# Globus Platform Services for Data Publication

**Conference Paper** · July 2018

DOI: 10.1145/3219104.3219127



**9 authors**, including:

Rachana Ananthakrishnan
University of Chicago
**41** PUBLICATIONS **907** CITATIONS
SEE PROFILE

Ben Blaiszik
University of Chicago
**49** PUBLICATIONS **2,241** CITATIONS
SEE PROFILE

Kyle Chard
University of Chicago
**122** PUBLICATIONS **1,504** CITATIONS
SEE PROFILE

Ryan Chard
Argonne National Laboratory
**35** PUBLICATIONS **312** CITATIONS
SEE PROFILE

# Globus Platform Services for Data Publication

Rachana Ananthakrishnan
Computation Institute, University of Chicago and Argonne National Laboratory
Chicago, Illinois
rachana@globus.org

Ben Blaiszik
Computation Institute, University of Chicago and Argonne National Laboratory
Chicago, Illinois
blaiszik@uchicago.edu

Kyle Chard
Computation Institute, University of Chicago and Argonne National Laboratory
Chicago, Illinois
chard@uchicago.edu

Ryan Chard
Argonne National Laboratory
Lemont, Illinois
rchard@anl.gov

Brendan McCollam
Computation Institute, University of Chicago and Argonne National Laboratory
Chicago, Illinois
bjmc@globus.org

Jim Pruyne
Computation Institute, University of Chicago and Argonne National Laboratory
Chicago, Illinois
pruyne@globus.org

Stephen Rosen
Computation Institute, University of Chicago and Argonne National Laboratory
Chicago, Illinois
sirosen@globus.org

Steven Tuecke
Computation Institute, University of Chicago and Argonne National Laboratory
Chicago, Illinois
tuecke@globus.org

Ian Foster
Computation Institute, University of Chicago and Argonne National Laboratory
Chicago, Illinois
foster@anl.gov

## ABSTRACT

Data publication systems are typically tailored to the requirements and processes of a specific domain, collaboration, and/or use case. We propose here an alternative approach to engineering such systems, based on customizable compositions of simple, independent platform services, each of which provides a distinct function such as identification, metadata association, and discovery. We argue that this approach can reduce costs and increase flexibility and overall service quality. We describe a collection of such services that we are developing within Globus, which initially provide persistent identifier association, data management, and discovery capabilities; we are also working towards an automation service that can reliably and flexibly coordinate these and other services to satisfy varied user needs. We describe data publication use cases that motivate our design, present our vision for a data publication platform, and report on current implementation status.

## CCS CONCEPTS

• **Information systems → Data management systems**; • **Applied computing**; • **Computer systems organization → Cloud computing**;

## KEYWORDS

Globus, data publication, Platform-as-a-Service

**ACM Reference Format:**
Rachana Ananthakrishnan, Ben Blaiszik, Kyle Chard, Ryan Chard, Brendan McCollam, Jim Pruyne, Stephen Rosen, Steven Tuecke, and Ian Foster. 2018. Globus Platform Services for Data Publication. In *Proceedings of Practice and Experience in Advanced Research Computing (PEARC '18).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3219104.3219127


*PEARC '18, July 22–26, 2018, Pittsburgh, PA, USA*

ACM ISBN 978-1-4503-6446-1/18/07…$15.00
https://doi.org/10.1145/3219104.3219127

## 1 INTRODUCTION

The shift towards data-driven and computational research methodologies has, appropriately, been accompanied by a call for increased disclosure and publication of the data used in generating scientific results [25]. In turn, software platforms and online services have been developed to aid in the publication of data. These include standalone, user-installed data publication software (e.g., DSpace [22], Dataverse [15], Fedora [4]), and cloud-hosted, Software-as-a-Service (SaaS) offerings (e.g., Globus Data Publication [12], DuraCloud [2], figshare [5]). However, these systems are implemented as monoliths that encapsulate various capabilities (e.g., data management and metadata association) and are designed to support a specific publication use case and/or community. Thus, it is difficult to adapt these systems to address varied data publication needs.

Over the past two years, we have operated a SaaS data publication system [12]. This service has been used by almost 1800 users to create and access more than 600 datasets spanning disciplines such as biology, climate science, and materials science. We have observed that this service is well-suited to particular publication use cases. For example, when researchers have well-formed datasets with standard metadata and a need for citable persistent identifiers such as Digital Object Identifiers (DOIs) [19]. However, it is less suitable for situations in which researchers need to create identifiers for

active data, to publish datasets which continue to grow and evolve (as is common, for example, in climate science), to integrate their own services in the publication pipeline, or conversely, to integrate publication pipelines into their own applications.

We argue that a Platform-as-a-Service (PaaS) approach, comprising a suite of independent services, provides a more flexible model on which to deliver robust data publication capabilities to users. Each platform service provides a distinct function such as associating a persistent identifier, managing data access, and indexing metadata for discovery. We postulate that this approach can reduce costs and increase flexibility, reliability, and overall service quality. As a platform, these services support easy use, integration, extension, and automation via REST APIs, client Software Development Kits (SDKs), and command line tools. As a suite of services, rather than a single monolithic system, they may be combined in arbitrary ways to match the diverse use cases and varied phases of data publication.

We describe the architecture of the Globus data publication platform and report on the development status of several crucial services: data management, persistent identification, and data discovery. We describe our plans to develop a new automation service for combining these services into flexible, repeatable, user-defined publication pipelines, and suggest several other services that would be valuable in such a platform. We motivate our approach by presenting data publication use cases and highlight how the platform can enable these scenarios. Our aim in presenting this approach is to solicit community feedback into the design of this platform and to identify other services that are needed to address evolving data publication scenarios.

## 2 MOTIVATING USE CASES

We present here two different data publication use cases that we have encountered while operating a data publication service and working with various groups.

### 2.1 Citable data publication

Perhaps the most fundamental interpretation of data publication is to prepare data such that they can be properly cited in scholarly works. Requirements for this type of publication typically include storing the data in a durable online repository from which it can be retrieved and verified by appropriate parties, generation of a persistent identifier via which it can be uniquely referenced, and association of metadata which is consistent with the domain and which facilitates discovery of the results. Such publication capabilities are often provided by institutions or communities.

*2.1.1 Institution data publication.* Institutions, and in particular academic libraries and research computing centers, are often tasked with providing data publication capabilities for their researchers, staff, and students. While libraries have a long history of providing institutional repositories for documents, the growing need to support analogous capabilities for data has lead to the development and deployment of many data publication systems (e.g., Purdue University Research Repository [6]). Institution focused data publication systems must support a diverse set of researchers from many domains. They often require custom metadata schemas (e.g., related to the institution as well as the research domain), local storage on institutional resources, preservation archives, and mandate curation by library staff.

*2.1.2 Community data publication.* Community data publication systems focus on addressing the needs of a specific scientific community. As such, their processes, metadata models, and identifier schemes are aligned with those used in that community. The Materials Data Facility (MDF) [8] is one such example that services the materials science community. MDF provides capabilities to publish large, remotely stored, data; index fine-grained metadata in a common schema to enable discovery and mixing; and facilitate access to these data for dissemination. MDF also integrates with external community repositories both to make externally published data available in MDF and also to disseminate aggregate data to external repositories.

### 2.2 Active data management and publication

The research lifecycle includes many activities that may be considered forms of publication. For example, during the "active" phase of research in which data are being created and explored, data are often shared with colleagues for review, or used as input to an analysis pipeline to generate new derived data. It is beneficial if such actions are performed using unambiguous names, with associated metadata, and even registered in a catalog for future management and discovery. Further, to aid reproducibility it is important that derived data can be traced back to the raw and intermediary data from which it was created.

In prior work, we have developed services to support naming and cataloging of active research data [10, 26]. These services allow users to associate a minid [10] (a lightweight persistent identifier) with a file, irrespective of the mutability or location of that file. They may then register files in a multi-user catalog and associate user-defined metadata with their files. Using these services researchers can reference files using standard names (rather than potentially volatile URLs), organize their data independently from file system location, collaborate with others by sharing names and metadata, and discover when, where and by whom a file was created.

## 3 DATA PUBLICATION PLATFORM SERVICES

Based on the needs of the use cases described above we propose a data publication platform initially comprised of three core services: **`Transfer`** facilitates data management, secure data access, and reliable data transfer; **`Identifier`** provides persistent, verifiable identifiers; and **`Search`** enables scalable indexing and search capabilities. These services are linked together using two overarching services: **`Automate`** coordinates the execution of user-defined data transformation and publication pipelines; and **`Auth`** manages authentication and authorization across all of the component services. Figure 1 illustrates how this platform can be used to implement a common publication pipeline. We show an additional service (**`Describe`**) that demonstrates how user input could, in the future, be added to the platform.

### 3.1 Data management

Our use cases highlight several needs of the data management component of the publication platform including: a uniform access

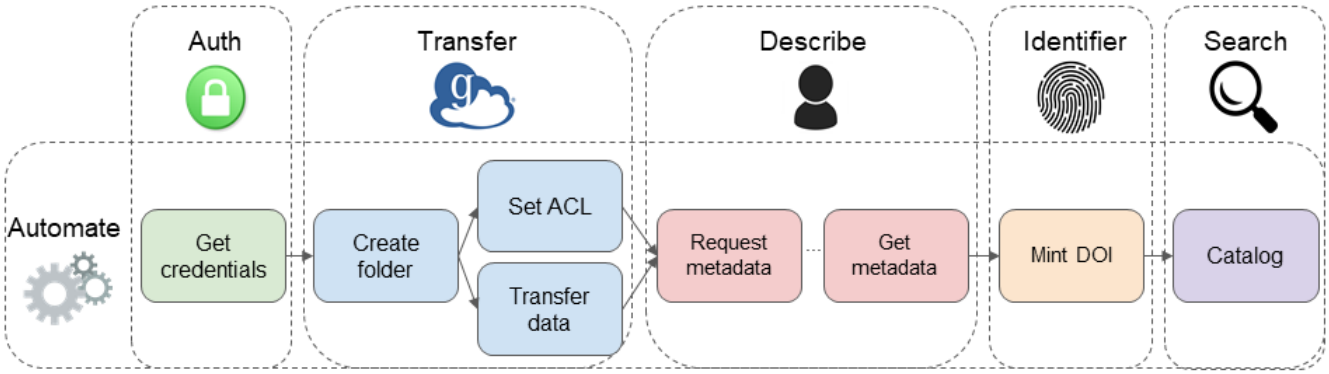


**Figure 1: An implementation of a publication pipeline in which data are assembled and described, associated with a persistent identifier, and made discoverable in a catalog, showing the steps involved and the Globus services used to perform those steps.**

interface, support for managing distributed data, efficient transfer between users and managed storage, in-place publication of data, flexible access control to implement various access policies (e.g., embargo, groups, audit, etc.), and many more.

We use Globus **`Transfer`** [13] to satisfy these needs. This service supports the management, synchronization, transfer, and sharing of data on arbitrary storage systems on which the Globus Connect agent is deployed. Globus Connect is a software component that provides a uniform access interface to myriad storage repositories, ranging from POSIX file systems to cloud storage. **`Transfer`** supports not only high performance data movement between storage endpoints but also the ability to *share* data in-place. In this mode, data owners may associate access control lists (ACLs), enumerating users and groups and their access rights, to files and folders. **`Transfer`** enforces these ACLs, thereby allowing controlled access to data without provisioning user accounts on each endpoint. The **`Transfer`** REST API provides access to these capabilities, allowing publication pipelines to manage remote storage endpoints (e.g., creating directories, moving files, etc.), transfer data between storage locations (e.g., from remote storage systems to durable publication storage), and manage ACLs on files and folders dynamically (e.g., to implement access policies).

## 3.2 Identifier management

When working with research data throughout its lifecycle there is a need to provide an unambiguous name rather than a potentially volatile reference to a storage location. By enabling such naming, data can be uniquely referenced and located, and they can be easily shared. Globus **`Identifier`** addresses this need by allowing users to create and associate a unique persistent *identifier* with data.

**`Identifier`** exposes a REST API for Create, Update, and Delete (CRUD) operations on identifiers and *namespaces*. Namespaces abstract use of an external *persistent identifier* (PID) provider and a valid account (or shoulder) within that provider. Identifiers minted within a namespace use the external PID provider to create a persistent identifier. Namespaces also define a collection of policies regarding their use, such as which users (or groups of users) are able to create identifiers and what visibility policies are imposed on identifiers minted in the namespace. **`Identifier`** currently supports two external PID types: Digital Object Identifiers (DOIs) and Archival Resource Keys (ARKs).

When minting an identifier in a namespace the following information must be provided: 1) one or more *locations* to access the data such as a URL representing a particular path on a Globus endpoint; 2) metadata describing a mixture of publication-specific attributes (e.g., creator, checksum) and optionally extensible, user-defined attributes; 3) access policies if different than those defined by the namespace. **`Identifier`** stores the minted identifier and provides a REST interface to retrieve information about an identifier and also, optionally, an HTML landing page with human-consumable information about the identifier.

## 3.3 Indexing and search

While identifiers provide unique and durable references for datasets, they do not address discovery. Globus **`Search`** addresses this need. **`Search`** provides a schemaless model for indexing metadata and a flexible free-text query model for discovery. **`Search`** is implemented as a multi-user service, in which user-managed *indexes* are populated with metadata. Indexes are defined with various policies including which users or groups can manage the index, ingest metadata, and query the index. **`Search`** implements two primary APIs: one for adding entries to an index and a second for querying an index.

The first **`Search`** API supports scalable indexing of arbitrary *entries* into a selected index. An *entry* is comprised of three types of information: 1) a subject, which represents a name or target for the entry (e.g., a URL for a file or an identifier); 2) arbitrary metadata represented as a collection of attributes in a nested JSON document; and 3) a visibility policy that defines which users or groups are able to view and query the subject and its metadata.

Multiple entries in an index may refer to the same subject with each entry containing separate metadata and visibility policy. This allows, for example, different sets of metadata to be made visible to different users. Perhaps some metadata is visible to all users (`public`) and some is restricted to collaborators. Multiple users may provide entries related to the same subject without concern

for collision. In query results, all visible entries are merged into a single subject so users need not iterate through results to combine the various entries for the subject.

The second **Search** API supports two types of queries against an index: simple and complex. Simple queries perform basic substring matching against any metadata fields that are visible to the querying user. As with web search, the results of a simple query are ordered based on the computed "best match" for the query. Complex queries, in contrast, take the form of a structured JSON document, and are more commonly used when queries are created programmatically. They may reference specific metadata fields, and may apply criteria such as value ranges, wildcards, and regular expressions among others. Complex queries may also specify facets to generate categories and associated frequencies for particular metadata fields. Both forms of query provide other basic mechanisms, such as pagination, to scroll through large numbers of results.

## 3.4 Publication pipeline automation

Each of the services described above address important data publication requirements. However, it is the ability to securely and reliably integrate these, and other, services that will make the publication platform uniquely useful. To this end, we are developing Globus **Automate**: a service designed to enable the composition and execution of arbitrary data publication pipelines.

In our prototype we leverage a simple registration model, in which each service is registered with a URL and a description of its input/output data. The service must implement an **Automate** API that exposes the action to be executed and an endpoint to query status. **Automate** supports both "synchronous" and "asynchronous" tasks. In the case of an asynchronous task, the task invocation returns an identifier that can be later used to check the status of the task. This allows for potentially long-running tasks such as data transfers or human-in-the-loop actions to be executed.

An **Automate** pipeline is defined in a JSON document which enumerates the services to be used, their execution order and dependencies, and the input data needed for each stage. We build on the Amazon States Language [1] in our prototype. An example pipeline definition is presented in Listing 1. This publication pipeline moves data to a specific publication endpoint, associates an identifier with that data, and ingests metadata into a search index. Each of the named tasks in the pipeline is pre-registered with **Automate**. This allows **Automate** to manage the execution of that task via the registered service API. The pipeline includes a description of the input arguments needed for execution, such as input data, identifier namespace, and search index (not shown in the listing). This information is stored in the pipeline's "state" as a JSON object. The *InputPath* and *ResultPath* attributes in the description define what pipeline state is passed to and from each task. In this case $ represents that all state is passed to each task. Thus, the pipeline state will grow or change during execution. Pipelines are executed via the **Automate** REST API. Once a pipeline is defined, it is made immutable, such that the set of services and their order cannot be changed. However, the pipeline can be copied and modified for reuse by others or use in different scenarios.

**Listing 1: An example `Automate` definition in which `Transfer`, `Identifier`, and `Search` are used to publish a dataset. In this example, the entire execution state is passed to each service (denoted by $). The pipeline is instantiated with arguments describing the dataset to be published.**

```
{
  "Name": "Example publication pipeline",
  "StartAt": "transfer-data",
  "Tasks": {
    "transfer-data": {
      "Type": "Transfer",
      "InputPath": "$",
      "ResultPath": "$",
      "Next": "generate-identifer",
      "Data": {"Destination": "Globus Tutorial Endpoint 1"}
      "Retry": [
        {
          "ErrorEquals": ["States.ALL"],
          "IntervalSeconds": 1,
          "MaxAttempts": 3,
          "BackoffRate": 2.0
        }
      ]
    },
    "generate-identifer": {
      "Type": "Identify",
      "InputPath": "$",
      "ResultPath": "$",
      "Next": "ingest-search",
      "Data": {"Type": "DOI"}
      "Retry": [
        {
          "ErrorEquals": ["States.ALL"],
          "IntervalSeconds": 1,
          "MaxAttempts": 3,
          "BackoffRate": 2.0
        }
      ]
    },
    "ingest-search": {
      "Type": "Search",
      "InputPath": "$",
      "ResultPath": "$",
      "End": true,
      "Data": {"Index": "test-index"}
      "Retry": [
        {
          "ErrorEquals": ["States.ALL"],
          "IntervalSeconds": 1,
          "MaxAttempts": 3,
          "BackoffRate": 2.0
        }
      ]
    }
  }
}
```

## 3.5 Authentication and authorization

Integration of multiple, distinct services is hampered by the mutual need for users and services to trust one another. Our platform integrates Globus **Auth** [23] to broker authentication and authorization interactions between users, identity providers, services, and clients.

**Auth** implements an identity brokering model in which it acts as an intermediary between heterogeneous identity providers and subscribing services. In so doing, it allows users to demonstrate ownership of an identity via an authentication flow with a third-party identity provider (e.g., an institution or Google). Subscribing services, such as our data publication platform services, may then use **Auth** APIs to validate presented identities and consistently reference identities across services.

**Auth** also acts as an OAuth 2 authorization server. In this mode it issues *access tokens* to a client after successful authentication and after obtaining authorization to access a service. **Auth** provides a

set of unique "scopes" for each service. Scopes represent permitted actions or access to specific resources. During the authentication flow, users must explicitly consent that a client may perform the actions associated with these scopes. The access tokens may then be used by one service, with consent of the user, to access their resources (e.g., profile attributes) or invoke another service on their behalf.

The data publication platform also relies on a consistent representation of groups. For example, to specify groups of users who are able to manage identifier namespaces or ingest metadata into a search index. For this purpose we use Globus **Groups** [11]. **Groups** implements a user-managed, hierarchical group model. It exposes REST APIs and web interfaces for creating and managing groups, allowing administrators to specify group policies such as membership requirements and visibility.

### 3.6 Human-centric services

The services described so far do not require human input. However, in many publication pipelines there is a need to solicit input from users. While we have not yet developed the following services, we briefly outline three such services that could be offered by our publication platform: **Describe** to manage schemas and enable metadata entry; **Curate** to allow review and approval; and **Notify** to inform users of tasks.

One significant challenge for data publication systems is the need to obtain descriptive metadata about the subject. While automated methods for metadata association show promise, there is no adequate replacement for metadata provided by humans. **Describe** would allow administrators to create and manage metadata schemas and then render these schemas into web forms via which users could enter metadata.

In most publication pipelines there is an entity (e.g., community, project, or institution) to which a published dataset "belongs." It is most often the case that a user, or group of users, is responsible for managing the collection of datasets associated with that entity and validating that the published datasets meet pre-defined policies (e.g., relevant to the collection, suitably described, appropriate access policies). **Curate** would enable human review, and perhaps modification, of any aspect of a publication (e.g., data, metadata, license, etc.). It would allow users to approve or reject a dataset.

When humans take part in a publication pipeline, they must be made aware of the need to complete particular tasks such as curating metadata or approving publications. Thus, a task management service could be used to notify users of waiting tasks and to manage the completion of these tasks. **Notify** would satisfy these needs. It could implement methods to manage task lifecycles, create tasks based on external operations, asynchronously notify users of tasks (e.g., via email), and manage task deadlines.

## 4 REALIZING USE CASES ON THE PLATFORM

In Section 2 we described two data publication use cases. Here, we show how they can be realized with our data publication platform. For each, we show a simplified version of an **Automate** pipeline definition which would implement the use case.

**Inputs**

- Authenticated user identity
- Current location of the dataset to be published

**Steps**

(1) **Transfer** (Synchronous): create a folder for the publication on the durable storage, and make the folder writable by the user initiating the publication. Store the folder location in the pipeline state.
(2) **Transfer** (Asynchronous): copy data from its current location, specified in the input, to the publication location stored in the pipeline state by the previous step.
(3) **Transfer** (Synchronous): make the data location immutable (set ACL to read-only).
(4) **Describe** (Asynchronous): generate a metadata entry form for the user based on a schema referenced in the pipeline definition. This step may optionally use **Notify** to queue the form entry task for the user. Upon completion, store the user entered metadata into the pipeline state.
(5) **Identifier** (Synchronous): create a new identifier, populated with the metadata from the pipeline's state, and minted by the persistent identifier provider specified in the pipeline definition. Store the identifier in the pipeline state.
(6) **Search** (Synchronous): index the metadata and the identifier for discovery of the publication into the index specified in the pipeline definition.

**Outputs**

- Location of the published data
- Persistent identifier for the published data

**Figure 2: High-level description of a publication pipeline designed to produce citable data.**

In Figure 2 we present a baseline data publication pipeline that implements the citable data publication use case described in Section 2.1. This pipeline performs data preparation (steps 1-3), metadata entry (step 4) and identification and indexing (steps 5-6) of the published dataset.

In Figure 3 we present a data publication pipeline for the Materials Data Facility as an example of community data publication. This pipeline extends the citable data publication pipeline in three ways: it uses a community specific data processing code to automatically extract metadata from files rather than relying on user input (step 2); it may optionally publish extracted metadata in community catalogs (step 2); and it requires human curation and approval before it can be published (step 6).

## 5 RELATED WORK

Requirements for data publication have been well studied [12, 14, 25] and as a result many data publication systems have been developed and are currently in use (e.g., Dataverse [15], DuraCloud [2],

**Inputs**

- Authenticated user identity
- Current location of the dataset to be published
- User selected community registries

**Steps**

(1) **`Transfer`** (Asynchronous): Copy data to an intermediary location for processing.
(2) **`Community-specific data processing`** (Asynchronous): Employ a community code to extract materials-specific metadata from the dataset. Initiate registration in the NIST Materials Resource Registry, Citrination, or other community registries. Return extracted metadata and community registry location to the pipeline state.
(3) **`Transfer`** (Asynchronous): Remove data from intermediary location used for processing.
(4) **`Transfer`** (Asynchronous): Copy data from original location to the publication repository
(5) **`Notify`** (Asynchronous): Notify curators of the new dataset.
(6) **`Curate`** (Asynchronous): Wait for curators to review and approve the publication.
(7) **`Identifier`** (Synchronous): create a new identifier, populated with extracted metadata, and minted by the selected persistent identifier provider. Store the identifier in the pipeline state.
(8) **`Search`** (Synchronous): index the metadata and the identifier for discovery of the publication.

**Outputs**

- Location of the published data
- Persistent identifier for the published data

**Figure 3: Materials Data Facility data publication pipeline.**

and figshare [5]). However, due to the significant differences between communities and data publication scenarios (e.g., domain-specific metadata, various standards, data types, and curation requirements) these systems typically lack the generalizability required to accommodate data publication for diverse use cases.

There are several well known persistent identifier schemes including DOIs, ARKs, and Handles. Rather than create yet another identifier scheme, **`Identifier`** instead focuses on ease of use and ease of integration into arbitrary publication pipelines. Thus, our approach is more focused on providing a common interface and management infrastructure for using existing persistent identifier services. Others have also created similar interfaces including Datacite [9] and EZID [21], however, neither at present offer a JSON-based REST API for identifier creation, support a wide range of persistent identifier providers, or integrate with a flexible authentication platform.

Hosted search services are equally common. Many cloud providers offer hosted search services (e.g., Amazon Elasticsearch Service). Elastic, the company behind ElasticSearch, offers a flexible, cloud-hosted search service called Elastic Cloud [3]. **`Search`** differs in that it takes a multi-tenant approach with user-managed indexes protected with access control enforcement. We further provide fine-grained access control on individual entries in an index.

Automation services are most often concerned with the coordinated execution of applications or services. Scientific workflow engines, such as Galaxy [17], Parsl [7], Pegasus [16], Swift [24], and Taverna [18], facilitate the fault-tolerant and reproducible execution of sequences of tools and services. However, while some (e.g., Taverna) support Web service composition, they are designed for dataflow modeling and pipeline execution. **`Automate`**, in contrast, is concerned with orchestrating a mix of software and human-based tasks. In this regard, it is similar to the Business Process Execution Language (BPEL) [20], a language for specifying business processes with web services. However, we focus on providing a lightweight, and less general, JSON-based model for representing pipelines.

## 6 CONCLUSION

The diverse needs for data publication require a more flexible approach than "one-size-fits-all" data publication systems can adequately provide. We proposed a data publication platform comprised of many specialized services to support a diverse range of data publication requirements. Use of the platform will not only satisfy a broad range of data publication use cases but will also reduce costs associated with developing and running monolithic publication stacks.

Our initial offering is focused on insuring that publication use cases already supported by our SaaS data publication service can be supported while allowing for customization and flexibility to address other use cases. We have completed the development of data management, persistent identifier, and discovery services, and are currently prototyping an automation service capable of orchestrating arbitrary pipelines using these services. The platform builds on existing services (i.e., Globus Auth and Groups) to provide seamless authentication and authorization as well as consistent use of identities and groups across services. The platform is designed to simplify not only end-user use cases, but also enable extension, integration in third party applications, and automation via open APIs, SDKs, and command-line and web-based tools.

Going forward, we anticipate extending the platform to introduce new services with increased functionality. In particular, to reach parity with current, monolithic end-user focused offerings we will continue to develop human-centric services and ensure that interacting with these services is seamless when driven by automated pipelines. We welcome and encourage participation from users and communities with specific data publication needs to help define and develop the platform.

## ACKNOWLEDGMENTS

This work was partially supported under financial assistance award 70NANB14H012 from U.S. Department of Commerce, National Institute of Standards and Technology as part of the Center for Hierarchical Material Design (CHiMaD). We thank Compute Canada for contributing to the design, development, and evaluation of the platform.

## REFERENCES

[1] 2018. Amazon States Language. (2018). Retrieved March 22, 2018 from https://states-language.net/spec.html

[2] 2018. DuraCloud. (2018). Retrieved March 22, 2018 from http://duracloud.org/
[3] 2018. Elastic Cloud. (2018). Retrieved March 22, 2018 from https://www.elastic.co/cloud
[4] 2018. Fedora Repository. (2018). Retrieved March 22, 2018 from http://fedorarepository.org/
[5] 2018. figshare. (2018). Retrieved March 22, 2018 from https://figshare.com/
[6] 2018. PURR: Purdue University Research Repository. (2018). Retrieved March 22, 2018 from http://purr.purdue.edu
[7] Yadu Babuji, Alison Brizius, Kyle Chard, Ian Foster, Daniel S. Katz, Michael Wilde, and Justin Wozniak. 2017. Introducing Parsl: A Python Parallel Scripting Library. (Aug. 2017). https://doi.org/10.5281/zenodo.891533
[8] B. Blaiszik, K. Chard, J. Pruyne, R. Ananthakrishnan, S. Tuecke, and I. Foster. 2016. The Materials Data Facility: Data Services to Advance Materials Science Research. *Journal of the Minerals, Metals & Materials Society (JOM)* 68, 8 (2016), 2045–2052. https://doi.org/10.1007/s11837-016-2001-3
[9] J. Brase. 2009. DataCite – A Global Registration Agency for Research Data. In *4th International Conference on Cooperation and Promotion of Information Resources in Science and Technology*. 257–261. https://doi.org/10.1109/COINFO.2009.66
[10] K. Chard, M. D'Arcy, B. Heavner, I. Foster, C. Kesselman, R. Madduri, A. Rodriguez, S. Soiland-Reyes, C. Goble, K. Clark, E. W. Deutsch, I. Dinov, N. Price, and A. Toga. 2016. I'll take that to go: Big data bags and minimal identifiers for exchange of large, complex datasets. In *IEEE International Conference on Big Data (Big Data)*. 319–328. https://doi.org/10.1109/BigData.2016.7840618
[11] K. Chard, M. Lidman, B. McCollam, J. Bryan, R. Ananthakrishnan, S. Tuecke, and I. Foster. 2016. Globus Nexus: A Platform-as-a-Service provider of research identity, profile, and group management. *Future Generation Computer Systems* 56 (2016), 571–583. https://doi.org/10.1016/j.future.2015.09.006
[12] K. Chard, J. Pruyne, B. Blaiszik, R. Ananthakrishnan, S. Tuecke, and I. Foster. 2015. Globus Data Publication as a Service: Lowering Barriers to Reproducible Science. In *11th IEEE International Conference on e-Science*. 401–410. https://doi.org/10.1109/eScience.2015.68
[13] K. Chard, S. Tuecke, and I. Foster. 2014. Efficient and Secure Transfer, Synchronization, and Sharing of Big Data. *IEEE Cloud Computing* 1, 3 (Sept 2014), 46–55. https://doi.org/10.1109/MCC.2014.52
[14] M.J. Costello. 2009. Motivating Online Publication of Data. *BioScience* 59, 5 (2009), 418–427. https://doi.org/10.1525/bio.2009.59.5.9
[15] M. Crosas. 2011. The Dataverse Network: An Open-Source Application for Sharing, Discovering and Preserving Data. *D-Lib Magazine* 17, 1/2 (2011).
[16] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P.J. Maechling, R. Mayani, W. Chen, R.F. da Silva, M. Livny, et al. 2015. Pegasus, a workflow management system for science automation. *Future Generation Computer Systems* 46 (2015), 17–35.
[17] J. Goecks, A. Nekrutenko, and J. Taylor. 2010. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology* 11, 8 (2010), R86.
[18] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M.R. Pocock, P. Li, and T. Oinn. 2006. Taverna: A tool for building and running workflows of services. *Nucleic Acids Research* 34 (2006), W729–W732.
[19] ISO 26324:2012 2012. *Information and documentation – Digital object identifier system*. Standard. International Organization for Standardization, Geneva, CH. https://www.iso.org/standard/43506.html
[20] M.B. Juric. 2006. *Business Process Execution Language for Web Services BPEL and BPEL4WS 2Nd Edition*. Packt Publishing.
[21] J. Starr. 2013. EZID: a digital library data management service. In *A handbook of digital library economics*. Elsevier, 175–183.
[22] R. Tansley, M. Bass, D. Stuve, M. Branschofsky, D. Chudnov, G. McClellan, and M. Smith. 2003. The DSpace institutional digital repository system: current functionality. In *3rd ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '03)*. 87–97.
[23] S. Tuecke, R. Ananthakrishnan, K. Chard, M. Lidman, B. McCollam, S. Rosen, and I. Foster. 2016. Globus Auth: A research identity and access management platform. In *12th IEEE International Conference on e-Science (e-Science)*. 203–212. https://doi.org/10.1109/eScience.2016.7870901
[24] M. Wilde, M. Hategan, J.M. Wozniak, B. Clifford, D.S. Katz, and I. Foster. 2011. Swift: A language for distributed parallel scripting. *Parallel Comput.* 37, 9 (2011), 633–652.
[25] M.D. Wilkinson, M. Dumontier, I.J. Aalbersberg, G. Appleton, M. Axton, A. Baak, et al. 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data* 3 (2016), 160018. https://doi.org/10.1038/sdata.2016.18
[26] J. M. Wozniak, K. Chard, B. Blaiszik, R. Osborn, M. Wilde, and I. Foster. 2015. Big Data Remote Access Interfaces for Light Source Science. In *2nd IEEE/ACM International Symposium on Big Data Computing (BDC)*. 51–60.