DataAPIs

2021 report - Python Data APIs Consortium

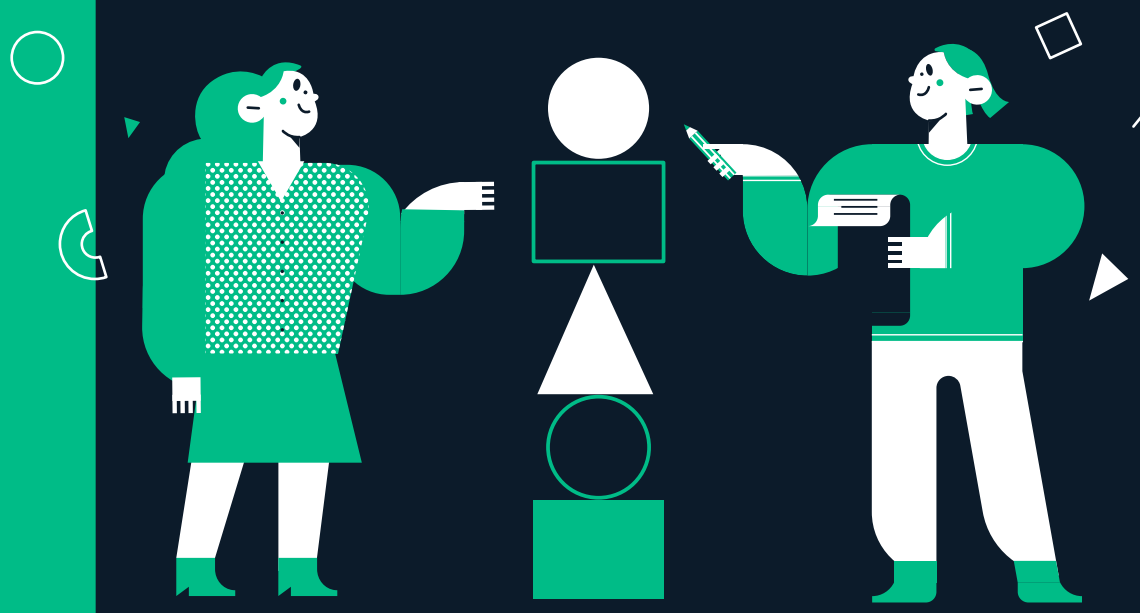


Table of Contents

Executive summary

Over the past decade, we've witnessed increased fragmentation within the Python data ecosystem. This fragmentation largely stems from the increased popularity of data science, numerical computation, and deep learning and the proliferation of new libraries intended to serve those needs. While the growth of new libraries and frameworks has contributed to significant innovation within the ecosystem, the resulting fragmentation has a cost, as users and downstream library maintainers cannot readily interoperate among the various libraries and must frequently develop programs which only target a single library. The Python Data APIs consortium aims to address this problem by standardizing the fundamental data structures of arrays and dataframes and an associated set of common APIs for working with those data structures, thus facilitating interchange and interoperation.

To fulfill this aim, the consortium invited industry stakeholders and maintainers of array and dataframe libraries to participate in the standardization decision-making process. That decision-making process had the following objectives for 2021:

- Define a standardization methodology.
- Develop the tooling necessary to support the standardization methodology.
- Publish an array API standard RFC.
- Publish a dataframe interchange protocol RFC.
- FInalize 2021.0x API standards after community review.

After releasing the 2021.0x array API standard and dataframe interchange protocol, focus shifted to their adoption for the remainder of 2021 and continuing into early 2022. Evolving alongside the specifications, reference library implementations provide guidance for specification adoption by other libraries and frameworks. This coevolution ensures specification alignment and lays the foundation for the 2022.0x revision of the array and dataframe API standards. For more information, consult the formal text of the respective specifications:

Array API: https://data-apis.org/array-api/latest/
Dataframe interchange protocol:
https://data-apis.org/dataframe-protocol/latest/index.html

History

While the Python programming language was not designed for numerical computing, the language gained initial popularity in the scientific and engineering community soon after its release. The first array computing library for numerical and scientific computing in Python was Numeric, developed in the mid-1990s. To better accommodate this library and its use cases, Python's syntax was extended to include indexing syntax.

In the early 2000s, a similar library, Numarray, introduced a more flexible data structure. Numarray had faster operations for large arrays. However, the library was slower for small arrays. Subsequently, both Numeric and Numarray coexisted to satisfy different use cases.

In early 2005, NumPy was written to unify Numeric and Numarray as a single array package by porting Numarray's features to Numeric. This effort was largely successful and resolved the fragmentation at the time, and, for roughly a decade, NumPy was the only widely used array library. Building on NumPy, pandas was subsequently introduced in 2008 in order to address the need for a high performance, flexible tool for performing quantitative analysis on labeled tabular data.

Over the past 5 years, the rise of deep learning and the emergence of new hardware has led to a proliferation of new libraries and a corresponding fragmentation within the PyData array and dataframe ecosystem. These libraries often borrowed concepts from, or entirely copied, the APIs of older libraries, such as NumPy, and then modified and evolved those APIs to address new needs and use cases. While the communities of each individual library discussed interchange and interoperation, until the founding of this consortium, no process for coordination among libraries arose to avoid further fragmentation and to arrive at a common set of API standards.

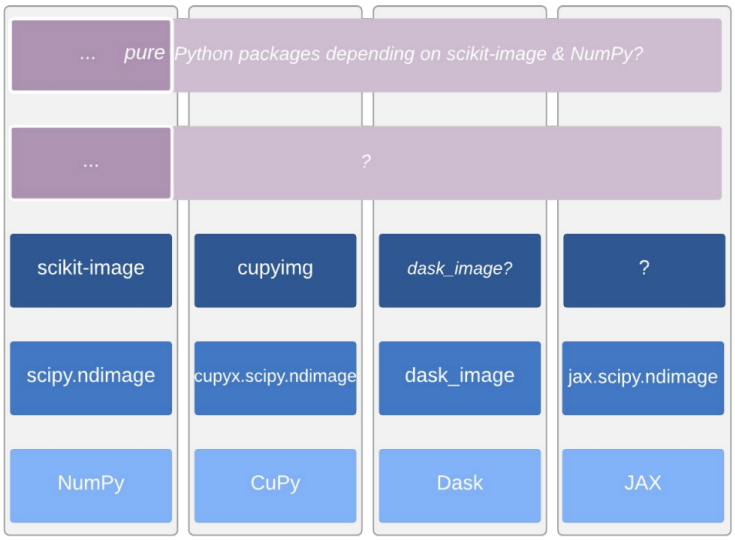


Figure 1. Downstream library silos. Across the Python computing landscape, downstream libraries have little choice but to tailor their implementations to a specific array library (as indicated by the grey vertical boxes) due to API fragmentation and non-portability. A goal of this consortium is to allow downstream libraries to break out of their silos and seamlessly support multiple array libraries.

The genesis for the consortium grew out of many conversations among maintainers during 2019-2020. During those conversations, it quickly became clear that any attempt to write a new reference library to fix the current fragmentation was infeasible. Unlike in 2005, too many different use cases and varying stakeholders now exist. Furthermore, the speed of innovation of both hardware and software is simply too great. In May 2020, an initial group of maintainers and industry stakeholders assembled to form the Consortium for Python Data API Standards to begin drafting specifications for array and dataframe APIs, which could then be adopted by each of the existing array and dataframe libraries and any new libraries which arose.

Motivation for starting Data APIs

Today, Python users have a wealth of choice for libraries and frameworks for numerical computing, data science, machine learning, and deep learning. New frameworks pushing forward the state of the art in these fields appear every year. One unintended consequence of all this activity and creativity has been fragmentation in the fundamental building blocks—multidimensional arrays (a.k.a. tensors) and dataframes—that underpin the Python data ecosystem. Tensors are fragmented among Tensorflow, PyTorch, NumPy, CuPy, MXNet, Xarray, Dask, and others. Dataframes are fragmented among pandas, PySpark, cuDF, Vaex, Modin, Dask, Ibis, Apache Arrow, and more.

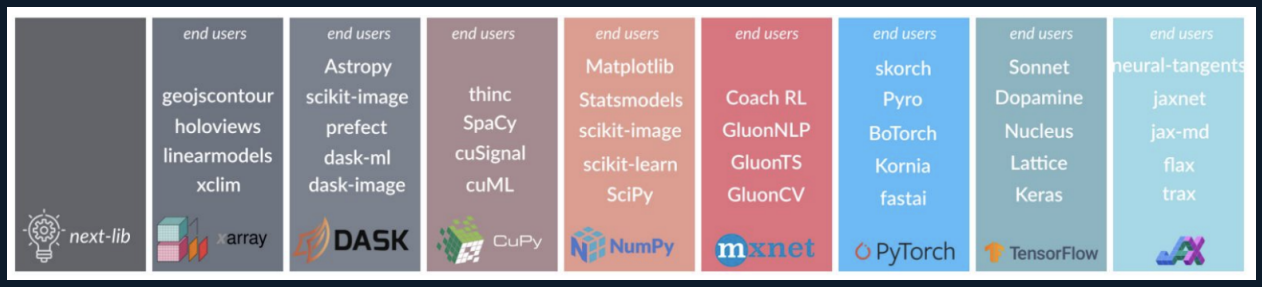


Figure 2. Ecosystem fragmentation. Summary of libraries that depend on or extend individual array libraries.

This fragmentation comes with significant costs, from reinvention and implementation of arrays and dataframes to the proliferation of user guides providing guidance on how to convert between, and interoperate among, libraries. Too often, the APIs of each library are largely similar, but each have enough differences that end users have to relearn and rewrite code in order to work with multiple libraries. This process can be very painful given that the translation is not seamless. The array and dataframe API standards aim to address this issue by specifying standardized APIs for the most common array and dataframe operations.

Our goal is not to take an existing API and use it as the new standard, as good reasons often exist for the current inconsistencies and diversity among current libraries. For instance, the most obvious candidates for existing APIs are NumPy for arrays and pandas for dataframes. However, these libraries were not designed with non-CPU devices, graph-based libraries, or JIT compilers in mind. Choices made in this new API standard are often aligned with these APIs as reference, but the design choices are different, where necessary, in order to ensure that all existing and future array and dataframe libraries can adopt and adhere to the same set of common operations.

Consortium formation and goals

The Data APIs consortium comprises both maintainers of array and dataframe libraries and the industry stakeholders who depend on those libraries. The consortium's goals are twofold. The first goal is to enable writing code and packages which can support multiple array and/or dataframe libraries. The second goal is to facilitate interchange among the array and dataframe data structures.

The consortium is sustained by industry stakeholders, who sponsor and fund the required engineering, technical writing, and participation of key community contributors. A working group sets the high-level goals, requirements, and user stories necessary to start making initial decisions. With this framework in place, engineers build required tooling, prepare data, and draft specification documents. The specification drafts undergo multiple iterations based on working group feedback. Once specification drafts have a concrete outline, input from library maintainer members is requested. Upon working group approval, drafts are released as a Request for Comments (RFC) as part of a public review process.

Throughout the standardization process, certain decisions are defined as in and out of scope. The objectives defined as in scope include syntax and semantics of functions and objects in the API, casting rules, broadcasting, indexing, Python operator support, data interchange, and device support. However, the objectives do not include execution semantics, non-standard data types, masked arrays, I/O routines, array object subclassing, and C APIs. Accordingly, task scheduling, parallelism, lazy evaluation, error handling, and comprehensive behavior of invalid inputs to functions and methods are not in scope.

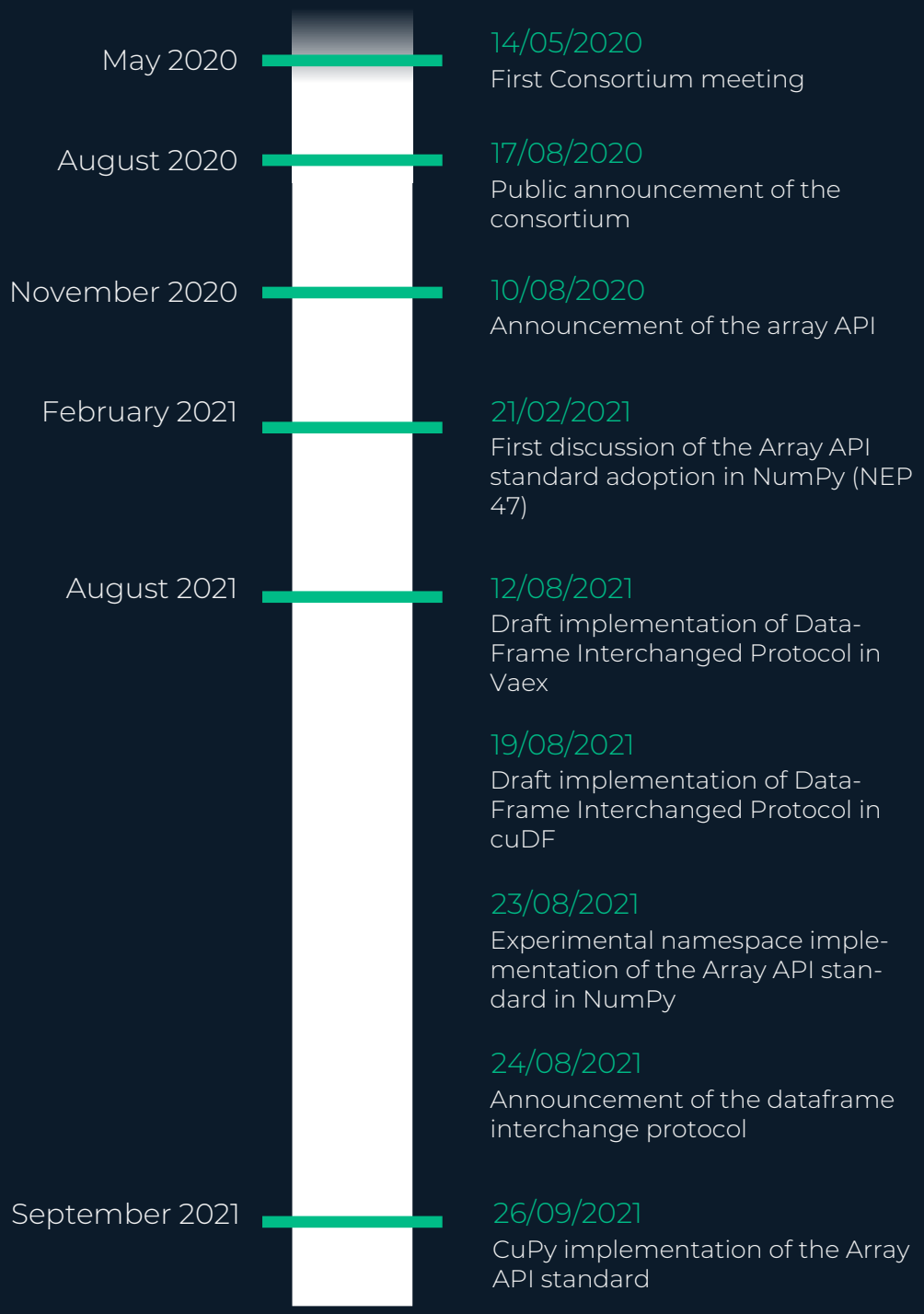
For 2021, the objectives were as follows:

- Define standardization methodology.
- Develop the tooling necessary to support the standardization methodology.
- Publish an array API standard RFC.
- Publish a dataframe interchange protocol RFC.
- FInalize 2021.0x API standards after community review.

Progress in year 1

a) Timeline

A timeline covering the progress of the consortium since its initial meeting is shown below.



b) Organizational goals

The proliferation of array and dataframe libraries is reflective of the heterogeneity in use cases and hardware. Accordingly, to ensure that the specifications produced by the consortium adequately address cross-cutting concerns and find general applicability, consortium membership must be a representative cross-section of array and dataframe libraries and their stakeholders. To this end, the consortium sought to ensure membership representation of the most commonly used array and dataframe libraries.

As of 31 December 2021, the following array libraries are represented in the consortium:

- NumPy
- CuPy
- PyTorch
- JAX
- Dask
- TensorFlow
- Apache MXNet

As of 31 December 2021, the following dataframe libraries are represented in the consortium:

- pandas
- cuDF
- Vaex
- Modin
- Ibis
- Dask
- Koalas
- Apache Arrow

IInstitutional support of array and dataframe libraries is critical to their continued success and evolution. To foster collaboration and dialogue among libraries and industry stakeholders, the consortium sought membership of industry partners representing a cross-section of hardware vendors, development sponsors, and end-users. As of 31 December 2021, the following industry partners are represented in the consortium:

- Google Research
- Intel
- Microsoft
- The D. E. Shaw Group
- LG Electronics
- Quansight

Since its inception, the consortium has held weekly working group meetings, which alternate between array and dataframe topics. In 2021, the consortium held 19 meetings focused on array topics and 18 meetings focused on dataframe topics.

c) Tooling

In order to understand the Python data API landscape, the consortium developed tooling for assessing API divergence, recording downstream usage, and testing specification compliance. The current list of available tools is as follows:

Array API comparison: https://github.com/data-apis/array-api-comparison
Dataframe API comparison:
https://github.com/data-apis/dataframe-api-comparison
Python record API: https://github.com/data-apis/python-record-api
Array API standard test suite: https://github.com/data-apis/array-api-tests

The array API and dataframe API comparison tools facilitate the analysis of the commonalities and differences across libraries with the goal of deriving a common API subset suitable for standardization. By analyzing commonalities and differences, the working group can better ensure consistency in attribute and method names and positional and keyword arguments. The array API comparison tooling supports the following array libraries: NumPy (which serves as the reference API), CuPy, Dask, JAX, MXNet, PyTorch, Pydata/Sparse, and TensorFlow. The dataframe API comparison tooling supports the following libraries: pandas (which serves as the reference API), Dask, cuDF, Vaex, Koalas, Ibis, and Modin.

The python record API tooling is meant to understand how a target Python module is used by downstream consumers. The tooling instruments all function calls when running a library module to obtain API usage information for a target library. Upon running the test suite of a downstream library, collected data is used to derive synthetic APIs. The synthetic APIs are then compared across downstream libraries to access and rank common API usage patterns. Understanding usage patterns is an important component of the standardization process, as such understanding facilitates a data-driven decision making process and ensures that standardization efforts satisfy real-world use cases and downstream consumer needs.

The array API standard test suite enables a target array library to measure array API standard compliance. Similar to a unit testing framework, the test suite runs a battery of tests to verify that array libraries are specification compliant. By providing a ready-to-run test suite, array libraries can leverage test-driven development to both accelerate library development and quickly detect regressions upon library changes.

d) Methodology

Initial consortium discussions focused on how to organize and approach the standardization process. One outcome of these discussions was the determination that arrays and dataframes each required their own standardization methodology due to their diverging needs, use cases, and ecosystem maturity. Accordingly, array and dataframe discussions took place on parallel tracks which arrived at different methodologies with their own results.

For both array and dataframes, a necessary first step was to build API comparison tools in order to provide a high level overview of common APIs, design patterns, and requirements. Comparison of array and dataframe libraries, respectively, not only aided the discovery of common APIs, but also captured the current state of fragmentation across the PyData ecosystem. Subsequently combining the comparison data with usage data obtained from the Python record API tooling allowed finding the most commonly used APIs which could then be used as a starting point for standardization.

The first array standardization milestone was defining a common array data structure. Next, a common API subset was proposed, approved, and subsequently included in the standard. This subset primarily included APIs for element-wise array operations, such as the computation of transcendental functions, value comparison, and arithmetic. Standardization efforts then progressed to non-trivial functions (e.g., reductions, array creation and manipulation, and linear algebra) where the working group sought to understand API design fragmentation and arrive at minimal, unified APIs capable of satisfying individual array library requirements (e.g., device agnosticism, copy-view behavior, and supported data types).

The evolution of the array API standard specification is an iterative process in which key maintainers continually monitor, review, and guide standardization efforts. Once an initial draft proposal is ready, the proposal is made public as an RFC in order to solicit community feedback. Once a proposal is approved, reference array libraries begin implementing the proposed features and/or changes, providing feedback throughout the process in order to further refine the proposal. Reference library implementations subsequently provide other libraries a starting point for their own efforts to ensure conformance with the array API specification.

In contrast to array libraries which enjoy significant agreement and have design requirements which are relatively well understood, dataframe libraries are wildly divergent in both their implementations and requirements. Among dataframe libraries, the definition of a dataframe varies significantly, resulting in high fragmentation. As a consequence, applying the same methodology as used for array API standardization proved infeasible—namely, finding and standardizing common API subsets. Accordingly, standardization efforts focused on the fundamental problem of interchanging data between dataframe libraries, thus leading to a dataframe interchange protocol. Similar to array API standardization, the protocol followed a similar release schedule: proposal, RFC, community feedback, and reference library implementations.

At the time of this report, multiple array and dataframe libraries are working on specification compliance. Their efforts help uncover edge cases, challenges, and needs for further specification refinement, thus feeding an iterative process for specification evolution.

Array API standard

a) Current state

The array API specification contains one array object, eleven data type literals, one device object, four constants, and more than 125 functions for array creation and manipulation, element-wise mathematics and comparison, statistics, and linear algebra. The first version of the specification includes a linear algebra extension, but does not yet include support for complex number data types.

The specification defines a minimal array object with associated attributes and methods. Attributes include those for returning the number of dimensions, shape, size, data type, device, and transpose. In order to support native Python operators, the specification defines expected behavior for a common subset of dunder methods. The specification further defines an additional array object method for array interchange via DLPack. An array object is expected to support the following data types: bool, (u)int8/16/32/64, and float32/64.

A few design topics have proved difficult to standardize, resulting in ongoing discussions concerning expected behavior and requirements. These topics include mutability and copies/views, data-dependent output shapes, and device-aware zero-copy control.

Mutable operations are important for strided in-memory array implementations, such as in NumPy. However, for libraries based on immutable data structures and/or delayed evaluation, such as JAX, MXNet, Dask, and TensorFlow, mutable operations are problematic. To accommodate both needs, the specification requires support for inplace operators and slice assignment, but the specification does not require support for an *out* keyword in element-wise operations and includes a warning to steer users away from mixing mutation operations with views, as doing so may result in implementation-specific behavior.

Data-dependent output shapes are problematic due to static memory allocation requirements and delayed evaluation as found in array libraries employing graph-based computational models (e.g., TensorFlow, JAX, Dask, PyTorch, and others). The specification accommodates these libraries by identifying operations having data-dependent output shapes and making these operations optional.

Finally, the specification includes a device-aware zero-copy protocol using DLPack. This protocol allows for array interchange and interoperation. The protocol allows copying array data between arbitrary devices and supports multiple data types and basic array attributes. To support this protocol, the specification includes a *from_dlpack* API and defines a mechanism for future extensions.

The array API standard is currently published for community review. Open discussion points include specification extensions (FFTs, random number generation, and deep learning), additional API standardization candidates, and view reinterpretation. The API specification is available on the publicly accessible consortium website. Community feedback can be provided by opening an issue on the specification's GitHub repository. For previous discussions, one can consult meeting minutes and existing issues and pull requests as found on GitHub.

b) Adoption

NumPy has been, and continues to be, a primary reference implementation for array libraries. NumPy's adoption of the array standard is a critical first step in ensuring general adoption among both libraries and users and galvanizing other libraries to commence work on specification compliance. To this end, NEP 47 was proposed and merged in the NumPy repository. Following NEP 47, an experimental implementation of the standard was merged into NumPy under a dedicated *array_api* namespace.

In addition to NumPy, PyTorch, CuPy, and MXNet have moved toward specification adoption. PyTorch is adding support for the standard in its main namespace (issue 54581) with some issues remaining to be resolved before achieving full compliance (tracking issue 58743). Endeavoring to be a drop-in NumPy replacement, CuPy agreed to adopt the standard (issue 4789) and is following the experimental NumPy implementation by adding support for the standard in a dedicated *array_api* namespace. MXNet proposed standard adoption in RFC 20501 and is tracking adoption in issue 20579.

JAX, Dask, and TensorFlow have verbally committed to adopt the array API standard; however, their progress toward adoption has not been made publicly available.

c) Next steps

While the consortium has made significant progress toward array API standardization, much work remains to be done. Future work may be broken down into four areas.

- **Implementation.** Accelerating adoption of the standard is critical to allowing downstream libraries, such as SciPy, Scikit-learn, scikit-image, and other domain-specific libraries, to begin using the standard and support array library interoperability.

- **Compliance.** Completion of the library-independent test suite is necessary for measuring specification compliance, guiding array library development, and providing a mechanism for downstream users to access adoption progress across the PyData ecosystem.

- **Extension.** A specification is a living document and, as array libraries continue to evolve, so too must the specification. Opportunities for further API extension include standardized APIs for complex numbers, FFTs, deep learning, and random number generation.

- **Outreach.** The faster the ecosystem adopts the standard, the sooner the consortium can focus efforts on meeting the needs of current and future array libraries. Community outreach is necessary to accelerate adoption. Outreach may consist of blog posts, tutorials, conference talks, prototype implementations for downstream API consumers, and community engagement on issue trackers and developer channels.

Dataframe interchange protocol

a) Current state

Dataframes have presented significant challenges for standardization given their implementation heterogeneity and complexity. Among dataframe libraries, the very definition of what constitutes a dataframe varies. Accordingly, rather than attempt to standardize a complete API specification similar to arrays, efforts were focused on standardizing a dataframe interchange protocol in order to allow zero-copy data interchange among dataframe libraries.

For this purpose, the consortium defined a minimal dataframe as an ordered collection of columns which has the following characteristics:

- A dataframe or column may be chunked (i.e., data may not be contiguous in memory).
- A column is defined as a one-dimensional array with a data type and missing data support.
- Column names must be unique strings.

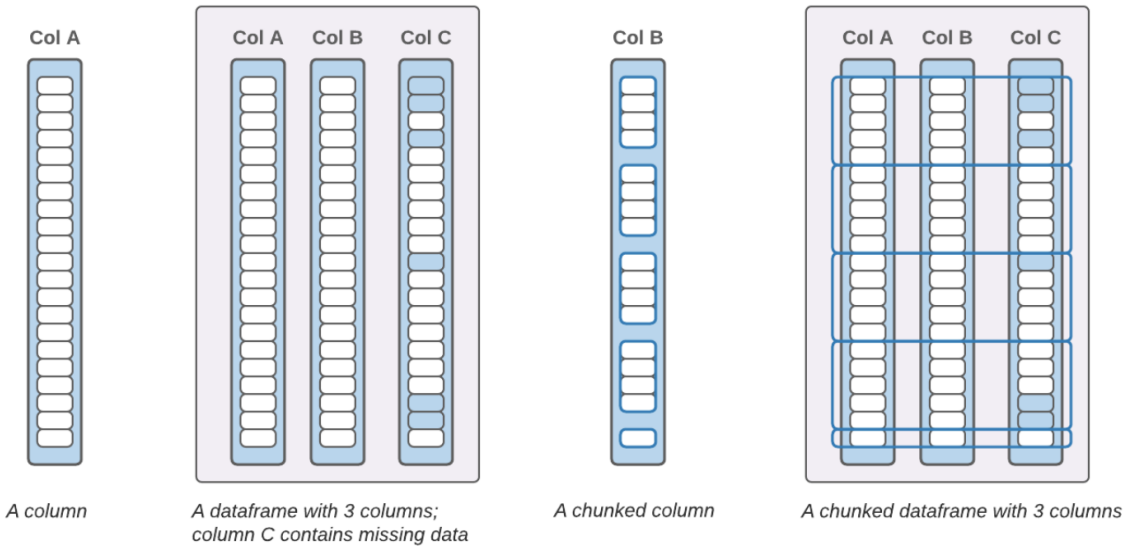


Figure 3. A conceptual model of a minimal dataframe.

Building on this working dataframe model, the consortium proposed a dataframe interchange protocol to allow data interchange change among dataframe libraries. The protocol provides a basic means for inspecting dataframe properties, such as the number of columns, column names, and column data types. The protocol does not assume a particular dataframe implementation, instead choosing to describe memory down to the level of contiguous one-dimensional blocks of memory (i.e., buffers). By specifying at the buffer level, connecting this protocol with the array API standard is possible via DLPack with the restriction that the libraries involved in the interchange must all support the device on which the data resides.

As part of the protocol, the consortium proposed a *from_dataframe* API for dataframe libraries to include in their top-level namespace. This API provides a standardized API for dataframe creation and a universal mechanism by which dataframe libraries can construct a library-specific dataframe instance from any other dataframe object. As a consequence of this protocol, libraries which consume dataframes will enjoy enhanced portability and be able to better support dataframe object interoperation.

The current version of the dataframe interchange protocol is available for public review. Community feedback can be provided by opening an issue on the specification's GitHub repository.

b) Adoption

Similar to NumPy, pandas serves as a primary reference implementation for dataframe libraries. Thus, pandas' adoption of the interchange protocol is a critical first step in ensuring general adoption among dataframe libraries. To this end, a prototype pandas implementation of the interchange protocol was drafted with the goal of moving to the pandas repository upon further community refinement and feedback.

In addition to pandas, cuDF and Vaex have working draft implementations of the protocol (see cuDF 9071 and Vaex 1509).

c) Next steps

Once the dataframe interchange protocol is more widely adopted, focus will shift toward specifying a more complete dataframe API. In contrast to array libraries, dataframe library maintainers express less general agreement regarding API design and requirements. Accordingly, standardization efforts will demand tailored methodologies for dataframe API standardization.

Initial work will be limited to a minimal developer-focused dataframe API with clear semantics, no performance cliffs, and explicit APIs. The primary goal will be to distill the high complexity of dataframe APIs into a core set of composable functions from which higher order functions can be derived. This set of functions will satisfy the dual demands of allowing end-user API specialization and providing a lower-level intermediate layer for common operations over heterogeneous backends.

Note: *Since this report was finalized on 31 December 2021, the consortium workgroup has shifted focus to standardizing user-focused pandas API behavior with plans of returning to a developer-focused dataframe API toward the end of 2022/beginning of 2023. Initial 2022 priorities will be posted on the data-apis/dataframe-api issue tracker.*

What's next in 2022

The consortium is aiming for long-term sustainability. Achieving this aim requires accelerating specification adoption, measuring compliance, and soliciting new and continued sponsorships from key stakeholders. In 2022, the array API specification will release an updated revision which is projected to include complex number data type support, fast fourier transforms (FFTs), and additional APIs for common array operations. The dataframe API will draft a minimal developer-focused API specification exposing APIs which end-user dataframe APIs can target independent of device. And finally, continued outreach to array and dataframe library communities will ensure increased adoption, compliance, and alignment.

A proposed roadmap is provided below.

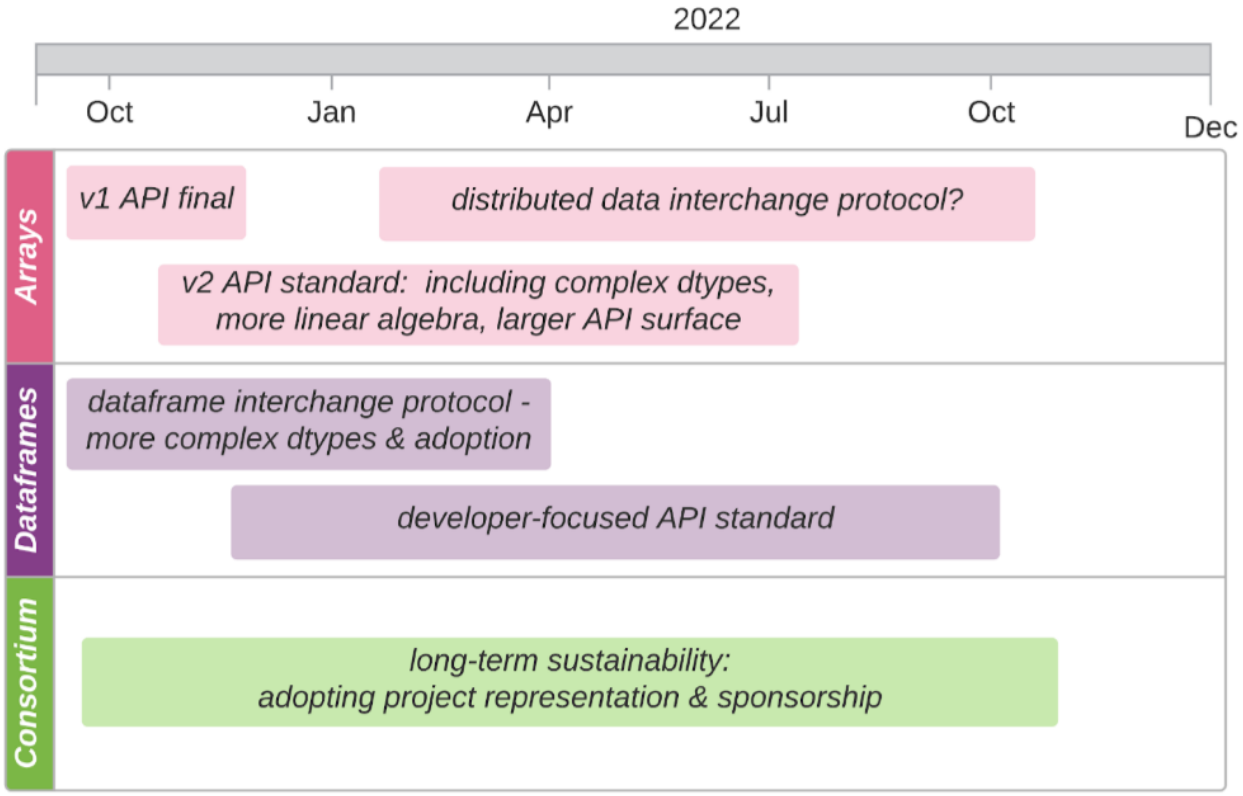


Figure 4. Proposed roadmap for 2022.

Acknowledgements

a) Members

As of 31 December 2021, consortium members, along with their sponsor and array/dataframe library affiliations, are as follows:

Current

Aaron Meurer (Quansight)
Adam Paszke (Google Research, PyTorch, JAX)
Alex Baden (OmniSciDB)
Andreas Mueller (Microsoft)
Areg Melik-Adamyan (Intel)
Arvid Bessen (The D. E. Shaw Group)
Ashish Agarwal (Google Research, TensorFlow)
Ashwin Srinath (cuDF)
Athan Reines (Quansight)
Carlo Curino (Microsoft)
Devin Petersohn (Modin)
Edward Loper (Google Research, TensorFlow)
Hyukjin Kwon (Apache Spark, Koalas)
Javad Heydari (LG)
Jeff Reback (pandas, Ibis)
John Kirkham (Dask)
Joris Van den Bossche (pandas, GeoPandas, Apache Arrow)
Leo Fang (CuPy)
Maarten Breddels (Vaex)
Markus Weimer (Microsoft, MXNet)
Matthew Barber (Quansight)
Mike McCarty (NVIDIA)
Oleksandr Pavlyk (Intel)

Ralf Gommers (Quansight, NumPy)
Rohan Jain (Google Research, TensorFlow)
Sheng Zha (Apache MXNet, ONNX)
Stephan Hoyer (Google Research, JAX, NumPy, XArray)
Stephannie Jiménez Gacha (Quansight)
Takuya Ueshin (Apache Spark, Koalas)
Vasily Litvinov (Intel)
Xiao Li (Apache Spark)

Emeritus

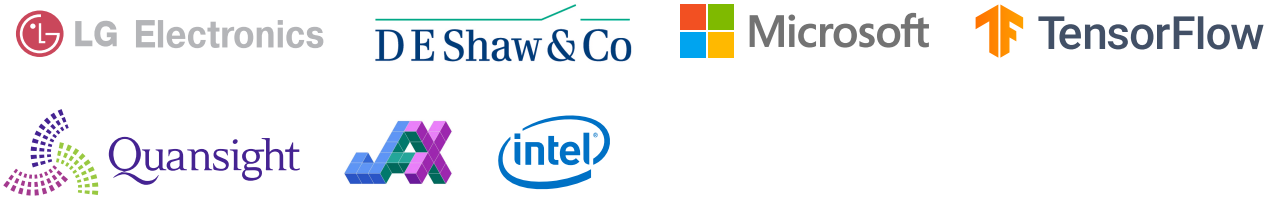
Alex Passos (Google Research, TensorFlow)
Anthony Scopatz (Quansight)
Keith Kraus (cuDF)
Marc Garcia (Quansight, pandas, Ibis)
Mohak Shah (LG)
Paige Bailey (Google Research, TensorFlow, JAX)
Saul Shanabrook (Quansight)
Tiankai Tu (The D. E. Shaw Group)
Tom Augspurger (Dask, pandas)
Unmesh Kurup (LG)

b) Contributors

A list of consortium contributors may be found in the array API specification README. In addition to the people listed, standardization efforts have benefited from dozens of others who provided design feedback, organized workshops and other outreach activities, and/or reviewed implementations of API specifications in NumPy, CuPy, PyTorch, Vaex, and cuDF.

c) Sponsors

The engineering, technical writing, and organizational effort needed to bootstrap this consortium and draft the first versions of the array and dataframe API standards is supported by the following consortium sponsors:



Additional content

a) Blog posts

"Announcing the Consortium for Python Data API Standards", 17 Aug 2020, https://data-apis.org/blog/announcing_the_consortium/

"First release of the Array API Standard", 10 Nov 2020, https://data-apis.org/blog/array_api_standard_release/

"An RFC for a dataframe interchange protocol", 24 Aug 2021, https://data-apis.org/blog/dataframe_protocol_rfc/

b) Talks

Ralf Gommers, "Standardizing on a single N-dimensional array API for Python", Apache MXNet Day, 15 Dec 2020

Ralf Gommers, "Python array API standardization – current state and benefits", GTC, 10 Nov 2021

Yao-Lung Leo Fang, "High-Performance Python GPU Programming with CuPy", APS Scientific Computation Seminar at Argonne National Laboratory, 24 May 2021

Stephannie Jimenez, "Data APIs: Estandarización de arreglos N-dimensionales y dataframes", Scipy Latam, 12 Dec 2021

a) Panels

Ashish Agarwal, Michael Bauer, Emilio Castillo, Ralf Gommers, Adam Paszke, Wen-Ming Ye, Sheng Zha, "NumPy API standardization across frameworks", Apache MXNet Day, 15 Dec 2020

Frederic Bastien, Michael Bauer, Ralf Gommers, Keith Krauss, Adam Paszke, Vartika Singh, Sheng Zha, "Standardizing on an Array API for Python across Deep Learning Frameworks", GTC, 13 Apr 2021

b) Webinars

Athan Reines, Lais Carvalho, and Ralf Gommers, "Python Data APIs Quickshop", 20 Oct 2020, https://youtu.be/b11wBxYCEGE

b) Podcasts

Ralf Gommers, Areg Melik-Adamyan, and Travis Oliphant, "Building Common Standards for Python Data APIs", Code Together, 3 Nov 2021, https://soundcloud.com/codetogether/building-common-standards-for-python-data-apis

Learn more

https://data-apis.org/annual-reports/

DataAPIs