COLUMBIA UNIVERSITY
Information Technology

# Columbia University API Strategy

*November 2016*

## Introduction

The broad availability of easy-to-use public APIs from commercial as well as government and nonprofit entities has created an "API economy" that has broken down the traditional walls between holders and users of data. The API-centered approach facilitates rapid development of new web and mobile apps which "mash up" data and application services from many sources.

This API strategy for Columbia explains our motivation and approach to building an API ecosystem for the benefit of our faculty, students, staff, peers and others in promoting the University's mission to advance knowledge and learning at the highest level, and to convey the products of its efforts to the world [13].

This API strategy is part of an overarching application and data integration convergence strategy, which allows organizations to consistently leverage integration competency across both application and data disciplines. [14]

## What are APIs?

API stands for Application Programming Interface. This term has been in use for decades to describe the standard way that programmers can invoke application services. However, today's modern definition of API is specific and includes some key attributes:

- Based on the web – APIs are accessed via HTTP GET, PUT, POST, etc.
- Used to cleanly separate user experience from the underlying processes and systems [15].
- Modern successful APIs are implemented using the REST [2] paradigm which builds standardized data access on top of the highly successful HTTP model. REST APIs are easy for developers to understand and use, something that cannot always be said of predecessor approaches such as Service Oriented Architecture Web Services like SOAP [11] and CORBA [12].
- Data is represented using a shared convention, typically as JSON documents, following a standard style such as JSON API [3] or HAL [4].

Some examples of widely-used web APIs include the Google Maps APIs [5], which allow developers to leverage the power of Google Maps in their apps, the Twitter API [6], US Postal Service [7], and the NASA Earth API [8].

# Why APIs?

## APIs to disintermediate user interfaces

"Fat" client-server apps have been largely replaced by "thin" web-client apps. However, the user interaction with the app server is generally defined and constrained by a human user interface (UI). Using such an interface from another app (screen scraping) is difficult and prone to future errors as the user interface is updated. By disintermediating the web UI from the API, it becomes possible to "re-skin" the interface or develop multiple interfaces to the same service, e.g., a mobile app or integration with another app. This disintermediation is a modern take on the Model-View-Controller design pattern [9].

## APIs for enabling self-service for app developers and app users

Currently it is an uphill battle for application developers to convince data custodians to share the right amount of data, with the right frequency and security. This generally results in development of a new system-to-system batch file transfer "interface" and does not free up the data for ad-hoc innovative use in appropriately secured ways. Deployment of modern, secure RESTful APIs can streamline this process for the data custodian (develop once), encouraging reuse of canonical interfaces by many (as-of-yet unknown) applications.

## APIs to reduce barriers to accessing enterprise data

Currently there is no established process at CU to triage enterprise data sharing requests; every data request is treated differently and hence it is difficult to estimate when the enterprise data will be available for use. Establishing an API intake and governance process with reusable application building blocks will provide better visibility of available APIs, of request submission when APIs don't already exist, approvals to release data—if at all, and with appropriate protections—and, prioritization of the work necessary to create these APIs.

## APIs for integrations

A common approach (and frustration) is for an integration to be established between application X which needs some data from application Y. This is all too frequently implemented as a daily batch job that feeds a data extract from application Y's database which is then loaded into application X. Using an API call instead, application X can just grab the (typically small number of) data elements from application Y on the fly in real time, resulting in current rather than day-old data.

Furthermore, this API integration point can then be easily reused when application Z comes along and also needs similar data from application Y.

### APIs for microservices

Microservices[10] are small, lightweight components that cooperate to implement an application. Using loosely-coupled microservices, individual components can be upgraded or replaced as needed in a continuous deployment model.

### APIs for innovation

Ultimately, the value of APIs is in supporting innovation. Developers can gain quick, easy, and —through appropriate security controls—secure access to data and services which they can use to build new apps. These developers will frequently not be "central" staff but located throughout the organization and our "customers" (e.g., students, members of the public).

## Building the API ecosystem at Columbia

In order to build a successful API ecosystem we need to:

- Establish standard reusable API design patterns
- Incorporate appropriate data security and auditability measures, to ensure access, usage, and transmission is governed and is trackable for audit and compliance purposes
- Develop a mechanism for developers to find, use, and share their APIs
- Create logical data models and schemata which accurately and consistently reflect data being shared
- Develop and establish data governance and stewardship across platforms, and an awareness to keep the data clean and consistent

As a start in this direction, CUIT is leading the initiative to establish appropriate University-wide logical data models and data governance, and is establishing specific tools and standards for API discovery, development, and deployment which will be shared among developers at CU and, as appropriate, developers outside CU.

## References

Readers interested in further information in support of this strategy are referred to the following:

[1] http://www.programmableweb.com/api-university/what-are-apis-and-how-do-they-work
[2] Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000. Accessed 7/25/2016 at https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
[3] http://jsonapi.org/
[4] http://stateless.co/hal_specification.html
[5] https://developers.google.com/maps/web-services/

[6] https://dev.twitter.com/rest/public
[7] https://www.usps.com/business/web-tools-apis/welcome.htm
[8] https://api.nasa.gov/api.html
[9] https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
[10] https://en.wikipedia.org/wiki/Microservices
[11] https://www.w3.org/TR/soap/
[12] Henning, M. The Rise and Fall of CORBA. ACM Queue v. 4 n. 5, June 2006. Accessed 8/8/2016 at http://queue.acm.org/detail.cfm?id=1142044
[13] Mission Statement. Columbia University in the City of New York. Accessed 9/12/2016 at http://www.columbia.edu/content/mission-statement.html
[14] Implement an application and data integration convergence strategy with three best practices. Gartner article - September 2015 - ID: G00279084 (local copy)
[15] API-led Connectivity: The Next Step in the Evolution of SOA. MuleSoft whitepaper. Accessed 6/16/2015 at https://www.mulesoft.com/lp/whitepaper/api/api-led-connectivity (local copy)