Methods in Pharmacology and Toxicology
DOI 10.1007/7653_2018_30

Programmatic Retrieval of Small Molecule Information from PubChem Using PUG-REST

Sunghwan Kim, Paul A. Thiessen, and Evan E. Bolton

Abstract

PubChem (https://pubchem.ncbi.nlm.nih.gov) is an open archive which contains information on small molecules as well as other chemical entities such as lipids, carbohydrates, and (chemically modified) amino acid and nucleic acid sequences (including siRNA and miRNA). Developed and maintained by the US National Institutes of Health, PubChem is a chemical information hub, collecting chemical information from various data sources and disseminating it to the public free of charge. PubChem provides multiple programmatic access routes, including E-Utilities, Power User Gateway (PUG), PUG-SOAP, and PUG-REST. This chapter describes how to access PubChem programmatically through PUG-REST. The syntax of the PUG-REST request URL is explained with many examples that cover various tasks and a series of Perl scripts are provided to demonstrate how these URLs can be included in actual programs.

Keywords Cheminformatics, E-Utilities, Programmatic access, PubChem, PUG-REST, Representational state transfer (REST)

1 Introduction

PubChem (https://pubchem.ncbi.nlm.nih.gov) [1–3] is a public chemical information resource developed and maintained by the National Center for Biotechnology Information (NCBI) at the National Library of Medicine (NLM), an institute within the US National Institutes of Health. It provides information on small molecules and other chemical entities including small molecules, lipids, carbohydrates, and (chemically modified) amino acid and nucleic acid sequences. Since its launch in 2004, PubChem has been an important resource for biomedical research communities in many areas including cheminformatics, chemical biology, medicinal chemistry, and drug discovery. More details about PubChem are described elsewhere [1–3], and only a brief overview is given in the present article.

Electronic supplementary material: The online version of this chapter (https://doi.org/10.1007/7653_2018_30) contains supplementary material, which is available to authorized users.

PubChem consists of three inter-linked primary databases: Substance, Compound, and BioAssay. The Substance database contains chemical substance descriptions provided by more than 600 data contributors. Unique chemical structures are extracted from the Substance database through the standardization process and stored in the Compound database. The BioAssay database contains descriptions and results of biological assay experiments. Unique numeric identifiers called SID (Substance ID), CID (Compound ID), and AID (Assay ID) are assigned to individual records in the Substance, Compound, and BioAssay databases, respectively.

Currently, PubChem contains more than 254 million substance descriptions, 95 million unique chemical structures, and 236 million biological activities from over one million biological assays covering 10,000 protein target sequences (as of June 2018). This vast amount of data presents new opportunities to the scientific community in the age of "big data." The application of these data for biomedical research often requires programmatic access to them.

PubChem provides multiple programmatic access routes [4–6]. NCBI's Entrez Utilities (also known as E-Utilities or E-Utils) [7] are used to access information contained in the Entrez system [8], which is a primary text search and retrieval system integrating PubChem and other NCBI's databases. While E-Utilities are appropriate to access text- or numeric-fielded data, it cannot handle complex data types specific to PubChem, such as chemical structures and tabular bioactivity data. Therefore, PubChem developed the Power User Gateway (PUG), which provides programmatic access specialized for PubChem data and analysis services. Later, PubChem introduced two easier-to-use web service access methods: PUG-SOAP, which uses the simple object access protocol (SOAP) [9], and PUG-REST, which is a Representational State Transfer (REST)-style interface [10, 11]. An overview of these methods is given elsewhere [4].

The present chapter describes how to access PubChem data through PUG-REST with various examples. The syntax of PUG-REST request URL is explained with many examples that cover various tasks. In addition, a series of Perl scripts are provided to demonstrate how these URLs can be included in actual programs (*see* Electronic Supplementary Material, Data 1). A more detailed description of PUG-REST is given in our previous papers [4, 5] as well as in the specification document [12] and the tutorial-style document [13]. Because PUG-REST is a continuously evolving service, users are advised to check these two documents regularly for any updates.

2 URL Syntax of PUG-REST

The conceptual framework of the PUG-REST service is illustrated in Fig. 1. A PUG-REST request requires three pieces of information: (1) one or more input identifiers that represent records in PubChem's three primary databases (i.e., CIDs, SIDs, and AIDs), (2) what operation the PubChem server should do with the input identifiers, and (3) the desired format of the output that is returned to the user. PUG-REST encodes these three pieces of information into a simple one-line uniform resource locator (URL), which consists of the input, operation, and output parts, preceded by a prefix common to all PUG-REST requests. Some PUG-REST requests require additional information on operation-specific options. This information can be provided as URL arguments after the "?" mark at the end of the URL path. Therefore, the general syntax of the PUG-REST request is following:

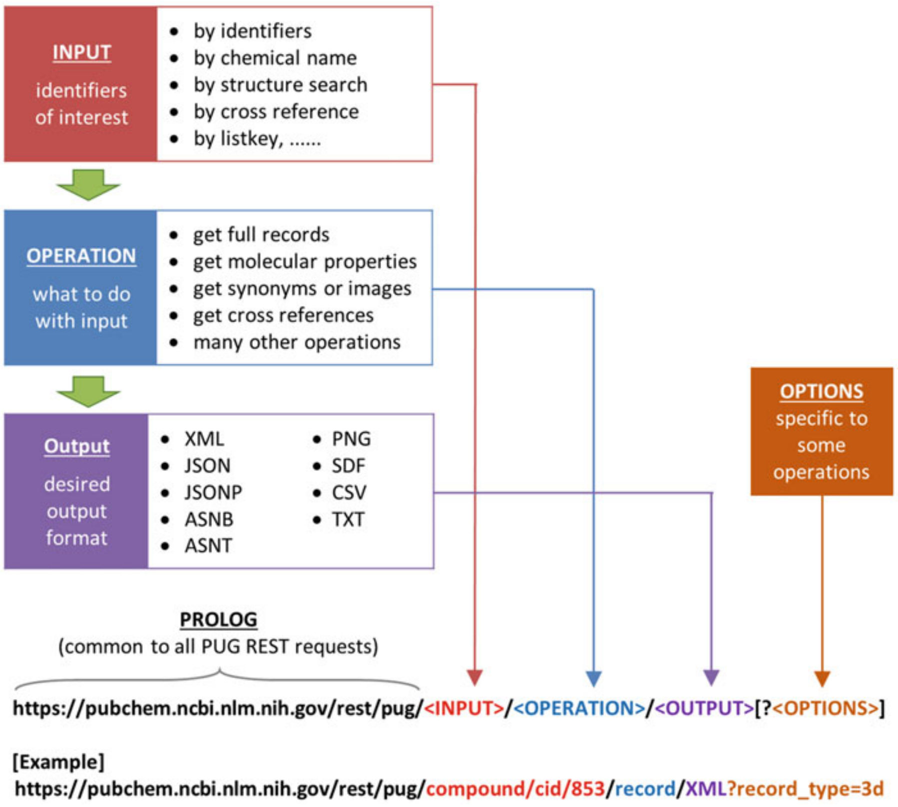
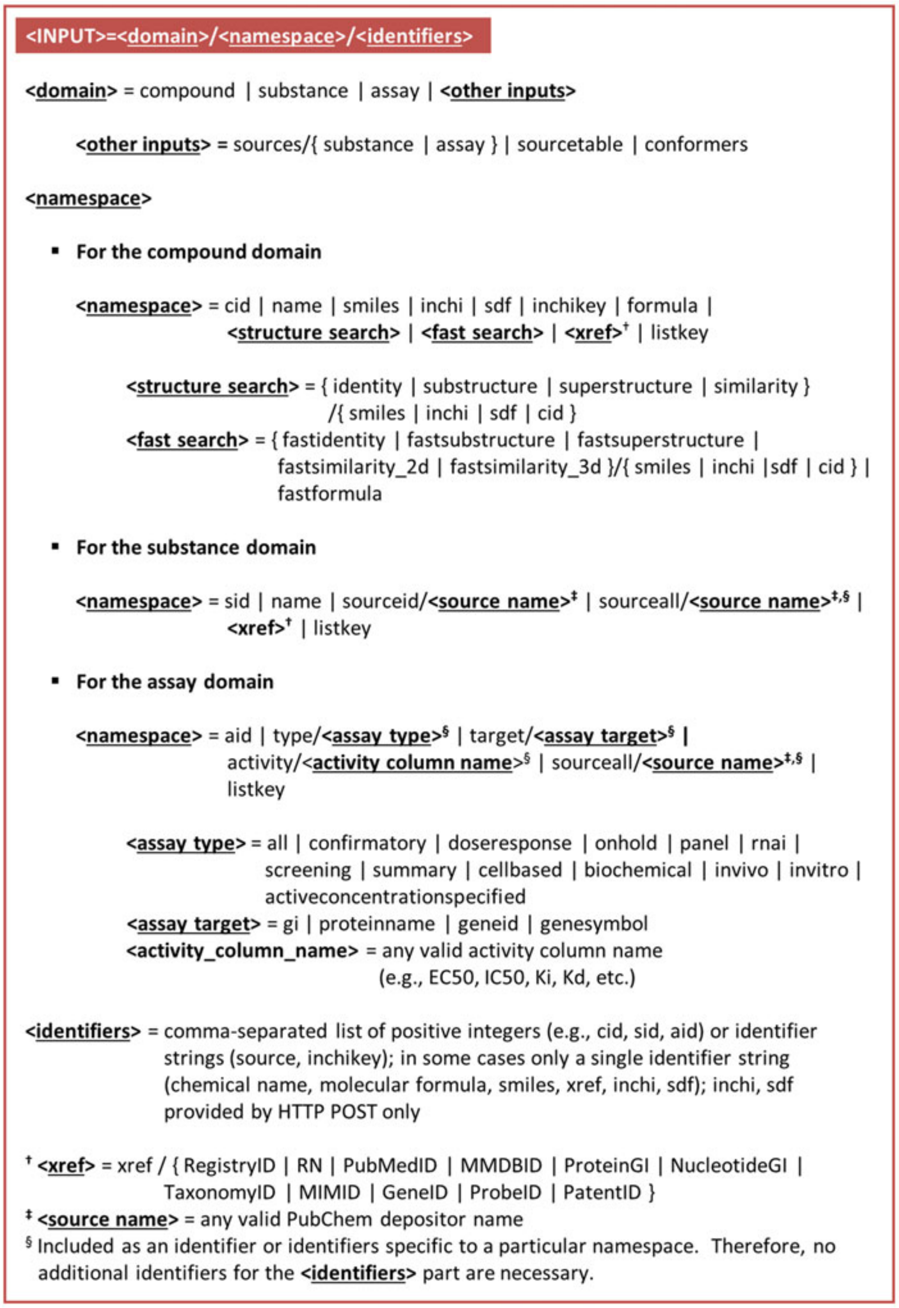


Fig. 1 Conceptual framework and URL construction for PUG-REST requests

https://pubchem.ncbi.nlm.nih.gov/rest/pug/<INPUT>/<OPERATION>/<OUTPUT>[?OPTIONS]

Figures 2 and 3 show the syntax of input, operation, and output parts of PUG-REST URL paths, and many examples are listed in Figs. 4, 5, and 6 (to be explained in the next section).



<INPUT>=<domain>/<namespace>/<identifiers>

<domain> = compound | substance | assay | **<other inputs>**

<other inputs> = sources/{ substance | assay } | sourcetable | conformers

<namespace>

- **For the compound domain**

<namespace> = cid | name | smiles | inchi | sdf | inchikey | formula | **<structure search>** | **<fast search>** | **<xref>**† | listkey

<structure search> = { identity | substructure | superstructure | similarity } /{ smiles | inchi | sdf | cid }

<fast search> = { fastidentity | fastsubstructure | fastsuperstructure | fastsimilarity_2d | fastsimilarity_3d }/{ smiles | inchi |sdf | cid } | fastformula

- **For the substance domain**

<namespace> = sid | name | sourceid/**<source name>**‡ | sourceall/**<source name>**‡,§ | **<xref>**† | listkey

- **For the assay domain**

<namespace> = aid | type/**<assay type>**§ | target/**<assay target>**§ | activity/**<activity column name>**§ | sourceall/**<source name>**‡,§ | listkey

<assay type> = all | confirmatory | doseresponse | onhold | panel | rnai | screening | summary | cellbased | biochemical | invivo | invitro | activeconcentrationspecified

<assay target> = gi | proteinname | geneid | genesymbol

<activity_column_name> = any valid activity column name (e.g., EC50, IC50, Ki, Kd, etc.)

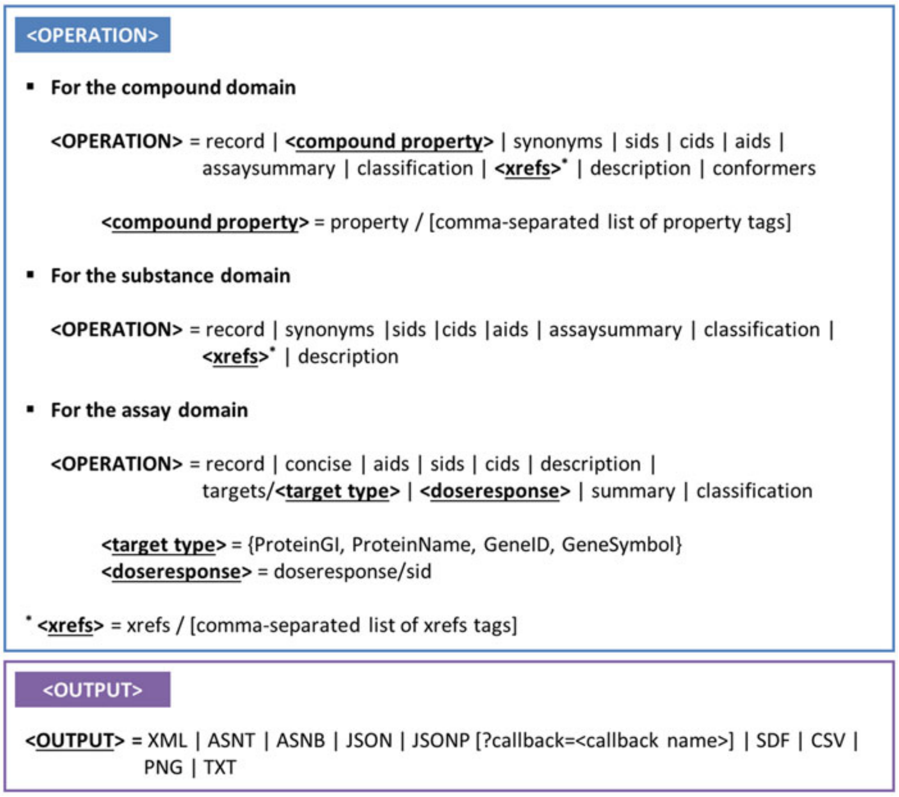
<identifiers> = comma-separated list of positive integers (e.g., cid, sid, aid) or identifier strings (source, inchikey); in some cases only a single identifier string (chemical name, molecular formula, smiles, xref, inchi, sdf); inchi, sdf provided by HTTP POST only

† **<xref>** = xref / { RegistryID | RN | PubMedID | MMDBID | ProteinGI | NucleotideGI | TaxonomyID | MIMID | GeneID | ProbeID | PatentID }

‡ **<source name>** = any valid PubChem depositor name

§ Included as an identifier or identifiers specific to a particular namespace. Therefore, no additional identifiers for the **<identifiers>** part are necessary.

Fig. 2 Syntax for the input part of PUG-REST URL paths



<OPERATION>

- **For the compound domain**

  <OPERATION> = record | **<compound property>** | synonyms | sids | cids | aids | assaysummary | classification | **<xrefs>*** | description | conformers

  <compound property> = property / [comma-separated list of property tags]

- **For the substance domain**

  <OPERATION> = record | synonyms |sids |cids |aids | assaysummary | classification | **<xrefs>*** | description

- **For the assay domain**

  <OPERATION> = record | concise | aids | sids | cids | description | targets/**<target type>** | **<doseresponse>** | summary | classification

  <target type> = {ProteinGI, ProteinName, GeneID, GeneSymbol}
  <doseresponse> = doseresponse/sid

* **<xrefs>** = xrefs / [comma-separated list of xrefs tags]

<OUTPUT>

<OUTPUT> = XML | ASNT | ASNB | JSON | JSONP [?callback=<callback name>] | SDF | CSV | PNG | TXT

Fig. 3 Syntax for the operation and output parts of PUG-REST URL paths

Some types of input are not compatible with the PUG-REST URL syntax. An Example is any chemical names or representations that contain special characters reserved for the PUG-REST URL syntax (such as "/" (forward slash)). In addition, a structure-data file (SDF), which has multiple lines, cannot be put into a single-line URL path. These inputs can be handled using the HTTP POST method, which is described later in this chapter.

3 URL Examples for PUG-REST Requests

3.1 Full Record Retrieval

As shown in **Examples A1** through **A7** in Fig. 4, full records for PubChem substances, compounds, and assays can be retrieved by setting the operation part to "record." The supported output formats are ASNT/B, XML, JSON(P), CSV (for assays only), and SDF (for compounds and substances only). Full record retrieval is the default action unless the operation is specified.

By default, full record retrieval of compounds and substances includes two-dimensional (2-D) structure information. However, three-dimensional (3-D) structure information may also be

(PROLOG=https://pubchem.ncbi.nlm.nih.gov/rest/pug)

A. Full record retrieval

A1. PROLOG/substance/sid/822166/record/XML
A2. PROLOG/substance/sourceid/DTP.NCI/36345,715005/record/SDF
A3. PROLOG/compound/cid/3672/record/XML
A4. PROLOG/compound/name/ibuprofen/record/SDF?record_type=3d
A5. PROLOG/assay/aid/714,1068/record/XML
A6. PROLOG/assay/aid/714/record/CSV
A7. PROLOG/assay/aid/714/record/CSV?sid=842627,842919,843300

B. Chemical image retrieval

B1. PROLOG/substance/sid/2096/record/PNG?image_size=small
B2. PROLOG/compound/smiles/CCOC(=O)N/record/PNG?image_size=large&record_type=3d
B3. PROLOG/compound/inchikey/IKQCSJBQLWJEPU-UHFFFAOYSA-M
/record/PNG?image_size=250x200

C. Chemical name retrieval

C1. PROLOG/substance/sid/611158/synonyms/TXT
C2. PROLOG/compound/cid/3672/synonyms/TXT
C3. PROLOG/compound/cid/1983,3672/synonyms/XML
C4. PROLOG/compound/cid/1983,3672/description/XML

D. Compound property retrieval

D1. PROLOG/compound/cid/2200,2725 /property/MolecularFormula,MolecularWeight/XML
D2. PROLOG/compound/cid/2319,3827,4034/property/XLogP/TXT

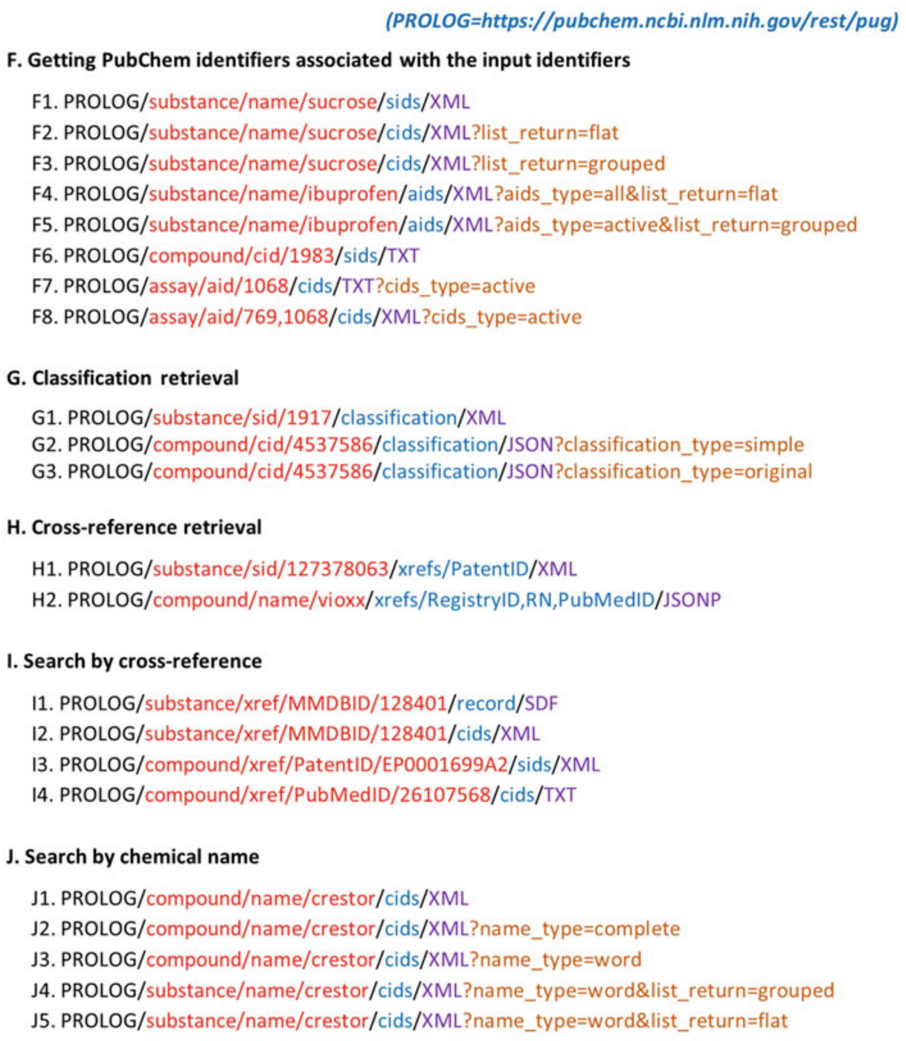
E. Assay data retrieval

E1. PROLOG/assay/aid/490/description/XML
E2. PROLOG/assay/aid/1000/summary/XML
E3. PROLOG/assay/aid/490,1000/targets/ProteinGI,ProteinName,GeneID,GeneSymbol/XML
E4. PROLOG/assay/aid/504526/doseresponse/XML
E5. PROLOG/assay/aid/504526/doseresponse/CSV?sid=104169547,109967232
E6. PROLOG/compound/cid/1000,1001/assaysummary/CSV
E7. PROLOG/substance/sid/104234342/assaysummary/XML

Fig. 4 PUG-REST URL examples A1 through E7

retrieved by specifying the "record_type" option to "3d" (**Example A4**). For a compound record, the first diverse conformer computationally generated by PubChem3D [14–16] is retrieved, while a substance record is returned with the 3-D coordinates deposited by the data contributor, which may be experimentally determined or computationally derived.

Assay record retrieval is limited to 10,000 SIDs per assay at a time; one may retrieve an assay record with more than 10,000 SIDs by dividing the SID list into smaller chunks and getting the data for



(PROLOG=https://pubchem.ncbi.nlm.nih.gov/rest/pug)

F. Getting PubChem identifiers associated with the input identifiers

```
F1. PROLOG/substance/name/sucrose/sids/XML
F2. PROLOG/substance/name/sucrose/cids/XML?list_return=flat
F3. PROLOG/substance/name/sucrose/cids/XML?list_return=grouped
F4. PROLOG/substance/name/ibuprofen/aids/XML?aids_type=all&list_return=flat
F5. PROLOG/substance/name/ibuprofen/aids/XML?aids_type=active&list_return=grouped
F6. PROLOG/compound/cid/1983/sids/TXT
F7. PROLOG/assay/aid/1068/cids/TXT?cids_type=active
F8. PROLOG/assay/aid/769,1068/cids/XML?cids_type=active
```

G. Classification retrieval

```
G1. PROLOG/substance/sid/1917/classification/XML
G2. PROLOG/compound/cid/4537586/classification/JSON?classification_type=simple
G3. PROLOG/compound/cid/4537586/classification/JSON?classification_type=original
```

H. Cross-reference retrieval

```
H1. PROLOG/substance/sid/127378063/xrefs/PatentID/XML
H2. PROLOG/compound/name/vioxx/xrefs/RegistryID,RN,PubMedID/JSONP
```

I. Search by cross-reference

```
I1. PROLOG/substance/xref/MMDBID/128401/record/SDF
I2. PROLOG/substance/xref/MMDBID/128401/cids/XML
I3. PROLOG/compound/xref/PatentID/EP0001699A2/sids/XML
I4. PROLOG/compound/xref/PubMedID/26107568/cids/TXT
```

J. Search by chemical name

```
J1. PROLOG/compound/name/crestor/cids/XML
J2. PROLOG/compound/name/crestor/cids/XML?name_type=complete
J3. PROLOG/compound/name/crestor/cids/XML?name_type=word
J4. PROLOG/substance/name/crestor/cids/XML?name_type=word&list_return=grouped
J5. PROLOG/substance/name/crestor/cids/XML?name_type=word&list_return=flat
```

Fig. 5 PUG-REST URL examples F1 through J5

each chunk, as shown in **Example A7**, in which a subset of the SIDs of an assay is specified as an option.

A full record retrieval for multiple compounds, substances, or assays may be requested at once (**Example A5**). However, because there is a timeout limit (30 s) on a PUG-REST request, it is not practical to retrieve a very long list of records. In such cases, one should use PubChem's bulk download facilities [17, 18] or File Transfer Protocol (FTP) site [19].

3.2 Image Retrieval

When the output format for full record retrieval of a compound is given as "PNG" (**Examples B1** through **B3**), the 2-D or 3-D

(PROLOG=https://pubchem.ncbi.nlm.nih.gov/rest/pug)

K. Simple identity search (exact match)

```
K1. PROLOG/compound/smiles/CC(=O)NC(CC1=CNC2=CC=CC=C21)C(=O)O/cids/TXT
```

L. Identity search using various contexts of structural identity

```
L1. PROLOG/compound/fastidentity/smiles
    /CC(=O)NC(CC1=CNC2=CC=CC=C21)C(=O)O/cids/TXT
L2. PROLOG/compound/fastidentity/smiles
    /CC(=O)NC(CC1=CNC2=CC=CC=C21)C(=O)O/cids/TXT?identity_type=same_isotope
L3. PROLOG/compound/fastidentity/smiles
    /CC(=O)NC(CC1=CNC2=CC=CC=C21)C(=O)O/cids/TXT?identity_type=same_stereo
L4. PROLOG/compound/fastidentity/smiles
    /CC(=O)NC(CC1=CNC2=CC=CC=C21)C(=O)O/cids/TXT?identity_type=same_connectivity
```

M. Substructure and superstructure search.

```
M1. PROLOG/compound/fastsubstructure/cid/88737084/cids/TXT
M2. PROLOG/compound/fastsubstructure/cid/88737084/cids/TXT?StripHydrogen=true
M3. PROLOG/compound/fastsuperstructure/smiles/CC(CC1=CC=CC=C1)NC/cids/XML
```

N. Similarity search

```
N1. PROLOG/compound/fastsimilarity_2d/cid/60823
    /cids/TXT?Threshold=80&MaxRecords=50&MaxSeconds=10
N2. PROLOG/compound/fastsimilarity_3d/cid/446157
    /property/MolecularFormula,HeavyAtomCount,RotatableBondCount/XML
```

O. Molecular formula search

```
O1. PROLOG/compound/fastformula/C2H6O/cids/XML
O2. PROLOG/compound/fastformula/C2H6O/cids/XML?AllowOtherElements=True
```

Fig. 6 PUG-REST URL examples K1 through O2

structure image of the compound is returned, depending on the value for the "record_type" option. The image retrieval for substances is also supported. The image size can be large (300 × 300 px) or small (100 × 100 px). A custom image size is supported for 2-D images (**Example B3**). When a list of multiple compounds or substances is provided as an input, the image request will only return the image of the first SID or CID in the list. PubChem's download service may be used for retrieval of multiple images.

3.3 Chemical Name Retrieval

The "synonyms" operation on substance records returns an "original" list of chemical names provided by data contributors (**Example C1**). For compound inputs, this operation gives a "filtered" list of chemical names, which removes potentially incorrect names by checking consistency in name–structure association within and between depositors (**Examples C2** and **C3**). Supported output formats are XML, JSON(P), ASNT/B, and TXT (limited). To get only

Table 1
Compound properties available for PUG-REST requests

• MolecularFormula	• UndefinedAtomStereoCount
• MolecularWeight	• BondSteroCount
• CanonicalSMILES	• DefinedBondStereoCount
• IsomericSMILES	• UndefinedBondStereocount
• InChI	• CovalentUnitCount
• InChIKey	• Fingerprint2D
• IUPACName	• ConformerModelRMSD3D
• XLogP	• EffectiveRotorCount3D
• ExactMass	• ConformerCount3D
• MonoisotopicMass	• Volume3D
• TPSA	• XStericQuadrupole3D
• Complexity	• YStericQuadrupole3D
• Charge	• ZStericQuadrupole3D
• HBondDonorCount	• FeatureAcceptorCount3D
• HBondAcceptorCount	• FeatureDonorCount3D
• RotatableBondCount	• FeatureAnionCount3D
• HeavyAtomCount	• FeatuerCationCount3D
• IsotopeAtomCount	• FeatureRingCount3D
• AtomStereoCount	• FeatureHydrophobeCount3D
• DefinedAtomStereoCount	• FeatureCount3D

the chemical name used as the title of the compound summary page of a compound, use the "description" operation (**Example C4**).

3.4 Compound Property Retrieval

A table of pre-computed properties of PubChem compounds may be retrieved through PUG-REST (**Examples D1** and **D2**). Available properties are listed in Table 1. Multiple properties may be retrieved by providing a comma-separated list of properties in a PUG-REST request URL. Supported output formats are XML, ASNT/B, JSON(P), CSV, and TXT (limited to a single property).

3.5 Assay Data Retrieval

An assay record in PubChem can be considered to have two parts: (1) information on the assay experiment, including data contributor, general description of the assay, protocol, definitions of the data readout columns, etc., and (2) the actual bioactivity data for

Table 2
Options for CIDS/AIDS/SIDS operations

Input domain	Operation	Option
Compound	cids	cids_type = (**original**, parent, component,similar_2d, similar_3d,same_stereo, same_isotopes,same_connectivity,same_tautomer,same_parent, same_parent_stereo,same_parent_isotopes, same_parent_connectivitysame_parent_tautomer)
	sids	sids_type = (all, **standardized**, component)
	aids	aids_type = (**all**, active, inactive)
Substance	cids	cids_type = (all, **standardized**, component)
	sids	sids_type = (**original**, same_exact,same_stereo, same_isotopes, same_connectivity,same_tautomer,same_parent,same_parent_stereo, same_parent_isotopes,same_parent_connectivitysame_parent_tautomer)
	aids	aids_type = (**all**, active, inactive)
Assay	cids	cids_type = (**all**, active, inactive)
	sids	sids_type = (**all**, active, inactive,doesresponse)
	aids	Not implemented

substances tested in that assay. Retrieval of the first part (i.e., information on the assay experiment without actual test results) can be done using the "description" operation (**Example E1**). A shorter description of this part can be retrieved through the "summary" operation (**Example E2**). The assay target information can be retrieved by the "targets" operation (**Example E3**). Dose–response data, if available, can also be retrieved (**Examples E4** and **E5**). Bioactivity data for compounds or substances (across multiple assays) can be retrieved through the "assaysummary" operation (**Examples E6** and **E7**).

3.6 Getting PubChem Identifiers Associated with the Input Identifiers

The "sids," "cids," and "aids" operations return a list of SIDs, CIDs, and AIDs, respectively, associated with the input identifiers in various contexts specified with options summarized in Table 2 (**Examples F1** through **F8**). For example, **Example F1** returns SIDs of the substances whose names are "sucrose," and **Examples F2** and **F3** returns CIDs of the compounds standardized from the substances whose names are "sucrose." The "list_return" option specifies whether one-to-one mapping information between the SIDs and CIDs should be retained (=grouped) or whether the list should be flatted to return only unique CIDs (=flat). The "aids_type" option in **Examples F4** and **F5** is used to specify whether to return all AIDs in which the input substances are tested or only those in which the input substances are tested to be active.

In **Examples F7** and **F8**, the “cids_type” option is used to retrieve compounds that are found active in the input assay(s).

3.7 Classification Retrieval

PubChem records are annotated with terms from various classification systems, such as Medical Subject Headings (MeSH) [20], Anatomical Therapeutic Chemical (ATC) Classification System [21], and International Patent Classification (IPC) [22], and these classification systems have a hierarchical tree structure. The “classification” operation (**Examples G1-G3**) allows one to retrieve the nodes with which a PubChem record is annotated. By default, the “classification_type” is set to “simple,” meaning that the nodes are returned in a simplified tree structure, in which each node has a single parent (**Example G2**). When used with the “classification_type=original” option, the PUG-REST request retrieves the nodes as given by the depositor and they may have multiple parents (**Example G3**).

3.8 Cross-Reference Retrieval and Search by Cross-Reference

PubChem records have various types of cross-references, such as patent identifiers, PMID, MMDBID, Registry Number, and many others; available cross-references are listed in Table 3. Through PUG-REST, it is possible to retrieve cross-references associated with a particular PubChem record (**Examples H1** and **H2**) or find PubChem records that are associated with a particular cross-reference (**Examples I1** through **I4**). Note that the “xrefs” operation (with an “s” at the end) is used in **Examples H1** and **H2**, whereas the “xref” namespace (with no “s” at the end) is used in the input part of **Examples I1** through **I4**.

3.9 Search by Chemical Name

Examples J1 through **J5** demonstrate how to search compounds or substances by chemical name. By default, the depositor-provided synonyms of compounds or substances will be searched for an “exact” match to the given name (**Example J1**), which is equivalent to the “name_type=complete” option (**Example J2**). If the “name_type” option is set to “word,” a partial match to the input chemical name will be performed (**Examples J3** through **J5**).

3.10 Search by Chemical Structure Identity

The PUG-REST request in **Example K1** returns only exact matches for the input chemical structure representation. However, the compound namespace called “fastidentity” allows one to specify chemical structures through various partial identity search (**Examples L1** through **L4**). By using different “identity_type” options, one may retrieve the data for compounds with the same connectivity, isotopes, or stereochemistry as the query chemical structure. By default, the “fastidentity” returns compounds with the same isotope and stereochemistry as the query (i.e., identity_type=same_stereo_isotope).

Table 3
Options for CIDS/AIDS/SIDS operations

Cross-reference	Meaning	xref[a]
RegistryID	External registry identifier	O
RN	Registry number	O
PubMedID	NCBI PubMed identifier	O
MMDBID	NCBI MMDB identifier	O
DBURL	External database home page URL	X
SBURL	External database substance URL	X
ProteinGI	NCBI protein GI	O
NucleotideGI	NCBI nucleotide GI	O
TaxonomyID	NCBI taxonomy identifier	O
MIMID	NCBI MIM identifier	O
GeneID	NCBI gene identifier	O
ProbeID	NCBI probe identifier	O
PatentID	Patent identifier	O
SourceName	External depositor name	O[b]
SourceCategory	Depositor category	X

[a]This column indicates whether a given cross-reference type can be used to specify the input identifiers
[b]The source name should be used with the "sourceid" namespace. See Fig. 2

3.11 Search by Substructure and Superstructure

The compound namespace inputs "fastsubstructure" and "fastsuperstructure" returns compounds that have a particular substructure or superstructure search, respectively (**Examples M1** through **M3**). By default, the explicit hydrogen atoms in the query structure are retained in a substructure search. However, the "StripHydrogen=true" option will remove all explicit hydrogen atoms from the query structure before the search.

3.12 2-D and 3-D Similarity Search

The fastsimilairty_2d and fastsimilarity_3d namespace inputs retrieve CIDs by 2-D and 3-D similarity search, respectively (**Examples N1** and **N2**). The 2-D similarity between the query compound and other compounds in PubChem is evaluated using the PubChem substructure fingerprint [23] and the Tanimoto coefficient [24–26]. The Tanimoto score ranges from 0 (no similarity) to 1 (identical molecules), and by default, the "fastsimilarity_2d" will return compounds with a Tanimoto score of $\geq$0.90. This Tanimoto threshold is adjustable using the "Threshold" option (**Example N1**). The 3-D similarity is estimated with shape-Tanimoto (ST) [14, 27–29] and color-Tanimoto (CT) scores [14, 27, 28], both of which range from 0 to 1.

The default threshold for the “fastsimilarity_3d” namespace input is ST $\geq$ 0.8 and CT $\geq$0.5, which is identical to those used for PubChem 3-D neighboring [28]. However, fastsimilarity_3d uses only one conformer per compound. Note that, in PUG-REST, all similarity scores and thresholds (including Tanimoto scores for fastsimilarity_2d and ST and CT scores for fastsimilarity_3d) are converted to integer percentages (i.e., ranging from 0 to 100). As a result, the threshold in **Example N1** is set to 80 (not 0.80).

3.13 Search by Molecular Formula

The fastformula namespace supports specifying input compounds by molecular formula search (**Examples O1** and **O2**). By default, the fastformula input will retrieve compounds with the exact query molecular formula. If the “AllowOtherElements” option is set to true, other elements are allowed to be present in addition to those specified in the query. For instance, **Example O1** returns compounds whose molecular formula is “C_2H_6O,” but compounds retrieved in **Example O2** include not only C_2H_6O but also, for example, $C_2H_6N_2O$, C_2H_6OS, and $C_2H_6Cl_2OSi$, along with many others.

4 PUG-REST Requests Using Scripts

4.1 PUG-REST Requests Using HTTP GET

Figure 7 illustrates a simple Perl script that retrieves a full record of CID 45375808, including 3-D structure information. This example uses the LWP::Simple module (Line 5) [30], which provides a simple procedural interface to the World Wide Web. [LWP stands for the libwww-perl collection, a set of perl modules that provide application programming interface (API) to the World Wide Web.] All information necessary for a PUG-REST request, including the prologue, input, operation, output, and option, is provided in Lines 7 through 14. Based on this information, a PUG-REST URL is constructed (Line 19) and sent to the PubChem server (Line 24). The returned SDF file is printed (Line 26).

The script shown in Fig. 8 performs exactly the same task as Fig. 7 but uses the full object-oriented interface provided by the LWP::UserAgent module [31]. In this approach, an LWP::UserAgent object (Line 25) with an HTTP::Request object (Line 30) [32] needs to be created to send the PUG-REST request (Line 35). The use of the LWP::UserAgent module provides more control over the requests sent and responses received. Importantly, this is very useful for sending a request via HTTP POST, as demonstrated in the examples below.

4.2 PUG-REST Requests Using HTTP POST

While most information necessary for a PUG-REST request can be encoded into a one-line URL string, some inputs are not compatible with URL syntax or size restrictions. Such examples are multi-line SDF files, any chemical names or representations that contain “/” (forward slash) or other special characters reserved in the URL

```perl
#!/opt/perl/5.16.3/bin/perl -w

use strict;
use warnings;
use LWP::Simple;

my $cid         =  "45375808";

my $pugrest     =  "https://pubchem.ncbi.nlm.nih.gov/rest/pug";

my $input       =  "compound/cid/${cid}";
my $operation   =  "record";
my $output      =  "sdf";
my $options     =  "record_type=3d";


#-- Assemble the PUG-REST URL.

my $url = "$pugrest/${input}/${operation}/${output}?${options}";


#-- Send the PUG-REST request.

my $sdf = get($url);

print $sdf;
```

Fig. 7 Example Perl script that uses the LWP::Simple class to make a PUG-REST request. This script retrieves the full record for CID 45375808 in the Structure-Data File (SDF) format

syntax, and lists of identifiers that are too big to put directly in the URL of an HTTP GET request. PUG-REST requests with these inputs should use the HTTP POST method, as demonstrated in Figs. 9, 10, 11, and 12.

As shown in Line 15 in Fig. 9, InChI encodes structure information of a molecule into a sequence of layers and sub-layers separated by "/" (forward slash), which is reserved in URL syntax. Therefore, InChI strings cannot be directly encoded into a PUG-REST URL. Similarly, some SMILES may be incompatible with the URL syntax, because isomeric SMILES uses "/" (and "\") for representing cis–trans conformations in a molecule (as in Line 17 of Fig. 10). Isomeric SMILES for molecules with a double bond may also conflict with the URL syntax. These inputs should be provided in the HTTP POST body formatted according to either of the two existing form content types: "application/x-www-form-urlencoded" (Figs. 9 and 10) and "multipart/form-data" (Figs. 11 and 12).

If the "application/x-www-form-urlencoded" content type is used (Figs. 9 and 10), the URL arguments can be provided in one line in the general format of "arg1=value1&arg2=value2&. . .." If the "multipart/form-data" content type is used, they should be provided as a file or string formatted in the way shown in Figs. 11 and 12. Note that this case must have DOS-style "CR (carriage return) + LF (line feed)" line endings (that is, "\r\n" at the end of

```
#!/opt/perl/5.16.3/bin/perl -w

use strict;
use warnings;
use LWP::Simple;
use LWP::UserAgent;

my $cid         =  "45375808";

my $pugrest     =  "https://pubchem.ncbi.nlm.nih.gov/rest/pug";

my $input       =  "compound/cid/${cid}";
my $operation   =  "record";
my $output      =  "sdf";
my $options     =  "record_type=3d";


#-- Assemble the PUG-REST URL.

my $url = "$pugrest/${input}/${operation}/${output}?${options}";


#-- Create HTTP user agent.

my $ua = new LWP::UserAgent;


#-- Create HTTP request object.

my $req = new HTTP::Request GET => "$url";


#-- Send the HTTP request.

my $response = $ua->request($req);

print $response->content;
```

Fig. 8 Example Perl script that uses the LWP::UserAgent class to make a PUG-REST request. This script returns exactly the same result as the script in Fig. 7

Lines 23–28 of Fig. 11 and Lines 35–40 of Fig. 12) and that there must be an empty line between the content headers and actual data line(s) (and no blank lines anywhere else). In this format, no further escaping of special characters is needed. It is essential that the arbitrary boundary string given in the header ("AaB03x" in Line 36 of Fig. 11 and Line 48 of Fig. 12) matches what is used in the POST body.

Formulating the HTTP POST body in the script shown in Fig. 12 involves two inconveniences: the need to read the contents in from the input SD file and the use of the DOS-style "CR+LF" line endings. Figure 13 presents a simpler script that avoids these inconveniences, using the HTTP::Request::Common module [33]. This script performs exactly the same task as Fig. 12 but does not read the data in from the SD file nor add the "CR+LF"

```
 1  #!/opt/perl/5.16.3/bin/perl -w
 2
 3  use strict;
 4  use warnings;
 5  use LWP::Simple;
 6  use LWP::UserAgent;
 7
 8  my $pugrest    =  "https://pubchem.ncbi.nlm.nih.gov/rest/pug";
 9
10  my $input      =  "compound/inchi";
11  my $operation  =  "record";
12  my $output     =  "xml";
13  my $options    =  "record_type=3d";
14
15  my $inchi      =  "InChI=1S/C6H6/c1-2-4-6-5-3-1/h1-6H";
16
17
18  #-- Assemble the PUG-REST URL as an HTTP POST call.
19
20  my $url = "$pugrest/${input}/${operation}/${output}?${options}";
21
22  my $url_params = "inchi=$inchi";
23
24
25  #-- Create HTTP user agent.
26
27  my $ua = new LWP::UserAgent;
28
29
30  #-- Create HTTP request object.
31
32  my $req = new HTTP::Request POST => "$url";
33  $req->content_type('application/x-www-form-urlencoded');
34  $req->content("$url_params");
35
36
37  #-- Post the HTTP request.
38
39  my $response = $ua->request($req);
40
41  print $response->content;
```

Fig. 9 Example Perl script that retrieves the full record for the compound represented with an InChI string. This script uses the HTTP POST method to provide the input InChI string, which contains special characters incompatible with the URL syntax for a PUG-REST request. The HTTP POST body in this script is formatted according to the content type "application/x-www-form-urlencoded"

line endings. The HTTP POST body will be automatically generated from the information given in Lines 30–32.

Both the "application/x-www-form-urlencoded" and "multipart/form-data" content types are used to make a PUG-REST request that contains special characters incompatible with the URL syntax (as shown in Figs. 9, 10, and 11). However, the multiline data such as the SDF file can only be provided in the HTTP POST body formatted according to the "multipart/form-data" content type (as shown in Figs. 12 and 13).

```perl
#!/opt/perl/5.16.3/bin/perl -w

use strict;
use warnings;
use LWP::Simple;
use LWP::UserAgent;

my $pugrest    =  "https://pubchem.ncbi.nlm.nih.gov/rest/pug";

my $input      =  "compound/fastsimilarity_2d/smiles";
my $operation  =  "cids";
my $output     =  "txt";

my $threshold   =  "95";
my $maxrecords  =  "25";

my $smiles      =  'C[C@@H](/C=C\C1=CC=[N+](C=C1)C)O';

$smiles =~ s/\+/%2B/g;


#-- Assemble the PUG-REST URL as an HTTP POST call.

my $url        = "$pugrest/${input}/${operation}/${output}";

my $url_params = "smiles=$smiles" .
                 "&" . "Threshold=$threshold" .
                 "&" . "MaxRecords=$maxrecords" ;


#-- Create HTTP user agent.

my $ua = new LWP::UserAgent;


#-- Create HTTP request object.

my $req = new HTTP::Request POST => "$url";
$req->content_type('application/x-www-form-urlencoded');
$req->content("$url_params");


#-- Post the HTTP request.

my $response = $ua->request($req);

print $response->content;
```

Fig. 10 Perl script that performs 2-D similarity search for compounds that are similar to the query molecule represented by a SMILES string, which contains special characters incompatible with the PUG-REST URL syntax. The SMILES string is provided in the HTTP POST body formatted according to the content type "application/x-www-form-urlencoded"

```perl
#!/opt/perl/5.16.3/bin/perl -w

use strict;
use warnings;
use LWP::Simple;
use LWP::UserAgent;

my $pugrest    =  "https://pubchem.ncbi.nlm.nih.gov/rest/pug";

my $input      =  "compound/inchi";
my $operation  =  "record";
my $output     =  "xml";
my $options    =  "record_type=3d";

my $inchi      =  "InChI=1S/C6H6/c1-2-4-6-5-3-1/h1-6H";


#-- Assemble the PUG-REST URL as an HTTP POST call.

my $url  =  "$pugrest/$input/$operation/$output?$options";

my $url_params  =
        '--AaB03x'                                      . "\r\n"
      . 'Content-Disposition: form-data; name="inchi"'  . "\r\n"
      . 'Content-Type: text/plain'                      . "\r\n"
      . ''                                              . "\r\n"
      . "$inchi"                                        . "\r\n"
      . '--AaB03x--'                                    . "\r\n";


#-- Create HTTP user agent & POST request objects.

my $ua  =  new LWP::UserAgent;

my $req  =  new HTTP::Request POST => "$url";
$req->content_type('multipart/form-data; boundary=AaB03x');
$req->content("$url_params");


#-- Post the HTTP request.

my $response = $ua->request($req);

print $response->content;
```

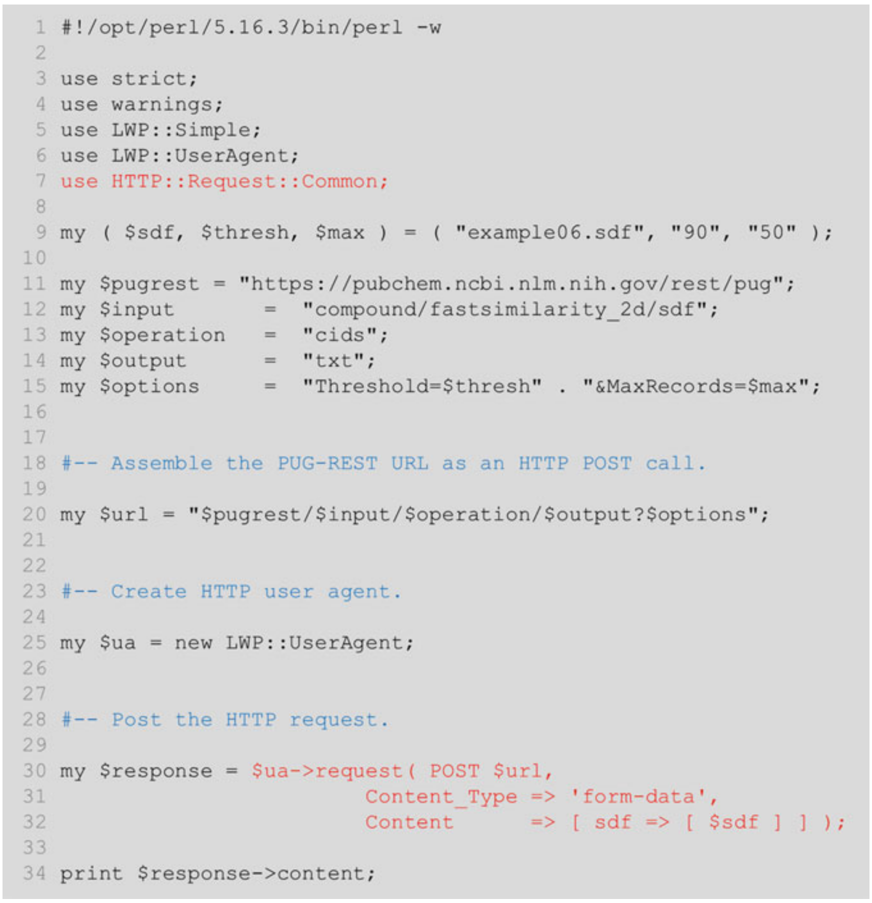
Fig. 11 Perl script that retrieves the full record for a compound specified by an InChI string, which are incompatible with the PUG-REST URL syntax. The InChI string is provided in the HTTP POST body formatted according to the content type "multipart/form-data"

4.3 Using PUG-REST with E-Utilities

Currently, PUG-REST does not support input by molecular property values. For example, it is not possible to set the input identifiers to the compounds whose molecular weight falls within a particular value range. The script in Fig. 14 demonstrates how to accomplish

```perl
#!/opt/perl/5.16.3/bin/perl -w

use strict;
use warnings;
use LWP::Simple;
use LWP::UserAgent;

my ( $sdf, $thresh, $max ) = ( "example06.sdf", "90", "50" );

my $pugrest     =  "https://pubchem.ncbi.nlm.nih.gov/rest/pug";
my $input       =  "compound/fastsimilarity_2d/sdf";
my $operation   =  "cids";
my $output      =  "txt";
my $options     =  "Threshold=$thresh" . "&MaxRecords=$max";


#--  Read data from the SDF file.

my $data;

open(SDF,$sdf) || die "Can't open file $sdf for reading: $!\n";

while(<SDF>) {
    $data .= $_;
}

close(SDF) || die "Can't close file $sdf: $!\n";


#-- Assemble the PUG-REST URL as an HTTP POST call.

my $url = "$pugrest/$input/$operation/$output?$options";

my $url_params =
       '--AaB03x'                                      . "\r\n"
     . 'Content-Disposition: form-data; name="sdf"'    . "\r\n"
     . 'Content-Type: text/plain'                      . "\r\n"
     . ''                                              . "\r\n"
     . "$data"                                         . "\r\n"
     . '--AaB03x--'

#-- Create HTTP user agent & POST request objects.

my $ua = new LWP::UserAgent;

my $req = new HTTP::Request POST => "$url";
$req->content_type('multipart/form-data; boundary=AaB03x');
$req->content("$url_params");


#-- Post the HTTP request.

my $response = $ua->request($req);

print $response->content;
```

Fig. 12 Perl script that performs 2-D similarity search for compounds similar to the query molecule provided in an SD file. The multiline data in the SD file is provided in the HTTP POST body formatted according to the content type "multipart/form-data"



```
#!/opt/perl/5.16.3/bin/perl -w

use strict;
use warnings;
use LWP::Simple;
use LWP::UserAgent;
use HTTP::Request::Common;

my ( $sdf, $thresh, $max ) = ( "example06.sdf", "90", "50" );

my $pugrest = "https://pubchem.ncbi.nlm.nih.gov/rest/pug";
my $input        =  "compound/fastsimilarity_2d/sdf";
my $operation    =  "cids";
my $output       =  "txt";
my $options      =  "Threshold=$thresh" . "&MaxRecords=$max";


#-- Assemble the PUG-REST URL as an HTTP POST call.

my $url = "$pugrest/$input/$operation/$output?$options";


#-- Create HTTP user agent.

my $ua = new LWP::UserAgent;


#-- Post the HTTP request.

my $response = $ua->request( POST $url,
                             Content_Type => 'form-data',
                             Content      => [ sdf => [ $sdf ] ] );

print $response->content;
```

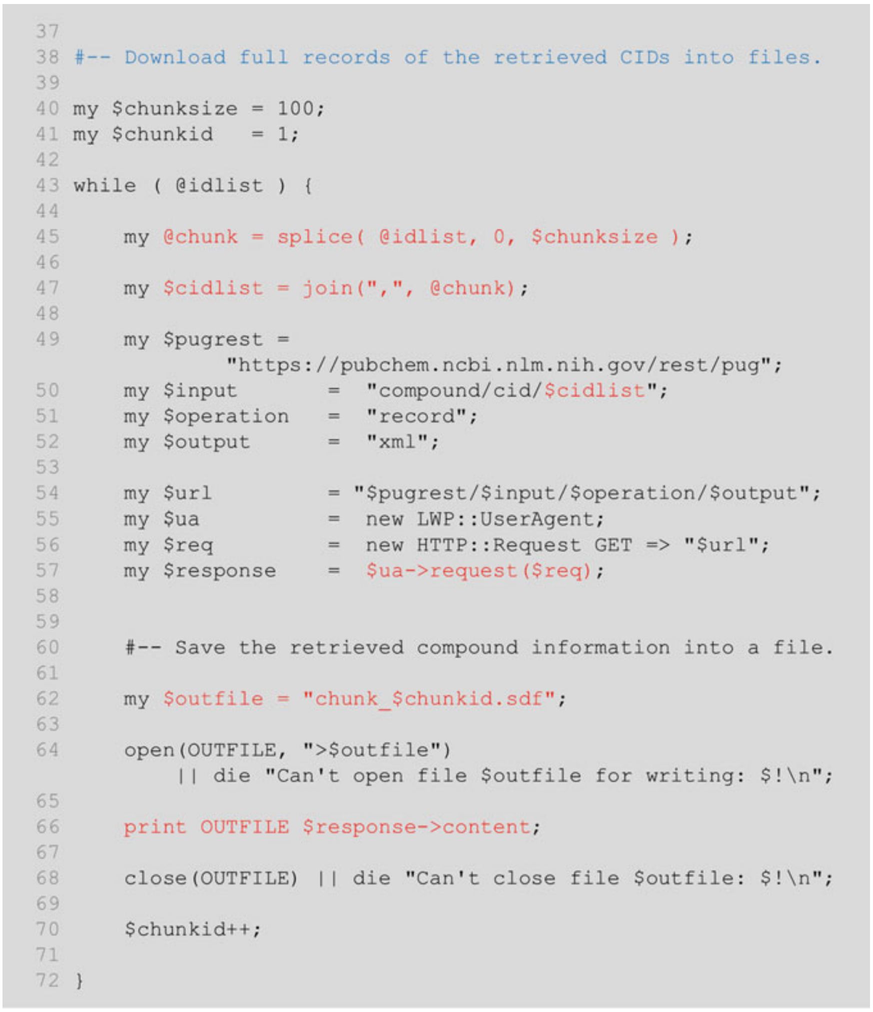
Fig. 13 Perl script that performs the same task as Fig. 12, using the HTTP::Request::Common module. Setting the content type to “form-data” invokes formulation of the same HTTP POST body as in Fig. 12, without reading the contents in the SD file and adding the DOS-type CR (carriage return) + LF (line feed) line endings

this using E-Utilities and PUG-REST together. It retrieves compounds whose molecular weight ranges from 449.995 to 450.005 Da (Lines 11–21), using the Esearch utility. From the Esearch result returned in an XML format, the retrieved CIDs are extracted and stored into an array (Lines 26–35). This array is chunked into smaller arrays, which are provided as input identifiers in subsequent PUG-REST requests for full-record retrieval (Lines 40–72).

PubChem has a standard time limit of 30 s for a PUG-REST request, and any request that takes longer than 30 s returns a time-out error. Therefore, a PUG-REST request with a long list of input identifiers needs to be divided into multiple requests with shorter input identifier lists, as shown in Fig. 14.

```
#!/opt/perl/5.16.3/bin/perl -w

use strict;
use warnings;
use LWP::Simple;
use LWP::UserAgent;


#-- Get CIDs with M.W. between 449.995 and 450.005
    using E-Util's esearch.

my $db = "pccompound";

my ( $mw_min, $mw_max ) = ( 449.995, 450.005 );

my $query = "$mw_min".":"."$mw_max"."[MolecularWeight]";

my $eutils = "https://www.ncbi.nlm.nih.gov/entrez/eutils";

my $esearch =
   "$eutils/esearch.fcgi?"."db=$db&retmax=10000&term=$query";

my $esearch_result = get( $esearch );


#-- Extract CIDs from the Entrez Search result.

my @idlist;
my $filehandle;

open $filehandle, "<", \$esearch_result;

while(<$filehandle>) {

    push( @idlist, $1 ) if ( /<Id>(\d+)<\/Id>/ );

}

```

Fig. 14 Perl script that retrieves the full records of compounds whose molecular weight ranges from 449.995 to 450.005. The CIDs within the target molecular weight range is retrieved using E-Utilities, and then the full records are retrieved using PUG-REST



```

#-- Download full records of the retrieved CIDs into files.

my $chunksize = 100;
my $chunkid   = 1;

while ( @idlist ) {

    my @chunk = splice( @idlist, 0, $chunksize );

    my $cidlist = join(",", @chunk);

    my $pugrest =
            "https://pubchem.ncbi.nlm.nih.gov/rest/pug";
    my $input        =  "compound/cid/$cidlist";
    my $operation    =  "record";
    my $output       =  "xml";

    my $url          = "$pugrest/$input/$operation/$output";
    my $ua           =  new LWP::UserAgent;
    my $req          =  new HTTP::Request GET => "$url";
    my $response     =  $ua->request($req);


    #-- Save the retrieved compound information into a file.

    my $outfile = "chunk_$chunkid.sdf";

    open(OUTFILE, ">$outfile")
        || die "Can't open file $outfile for writing: $!\n";

    print OUTFILE $response->content;

    close(OUTFILE) || die "Can't close file $outfile: $!\n";

    $chunkid++;

}
```

Fig. 14 (continued)

5 PUG-REST Requests from Command Line

One may often want to make a simple PUG-REST request from the command line. One way to do this on a Unix system is with the cURL command line tool, as shown in Fig. 15. The examples in Fig. 15 perform the same tasks as the scripts in Figs. 7, 8, 9, and 10. The "-d" option is used to provide data in the POST body, whose format is specified using the "-H" option. Note that, in the third example, the special character "+" is replaced with "%2B," as in Fig. 10.

```
❖ Equivalent to Figures 7 and 8

> curl https://pubchem.ncbi.nlm.nih.gov/rest/pug
       /compound/cid/45375808/record/sdf?record_type=3d

❖ Equivalent to Figure 9

> curl -H 'Content-Type: application/x-www-form-urlencoded'
       -d 'inchi=InChI=1S/C6H6/c1-2-4-6-5-3-1/h1-6H'
       https://pubchem.ncbi.nlm.nih.gov/rest/pug
       /compound/inchi/record/xml?record_type=3d

❖ Equivalent to Figure 10

> curl -H 'Content-Type: application/x-www-form-urlencoded'
       -d 'smiles=C[C@@H](/C=C\C1=CC=[N%2B](C=C1)C)O
           &Threshold=95&maxrecords=25'
       https://pubchem.ncbi.nlm.nih.gov/rest/pug
       /compound/inchi/record/xml?record_type=3d
```

Fig. 15 PUG-REST requests from the command line, equivalent to the scripts in Figs. 7, 8, 9, and 10. Although shown in multiple lines for legibility, each example should be typed in a single line

6 Usage Policies and Future Directions

Because PUG-REST is not designed to handle a very large number (millions) of requests in a short time, any script or application should not make more than five requests per second to avoid overloading the PubChem servers. More detailed usage policies on programmatic request limits are described elsewhere [5, 6]. Violation of the usage policies may result in the user being temporarily blocked from accessing PubChem (or NCBI) resources. If you have a large data set that you need to compute with, please contact PubChem for help on optimizing your task, because more efficient ways likely exist to approach such bulk queries.

Acknowledgments

This research was supported in part by the Intramural Research Program of the National Library of Medicine, National Institutes of Health, the US Department of Health and Human Services.

References

1. Kim S, Thiessen PA, Bolton EE, Chen J, Fu G, Gindulyte A, Han L, He J, He S, Shoemaker BA, Wang J, Yu B, Zhang J, Bryant SH (2016) PubChem substance and compound databases. Nucleic Acids Res 44(D1):D1202–D1213. https://doi.org/10.1093/nar/gkv951

2. Wang YL, Suzek T, Zhang J, Wang JY, He SQ, Cheng TJ, Shoemaker BA, Gindulyte A, Bryant SH (2014) PubChem BioAssay: 2014 update. Nucleic Acids Res 42(D1): D1075–D1082. https://doi.org/10.1093/nar/gkt978

3. Kim S (2016) Getting the most out of PubChem for virtual screening. Expert Opin Drug Discov 11(9):843–855. https://doi.org/10.1080/17460441.2016.1216967
4. Kim S, Thiessen PA, Bolton EE, Bryant SH (2015) PUG-SOAP and PUG-REST: web services for programmatic access to chemical information in PubChem. Nucleic Acids Res 43(W1):W605–W611. https://doi.org/10.1093/nar/gkv396
5. Kim S, Thiessen PA, Cheng T, Yu B, Bolton EE (2018) An update on PUG-REST: RESTful interface for programmatic access to PubChem. Nucleic Acids Res 46(W1): W563–W570. https://doi.org/10.1093/nar/gky294
6. Programmatic Access to PubChem. https://pubchemdocs.ncbi.nlm.nih.gov/programmatic-access
7. Entrez programming utilities help. https://www.ncbi.nlm.nih.gov/books/NBK25501
8. Entrez Help (2005) National Center for Biotechnology Information. https://www.ncbi.nlm.nih.gov/books/NBK3836/. Accessed 9 Nov 2015
9. SOAP Specifications. http://www.w3.org/TR/soap/
10. Fielding RT (2000) Representational state transfer (REST). In: Architectural styles and the design of network-based software architectures. University of California, Irvine
11. Fielding RT, Taylor RN (2000) Principled design of the modern Web architecture. Proceedings of the 22nd international conference on software engineering. pp 407–416. https://doi.org/10.1145/337180.337228
12. PUG-REST. https://pubchemdocs.ncbi.nlm.nih.gov/pug-rest
13. A PUG-REST Tutorial. https://pubchemdocs.ncbi.nlm.nih.gov/pug-rest-tutorial
14. Bolton EE, Chen J, Kim S, Han L, He S, Shi W, Simonyan V, Sun Y, Thiessen PA, Wang J, Yu B, Zhang J, Bryant SH (2011) PubChem3D: a new resource for scientists. J Cheminform 3(1):32. https://doi.org/10.1186/1758-2946-3-32
15. Bolton EE, Kim S, Bryant SH (2011) PubChem3D: conformer generation. J Cheminform 3(1):4. https://doi.org/10.1186/1758-2946-3-4
16. Kim S, Bolton EE, Bryant SH (2013) PubChem3D: conformer ensemble accuracy. J Cheminform 5(1):1. https://doi.org/10.1186/1758-2946-5-1
17. PubChem Structure Download Service. https://pubchem.ncbi.nlm.nih.gov/pc_fetch/pc_fetch.cgi
18. PubChem Assay Download Service. https://pubchem.ncbi.nlm.nih.gov/assay/assaydownload.cgi
19. PubChem FTP Site. ftp://ftp.ncbi.nlm.nih.gov/pubchem/
20. Medical Subject Headings. https://www.ncbi.nlm.nih.gov/mesh
21. Anatomical Therapeutic Chemical (ATC) Classification System. http://www.who.int/classifications/atcddd/en/
22. International Patent Classification (IPC). http://www.wipo.int/classifications/ipc/en/
23. PubChem substructure fingerprint description. ftp://ftp.ncbi.nlm.nih.gov/pubchem/specifications/pubchem_fingerprints.pdf
24. Holliday JD, Hu CY, Willett P (2002) Grouping of coefficients for the calculation of intermolecular similarity and dissimilarity using 2D fragment bit-strings. Comb Chem High Throughput Screen 5(2):155–166
25. Holliday JD, Salim N, Whittle M, Willett P (2003) Analysis and display of the size dependence of chemical similarity coefficients. J Chem Inf Comput Sci 43(3):819–828. https://doi.org/10.1021/ci034001x
26. Chen X, Reynolds CH (2002) Performance of similarity measures in 2D fragment-based similarity searching: comparison of structural descriptors and similarity coefficients. J Chem Inf Comput Sci 42(6):1407–1414. https://doi.org/10.1021/ci025531g
27. ROCS—Rapid Overlay of Chemical Structures (2011) 3.1.1 ed. OpenEye Scientific Software, Inc., Santa Fe, NM
28. Bolton EE, Kim S, Bryant SH (2011) PubChem3D: similar conformers. J Cheminform 3(1):13. https://doi.org/10.1186/1758-2946-3-13
29. Grant JA, Gallardo MA, Pickup BT (1996) A fast method of molecular shape comparison: a simple application of a Gaussian description of molecular shape. J Comput Chem 17 (14):1653–1666
30. LWP::Simple—simple procedural interface to LWP. https://metacpan.org/pod/distribution/libwww-perl/lib/LWP/Simple.pm
31. LWP::UserAgent—Web user agent class. https://metacpan.org/pod/release/ETHER/libwww-perl-6.15/lib/LWP/UserAgent.pm
32. HTTP::Request—HTTP style request message. https://metacpan.org/pod/release/ETHER/HTTP-Message-6.11/lib/HTTP/Request.pm
33. HTTP::Request::Common—Construct common HTTP::Request objects. https://metacpan.org/pod/release/ETHER/HTTP-Message-6.11/lib/HTTP/Request/Common.pm