REQUEST FOR INFORMATION (RFI): ALLOTROPE DATA FORMAT (ADF) LIBRARY REFACTORING 2020

Allotrope Foundation

1500 K Street, NW

Washington, DC 20005

May 2020

Executive Summary: ADF Library Refactoring

The Allotrope Data Format (ADF) is a vendor, platform, and method-agnostic format adaptable to store data, contextual metadata, and any ancillary files. Advanced semantic technology is used as the foundation for the contextual metadata layer.

The current ADF software library is written in Java which is the language of choice for many enterprise applications. An additional transpiled version is written in C#. In recent years Java and C# are being overshadowed within the data science community by other languages with a growing momentum such as Python. As a long-term solution Allotrope Foundation (AF) would like to evaluate the technical alternatives to refactor the current ADF library written in Java to a thin, efficient, portable, and easily wrappable middleware written in C++ in order to address the support for additional software languages as well as other long-term objectives such as performance and portability.

While AF is recognizing that there are several technical alternatives for the long-term refactoring of the ADF library, AF would also like to evaluate short-term and intermediate technical solutions to enable the use of the ADF library with a more specific language such as Python, a different user groups such as the scientific computing community and different applications in the analytical space.

This RFI is written to evaluate an optimal path forward for the evolution of the ADF class libraries to meet the Foundation's goals. The RFI is organized around the long-term and the short-term goals, objectives, and alternatives where a reference link is provided to the appropriate response section.

Respondents may choose to respond to any of the alternatives and they are free to suggest their own solution. Especially they are encouraged to address the shorter-term Python alternatives.

1 Purpose and Problem Statement

The purpose of this RFI is for Allotrope Foundation (AF) to better understand potential solutions and to assess the collated information with the Allotrope Product Team and community as needed to align on an optimal path forward for the evolution of the Allotrope Data Format (ADF) class libraries to meet the Foundation's goals. Subsequent to the RFI and the information provided by the respondents the Allotrope Product Team may create and release a formal RFP for applicable development. If you are unable to submit a detailed RFI response at this point in time please respond to this RFI to confirm your company's interest in submitting a proposal as part of the RFP process, or in general.

1.1 Background

The Allotrope Data Format (ADF) is a vendor, platform, and method-agnostic format adaptable to store the data, contextual metadata, and any ancillary files. Advanced semantic technology is used as the basis for the contextual metadata layer within the Allotrope Data Format. It is integrated within HDF5 file format for fast I/O processing and storage.

The ADF class libraries and APIs provide a set of reusable software components to create new solutions or adapt existing applications to work with the ADF file. The ADF library ensures the consistent adoption of the standard ADF framework.

Currently used for laboratory techniques in pharma, Allotrope members utilize the ADF software library and file format to manage, analyze, process, and store laboratory heterogeneous data. Contextual metadata is used to drive laboratory automation that fuels downstream analytics.

For technical information on the ADF software library and framework specifications please refer to the following documentations:

- The ADF library specifications can be found at: http://docs.allotrope.org/
- The latest API library documentation (Javadoc) can be found at*: https://community.allotrope.org/resources/reference/adf/api/

  *Please note: The API library documentation is only available to the AF community members. The Allotrope product team is open to conversation with non-Allotrope community parties.

In general, SDK libraries are a great tool to streamline and simplify the way software and application developers interact with products as they remove the complexity of interacting with APIs and abstracts them from implementation details.

The current ADF library is written in Java while there is also a transpiled version in C#. While Java is the language of choice for many enterprise applications with a large supporting community, it is being overshadowed within the data science community by other languages with growing momentum and support. AF is considering refactoring its current ADF library written in Java to a thin, efficient, portable, and easily wrappable middleware written in C++ (with possible C API).

1.2 Short-term Refactoring Goals

Refactoring of the ADF library software stack has the following immediate short-term goals:

1. **Adoption**: Addressing different user groups and applications:
   a. Scientific computing community (Python, R)
   b. Enterprise application and instrumentation integration (Java, C#, C++)
2. **Domain Access:** Immediate access to the ADF files for early experimentation and assessment by analytics and data science applications
3. **Positioning & Growth**: Better positioned to integrate into existing and future data environments

Short-term alternatives and practical approaches that address the short-term goals are encouraged to be introduced by the respondents. Please refer to the section: Other optional solutions

1.3 Long-term Refactoring Goals

Refactoring of the ADF library software stack has the following long-term general goals:

1. **Enablement**: Simplify the path to additional language(s) of choice of the data science community to align with their existing development tools and applications. Therefore, the following language wrappers are needed:
   a. Python wrapper
   b. R wrapper

2. **Development efficiency**: Writing many libraries in their native syntax can be costly, and it slows down development, prototyping, and experimentation
3. **Multiple language support**: C/C++ is more easily wrapped (e.g. a Python wrapper would not need to first marshal data through Java).
4. **Performance**: High performance codes are often written as C/C++ libraries
5. **Portability**: C/C++ compilers available almost anywhere and have fewer deployment issues.
6. **Tighter integration with HDF5**: ADF underlying file encapsulation is HDF5. Refactoring can make more direct use of the HDF5 C library and improve performance.
7. **Compatibility**: API compatibility with the existing ADF library implementation to support easy migration of the refactored library to work with existing applications. Therefore, the following language compatibility is needed:
    a. Java wrapper
    b. C#/.NET wrapper

A long-term refactoring solution can be introduced and presented by the respondents later in this document. Please refer to the section: Long-term Refactoring Solution

1.4 Intermediate Short-term Python Solution Objectives

Both Java and Python stand out as widely spread programming languages and have been deployed in a variety of applications and systems. Python stands out to be a very concise, easy to use and read with an extensive set of tools for scientific computing. Developers have wanted to get the two languages to integrate cleanly and developed open source tools to enable it. An intermediate solution that utilizes those tools can be considered together with the following objectives in mind:

1. Quick proof of concept
2. Shorter time for project completion
3. Enabling the AF community to start with early experimentation and assessment of the ADF library and file format for analytics and data science applications.
4. Evaluation of Python APIs to address the need for "Ease of Use" of the ADF file
5. Early comparison with the HDF5 for Python library – h5py
6. Maintaining the Java code as the baseline in the short term.
7. Optionally use existing off-the-shelf open source tools
8. Profiled as a software integration project and less as a software development project
9. Minimizing the initial project resources, cost, and risk

A secondary objective and possible outcome of the intermediate Python solution will be an evaluation and possible development of requirements for the ADF library APIs as well as the High-Level APIs in Python prior to a full long-term refactoring project.

Some of the intermediate short-term Python alternatives can be introduced and presented by the respondents later in this document. Please refer to the section: Intermediate Short-term Python Alternatives

2 Additional Refactoring Options

2.1 Cross Platform Solution

Some other alternatives such as cross platform solutions can be introduced by the respondents. They are presented later in this document. Please refer to the section: Optional Cross Platform Solutions

2.2 R Specific Solution

Integrating R with Java could create some high-end analytics-based applications. There are several possible ways to connect R and Java using off-the-shelf packages. Some of the R integration alternatives can be introduced by the respondents, few of them are presented later in this document. Please refer to the section: Optional R Specific Solutions

2.3 Other Options

The RFI is open to other ideas and technology solutions that address the problem statement above. Some of the factors to be considered may include:

- Proof of concept
- Low cost alternatives
- Shorter time for project completion
- Integration of existing off-the-shelf solutions

Other alternatives can be introduced and presented by the respondents later in this document. Please refer to the section: Other optional solutions

3 Response Guidelines and Timeline

3.1 Timeline

#	Action	Date*	Remarks
	RFI release date	June 8, 2020	
	RFI Q & A	Between the release date to submission date	AF Product Team and AF Secretariat are open to informal or formal discussions from the release date until the submission date as needed. Contact information is available later in this section
	"Intent to respond" notification	June 19, 2020	AF Product Team is requesting for a notification about the intention to respond

			to the RFI from the respondents.
	RFI submission due date	June 30, 2020	All responses to this RFI are due via email no later than 5pm EDT of the due date. Contact information is available later in this section

*All timelines and anticipated dates and are subject to change

3.2 General Guidelines

3.2.1 Notice

Respondents may choose to respond to any of the alternative solutions. Especially they are encouraged to address the shorter-term and Python alternatives.

3.2.2 Notice

While not required, the AF Product Team is requesting for a notification about the intention to respond to the RFI from the respondents. Please send a notification by the date within the timeline table in this section.

3.3 Budget, Resources, and Project Timeline

Each of the solution alternatives must include ballpark estimations of cost in US Dollars, resources, and timeline to completion.

3.4 IP and License Limitations

Each of the solution alternatives must include information regarding any IP, open source, and license limitations.

NOTE: AF will strongly prefer solutions where all (or most) of the solution is either owned by AF or under a permissive (i.e., non-reciprocal open source license) so that AF has flexibility in licensing to the community. For any pre-existing technology that will be licensed to AF, AF will need a broad license to modify, create derivative works and redistribute such technology.

3.5 Eligibility

Any company or organization is eligible to respond to this RFI. However, to be awarded a contract based upon this RFI, the company or organization must be a member of the Allotrope Partner Network (APN) on or before the contract execution date and throughout the project.

3.6 Contact Information

3.6.1 Q&A and Notifications Contact

Allotrope Product Team: amnon.ptashek@allotrope.org

3.6.2 Submission and Notifications Contact

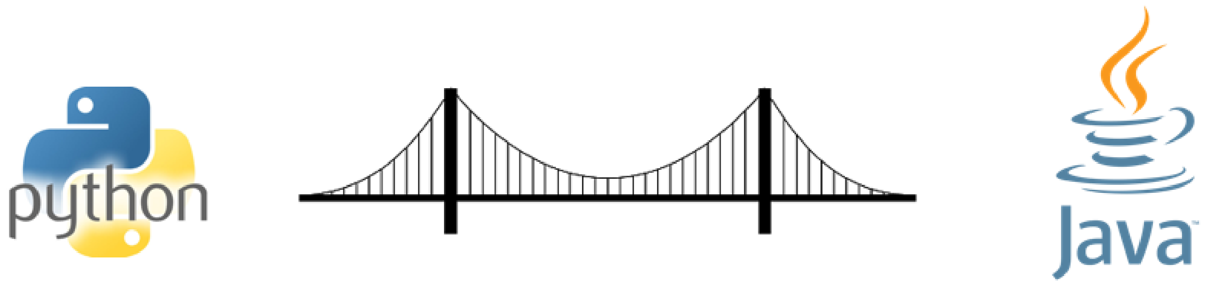
Please submit your response by via email to: amnon.ptashek@allotrope.org

4 Intermediate Short-term Python Alternatives

The following section presents few off-the-shelf Python/Java solutions. AF welcomes other approaches to be introduced by the respondents

4.1 Off-the-shelf Python/Java Solutions

An optional intermediate short-term solution for Python support may be achieved using off-the-shelf Python/Java solution to access the ADF Java library.



The available integration tools between Java and Python approach the integration in different ways: Py4j makes it possible to integrate code in both languages. It allows code written in Python to access Java objects from the JVM (Java Virtual Machine) as if they were both running in the same interpreter. Jython is an implementation of Python written specifically to integrate into the existing Java platform. As opposed to Jython, Py4J does not execute the Python code in the JVM so it does not need to reimplement the Python language and developers can use any libraries supported by their Python interpreter

AnalysisRPC has been integrated into the Eclipse IDE. It is a bridge between the two languages that utilizes different translation layers to make it possible to call methods from Python to Java code as well as vice versa.

Some of the available Python/Java solutions are listed below:

- Py4J: Py4J is a “bridge” between Python and Java. It enables Python programs running in a Python interpreter to dynamically access Java objects in a Java Virtual Machine https://www.py4j.org/

- JCC: JCC is a C++ code generator that produces a C++ object interface wrapping a Java library via Java's Native Interface (JNI). JCC also generates C++ wrappers that conform to Python's C type system making the instances of Java classes directly available to a Python interpreter. https://lucene.apache.org/pylucene/jcc/index.html

- javabridge: running and interacting with the JVM from Python
  https://pythonhosted.org/javabridge/index.html

- Jython: Jython is a Java implementation of Python. Jython is complementary to Java and is especially suited for the following tasks:
  - Embedded scripting - Java programmers can add the Jython libraries to their system to allow end users to write simple or complicated scripts that add functionality to the application.
  - Interactive experimentation - Jython provides an interactive interpreter that can be used to interact with Java packages or with running Java applications. This allows programmers to experiment and debug any Java system using Jython.
  - Rapid application development - Python programs are typically 2-10x shorter than the equivalent Java program. This translates directly to increased programmer productivity. The seamless interaction between Python and Java allows developers to freely mix the two languages both during development and in shipping products.
    https://www.jython.org/

- JPype: JPype is an effort to allow Python programs full access to Java class libraries. This is achieved not through re-implementing Python, as Jython/JPython has done, but rather through interfacing at the native level in both virtual machines.
  https://jpype.readthedocs.io/en/latest/

- AnalysisRPC: AnalysisRPC is a Python-to-Java bridge that provides a generic way to call Python functions from Java as well as Java functions from Python.

4.2 Request for Information

Please provide the information that addresses such an optional intermediate Python/Java solution.

#	Requirement	Details	RFI Response
4.3	**Off-the-shelf Python/Java solutions**	off-the-shelf Python/Java solution to access the ADF Java library. Please provide a detailed design consideration and any (if required) software customization.	
4.4	**Budget, resources, and project timeline**	Please provide ballpark estimations of cost in US Dollars, resources, and timeline to completion.	
4.5	**IP and License limitations**	Please provide the information regarding any IP, open source, and license limitations.	

5 Long-term Refactoring Solution

5.1 C++ Refactoring with Multi Language Wrappers

Refactoring the ADF library with several language wrappers is positioned as the longer-term solution that can address the project refactoring goals.

5.2 Request for Information

Please provide the information that addresses the following design principles and requirements:

Please scroll down to the next page!

#	Requirement	Details	RFI Response
5.3	**Target Users**	The ADF library is created for developers and the scientific computing community. It is required to provide clear and accurate technical documentation, coding standards and best practices, software testing and tools.	
5.4	**Multi language**	Multi language support	
5.4.1		C	
5.4.2		C++	
5.4.3		Java (possible JNI)	
5.4.4		C#/.Net	
5.4.5		Python	
5.4.6		R	
5.4.7		Other: Please specify	
5.5	**Wrappers Development**	Tools for Multi language support (such as SWIG - Simplified Wrapper and Interface Generator)	
5.6	**Performance**	The ADF library may be embedded into other applications that have specific performance requirements. The ADF library must not be a bottleneck. It is essential to ensure that performance is as efficient as possible. The refactored library needs to demonstrate similar or better performance than the current ADF libraries. Microbenchmarks and profilers may be considered. Please specify performance and tuning consideration in the design with respect to: - file read/write - file close - file size - checksum verification - Audit trail enabled - other	
5.7	**Portability**	Cross Platform Support for:	
5.7.1		Windows, MacOS Operating Systems	
5.7.2		Linux Operating System APIs (Debian and Redhat-RHEL)	
5.7.3		RTOS support (POSIX compatibility)	

5.8	**Scalability**	The ADF library needs to be flexible enough to let the applications scale without issue. Well defined contracts with the software components is required to support scaling factors such as the needs to run custom processes, dependency with a cache system or a datastore, or shared resources (such as memory: C/C++ have flexible memory management unlike Java/ C# that have unchangeable built in memory management).	
5.9	**Concurrency**	The ADF library may have dependency with a shared resource or works with external services. High level of concurrency can avoid issues like “race conditions” and bottlenecks result of lock contention in the code. Efficient Multithreading is required to support concurrency. The current ADF libraries have been built with thread-safety enabled	
5.10	**Parallelism**	Different applications may run a different copy of the ADF library program simultaneously but executed on different data. The current ADF library was design for parallel processing.	
5.11	**Exception and error handling**	A robust exception and Error handling strategy is required. It is important to define it early in the development thoroughly. This includes detecting an error, transmitting information about an error to some handler code, preserve the state of a program in a valid state and avoid resource leaks	
5.12	**Best practice**	Best practice tools for:	
5.12.1		Source Code Management: Gitlab	
5.12.2		Continuous Integration/Continuous Deployment: Gitlab	
5.12.3		Artifacts repository: jFrog	
5.12.4		Static code analysis	
5.12.5		Dynamic code analysis	
5.12.6		Test automation	
5.12.7		IDE	
5.12.8		API documentation tools (for example Swagger)	
5.13	**Library deployment, packaging, and distribution**	Proposed software deployment, package management and distribution For example: Package Index (PyPI) for Python	
5.14	**Refactoring lifecycle**	Refactoring steps methodology (for example: small refactoring steps and then retesting/regression testing)	

5.15	**Triple store**	RDF Triple store integration:	
5.15.1		Off-the-shelf vs build	
5.15.2		C/C++ Triple store (such as RDFox) https://en.wikipedia.org/wiki/Comparison_of_triplestores	
5.15.3		Abstraction for interfacing with an external triple store	
5.16	**Version**	Version of C++ standard library	
5.17	**AF High-Level APIs**	Support for the new and under development High-Level APIs that are tailored around use cases and abstract the ADF library APIs. Example for LCUV read APIs*: https://gitlab.com/allotrope-open-source/high-level-lcuv-api *Please note: Currently the access to documentation of the High-Level APIs is only available to the AF community members. The Allotrope product team is open to conversation with non-Allotrope community parties	
5.18	**Budget, resources, and project timeline**	Please provide ballpark estimations of cost in US Dollars, resources, and timeline to completion.	
5.19	**IP and License limitations**	Please provide the information regarding any IP, open source, and license limitations.	

6 Optional Cross Platform Solutions

6.1 Cross Platform Solutions

Other ways for addressing the problem are cross languages/cross platform services. Some of them are listed below:

- The Apache Thrift software framework, for scalable cross-language services development, combines a software stack with a code generation engine to build services that work efficiently and seamlessly between C++, Java, Python, PHP, Ruby, Perl, C#, JavaScript, Node.js, and Delphi and other languages.
  https://thrift.apache.org/

6.2 Request for Information

Please provide the information that addresses such optional cross platform solutions.

#	Requirement	Details	RFI Response
6.3	**Cross platform solutions**	cross languages/cross platform services	
6.4	**Budget, resources, and project timeline**	Please provide ballpark estimations of cost in US Dollars, resources, and timeline to completion.	
6.5	**IP and License limitations**	Please provide the information regarding any IP, open source, and license limitations.	

7 Optional R Specific Solutions

7.1 R Specific Solutions

There are several possible ways to connect R and Java. Some of the off-the-shelf packages to integrate R and Java are listed below:

- rJava: rJava is a low-level interface to Java VM. It allows creation of objects, calling methods and accessing fields.
  https://cran.r-project.org/web/packages/rJava/index.html

- Rserve: Rserve acts as a socket server (TCP/IP or local sockets) which allows binary requests to be sent to R.
  https://cran.r-project.org/web/packages/Rserve/index.html

7.2 Request for Information

Please provide the information that addresses such an R specific solution.

#	Requirement	Details	RFI Response
7.3	**Off-the-shelf R/Java solutions**	off-the-shelf R/Java solution to access the ADF Java library. Please provide a detailed design consideration and any (if required) software customization.	
7.4	**Budget, resources, and project timeline**	Please provide ballpark estimations of cost in US Dollars, resources, and timeline to completion.	
7.5	**IP and License limitations**	Please provide the information regarding any IP, open source, and license limitations.	

8 Other optional solutions

8.1 Optional Request for Information

Please provide the information.

#	Requirement	Details	RFI Response
8.2	**Other solutions**		
8.3	**Budget, resources, and project timeline**	Please provide ballpark estimations of cost in US Dollars, resources, and timeline to completion.	
8.4	**IP and License limitations**	Please provide the information regarding any IP, open source, and license limitations.	

9 Notice

The contents and information provided in this RFI are meant to identify potential solutions and sources that align to the purpose and address the problem statement in this document.   Please note that a successful respondent selected by AF for a project will be required to execute an Agreement that will govern the terms of the project. When responding to this RFI, please note the following:

- This RFI is not an offer or a contract
- Responses submitted in response to this RFI become the property of AF
- Respondents will not be compensated or reimbursed for any costs incurred as part of the RFI process

- If AF receives and responds to questions from RFI respondents, AF reserves the right to anonymize the questions and make the questions and AF's responses available to all respondents via our website
- Responses to RFIs should contain only high-level discussions of product development efforts and should not contain trade secrets or confidential information. AF does not make any confidentiality commitments with respect to RFI submissions but agrees not to publicly distribute RFI responses outside of AF or share RFI responses with other respondents.
- AF is not obligated to contract for any of the products or services described in this RFI
- AF reserves the right to:
    - Accept or reject any or all proposals
    - Waive any anomalies in proposals
    - Negotiate with any or all bidders
    - Modify or cancel this RFI at any time