Secure API-Driven Research Automation to Accelerate Scientific Discovery

TYLER J. SKLUZACEK, PAUL BRYANT, A.J. RUCKMAN, DANIEL ROSENDO, SUZANNE PRENTICE, MICHAEL J. BRIM, RYAN ADAMSON, SARP ORAL, MALLIKARJUN SHANKAR, and RAFAEL FERREIRA DA SILVA, Oak Ridge National Lab, USA

The Secure Scientific Service Mesh (S3M) provides API-driven infrastructure to accelerate scientific discovery through automated research workflows. By integrating near real-time streaming capabilities, intelligent workflow orchestration, and fine-grained authorization within a service mesh architecture, S3M transforms programmatic access to high performance computing (HPC) while maintaining uncompromising security. This framework allows intelligent agents and experimental facilities to dynamically provision resources and execute complex workflows, accelerating experimental lifecycles, and unlocking the full potential of AI-augmented autonomous science. S3M represents a paradigm shift in scientific computing infrastructure that eliminates traditional barriers between researchers, computational resources, and experimental facilities.

Additional Key Words and Phrases: Scientific APIs, Autonomous Science, Data Streaming, Workflows

ACM Reference Format:

Tyler J. Skluzacek, Paul Bryant, A.J. Ruckman, Daniel Rosendo, Suzanne Prentice, Michael J. Brim, Ryan Adamson, Sarp Oral, Mallikarjun Shankar, and Rafael Ferreira da Silva. 2025. Secure API-Driven Research Automation to Accelerate Scientific Discovery. In *Practice and Experience in Advanced Research Computing (PEARC '25), July 20–24, 2025, Columbus, OH, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Modern scientific research increasingly demands seamless integration between experimental and computational facilities, their computing resources, and data management systems to enable autonomous discovery. The emerging paradigm of "self-driving autonomous laboratories" requires programmatic research interfaces that can coordinate complex workflows that span multiple facilities without (or with minimum) human intervention [8]. Researchers have traditionally relied on manual methods—logging into compute clusters via SSH, submitting batch jobs, and asynchronously retrieving data, but these approaches fundamentally limit the potential for near-real-time experiment steering, active data analysis, and on-demand resource allocation; functionalities needed for next-generation science.

The rise of generative AI models and reinforcement learning agents capable of scientific reasoning has created new imperatives for the experimental infrastructure. To create truly autonomous AI-based experimentation, researchers must programmatically integrate their research capital—instruments, compute resources, and data repositories—with their AI training, testing, and inference pipelines [4, 16]. Implementing such interfaces poses not only technical challenges but also critical security and policy concerns [7]. Although HPC facilities employ strict authentication frameworks to

Authors' Contact Information: Tyler J. Skluzacek; Paul Bryant; A.J. Ruckman; Daniel Rosendo; Suzanne Prentice; Michael J. Brim; Ryan Adamson; Sarp Oral; Mallikarjun Shankar; Rafael Ferreira da Silva, Oak Ridge National Lab, Oak Ridge, TN, USA.

Manuscript submitted to ACM

protect resources and data, these same protections create barriers for automated systems, laboratory instruments, edge devices, and AI agents that need to trigger computations in response to experimental results or predictive insights.

The absence of standardized, secure mechanisms to orchestrate workflows between experimental and computational facilities results in fragmented solutions, communication inefficiencies, and missed opportunities for AI-accelerated scientific discovery. Oak Ridge Leadership Computing Facility's (OLCF) Secure Scientific Service Mesh (S3M) addresses these challenges by providing a facility API—the first of its kind to leverage a flexible service mesh architecture—that enables authenticated external systems and intelligent agents to securely provision resources, stream data, and trigger compute jobs dynamically. This architecture ensures modularity, scalability, and policy-driven security enforcement across computational services. In this paper, we present our work-in-progress architecture, API components, security model, and user interfaces of S3M, demonstrating how this infrastructure enables a new generation of autonomous scientific workflows at OLCF.

Concise Perspective on Related Scientific APIs. S3M extends previous work on scientific APIs. The Superfacility API [6] provides RESTful interfaces to HPC resources, allowing experiments to transfer data to compute facilities and trigger analysis jobs. FirecREST [5] offers a RESTful web API infrastructure that connects scientific gateways to HPC systems. Globus Flows [4] provides automation of the research process through the cloud-hosted execution of flows on heterogeneous resources. Tapis [15] is a platform for distributed computational research that offers fine-grained authorization, data management, and code execution capabilities. SCEAPI [13] provides a unified RESTful API for accessing HPC resources in multiple Chinese supercomputing centers, supporting authentication, file transfer, and job management. S3M distinguishes itself through a service mesh architecture, allowing highly customizable services, fine-grained policy enforcement, and dynamic routing, capabilities not possible in traditional API gateways. It introduces advanced streaming for low-latency data exchange enabling near real-time decision making, seamless workflow orchestration, and support for custom API extensions. These features are invaluable for high-security experimental and computational facilities that need to quickly process and act on incoming data streams.

2 The Scientific Service Mesh for Automated Science

The Secure Scientific Service Mesh (S3M) provides programmatic access to OLCF's HPC resources by integrating distinct services, managed policies, and fine-grained authorization into a unified framework. Built on a flexible service mesh architecture using OpenShift, deployed in OLCF's Slate clusters [11], S3M enables scientific instruments, workflows,

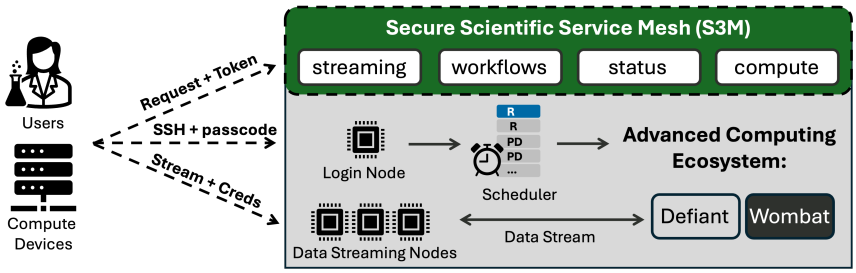

Fig. 1. S3M Architecture diagram. Automated and human clients can make requests to S3M if they have an S3M Access Token. Authorized users have access to the various APIs hosted in front of OLCF resources. S3M then handles the provisioning of streaming nodes and communication with the compute resources' schedulers.

and intelligent agents to interact securely with HPC systems while maintaining strong security through layered policy-as-code access controls and project-scoped authentication. A *service mesh* is an infrastructure layer that facilitates secure and efficient communication between services in a distributed system, abstracting networking complexity while enforcing authentication, authorization, and traffic management policies [10]. In S3M, this architecture ensures modularity, scalability, and security, allowing independent management of core services, including advanced streaming for near real-time data exchange, workflow orchestration across facility boundaries, status monitoring, and compute scheduling. This design, visualized in Fig. 1, allows new capabilities to be explored without disrupting existing OLCF infrastructure, which is particularly valuable for AI-driven experimental feedback loops that must operate within the OLCF's HPC environment.

S3M relies on Istio [1], an open-source service mesh platform that provides fine-grained traffic management, security, and observability features, to enforce multilayered validation through authentication, authorization, and policy compliance checks. The traceability features in Istio offer a comprehensive view of all requests and behavior within the mesh, supporting compliance and security auditing. Users obtain project-scoped authentication tokens with strictly defined permissions, and S3M validates every request against project allocations and resource policies before execution. This approach enables dynamic resource provisioning and workflow automation, while preserving the integrity of OLCF's computing environment.

2.1 Secure API Communications Framework

S3M provides an extensive set of APIs, accessible at both gRPC+Protobuf [2, 9] and RESTful JSON endpoints, that enable researchers to interact with OLCF resources programmatically. Each endpoint serves a specific purpose within the scientific workflow automation ecosystem, from monitoring resource availability to submitting large and complex compute tasks. The core API components, summarized in Table 1, are designed to support diverse scientific needs while maintaining strict access controls and tightly integrating a wide range of access policies. In the following, we describe in more detail two of S3M's unique APIs: Streaming and Workflows.

API Endpoint	Description
/status	Provides resource availability information, including overall system status, specific resource states, and scheduled downtimes.
/compute	Supports job submission and management, allowing users to submit, track, and cancel compute jobs on available resources.
/streaming	Manages data streaming resources, enabling provisioning, listing, and deallocation of Redis or RabbitMQ instances for low-latency scientific workflows.
/environment	Retrieves dependency and runtime environment information for computing workflows.
/tokens	Manages secure API access tokens for authenticated service interactions.
/workflows	Facilitates workflow automation by allowing submission, status retrieval, and cancellation of complex workflows across heterogeneous computing resources.

Table 1. Core S3M API Endpoints and Their Functionality.

The ***Streaming API*** is one of S3M's most transformative capabilities [3]. While many facility APIs provide well-defined interfaces for accessing individual resources, they typically lack capabilities to connect compute jobs with experiment control applications. As a result, researchers must manually deploy and maintain external messaging services to facilitate data exchange. The Streaming API enables researchers to provision RabbitMQ or Redis messaging services on dedicated high-throughput streaming nodes near computational resources through simple API calls. This feature is crucial for interactive science applications that require low-latency data exchange between experimental facilities and compute resources, enabling near real-time decision-making and instrument feedback loops. By abstracting

the complexity of message broker management behind a unified interface, the Streaming API simplifies the development of these data-intensive scientific workflows.

Previously, OLCF users needing live interactions with compute jobs faced a tedious, multistep process: (1) securing an allocation on one of our Kubernetes application clusters or finding their own hardware to host a message broker; (2) installing and configuring their broker; (3) requesting firewall exceptions to enable communication between instruments and facilities; and (4) maintaining the health and security of their broker. Furthermore, externally hosted brokers were often physically distant from control applications or compute resources, leading to increased latency and inconsistent throughput. The Streaming API eliminates these challenges by automating broker provisioning in secure, OLCF-approved environments that are proximal to both OpenShift application clusters and computational resources. Fig. 2 illustrates the interaction flow between the researcher, the S3M Streaming and Compute APIs, and the underlying infrastructure that enables these capabilities.

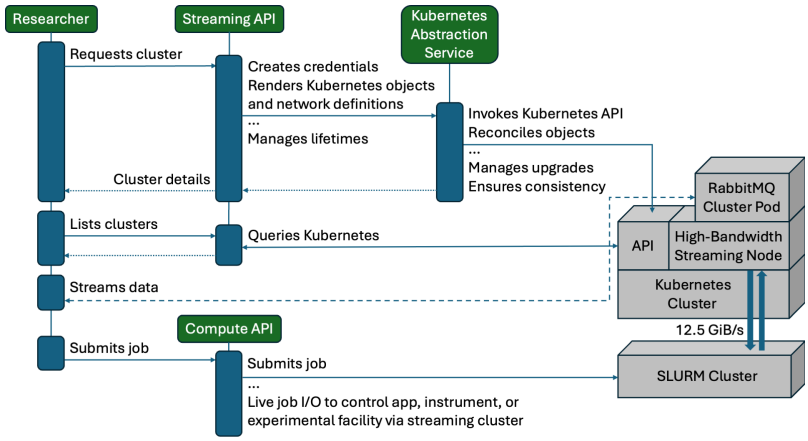

Fig. 2. S3M Streaming Service Interaction Flow. This diagram shows how researchers interact with S3M services to provision streaming infrastructure and interface with a compute cluster. The Stream Manager creates Kubernetes objects while the Abstraction Service deploys the RabbitMQ cluster. Researchers can then develop data pipelines with these resources.

The ***Workflow API*** extends S3M's capabilities by integrating with Argo Workflows [12], enabling researchers to orchestrate complex, multistep scientific processes with minimal manual intervention. By supporting the submission of Argo Workflow Templates, this API allows users to define, reuse, and share sophisticated execution pipelines that seamlessly incorporate custom endpoints and data management tools while handling parallel workflow invocations; data dependencies and artifact management; and fault tolerance. This abstraction layer significantly reduces the developer error and execution overhead, allowing scientists to focus on research objectives rather than managing computational logistics. The Workflow API represents a critical component for autonomous science, where reproducible and efficient processing chains across distributed resources are essential to discovery.

2.2 Layered Authentication and Authorization Framework

S3M enforces a multi-tiered security architecture to validate all remote client interactions with OLCF resources. Each API request must include an authorization token generated by a valid user through our trusted web portal. This ensures that only users with appropriate project access and a sufficient account status can generate credentials. All network traffic is encrypted at the gateway, mitigating risks from credential interception or adversary-in-the-middle attacks.

Once a request reaches S3M, the system validates the user's identity, project affiliations, and resource access against the OLCF infrastructure. Requests that fail these checks are immediately rejected, preventing unauthorized access before reaching internal services. If a request passes the authentication and authorization layers, it is routed to the relevant service, such as compute job submissions to Slurm schedulers [14]. To manage security across internal communications, all S3M services communicate over mutual TLS (mTLS), reducing unauthorized access risks within the system. Extensive logging captures details for all requests in addition to internal communication throughout the service mesh, ensuring full traceability for compliance and anomaly detection. This model minimizes implicit trust and strengthens access controls, supporting broader efforts to adopt zero-trust principles across OLCF infrastructure.

3 Programmatic Research Interface

The Software Development Kit (SDK) provides a simple Python interface for both human researchers and automated systems to interact with OLCF. By encapsulating complex API interactions into intuitive service classes, the SDK eliminates common implementation challenges around authentication, request formatting, or error handling, allowing scientists to focus on research objectives rather than infrastructure mechanics. The package is installed via pip and securely manages the authentication token using an environment variable to prevent credential exposure.

The streaming service example in Listing 1 illustrates the dynamic provisioning of a dedicated messaging infrastructure for the exchange of near real-time data between scientific instruments and computational resources. With just a few lines of code, researchers can orchestrate the entire messaging infrastructure lifecycle: from dynamically a fully configured RabbitMQ cluster with precise CPU and memory allocations, to seamlessly transmitting experimental data through established channels, to automatically decommissioning resources upon completion. This capability is particularly valuable for automated scientific workflows that require low-latency communication for experimental steering and adaptive decision-making based on emerging computational results. Listing 2 illustrates the description of a multitask directed acyclic graph (DAG) workflow in Argo. This workflow reuses predefined templates (e.g., Listing 1) to deploy the streaming service, submit the compute job, and check the job status. Such workflow automation and template reusability help to lower the barrier for reproducing complex experiments.

```python
from olcf_s3m_api.client import OLCFAPIClient
from olcf_s3m_api.streaming import StreamingService

client = OLCFAPIClient(token=os.environ['S3M_TOKEN'])
service = StreamingService(service_name="rabbitmq",
                           api_client=client)

status = service.start_cluster(
    cluster_name = "my-rmq-cluster",
    node_count   = 1,
    cpu_count    = 4,
    ram_gib      = 4
)

# calls to RabbitMQ cluster using Pika library
. . .
service.stop_cluster(cluster_name="man-cluster")
```

Listing 1. Data Streaming Cluster Provisioning and Management Through S3M Streaming Service.

```yaml
kind: Workflow
spec:
  templates:
      dag:
        tasks:
          - name: deploy-streaming-service
            templateRef:
              template: deploy-streaming
          - name: submit-job
            dependencies: [deploy-streaming-service]
            templateRef:
              template: submit-job
          - name: check-job-status
            dependencies: [submit-job]
            templateRef:
              template: check-job-status
            arguments:
              parameters:
                - name: JOB_ID
                  value: "{{tasks.submit-job.outputs.parameters.JOB_ID}}"
```

Listing 2. Argo Workflow Executed Through S3M Workflow API.

4 Future Directions for Scientific Automation

S3M represents a transformative approach to autonomous API-driven scientific workflows by using a service mesh architecture to enable secure and scalable interactions between researchers, AI agents, and HPC resources. By unifying fine-grained authorization, dynamic resource provisioning, and low-latency data streaming under a cohesive framework, S3M establishes the foundation for next-generation scientific automation. Although currently available to internal users on select clusters, our roadmap includes refining authentication policies with input from diverse science projects, integrating advanced workflow management systems, publishing comprehensive SDK documentation, and expanding to external user access. As S3M evolves toward deployment on the OLCF's exascale Frontier supercomputer, this framework will dramatically accelerate experimental lifecycles, enable adaptive research workflows in near real-time, and ultimately transform how AI-augmented science operates by eliminating traditional barriers between instruments, computational resources, and researchers.

Acknowledgments. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. The authors acknowledge the guidance and expertise of Verónica G. Melesse Vergara in these efforts.

References

[1] The Istio Authors. 2025. Istio: Connect, secure, control, and observe services. https://istio.io/.

[2] The Protocol Buffers Authors. 2025. Protocol Buffers: Language-neutral, platform-neutral extensible mechanisms for serializing structured data. https://protobuf.dev. Accessed: March 2025.

[3] Michael J. Brim et al. 2024. *A High-level Design for Bidirectional Data Streaming to High-Performance Computing Systems from External Science Facilities.* Technical Report. Oak Ridge National Laboratory (ORNL), Oak Ridge, TN (United States). doi:10.2172/2338264

[4] Ryan Chard et al. 2023. Globus automation services: Research process automation across the space–time continuum. *Future Generation Computer Systems* 142 (2023).

[5] Felipe A Cruz et al. 2020. FirecREST: a RESTful API to HPC systems. In *2020 IEEE/ACM International Workshop on Interoperability of Supercomputing and Cloud Technologies (SuperCompCloud).*

[6] Bjoern Enders et al. 2020. Cross-facility science with the superfacility project at lbnl. In *2020 IEEE/ACM 2nd Annual Workshop on Extreme-scale Experiment-in-the-Loop Computing (XLOOP).*

[7] Brian Etz et al. 2025. Enabling Seamless Transitions from Experimental to Production HPC for Interactive Workflows. In *Fifth Combined Workshop on Interactive and Urgent High-Performance Computing (WIUHPC).*

[8] Rafael Ferreira da Silva et al. 2024. *Shaping the Future of Self-Driving Autonomous Laboratories Workshop.* Technical Report ORNL/TM-2024/3714. Oak Ridge National Laboratory. doi:10.5281/zenodo.14430233

[9] The gRPC Authors. 2025. gRPC: A high performance, open source universal RPC framework. https://grpc.io/.

[10] Wubin Li et al. 2019. Service Mesh: Challenges, State of the Art, and Future Research Opportunities. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE).* 122–1225. doi:10.1109/SOSE.2019.00026

[11] Oak Ridge Leadership Computing Facility. 2025. Slate: Container Orchestration at OLCF. https://docs.olcf.ornl.gov/services_and_applications/slate/overview.html. Accessed: 2025-03-21.

[12] Argo Project. 2024. Argo Workflows - The Workflow Engine for Kubernetes. https://argoproj.github.io/argo-workflows/.

[13] Cao Rongqiang et al. 2017. Sceapi: A unified restful web api for high-performance computing. In *Journal of Physics: Conference Series*, Vol. 898. IOP Publishing, 092022.

[14] SchedMD. 2024. SchedMD: Slurm Workload Manager. https://www.schedmd.com/ Accessed: 2024-03-20.

[15] Joe Stubbs et al. 2021. Tapis: An API platform for reproducible, distributed computational research. In *Advances in Information and Communication: 2021 Future of Information and Communication Conference (FICC), Volume 1.*

[16] Akshay Subramanian et al. 2024. Closing the execution gap in generative AI for chemicals and materials: freeways or safeguards. (2024).