Findings on API Governance Models in the Public and Private Sectors

Summary

Department of Veterans Affairs (VA) awarded Sprezzatura Management Consulting, LLC a "micro task order" to research governance best practices for Application Programming Interfaces (APIs) in the public and private sectors. This document explains the research method, provides observations and analysis, and concludes by highlighting specific recommendations. It focuses on lessons learned in the interviews, and documents strengths (and a few weaknesses) observed in the areas of prioritizing API's to build, implementation patterns and customer (developer) experience.

Research Process

Who we interviewed? During the engagement, we contacted over 30 organizations and conducted a total of 9 interviews. For quality assurance reasons, we ultimately limited our analysis to 6 organizations including 3 from the public sector, and 3 from the private sector. Below is a list of the organizations on which we focused. We started each interview with scripted questions, but conducted the conversation in a "free-flowing" way allowing us ask follow up questions on comments we thought interesting from a VA perspective.

- **General Services Administration (GSA)** - We interviewed the lead of the Emerging Citizen Technology Program Office for GSA's Technology Transformation Service. This team's mission is to unite federal managers to improve the creation, efficient adoption and performance of new technologies.

- **Internal Revenue Service (IRS)** - We spoke with team members from the Modernized e-File system. They maintain a set of API's to ingest taxpayer forms from third party systems (e.g. Turbo Tax) with requirements roughly analogous to some of those of Veterans Benefits Administration (VBA) ingesting forms from Veterans Service Organizations (VSO's).

- **Forest Service** - We conferenced with the Solution Architect from the CIO's office and the Chief of Enterprise Business Solutions Services. They are in the midst of transitioning to an microservice and API architecture which has included developing enterprise documentation and running several pilots.

- **MuleSoft** - We interviewed the Director of Global Customer Success Architecture and a Client Architect. Given MuleSoft's current experience with the Digital Veterans' Platform (DVP) we focused on gaps they see between and other clients, such as large financial services firms.

- **1upHealth** - We met with the Chief Executive Officer. With roots in the United States Digital Service, 1upHealth supplies a platform that enables exchange of electronic health data between patients, providers, and application developers. Today, they connect with 166 healthcare systems across the US including Mayo Clinic, Inova, etc.

- **SalesLoft** - We met with the Vice President of Product and a Product Manager. SalesLoft has spent the past two and a half years engaging third party developers to enable them to build powerful, usable, custom experiences for their users by utilizing SalesLoft APIs and coding guides.

Findings on API Governance Models in the Public and Private Sectors

What we focused on and why? - We focused our questions in the areas of How API's Get Prioritized for Build, Implementation Patterns, and Consumer (Developer) Experience. We thought these areas would be interesting because of our experience within Enterprise Shared Services and Digits to Digits. From this perspective, we prepared questions for the interviews answers to which we thought would benefit VA. answers could be readily usable by VA. The questions we used are included at the end of the document.

Bottom Line: No organization we interviewed had a "monolithic" governance scheme with a master set of "rolled-up" procedures to provide a uniform process for addressing all aspects of their API's exposure to consumers and lifecycle. The IRS came closest to this, but even they had a separation between their business and IT teams' governance work. We suspect, but do not have direct evidence, that this is the result of a number of contributing factors. These could include the need to maintain the speed of developer teams, a willingness to adopt best-practices at low levels of an organization in order to speed adoption; and in government agencies, the difficulty in imposing mandated change across a large enterprise.

Each organization did have patterns and degrees of formalized controls they used in the areas we queried. We identified usable lessons learned in the areas of evangelism, authorization (whitelisting/blacklisting), maximizing developer experience, enterprise architecture, CI/DC support and documentation in a federated environment, among others.

Analysis

Area 1: Prioritizing What API's to Build

Strength: A Dedicated User Permission API - 1upHealth created a single API "to programmatically create users and granularly manage their data permissions." This uses OAuth2 tokens and manages permissions across the rest of the API's 1upHealth exposes.

> Analysis: VA already has multiple permission schemes for its publicly facing APIs. D2D, the DVP Pilot, and VLER VHIE interfaces are not coordinated on this. VA could potentially focus on building or acquiring its own dedicated User Permission API and exposing that early in its API Platform program. The requirements effort on this would likely be a complicated one given VA 6500, HIPAA, PHI, multiple business line requirements, etc, with multiple user roles and "granular" permissions that would likely be required. This would however likely speed adoption of subsequent APIs.

Strength: Building API's Above Cerner, Epic, and other Provider EHRM API's - 1upHealth creates value for individuals and 3rd party application developers by layering above the Cerner and Epic (and other) EHR's of providers. They maintain their own record of the patient with its own canonical model mapped to popular EHR's. This record can include inputs from multiple providers. It also includes patient/user input as well as sensor data (e.g. wristband input).

> Analysis: VA will have a challenging time deciding what API's and data it should own as well as deciding what it should share with others. This has implications for choosing what API's to develop. For instance, should VA own an API (and accept data) that enables a Veteran to store information from outside provider for treatment not

associated with a service disability? Or, will VA be able to ingest sensor data from a third party API? We recommend that any API Governance initiative include liaison with Veterans Health Administration (VHA) on forward looking policy questions concerning health data sharing beyond what is maintained within the Veteran VistA and Cerner records.

Area 2: Implementation Patterns

Weakness: Heavy Handed Enterprise Architecture Kills Momentum - In its consulting engagements, Mulesoft observed that a heavy governance process that centrally manages what API's get built and how they fit into a canonical SOA model, kills developer team momentum.

Analysis: Mulesoft recommends embedding enterprise level architects in individual development teams as collaborators. The goal is not to conduct a formal check of what API's get built at the enterprise level. Breaks should only be put on development teams when there is an obvious duplication of effort.

Strength: Defined Partner "Trusting" Process - IRS processes third parties through a security vetting process prior to granting them "Trusted Partner" status and allowing them to connect to the API's.

Analysis: VA may want to consider and enterprise level process for trusting partners, especially for API's that allow Creation, Update, Delete responsibilities.

Strength: Lightweight Whitelisting Process - GSA discussed the process to whitelist users of an API, including that of the White House's "We The People" API. GSA advised that the whitelist process should be light, except when giving write privileges. They noted the IRS process was an extreme example of a heavy whitelisting process, driven by their requirements.

Analysis: We anticipate that it will be challenging for VA to develop an enterprise whitelisting/authorization to use process due to the number of business units, variety of use cases, and potential to handle PII and PHI. We recommend that the API Platform governance team create tiered templates for lightweight whitelisting and heavyweight whitelisting. These could then be presented as "out of the box" solutions for specific business line requirements.

Strength: Robust Testing Prior to Engaging Third Parties - IRS fully tests all API's prior to exposing them to vendors. They then call vendors into a single test environment to validate their own systems against the APIs. They allow vendors to do this for a two month period before a hard cut-over in production. IRS does not intend this testing with third parties to uncover defects in its own APIs as IRS only places production ready API's in this environment.

Analysis: Somewhat counterintuitively, it may be better to minimize interaction with vendors during the testing phase in order to ensure they only have to interact with a stable API. The IRS model of a hard cutover will not work for VA, who receives claims continuously and who has requirements around establishing the date of claim.

Strength: When Third Party Testing Is Required, Engage a Motivated Consumer - 1upHealth will not engage any developers in testing until 1upHealth has thoroughly tested their own API. They use a reference implementation model whereby they test the API's as if they were a Consumer

API. Once that is complete, they will invite in the application developer that originally requested the functionality for a final check of the API and documentation before going to Production. This is more about a test of the third party's consuming API, than about testing 1upHealth's, but it does provide a final check on 1upHealth functionality.

> Analysis: This is consistent with our testing schemes in our interview process. We recommend that VA require its contracted developers to be responsible for Reference Implementation testing of their own API's prior to any final testing with a consumer.

Strength: Provisioning of an Enterprise CI/CD Platform and Associated Training - Mulesoft recommended heavier governance in the area of CI/CD pipelines support for development teams. They cited the adage that "it is easier to build then deploy." It is inefficient to have each development team built and manage its own pipeline. That can be done centrally. There is then a requirement for education and documentation to support the development team's use of the pipeline.

> Analysis: While management of a central CI/CD deployment pipeline is a governance issue, it may be better for VA to plan for resourcing such an effort first, and then make this a separate acquisition from either PMO services or developer services. This is a different model than currently pursued in VA acquisitions.
>
> Either in a CI/CD "as a service" acquisition, or in a governance support acquisition, we recommend including a developer education and documentation component beyond standard VIP documentation requirements.

Strength: Not forcing OAuth2 in Test Environment - Salesloft discussed their practice of not using the OAuth2 protocol on API's in the public facing test environment. They did this to enhance the developer experience. Their believe was that it was burdensome for third party developers to build the flows necessary to implement OAuth2. This could be a prohibitive amount of effort for developers who simply want to test a Salesloft API to determine whether or not it is worth committing the effort to develop their own third party application to interface.

> Analysis - We recommmend that VA consider this approach when developing its authorization governance model. The goal would be to enhance the developer experience in the public facing test environment. The trade off would be in the engineering resources required to ensure a version of an API exposed in test does not required OAuth2 (or other) authentication. It would likely be prudent to publish a different version of the API test with authentication turned on in order to provide that validation prior to migration to production.

Weakness: Ancillary Tool Requirements to an API Gateway Tool Can Add to Licensing Costs - WIthout getting into specific tool pilots, Forest Service noted that they witnessed licensing costs increased by ancillary tools required for API Management. They noted that many API Management tools on the market do not offer all of the offer full functionality. There could be different tools for the Gateway, queues for microservices, monitoring, SLA enforcement, etc. The expectation from vendors being that users by other tools from the offering's family to round out functionality. Forest Service noted that they used Gartner research and their own pilots to examine different tools.

Analysis - VA currently uses a variety of tools for API and microservice platforms, gateways, etc. and has conducted its owns studies of what to acquire over the previous few years. It is not clear that there is an obvious answer for large federal agencies based on the VA and FS' experience. Regardless, a single API directory and gateway would appear to be a requirement for ease of use for 3rd party developers. Early governance activities could focus on making a decision for an enterprise gateway tool, factoring in lifecycle cost considerations. We anticipate this being a challenging decision requiring focused resources to properly account for trade-offs.

Strength: Purposeful Division of Teams on Github to Enforce Architectural Goals - Forest Service reported that they used Github as a governance structure for enabling their target API and Microservice architecture. For any project, they forced the division of the project into three teams on Github:) UI/UX; 2) API Development; and 3) Data Services/MicroServices. This helped ensure a microservice architecture on new initiatives.

Analysis - This is interesting from a VA perspective given the amount of effort put into building the current enterprise schema in Rational as well as the mandated use of Compliance Epics. To our knowledge, there is no incentivizing of an API or microservice approach within the VIP requirements. VA could consider amending VIP and the Rational schema/Compliance epics to encourage APIs and microservices. This could potentially be piloted within Enterprise Shared Services or encouraged on any new project using Github.

Weakness: Reliance on Non-Cloud Based Legacy Systems Decreases API Availability -The Forest Service confirmed that they do have experience API uptime issues associated with API's built on top of legacy data stores not yet located to the cloud. They observed that cloud based systems are more reliable resulting in better uptime for associated API's.

Analysis - We see the governance implications for API development in the VA domain as two-fold. First, as VA initiates its own Cloud migration initiative, it may be worth factoring in the API pipeline when considering prioritization of legacy system migration. That is, upon examination of the API build pipeline.

Weakness: Even with Mature Governance, Third Parties Cannot Be Trusted to Use API's as Intended - IRS has observed that partners don't use API's as designed. The example they cited was exceeding the number of planned calls per second due to implementation errors on the side of the third party. 1upHealth made a similar observation.

Analysis: VA could address this both through SLA's monitored and enforced through an API Management tool. The SLA would define limits on calls to an API. A modern API Management tool could then make this easily configurable, such that the SLA could be operationalized and issues addressed without touching the code base.

Area 3: Consumer (Developer) Experience

Strength: Dedicated Evangelism Function - GSA noted the importance of the larger community of federal-related API developers and their participation on certain web sites and even in-person meet ups in the Washington, DC area. Outside of possibly Digital Services teams at VA, it is unclear to us that Enterprise Shared Services, or other development teams, participate.

Analysis: We recommend that evangelism be explicitly addressed in future API Governance initiatives. This evangelism should include external outreach as well as internal outreach to VA development teams that could either ultimately expose an API publicly, or who could provide Process or System layer API's/services supporting exposed API's.

Strength: Two Tiers of Technical Documentation - The IRS maintains two different tiers of technical documentation for security reasons. The first tier is publicly exposed via the EFS website. This is scrubbed of all specific information that could enable a cyber attack. The second tier of technical documentation is only available once a third party clears the onboarding security review and becomes a "Trusted Partner." It has technical details necessary for actually enabling the interface.

Analysis: VA could consider which APIs carry a similar security risk and then provide documentation through an analogous process.

Strength: Use of Peer Review To Ensure a Positive Developer Experience - GSA recommended having 2-3 "outside" developers come in and review web documentation on a new API. This "murder board" approach will poke holes in assumptions made by the development team.

Analysis: This is a useful idea to adopt and could be placed as a task or optional task in a contract. Our experience is that VA will ask an API consumer to do this. That's fine, but it risks wearing on the consumer as we're asking them to expend funds to support the VA's API development process. Recommend linking the requirement to have a small cadre of developers available for this type of review to that of a formal API Evangelist role within a governance structure (see below). Recommend that outside developers be engaged only when VA feels it has fully mature documentation based on a best practice (also see below).

Strength: Copying Best Practices - When GSA plans to produce and publish developer notes to augment their API contracts, they look to other sites to determine best practices and then emulate those on their own API web pages targeted at developers.

Analysis: Documentation that is developer friendly, and highly accessible through public web pages, is typically not a requirement/specification within T4NG contracts nor necessarily a core competence of the current VA teams developing middleware services. Recommend that prior to any new contracts for API development being written, that a review of some API Developer webpages be completed in order to identify common denominator elements. These can be published in Playbooks, Design Patterns, and required in future contracts.

Weakness: Communicating System Outages in a Multilayered Orchestration - 1upHealth experiences a similar challenge to the D2D model in VA. They must be able to communicate outages in a multi-layered orchestration to a third party application consuming their services. They do this by providing a web page dashboard that system administrators of consumer applications can manually check. They are still maturing this process with the idea being that 1upHealth makes periodic (every few minutes) calls in Production to underlying provider systems using test data. The dashboard can then me automatically updated based on these service-level calls.

Analysis: VA could follow a similar maturation path for providing consumers of its API's visibility into the overall system health. The challenging part will be staging of test data for this in VA legacy systems or establishing a protocol for testing using the Production data in a way that does not impact the Veteran record.

Strength: Designate a Customer Success Engineer - Mulesoft discussed the importance of having a dedicated "Customer Success Engineer." This person would be responsible for direct liaison with Consumer API Developers when VA develops a new API for them and the associated documentation. This person would be on a VA team, but maintain the perspective of an outside developer consuming the VA API, focusing ease of use of the interface.

Analysis: We recommend VA acquire services for such a roll. This would need to be placed as an explicit requirement in either a PMO contract or a API development contract. There should be governance over the full interaction with Consumer Developers (design, authorization, testing, etc) and such a contracted resource would need to have explicit responsibilities in each step of the process. The emphasis should be on minimizing effort by the third party developer and simplifying their experience.

Strength: Verify Third Party Applications Function As Expected Prior to Whitelisting - Salesloft reported that their Product Managers engaged in an effort to verify 3rd party applications prior to them being whitelisted. Once a developer builds their consuming application and tests it, they then contact the Salesloft integration team in order to be whitelisted. The team will then perform some testing of the third party application to ensure it does in fact function properly with the Salesloft API and data.

IRS has a somewhat analogous process. They publish a set of test cases for third parties to run in the IRS test environment prior to that third party being authorized to use production.

Analysis - VA builds and exposes more API's to be consumed by third party applications, it will likely become increasingly difficult to verify those third party apps function as expected for Veterans and other users. As part of its governance model, VA could consider how much governance it wants to place over 3rd party apps prior to them being whitelisted in production. For example, is there a requirement for VA to itself test some third party applications? We anticipate a trade-off here between making it easy for developers to deploy their own apps consuming VA data and imposing a heavy testing and whitelisting process. We recommend VA builds this aspect of governance incrementally, with the input from any Product Managers or Customer Experience Engineers it retains.

Selected Recommendations

The analyses above include numerous recommendations that could be applied in to an enterprise API governance model for VA. Below, is a summary of the ones that our team believes will be most impactful. This is based on our experience within the Enterprise Shared Services domain and our familiarity with the ongoing Digital Veterans Platform pilot.

Findings on API Governance Models in the Public and Private Sectors

Apply Governance Resources Early to Development of a Partner Trusting Process and Associated Permissions API - Based on our interviews, we believe that applying governance early to the partner trusting process and linking that to a permissions API will enable third party developers to interface easily with VA produced API's. The IRS, 1upHealth, and Salesloft all discussed their respective processes with us. Some processes, such as the IRS' were "heavy" in that there was a high bar that developers had to pass in order to connect to the IRS API's. This will likely be the case for many of the VA API's due to the sensitivity of data (PII, PHI, etc). Governance areas would include: 1) process definition, 2) associating that process with configurable permissions (such as the OAuth2 protocol), 3) ensuring a clear developer experience for the trusting process and permissions, and 4) a defined testing process of third party applications prior to whitelisting. We anticipate this taking an investment of planning resources up front given the breadth of possible VA use cases that could be supported through API's. We anticipate VBA, VHA, and NCA having varying requirements. It could prove difficult to develop a process and configurable permissions scheme that easily adapts to new API's with varying functionality. VA will likely face trade-off decisions in this process. Regardless, we recommend expending the effort.

Apply More Effort to CI/CD Pipeline Governance and Support - Our understanding is that VA intends to ultimately have multiple development teams, potentially under different contracts or not even under contract, building API's on an enterprise platform. To support these efforts, we recommend that VA provide a single cloud native CI/CD pipeline that any of these teams could use. Instead of mandating use of the pipeline, VA could focus on ensuring the governance is sufficient and user-friendly enough that individual teams would be hard-pressed to justify building their own pipeline. This initiative would extend beyond merely a well governed and documented process of automating the blending of individual developer's code, compilation, testing, and deploying through test and production. It would also focus on training and internal evangelism targeted to development teams. By applying governance in a way that eases deployments would provide development teams, CI/CD "as a service" such they could focus on building, and not deploying.

Such an initiative would likely have could have contractual applications requiring governance planning. VA would need to decide if this provisioning would best be done under an API build initiative, or procured independently. The responsibilities for configuration, documentation, education, and evangelism may also have to be organized between contractors. We recommend either contracting for these services separately, or if placed on a larger API development contract, ensuring there are contractual incentives for the awardee to encourage use by other development teams.

Apply Governance to the Developer Experience via a Customer Success Engineer or Product Owner - We recommend staffing a dedicated position of Customer Success Engineer or Product Owner. Both Mulesoft and Salesloft did this explicitly. This position would be responsible for guaranteeing a developer friendly experience for 3rd parties interfacing with VA API's. This would include ensuring that URL and version naming conventions are sensical, that RESTful inputs and outputs are easy to understand, and that supporting documentation is appropriately posted and easy to follow. In the case were there is collaboration required with VA API developers, this role would also act as the consumer's advocate in requirements, design, and testing processes. With a focus on making the experience efficient and pleasurable for the third party.

Appendix A - Additional Information

Questionnaire Questions

As a team, we used these listed questions below only as a basis for conversation. We strayed from this starting point during interviews based on what we found interesting.

Our interview approach was to use these questions as a basis for each interview, but we intentionally did not "stick to the script." We instead followed a conversational method where we asked questions that made sense within the context of the discussion.

Interview Agenda Items/Questions:

General:

- What is your role in your organization?
- How many APIs do you build each year? How many are for external consumption? How many for Internal consumption? What are the security concerns for internal versus external consumption? Firewall relaxation process and procedures.
- Do you utilize external industry publications such as Gartner?

Prioritizing APIs to Build:

- How do you prioritize the APIs you're building? Do you use an Agile Backlog, or other method? Do you evaluate based on Level of Effort and ROI, or use other metrics? Do you prioritize APIs as to their eventual utilization across the enterprise versus a single project?
- How much of your API backlog is defined by a Consumer's request? How much do you attempt to anticipate the needs of Consumers and build APIs speculatively? Should speculative API development be more or less constrained by defined governess specifications?
- How do you make choices between building Experience layer APIs vs Process layer vs System layer APIs? Experience layer APIs vs Process layer development could conceivably be performed by different development team – how will the governance model be enforced when there is a multi-team development environment?
- Do you allow Experience layer requirements to drive build prioritization on Process and System layer APIs? If not, why not? Or is the option of the building of "mocked" lower level API implementations (Process and System) ?

- If you have APIs/services on legacy platforms, how do you decide when port those over to a new platform? Additionally, how these legacy applications will be "ported" – re-hosted, re-factored, re-built?

Standards to Which to Build

- Do you have multiple, unrelated teams, delivering APIs into the same environment? If so, what lessons have you learned about configuration management across the environment/teams? What level of maturity are your Agile, CI/CD, Dev/Ops capabilities?
- Does your organization maintain formal standards for: API Contracts, Naming Conventions; Version Control, Branching, and Merging; Testing Requirements; Exception Handling; Logging; Security? Would you be able to share any documentation? How is this documentation presented – web pages, WIKIs, Playbook SharePoint sites etc.?
- Do you experience a need to keep documentation and configuration control minimal/light? How do you ensure you're minimizing paperwork/bureaucracy and maximizing code delivery? Again Agile, CI/CD, Dev/Ops capability levels.
- How do you enforce standards across multiple development teams?

Consumer Experience

- How many customers external to your organization consume your APIs?
- What have you learned about development, testing, deployment that you apply to maximizing consumer experience?

Closing Questions

- Are there any internal documents (not published on your webpage) that you could share that would show how you apply governance internally?
- To whom do you look to for API governance best practices, with whom else should we speak?
- May we please circle back with any follow-up questions?

Findings on API Governance Models in the Public and Private Sectors

Companies Contacted

Company or Agency Name/Public or Private
RedHat (Private)**
RepreZen (Private)**
DHS (Public)
GSA (Public)*
Department of Ed (Public)
Salesforce (Private)
Microsoft (Private)
EHR (Private)
Avant (Private)
IBM Peru (Private)
FRTIB (Public)
IRS (Public)*
GEICO (Private)
MuleSoft (Private)*
US Forest Service (Public)*
SmartBear (Private)**
Office of National Coordinator (Public)
Allstate/ ARITY (Private)
State Farm (Private)
Liberty Mutual (Private)
Humana (Private)
PNC Bank (Private)
Walgreens (Private)
Mastercard API (Private)
1upHealth (Private)*
Chase Bank (Private)
Wells Fargo (Private)
PokitDok (Private)
Lemonade (Private)
USAA (Private)
SalesLoft (Private)*

*Interviewed_Analysis Submitted
**Interviewed_Not Considered For Analysis

Team Retrospective / Lessons Learned

[Top 7 related to the '*Process of Working the Microtask*']

1. The four week window provided to complete the microtask was as good timeframe. Our approach to the micro task was focused, but definitely not dedicated resources. Flexibility was key.
2. Collaborative approach, flexibility for interactions, as well as timeframe for deliverables (putting quality over rigid timeframes) were all very helpful in the timely completion of this microtask.
3. Difficult to respond to other available microtasks while working on the current microtask (resource constraints and dollar constraints). Also (relational), difficult to perform work on multiple microtasks at the same time. Challenge of the microtask model. Should add more time to respond to the microtask RFIs (14 days versus 7 days would be helpful).
4. Level of effort of the total team combined may go beyond the awarded $10K dollar amount for this microtask. Recommend reviewing the dollar amounts associated with individual microtasks to determine the breadth and depth of reasonable scope of work for that particular dollar amount.
5. Appreciated the nimbleness and flexibility, and the streamlined nature of the process (including the RFI/proposal process, the open door policy to ask questions of the government, the flexible time frame offered for working on and completion of the deliverables). Truly, an Agile approach.
6. Responding to the microtask worked well. Good to respond in a public forum through the draft RFI process. Good thing to having sharing online (including others that responded to the microtask; seeing things out on GitHub shared).
7. Maximized the value of each call with the interviewees. Opportunity to do things better (I.e. get interview questions to those being interviewed ahead of time to aid in the call structure and interview process). Establish expectations for follow-up at the end of each call, including opening the door for direct interaction with VA personnel associated with the particular microtask.