WorldFAIR Chemistry: Protocol Services

WorldFAIR Chemistry

2023-10-07

Interactive Demonstration

This notebook is intended as an interactive demonstration of the services being proposed by the IUPAC WorldFAIR Chemistry D3.3 project team. A complete description of the project is available at https://iupac.github.io/WFChemProtocols/intro.html.

This notebook is an RMarkdown version of the original Jupyter Notebook, which is available at https://github.com/IUPAC/WFChemProtocols/blob/main/IUPACProtocolsDemo.ipynb

Resolver Summary

While more detail is provided in the documentation linked above, in short what is described here is a web service called a "resolver" that performs two main functions:

1. Check for the presence of a chemical record in the hosting organization's database.
2. Validate the machine-readable chemical structure according to the hosting organization's rules.

Resolver Base URL

The service being proposed in this project is a regular HTTP web service, using standard CGI URL syntax, and a well-defined data model for the information returned. This demonstration uses a prototype service hosted by PubChem, using JSON as the response format (although in principle it could be XML or any other structured data format).

One key point of this proposal is that the base URL for the resolver CGI would vary from one institution to another, but the inputs (CGI arguments) and outputs (JSON data) would be standard, the same for any organization implementing the service. So simply by switching the base URL, one can run the same query on multiple different sites, without otherwise needing to change any code.

In R, this could look like this:

```
library(httr)
library(rjson)
resolver_base_url <- "https://pubchem.ncbi.nlm.nih.gov/resolver/resolver.cgi"
```

When called without any arguments, the resolver will return some information about what inputs and outputs it can handle.

```
url <- resolver_base_url
res <- GET(URLencode(url))
url_data <- httr::content(res, type="text", encoding="UTF-8")
```

```
# display the results
url
```

```
## [1] "https://pubchem.ncbi.nlm.nih.gov/resolver/resolver.cgi"
```

```
cat(url_data)
```

```
## {
##   "Result": {
##     "ServiceDetails": [
##       {
##         "Resource": "PubChem",
##         "ResourceURL": "https://pubchem.ncbi.nlm.nih.gov",
##         "ResolverURL": "https://pubchem.ncbi.nlm.nih.gov/resolver/resolver.cgi",
##         "AvailableInputs": {
##           "SDF": true,
##           "SMILES": true,
##           "InChI": true,
##           "InChIKey": true,
##           "PNG": false,
##           "Name": true
##         },
##         "AvailableOutputs": {
##           "IUPACName": true,
##           "SMILES": true,
##           "InChI": true,
##           "InChIKey": true,
##           "ResourceIdentifier": true,
##           "RecordURL": true,
##           "ImageURL": true
##         }
##       }
##     ]
##   }
## }
```

Chemical Lookup

The resolver service can check to see whether a given chemical is present in the host organization's database. Examples are below, but note that in this interactive document, one can edit the inputs to query whatever chemical is desired.

First, to look up by SMILES string:

```
query_type <- "smiles"
query <- "CCCC"
url <- paste0(resolver_base_url,"?",query_type,"=",query)
# get results
res <- GET(URLencode(url))
url_data <- httr::content(res, type="text", encoding="UTF-8")

# display URL and results
url
```

```
## [1] "https://pubchem.ncbi.nlm.nih.gov/resolver/resolver.cgi?smiles=CCCC"
```

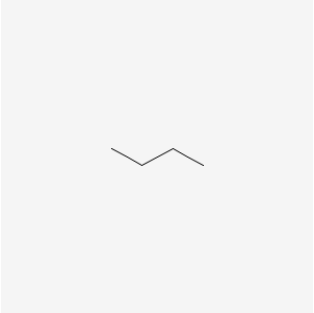
```
cat(url_data)
```

```
## {
##   "Result": {
##     "Match": [
##       {
##         "Resource": "PubChem",
##         "ResourceURL": "https://pubchem.ncbi.nlm.nih.gov",
##         "ResourceIdentifier": "7843",
##         "ResourceIdentifierType": "CID",
##         "RecordURL": "https://pubchem.ncbi.nlm.nih.gov/compound/7843",
##         "ImageURL": "https://pubchem.ncbi.nlm.nih.gov/image/imgsrv.fcgi?t=l&cid=7843",
##         "IUPACName": "butane",
##         "SMILES": "CCCC",
##         "InChI": "InChI=1S/C4H10/c1-3-4-2/h3-4H2,1-2H3",
##         "InChIKey": "IJDNQMDRQITEOD-UHFFFAOYSA-N"
##       }
##     ]
##   }
## }
```

In this example code, the URL is first constructed using the `query` and `query_type` inputs, then encoded and retrieved using the `httr` package. The resulting data indicates that there is indeed a matching record in the host's database, and various record fields are provided that would allow the user to get more information directly from the hosting site; this is not intended for full record retrieval, but rather a simplified response that says whether the chemical is found and where to go to get more detail. So in this case the user can follow the link to the full PubChem record:

https://pubchem.ncbi.nlm.nih.gov/compound/7843

Or see an image of the chemical structure (although not terribly interesting in this case!):



https://pubchem.ncbi.nlm.nih.gov/image/imgsrv.fcgi?t=l&cid=7843

If the chemical is not in the database, the response would be something like this, where an empty result means nothing was found (this could also potentially be indicated by an HTTP 404 response, but is not done that way in this sample implementation):

```
query_type <- "smiles"
query <- "CCCC(Br)CC(F)(Cl)CCC"
url <- paste0(resolver_base_url,"?",query_type,"=",query)
# get results
res <- GET(URLencode(url))
url_data <- httr::content(res, type="text", encoding="UTF-8")

# display URL and results
```

```
url
```

```
## [1] "https://pubchem.ncbi.nlm.nih.gov/resolver/resolver.cgi?smiles=CCCC(Br)CC(F)(Cl)CCC"
```

```
cat(url_data)
```

```
## {
##   "Result": {
##   }
## }
```

The resolver can handle multiple input formats for the chemical structure, as listed in the previous section. So all of these would return the same result, which can be verified by (un)commenting various query lines below:

```
# # SMILES
# query_type <- "smiles"
# query <- "CCCC"

# InChI
query_type <- "inchi"
query <- "InChI=1S/C4H10/c1-3-4-2/h3-4H2,1-2H3"

# # InChIKey
# query_type <- "inchikey"
# query <- "IJDNQMDRQITEOD-UHFFFAOYSA-N"

# # Name
# query_type <- "name"
# query <- "butane"


url <- paste0(resolver_base_url,"?",query_type,"=",query)
# get results
res <- GET(URLencode(url))
url_data <- httr::content(res, type="text", encoding="UTF-8")

# display URL and results
url
```

```
## [1] "https://pubchem.ncbi.nlm.nih.gov/resolver/resolver.cgi?inchi=InChI=1S/C4H10/c1-3-4-2/h3-4H2,1-2
```

```
cat(url_data)
```

```
## {
##   "Result": {
##     "Match": [
##       {
##         "Resource": "PubChem",
##         "ResourceURL": "https://pubchem.ncbi.nlm.nih.gov",
##         "ResourceIdentifier": "7843",
##         "ResourceIdentifierType": "CID",
##         "RecordURL": "https://pubchem.ncbi.nlm.nih.gov/compound/7843",
##         "ImageURL": "https://pubchem.ncbi.nlm.nih.gov/image/imgsrv.fcgi?t=l&cid=7843",
##         "IUPACName": "butane",
##         "SMILES": "CCCC",
##         "InChI": "InChI=1S/C4H10/c1-3-4-2/h3-4H2,1-2H3",
##         "InChIKey": "IJDNQMDRQITEOD-UHFFFAOYSA-N"
```

```
##       }
##     ]
##   }
## }
```

Note that the InChI full string needs to be URL-encoded in order to be passed as an argument to the CGI, as would some SMILES strings with special characters. This is handled using `URLencode` in these examples.

Chemical Structure Validation

TBC...