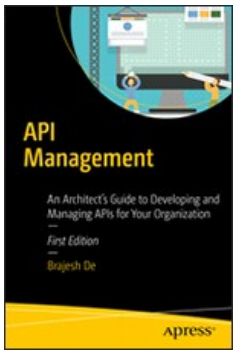

API Management: An Architect's Guide to Developing and Managing APIs for Your Organization

by Brajesh De

Chapter 12: API Governance

B. De, *API Management*, DOI 10.1007/978-1-4842-1305-6_12

Overview

API governance is distinct from SOA governance. API governance provides a policy-driven approach that helps to enforce standards and checkpoints throughout the API lifecycle. It encompasses not only the API runtime, but also design through development processes. It includes the guidelines, standards, and processes to be followed for API identification, interface documentation, development, testing, deployment, run, and operation. Standards and principles defined by API governance provide API quality assurance, such as security, availability, scalability, and reliability. It underpins the API enablement aspect that is critical for the successful adoption of APIs.

The Scope of API Governance

API governance encompasses activities, starting with the API proposal all the way to its adoption, through requirements gathering, build and deploy, and operations during general availability. Figure 12-1 shows the high-level phases where API governance plays a critical role.

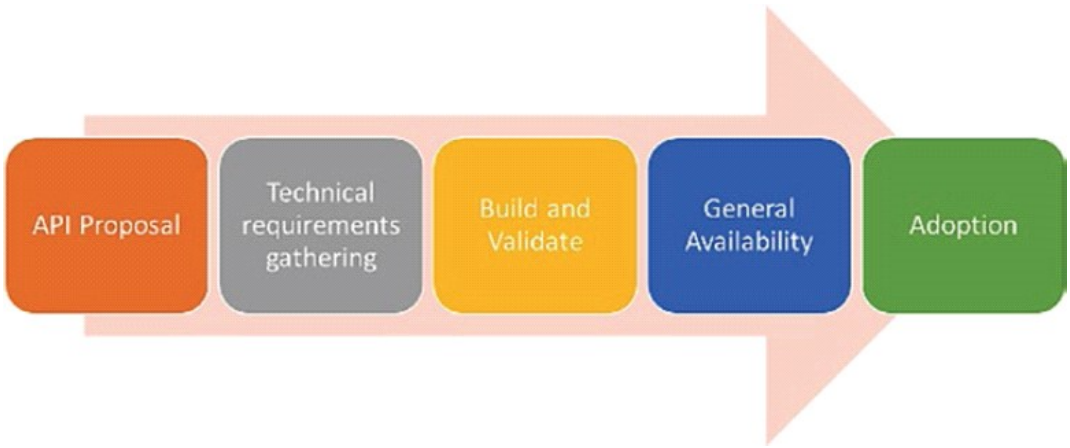

Figure 12-1: API Governance Phases

The following describes the phases.

- **API Proposal:** This is the first stage, where new API or change requests are proposed by the organization. This is done due to new business agreements, changes in existing business agreements, or a new change request submitted to the API governance body. Community managers create an ecosystem: talk to partners, competitors, regulators, and independent developers to identify and propose APIs that are aligned to the business strategy
- **Technical requirements gathering:** After the API proposal, the next step is to gather the requirements and create the specifications for the API. API architects and business analysts work together to create API definitions. The API governance defines the process and standards for the API interface definition. The following are some important questions that API governance must consider and enforce:
  - Which API specifications standards should be used? Swagger or RAML, or any other standards for API interface documentation?
  - Who is responsible for the review and approval of the API specifications?
  - What is the API versioning approach? When is a new version created?
  - Is there JSON schema versioning?
  - Which back-end services are connected to these APIs? Are there field mappings?
- **Build and validate:** After the API specification and requirements are finalized, the development process starts. The scrum master, API team, and all the members in the API program work together in this phase. Test scripts are created and APIs are validated for compliance to API specification. API governance during the build and validate phase must define guidelines for the following questions:
  - What tools are to be used for the entire API development lifecycle?
  - Which source code repository must be used for configuration management?
  - Which best practices must be followed for API development?
  - What should be the testing approach and the tools to be used for API interface, functional and load testing?

- What is the review process? What are the checkpoints to ensure the quality of APIs?
- Which policies must be implemented for APIs?
- Is there isolation between the non-production and production environments?
- How should the API interface lifecycle be managed?
- What is the promotion process, from the lowest development environment to production, and eventually to the retirement of the APIs?

- **General availability:** After the implementation is done and the APIs are deployed, they need to be published to the developer portal for API subscribers. Before publishing an API, consider commercial questions, such as how to monetize the API. Since APIs may expose data to the consumers, the terms and conditions for the use of the APIs and the associated data should be finalized with the legal team. The marketing team should review and ensure that brand use and quality are satisfactory. After approval from the commercial, legal, and marketing teams, the API can be deployed to the production server and released for beta or general availability. After an API has been made available for general use, there must be proper tracking and metrics to provide information that answers the following questions:
  - Which API is deployed to what environment?
  - What is the performance of the deployed API?
  - Which apps are using which API?
  - What are the usage patterns for the API by app, geography, and time?

  Note The API governance process must define all the API metrics to be tracked and how the tracking is done. It must define the necessary steps to be taken in case of any service-level agreement violations.

- **Adoption and sunsetting:** During this phase, developers start exploring and using APIs to build apps around them. API governance should facilitate the easy but secure signup and onboarding of developers and their apps. The governance process should monitor how the APIs are performing and being used by developers. Some important metrics to look for must answer the following:
  - What are the top 10 APIs being used?
  - Who are the top 10 users of an API?

  Note This step should cover mechanisms that address any issues reported by developers on API usage. When new versions of an API are introduced, the API governance process must address how to sunset and retire older versions with minimal to no impact on the apps still using them.

The Aim of API Governance

API governance must address the following.

- Governance at the time of the API proposal (new/updates) must ensure that the identified APIs align to the business strategy and meet the business requirements. Funding for API development must be approved by the business and other stakeholders.
- API design and development time governance must ensure that the API software quality is maintained. It must address API versioning strategy and focus on development standards and best practices to be followed. Appropriate reviews and checkpoints must be enforced to ensure quality of he API.
- API governance must help define the right API testing strategy to ensure that APIs are delivering the necessary level of security, reliability, and governance.
- API runtime governance must look at aspects such as API monitoring, deployment, and dynamic provisioning to guarantee API runtime quality.
- API governance must ensure that the service-level agreement is followed by the API provider and the consumer.

API Governance Model

The high-level API governance activities, checkpoints, preconditions, and the roles required for each of the phases of the API governance model are described in Tables 12-1 to 12-5.

Table 12-1: API Proposal

Title	Description
Input	▪ New API requests made by business analysts and solution architects. An outline solution document for the new API to be submitted for review. ▪ API change requests can be made by solution architects and an outline solution document for the API is submitted. ▪ Design lead, tech arch and solution architect review the submitted API proposal. Proposal should be approved by architecture review board. The approval process can be similar to an existing SDLC. A lighter version of the existing approval process can be followed for API proposals.

Process	For new APIs ▪ New API request/business agreement to be submitted for review. ▪ API details to be completed in a template that captures the requirements of the API. It should highlight the use cases related to the API. This evaluates the alignment of the API to business needs. The template must also highlight the information model and the related entities that used by the API. This explains the business assets that are API-enabled. The service interface definition of the API should also be documented at this stage. ▪ Governance review by architecture review board (ARB) of the new API proposal decides if the API should be built or not. For API change requests ▪ Business analyst/solution architects submit a change request to an existing API. ▪ API details to be completed in the same template as that of a new API. ▪ ARB governance review of the API change request decides if the API should be built or not.
Output	▪ *API profile template* document with API specification for new APIs. ▪ Updated API profile template for API change requests. ▪ A new project (for a CR there is a new version of the project) for the creation of API.
Checkpoint	▪ Fortnightly or monthly governance reviews organized to review new API requests or change requests. The frequency may change depending on the business needs for the APIs.
Exit Pre-Conditions	▪ Resourcing availability for API development (an API team to be formed). ▪ An API spec should be reviewed and approved by the ARB or API governance body. ▪ Funding for API development should be approved by business and other stakeholders.
Actors and Roles	▪ *API business owner*: Responsible for establishing and validating the business needs of the API and the requirements for approval of funding. ▪ *API product owner*: Responsible for interfacing with various API delivery teams to ensure the quality and delivery of the APIs. ▪ *API spec lead*: Responsible for the creation of API specification. ▪ *API architect*: Responsible for the technical architecture of the API solution. ▪ *API leadership team*: Responsible for validating the business requirements and providing funding to build the API.

Table 12-2: Technical Requirements Gathering

Title	Description
Input process	▪ API profile template ▪ Create API specifications document from the business requirements. ▪ Define data mappings between API interfaces and back-end services. ▪ Requirements should be stored and/or updated in acentral requirements management tool.
Output	▪ API specification and data mappings document
Checkpoint	▪ Review with the business analyst and the API architect, and sign off the API specification
Exit Pre Conditions	▪ Governance guidelines and rules followed ▪ API profile requirements are updated in JIRA
Cross-functional Implication	▪ API specification review for any impact on existing functionality
Actors and Roles	▪ *API business analyst*: Gathers the business requirements for API enablement and identifying the services to be exposed as APIs. ▪ *API solution architect*: Works with the business analyst to define the API specification document and data mapping to back-end services. ▪ *API spec lead*: Defines the API specifications and working with the business analyst and solution architect. ▪ *API project team*: Informed about the new API requirements at this stage. Reviews the API specifications. ▪ *API governance committee*: Ensures that the process is followed, criteria are met, and quality is maintained. ▪ *Scrum master*: Conducts a spec jam.

Table 12-3: Build and Validate

Title	Description

Input	▪ Approved API specification and data mapping documents ▪ Business requirements document
Process	In the API build and validate phase, the API team consists of scrum master, API architect, API designer, API developers, API testers, and DevOps team, who work together to build the API per specifications and business requirements. API development is done in short sprints of three to four weeks using the agile development methodology. The high-level activities are as follows: ▪ The *scrum master* grooms the requirements and fills in action log in a requirements management tool like JIRA. ▪ The *API development lead* reviews the API specifications, captures comments, and updates action logs. ▪ The *API teams* are responsible for the following: ○ Reviewing the specification, update business agreement (after revival) and update action logs ○ API implementation ○ Committing to SCM ○ Creating test kits ○ Code review ○ Demoing to the client and validation by the client ○ Publishing deployable artifacts to repository ○ Updating developer portal links ○ Publishing information on the developer portal for API subscribers
Output	▪ Completed action list captured during previous discussions ▪ Follow-up action plan created in action log ▪ Reference implementation running in development ▪ Artifacts uploaded to a repository, like GitHub ▪ Developer portal updated
Checkpoint	▪ Implementation should be compliant to API specification document (mappings are validated)
Exit Pre-Conditions	▪ Final review (config review, demo to the client) ▪ Governance guidelines and rules followed ▪ API conformant to design guidelines ▪ API versioning policies followed ▪ Any deviations are documented
Cross-functional Implication	▪ Review for any impact on existing functionality
Actors and Roles	▪ *API program manager*: Responsible for the overall program delivery of the APIs. ▪ *API architect*: Architects the API solution and defines the API REST interface. ▪ *API designer*: Designs the APIs for proxy configurations. ▪ *API developers*: Configures API proxies in the API gateway ▪ *API testers*: Creates automated test cases and testing API interfaces. ▪ *API DevOps*: Builds a DevOps framework to support CI and CD for API enablement

Table 12-4: General Availability

Title	Description
Input	▪ API interface definition in repository ▪ API test console availability ▪ Developer portal updated ▪ Pre-prod environment running reference implementation ▪ API config uploaded to SCM or repository
Process	During the general availability phase the following activities are performed to publish the APIs: ▪ API deployment from the repository to production and sandbox

	▪ API documentation published on the developer portal for API subscribers ▪ Developers access APIs and create apps ▪ API health monitoring is set up
Output	▪ APIs deployed to production and sandbox environments ▪ API documentation updated in the developer portal ▪ Apps built against APIs
Checkpoint	▪ Check API's running status ▪ Check API documentation and test console in the developer portal ▪ Check API analytics for API traffic and performance ▪ API health monitors are configured
Actors and Roles	▪ *Project team*: Responsible for overall API delivery. ▪ *API support team*: Supports reported issues. ▪ *Operations/run team*: Deploys APIs and monitors their health.

Table 12-5: Adoption

Title	Description
Input	▪ The number of developers signed up ▪ API traffic reports ▪ The number of hits on developer portal ▪ The number of mentions in social networks ▪ The number of blogs and forum posts
Process	▪ During the adoption phase of an API, it is important to have a plan that facilitates easy onboarding of developers and apps, and tracks the usage of the API. For this, the following activities need to be performed: ○ Develop an adoption plan and identify targets. ○ Target and inform specific development communities about the availability of the new API. ○ Target/organize hackathons to support adoption. ○ Track/follow up with members who are in the member adoption forum. ○ Update the adoption list, developer portal, and API website. ○ Ensure the publicity of API through various developer channels ○ Identify and inform other ecosystems of API availability. ○ Conduct webinars driven by members to share experience in adoption.
Checkpoint	▪ The number of developers and apps onboarded ▪ The number of active developers and apps ▪ API traffic patterns ▪ The number of API issues reported from different channels
Actors and Roles	▪ *API operations team*: Facilitates the developer onboarding and monitors API traffic. ▪ *API support team*: Resolves issues reported about the APIs