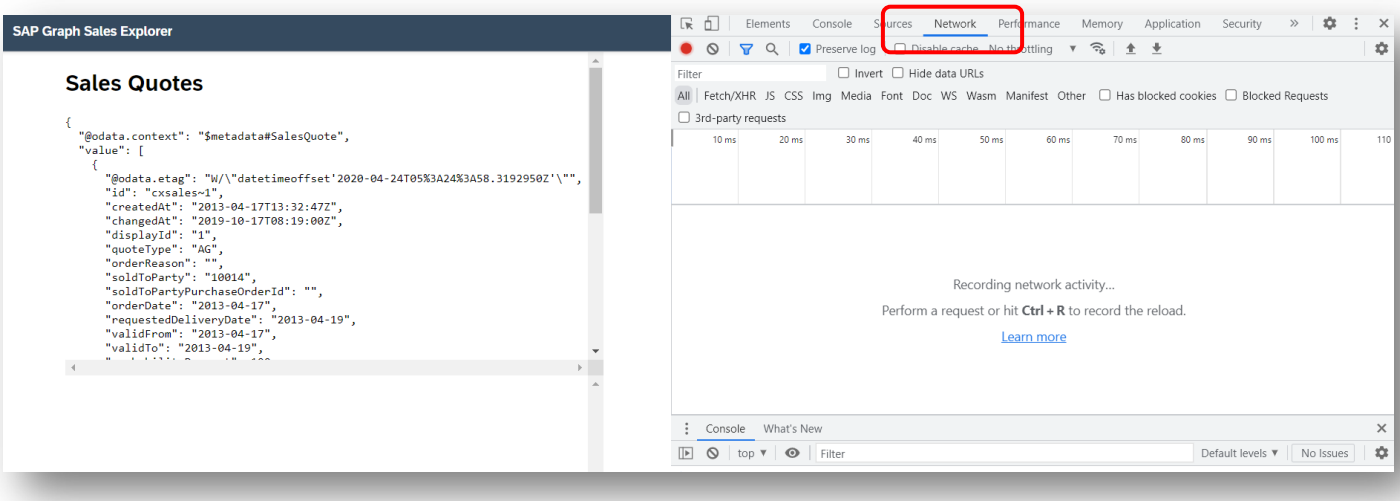
5. Load the file **`index.html`** in your browser (on a PC: right-click, *open-with Google Chrome*, for instance). If all went well, you should now see:



We are ready for our first task.

Task 1 – order all quotes by decreasing amount, list customer name and quote date (20 minutes)

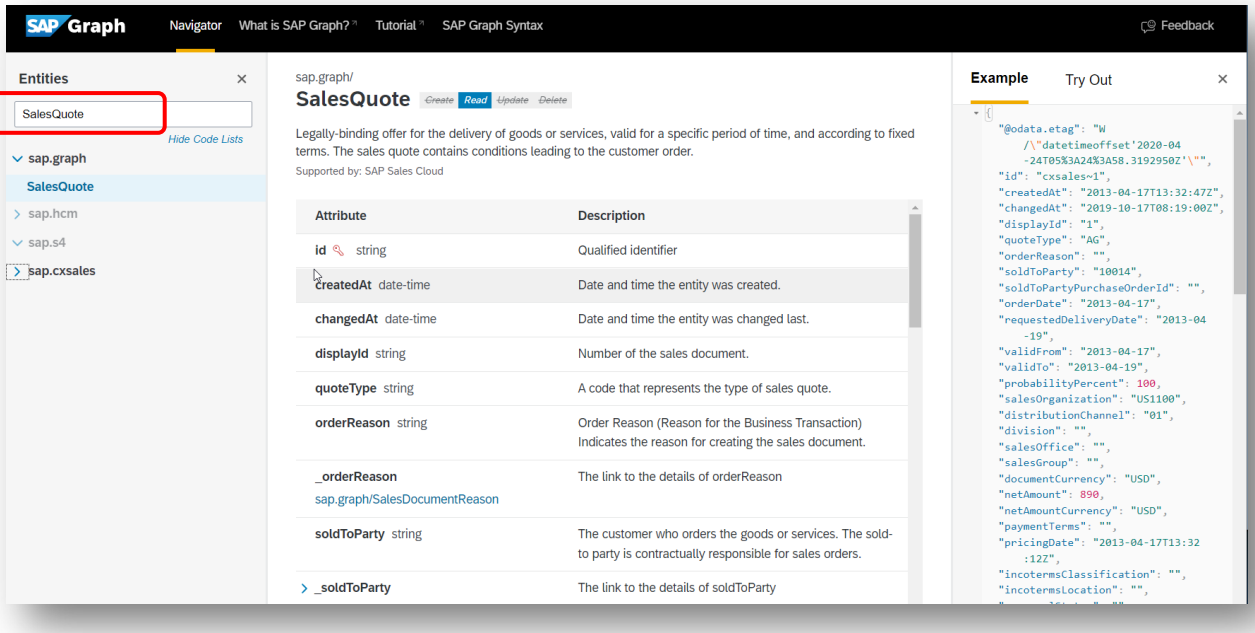
To help you debugging, it is useful to open the developer tools of your browser, side-by-side. On a PC, you can do this with `CTRL+SHIFT+I`, and then select the "Network" tab. Your screen should look like this:



1. Now open the **`SalesQuotes.js`** file in your editor. Go ahead and change the value of **`topValue`** on line 2 from "1" to "2". Save the file, refresh the page in the browser. You should now see two quotes, which are also nicely captured on the network tab of the developer tools.

2. Your first task is to improve and shape the OData query starting on line 11, using combinations of selection (`$select`), expansion (`$expand`) and ordering (`$orderby`) the quotes.

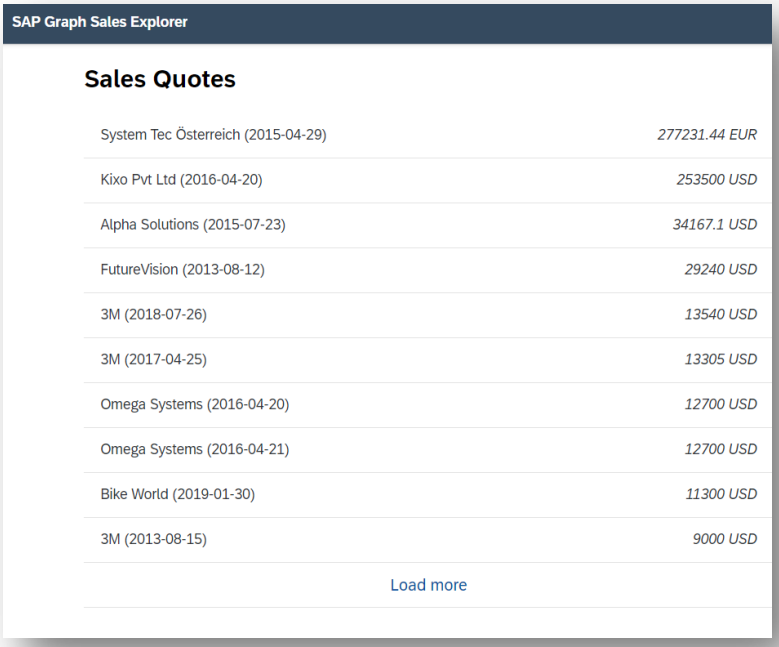
   To help you understand which fields can be expanded and selected, take a look at the SAP Graph Navigator at https://navigator.graph.sap/entities . For instance, search for SalesQuote:



Your goal is to end up with a query whose (first) returned result looks like this:

Note that this is the quote with the largest **`netAmount`** in the system.

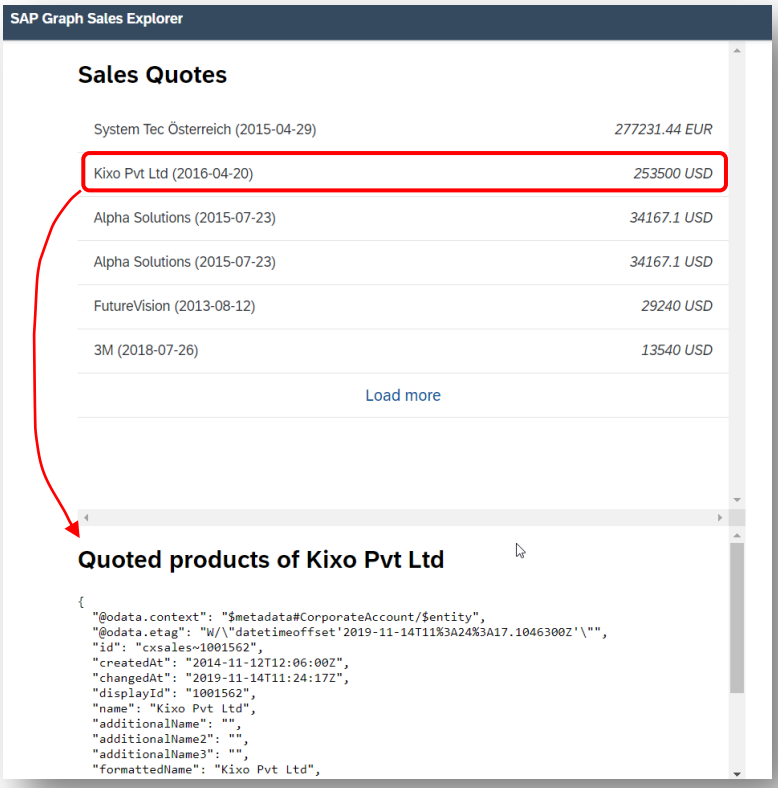
3. Once you have achieved this result, set the value of "**debug1**" on line 1 to **false** in order to replace the dump with a more elegant listing of customer quotes. Refresh to get the desired outcome of task 1:



Let's wait for everybody to catch up, before we continue.

Task 2: fetch all the quotes of the selected customer and display aggregated quantities (20 minutes)

The rows of the displayed table are clickable. Select and click one of them, you will see:



This is the result of the code in file **`CorporateAccount.js`**. We want to do a bit better.

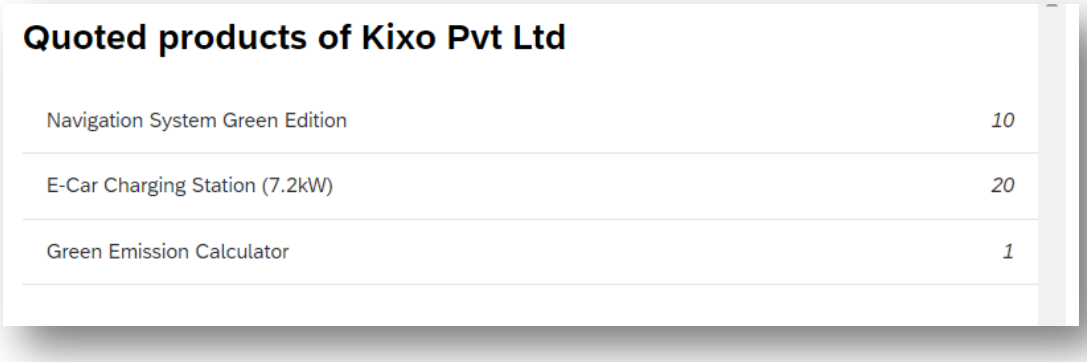
Task 2 is to identify all the quotes in the system for the specific customer and sum up all quantities of all products quoted to this customer. Here is the desired result. A list of all the quotes for that customer, with expanded item information (we randomly picked "Kixo"):

Quoted products of Kixo Pvt Ltd


```
{
  "@odata.context": "$metadata#CorporateAccount(name,_salesQuotes(requestedDeliveryDate,items(product,itemText,quantity)))/$entity",
  "@odata.etag": "W/\"datetimeoffset'2019-11-14T11%3A24%3A17.1046300Z'\"",
  "id": "cxsales~1001562",
  "name": "Kixo Pvt Ltd",
  "_salesQuotes": [
    {
      "requestedDeliveryDate": "2016-04-22",
      "items": [
        {
          "itemText": "Navigation System Green Edition",
          "product": "P100311",
          "quantity": 10
        },
        {
          "itemText": "E-Car Charging Station (7.2kW)",
          "product": "P300103",
          "quantity": 20
        }
      ]
    },
    {
      "requestedDeliveryDate": "2020-02-15",
      "items": [
        {
          "itemText": "Green Emission Calculator",
```

Again, this is achieved by expanding and selecting only the interesting details.

Once you have achieved a result that looks like the above, change the value of **debug2** to false, and refresh the browser:



You can play with other customers, like Alpha Solutions, to see how it all comes together. Finally, click on one of the product rows, to see a dump of the product information, and examine the code of **`product.js`** to see how it was created.

Congratulations!

We will wait for everybody to catch up, and then summarize, and provide an opportunity for Q&A.