

The SIRIUS API: Programmatic Access to Metabolite Annotation

Jonas A. Emmert^{1,2}, Martin A. Hoffmann³, Marcus Ludwig³, Kai Dührkop³, Martin Engler-Lukajewski³, Lukas Scholz¹, Nils A. Haupt¹, Sebastian Böcker^{1¶}, and Markus Fleischauer^{1,3¶}

¹ Chair for Bioinformatics, Institute for Computer Science, Friedrich Schiller University Jena, Jena, Germany ² International Max Planck Research School “Chemical Communication in Ecological Systems”, Max Planck Institute for Chemical Ecology, Jena, Germany ³ Bright Giant GmbH, Jena, Germany ¶ Corresponding author

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#)
- [Repository](#)
- [Archive](#)

Editor: [✉](#)

Submitted: 26 January 2026

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

Summary

Metabolomics plays a crucial role in understanding biological systems, with mass spectrometry being the method of choice for analyzing complex metabolite mixtures, due to its high sensitivity and throughput. However, compound annotation remains a major bottleneck in metabolomics workflows, as for the majority of detected signals no reference spectra are found in spectral libraries. This so-called “dark metabolome” (da Silva et al., 2015) necessitates approaches that can annotate metabolites without relying on spectral library matching. SIRIUS (Böcker et al., 2009; Dührkop et al., 2019) is a comprehensive software framework for annotating metabolites and other small molecules from tandem mass spectrometry (MS/MS) data without the need of reference spectra, while still supporting spectral matching when reference spectra are available.

Here, we introduce a RESTful API for SIRIUS that adheres to the OpenAPI Specification. This enables programmatic access and seamless integration of SIRIUS into computational workflows while future-proofing projects towards later changes by separating the storage layer from the access layer. Built on this foundation, we present *PySirius* and *RSirius*, client libraries for Python and R available via conda-forge, alongside a SIRIUS Java SDK. These clients facilitate integration into bioinformatics pipelines, statistical analyses, data visualization workflows, and other software. Actively maintained on GitHub, the API and client libraries democratize access to state-of-the-art metabolite annotation and enable reproducible research practices in computational metabolomics.

Statement of need

Many specialized metabolomics tools, including SIRIUS for annotation of MS/MS spectra with molecular formulas, structures, and compound classes, are designed as standalone applications accessed through command-line or graphical interfaces. Clearly, a graphical user interface is of massive value for allowing inexperienced user accessing a computational tool (Ludwig et al., 2020). Yet, this architecture also creates barriers for the integration into automated computational workflows and advanced data mining: Users must parse text outputs or intermediate files, and cannot access granular results programmatically.

This challenge was exemplified in SIRIUS 5, where a project directory of plain-text files served as both the storage layer and the primary point of user access. This tight coupling created a difficult dilemma: either break established user workflows by changing file formats to add new features, or hinder innovation to maintain backward compatibility. The transition to an

41 embedded database in SIRIUS 6 solved the innovation bottleneck, but by removing direct file
42 access, it increases the barrier for users needing to programmatically integrate SIRIUS results.

43 The SIRIUS API addresses these limitations by providing programmatic access to all SIRIUS
44 capabilities through a standardized, storage-independent interface. This decoupled architecture
45 separates the data storage layer from the data access layer. It enables direct retrieval of
46 specific information at any granularity, and allows true integration into existing data analysis
47 environments while simultaneously permitting the internal storage formats to evolve without
48 breaking user-facing compatibility.

49 Furthermore, the SIRIUS API enables a more interactive data analysis paradigm. Users
50 are no longer required to load entire project spaces (which can be gigabytes in size) into
51 memory. Instead, they can fetch specific, fine-grained results on demand or selectively perform
52 (re-)computations with new parameters.

53 State of the field

54 Python and R dominate metabolomic data analysis, backed by libraries like *pyOpenMS* (Röst
55 et al., 2014), *matchms* (Huber et al., 2020), and the RforMassSpectrometry initiative including
56 *Spectra* and *Metabonaut* (Louail & Rainer, 2025). *PySirius* and *RSirius* bring SIRIUS into
57 these established ecosystems for the first time, enabling seamless workflows alongside statistical
58 analyses, visualizations, and machine learning models.

59 SIRIUS integrations for multiple widely used mass spectrometry tools such as *MZmine* (v4.8)
60 (Schmid et al., 2023), *Agilent MassHunter Explorer* (v2.0) and *RuSirius* (RforMassSpectrometry)
61 are already based on the SIRIUS API, further demonstrating community demand for
62 programmatic access to SIRIUS.

63 Software design

64 The SIRIUS Application Programming Interface (API), initially released with SIRIUS version
65 6.0.0, is a major architectural update that offers several strategic advantages:

66 Comprehensive programmatic access to functionality

67 The API exposes the complete spectrum of SIRIUS analytical capabilities through programmatic
68 interfaces, encompassing spectral library searching, computation of fragmentation trees (Böcker
69 & Dührkop, 2016; Böcker & Rasche, 2008), annotation of molecular formulas (Dührkop et al.,
70 2013) (orange in Figure 1), structure database search with CSI:FingerID (Dührkop et al.,
71 2015), compound class prediction with CANOPUS (Dührkop et al., 2021), *de novo* molecular
72 structure prediction with MSNovelist (Stravs et al., 2021) and additional methods. This avoids
73 the need to parse command-line interface outputs or manipulate intermediate file formats.

74 Fine-grained and direct data retrieval

75 The API allows direct retrieval of specific data elements at arbitrary levels of granularity
76 (Figure 1). This spans from individual feature-level results to project-wide summaries, thereby
77 enabling more efficient and precisely targeted analytical workflows.

78 Decoupled architectural design

79 The API separates the data persistence layer from the data access layer, which permits
80 modifications to core SIRIUS functionality without invalidating existing project data structures.
81 This architectural decision ensures forward compatibility for future SIRIUS updates, addressing
82 the compatibility limitations encountered during the transition from SIRIUS version 5 to 6.

Platform-agnostic integration capabilities

The implementation according to the OpenAPI Specification guarantees that SIRIUS functionality remains accessible from any programming language ecosystem equipped with standard HTTP communication libraries.

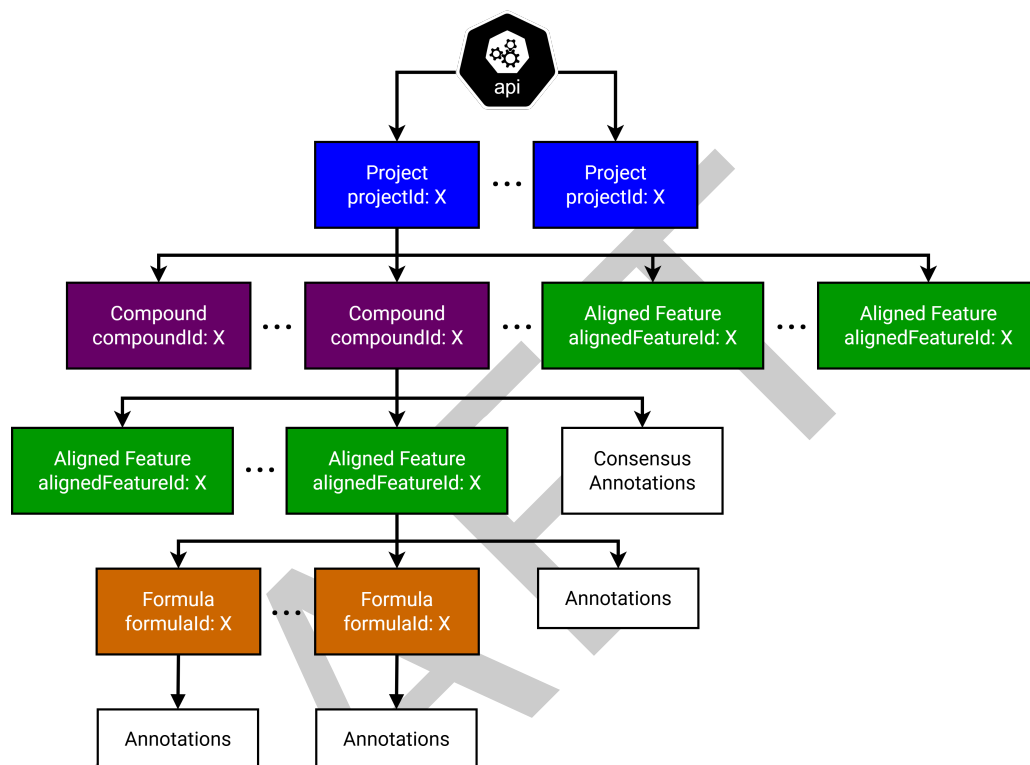


Figure 1: Annotation access hierarchy of SIRIUS

API architecture

The core of the API consists of modules that enable programmatic control over the small molecule annotation workflow. These modules operate on the hierarchical structure of a SIRIUS project (Figure 1), which typically correlates to one LC-MS/MS experiment that can include multiple runs. SIRIUS operates on preprocessed data, accepting either preprocessed spectral inputs or raw mzML files that undergo automated preprocessing. This hierarchy begins at the molecular level, where compounds (purple in Figure 1) represent individual chemical entities. During mass spectrometric analysis, a single compound can generate multiple features, signals from molecules or ions detected across experimental runs, encompassing different ionization states that vary in retention time and mass-to-charge ratio. Alignment of these features over the different runs produces aligned features (green in Figure 1), the main access point for annotations in SIRIUS. This one-to-many relationship enables SIRIUS to provide consensus annotations by integrating evidence across features associated with the same compound. The API focuses on enabling programmatic access to molecular annotations rather than low-level signal processing, though experimental data like traces remain accessible when needed.

Projects API

The projects API module administers SIRIUS project spaces (blue in Figure 1), which are isolated computational environments containing input spectral data and associated analysis results. This module allows to manage (create, open, close, delete) projects and import

106 mass spectrometry data from various formats. It further provides access to prediction vector
107 definitions utilized by CSI:FingerID and CANOPUS.

108 **Compounds API**

109 The compounds API module manages compound entities (purple in [Figure 1](#)) within projects.
110 Compound manipulation operations encompass addition, deletion, and information retrieval.

111 **Features API**

112 The features API module facilitates manipulation of aligned features (green in [Figure 1](#)) as well
113 as retrieval of their associated annotation results. Analogous to the compounds API, available
114 operations include addition, deletion, and information retrieval. Accessible annotations include
115 essentially all computational results generated by methods implemented within SIRIUS.

116 **Searchable databases API**

117 The searchable databases API module manages custom spectral and structural database
118 resources. This module supports creation and deletion of custom databases, import of new
119 entries into existing databases, retrieval of database metadata, and enumeration of all custom
120 and built-in databases available within the SIRIUS environment. As spectra must be annotated
121 with a structure, any spectral library also functions as a structure database.

122 **Jobs API**

123 The jobs API module controls annotation job workflows. Users can define custom job
124 configurations specifying which analytical methods to apply to designated datasets. These
125 jobs can then be executed, monitored, terminated, or deleted as needed. Job configurations
126 can be saved for reuse, and their command-line equivalents retrieved for reproducibility.

127 **Client libraries**

128 Built on the RESTful API, *PySirius*, *RSirius*, and a Java client library provide idiomatic
129 interfaces for Python, R, and Java users, respectively. The libraries are automatically generated
130 from the OpenAPI specification using the [OpenAPI Generator](#), occasionally requiring custom
131 templates and automated patches for full compatibility. The SIRIUS development team actively
132 maintains all client libraries to ensure continued compatibility and feature parity with the API.

133 All client libraries include *SiriusSDK* helper classes that manage the SIRIUS REST service
134 lifecycle. These classes enable starting the service from within code with automated process
135 detection, shutdown and restart capabilities, and attachment to existing instances. Methods
136 return a central API client for streamlined access to all API modules and endpoints. Researchers
137 can now analyze tandem mass spectrometry data with SIRIUS using standard Python, R and
138 Java code. This includes loading data, running jobs, and retrieving results without leaving
139 their analytical environment. The clients enable sharing of complete workflows in contained
140 environments rather than describing multi-tool methodologies.

141 **Updates and maintainability**

142 A CI/CD pipeline automatically generates and tests the R and Python client libraries against
143 the latest SIRIUS API for each new release. The new client versions are subsequently published
144 to conda-forge. This automated workflow ensures the client libraries remain maintainable,
145 up-to-date, and resilient as new clients are added. The SIRIUS Java SDK is part of the SIRIUS
146 main application and is therefore built and distributed alongside it.

Expandability and contribution

The OpenAPI specification enables generation of client libraries for any [supported programming language](#) of the OpenAPI Generator. We welcome community contributions of additional client libraries and relegate interested readers to our [contribution guidelines](#).

Installation via conda-forge

Installation of either *PySirius* or *RSirius* via conda-forge automatically installs SIRIUS as a dependency.

```
conda install -c conda-forge py-sirius-ms # Python client library (PySirius)
conda install -c conda-forge r-sirius-ms  # R client library (RSirius)
conda install -c conda-forge sirius-ms    # SIRIUS (included in the above)
```

Examples and documentation

Minimalistic examples illustrating the fundamental usage of *RSirius* and *PySirius* are provided within the client library source code repository. Exemplary implementations demonstrating the usage of individual functions corresponding to each [API module](#) are available in the test suites for *RSirius* and *PySirius*. Comprehensive documentation for *RSirius* and *PySirius* is also provided in the repository.

Research impact statement

SIRIUS and its contained methods have been named “methods to watch” in Nature Methods on multiple occasions ([Singh, 2020, 2023](#)). The core methods, such as CSI:FingerID and CANOPUS, are established standards in the field, evidenced by hundreds of citations for their foundational papers. Furthermore, the web services for these tools recently surpassed a combined [one billion queries](#), demonstrating large scale adoption.

The Java client has enabled integration of SIRIUS into major metabolomics platforms. *MZmine* has incorporated SIRIUS functionality directly into its workflow since version 4.8, while Agilent’s *Mass Hunter Explorer* v2.0 integrates SIRIUS capabilities through a proprietary client implementation. The RforMassSpectrometry initiative is currently adapting *RSirius* as the foundation for *RuSirius*. *PySirius* and *RSirius* show more than fifty thousand combined downloads on [Anaconda.org](#).

Availability

Both the [client library source code](#) and the [SIRIUS source code](#) are openly available on GitHub. The client libraries are distributed via conda-forge as [py-sirius-ms](#) (*PySirius*) and [r-sirius-ms](#) (*RSirius*). SIRIUS itself is available via conda-forge as [sirius-ms](#) or as direct [releases](#) on GitHub. The Java client is hosted on a GitLab [package registry](#), and the SIRIUS API implementation is part of the [SIRIUS repository](#).

AI usage disclosure

AI assistance was utilized for text editing of this manuscript and the [contribution guidelines](#) of the client repository. All changes proposed by AI systems were manually reviewed.

Competing interests

MF, ML, MAH, KD and SB are co-founders of Bright Giant GmbH.

Acknowledgements

NAH, SB, MAH and MEL are supported by the Thüringer Ministerium für Wirtschaft, Wissenschaft und Digitale Gesellschaft (TMWWDG) with funds from the European Union as part of the European Regional Development Fund (ERDF, 2023 VFE 0003 and 0029).

References

- Böcker, S., & Dührkop, K. (2016). Fragmentation trees reloaded. *J Cheminformatics*, 8, 5. <https://doi.org/10.1186/s13321-016-0116-8>
- Böcker, S., Letzel, M., Lipták, Zsuzsanna, & Pervukhin, A. (2009). SIRIUS: Decomposing isotope patterns for metabolite identification. *Bioinformatics*, 25(2), 218–224. <https://doi.org/10.1093/bioinformatics/btn603>
- Böcker, S., & Rasche, F. (2008). Towards de novo identification of metabolites by analyzing tandem mass spectra. *Bioinformatics*, 24, 149–155. <https://doi.org/10.1093/bioinformatics/btn270>
- da Silva, R. R., Dorrestein, P. C., & Quinn, R. A. (2015). Illuminating the dark matter in metabolomics. *Proc Natl Acad Sci U S A*, 112(41), 12549–12550. <https://doi.org/10.1073/pnas.1516878112>
- Dührkop, K., Fleischauer, M., Ludwig, M., Aksenov, A. A., Melnik, A. V., Meusel, M., Dorrestein, P. C., Rousu, J., & Böcker, S. (2019). SIRIUS 4: A rapid tool for turning tandem mass spectra into metabolite structure information. *Nat Methods*, 16(4), 299–302. <https://doi.org/10.1038/s41592-019-0344-8>
- Dührkop, K., Nothias, L. F., Fleischauer, M., Reher, R., Ludwig, M., Hoffmann, M. A., Petras, D., Gerwick, W. H., Rousu, J., Dorrestein, P. C., & Böcker, S. (2021). Systematic classification of unknown metabolites using high-resolution fragmentation mass spectra. *Nat Biotechnol*, 39(4), 462–471. <https://doi.org/10.1038/s41587-020-0740-8>
- Dührkop, K., Scheubert, K., & Böcker, S. (2013). Molecular formula identification with SIRIUS. *Metabolites*, 3, 506–516. <https://doi.org/10.3390/metabo3020506>
- Dührkop, K., Shen, H., Meusel, M., Rousu, J., & Böcker, S. (2015). Searching molecular structure databases with tandem mass spectra using CSI:FingerID. *Proc Natl Acad Sci U S A*, 112(41), 12580–12585. <https://doi.org/10.1073/pnas.1509788112>
- Huber, F., Verhoeven, S., Meijer, C., Spreeuw, H., Castilla, E. M. V., Geng, C., Hooft, J. J. van der, Rogers, S., Belloum, A., Diblen, F., & Spaaks, J. H. (2020). Matchms - processing and similarity evaluation of mass spectrometry data. *Journal of Open Source Software*, 5, 2411. <https://doi.org/10.21105/joss.02411>
- Louail, P., & Rainer, J. (2025). *RforMassSpectrometry/Metabonaut: Metabonaut version 1.0.0*. <https://doi.org/10.5281/zenodo.15062929>
- Ludwig, M., Fleischauer, M., Dührkop, K., Hoffmann, M. A., & Böcker, S. (2020). De novo molecular formula annotation and structure elucidation using SIRIUS 4. In S. Li (Ed.), *Computational methods and data analysis for metabolomics* (Vol. 2104, pp. 185–207). Springer US. https://doi.org/10.1007/978-1-0716-0239-3_11
- Röst, H. L., Schmitt, U., Aebersold, R., & Malmström, L. (2014). pyOpenMS: A Python-based interface to the OpenMS mass-spectrometry algorithm library. *Proteomics*, 14, 74–77. <https://doi.org/10.1002/pmic.201300246>
- Schmid, R., Heuckeroth, S., Korf, A., Smirnov, A., Myers, O., Dyrland, T. S., Bushuiev, R., Murray, K. J., Hoffmann, N., Lu, M., Sarvepalli, A., Zhang, Z., Fleischauer, M., Dührkop, K., Wesner, M., Hoogstra, S. J., Mokshyna, O., Brungs, C., Ponomarov, K., ... Pluskal,

- 231 T. (2023). Integrative analysis of multimodal mass spectrometry data in MZmine 3. *Nat*
232 *Biotechnol*, 41(4), 447–449. <https://doi.org/10.1038/s41587-023-01690-2>
- 233 Singh, A. (2020). Tools for metabolomics. *Nat Methods*, 17(1), 24–24. [https://doi.org/10.](https://doi.org/10.1038/s41592-019-0710-6)
234 [1038/s41592-019-0710-6](https://doi.org/10.1038/s41592-019-0710-6)
- 235 Singh, A. (2023). Annotating unknown metabolites. *Nat Methods*, 20(1), 33. [https:](https://doi.org/10.1038/s41592-022-01735-5)
236 [//doi.org/10.1038/s41592-022-01735-5](https://doi.org/10.1038/s41592-022-01735-5)
- 237 Stravs, M. A., Dührkop, K., Böcker, S., & Zamboni, N. (2021). MSNovelist: De novo structure
238 generation from mass spectra. *bioRxiv*. <https://doi.org/10.1101/2021.07.06.450875>

DRAFT