PubChem Power User Gateway (PUG)

V1.3.0 http://pubchem.ncbi.nlm.nih.gov

1. Introduction.

The PubChem **P**ower **U**ser **G**ateway (**PUG**) provides access to PubChem services via a programmatic interface. The basic design principle is straightforward. There is a single CGI (pug.cgi, referred to hereafter as simply PUG) that is the central gateway to multiple PubChem functions. PUG takes no URL arguments; all communication with PUG is through XML. To perform any request, one formulates input in XML and then HTTP POST it to PUG. The CGI interprets your incoming request, initiates the appropriate action, then returns results (also) in XML format. (This document assumes a basic familiarity with XML tags and data structures. To learn more about XML, visit the URL: http://en.wikipedia.org/wiki/XML)

PubChem services are queued. As such, a submitted task will (usually) complete sometime after PUG responds to the initial request. The initial PUG response contains the request ID of your task. This request ID must be used for further communication with PUG concerning your submitted task. When PUG is interrogated about an outstanding request using the request ID, PUG will return either the results of your task, if completed, or the status of your task.

Each PubChem service enabled for use with PUG is documented separately. This service by service documentation will detail the input, output, and options. All XML used by PUG is specified in the data type definition (DTD), which may be found at:

http://pubchem.ncbi.nlm.nih.gov/pug/pug.dtd

or in the equivalent XML Schema definition at:

http://pubchem.ncbi.nlm.nih.gov/pug/pug.xsd

We strongly recommend using an XML parser/generator tool to read and write the XML data, rather than composing XML manually.

PubChem PUG enabled services have the ability to save and open valid PUG requests designed for that service. You can use this feature to learn how to compose valid PUG XML requests and to verify that your PUG XML request does what is intended. Examples of such services are provided in this document.

Additional documentation on PubChem and its services may be found at http://pubchem.ncbi.nlm.nih.gov and via help links throughout PubChem's web site. If you cannot find what you need there, further requests for information or help may be sent to the highly knowledgeable and responsive NCBI help desk at info@ncbi.nlm.nih.gov.

2. Interacting with PUG.

All communication to PUG is via XML sent to the CGI at the URL:

http://pubchem.ncbi.nlm.nih.gov/pug/pug.cgi

The primary data container used in all transactions is <*PCT-Data*>, the top-level container for any PUG input or output. ("PCT" stands for PubChem Tools, a data specification that is shared by both PUG and internal PubChem applications. See the introduction in this document for more information.) This <*PCT-Data*> object may contain either a <*PCT-InputData*> or a <*PCT-OutputData*> object. Users of PUG will always send <*PCT-Data*> containing <*PCT-InputData*>, and always receive <*PCT-Data*> containing <*PCT-OutputData*>.

After a new task is submitted to PUG, your request is queued, rather than executing immediately. As such, PUG will return an XML message containing a request ID to be used for further actions on your request. In the example PUG XML reply below, the message says that the request was successfully submitted and that the request ID is "402936103567975582". It will then be up to you to (periodically) poll PUG, using the request ID, until your task is complete. When your task is completed, PUG will return the result; otherwise it will simply return a status message. See examples in other sections of this document on how to properly poll PUG.

```
Example of a PUG reply to a newly submitted request:
    <PCT-Data>
      <PCT-Data_output>
        <PCT-OutputData>
          <PCT-OutputData_status>
            <PCT-Status-Message>
              <PCT-Status-Message_status>
                <PCT-Status value="success"/>
              </PCT-Status-Message_status>
            </PCT-Status-Message>
          </PCT-OutputData_status>
          <PCT-OutputData_output>
            <PCT-OutputData_output_waiting>
              <PCT-Waiting>
                <PCT-Waiting_reqid>402936103567975582</PCT-Waiting_reqid>
              </PCT-Waiting>
            </PCT-OutputData_output_waiting>
          </PCT-OutputData_output>
        </PCT-OutputData>
      </PCT-Data_output>
    </PCT-Data>
```

The <*PCT-InputData*> object is a choice between request types. Tasks specific to various PubChem services are contained by <*PCT-InputData*> and are described in different sections of this document. Primary to the use of PUG is the <*PCT-InputData*> input type used to perform request management, <*PCT-Request*>. Request management enables you to enquire about the status of or to cancel a previous PUG request. For example, to cancel a PUG request with request ID "402936103567975582", the PUG XML input message will look like this:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_request>
        <PCT-Request>
          <PCT-Request_reqid>402936103567975582</PCT-Request_reqid>
          <PCT-Request_type value="cancel"/>
        </PCT-Request>
      </PCT-InputData_request>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

The <*PCT-OutputData*> object contained in the output from PUG will always include a status message in a <*PCT-Status-Message*>, which consists of an enumerated status in <*PCT-Status*> and an optional message string. When a new task is queued by PUG, the <*PCT-OutputData*> returned to you will (likely) contain a <*PCT-Waiting*> which contains your request ID. If the request finishes quickly, the initially returned <*PCT-*

OutputData> object will actually contain the appropriate result of your task specific to the requested service. Similarly, when polling PUG using your request ID, the *<PCT-OutputData>* object will contain either your task result or a status message.

3. PUG Tasks.

PubChem services currently enabled for use by PUG include PubChem Download (http://pubchem.ncbi.nlm.nih.gov/pc_fetch/pc_fetch.cgi), PubChem Compound Structure Search (http://pubchem.ncbi.nlm.nih.gov/search/search.cgi), PubChem BioAssay Data Download (http://pubchem.ncbi.nlm.nih.gov/assay/assay.cgi), and PubChem Structure Standardization (http://pubchem.ncbi.nlm.nih.gov/standardize/standardize.cgi). Each PUG service has its own expected input and provided output. The sections below detail how to use each service with PUG.

3.1. PubChem Substance/Compound Download Tasks.

This service allows you to download sets of PubChem records - substances or compounds - using PUG's *<PCT-Download>* sub-object. You will need to specify which records to download, using a *<PCT-QueryUids>* object, the desired output format (ASN.1, XML, or SDF), and, optionally, the desired compression method (gzip or bzip2). The options available through PUG are equivalent to those for the interactive PubChem Download service.

The *<PCT-QueryUids>* object enables you to specify an explicit list of record IDs, or to provide an existing Entrez history key (see eUtils documentation: http://eutils.ncbi.nlm.nih.gov/entrez/query/static/eutils_help.html) from either the PubChem Compound ("pccompound") or the PubChem Substance ("pcsubstance") Entrez databases. Currently there is an upper limit of 250,000 structures per download request; if you find this limit too restrictive for your purposes, please consider using the PubChem FTP site which contains all available PubChem contents:

ftp://ftp.ncbi.nlm.nih.gov/pubchem/

When your download request is successfully completed, the returned *<PCT-OutputData>* object will hold a *<PCT-Download-URL>* containing the URL you may use to download your results. Again, please note that the result of a download task is an URL, not the record data itself (which may be quite large). To obtain the data requested, you must use the provided URL.

Example: You want to download CID 1 and CID 99 - being uids 1 and 99 in the "pccompound" Entrez database - in SDF format with gzip compression.

The typical flow of information is as follows. First, the initial input XML is sent to PUG via HTTP POST. Note the input data container with the download request and uid and format options:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_download>
        <PCT-Download>
          <PCT-Download_uids>
```

```
                    <PCT-QueryUids>
                      <PCT-QueryUids_ids>
                        <PCT-ID-List>
                          <PCT-ID-List_db>pccompound</PCT-ID-List_db>
                          <PCT-ID-List_uids>
                            <PCT-ID-List_uids_E>1</PCT-ID-List_uids_E>
                            <PCT-ID-List_uids_E>99</PCT-ID-List_uids_E>
                          </PCT-ID-List_uids>
                        </PCT-ID-List>
                      </PCT-QueryUids_ids>
                    </PCT-QueryUids>
                  </PCT-Download_uids>
                  <PCT-Download_format value="sdf"/>
                  <PCT-Download_compression value="gzip"/>
                </PCT-Download>
              </PCT-InputData_download>
            </PCT-InputData>
          </PCT-Data_input>
        </PCT-Data>
```

If the request is small and finishes very quickly, you may get a final URL right away (see further below). But usually PUG will respond initially with a waiting message and a request ID (*<PCT-Waiting_reqid>*) such as:

```
<PCT-Data>
  <PCT-Data_output>
    <PCT-OutputData>
      <PCT-OutputData_status>
        <PCT-Status-Message>
          <PCT-Status-Message_status>
            <PCT-Status value="success"/>
          </PCT-Status-Message_status>
        </PCT-Status-Message>
      </PCT-OutputData_status>
      <PCT-OutputData_output>
        <PCT-OutputData_output_waiting>
          <PCT-Waiting>
            <PCT-Waiting_reqid>402936103567975582</PCT-Waiting_reqid>
          </PCT-Waiting>
        </PCT-OutputData_output_waiting>
      </PCT-OutputData_output>
    </PCT-OutputData>
  </PCT-Data_output>
</PCT-Data>
```

You would then parse out this request id, being "`402936103567975582`", in this case, and use this id to "poll" PUG on the status of the request, composing an XML message like:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_request>
        <PCT-Request>
          <PCT-Request_reqid>402936103567975582</PCT-Request_reqid>
          <PCT-Request_type value="status"/>
        </PCT-Request>
      </PCT-InputData_request>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

Note that here the request type "status" is used; there is also the request type "cancel" that you may use to cancel a running job.

If the request is still running, you well get back another waiting message as above, and then you would poll again after some reasonable interval. If the request is finished, you will get a final result message like:

```
<PCT-Data>
  <PCT-Data_output>
    <PCT-OutputData>
      <PCT-OutputData_status>
        <PCT-Status-Message>
          <PCT-Status-Message_status>
            <PCT-Status value="success"/>
          </PCT-Status-Message_status>
        </PCT-Status-Message>
      </PCT-OutputData_status>
      <PCT-OutputData_output>
        <PCT-OutputData_output_download-url>
          <PCT-Download-URL>
            <PCT-Download-URL_url>ftp://ftp-
private.ncbi.nlm.nih.gov/pubchem/.fetch/1064385222466625960.sdf.gz</PCT-Download-
URL_url>
          </PCT-Download-URL>
        </PCT-OutputData_output_download-url>
      </PCT-OutputData_output>
    </PCT-OutputData>
  </PCT-Data_output>
</PCT-Data>
```

You would parse out the URL from the *<PCT-Download-URL_url>* tag, and then use a tool of your choice to connect to that URL to retrieve the actual requested data.

3.2. Query Tasks.

The *<PCT-Query>* object may be used to perform queries against PubChem data that are not possible using Entrez. The *<PCT-Query>* object consists of a series of queries and a database to query against. One must be careful when formulating queries to select compatible database and query types, as outlined in the documentation of each query task type. For example, you will not want to perform a chemical similarity search on a list of bioassay identifiers (AIDs), since chemical searches may only be performed on compound identifiers (CIDs).

The *<PCT-Query>* object can perform multiple queries in a single request. The semantic of multiple queries in a single task is to "AND" the result between queries, which is to say that the resulting list of identifiers will satisfy all queries requested. Sometimes it is best to perform multiple search tasks individually rather than in a single task, unless otherwise noted in the task documentation.

3.2.1. Chemical Structure Query Tasks.

To perform PubChem Compound structure searches using PUG, you will need to make a request using a *<PCT-Query>* object. Chemical structure search tasks use the query objects *<PCT-QueryCompoundCS>* and *<PCT-QueryCompoundEL>*. You may submit a structure search by mixing and matching more than one of these two query types in a series. Furthermore, only the PubChem Compound ("pccompound") Entrez database may be specified in *<PCT-QueryUids>*, when performing chemical structure queries.

The *<PCT-QueryCompoundCS>* and *<PCT-QueryCompoundEL>* objects can encode many different types of chemical structure searches. To help you understand how to encode a structure search, please consider using the PubChem Structure Search web site. It has the ability to translate your structure search query into the XML necessary for use with PUG, and can be very helpful to demonstrate how to encode complex queries. The PubChem Structure Search system is located at the URL:

http://pubchem.ncbi.nlm.nih.gov/search/

Please note that the output result of a chemical structure search is an Entrez history key (see eUtils documentation). To obtain the list of compounds matching your query, you must use eUtils; more information on eUtils is below. There is currently a limit of two million compound identifiers returned by the structure search (through either PUG or the interactive web site).

Example: You wish to perform a chemical similarity search of CID 2244 at a Tanimoto similarity value of 80% with at most 300 results returned.

The initial HTTP POST to PUG to initiate the search would contain XML like:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_query>
        <PCT-Query>
          <PCT-Query_type>
            <PCT-QueryType>
              <PCT-QueryType_css>
                <PCT-QueryCompoundCS>
                  <PCT-QueryCompoundCS_query>
                    <PCT-QueryCompoundCS_query_data>2244</PCT-
QueryCompoundCS_query_data>
                  </PCT-QueryCompoundCS_query>
                  <PCT-QueryCompoundCS_type>
                    <PCT-QueryCompoundCS_type_similar>
                      <PCT-CSSimilarity>
                        <PCT-CSSimilarity_threshold>80</PCT-
CSSimilarity_threshold>
                      </PCT-CSSimilarity>
                    </PCT-QueryCompoundCS_type_similar>
                  </PCT-QueryCompoundCS_type>
                  <PCT-QueryCompoundCS_results>300</PCT-QueryCompoundCS_results>
                </PCT-QueryCompoundCS>
              </PCT-QueryType_css>
            </PCT-QueryType>
          </PCT-Query_type>
        </PCT-Query>
      </PCT-InputData_query>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

If the request is processed and started successfully, PUG would respond with a waiting message and request id, for example:

```
<PCT-Data>
  <PCT-Data_output>
    <PCT-OutputData>
      <PCT-OutputData_status>
        <PCT-Status-Message>
          <PCT-Status-Message_status>
            <PCT-Status value="success"/>
          </PCT-Status-Message_status>
        </PCT-Status-Message>
      </PCT-OutputData_status>
      <PCT-OutputData_output>
        <PCT-OutputData_output_waiting>
          <PCT-Waiting>
```

```
                <PCT-Waiting_reqid>271473836860076709</PCT-Waiting_reqid>
                <PCT-Waiting_message>Structure search job was submitted</PCT-
Waiting_message>
              </PCT-Waiting>
            </PCT-OutputData_output_waiting>
          </PCT-OutputData_output>
        </PCT-OutputData>
      </PCT-Data_output>
    </PCT-Data>
```

You would then use this request ID, "`271473836860076709`" in this case, to "poll" PUG for the status of the request:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_request>
        <PCT-Request>
          <PCT-Request_reqid>271473836860076709</PCT-Request_reqid>
          <PCT-Request_type value="status"/>
        </PCT-Request>
      </PCT-InputData_request>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

If the search is still running, you would get another waiting message as above, and you would then need to poll again after a reasonable interval. If the search task is completed, PUG would give an Entrez history key for the resulting CID (compound identifier) list:

```
  <PCT-Data>
    <PCT-Data_output>
      <PCT-OutputData>
        <PCT-OutputData_status>
          <PCT-Status-Message>
            <PCT-Status-Message_status>
              <PCT-Status value="success"/>
            </PCT-Status-Message_status>
            <PCT-Status-Message_message>Your search has already been
  completed.</PCT-Status-Message_message>
          </PCT-Status-Message>
        </PCT-OutputData_status>
        <PCT-OutputData_output>
          <PCT-OutputData_output_entrez>
            <PCT-Entrez>
              <PCT-Entrez_db>pccompound</PCT-Entrez_db>
              <PCT-Entrez_query-key>1</PCT-Entrez_query-key>
              <PCT-Entrez_webenv>
0Hm9YDD1X4wor4nONvSWx9vkKmEqFXTiq84JO47pgxmSw_cIuDBVcG46Yr@2B5C47D162BBD720_0008SID
              </PCT-Entrez_webenv>
            </PCT-Entrez>
          </PCT-OutputData_output_entrez>
        </PCT-OutputData_output>
      </PCT-OutputData>
    </PCT-Data_output>
  </PCT-Data>
```

More information on using Entrez history (and eUtils) to retrieve hit lists is below.

If for some reason your initial query cannot be properly interpreted, PUG would respond with an error message with some indication of the problem encountered:

```
<PCT-Data>
  <PCT-Data_output>
    <PCT-OutputData>
      <PCT-OutputData_status>
        <PCT-Status-Message>
          <PCT-Status-Message_status>
```

```
              <PCT-Status value="data-error"/>
            </PCT-Status-Message_status>
            <PCT-Status-Message_message>Programatic Error:Non-decodeable query
specified. Input a valid SMILE/SMARTS or a CID.</PCT-Status-Message_message>
          </PCT-Status-Message>
        </PCT-OutputData_status>
      </PCT-OutputData>
    </PCT-Data_output>
  </PCT-Data>
```

If you wish to cancel a queued or running request, you would send to PUG:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_request>
        <PCT-Request>
          <PCT-Request_reqid>271473836860076709</PCT-Request_reqid>
          <PCT-Request_type value="cancel"/>
        </PCT-Request>
      </PCT-InputData_request>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

And when PUG cancels your task, you would get back:

```
<PCT-Data>
  <PCT-Data_output>
    <PCT-OutputData>
      <PCT-OutputData_status>
        <PCT-Status-Message>
          <PCT-Status-Message_status>
            <PCT-Status value="running"/>
          </PCT-Status-Message_status>
          <PCT-Status-Message_message>Your search will be stopped, please
wait...</PCT-Status-Message_message>
        </PCT-Status-Message>
      </PCT-OutputData_status>
      <PCT-OutputData_output>
        <PCT-OutputData_output_waiting>
          <PCT-Waiting>
            <PCT-Waiting_reqid>271473836860076709</PCT-Waiting_reqid>
            <PCT-Waiting_message>Your search will be stopped, please wait...</PCT-
Waiting_message>
          </PCT-Waiting>
        </PCT-OutputData_output_waiting>
      </PCT-OutputData_output>
    </PCT-OutputData>
  </PCT-Data_output>
</PCT-Data>
```

3.2.2. PubChem BioAssay Data Query and Download.

To perform PubChem BioAssay Data Download using PUG, you will need to generate an input XML file. This XML file has information about: what to download [being data for either particular PubChem BioAssay identifiers (AIDs), particular AIDs and PubChem Compound identifiers (CIDs), or particular AIDs and PubChem Substance identifiers (SIDs)]; the download format (XML, ASN.1, or CSV); and the output dataset type (complete or concise).

One way to generate the PUG input XML file is to use the "Save PUG Query" function provided by the PubChem BioAssay data select tool (e.g., using the URL:

http://pubchem.ncbi.nlm.nih.gov/assay/assay.cgi?aid=430,431&q=t

where "aid=…" specifies a list of BioAssays. With this tool, one can define the query options for retrieving BioAssay data and save the XML query file for use in PUG. Six examples demonstrating the use of the BioAssay download tool are shown below.

Example 1: You wish to download the complete BioAssay Data Table for AIDs 523 and 820 in CSV format. The initial HTTP POST to PUG to initiate the download would contain XML like:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_query>
        <PCT-Query>
          <PCT-Query_type>
            <PCT-QueryType>
              <PCT-QueryType_bas>
                <PCT-QueryAssayData>
                  <PCT-QueryAssayData_output value="csv">3</PCT-
QueryAssayData_output>
                  <PCT-QueryAssayData_id>
                    <PCT-ID-List>
                      <PCT-ID-List_db>pcassay</PCT-ID-List_db>
                      <PCT-ID-List_uids>
                        <PCT-ID-List_uids_E>523</PCT-ID-List_uids_E>
                        <PCT-ID-List_uids_E>820</PCT-ID-List_uids_E>
                      </PCT-ID-List_uids>
                    </PCT-ID-List>
                  </PCT-QueryAssayData_id>
                  <PCT-QueryAssayData_dataset value="complete">0</PCT-
QueryAssayData_dataset>
                </PCT-QueryAssayData>
              </PCT-QueryType_bas>
            </PCT-QueryType>
          </PCT-Query_type>
        </PCT-Query>
      </PCT-InputData_query>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

Example 2: You wish to download the concise BioAssay Data Table for AIDs 523 and 820 but just for the CIDs 3243128 and 3240114 in PubChem XML format. The initial HTTP POST to PUG to initiate the download would contain XML like:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_query>
        <PCT-Query>
          <PCT-Query_type>
            <PCT-QueryType>
              <PCT-QueryType_bas>
                <PCT-QueryAssayData>
                  <PCT-QueryAssayData_output value="assay-xml">1</PCT-
QueryAssayData_output>
                  <PCT-QueryAssayData_id>
                    <PCT-ID-List>
                      <PCT-ID-List_db>pcassay</PCT-ID-List_db>
                      <PCT-ID-List_uids>
                        <PCT-ID-List_uids_E>523</PCT-ID-List_uids_E>
                        <PCT-ID-List_uids_E>820</PCT-ID-List_uids_E>
                      </PCT-ID-List_uids>
                    </PCT-ID-List>
                    <PCT-ID-List>
```

```
                      <PCT-ID-List_db>pccompound</PCT-ID-List_db>
                      <PCT-ID-List_uids>
                        <PCT-ID-List_uids_E>3243128</PCT-ID-List_uids_E>
                        <PCT-ID-List_uids_E>3240114</PCT-ID-List_uids_E>
                      </PCT-ID-List_uids>
                    </PCT-ID-List>
                  </PCT-QueryAssayData_id>
                  <PCT-QueryAssayData_dataset value="concise">1</PCT-
QueryAssayData_dataset>
                </PCT-QueryAssayData>
              </PCT-QueryType_bas>
            </PCT-QueryType>
          </PCT-Query_type>
        </PCT-Query>
      </PCT-InputData_query>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

Example 3: You wish to download the concise BioAssay Data Table for AIDs 523 and 820 but only for the SIDs 16952359 and 16952361 in PubChem ASN.1 format. The initial HTTP POST to PUG to initiate the download would contain XML like:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_query>
        <PCT-Query>
          <PCT-Query_type>
            <PCT-QueryType>
              <PCT-QueryType_bas>
                <PCT-QueryAssayData>
                  <PCT-QueryAssayData_output value="assay-asn">2</PCT-
QueryAssayData_output>
                  <PCT-QueryAssayData_id>
                    <PCT-ID-List>
                      <PCT-ID-List_db>pcassay</PCT-ID-List_db>
                      <PCT-ID-List_uids>
                        <PCT-ID-List_uids_E>523</PCT-ID-List_uids_E>
                        <PCT-ID-List_uids_E>820</PCT-ID-List_uids_E>
                      </PCT-ID-List_uids>
                    </PCT-ID-List>
                    <PCT-ID-List>
                      <PCT-ID-List_db>pcsubstance</PCT-ID-List_db>
                      <PCT-ID-List_uids>
                        <PCT-ID-List_uids_E>16952359</PCT-ID-List_uids_E>
                        <PCT-ID-List_uids_E>16952361</PCT-ID-List_uids_E>
                      </PCT-ID-List_uids>
                    </PCT-ID-List>
                  </PCT-QueryAssayData_id>
                  <PCT-QueryAssayData_dataset value="concise">1</PCT-
QueryAssayData_dataset>
                </PCT-QueryAssayData>
              </PCT-QueryType_bas>
            </PCT-QueryType>
          </PCT-Query_type>
        </PCT-Query>
      </PCT-InputData_query>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

Example 4: You wish to download the complete BioAssay Data Table for AID 523 but only for those substances with "IC-50" value (the readout provided through

TID 2) between 1 and 10 uM in CSV format. The initial HTTP POST to PUG to initiate the download would contain XML like:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_query>
        <PCT-Query>
          <PCT-Query_type>
            <PCT-QueryType>
              <PCT-QueryType_bas>
                <PCT-QueryAssayData>
                  <PCT-QueryAssayData_output value="csv">3</PCT-
QueryAssayData_output>
                  <PCT-QueryAssayData_id>
                    <PCT-ID-List>
                      <PCT-ID-List_db>pcassay</PCT-ID-List_db>
                      <PCT-ID-List_uids>
                        <PCT-ID-List_uids_E>523</PCT-ID-List_uids_E>
                      </PCT-ID-List_uids>
                    </PCT-ID-List>
                  </PCT-QueryAssayData_id>
                  <PCT-QueryAssayData_dataset value="complete">0</PCT-
QueryAssayData_dataset>
                  <PCT-QueryAssayData_readouts>
                    <PCT-Assay-Readout>
                      <PCT-Assay-Readout_aid>523</PCT-Assay-Readout_aid>
                      <PCT-Assay-Readout_query>
                        <PCT-Assay-Readout-Query>
                          <PCT-Assay-Readout-Query_fquery>
                            <PCT-Assay-Readout-Float-Query>
                              <PCT-Assay-Readout-Float-Query_tid>2</PCT-Assay-
Readout-Float-Query_tid>
                              <PCT-Assay-Readout-Float-Query_lower>1</PCT-Assay-
Readout-Float-Query_lower>
                              <PCT-Assay-Readout-Float-Query_upper>10</PCT-Assay-
Readout-Float-Query_upper>
                            </PCT-Assay-Readout-Float-Query>
                          </PCT-Assay-Readout-Query_fquery>
                        </PCT-Assay-Readout-Query>
                      </PCT-Assay-Readout_query>
                    </PCT-Assay-Readout>
                  </PCT-QueryAssayData_readouts>
                </PCT-QueryAssayData>
              </PCT-QueryType_bas>
            </PCT-QueryType>
          </PCT-Query_type>
        </PCT-Query>
      </PCT-InputData_query>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

Example 5: You wish to download the complete BioAssay Data Table of AID 523 with "Activity Outcome" as "Active" and "Activity Score" value between 10 and 40 in CSV format. The initial HTTP POST to PUG to initiate the download would contain XML like:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_query>
        <PCT-Query>
          <PCT-Query_type>
            <PCT-QueryType>
              <PCT-QueryType_bas>
                <PCT-QueryAssayData>
```

```
                  <PCT-QueryAssayData_output value="csv">3</PCT-
QueryAssayData_output>
                  <PCT-QueryAssayData_id>
                    <PCT-ID-List>
                      <PCT-ID-List_db>pcassay</PCT-ID-List_db>
                      <PCT-ID-List_uids>
                        <PCT-ID-List_uids_E>523</PCT-ID-List_uids_E>
                      </PCT-ID-List_uids>
                    </PCT-ID-List>
                  </PCT-QueryAssayData_id>
                  <PCT-QueryAssayData_dataset value="complete">0</PCT-
QueryAssayData_dataset>
                  <PCT-QueryAssayData_readouts>
                    <PCT-Assay-Readout>
                      <PCT-Assay-Readout_aid>523</PCT-Assay-Readout_aid>
                      <PCT-Assay-Readout_query>
                        <PCT-Assay-Readout-Query>
                          <PCT-Assay-Readout-Query_outcome>
                            <PCT-Assay-Activity-Outcome-Query
value="active">2</PCT-Assay-Activity-Outcome-Query>
                          </PCT-Assay-Readout-Query_outcome>
                        </PCT-Assay-Readout-Query>
                        <PCT-Assay-Readout-Query>
                          <PCT-Assay-Readout-Query_score>
                            <PCT-Assay-Activity-Score-Query>
                              <PCT-Assay-Activity-Score-Query_lower>10</PCT-Assay-
Activity-Score-Query_lower>
                              <PCT-Assay-Activity-Score-Query_upper>40</PCT-Assay-
Activity-Score-Query_upper>
                            </PCT-Assay-Activity-Score-Query>
                          </PCT-Assay-Readout-Query_score>
                        </PCT-Assay-Readout-Query>
                      </PCT-Assay-Readout_query>
                    </PCT-Assay-Readout>
                  </PCT-QueryAssayData_readouts>
                </PCT-QueryAssayData>
              </PCT-QueryType_bas>
            </PCT-QueryType>
          </PCT-Query_type>
        </PCT-Query>
      </PCT-InputData_query>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

Example 6: You wish to download the complete BioAssay Data Table of AID 523 with "IC-50" value (the readout provided through TID 2) between 1 and 10 micromolar in CSV format. You want to obtain just two bioassay (TID) columns (TID 1: IC50 Qualifier and TID 2: IC50). The initial HTTP POST to PUG to initiate the download would contain XML like:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_query>
        <PCT-Query>
          <PCT-Query_type>
            <PCT-QueryType>
              <PCT-QueryType_bas>
                <PCT-QueryAssayData>
                  <PCT-QueryAssayData_output value="csv">3</PCT-
QueryAssayData_output>
                  <PCT-QueryAssayData_id>
                    <PCT-ID-List>
                      <PCT-ID-List_db>pcassay</PCT-ID-List_db>
                      <PCT-ID-List_uids>
                        <PCT-ID-List_uids_E>523</PCT-ID-List_uids_E>
                      </PCT-ID-List_uids>
                    </PCT-ID-List>
                  </PCT-QueryAssayData_id>
                  <PCT-QueryAssayData_dataset value="complete">0</PCT-
QueryAssayData_dataset>
                  <PCT-QueryAssayData_readouts>
                    <PCT-Assay-Readout>
                      <PCT-Assay-Readout_aid>523</PCT-Assay-Readout_aid>
                      <PCT-Assay-Readout_retrieve>
                        <PCT-Assay-Readout-Retrieve>
                          <PCT-Assay-Readout-Retrieve_tid>2</PCT-Assay-Readout-
Retrieve_tid>
                          <PCT-Assay-Readout-Retrieve_data-type
value="int">1</PCT-Assay-Readout-Retrieve_data-type>
                        </PCT-Assay-Readout-Retrieve>
                        <PCT-Assay-Readout-Retrieve>
                          <PCT-Assay-Readout-Retrieve_tid>1</PCT-Assay-Readout-
Retrieve_tid>
                          <PCT-Assay-Readout-Retrieve_data-type
value="string">4</PCT-Assay-Readout-Retrieve_data-type>
                        </PCT-Assay-Readout-Retrieve>
                      </PCT-Assay-Readout_retrieve>
                      <PCT-Assay-Readout_query>
                        <PCT-Assay-Readout-Query>
                          <PCT-Assay-Readout-Query_fquery>
                            <PCT-Assay-Readout-Float-Query>
                              <PCT-Assay-Readout-Float-Query_tid>2</PCT-Assay-
Readout-Float-Query_tid>
                              <PCT-Assay-Readout-Float-Query_lower>1</PCT-Assay-
Readout-Float-Query_lower>
                              <PCT-Assay-Readout-Float-Query_upper>10</PCT-Assay-
Readout-Float-Query_upper>
                            </PCT-Assay-Readout-Float-Query>
                          </PCT-Assay-Readout-Query_fquery>
                        </PCT-Assay-Readout-Query>
                      </PCT-Assay-Readout_query>
                    </PCT-Assay-Readout>
                  </PCT-QueryAssayData_readouts>
                </PCT-QueryAssayData>
              </PCT-QueryType_bas>
            </PCT-QueryType>
          </PCT-Query_type>
        </PCT-Query>
      </PCT-InputData_query>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

If one of the above requests is processed and started successfully, PUG would respond with a waiting message and request id, for example:

```
<PCT-Data>
  <PCT-Data_output>
    <PCT-OutputData>
      <PCT-OutputData_status>
        <PCT-Status-Message>
          <PCT-Status-Message_status>
            <PCT-Status value="running"/>
          </PCT-Status-Message_status>
        </PCT-Status-Message>
      </PCT-OutputData_status>
      <PCT-OutputData_output>
        <PCT-OutputData_output_waiting>
          <PCT-Waiting>
            <PCT-Waiting_reqid>336289408724569820</PCT-Waiting_reqid>
          </PCT-Waiting>
        </PCT-OutputData_output_waiting>
      </PCT-OutputData_output>
    </PCT-OutputData>
  </PCT-Data_output>
```

```
</PCT-Data>
```

You would then use this request ID, "`336289408724569820`" in this case, to "poll" PUG for the status of the request:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_request>
        <PCT-Request>
          <PCT-Request_reqid>336289408724569820</PCT-Request_reqid>
          <PCT-Request_type value="status"/>
        </PCT-Request>
      </PCT-InputData_request>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

If the download is still running, you would get another waiting message as above, and you would then need to poll again after a reasonable interval. If the download task is completed, PUG would give a ftp URL, where the requested data is available:

```
<PCT-Data>
  <PCT-Data_output>
    <PCT-OutputData>
      <PCT-OutputData_status>
        <PCT-Status-Message>
          <PCT-Status-Message_status>
            <PCT-Status value="success"/>
          </PCT-Status-Message_status>
        </PCT-Status-Message>
      </PCT-OutputData_status>
      <PCT-OutputData_output>
        <PCT-OutputData_output_download-url>
          <PCT-Download-URL>
            <PCT-Download-URL_url>ftp://ftp-
private.ncbi.nlm.nih.gov/pubchem/.fetch/336289408724569820.csv</PCT-Download-
URL_url>
          </PCT-Download-URL>
        </PCT-OutputData_output_download-url>
      </PCT-OutputData_output>
    </PCT-OutputData>
  </PCT-Data_output>
</PCT-Data>
```

If for some reason your initial query cannot be properly interpreted, PUG would respond with an error message with some indication of the problem encountered:

```
<PCT-Data>
  <PCT-Data_output>
    <PCT-OutputData>
      <PCT-OutputData_status>
        <PCT-Status-Message>
          <PCT-Status-Message_status>
            <PCT-Status value="server-error"/>
          </PCT-Status-Message_status>
          <PCT-Status-Message_message>Status: server-error. Contact
info@ncbi.nlm.nih.gov for assistance. Please include your request ID if
possible.</PCT-Status-Message_message>
        </PCT-Status-Message>
      </PCT-OutputData_status>
    </PCT-OutputData>
  </PCT-Data_output>
</PCT-Data>
```

If you wish to cancel a queued or running request, you would send to PUG:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_request>
        <PCT-Request>
          <PCT-Request_reqid>336289408724569820</PCT-Request_reqid>
          <PCT-Request_type value="cancel"/>
        </PCT-Request>
      </PCT-InputData_request>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

And when PUG cancels your task, you would get back:

```
<PCT-Data>
  <PCT-Data_output>
    <PCT-OutputData>
      <PCT-OutputData_status>
        <PCT-Status-Message>
          <PCT-Status-Message_status>
            <PCT-Status value="running"/>
          </PCT-Status-Message_status>
          <PCT-Status-Message_message>Request cancelled</PCT-Status-
Message_message>
        </PCT-Status-Message>
      </PCT-OutputData_status>
      <PCT-OutputData_output>
        <PCT-OutputData_output_waiting>
          <PCT-Waiting>
            <PCT-Waiting_reqid>336289408724569820</PCT-Waiting_reqid>
          </PCT-Waiting>
        </PCT-OutputData_output_waiting>
      </PCT-OutputData_output>
    </PCT-OutputData>
  </PCT-Data_output>
</PCT-Data>
```

3.3. PubChem Standardization Tasks.

The PubChem Standardization service allows you to standardize the representation of a chemical structure using a *<PCT-Standardize>* sub-object. PubChem uses a normalization procedure on all PubChem substance records to remove variation due to different representations of functional groups, tautomeric or resonance forms, etc., to create the PubChem Compound database, which contains the unique chemical structures in the PubChem Substance database. This procedure verifies and validates that a chemical structure is reasonable (to a certain degree) through examination of the atoms and their valence and involves a valence-bond canonicalization processing for tautomer invariance. The input to structure standardization is a chemical structure and the output is either a failure message or a chemical structure. To use this service, you will need to specify an input structure and its format. You also need to specify the output format you desire. This service operates on only a single structure at a time.

Example: You would like to standardize the representation of guanine input in SMILES format and output in SDF format.

The typical flow of information is as follows. First, the initial input XML is sent to PUG via HTTP POST. Note the input data container with the download request and uid and format options:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_standardize>
        <PCT-Standardize>
          <PCT-Standardize_structure>
            <PCT-Structure>
              <PCT-Structure_structure>
                <PCT-Structure_structure_string>C1=NC2=C(N1)C(=O)N=C(N2)N</PCT-
Structure_structure_string>
              </PCT-Structure_structure>
              <PCT-Structure_format>
                <PCT-StructureFormat value="smiles"/>
              </PCT-Structure_format>
            </PCT-Structure>
          </PCT-Standardize_structure>
          <PCT-Standardize_oformat>
            <PCT-StructureFormat value="smiles"/>
          </PCT-Standardize_oformat>
        </PCT-Standardize>
      </PCT-InputData_standardize>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

If the request is small and finishes very quickly, you may get a final URL right away (see further below). But usually PUG will respond initially with a waiting message and a request ID (*<PCT-Waiting_reqid>*) such as:

```
<PCT-Data>
  <PCT-Data_output>
    <PCT-OutputData>
      <PCT-OutputData_status>
        <PCT-Status-Message>
          <PCT-Status-Message_status>
            <PCT-Status value="success"/>
          </PCT-Status-Message_status>
        </PCT-Status-Message>
      </PCT-OutputData_status>
      <PCT-OutputData_output>
        <PCT-OutputData_output_waiting>
          <PCT-Waiting>
            <PCT-Waiting_reqid>402936103567975582</PCT-Waiting_reqid>
          </PCT-Waiting>
        </PCT-OutputData_output_waiting>
      </PCT-OutputData_output>
    </PCT-OutputData>
  </PCT-Data_output>
</PCT-Data>
```

You would then parse out this request id, being "`402936103567975582`", in this case, and use this id to "poll" PUG on the status of the request, composing an XML message like:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_request>
        <PCT-Request>
          <PCT-Request_reqid>402936103567975582</PCT-Request_reqid>
          <PCT-Request_type value="status"/>
        </PCT-Request>
      </PCT-InputData_request>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

Note that here the request type "status" is used; there is also the request type "cancel" that you may use to cancel a running job.

If the request is still running, you well get back another waiting message as above, and then you would poll again after some reasonable interval. If the request is finished, you will get a final result message like:

```
  <PCT-Data>
    <PCT-Data_output>
      <PCT-OutputData>
        <PCT-OutputData_status>
          <PCT-Status-Message>
            <PCT-Status-Message_status>
              <PCT-Status value="success"/>
            </PCT-Status-Message_status>
          </PCT-Status-Message>
        </PCT-OutputData_status>
        <PCT-OutputData_output>
          <PCT-OutputData_output_structure>
            <PCT-Structure>
              <PCT-Structure_structure>
                <PCT-Structure_structure_string>C1=NC2=C(N1)C(=O)N=C(N2)N</PCT-
Structure_structure_string>
              </PCT-Structure_structure>
              <PCT-Structure_format>
                <PCT-StructureFormat value="smiles"/>
              </PCT-Structure_format>
            </PCT-Structure>
          </PCT-OutputData_output_structure>
        </PCT-OutputData_output>
      </PCT-OutputData>
    </PCT-Data_output>
  </PCT-Data>
```

You would parse out the output from the *<PCT-Download-URL_url>* tag to retrieve the standardized structure.

4. PUG, NCBI eUtils, and Entrez History.

NCBI's Entrez integrates the scientific literature, DNA and protein sequence databases, 3D protein structure and protein domain data, population study datasets, expression data, assemblies of complete genomes, taxonomic information, and PubChem Compound, Substance, and BioAssay databases (among others) into a tightly interlinked system. It is a retrieval system designed for searching its linked databases. Entrez history provides a record of the searches performed during a search session. PubChem communicates with Entrez history through Entrez Programming Utilities (eUtils) to enhance data analysis.

NCBI's eUtils are used extensively by PubChem services. Results from queries are often provided in the form of an Entrez history, which represents a list of database specific identifiers within the Entrez search system. These identifiers are, for example, your PubChem CIDs (compound identifiers). This allows you, the user, to interact with other Entrez databases and to perform hit list management tasks using eUtils, *e.g.*, to logically combine the results of different queries using AND, OR, or NOT operations. PubChem services typically accept an Entrez history as a means to provide a subset of identifiers as input, so that your query operates only on a subset of a PubChem database contents. Use of Entrez history can help you avoid sending and receiving (potentially) very large lists of identifiers. To learn more about eUtils, please visit the URL:

http://eutils.ncbi.nlm.nih.gov/entrez/query/static/eutils_help.html

Histories in Entrez are database specific. Each time an Entrez search is executed, the search terms, the time the search was executed, and the search results are numbered consecutively and saved automatically in Entrez history for that database. The history can be recalled at any time during a search session, but histories are lost after 8 hours of inactivity. There is also a running limit of 100 searches (across all databases) saved in any given session.

PUG is integrated with Entrez in that it may use Entrez history keys (also know as "webenv" keys) as both input and output, depending on the task. For example, structure search via PUG may return an Entrez history, and the resulting hit list can be retrieved as a list of CIDs using Entrez's eFetch utility. PUG can also take a history key as input, if you wanted to download the records resulting from either a prior structure search or a programmatic Entrez search via the eSearch utility. (See the eUtils documentation for more details.)

Entrez histories are referred to programmatically by the trio of a database name, a WebEnv string, and a query key number. You can see this in the example structure search above. The part of PUG's response that contains this information is the *<PCT-Entrez>* tag:

```
                <PCT-Entrez>
                  <PCT-Entrez_db>pccompound</PCT-Entrez_db>
                  <PCT-Entrez_query-key>1</PCT-Entrez_query-key>
                  <PCT-Entrez_webenv>
    0Hm9YDD1X4wor4nONvSWx9vkKmEqFXTiq84JO47pgxmSw_cIuDBVcG46Yr@2B5C47D162BBD720_0008SID
                  </PCT-Entrez_webenv>
                </PCT-Entrez>
```

This Entrez history information may be used in a variety of ways. If you want to view these hits on a regular web page, you can direct a browser to an URL as follows, which shows the results in HTML in the usual Entrez docsum format:

```
http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Select+from+History&WebEnvRq=1&d
b=pccompound&query_key=1&WebEnv=0Hm9YDD1X4wor4nONvSWx9vkKmEqFXTiq84JO47pgxmSw_cIuD
BVcG46Yr@2B5C47D162BBD720_0008SID
```

On the other hand, if you are writing an application and want to retrieve the hit list directly via HTTP, you can use eFetch with the same information, which can return the list in XML (with its own DTD/XSD that is not related to PUG's), for example:

```
http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?retmode=xml&rettype=uilis
t&WebEnvRq=1&db=pccompound&query_key=1&WebEnv=0Hm9YDD1X4wor4nONvSWx9vkKmEqFXTiq84J
O47pgxmSw_cIuDBVcG46Yr@2B5C47D162BBD720_0008SID
```

Finally, if you want to download the compounds from this search in, *e.g.*, SDF format with gzip compression, you would send PUG a request with the *<PCT-Entrez>* information instead of an explicit CID list. From here, the download process would continue as in the example above:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_download>
```

```
                <PCT-Download>
                  <PCT-Download_uids>
                    <PCT-QueryUids>
                      <PCT-QueryUids_entrez>
                        <PCT-Entrez>
                          <PCT-Entrez_db>pccompound</PCT-Entrez_db>
                          <PCT-Entrez_query-key>1</PCT-Entrez_query-key>
                          <PCT-Entrez_webenv>
      0Hm9YDD1X4wor4nONvSWx9vkKmEqFXTiq84JO47pgxmSw_cIuDBVcG46Yr@2B5C47D162BBD720_0008SID
                          </PCT-Entrez_webenv>
                        </PCT-Entrez>
                      </PCT-QueryUids_entrez>
                    </PCT-QueryUids>
                  </PCT-Download_uids>
                  <PCT-Download_format value="sdf"/>
                  <PCT-Download_compression value="gzip"/>
                </PCT-Download>
              </PCT-InputData_download>
            </PCT-InputData>
          </PCT-Data_input>
        </PCT-Data>
```

PUG and eUtils together make possible a wide variety of powerful programmatic data analysis tools for PubChem and other Entrez databases.

5. PUG and SOAP.

There is a SOAP wrapper for PUG. Documentation and a WSDL can be found at:

http://pubchem.ncbi.nlm.nih.gov/pug_soap/

The interface includes much of PUG's functionality, but with simplified functions that are accessible from GUI workflow applications and SOAP-aware programming languages.

6. FAQs

Some simple workflows help to illustrate the use of PUG. All PUG messages must be sent via a HTTP POST to the URL:

http://pubchem.ncbi.nlm.nih.gov/pug/pug.cgi

Scenario 1. I would like to retrieve the SMILES (in gzip compressed format) for a list of PubChem Compound CIDs: 1, 2, and 3.

Compose your PUG message:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_download>
        <PCT-Download>
          <PCT-Download_uids>
            <PCT-QueryUids>
              <PCT-QueryUids_ids>
                <PCT-ID-List>
                  <PCT-ID-List_db>pccompound</PCT-ID-List_db>
                  <PCT-ID-List_uids>
                    <PCT-ID-List_uids_E>1</PCT-ID-List_uids_E>
                    <PCT-ID-List_uids_E>2</PCT-ID-List_uids_E>
                    <PCT-ID-List_uids_E>3</PCT-ID-List_uids_E>
                  </PCT-ID-List_uids>
                </PCT-ID-List>
              </PCT-QueryUids_ids>
            </PCT-QueryUids>
```

```
              </PCT-Download_uids>
              <PCT-Download_format value="smiles"/>
              <PCT-Download_compression value="gzip"/>
            </PCT-Download>
          </PCT-InputData_download>
        </PCT-InputData>
      </PCT-Data_input>
    </PCT-Data>
```

PUG will send you back a response, for example:

```
    <PCT-Data>
      <PCT-Data_output>
        <PCT-OutputData>
          <PCT-OutputData_status>
            <PCT-Status-Message>
              <PCT-Status-Message_status>
                <PCT-Status value="success"/>
              </PCT-Status-Message_status>
            </PCT-Status-Message>
          </PCT-OutputData_status>
          <PCT-OutputData_output>
            <PCT-OutputData_output_waiting>
              <PCT-Waiting>
                <PCT-Waiting_reqid>402936103567975582</PCT-Waiting_reqid>
              </PCT-Waiting>
            </PCT-OutputData_output_waiting>
          </PCT-OutputData_output>
        </PCT-OutputData>
      </PCT-Data_output>
    </PCT-Data>
```

This response contains a request ID, being "402936103567975582". You will use this request ID to query PUG on the status of your request by composing and sending another PUG message, as such:

```
    <PCT-Data>
      <PCT-Data_input>
        <PCT-InputData>
          <PCT-InputData_request>
            <PCT-Request>
              <PCT-Request_reqid>402936103567975582</PCT-Request_reqid>
              <PCT-Request_type value="status"/>
            </PCT-Request>
          </PCT-InputData_request>
        </PCT-InputData>
      </PCT-Data_input>
    </PCT-Data>
```

If your request is still being processed, you will receive a response message as before. When your request completes, PUG will return an XML message like:

```
    <PCT-Data>
      <PCT-Data_output>
        <PCT-OutputData>
          <PCT-OutputData_status>
            <PCT-Status-Message>
              <PCT-Status-Message_status>
                <PCT-Status value="success"/>
              </PCT-Status-Message_status>
            </PCT-Status-Message>
          </PCT-OutputData_status>
          <PCT-OutputData_output>
            <PCT-OutputData_output_download-url>
              <PCT-Download-URL>
                <PCT-Download-URL_url>ftp://ftp-
    private.ncbi.nlm.nih.gov/pubchem/.fetch/656213441898678492.txt.gz</PCT-Download-
    URL_url>
              </PCT-Download-URL>
```

```
                </PCT-OutputData_output_download-url>
              </PCT-OutputData_output>
            </PCT-OutputData>
          </PCT-Data_output>
        </PCT-Data>
```

The PUG response gives you an URL where you may retrieve your results:
ftp://ftp-private.ncbi.nlm.nih.gov/pubchem/.fetch/656213441898678492.txt.gz

Scenario 2a. I have a SMILES of L-tyrosine and I would like to get back an SDF file containing the PubChem Compound record(s) exactly matching this structure (with the same stereo and isotopes).

The workflow for this scenario is very similar to **Scenario 1**, where you . Borrowing the workflow from the first scenario, the PUG message you HTTP POST would be:

```
        <PCT-Data>
          <PCT-Data_input>
            <PCT-InputData>
              <PCT-InputData_query>
                <PCT-Query>
                  <PCT-Query_type>
                    <PCT-QueryType>
                      <PCT-QueryType_css>
                        <PCT-QueryCompoundCS>
                          <PCT-QueryCompoundCS_query>
                            <PCT-
        QueryCompoundCS_query_data>C1=CC(=CC=C1C[C@@H](C(=O)O)N)O</PCT-
        QueryCompoundCS_query_data>
                          </PCT-QueryCompoundCS_query>
                          <PCT-QueryCompoundCS_type>
                            <PCT-QueryCompoundCS_type_identical>
                              <PCT-CSIdentity value="same-stereo-isotope">5</PCT-
        CSIdentity>
                            </PCT-QueryCompoundCS_type_identical>
                          </PCT-QueryCompoundCS_type>
                          <PCT-QueryCompoundCS_results>2000000</PCT-
        QueryCompoundCS_results>
                        </PCT-QueryCompoundCS>
                      </PCT-QueryType_css>
                    </PCT-QueryType>
                  </PCT-Query_type>
                </PCT-Query>
              </PCT-InputData_query>
            </PCT-InputData>
          </PCT-Data_input>
        </PCT-Data>
```

The eventual result of the search will look something like:

```
        <PCT-Data>
          <PCT-Data_output>
            <PCT-OutputData>
              <PCT-OutputData_status>
                <PCT-Status-Message>
                  <PCT-Status-Message_status>
                    <PCT-Status value="success"/>
                  </PCT-Status-Message_status>
                  <PCT-Status-Message_message>Your search has completed
        successfully!</PCT-Status-Message_message>
                </PCT-Status-Message>
              </PCT-OutputData_status>
              <PCT-OutputData_output>
                <PCT-OutputData_output_entrez>
                  <PCT-Entrez>
                    <PCT-Entrez_db>pccompound</PCT-Entrez_db>
                    <PCT-Entrez_query-key>3</PCT-Entrez_query-key>
                    <PCT-Entrez_webenv>0hJ--NzxiSKFxJzc4SnMb5PvxBP8HKJvZ-2s-
        XE19WBZSHG0xIO_k_xPrU@1FBE6DE17A397ED0_0011SID</PCT-Entrez_webenv>
                  </PCT-Entrez>
                </PCT-OutputData_output_entrez>
              </PCT-OutputData_output>
            </PCT-OutputData>
          </PCT-Data_output>
        </PCT-Data>
```

When the query is complete, the result is an Entrez query key and webenv. The query key identifies the query result and the webenv provides your session identifier. You use this Entrez query key and webenv as your source of CIDs to compose another PUG message to download the gzipped compressed SDF file of the query hits:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_download>
        <PCT-Download>
          <PCT-Download_uids>
            <PCT-QueryUids>
              <PCT-QueryUids_entrez>
                <PCT-Entrez>
                  <PCT-Entrez_db>pccompound</PCT-Entrez_db>
                  <PCT-Entrez_query-key>3</PCT-Entrez_query-key>
                  <PCT-Entrez_webenv>0hJ--NzxiSKFxJzc4SnMb5PvxBP8HKJvZ-2s-
XE19WBZSHG0xIO_k_xPrU@1FBE6DE17A397ED0_0011SID</PCT-Entrez_webenv>
                </PCT-Entrez>
              </PCT-QueryUids_entrez>
            </PCT-QueryUids>
          </PCT-Download_uids>
          <PCT-Download_format value="sdf"/>
          <PCT-Download_compression value="gzip"/>
        </PCT-Download>
      </PCT-InputData_download>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

The final result, as in the first scenario, will contain an URL to the results containing CID 6057:

```
<PCT-Data>
  <PCT-Data_output>
    <PCT-OutputData>
      <PCT-OutputData_status>
        <PCT-Status-Message>
          <PCT-Status-Message_status>
            <PCT-Status value="success"/>
          </PCT-Status-Message_status>
        </PCT-Status-Message>
      </PCT-OutputData_status>
      <PCT-OutputData_output>
        <PCT-OutputData_output_download-url>
          <PCT-Download-URL>
            <PCT-Download-URL_url>ftp://ftp-
private.ncbi.nlm.nih.gov/pubchem/.fetch/693081357064045880.sdf.gz</PCT-Download-
URL_url>
          </PCT-Download-URL>
        </PCT-OutputData_output_download-url>
      </PCT-OutputData_output>
    </PCT-OutputData>
  </PCT-Data_output>
</PCT-Data>
```

Scenario 2b. I have a SMILES of L-tyrosine and I would like to get back an SDF file containing the PubChem Compound records containing the same isotopes (but stereo can be different).

This scenario is identical to **Scenario 2a**, except that the initial PUG message would look like:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_query>
        <PCT-Query>
          <PCT-Query_type>
            <PCT-QueryType>
              <PCT-QueryType_css>
                <PCT-QueryCompoundCS>
                  <PCT-QueryCompoundCS_query>
                    <PCT-
QueryCompoundCS_query_data>C1=CC(=CC=C1C[C@@H](C(=O)O)N)O</PCT-
QueryCompoundCS_query_data>
                  </PCT-QueryCompoundCS_query>
                  <PCT-QueryCompoundCS_type>
                    <PCT-QueryCompoundCS_type_identical>
                      <PCT-CSIdentity value="same-isotope">4</PCT-CSIdentity>
                    </PCT-QueryCompoundCS_type_identical>
                  </PCT-QueryCompoundCS_type>
                  <PCT-QueryCompoundCS_results>2000000</PCT-
QueryCompoundCS_results>
                </PCT-QueryCompoundCS>
              </PCT-QueryType_css>
            </PCT-QueryType>
          </PCT-Query_type>
        </PCT-Query>
      </PCT-InputData_query>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

The final result would contain three records, being CIDs 6057, 1153, and 71098.

Scenario 2c. I have a SMILES of L-tyrosine and I would like to get back an SDF file containing the PubChem Compound records that have a similarity of 95%.

This scenario is identical to **Scenario 2a**, except that the initial PUG message would look like:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_query>
        <PCT-Query>
          <PCT-Query_type>
            <PCT-QueryType>
              <PCT-QueryType_css>
                <PCT-QueryCompoundCS>
                  <PCT-QueryCompoundCS_query>
                    <PCT-
QueryCompoundCS_query_data>C1=CC(=CC=C1C[C@@H](C(=O)O)N)O</PCT-
QueryCompoundCS_query_data>
                  </PCT-QueryCompoundCS_query>
                  <PCT-QueryCompoundCS_type>
                    <PCT-QueryCompoundCS_type_similar>
                      <PCT-CSSimilarity>
                        <PCT-CSSimilarity_threshold>95</PCT-
CSSimilarity_threshold>
                      </PCT-CSSimilarity>
                    </PCT-QueryCompoundCS_type_similar>
                  </PCT-QueryCompoundCS_type>
                  <PCT-QueryCompoundCS_results>2000000</PCT-
QueryCompoundCS_results>
                </PCT-QueryCompoundCS>
              </PCT-QueryType_css>
            </PCT-QueryType>
          </PCT-Query_type>
        </PCT-Query>
      </PCT-InputData_query>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

The final result would contain (currently) the SDF records for 191 CIDs.

Scenario 2d. How do I query using a molecular formula C_2H_7O and get back an SDF file containing the PubChem Compound records matching exactly?

This scenario is identical to **Scenario 2a**, except that the initial PUG message would look like:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_query>
        <PCT-Query>
          <PCT-Query_type>
            <PCT-QueryType>
              <PCT-QueryType_css>
                <PCT-QueryCompoundCS>
                  <PCT-QueryCompoundCS_query>
                    <PCT-QueryCompoundCS_query_data>C2H7NO</PCT-
QueryCompoundCS_query_data>
                  </PCT-QueryCompoundCS_query>
                  <PCT-QueryCompoundCS_type>
                    <PCT-QueryCompoundCS_type_formula>
                      <PCT-CSMolFormula></PCT-CSMolFormula>
                    </PCT-QueryCompoundCS_type_formula>
                  </PCT-QueryCompoundCS_type>
                  <PCT-QueryCompoundCS_results>2000000</PCT-
QueryCompoundCS_results>
                </PCT-QueryCompoundCS>
              </PCT-QueryType_css>
            </PCT-QueryType>
          </PCT-Query_type>
        </PCT-Query>
      </PCT-InputData_query>
    </PCT-InputData>
  </PCT-Data_input>
</PCT-Data>
```

The final result would contain (currently) the SDF records for 20 CIDs.

Scenario 3. How do I retrieve the SDF file of all PubChem Compound records within the mass range 100.00 to 100.01 atomic mass units?

Unlike the first two scenarios, you will query Entrez (not PUG) initially to generate the list of CIDs. To perform the Entrez query, using "eSearch", use the URL:
http://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=pccompound&usehistory=y&retmax=0&term=100:100.01[exactmass]

Please note that the "retmax=0" argument in the URL above prevents the actual result list of PubChem Compound CIDs from being returned. The "usehistory=y" creates an Entrez

history item, required for the next step in this scenario. If a CID list is desired, simply omit both, e.g.:
http://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=pccompound&term=100:100.01[exactmass]

The XML return message from eSearch, using the former URL above (rather than the latter), will look like:

```
<eSearchResult>
        <Count>81</Count>
        <RetMax>0</RetMax>
        <RetStart>0</RetStart>
        <QueryKey>26</QueryKey>
        <WebEnv>0BPLhFE_YfmLOCUMsO7FDRuhXLxgPqzfs-aB_O2nILEnCpSEb-
AIRQzeQ0LTaQNNlpK8XkxiDcX71it@46C3203C79E0BE30_0000SID</WebEnv>
        <IdList>
        </IdList>
        <TranslationSet>
        </TranslationSet>
        <TranslationStack>
                <TermSet>
                        <Term>0000100.000000[ExactMass]</Term>
                        <Field>ExactMass</Field>
                        <Count>-1</Count>
                        <Explode>Y</Explode>
                </TermSet>
                <TermSet>
                        <Term>0000100.010000[ExactMass]</Term>
                        <Field>ExactMass</Field>
                        <Count>-1</Count>
                        <Explode>Y</Explode>
                </TermSet>
                <OP>RANGE</OP>
        </TranslationStack>
        <QueryTranslation>0000100.000000[ExactMass] :
0000100.010000[ExactMass]</QueryTranslation>
</eSearchResult>
```

When the query is complete, the result is an Entrez query key and webenv. The query key identifies the query result and the webenv provides your session identifier. You use this Entrez query key and webenv as your source of CIDs to compose another PUG message to download the gzipped compressed SDF file of the query hits:

```
<PCT-Data>
  <PCT-Data_input>
    <PCT-InputData>
      <PCT-InputData_download>
        <PCT-Download>
          <PCT-Download_uids>
            <PCT-QueryUids>
              <PCT-QueryUids_entrez>
                <PCT-Entrez>
                  <PCT-Entrez_db>pccompound</PCT-Entrez_db>
                  <PCT-Entrez_query-key>26</PCT-Entrez_query-key>
                  <PCT-Entrez_webenv>0BPLhFE_YfmLOCUMsO7FDRuhXLxgPqzfs-
aB_O2nILEnCpSEb-AIRQzeQ0LTaQNNlpK8XkxiDcX71it@46C3203C79E0BE30_0000SID</PCT-
Entrez_webenv>
                </PCT-Entrez>
              </PCT-QueryUids_entrez>
            </PCT-QueryUids>
          </PCT-Download_uids>
          <PCT-Download_format value="sdf"/>
          <PCT-Download_compression value="gzip"/>
        </PCT-Download>
      </PCT-InputData_download>
    </PCT-InputData>
  </PCT-Data_input>
```

```
</PCT-Data>
```

The final result, as in the first scenario, is an URL to the results:

```
<PCT-Data>
  <PCT-Data_output>
    <PCT-OutputData>
      <PCT-OutputData_status>
        <PCT-Status-Message>
          <PCT-Status-Message_status>
            <PCT-Status value="success"/>
          </PCT-Status-Message_status>
        </PCT-Status-Message>
      </PCT-OutputData_status>
      <PCT-OutputData_output>
        <PCT-OutputData_output_download-url>
          <PCT-Download-URL>
            <PCT-Download-URL_url>ftp://ftp-
private.ncbi.nlm.nih.gov/pubchem/.fetch/816930703564580480.sdf.gz</PCT-Download-
URL_url>
          </PCT-Download-URL>
        </PCT-OutputData_output_download-url>
      </PCT-OutputData_output>
    </PCT-OutputData>
  </PCT-Data_output>
</PCT-Data>
```

In this example, 81 hits are (currently) returned.

Scenario 4. How do I get back the PubMed abstracts for PubChem Compound CID 2244 (aspirin)?

Similar to Scenario 3, you will query Entrez (not PUG) to get a list of PubMed abstracts linked to a PubChem Compound. To perform the Entrez query, using "eLink", for use the URL:
http://eutils.ncbi.nlm.nih.gov/entrez/eutils/elink.fcgi?dbfrom=pccompound&id=2244&db=pubmed

Given that a list of abstracts may be long, it is a good idea to create an Entrez history item, e.g., using the URL:
http://eutils.ncbi.nlm.nih.gov/entrez/eutils/elink.fcgi?dbfrom=pccompound&id=2244&db=pubmed&cmd=neighbor_history

The XML return message from eLink, using the latter URL above (rather than the former), will look like:

```
<eLinkResult>
        <LinkSet>
        <DbFrom>pccompound</DbFrom>
        <IdList>
                <Id>2244</Id>
        </IdList>
        <LinkSetDbHistory>
                <DbTo>pubmed</DbTo>
                <LinkName>pccompound_pubmed</LinkName>
                <QueryKey>5</QueryKey>
        </LinkSetDbHistory>
        <LinkSetDbHistory>
                <DbTo>pubmed</DbTo>
                <LinkName>pccompound_pubmed_mesh</LinkName>
                <QueryKey>6</QueryKey>
        </LinkSetDbHistory>
```

```
<WebEnv>0FFDpPFhSw2nzieR6fvaLMqXLnx9GKbcev9IJqI3EXp8pEzTEj38McL0EHmseTEpHXZkgacEGy
m2qtB@264F60B37ADDEBF0_0143SID</WebEnv>
        </LinkSet>
  </eLinkResult>
```

In the message above, two Entrez history items were created, one corresponding to the depositor provided links (pccompound_pubmed) and those derived through linkage of MeSH ontology with depositor provided synonyms (pccompound_pubmed_mesh). To retrieve the abstracts, one may provide PubMed ids, one at a time or all at once, using "eFetch". To retrieve a single abstract, e.g., for PubMed id 12767473, one would formulate an URL like:
http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=pubmed&id=12767473&retmode=xml&rettype=abstract

To retrieve abstracts for a list of PubMed ids contained in an Entrez history, e.g., for the eLink query above:
http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=pubmed&webenv=0FFDpPFhSw2nzieR6fvaLMqXLnx9GKbcev9IJqI3EXp8pEzTEj38McL0EHmseTEpHXZkgacEGym2qtB@264F60B37ADDEBF0_0143SID&query_key=5&retmode=xml&rettype=abstract

The output of the above URLs is omitted for brevity.

Document Version History.

V1.3.0 – 2008June13 -	Added new section on the PubChem BioAssay Query/Download service. Minor cleanup of the previous documentation. Corrected PUG SOAP URL.
V1.2.0 - 2008May20 -	Added new section for FAQs, providing usage scenarios. Updated the PUG SOAP documentation.
V1.1.0 - 2008Jan11 -	Added new section on the PubChem Standardization service. Minor cleanup of and additions to the previous documentation.
V1.0.0 - 2007May10 -	Initial release.