

GSA API Strategy - Navin Vembar and Joseph Castle - 3/4/16

“The rise of application programming interface (API) represents a business opportunity and a technical challenge¹.”

This GSA API Strategy describes GSA’s future direction for agency-wide API management including design, development, architecture, operations and support, and security.

Objectives

- Harness API management to maximize customer value and technical efficiencies.
 - Expose public data for transparency, data consumer engagement, app development, etc.
 - Promote system integration; promote migration of legacy systems by moving data management away from system logic allowing for future technology solutions.
- Adopt industry best practices with API design, development, and management.
- Consider opportunities for revenue through API offerings.

Figure 1 (below) provides an overview of the importance of API interactions with internal and external organizational environments. These are items to consider for pursuing an API strategy and management.

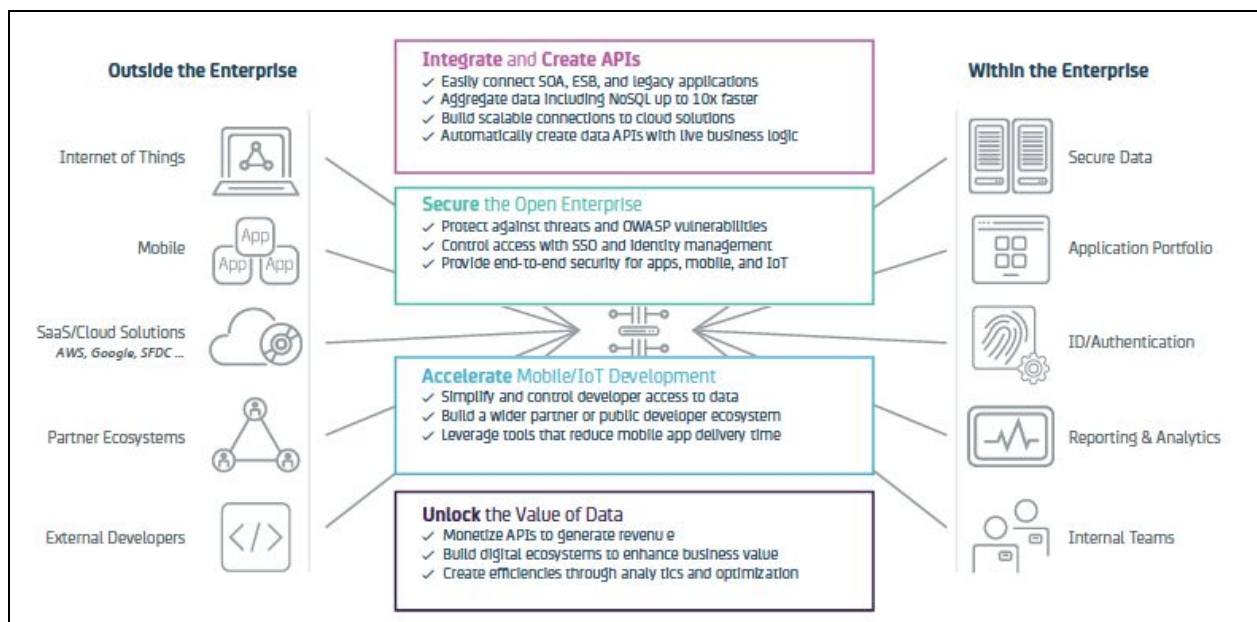


Figure 1: External/Internal API interactions²

¹ CA Technologies. 2016. *API Strategy and Architecture: A Coordinated Approach*.

² CA Technologies. 2016. *The API Management Playbook*.

Future API Development at GSA

The future of API development in GSA will follow industry best practice as discussed below. A way to consider API management is in two categories: 1.) the actors and corresponding technologies, and 2.) particular API specifications.

Actors and Technologies

A graphical representation of actors and technologies is featured in *Figure 1* and considers the components and actors of proper API management.

- Actors as Developers - App Developer, API Owner, API Architect - these individuals are concerned with creating an API to meet the business needs in exposing data.
- Actors as Consumers - End User, Client App - these individuals or technologies are interested in consuming the data exposed by the API and need stable code and supporting documentation.
- Developer Portal - provides an interface where developers access APIs, documentation, community forum, and other useful content (e.g., open.gsa.gov) (Note: need internal development environment to build, test, and deploy APIs; need proper development tools.). Positive examples include open.nasa.gov and open.fda.gov.
- API Gateway - delivers the security, caching, and orchestration to deploy an API. Defaulting to api.data.gov for public APIs.
- API Implementation and Backend System - these are core GSA systems where data exposure facilitates efficient processes.

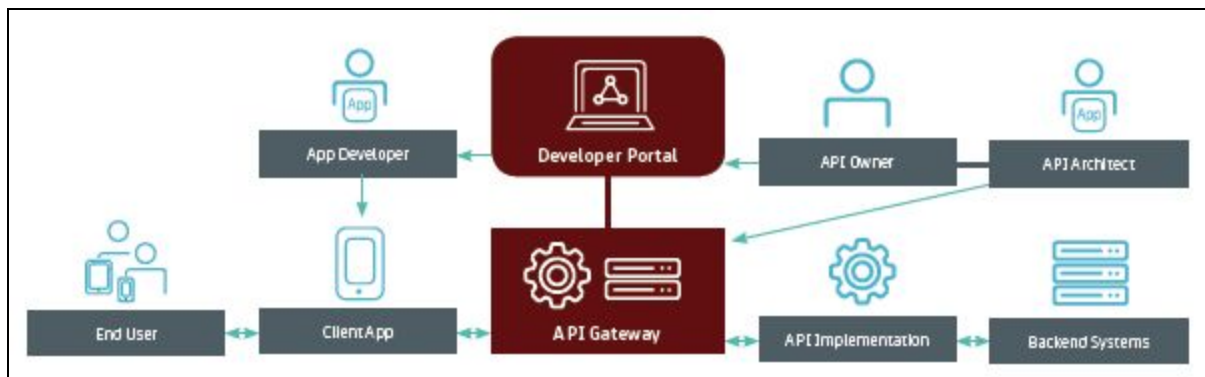


Figure 2: API Management Components by CA Technologies: API Strategy and Architecture³

Along with the actors and technological components, GSA API developers must consider the purpose for creating the API and the audience wanting to consume the API.

- Purpose for creating an API
 - GSA wants to expose data for compliance or for operational efficiency.

³ CA Technologies. 2016. *API Strategy and Architecture: A Coordinated Approach*.

- Consider the audience and the amount of support needed (i.e., What happens if the API changes? What happens if the data changes?).
- Reduce “hard” dependencies between implementation teams by agreeing to shared APIs that are built and consumed independently - moving us along a microservices model.
- Audience for consuming the API
 - Public - general data consumers - customers, agencies, industry, non-profits, universities, public, etc.
 - Private - internal GSA systems w/ corresponding program offices; external GSA systems w/ corresponding program offices. Internal implementations within a system.

API Specifications

The following outlines how future APIs should be designed and developed at GSA. APIs that do not meet these specifications can be brought in-line if deemed necessary by the API and business owners.

Goals of API Design⁴

- Enable self-service for app developers and app users.
- Reduce barriers to accessing valuable enterprise resources.
- Prioritize needs and preferences of client app developers.
- Encourage collaboration between and among internal and external resources.
- Address security and scaling issues of exposing IT assets to the public.

Design and Development Considerations for GSA

- Consider private versus public API creation - private similar to web services (but using RESTful when possible) for backend system integration and public for allowing 3rd party data use and application development.
- Move to current favored standard - RESTful (either REST (URI) or Hypermedia (True REST); possibly consider Event-Driven (IoT) (See *Figure 3* below).

⁴ CA Technologies. 2016. *API Strategy and Architecture: A Coordinated Approach*.





 <p>Web Service</p>	 <p>Pragmatic REST</p>	 <p>Hypermedia</p>	 <p>Event-Driven</p>
<p>SOA-Related Lots of tooling available Not suitable for mobile</p>	<p>Ideal for Web and mobile apps Familiar to most app devs May not be adaptable over time</p>	<p>Highly web-centric Scalable and evolvable Not familiar to many devs</p>	<p>Appropriate for IoT and devices Lightweight and dynamic Not suitable for standard scenarios</p>

Figure 3: Architectural Styles for API Design

- All new solutions should come with APIs, especially for exposing public data. No longer sufficient to provide only extracts; consider internal architecture particularly with private APIs.
- Consider integrations tied to APIs and not proprietary systems (e.g. Salesforce, Common Data Services).
- Determine criteria and development of API; priority to develop and resources.
- Features of API - metadata, rules, use, etc. (e.g., Data Standards of 18F and WH).
- Develop terms of service - <https://petitions.whitehouse.gov/how-why/api-terms-use>.
- Build-time governance to include a catalog of existing APIs allowing for individuals to understand what is already in existence before building something new.

Architecture (see Figure 2 below) - consider these four areas

- Security Layer - APIs open the world to internal enterprise resources which calls for proper security being enacted. The most popular is OAuth2 but others include OpenID Connect, Certificates, Keys and Key Management.
- Caching Layer - consider responsiveness provided to API developers and data consumers for timely delivery of common requests.
- Representation Layer - should be as developer friendly as possible. This should focus on creating a welcoming way into the API without impacting the backend resources or the API itself.
- Orchestration Layer - allows for multiple backend resources and composing new implementations with one or many APIs.

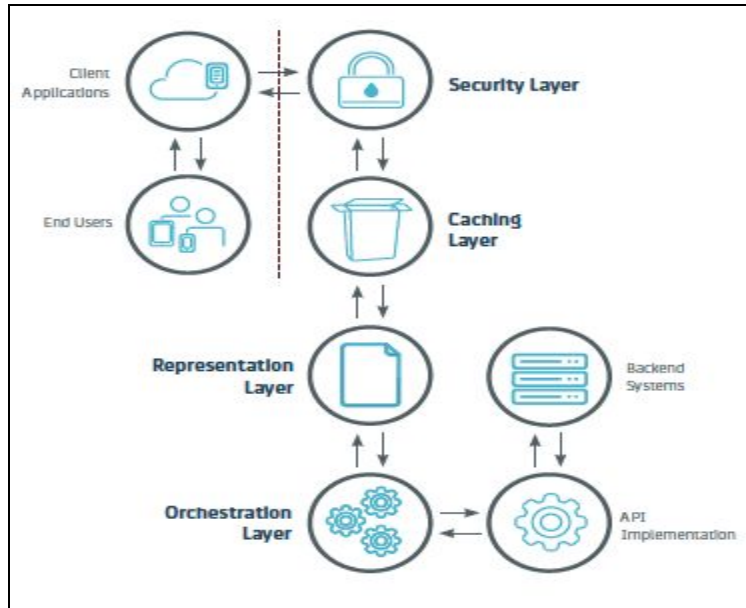


Figure 4: Architectural Layers⁵

Operations and Support

- Provision of APIs - consider how the API will be managed and how consumers will interact with the API Gateway and Developer Portal.
- Focus on documentation standards and use common tools (e.g., Swagger, RAML, others).
- Expand documentation and style on open.gsa.gov/developer.
- Determine API use and rules for consumers to obtain data (keys, analyze, and control their traffic (limit gets/posts per hour/minute etc)).
- Consider building a prototype for API testing of the design and target persona - consider commercial tools for assistance (e.g., Apiary).
- Need for API monitoring for holistic view of API activity across the GSA (Google Analytics for APIs.).
- Metrics for tracking use - collecting information about the consumer (e.g., IP address, sys name, login, etc.)
- Consider API management tools (e.g., RedHat, APIMAN).

Next Steps

The next steps consist of incorporating the Gartner Recommendations with implementing the recommendations for the agency including potential easy targets identified below.

Gartner Recommendations⁶

⁵ CA Technologies. 2016. *API Strategy and Architecture: A Coordinated Approach*.

⁶ Gartner. 2015. *Framework for Web APIs*.

- Manage Web APIs as a technology product, not a transient IT project.
- Adopt a customer-centric approach to Web API design that includes personas and scenario design.
- Explicitly define legal and technical contracts.
- Commit to implementing and sustaining strong post-production customer support.

Recommendations for GSA API Implementation Based on Strategy Research

- Complete inventory of APIs across GSA - find all APIs, consider owner, purpose, customer base, architecture, operational support, etc.
- Research and stand-up API Gateway and Developer Portal (Note: open.gsa.gov is a starting point for a Developer Portal).
- Consider API Management service providers and software for long-term design, development, and management of GSA APIs (maybe for learning and mimicking).
- Identify “best of breed” APIs as examples to mimic; identify a group of APIs of value (to us, to a consumer) and enhance them to match the prescriptions in this strategy doc (or standards doc to follow).
- Consider public data sets to have public APIs; consider private system connections to have private APIs and private with shareable APIs to trusted partners.
- Consider APIs that could have monetary value.

Easy Targets

- Public APIs: Open Data datasets, public acquisition data (eBuy, A!), building data.
- Private APIs: Travel approvals, speaking engagements, integration with Concur, current agency SLAs and VPNs connections, others.

Sources

18F API Standards - <https://github.com/18F/api-standards>.

CA Technologies. 2016. *API Strategy and Architecture: A Coordinated Approach*.

Forrester Research, Inc. 2015. *Brief: Four Ways APIs Are Changing Your Business*.

Gartner. 2015. *A Guidance Framework for Designing a Great Web API*.

Fielding, Roy. 2000. *Architectural Styles and the Design of Network-based Software Architectures*. Dissertation.

Jacobson, Daniel. 2011. *APIs: A Strategy Guide*. O'Reilly Publishing.

Lane, Kin. API Evangelist, www.apievangelist.com.

Newman, Sam. 2015. *Building Microservices*. O'Reilly Publishing.

Richardson, Leonard. 2013. *RESTful Web APIs*. O'Reilly Publishing.

White House API Standards - <https://github.com/WhiteHouse/api-standards>.

Examples

We the People API

open.nasa.gov

open.epa.gov

open.fda.gov