rcsb-api: Python Toolkit for Streamlining Access to RCSB Protein Data Bank APIs ☆

Dennis W. Piehl [1,*], Brinda Vallat [1,2], Ivana Truong [1], Habiba Morsy [1,†], Rusham Bhatt [1,‡], Santiago Blaumann [1,§], Pratyoy Biswas [1], Yana Rose [3], Sebastian Bittrich [3], Jose M. Duarte [3], Joan Segura [3], Chunxiao Bi [3], Douglas Myers-Turnbull [3], Brian P. Hudson [1], Christine Zardecki [1], and Stephen K. Burley [1,2,3,4,5]

*1 - **Research Collaboratory for Structural Bioinformatics Protein Data Bank and the Institute for Quantitative Biomedicine,** Rutgers, The State University of New Jersey, Piscataway, NJ 08854, USA*
*2 - **Rutgers Cancer Institute,** Rutgers, The State University of New Jersey, New Brunswick, NJ 08901, USA*
*3 - **Research Collaboratory for Structural Bioinformatics Protein Data Bank,** San Diego Supercomputer Center, University of California San Diego, La Jolla, CA 92093, USA*
*4 - **Department of Chemistry and Chemical Biology,** Rutgers, The State University of New Jersey, Piscataway, NJ 08854, USA*
*5 - **Rutgers Artificial Intelligence and Data Science (RAD) Collaboratory,** Rutgers, The State University of New Jersey, Piscataway, NJ 08854, USA*

***Correspondence to Dennis W. Piehl:** dennis.piehl@rcsb.org (D.W. Piehl)*
https://doi.org/10.1016/j.jmb.2025.168970
Editor: Michael Sternberg

Abstract

The Protein Data Bank (PDB) was founded in 1971 as the first open-access digital data resource in biology to serve as the single global archive for three-dimensional (3D) macromolecular structure data. Current PDB holdings exceed 230,000 experimentally determined structures of proteins, nucleic acids, viruses, and macromolecular machines. The RCSB Protein Data Bank RCSB.org research-focused web portal facilitates search, analyses, and visualization of every PDB structure along with more than one million Computed Structure Models from AlphaFold DB and the ModelArchive. It is powered by a set of publicly available Application Programming Interfaces (APIs) that both support RCSB.org users and provide programmatic access to PDB data. Given the breadth and levels of granularity encompassed in this rich data collection, efficiently accessing the information programmatically may be challenging for new users. RCSB PDB has developed a Python software package, *rcsb-api*, that facilitates easy and efficient use of RCSB PDB APIs within a Python environment. This software tool is designed to streamline access to the extensive corpus of data housed within the PDB, enabling researchers to search, retrieve, and analyze 3D biostructure data seamlessly. Its use will accelerate research in structural biology, molecular biology and biochemistry, drug discovery, and bioinformatics by providing more efficient tools for data integration and analysis. The new toolkit is available on GitHub (github.com/rcsb/py-rcsb-api) and published to the public Python package repository (PyPI) to foster wider usage and support basic and applied research in fundamental biology, biomedicine, and the energy sciences.

☆ This article is part of a special issue entitled: 'Computation Resources (2025)' published in Journal of Molecular Biology.

Introduction

The Protein Data Bank (PDB) was established in 1971 as the first open-access digital data resource in biology, serving as the global archive for experimentally determined, three-dimensional (3D) structures of biological macromolecules.[1,2] More than five decades later, the PDB archive has grown from just seven X-ray crystal structures of proteins to >230,000 structures coming from macromolecular crystallography (MX), 3D electron microscopy (3DEM), and nuclear magnetic resonance (NMR) spectroscopy. PDB data are freely available to users around the world with no limitations on usage under the most permissive Creative Commons CC0 License.

The Research Collaboratory for Structural Bioinformatics Protein Data Bank (RCSB PDB)[3,4] serves as the United States (US) data center of the Worldwide Protein Data Bank (wwPDB, wwpdb.org) partnership, which was established in 2003 to support joint management of the PDB archive as a global public good.[5] The RCSB PDB research-focused RCSB.org web portal enables tailored access to PDB data, providing an extensive collection of tools and features for searching, browsing, analyzing, and visualizing 3D biostructures.[4,6] Alongside the wealth of experimentally determined PDB structures, RCSB.org delivers parallel access to more than one million Computed Structure Models (CSMs) from the AlphaFold Protein Structure Database (alphafold.ebi.ac.uk)[7] and the ModelArchive (modelarchive.org), which substantially expands the structural coverage available to PDB data consumers, with inclusion of the complete human proteome plus proteomes of many model organisms, select human pathogens, food crop plants, and organisms relevant to bioenergy research, as well as CSMs of *S. cerevisiae* multiprotein complexes.[6] RCSB PDB enriches these data at each weekly release of the updated PDB archive (typically adding 300–400 new structures) with annotations from a wide range of trusted external biological and chemical data resources to provide additional evolutionary and functional context to PDB structures and CSMs.

To ensure that PDB data are Findable, Accessible, Interoperable, and Reusable (FAIR)[8] for many millions of users working and learning in nearly every country and territory recognized by the United Nations, RCSB.org offers various ways of accessing data.[9] Most users explore RCSB.org through a web browser (e.g., Safari, Chrome, Firefox), where they browse structures by annotation, search by structure (PDB or CSM) identifiers (IDs), view key structural and experimental details and even visualize and align structures in 3D with the built-in Mol* in-browser molecular graphics system.[10] For more focused research efforts, wherein users wish to interrogate specific biostructure data (accounting for ∼34% of RCSB.org usage in 2023, based on internal analytics), basic full-text and keyword-based searches can be utilized alongside our Advanced Search query builder (rcsb.org/search/advanced) to construct finely tuned searches (e.g., sequence similarity and motif matching, structure similarity searches, or any combination of these options). Finally, bioinformatics researchers wishing to perform search and data retrieval tasks at scale and/or as part of an automated workflow will find value in directly accessing RCSB PDB Application Programming Interface (API) services that power all RCSB.org tools and functionalities. These API services provide users with direct programmatic access to not only the rich collection of biostructure data in the PDB archive, similar to the APIs offered by wwPDB partner sites PDBe (Protein Data Bank in Europe)[11] and PDBj (Protein Data Bank Japan),[12] but also to the diverse suite of search and data retrieval functionalities available through RCSB.org. For example, programmatic users can leverage the RCSB.org search API service (search.rcsb.org) to retrieve a list of structure IDs based on a custom search query and use the data API service (data.rcsb.org) to fetch the metadata associated with a list of structure IDs, all from within an in-house script or pipeline.[9] Programmatic access to RCSB.org API services has been growing steadily since their introduction, accounting for nearly 10% of utilization in 2023.

Several resources and tools are provided to aid with API query design, including extensive documentation, online query editor interfaces (e.g., search API, search.rcsb.org/query-editor.html; data API, data.rcsb.org/graphql/index.html), plus search and data API buttons/links provided on search results pages and structure summary pages, respectively. Because each API service is built atop its own set of schemas (i.e., defined set of input parameters and response formats) and may not follow the same architecture, learning how to construct queries between different services can be time-consuming and require substantial manual effort (particularly for novices). Moreover, once users have constructed initial queries, making any manipulations or substitutions to one part of the query may require a set of coordinated changes to other parts of the query, thereby complicating workflow automation. Finally, users wishing to integrate queries into custom data processing workflows are likely faced with the need to write code to manage API requests and responses.

Recognizing such challenges, researchers have developed several software packages aimed at facilitating programmatic access to PDB data and RCSB.org APIs. *PyPDB*[13] and *localpdb*,[14] for example, provide valuable Python tooling for many data searching and fetching tasks; however, their potential is limited by the lack of full API support and the need to locally clone the PDB archive, respectively. Moreover, there remains a dearth of

API schema-aware client packages capable of supporting automated query construction. To help overcome these limitations, we present herein *rcsb-api*—an all-in-one Python utility providing streamlined access to the full suite of capabilities offered by RCSB.org search and data API services.

Developed with a schema-driven design, our new software tool greatly simplifies construction of PDB archive queries by minimizing user input and handling request overhead, while introducing features that facilitate previously manual tasks (e.g., file upload-based search and automatic query expansion), delivering notable advantages over existing tools. In addition, the new software provides useful functionalities for exploring search attributes and data fields available for each associated API service. Finally, by building this tool with the widely used and high-level Python programming language, *rcsb-api* lowers the barrier for non-expert computational researchers to access RCSB.org API services and enables seamless integration into custom Python-based data analysis pipelines and research workflows. *rcsb-api* was released under an open-source license in support of the FAIR principles for research software (FAIR4RS).[15] It is freely available to the research community on GitHub (github.com/rcsb/py-rcsb-api), encouraging collaboration and greater accessibility to the wealth of 3D biostructure data archived in the global PDB.

Methods

Package architecture

Our *rcsb-api* software provides programmatic access to search and data API services offered by RCSB.org. The package is organized into distinct modules for each supported API service: the *rcsbapi.search* module enables users to retrieve a list of PDB and/or CSM IDs based on specific search criteria, while the *rcsbapi.data* module allows for metadata retrieval for a given list of structure IDs. This organizational approach will allow incorporation of additional modules to support other RCSB.org API services.

Python was selected as the programming language for development in recognition of its popularity in scientific research, ease of use, and broad support across platforms. Data typing is applied throughout the codebase to enable automatic type hinting and facilitate package usage. The source code is publicly available on GitHub (github.com/rcsb/py-rcsb-api) under the highly permissible MIT license and published to the Python Package Index (PyPI) repository (pypi.org/project/rcsb-api).

RCSB PDB data organization

The design of the *rcsb-api* package is underpinned by RCSB PDB data standards and architecture. Central to our data ecosystem is the PDB exchange/macromolecular Crystallographic Information Framework (PDBx/mmCIF),[16,17] which serves as the global data standard for representing experimentally determined 3D biostructure data and related metadata across the PDB archive through a comprehensive dictionary of data item definitions (mmcif.wwpdb.org). Extension dictionaries are also publicly available, providing definitions for describing new types of methodologies and associated metadata (e.g., ModelCIF for CSMs).[18]

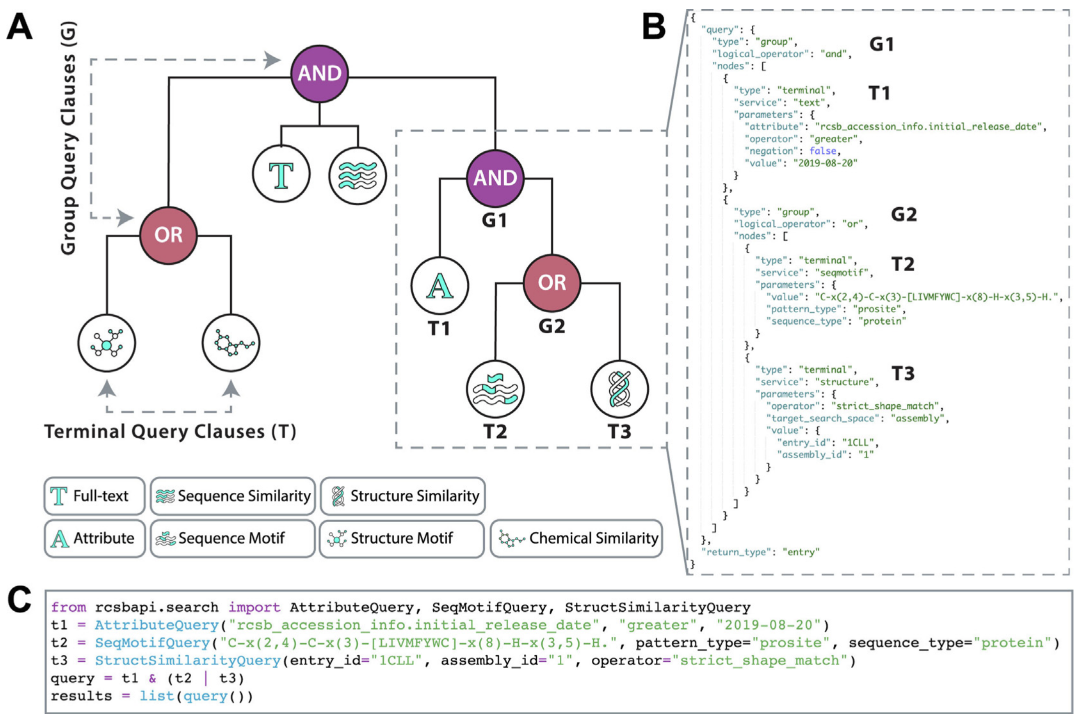
Internal RCSB.org databases are updated weekly in coordination with each release of ∼300–400 new structures into the growing PDB archive. During this process, source PDBx/mmCIF structure and small molecule (chemical component) files are subject to a series of extract, transform, and load (ETL) operations that organize the information into distinct but interrelated object types at different granularities based on the natural hierarchy of macromolecular structures. Specifically, data within each PDB structure may be broken down into the following hierarchical object types: entry (the entire structure accessible via a unique PDB or CSM ID), polymer entities (unique polymer sequences), polymer entity instances (individual instances/chains of a given polymer entity or sequence), non-polymer entities (unique non-polymers, such as small-molecule ligands), non-polymer entity instances (individual instances of non-polymers), branched entities (unique oligosaccharides), branched entity instances (individual instances of oligosaccharides), and assemblies (biologically relevant groups of one or more entity instances, such as the hemoglobin $\alpha_2\beta_2$ heterotetramer). Through this organizational scheme, data constituting a given PDB ID are split into different object types according to relevance, such as associating the experimental method with the entry object type and the protein sequence with the polymer entity type. Alongside this macromolecular-based organizational framework, other data sources such as the wwPDB Chemical Component Dictionary[19] and UniProt[20] are stored as separate object types. In parallel, CSMs are partitioned into the same database organization scheme. Treatment of CSMs as first-class objects allows them to be queried and accessed with the same API services as all other RCSB.org data, thereby enabling access to 3D structure information for entire proteomes.[6]

The specific set of data attributes loaded for each object type within the resulting databases is defined by the RCSB PDB core data schema (data.rcsb.org/#data-schema), which includes select attributes from the PDBx/mmCIF data dictionary, the ModelCIF data dictionary and other extension dictionaries, plus any attributes added during ETL in support of RCSB.org web services (e.g., software-calculated attributes or information integrated from trusted external resources). While the full range of data attributes present in RCSB.org databases is accessible via the data API service (data.rcsb.org/data-attributes.html), search is enabled only for a subset of structure attributes, which are routinely reviewed and extended (search.rcsb.org/structure-search-attributes.html and search.rcsb.org/chemical-search-attributes.html).

Search API module design

Design of the search API module, *rcsbapi.search*, is based on the OpenAPI schema specification (openapis.org) underlying the RCSB.org search API service (search.rcsb.org/redoc/index.html).[9] The search API service is a RESTful (REpresentational State Transfer)[21] interface that allows users to construct and execute Boolean-grouped queries with any combination and nesting of "terminal nodes" (operands) and "group nodes" (collections of terminal nodes), in which each terminal node can represent any of the available RCSB.org search services: full-text, structure attribute, chemical attribute, sequence similarity, sequence motif, structure similarity, structure motif,[22] and/or chemical similarity (Figure 1A). Each search service adheres to a specific schema that defines the set of input parameters, the format and/or enumerations of their corresponding values, plus possible response types and structures. For example, Figure 1B illustrates distinct parameters used for attribute, sequence motif, and structure similarity searching.

Paralleling this architecture, the *rcsbapi.search* module implements "Terminal" and "Group" classes that allow for arbitrary nesting of terminal and group nodes, plus a distinct class for each search service type with input parameters matching those defined by the associated API schema (Figure 1C). Additionally, lists of available structure and chemical attributes that can be queried by the attribute search service are dynamically fetched and stored in an internal data structure upon module initialization. For query construction, two syntaxes are supported—operator style and fluent style. In the operator syntax, terminal nodes can be combined using standard mathematical or logical operators (Figure 1C), making construction of complex or deeply nested queries easier and more intuitive. In the fluent syntax, terminal query nodes can be appended to and grouped with sequential nodes via method chaining, which may offer a more procedural approach to query construction. Once a search query has been constructed and executed, results are returned via automatic pagination. Automatically paginating over large result sets enables users to perform broad or



```
{
  "query": {
    "type": "group",
    "logical_operator": "and",
    "nodes": [
      {
        "type": "terminal",
        "service": "text",
        "parameters": {
          "attribute": "rcsb_accession_info.initial_release_date",
          "operator": "greater",
          "negation": false,
          "value": "2019-08-20"
        }
      },
      {
        "type": "group",
        "logical_operator": "or",
        "nodes": [
          {
            "type": "terminal",
            "service": "seqmotif",
            "parameters": {
              "value": "C-x(2,4)-C-x(3)-[LIVMFYWC]-x(8)-H-x(3,5)-H.",
              "pattern_type": "prosite",
              "sequence_type": "protein"
            }
          },
          {
            "type": "terminal",
            "service": "structure",
            "parameters": {
              "operator": "strict_shape_match",
              "target_search_space": "assembly",
              "value": {
                "entry_id": "1CLL",
                "assembly_id": "1"
              }
            }
          }
        ]
      }
    ]
  },
  "return_type": "entry"
}
```

```
from rcsbapi.search import AttributeQuery, SeqMotifQuery, StructSimilarityQuery
t1 = AttributeQuery("rcsb_accession_info.initial_release_date", "greater", "2019-08-20")
t2 = SeqMotifQuery("C-x(2,4)-C-x(3)-[LIVMFYWC]-x(8)-H-x(3,5)-H.", pattern_type="prosite", sequence_type="protein")
t3 = StructSimilarityQuery(entry_id="1CLL", assembly_id="1", operator="strict_shape_match")
query = t1 & (t2 | t3)
results = list(query())
```

Figure 1. RCSB.org search API query construction. (A) Schematic representation of a complex query with multiple levels of grouping on terminal (T) and group (G) nodes that combine different search services. (B) Corresponding query in the JSON syntax for the outlined subgroup in (A) (dashed box, including attribute (T1), sequence motif (T2), and structure similarity (T3) terminal search nodes). (C) Example query construction for the same outlined subgroup in (A) with the *rcsbapi.search* module using the operator syntax.

complex queries efficiently, minimizing the risk of timeouts and avoiding excessive server load from a single query.

Data API module design

In designing the *rcsbapi.data* module, the two data API architectures provided by RCSB.org were considered: a REST-based API, which offers access to the majority of RCSB PDB data via predefined endpoints, and a GraphQL API,[23] which enables access to all available data and allows users to create finely tuned requests for specific subsets of data across diverse collections.[9] An added advantage of a GraphQL API is that the hierarchical structure of PDB data aligns naturally with the graph-based organization, enabling seamless traversal of relationships to retrieve targeted subsets of data. Given these benefits, we implemented the *rcsbapi.data* module with support for GraphQL architecture, centering the design and functionality around the underlying API schema.

The GraphQL data API schema encompasses the complete set of data object types (e.g., "entry," "assembly", etc.) and their associated data attributes (as defined in the core RCSB PDB data schemas described earlier), together with information concerning relationships between these object types within the data hierarchy/organization. To internalize and make use of this information, upon import of the *rcsbapi.data* module, the GraphQL schema is fetched via an introspection query and subsequently transformed into a directed graph data structure using *rustworkx*.[24] The resulting graph object contains a collection of nodes that represent available data fields ("field nodes," representing query-accessible data) and associated types ("type nodes," defining available fields and/or scalars for the field node), plus any parent–child relationships between them (directed edges). Thus, this graph object establishes how the *rcsbapi.data* module can construct and validate queries to the RCSB.org data API service. For example, Figs. 2A–C illustrate how the graph data structure can be used to construct a query for the polymer entity feature data (e.g., residue-level positional properties) of PDB (and/or CSM) IDs by traversing through the hierarchical order of type and field nodes.

Results and Discussion

Search API module usage and features

The search API module provides users with a Pythonic interface for interacting with the RCSB.org search API service, which enables users to perform queries on >227,000 experimental PDB structures, >1,000,000 CSMs, and >42,000 chemical components to retrieve a list of matching identifiers. Moreover, users can combine multiple search service types into a single query using Boolean operators, including full-text, attribute-based, sequence similarity, sequence motif, structure similarity, structure motif, and chemical similarity searches. The module also supports all request options available through the RCSB.org search API service, such as specifying the result return type (e.g., "entries," "assemblies," "polymer entities," "non-polymer entities," or "molecular definitions"), returning associated scores from the search algorithm, executing faceted queries, grouping results (e.g., by sequence identity or UniProt ID), and including CSMs in the search query.

Beyond providing a means for programmatically interacting with the RCSB.org search API service directly from Python, the *rcsbapi.search* module provides several features that simplify the process of constructing and performing searches. First, the package supports query construction through both operator-based and fluent syntaxes, which eliminates the need to manipulate nested JSON objects manually. Furthermore, the operator syntax in particular offers a simplified and intuitive style of building highly complex queries via the use of bitwise (e.g., "&" and "|") or Python Boolean operators to group and/or separate multiple subqueries. Second, for structure-based searches (i.e., structure similarity and structure motif search), which support input of a user-uploaded structure file, the package allows users to simply point to a local structure file path (or a remote URL) in PDB, PDBx/mmCIF, or BinaryCIF format,[25] circumventing the need to manually upload files in advance (as would otherwise be required when interacting with the search API service directly). Third, the module includes helper methods to assist users in exploring available attributes for structure and chemical attribute searches, such as searching through attributes by substring (e.g., finding all attributes containing "exptl") and retrieving details about specific attributes.

Data API module usage and features

The data API module provides users with a streamlined RCSB.org data API GraphQL service interface, allowing retrieval of specific data subsets for a list of RCSB PDB object identifiers. Of note, users can construct queries by specifying three input arguments: (1) the input object type (e.g., "entries" or "polymer_entities"); (2) the set of identifiers to query corresponding to that type; and (3) the set of desired data attributes (or "field paths") to return. Figure 2A exemplifies fetching of polymer entity feature data associated with three different entries. Execution of the query returns a Python dictionary containing the requested data for each input ID.

To further simplify query construction, the module can automatically resolve partial (but unique) field paths requested within the data API query. First, for any field path under which sub-nested fields

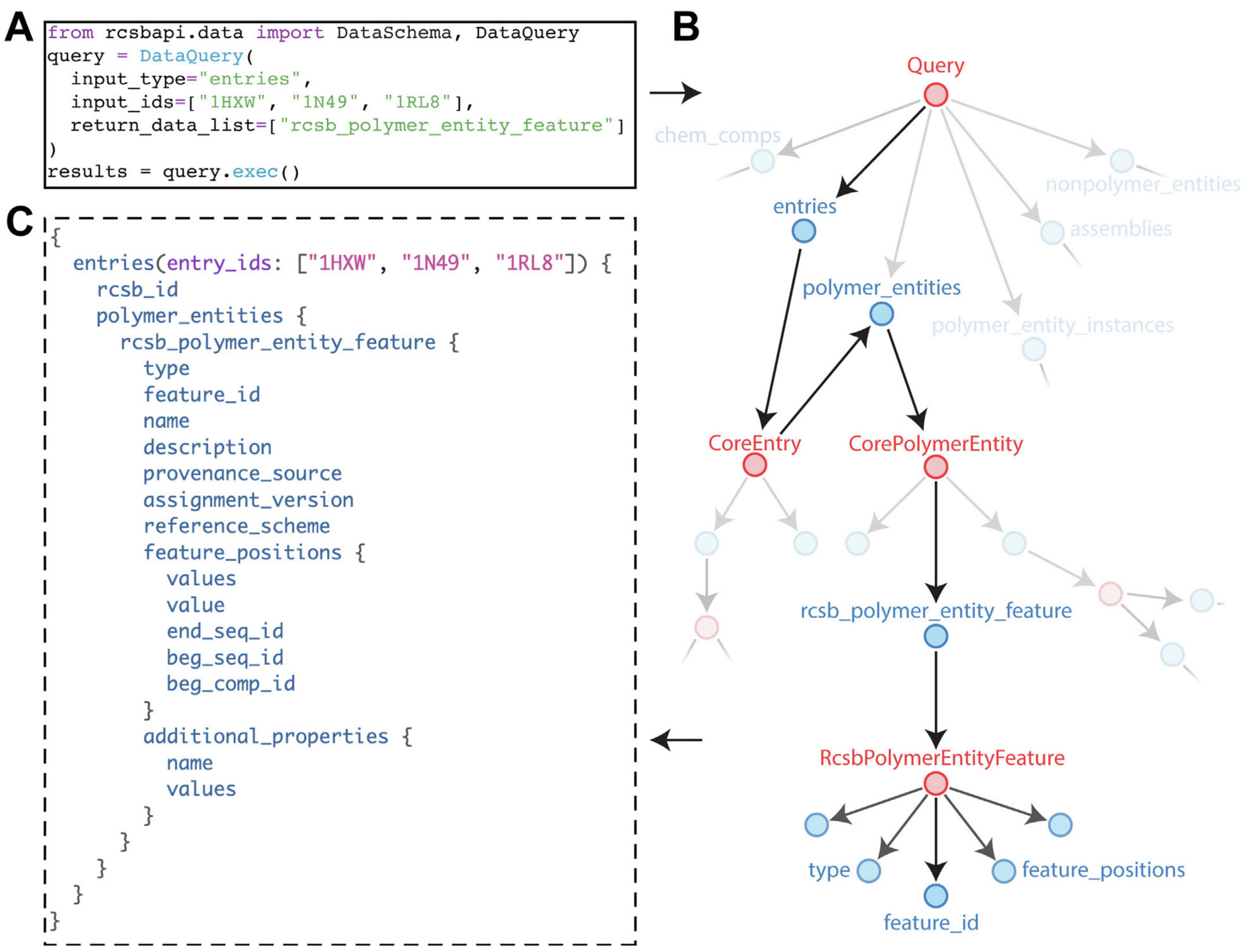

A
```
from rcsbapi.data import DataSchema, DataQuery
query = DataQuery(
  input_type="entries",
  input_ids=["1HXW", "1N49", "1RL8"],
  return_data_list=["rcsb_polymer_entity_feature"]
)
results = query.exec()
```
C
```
{
  entries(entry_ids: ["1HXW", "1N49", "1RL8"]) {
    rcsb_id
    polymer_entities {
      rcsb_polymer_entity_feature {
        type
        feature_id
        name
        description
        provenance_source
        assignment_version
        reference_scheme
        feature_positions {
          values
          value
          end_seq_id
          beg_seq_id
          beg_comp_id
        }
        additional_properties {
          name
          values
        }
      }
    }
  }
}
```

Figure 2. RCSB.org data API query and schema graph construction using the *rcsb-api* Python package. (A) Construction of a query object using the *rcsbapi.data* module to retrieve all polymer entity feature data (i.e., all data fields under the "rcsb_polymer_entity_feature" field node) for multiple PDB IDs. (B) Partial view of the schema graph object created upon initialization of the module starting at the root "Query" node, comprised of field type nodes (red) and field nodes (blue). The highlighted trace in the graph (i.e., non-faded portions) represents the traversal path computed by the module based on query parameters provided in (A). (C) Corresponding data API GraphQL query generated for the data query object defined in (A).

exist, the module automatically and recursively appends those sub-fields to the query so that they are returned with the response. This capability is depicted schematically in Figs. 2A–C, in which requesting just "rcsb_polymer_entity_feature" is sufficient to automatically populate all fields nested below that field (e.g., "feature_positions.values"). Hence, users are relieved of the need of prior knowledge of sub-fields beneath a particular data field path and making the effort to manually specify each item in the base request. Second, for any partial field path provided that uniquely corresponds to a fully qualified path, the package can automatically expand the path in constructing the query, as exemplified in Fig. 2A–C through the automatic expansion of "rcsb_polymer_entity_feature" to "entries.polymer_entities.rcsb_polymer_entity_feature".

Importantly, these two features require that the user-requested field path occurs uniquely in the GraphQL data schema—that is, it must resolve to a single fully qualified field path. This requirement obtains because some field names are redundant between different data contexts (e.g., the name "id" occurs under many different field paths). In such cases, the user must provide a more explicit (i.e., longer) field path for that particular data field so that it can be resolved uniquely. Users can also directly specify fully qualified field paths, which is recommended when developing code for longer-term use to safeguard against schema changes rendering a previously unique path redundant.

Because the data API GraphQL schema is both extensive and complex, the *rcsbapi.data* module includes several helper functions that assist schema exploration and data field/path retrieval.

One such helper function allows users to search for field names by a substring (e.g., all field names containing “exptl”), while another retrieves all fully qualified paths for a given field name and input type. Together, these helper functions can assist discovery of available data fields and learning their unique names.

Integration into research workflows

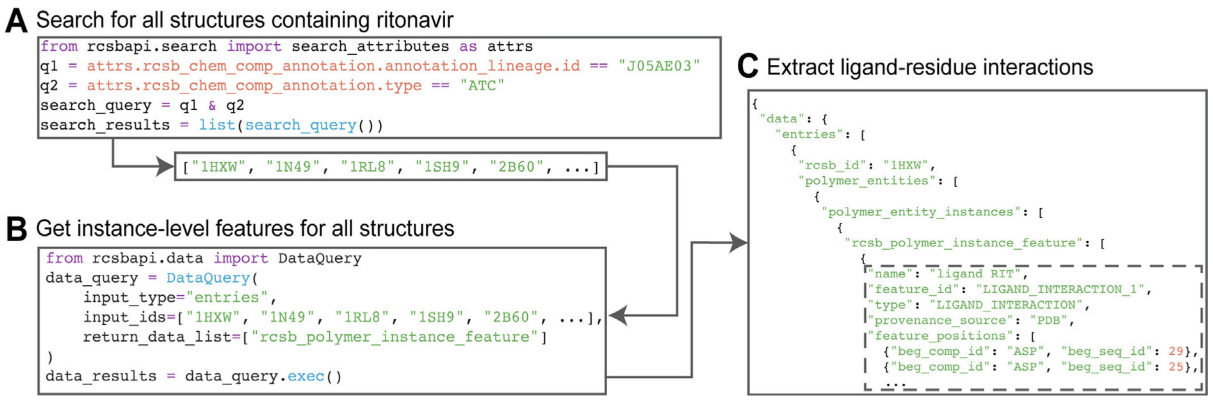
The *rcsb-api* toolkit presented above offers a streamlined/facilitated programmatic method for searching and accessing 3D biostructure data and metadata through RCSB.org API services. Ease of use is ensured by minimizing required inputs for constructing queries, thereby reducing both manual effort and training necessary for using RCSB.org APIs directly. Augmenting usability is the intuitive, Pythonic interface of *rcsb-api* together with implementation of useful features, such as operator-based query syntax, file-based searches, and automatic resolution of nested fields.

While each module offers significant value individually, the full potential of our Python toolkit can only be realized when both modules are used in tandem within integrated workflows. By combining functionalities of both modules, researchers can efficiently link search and data APIs to enable streamlined querying and data retrieval. For example, after using *rcsbapi.search* to identify relevant PDB structures based on specific criteria (e.g., structures containing a specified ligand), results can be directly passed to *rcsbapi.data* to retrieve detailed metadata for those structures (e.g., protein sequence positions interacting with ligand). Figure 3 illustrates how the package can be used to examine amino acid residues involved in binding the HIV protease inhibitor, ritonavir. First, the query identifies all PDB structures containing ritonavir (using its Anatomical Therapeutic Chemical (ATC)[26] code, “J05AE03”; Figure 3A). Second, positional feature data for those structures are fetched (Figure 3B), from which lists of amino acid residue-ligand interactions are extracted (Figure 3C). Furthermore, users are free to integrate these data with their in-house datasets to augment internal analyses, requiring only that the data extracted from RCSB.org be transformed into the appropriate data structure for merging. Overall, the ability to seamlessly link output from one API service to the input of the next enhances the efficiency of search-to-data pipelines, ensuring smoother transitions between querying and data analysis.

Conclusions

In this work, we present a Python toolkit that can be used to interact with RCSB.org search and data API services—*rcsb-api*. This package greatly simplifies the API user experience by reducing the complexity of query construction and automating tasks that would otherwise require considerable manual effort. Moreover, by abstracting the schema-level details of the API services into a Python interface, *rcsb-api* removes the need for users to learn the underlying API syntax (e.g., RESTful, GraphQL) and offers helpful tools for exploring both search and data attribute schemas. The toolkit can support research in structural bioinformatics (e.g., performing large-scale analyses of structures with specific sequence motifs), structure-guided drug discovery (e.g., analyzing structural motifs and ligand interactions to identify potential binding pockets), and beyond.

Importantly, the modular design of the package ensures future extensibility with the potential to


A Search for all structures containing ritonavir
B Get instance-level features for all structures
C Extract ligand-residue interactions

```
from rcsbapi.search import search_attributes as attrs
q1 = attrs.rcsb_chem_comp_annotation.annotation_lineage.id == "J05AE03"
q2 = attrs.rcsb_chem_comp_annotation.type == "ATC"
search_query = q1 & q2
search_results = list(search_query())
```

```
["1HXW", "1N49", "1RL8", "1SH9", "2B60", ...]
```

```
from rcsbapi.data import DataQuery
data_query = DataQuery(
    input_type="entries",
    input_ids=["1HXW", "1N49", "1RL8", "1SH9", "2B60", ...],
    return_data_list=["rcsb_polymer_instance_feature"]
)
data_results = data_query.exec()
```

```
{
 "data": {
   "entries": [
     {
      "rcsb_id": "1HXW",
      "polymer_entities": [
        {
         "polymer_entity_instances": [
           {
            "rcsb_polymer_instance_feature": [
              {
              "name": "ligand RIT",
              "feature_id": "LIGAND_INTERACTION_1",
              "type": "LIGAND_INTERACTION",
              "provenance_source": "PDB",
              "feature_positions": [
                {"beg_comp_id": "ASP", "beg_seq_id": 29},
                {"beg_comp_id": "ASP", "beg_seq_id": 25},
                ...
```

Figure 3. Using the *rcsb-api* package in a research workflow. The example shown here depicts the steps that can be followed to investigate amino acid residues involved in the binding sites of the HIV protease inhibitor, ritonavir. (A) First, the search API module can be used to identify all structures containing ritonavir by querying for all entries with ATC code “J05AE03” (for ritonavir). (B) The resulting entry ID list may then be provided as input to the data API module to fetch all polymer entity instance feature data associated with each structure, and (C) the returned feature data may be processed to extract all amino acid residue-ligand interaction information.

incorporate support for additional RCSB.org API services, such as 1D Coordinates[27] and ModelServer[28] APIs. We envision that *rcsb-api* will continue to serve as an all-in-one Python toolkit for RCSB.org APIs, supporting research reliant on open access to 3D biostructure information for many years to come.

To ensure adherence to FAIR4RS principles and FAIR Biomedical Research Software (FAIR-BioRS) guidelines,[29] *rcsb-api* was made fully open-source on GitHub and released under the MIT license. Beyond improving the FAIRness of RCSB PDB software, the new Python API client utility enhances the FAIRness of 3D biostructure data delivery at RCSB.org by facilitating use of powerful RCSB PDB API services.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used ChatGPT (4o) in order to improve grammar and language. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

CRediT authorship contribution statement

Dennis W. Piehl: Writing – review & editing, Writing – original draft, Visualization, Supervision, Software, Project administration, Conceptualization. **Brinda Vallat:** Writing – review & editing, Supervision, Software, Project administration, Conceptualization. **Ivana Truong:** Writing – review & editing, Writing – original draft, Visualization, Software. **Habiba Morsy:** Writing – review & editing, Writing – original draft, Visualization, Software. **Rusham Bhatt:** Writing – review & editing, Software. **Santiago Blaumann:** Software. **Pratyoy Biswas:** Writing – review & editing, Software. **Yana Rose:** Writing – review & editing, Visualization, Software, Project administration. **Sebastian Bittrich:** Writing – review & editing, Software. **Jose M. Duarte:** Writing – review & editing, Software. **Joan Segura:** Software. **Chunxiao Bi:** Software. **Douglas Myers-Turnbull:** Software. **Brian P. Hudson:** Writing – review & editing, Supervision. **Christine Zardecki:** Writing – review & editing, Writing – original draft, Supervision, Project administration. **Stephen K. Burley:** Writing – review & editing, Supervision, Funding acquisition.

DECLARATION OF COMPETING INTEREST

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors gratefully acknowledge Dr. Spencer Bliven (Paul Scherrer Institute, Switzerland), who authored the foundational open-source codebase of the search API module. We also acknowledge the Research Intensive Summer Experience (RISE) program at Rutgers, through which much of this work was performed. We thank Maria Voigt (RCSB PDB, Rutgers) for guidance on figure design. We acknowledge contributions from RCSB PDB members past and present and our wwPDB partners. Finally, we thank the tens of thousands of structural biologists working on all inhabited continents who have deposited structures to the PDB since 1971, and the many millions of researchers, educators, and students around the world who consume PDB data. RCSB PDB Core Operations are funded by the US National Science Foundation (DBI-2321666, P.I.: S.K. Burley), the US Department of Energy (DE-SC0019749, P.I.: S.K. Burley), and the National Cancer Institute, National Institute of Allergy and Infectious Diseases, and National Institute of General Medical Sciences of the National Institutes of Health under grant R01GM157729 (P.I.: S.K. Burley).

Received 15 November 2024;
Accepted 27 January 2025;
Available online 31 January 2025

Keywords:
RCSB Protein Data Bank;
RCSB.org;
structural biology;
bioinformatics;
FAIR principles

† Kean University, Union, NJ 07083, USA.
‡ University of Maryland Baltimore County, Baltimore, MD 21250, USA.
§ Cornell University, Ithaca, NY 14853, USA.

References

1. Protein Data Bank, (1971). Crystallography: Protein Data Bank. *Nature New Biol.* **233**, 223.
2. wwPDB consortium, (2019). Protein Data Bank: the single global archive for 3D macromolecular structure data. *Nucleic Acids Res.* **47**, D520–D528.
3. Burley, S.K., Bhikadiya, C., Bi, C., Bittrich, S., Chao, H., Chen, L., et al., (2023). RCSB Protein Data Bank (RCSB.org): Delivery of experimentally-determined PDB structures alongside one million Computed Structure Models of proteins from Artificial Intelligence/Machine Learning. *Nucleic Acids Res.* **51**, D488–D508.

4. Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., et al., (2000). The Protein Data Bank. *Nucleic Acids Res.* **28**, 235–242.
5. Berman, H.M., Henrick, K., Nakamura, H., (2003). Announcing the worldwide Protein Data Bank. *Nature Struct. Biol.* **10**, 980.
6. Burley, S.K., Bhatt, R., Bhikadiya, C., Bi, C., Biester, A., Biswas, P., et al., (2025). Updated resources for exploring experimental PDB structures and computed structure models at the RCSB Protein Data Bank. *Nucleic Acids Res.* **53**, D564–D574.
7. Varadi, M., Anyango, S., Deshpande, M., Nair, S., Natassia, C., Yordanova, G., et al., (2022). AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Res.* **50**, D439–D444.
8. Wilkinson, M.D., Dumontier, M., Aalbersberg, I.J., Appleton, G., Axton, M., Baak, A., et al., (2016). The FAIR guiding principles for scientific data management and stewardship. *Sci. Data* **3**, 1–9.
9. Rose, Y., Duarte, J.M., Lowe, R., Segura, J., Bi, C., Bhikadiya, C., et al., (2021). RCSB Protein Data Bank: architectural advances towards integrated searching and efficient access to macromolecular structure data from the PDB archive. *J. Mol. Biol.* **433**, 166704.
10. Sehnal, D., Bittrich, S., Deshpande, M., Svobodova, R., Berka, K., Bazgier, V., et al., (2021). Mol* Viewer: modern web app for 3D visualization and analysis of large biomolecular structures. *Nucleic Acids Res.* **49**, W431–W437.
11. Armstrong, D.R., Berrisford, J.M., Conroy, M.J., Gutmanas, A., Anyango, S., Choudhary, P., et al., (2020). PDBe: improved findability of macromolecular structure data in the PDB. *Nucleic Acids Res.* **48**, D335–D343.
12. Kinjo, A.R., Bekker, G.J., Wako, H., Endo, S., Tsuchiya, Y., Sato, H., et al., (2018). New tools and functions in data-out activities at Protein Data Bank Japan (PDBj). *Protein Sci.* **27**, 95–102.
13. Gilpin, W., (2015). PyPDB: a Python API for the Protein Data Bank. *Bioinformatics* **32**, 159–160.
14. Ludwiczak, J., Winski, A., Dunin-Horkawicz, S., (2022). localpdb—a Python package to manage protein structures and their annotations. *Bioinformatics* **38**, 2633–2635.
15. Barker, M., Chue Hong, N.P., Katz, D.S., Lamprecht, A.-L., Martinez-Ortiz, C., Psomopoulos, F., et al., (2022). Introducing the FAIR Principles for research software. *Sci. Data* **9**, 622.
16. Westbrook, J., Henrick, K., Ulrich, E.L., Berman, H.M., (2005). 3.6.2 The Protein Data Bank exchange data dictionary. In: Hall, S.R., McMahon, B. (Eds.), *International tables for crystallography*. Springer, Dordrecht, The Netherlands, pp. 195–198.
17. Westbrook, J.D., Young, J.Y., Shao, C., Feng, Z., Guranovic, V., Lawson, C., et al., (2022). PDBx/mmCIF ecosystem: foundational semantic tools for structural biology. *J. Mol. Biol.* **434**, 167599.
18. Vallat, B., Tauriello, G., Bienert, S., Haas, J., Webb, B.M., Zidek, A., et al., (2023). ModelCIF: an extension of PDBx/mmCIF data representation for computed structure models. *J. Mol. Biol.* **435**, 168021.
19. Westbrook, J.D., Shao, C., Feng, Z., Zhuravleva, M., Velankar, S., Young, J., (2015). The chemical component dictionary: complete descriptions of constituent molecules in experimentally determined 3D macromolecules in the Protein Data Bank. *Bioinformatics* **31**, 1274–1278.
20. UniProt Consortium, (2023). UniProt: the Universal Protein Knowledgebase in 2023. *Nucleic Acids Res.* **51**, D523–D531.
21. Fielding, R.T., (2000). Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine.
22. Bittrich, S., Burley, S.K., Rose, A.S., (2020). Real-time structural motif searching in proteins using an inverted index strategy. *PLoS Comput. Biol.* **16**, e1008502.
23. GraphQL Foundation/Linux Foundation (2019). GraphQL API oriented query language.
24. Treinish, M., Carvalho, I., Tsilimigkounakis, G., Sá, N., (2022). rustworkx: a high-performance graph library for Python. *J. Open Source Softw.* **7**, 3968.
25. Sehnal, D., Bittrich, S., Velankar, S., Koca, J., Svobodova, R., Burley, S.K., et al., (2020). BinaryCIF and CIFTools-Lightweight, efficient and extensible macromolecular data management. *PLoS Comput. Biol.* **16**, e1008247.
26. WHO Collaborating Centre for Drug Statistics Methodology, (2023). Guidelines for ATC Classification and DDD Assignment 2024. Norwegian Institute of Public Health, Oslo, Norway.
27. Segura, J., Rose, Y., Westbrook, J., Burley, S.K., Duarte, J.M., (2020). RCSB Protein Data Bank 1D tools and services. *Bioinformatics* **36**, 5526–5527.
28. Sehnal, D., Rose, A., Koca, J., Burley, S., Velankar, S., (2018). Mol*: Towards a common library and tools for web molecular graphics. *Proceedings of the Workshop on Molecular Graphics and Visual Analysis of Molecular Data*, **18**, pp 29–33.
29. Patel, B., Soundarajan, S., Ménager, H., Hu, Z., (2023). Making biomedical research software FAIR: actionable step-by-step guidelines with a user-support tool. *Sci. Data* **10**, 557.