Lighthouse API Management Platform
Request for Information Response
Solicitation Number – 36C10B18Q2600-001

October 27. 2017

Submitted to:
Mark Junda, Contracting Officer,
mark.junda@va.gov
Kelly Reale, Contracting Specialist,
kelly.reale@va.gov

27 October 2017

Submitted to:
Technology Acquisition Center
Mark Junda, Contracting Officer,
mark.junda@va.gov

Kelly Reale, Contracting Specialist,
kelly.reale@va.gov

Re: **36C10B18Q2600-001**

Dear Mr. Junda,

Agile Six Applications, Inc. (Agile Six), a Service-Disabled Veteran-Owned Small Business (SDVOSB) is pleased to provide you with the following response to the above-referenced request for information. Agile Six believes this team is representative of the very best minds in API development and digital service delivery. As a team, our shared experience, past performance and commitment to the federal agile and API community, gives us great confidence that we can develop the requirements outlined in the RFI.

Contact	Other Information
Agile Six Applications, Inc. Robert Rasmussen, CEO 861 Harold Place Suite 207 Chula Vista, CA 91914 Cell (619) 395-1422 Fax:(619) 900-7702 email: robert.rasmussen@agile6.com web: www.agile6.com	• DUNS: 079622664 • TIN: 47-2367520 • Founded in 2014 • Prime contract holder on IT-70 (GS-35F- 004GA), VA T4NG (VA118-16-D-1011) & CMS EATS (HHSM-500-2017-00004B) • Subcontract holder on GSA 18F Agile Services BPA & VA Vector

1 INTRODUCTION

Agile Six, together with our partners, proudly submits our response and expression of interest in response to the **36C10B18Q2600-001** Request for Information (RFI). Our team includes the most innovative firms in the federal digital service community, featuring:

- **Agile Six Applications, Inc.** (DUNS: 079622664) - A current VA T4NG prime whose principals bring 13 patents on the Amazon Web Services technology stack including Amazon Machine Learning.
- **540.co, LLC** (DUNS: 078812845) - a leader in API strategy and innovation for several DoD organizations
- **Skylight Digital** (DUNS: 80202677) - Featuring Kin Lane a world-recognized expert in APIs and former Presidential Innovation Fellow at VA

Agile Six was launched in direct response to the US Digital Services program where 'Teams of America's most capable problem solvers are striving to make critical services — like healthcare, student loans, and Veterans' benefits — as simple as buying a book online." Agile Six's core competencies focus on the technologies required to empower Digital Services Transformation (DX) including Scaled Agile training and consulting (SAFe and Scrum), Human-Centered Design (HCD), the Digital Services Playbook, Modern Agile Architectures, Cyber Security and Application Development. Agile Six currently works with the Veteran's Administration (VA), Department of Defense (DoD), Federal Labor Relations Authority (FLRA), and the Center for Medicare and Medicaid Services (CMS) delivering Agile Development, Agile Training (SAFe & Scrum), Coaching, PMO Transformation, and Human Centered Design (HCD).

540's mission is to help the Federal Government build modern tools and apply agile delivery services that keep pace with the rapid evolution of technology. 540 is currently working with numerous clients across the DoD to develop API-First strategies. Our team consists of data ninjas and API evangelists, and our strategy leverages open source technology to enable our client's data to be accessible and usable in web/mobile applications, analytical platforms, and other system integrations. 540 specializes in API modernization where legacy data schemas, XML layouts, and SOAP web services are transitioned to RESTful APIs. 540 demonstrates API expertise through our product FedAPI which offers developer-friendly RESTful APIs to the public. FedAPI acts as a research and development platform for the collection, correlation and sharing of public government data. This allows us to learn and prove concepts while supporting the open data community, and helps us accelerate solutions for our government clients by jumpstarting engagements with pre--harvested data or proven design patterns.

Skylight is a new HUB Zone digital government consultancy based out of Chapel Hill, North Carolina. Our mission is to make government work in a digital world using design, technology, and procurement. Our talented

team features several former Presidential Innovation Fellows and founders of 18F, including Kin Lane, a world-recognized expert in APIs. We specialize in modern digital delivery practices and technologies, including design thinking, lean-agile engineering, agile methods, agile architecture, DevSecOps, cloud computing, microservices, open-source software, APIs, legacy modernization, and data engineering, science & analytics.

2 REQUESTED INFORMATION

Question 1: Drawing on your experience with API platforms, how do you see it being leveraged further within the healthcare industry in such a manner as described above? What strengths and opportunities exist with such an approach in healthcare?

An API first platform as proposed by the VA reflects what is already occurring in the wider healthcare space. With veteran healthcare spending being such a significant portion of healthcare spending in this country, the presence of an API platform like this would not just benefit the VA, veterans, veteran hospitals, and veteran service organizations (VSO), it would benefit the larger economy. A VA API platform would be the seed for a federated approach to not just consume valuable government resources, but also deploy APIs that will benefit veterans, the VA, as well as the larger healthcare ecosystem.

Some of the strengths of this type of approach to supporting veterans via an open data and API platform with a centralized API operations strategy would be:

- **Consistency** - Delivering APIs as part of a central vision, guided by a central platform, but echoed by the federated ecosystem can help provide a more consistent platform of APIs.
- **Agility** - An API-first approach across all legacy, as well as modern VA systems will help break the agency, and its partners into much smaller, bite-size, reusable components.
- **Scope** - APIs excel at decoupling legacy systems that are often larger and more complex, delivering much smaller, reusable components that can be developed, delivered and deprecated in smaller chunks that work at scale, and maximize the return on investment of other VA initiatives such as migration to cloud infrastructures.
- **Observability** - A modern approach to delivering an API platform for government with a consistent model for API definitions, design, deployment, management, monitoring, testing, and other steps of the API lifecycle, combined with a comprehensive approach to measuring and analysis brings much needed observability to all stops along the lifecycle.
- **Feedback Loops** - An API platform at this scale, which the appropriate communication and support channels open feedback loops for all stops along the lifecycle, benefitting from community, and partner input from design to deprecation.

These platform strengths promote and facilitate engagement from the entire ecosystem of stakeholders, both within the platform community, and the wider healthcare ecosystem with which the VA systems will integrate. Platforms that engage a rich support ecosystem are more

robust and responsive to change. Stakeholder engagement may take many forms, including but not limited to the following:

- **Practitioner Participation** - An open API platform always starts with engaging backend domain experts, allowing for the delivery of APIs that deliver access to critical resources. Second, modern API platforms help open up and attract external domain experts, and in the case of a VA platform, this would mean more engagement with health care practitioners, further rounding off what goes into the delivery of critical veteran services.
- **Healthcare Vendor Integration** - Beyond practitioners, a public API platform for the VA would attract existing healthcare vendors, providing EHR, and other critical services. Vendors would be able to define, design, deploy, and integrate their API driven solutions outside the tractor beam of internal VA operations.
- **Partner Programs** - The establishment of formal partner program accompany the most mature API platforms available, allowing for more trusted engagement across the public and private sector. Partners will bring needed resources to the ecosystem, while benefitting from deeper access to VA resources.
- **Standardization** - A centralized API platform helps aggregates common schema, API definitions, models, and blueprints that can be used across the federated public and private sector. Helping establish consistency across VA API resources, but also in the tooling, services, and other applications built on top of VA APIs.
- **Data Ecosystems** - A significant number of VA APIs available on the platform will be driven by valuable data sets. The more APIs, and the consistency of APIs across many internal data stewards, as well as trusted partner sources will establish a data ecosystem that will benefit veterans, the VA, as well as the rest of the ecosystem.
- **Community Participation** - In our experience, an open API platform, like other open source software initiatives, engages the interests and efforts of a wide variety of "informal" stakeholders, including developers, data scientists, healthcare practitioners, and private citizens who are passionate about the domain served by the API. An open-standards based API for Veteran-focused services will likely engage a large support community, which will allow these systems to benefit from a far larger spectrum of skills and interests than would otherwise be the case. When it comes to supporting veterans, it is essential that there is a public platform and forum for the community to engage around VA resources.
- **Marketplace** - An open API platform, along with supporting governance processes, helps to establish a marketplace, wherein internal groups can publish APIs, supporting SDKs, and other applications and services, as well as discover existing resources developed by others. Such a marketplace can also be made available to trusted partners, and certified developers, allowing them to showcase their verified applications, and be listed as a certified developer or agency. Marketplaces promote efficient sharing and utilization of resources – both technical and human – which in turn allows development initiatives to focus on delivering new features and innovation instead of redundantly developing existing capabilities.

The benefits of an API platform such as the one proposed as part of this RFI is that they make the delivery of critical services a team effort. Domains within the VA can do what they do best,

and benefit from having a central platform, and team to help them deliver consistent, reliable, scalable API access to their resources, for use across web, mobile, device, spreadsheet, analysis, and other applications. Externally, healthcare vendors, hospitals, practitioners, veterans, and the people that support them can all work together to put valuable API resources to work, helping make sense of data, deliver more modular, meaningful applications that all focus on the veteran.

The fact that this RFI exists shows that the VA is looking to keep pace with where the wider healthcare sector is headed. Four of five of the leading EHR providers have existing API platforms, demonstrating where the healthcare sector is headed, and something that reflects the vision established in this RFI.

Question 2: Describe your experience with different deployment architecture supported by MuleSoft (e.g. SaaS only, On Premise Only, Private cloud, Hybrid – Mix of SaaS and Private Cloud) and in what industry or business process it was used? Please include whether your involvement was as a prime or subcontractor, and whether the work was in the commercial or government sector.

During his time at the Veterans Affairs, and working in government, our partner Kin Lane (principal at Skylight Digital) was actively involved with Mulesoft, regularly engaging with their product and sales team in several areas:
Industry Research – Mr. Lane was paid to conduct API industry research back in 2012 and 2013 by Mulesoft, providing regular onsite presentations regarding his work, contributing to their road map.
Messaging Partner - He worked with Mulesoft on the creation of short form and long form content around the API industry, and as part of his API industry conference.
Sales Process – Mr. Lane has sat in on several sales meetings for Mulesoft with government agencies, enterprise organizations, and higher educational institutions, and is familiar with what they offer.
RAML Governance - Mr. Lane was part of the early discussion, formation, and feedback on the RAML API specification form, but left shortly after he left the government.

Based upon this experience, and regular monitoring of their services, tooling, and engaging with some of their clients we are confident in our ability to tailor a platform strategy that would work with the API gateway, IAM, definitions, discovery, and other solutions. Mr. Lane has been one of the analysts in the API sector who studies Mulesoft, and the other API management providers, and understand the strengths, and weaknesses of each of the leading vendors, as well maintains and understanding of how API management is being commoditized, standardized, and applied across all the players. We are confident this knowledge will transfer to delivering an effective vision for the VA, involving Mulesoft solutions.

Question 3: Describe any alternative API Management Platforms that are offered as SaaS offerings, On Premise, Private Cloud, or Hybrid. Please detail how these solutions can scale to VA's needs managing approximately 56,000 transactions per second through connecting to

VistA and multiple commercial and open source EHRs (conforming to US Core Profiles of the HL7 FHIR standards), multiple commercial Enterprise Resource Planning (ERP) systems, various home grown systems including Veterans Benefit Management Service (VBMS), Corporate Data Warehouse, and VA Time & Attendance System (VATAS), and commercial real-time analytics software packages, and open source tools enabling rapid web and mobile application development.

Multi-Vendor - API gateways have become a commodity and are baked into every one of the major cloud providers. AWS has the lead in this space, with Google, and Azure following up. We stay in tune with a multi-vendor gateway approach to be able to support, and deliver solutions within the context of how our customers and clients are already operating. To support the number of transactions the VA is currently seeing, as well as with future API implementation, a variety of approaches will be required to support the requirements of many different internal groups, as well as trusted partners.

Multi-Cloud - As we mentioned, always look to support multiple gateways across multiple datacenter, and cloud providers. Our goal is to understand the native opportunities on each cloud platform, the environment to deliver hybrid networks and environments, as well as how each of the leading open source and proprietary API management providers operate within a cloud environment.

Reusable Models - One key to successful API management in any environment is the establishment of reusable models, and the reuse of definitions, schema, and templates which can be applied consistently across groups. Any API gateway and management solution should have an accompanying database, machine, or container image, and be driven using a machine-readable API definition using OpenAPI, API Blueprint, or RAML. All data should possess an accompanying definition using JSON Schema, and leverage standards like JSON Path for mapping relationships, and establishing mappings between resources. Everything should be defined for implementation, reuse, and continuous deployment as well as integration.

API-Driven - Every mature API management platform automates using APIs. Mulesoft, Apigee, 3Scale, AWS Gateway, all have APIs for managing the management of APIs. This allows for the seamless integration across all systems, even disparate backend legacy systems. Extending identity, service compositions, plans, rate limits, logging, analysis, and other exhaust from API management into any system it is needed. Lighthouse should be an API driven platform for defining, designing, deploying, and managing APIs the serve the VA.

Additionally, as part of Team Agile Six, 540 has helped organizations within the Department of Defense (DoD) develop and execute their API strategies as part of their digital transformation to break down data silos and better enable data sharing. The Defense Acquisition Visibility Environment (DAVE) is one example where APIs (including Apigee, as the API Management platform) were implemented to improve integration efforts with multiple existing legacy systems and service systems (Army, Navy, and Air Force). This helped unlock our government client's (ARA/EI) requirement to facilitate free data exchange to better enable the desired analytical platform (without disrupting existing functionality such as the statutory Congressional reporting requirements).

Apigee, as the selected API Management platform, was a critical architectural component to enable DAVE's success. From the outset, Apigee provided out-of-the-box capabilities that our development teams would have otherwise needed to develop. This included OAuth2 (for integrator authentication), Quota and Rate-Limiting policies, SOAP and other mediation policies, security policies, and a powerful analytics engine. With several SOAP-based services available from legacy systems, the ability to transform those interfaces to RESTful ones using Apigee (configuration over code) provided more time to upgrade outdated backend implementations. Apigee's Usergrid-based Backend-as-a-Service (BaaS) also provided API-enabled data storage for front-end applications helping speed deployment and user adoption.

Apigee's production deployment for DAVE is an on-premise deployment in DISA's milCloud (with a future SIPRNet on-premise deployment planned). Additionally, Apigee's Cloud SaaS offering was leveraged in some non-production environments. Though demands on DAVE's API Management Platform are less than the VA's expectation of 56,000 requests per second, DAVE was designed to have horizontally scalable zones to easily adjust to traffic spikes and large-scale adoption. For Apigee, several installation topologies are available to cater to the varying traffic requirements (including hundreds of thousands of requests per second and above).

Question 4: Describe your company's specific experience and the strategies that you have employed for ensuring the highest level of availability and responsiveness of the platform (include information about configuration based capabilities such as multi-region deployments, dynamic routing, point of presence, flexible/multi-level caching, flexible scaling, traffic throttling etc.).

Our approach to delivering API infrastructure involves assessing scalability at every level of API management within our control. When it comes to API deployment and management we aren't always in control over every layer of the stack, but we always work to configure, optimize, and scale whenever we possibly can. Every API will be different, and a core team will enjoy a different amount of control over backends, but we always consider the full architectural stack when ensuring availability and responsiveness across API resources, looking at eleven common layers and considerations:

- **Network -** When possible we work to configure the network to allow for prioritization of traffic, and isolation of resources at the network level.
- **System** - We will work with backend system owners to understand what their strengths and weaknesses are, and understand how systems are currently optimized and assess what new ways are possible as part of Lighthouse API operations.
- **Database** - When there is access to the database level we will assess the scalability of instances and tables in service of API deployment. If possible we will work with database groups to deploy slave implementations that can be dedicated to supporting API implementations.
- **Serverless** - Increasingly we are using serverless technology to help carry the load for backend systems, adding another layer for optimization behind the gateway. In some

situations, a serverless layer can act as a proxy, cache, and redundancy for backend systems.

- **Gateway** - At the gateway level there are opportunities for scaling, and delivering performance, as well caching, and rate limiting to optimize specific types of behaviors amongst consumers.
- **DNS** - DNS is another layer which will play a significant role in the reliability and performance of API operations. There are numerous opportunities for routing, caching, and multi-deployment optimizations at the DNS to support API operations.
- **Caching** - There are multiple levels to think about caching in API infrastructure, from the backend up to DNS. The entire stack should be considered on an API-by-API basis, with a robust approach to knowing when to use, and where not to use.
- **Regional** - One of the benefits of being multi-vendor, and multi-cloud, and on-premise, is the ability to deliver API infrastructure where it is needed. Delivering in multiple geographic regions is an increasingly common reason for using the cloud, as well as allowing for flexibility in the deployment, management, and testing of APIs in any US region.
- **Monitoring** - One aspect of availability and reliability is monitoring infrastructure 24x7, keeping an eye on availability and performance. Sustained monitoring, across providers, and regions is essential to understanding how to ensure APIs are available and dependable.
- **Analysis** - Extracted from monitoring, and logging, the analysis across API operations feeds into the overall availability and reliability of APIs. Real-time analysis of API availability and performance over time is critical for dependable infrastructure.
- **Partners** - We've experienced the lack of dependability of VA APIs first hand. During his time at the agency Mr. Lane was forced to shut down APIs he had worked on during the government shutdown. We feel external partners can help contribute to the redundancy and availability of APIs beyond what the agency can deliver on its own.

Question 5: The experiences you have and the strategies that you have employed to ensure the highest level of security for the platform. Please address policy / procedure level capabilities, capabilities that ensure security of data in transit (e.g. endpoint as well as payload), proactive threat detection etc.

API security is another dimension of the API sector Mr. Lane has researched. He has identified 20 separate areas to consider as we develop a strategy to deliver the high levels of security:

- **Encryption** - Encryption in transport, on the disk, in storage, and in database.
- **API Application Keys** - Requiring keys for ALL API access, whether internal or external.
- **Identity & Access Management (IAM)** - Establish roles, groups, users, policies and other IAM building blocks, healthy practices and guidance across API implementations and teams.

- **Service Composition** - Provide common definitions for how to organize APIs, establish and monitor API plans and usage, and use to secure and measure access to ALL API resources.
- **Common Threats** - Be vigilant for the most common threat from SQL injection to DDoS, making sure all APIs don't fall victim to the usual security suspects.
- **Incident Response** - Establish an incident response team, and protocol, making sure when a security incident occurs there is a standard response.
- **Governance** - Bake security into API governance, beyond just API design, but actually connecting API security to the service level agreements governing platform operations.
- **DNS** - Leverage the front line of API security and identifying threats at the DNS layer, and establish, and maintain a frontline of defense for API security.
- **Firewall** - Building on top of existing web security practices, and deploying, and maintaining a comprehensive set of rules at a firewall level.
- **Automation** - Implement security scanning, fuzzing, and automated approaches to crawling networks and infrastructure looking for security vulnerabilities.
- **Testing & Monitoring** - Dovetail API security with API testing and monitoring, identifying malformed API requests, responses, and other illnesses that can plague API operations.
- **Isolation** - Leveraging virtualization, serverless, and other API design and infrastructure patterns to keep API resources isolated, and minimizing any damage that can occur in a breach.
- **Orchestration** - Acknowledging the security risks that come with modern approaches to orchestrating API operations, continuously deploying, and integrating across the platform.
- **Client Automation** - Develop a strategy for managing API consumption, and identifying, understanding, and properly managing bots, and other types of client automation.
- **Machine Learning** - Leveraging the development and evolution of machine learning models that help look through log files, and other threat information sources, and using artificial intelligence to respond and address evolving security concerns.
- **Threat Information** - Subscribing to external sources of threat information, and working with external organizations to open the learning opportunities across platforms regarding the threats they face.
- **Metrics & Analysis** - Tap into the metrics and analysis that comes with API management, and build upon the awareness introduced when you consistently manage API consumption.
- **Bug Bounties** - Defining and executing public and private bug bounties to help identify security vulnerabilities early on.
- **Communications** - Establish a healthy approach to communicating around platform security. Openly discussing healthy practices, speaking at conferences, and participating

in external working groups. While have a regular cadence to communication in normal times, as well as protocol for communication during security incidents.

- **Education** - Develop curriculum and other training materials around platform security, and make sure API consumers, partners, and internal groups are all getting regular doses of API security education.

Our API security experience comes from past projects, working with vendors, and studying the best practices of API leaders. Increasingly API security is a group effort, with a growing number of opportunities to work with security organizations, and major tech platforms who see majority of the threats present today.

Question 6: Please describe your experience with all the capabilities that the platform offers and the way you have employed them to leverage existing enterprise digital assets (e.g. other integration service buses, REST APIs, SOAP services, databases, libraries)

Our team brings extensive experience in database design, data warehousing, web-based applications, REST APIs & SOAP Services, here are some of our current capabilities:

- **SOAP** - We can support web services, and established approaches to accessing data and other resources.
- **REST** - Web APIs that follow RESTful patterns is our primary focus, leveraging low-cost web infrastructure to securely making resources available.
- **Hypermedia** - Understanding the benefits delivered by adopting common hypermedia media types, delivering more experience focused APIs that emphasize relationships between resources.
- **GraphQL** - Understanding when the use of GraphQL makes sense for some data focused APIs, delivering resources to single page and mobile applications.
- **gRPC** - Staying in tune with being able to deliver higher performance APIs using gRPC, augmenting the features brought by web APIs, for use internally, and with trusted partners.
- **Websockets** - Leverage websockets for streaming of data in some situations, providing more real time integration for applications.
- **Webhooks** - Developing webhook infrastructure that makes APIs a two-way street, pushing notifications and data externally based upon events or on a schedule.
  Event Sourcing - Developing a sophisticated view of the API landscape based upon events, and identifying, orchestrating and incentivizing event-based behavior.

This reflects the core capabilities present across the API landscape. While SOAP and REST make up much of the landscape, hypermedia, GraphQL, event sourcing, and other approaches are seeing more adoption. We emphasize that every platform make REST, or web APIs the

central focus of the platform, keeping the bar low for API consumers, leverage web technology to reach the widest audience possible.

Question 7: Please describe your experience and strategies that you have employed at enterprises to create Experience Services from mashup/aggregation/combination of other API's, services, database calls etc.

Our team members have worked on API aggregation as a discipline for several years now, having worked with several groups to aggregate common APIs. It is an underdeveloped layer to the web API sector, but one that has a number of proprietary, as well as open source solutions available. We've worked on a number of API aggregation project, spanning a handful of business sectors:

- **Social** - Developed an open source, SaaS solution for aggregating Facebook, LinkedIn, Twitter, and other social networks.
- **Images** - Extended systems to work across image APIs, working with Flickr, Facebook, Instagram, and other image sharing solutions.
- **Wearables** - Developed a strategy for aggregating of health data across a handful of the leading wearable device providers.
- **Documents** - Partnered with a company who provides document aggregation across Box, Google Drive, and other leading document sharing platforms.
- **Real Estate** - Founded a startup that did MLS data aggregation across APIs, FTP locations, and scraped targets, providing regional, and specific zip code API solutions.

Here are some of the technology, and approaches we have been using to deliver on API aggregation:

- **OpenAPI** – We've heavily used the OpenAPI specification to aggregate API definitions, and leverage JSON Schema to make connections and map API and data resources together.
- **APIs.json** – Mr. Lane worked on the data.json project with the White House, inventorying open data inventory across federal agencies. He developed an open specification for indexing APIs, and building collections, and other aggregate definitions for processing at discovery or runtime.
- **JSON Schema** - All databases and data used as part of API operations possesses a JSON schema definition that accompanies the OpenAPI that defines access to an API. JSON Schema provides the ability to define references across and between individual schemas.
- **JSON Path -** XPath for JSON, enabling the ability to analyze, transform and selectively extract data from API requests and responses.

Question 8: Please describe your experience and strategies for Lifecycle Management of APIs for increasing productivity and enabling VA to deliver high value APIs rapidly, onboarding app developers and commercial off the shelf applications.

Our experiences in fulfilling an API lifecycle from the consumer perspective always begins with a central portal, but one that possesses all the required API management building blocks to not just deliver APIs rapidly, incentivize consumption, and feedback in a way that evolves them throughout their life cycle:

- **Portal** - Have a central location to discover and work with all API resources, even if there is a federated network of portals that work together in concert across groups.
- **Getting Started** - Providing clear, frictionless on-boarding for all API consumers in a consistent way across all API resources.
- **Authentication** - Simple, pragmatic authentication with APIs, that are part of a larger IAM scaffolding.
- **Documentation** – Up-to-date, API definition driven, interactive API documentation that demonstrates what an API does.
- **Code Resources** - Code samples, libraries, and SDKs that make integration painless, and frictionless, allow applications to quickly move from development to production.
- **Communications** - A regular drumbeat around API Platform communications, helping API consumers navigate discovery to integration, and keeps them informed regarding what is happening across the platform.
- **Road Map** - A clear roadmap, as we as resulting change log that keeps API consumers in tune with what's next, reducing the gap that can exist between API provider and consumer.
- **Applications** - A robust directory of certified applications, browser, platform, and 3rd party plugins and connectors, demonstrating what is possible via the platform.
- **Analysis** - Logging, measuring, and analyzing API platform traffic, sign ups, logins, and other key data points, developing an awareness of what is working when it comes to API consumption, and quickly identifying where the friction is, and eliminating with future releases.

Developer Experience (DX) is something that significantly speeds up the overall API lifecycle. Backend teams can efficiently define, deliver, and evolve their APIs without API consumers on-boarding, integrating, and providing feedback on what works, and what doesn't work. A central API portal strategy for Lighthouse API management is key to facilitating movement along the API lifecycle, reducing friction, eliminating roadblocks, and reducing the chance an API will never full be realized in production.

Question 9: Please describe your experience and strategies for establishing effective governance of the APIs.

Sadly, API governance is not something we have seen consistently applied across the enterprise, at institutions and government agencies. There is no standard for API governance. There are few case studies when it comes to API governance to learn from. Slowly we are seeing larger

enterprises share their strategies, and seeing some universities publish papers on the topic. By providing some common building blocks, we can organize into a coherent API governance strategy. These are the areas we are working on:

- **Design Guide** - Establishing API design guide(s) for use across organizations.
- **Deployment** - Ensuring there are machine-readable definitions for every deployment pattern, and they are widely shared and implemented.
- **Management** - Quantify what API management services, tooling, and best practices groups should be following, and putting to work in their API operations.
- **Testing** - Defining the API testing surface area, and provide machine readable definitions for all test patterns, while leverage OpenAPI and other definitions as a map of the landscape.
- **Communication** - Plan out the communication strategy as part of the API governance strategy.
- **Discovery** - Establish tools and processes for identifying and leveraging existing resources.
- **Support** - What support mechanisms are in place for governance, as well as definitions for best practices for supporting individual API implementations.
- **Deprecation** - Define what deprecation looks like even before an API is defined or ever delivered, establishing best practices for deprecation of all resources.
- **Analysis** - Measuring, analyzing, and reporting on API governance. Identifying where it is being applied effectively, and where the uncharted territory lies.
- **Coaches** - Train and deploy API coaches who work with every group on establishing, evolving, incentivizing, and enforcing API governance across groups.
- **Rating** - Establishing a set of rating system for quantifying how compliant APIs are when it comes to API governance, providing an easy to understand how close an API is, or isn't.
- **Training -** The development and execution of training around API governance, working with all external groups to help define API governance, and take active role in its implementation.

Question 10: Please describe your experience and methodologies for DevOps/RunOps processes across deployments. Highlight strategies for policy testing, versioning, debugging etc.

API management is a living entity, and we focus on delivering API operations with flat teams who have access to the entire stack, from backend to understanding application and consumer needs. All aspects of the API life cycle embraces a microservices, continuous evolution pace, with Github playing a central role from define to deprecation:
CI/CD - Shared repositories across all definition, code, documentation, and other modules that make up the lego pieces of API operations.
Testing & Monitoring - Ensuring every building block has a machine readable set of assertions,

tests, and other artifacts that can be used to verify operational integrity.
Microservices - Distilling down everything to a service that will meet a platform objective, and benefit an API consumer in as small as package as possible.
Versioning - Following a semantic versioning with version numbers reflecting a MAJOR.MINOR.PATCH, and kept in sync across backend, API, and client applications.
Dependencies - Scanning and assessing for vulnerabilities in libraries, 3rd party APIs, and other ways dependencies are established across architecture.
Security - Scanning and assessing security risk introduced by dependencies, and across code, ensuring that testing and monitoring reflects wider API security practices.

A DevOps focus is maintained across as many stops along an API life cycle, and reflected in API governance practices. However, it is also recognized that a DevOps will not always be compatible with existing legacy practices, and custom approaches might be necessary to maintain specific backend resources, until their practices can be evolved, and brought in alignment with wider practices.

Question 11: Please describe your experience and strategies employed for analytics related to runtime management, performance monitoring, usage tracking, trend analysis.

Logging and analysis is a fundamental component of API management, feeding an overall awareness of how API are being consumed, which contributes to the product road map, security, and overall platform reliability. The entire API stack should be analyzed from the backend, to the furthest application endpoints, whenever possible:

- **Network** - Logging, and analysis at the packet and network level, understanding where the network bottlenecks are.
- **Instance** - Monitoring and measuring any server instance, whether it is bare metal, virtual, or containerized, providing a complete picture of how each instance in the API chain is performing.
- **Database** - When access to the database layer is present, adding the logging, and other diagnostic information to the analysis stack, understand how backend databases are doing their job.
- **Serverless** - Understanding the performance and usage of each function in the API supply chain, making sure each building block is accounted for.
- **Regional** - Understanding how API resources deploy in various regions are performing, but also adding the extra dimension of measuring and monitoring APIs from a variety of geographic regions.
- **Management** - Leveraging API plans at the API gateway, and understanding API consumption at the user, application, and individual resource level.
- **Client** - Introducing logging, analysis, and tracking of errors at the SDK level, developing usage and performance reports from the production client vantage point.

API management has matured over the last decade to give us a standard approach to managing API access at consumption time, optimizing usage, limiting bad behavior, and incentivizing healthy behavior. API monitoring and testing practices have evolved this perspective of the health, availability, and how APIs are being used from the client perspective. This information gets funneled into the road map, refining API gateway plans, rate limits, and other run time systems to adjust and support desired API usage.

Question 12: Please describe your experience with integrating MuleSoft with Enterprise Identity Management, Directory Services and implementing Role Based access.

We do not have any experience in this area with Mulesoft specifically, however we have partners that do and we have worked in general with IAM and directory solutions in other platforms, including heavy usage on AWS, for securing the API gateway's interaction with existing backend system.

Question 13: Please describe your experience with generating and documenting API's and the type of standards they follow. Describe approaches taken for hosting these documentations and keeping them evergreen.

API documentation is always a default aspect of the API platform, and delivered just like code as part of DevOps and CI/CD workflows using Github. With almost every stop along the API life cycle, all documentation is API definition driven, keeping things interactive, with the following elements always in play:

- **Definitions** - All documentation is driven using OpenAPI, API Blueprint, or RAML, acting as the central machine-readable truth of what an API does, and what schema it uses. API definition driven API documentation contributes to them always being up to date, and reflect the freshest view of any API.
- **Interactive** - All API documentation is API driven, allowing for the documentation to be interactive, acting as an explorer, and dynamic dashboard for playing with and understanding what an API does.
- **Modular** - Following a microservices approach, all APIs should have small, modular, API definitions, that drive modular, meaningful, and useful API documentation.
- **Visualizations** - Evolving on top of interactive documentation features, we are beginning to weave in more visual, dashboard, and reporting features directly into API document.
- **Github** - Github plays a central role in the platform life cycle, with all API definitions and documentation running within Github repository, and integrated as part of CI/CD workflows, and DevOps frame of mind.

All platform documentation is a living, breathing, element of the ecosystem. If should be versioned, evolved, and deployed along with other supporting microservice artifacts. Mulesoft

has documentation to support this approach, as well as there being a suite of open source solutions we can consider for supporting a variety of different types of APIs.

Question 14: Please describe your proposed complete process lifecycle for publishing high quality, high value APIs with highest speed to market.

Evolving beyond question #8, and addressing the API provider side of the coin, we wanted to share a complete (enough) view of the life cycle from the API provider perspective. Addressing the needs of backend API teams, as well as the core API team when it comes to delivering usable, reliable, APIs that are active, and enjoy health release cycles. While not entirely a linear approach, here are many of the stops along our proposed API lifecycle, as applied to individual APIs, but applied consistently across the entire platform:

- **Definitions** - Every API starts as an API definition, which follows the API throughout its life, and drives every other stop along the way.
- **Design** - Planning, applying, evolving, and measuring how API design is applied, taking the definition, and delivering as a consistent API that complies with API governance.
- **Virtualization** - Delivery of mock, sandbox, and other virtualized instances of an API allowing for prototyping, testing, and playing around with an API in early stages, or throughout its life
- **Deployment** - Deploying APIs using a common set of deployment patterns that drives API gateway behavior, connectivity with backend systems, IAM, and other consistent elements of how APIs will be delivered.
- **Management** - The employment of common approaches to API management, some of which will be delivered and enforced by the API gateway, but also through other training, and sharing of best practices made available via API governance.
- **Testing & Monitoring** - Following DevOps, CI/CD workflows when it comes to developing tests, and other artifacts that can be used to monitor, test, and assert that APIs are meeting service level agreements, and in compliance with governance.
- **Security** - Considering security at every stop along the API life cycle, scanning, testing, and baking in security best practices all along the way.
- **Client & SDK** - Establish common approaches to generating, delivering, and deploying client solutions that help API consumer get up and running using an API.
- **Discovery** - Extending machine-readable definitions to ensure an API is indexed and discoverable via service directories, documentation, and other catalogs of platform resources.
- **Support** - Establishing POC, and support channels that are available as part of any single APIs operations, ensuring it has required support that meets the minimum bar for service operations.
- **Communications** - Identifying any additions to an overall communication strategy around an APIs life cycle, announcing it exists on the blog, changes and releases are on the road map, and showcasing integrations via the platform Twitter account.

- **Road Map** - Making sure there are road map, and change log entries for an API showing where it is going, and where it has been.
- **Deprecatio**n - Lay out the deprecation strategy for an API as soon as it is born, setting expectations regarding when an API will eventually go away.
  This life cycle will play out over and over for each API published on the Lighthouse API platform. It will independently be executed by API teams for each API they produce, and replayed with each major and minor release of an API.

Question 15: Please describe your experience and approach towards establishing a 24x7 technical support team for the users, developers and other stakeholders of the platform.

Support is an essential building block of any successful API platform, as well as a default aspect of every single APIs individual life cycle. We break API platform support into two separate distinct categories.

API support should begin with self-service support, encouraging self-sufficiency of API consumers, and minimizing the amount of platform resources needed to support operations:

- **FAQ** - Publishing a list of frequently asked questions as part of getting started and API documentation, keeping it an easy to access, up to date list of the most common questions API consumers face.
- **Knowledge Base** - Publishing of a help directory or knowledge base of content that can help API consumers understand a platform, providing access to high level concepts, as well as API specific functionality, errors, and other aspects of integration.
- **Forum** - Operate an active, moderated, and inclusive forum to assist with support, providing self-service answers to API consumers, which also allows them to asynchronously get access to answers to their questions.

Beyond self-service support all API platforms should have multiple direct support channels available to API consumers:

- **Monitoring** - Providing a live, third-party status page that shows monitoring status across the platform, and individual APIs, including historical data.
- **Email** - Allow API consumers to email someone and receive support.
  Tickets - Leveraging a ticketing system, such as Zendesk, or other SaaS or possibly open source solution, allowing API consumers to submit private tickets, and move each ticket through a support process.
- **Issues** - Leveraging Github issues for supporting each component of the API lifecycle, from API definition, to API code, documentation, and every other stop. Providing relevant issues and conversation going on around each of the moving parts.
- **SMS** - SMS notifications regarding platform events, support tickets, platform monitoring, and other relevant areas of platform operations.

- **Twitter** - Providing public support via a Twitter account, responding to all comments, and routing them to the proper support channel for further engagement.

A combination of self-service and direct support channels allows for a resource starved core API team, as well as individual backend teams to manage many developers and applications, across many API resources in an efficient manner.

Question 16: Please describe your experience in establishing metrics and measures for tracking the overall value and performance of the platform.

Understanding performance and reliability through the entire stack is critical to platform reliability, but the API management core is all about developing an awareness of how APIs are being consumed, and the value generated as part of this consumption. While performance definitely impacts value, we focus on API management to help us measure and understand consumption of each resource, and understand and measure the value delivered in consumption.

We've been studying how API management establishes the metrics needed, and measures and tracks API consumption, trying to understand value generation using the following elements:

- **Application Keys** - Every API call possess a unique key identifying the application and user behind consumption. This key is essential to understanding how value is being generated in the context of each API resources, as well as across groups of resources.
- **Active Users** - New, no name, automated API users are easy to attack. It is the active, identifiable, communicative users we are shooting for, so quantifying and measuring what an active user is, and how many exist on the platform is essential to understanding value generation.
- **Active APIs** - Which APIs are the APIs that consumers are integrating with, and using on a regular basis? Not all APIs will be a hit, and not all APIs will be transformative, but there can be utility, and functional APIs that make a significant impact.
- **Consumption** - Measuring the nuance of how API consumers are consuming APIs, beyond just the number of calls being made. What is being consumed? What is frequency of consumption? What are the limitations, constraints, and metrics for consumption, and incentives for increasing consumption?
- **Plans** - Establishing many different access tiers and plans, containing many different APIs, allowing for API service composition to evolve create different API packages, or products that deliver across many different web, mobile, and device channels.
- **Design Patterns** - Which design patterns are being used, and driving consumption and value creation. Are certain types of APIs favored, or specific types of parameters favored and backed up by API consumption numbers. Measuring successful API design, deployment, management, and testing patterns, and identifying the not so successful patterns, then incorporating these findings into the road map as well as platform governance.

- **Applications** - Considering how metrics can be applied at the client, and SDK level, providing visibility and insight into how APIs are being consumed, from the perspective of each type of actual application being used.

Question 17: Please describe your experience and the approach you would take as the API Program Core Team to deploy an effective strategy that will allow VA to distribute the development of API's across multiple teams and multiple contractor groups while reducing friction, risk, and time to market.

A microservice, CI/CD, DevOps approach to providing and consuming APIs has begun to shift the landscape for how API platforms are supporting the API life cycle across many disparate teams, trusted external partners, and even 3rd party application and system developers. Distilling all elements of the API supply chain to modular, well defined components, helps establish a platform that can be centralized, but encouraging a consistent way to approach the federated delivery of APIs and supporting resources.

We focus on making API operations more reusable, agile, and effective in a handful of areas:

- **Github** - Definitions, code, documentation, training, and all other resources exist as repositories allowing them to be versioned, forked, collaborated around, and delivered wherever they are needed.
- **Governance** - Establishing API design, deployment, management, testing, monitoring, and other governance practices, then making them available across teams as living, collaborative documents and practices, and allow teams to remain independent, but working together in concert at scale, using a common set of health practices.
- **Training** - Establishing a training aspect to platform operations, making sure materials is readily developed to support all stops along the API lifecycle, and the curriculum is circulated, evolved, and made available across teams.
- **Coaching** - Cultivate a team of API coaches who are embedded across teams, but also report back and forth with the central API team, helping train and educate around platform governance, but help work with and improve a team's ability to deliver within this framework.
- **Modular** - Keeping everything small, modular, and limiting the time and budget needed to define, design, deploy, manage, version, and deprecate, allowing the entire platform to move forward in concert, using smaller release cycles.
- **Continuous** - API governance, training, coaching, and all other operational level support across teams is also continuously being deployed, integrated, and evolved, ensuring that teams can work independently, but also be in sync when possible. Reducing bottlenecks, eliminating roadblocks, and minimizing friction between teams, and across APIs that originate from many different systems.

API operations needs to work much like each individual API. It needs to be small, modular, well defined, doing one thing well, but in a way that can be harnessed at scale to deliver on platform objectives. The API lifecycle needs to be well documented, with disparate teams well educated

regarding what healthy API design, deployment, management, and testing looks like.

Question 18: Please describe the roles and skill sets that you would be assembling to establish the API Program core team.

Our suggested team for the Lighthouse Core API management implementation reflects how a federated, yet centralized API platform team should work. There are a handful of disparate groups, each bringing a different set of skills to the table. We are always working to augment and grow this to meet each project we work on, and open to working with 3rd party developers. Bringing forward some skills we already have on staff, while also introducing some new skills to bring a well-rounded set of voices to the table to support the Lighthouse API platform:

- **Architects** - Backend, platform, and distributed system architects.
- **Database** - Variety of database administration, analysts, and scientists.
- **Veterans** - Making sure the veteran perspective is represented on the platform team.
- **Healthcare** - Opening the door to some healthcare practitioner advisor roles, making sure a diverse group of end users are represented.
- **Designers** - Bringing forward the required API design talent required to deliver consistent APIs at web scale.
- **Analysts** - It is critical to understand the domain(s) and workflows being served by an API, and help communicate the needs of Veterans to the engineers who will realize the solutions.
- **Developers** - Provide a diverse set of developer skills supporting a variety of programming languages and platforms.
- **Security** - Ensure there is necessary security operations skills to pay attention to the bigger security picture.
- **Testing** - Provide dedicated testing and quality assurance talent to round off the set of skills on the table.
- **Evangelists** - Make sure there is an appropriate amount of outreach, communication, and storytelling occurring.

As the number of APIs grow, and an awareness of what consumers are desiring additional team members will be sought to address specific platform needs around emerging areas like machine learning, streaming and real time, or other aspects we haven't addressed in this RFI.

Question 19: How would you recommend the Government manage performance through Service Level Agreements? Do you have commercial SLAs established that can be leveraged?

Each stop along the provider dimensions, as well as the consumer perspective will be quantified, logged, measured, and analyzed as part of API management practices. These are some of the areas that will be quantified, and measured, for inclusion in service level agreements:

- **Database** - What are the benchmarks for the database, and what are the acceptable ranges for operation. Log, measure, and evaluate historically, establishing a common scoring for delivery of database resources.
- **Server** - Similar to the database layer. What are the units of compute available for API deployment and management, and what are the benchmarks for measuring the performance, availability, and reliability of each instance, no matter how big or small.
- **Serverless** - Establishing benchmarks for individual functions, and defining appropriate scaling, service levels for each script beyond each API.
- **Gateway** - Through gateway logging, measuring, and understanding performance and availability at the gateway level. Factoring in caching, rate limits, as well as backend dependencies, then establishing a level of service that can / should be met.
- **APIs** - Through monitoring and testing, each API should have its tests, assertions, and overall level of service defined. Is the API directly meeting its service level agreement? Is an API meeting API governance requirements, as well as meeting its SLA? What is the relationship between performance and compliance?

Each stop along the API lifecycle should have its guidance when it comes to governance, and possess a set of data points for measuring monitoring, testing, and performance. All of this can be used to establish a comprehensive service level agreement definition that goes beyond just uptime and availability, ensuring that APIs are meeting the SLA, but are doing so through API governance practices that are working.

Question 20: If the Government required offerors to build APIs as part of a technical evaluation, what test environments and tools would be required?

We'd be willing to build APIs in support of a technical evaluation. We are willing to do this using an existing API gateway solution, via any of the existing cloud platforms. We would need access to deploy mock and virtualized APIs using a gateway environment, as well as configure and work with IAM, and some sort of DNS layer, to access addressing and routing for all prototypes deployed.

Our team is capable of working in any environment, and suggest test environments and tool reflect existing operations, supporting what is already in motion. We can make recommendations, and bring our open cloud, SaaS, and open tooling to the table, and build a case for why another tool might be needed, and make more sense as part of the technical evaluation.

To facilitate the defining, and evolution of a technical evaluation we recommend starting a private Github repository, either initiated by the VA, or off error, where a set of requirements, and definitions can be established as a seed for the technical evaluation, as well as quickly become many seeds for actual platform implementation.