MatDeck: Access to Chemical Databases - Mcul

In this example, we illustrate how to access the chemical database Mcul that has more than 44 millions compounds directly from a MatDeck document. The MatDeck script is used to prepare a query in json format, which is sent using the http post method. The result is also obtained in json format and displayed in the document both as textual and as graphical results. The supported query types of Mcul are currently: SMILES, InChIKey; Molfile or SDF string (V2000), IUPAC name, CAS number, Std InChI, ZINC ID and CHEMBL ID.

Preparing a query

Mcul database provides the http api, where queries are prepared in json format. The json queries are sent using the http post method from a MatDeck script. In a MatDeck document, the easiest way to prepare a json query is to use a text box widget as illustrated in examples below. However, the use of the text box widget is not supported within a script file. There are two additional ways to prepare json files in MatDeck Script; these will be illustrated in other examples.

Exact Search

Exact searches allow you to specify multiple queries. Here is a MatDeck Script code which is used to specify three different queries using a SMILES query type. It is recommended to use SMILES queries and to send queries in the SMILES format. An example of a SMILES string is:

C(N1C(C(OC2CCCN(C3=CC=CC=C3)C2)=O)CN(C)C1=O)(=O)C(=C)CN1CCOCC1

The idx value in the response corresponds to the index of the query in the queries list. As you can see from the above response, the first and the third query resulted in a hit, while the second query did not give any hits.

The query in json format is prepared using a text box widget. The text box is defined using line 1, and its size is set using line 2. The text box widget is embedded within a canvas, where it can be filled with an appropriate json query. The text box in the same canvas contains an example of a multiple exact json query.

```
1 json := static_text_box(0, 650, 150, "Arial", 10)
```


```
{
   "query": {
      "type": "exact",
      "queries": [
         "C(N1C(C(OC2CCCN(C3=CC=CC=C3)C2)=O)CN(C)C1=O)(=O)C(=C)CN1CCOCC1",
         "O(C1=CN(N=C1)C)CC1C=CC2C=CC=CC=2C=1",
         "C(N1C(C(OC2CCCN(C3=CC=CC=C3)C2)=O)CN(C)C1=O)(=O)C1CC2=CC=CC(OC)=C2SC1"
      ]
   }
```


Sending http post method

Next, we prepare the http post request with the json query prepared above. We have to defined the url, and

application type. The response obtained from the Mcul server is assigned to the variable called response as a string type.

```
url := "https://ultimateapp.mcule.com/api/v1/searches/"
response := http_post(url, "application/json", widget_value(json))
if(is_vector(response))
      response = vec2str(response)
```

Displaying results as image and text

The response from the server is in json format. We can display the original response in the text box in the canvas.

```
json_r := text_box(0, "")
set_size(json_r, 600, 200)
set_widget_text(json_r, response)
```

However, we are interested in the SMILES string and molecule drawing from SMILES. Therefore, we extract the SMILES from json using a matrix conversion and json_value function. The variable count represents the number of obtained compounds in the response.

```
mat_json := json2mat(response)
count := json_value(mat_json, "count")
smiles := vector_create(count, false, "")
for(n := 0; n < count; n += 1)
{
      smiles[n]= json_value(json_value(json_value(mat_json, "results")[n],
"compound"), "smiles")
}
```

Next, we use SMILES to generate a molecule images.

```
image1 := image_widget(0, 0)
mol1 := get_sm_image(smiles[0], "")
set_widget_value(image1, mol1)
set_size(image1, width(image1) *4, height(image1) *4)
image2 := image_widget(0, 0)
mol2 := get_sm_image(smiles[1], "")
set_widget_value(image2, mol2)
set_size(image2, width(image2) *4, height(image2) *4)
```

We can display the obtained SMILES strings separately in a canvas.

```
smiles[0] = "C(N1C(C(OC2CCCN(C3=CC=CC=C3)C2)=O)CN(C)C1=O)(=O)C(=C)CN1CCOCC1"
smiles[1] = "C(N1C(C(OC2CCCN(C3=CC=CC=C3)C2)=O)CN(C)C1=O)(=O)C1CC2=CC=CC(OC)=C2SC1"
```

Finally, here is completes json file obtained as response.


```
{"results":[{"compound":{"smiles":"C(N1C(C(OC2CCCN(C3=CC=CC=C3)C2)=O)CN(C)C1=O)
(=O)C(=C)CN1CCOCC1","inchi_key":"IHHITLNORGKBTP-UHFFFAOYSA-
N","archived_at":null},"query":"C(N1C(C(OC2CCCN(C3=CC=CC=C3)C2)=O)CN(C)C1=O)(=O)C(=C)CN1CCOCC1","idx":
1},{"compound":{"smiles":"C(N1C(C(OC2CCCN(C3=CC=CC=C3)C2)=O)CN(C)C1=O)
(=O)C1CC2=CC=CC(OC)=C2SC1","inchi_key":"CTQJGCXDDNMNMJ-UHFFFAOYSA-
N","archived_at":null},"query":"C(N1C(C(OC2CCCN(C3=CC=CC=C3)C2)=O)CN(C)C1=O)
(=O)C1CC2=CC=CC(OC)=C2SC1","idx":3}],"count":2}
```


```
get_sm_image(sm, title)
{
      if(!is_string(sm) || !is_string(title) || strlen(sm) == 0)
            return(void)

      url := "https://www.simolecule.com/cdkdepict/depict/bow/svg?smi="
      url += sm // code
      url += "%20"
      url += title // title
      url += "&abbr=on&hdisp=bridgehead&showtitle=true&zoom=1.6&annotate=none"

      return(http_get(url))
}
```