API Product Quality

A Comprehensive Reflection Guide

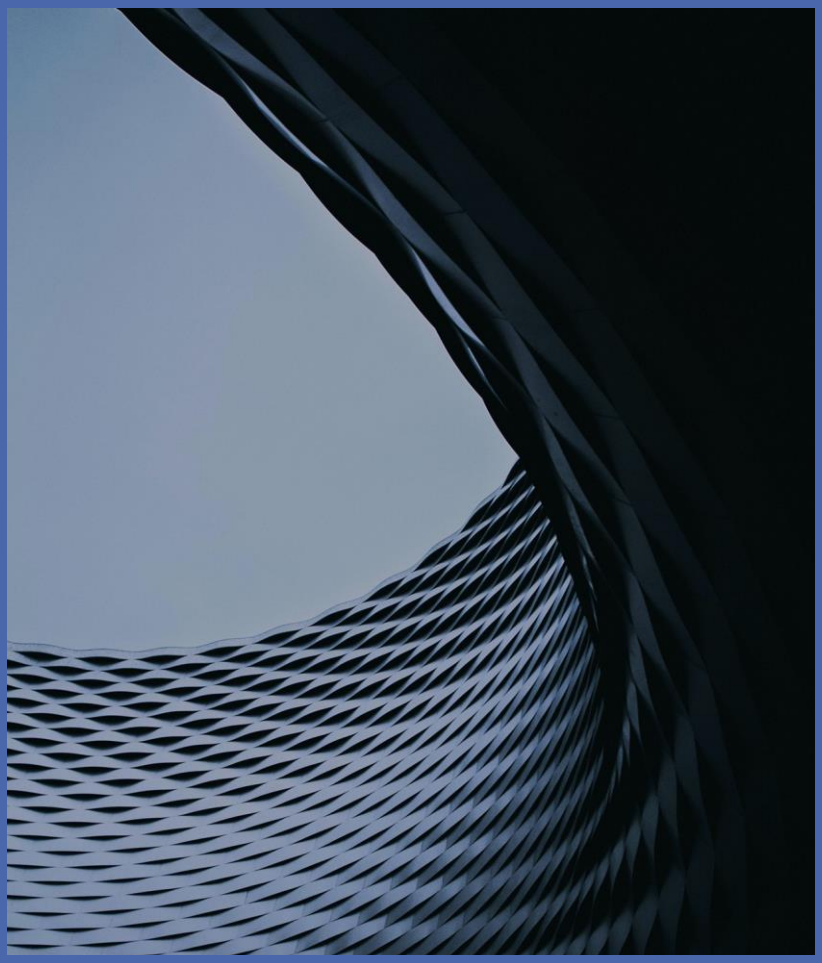
Eaton, Ben; Henningsson, Stefan; Hedman, Jonas; Schneider, Maximilian; Johansen, Benny Boye

Document Version
Final published version

DOI:
10.22439/DIGI.2024.01

Publication date:
2024

06 oktober 2024



API PRODUCT QUALITY

A COMPREHENSIVE REFLECTION GUIDE

Authors:

Ben Eaton (corresponding author)
be.digi@cbs.dk

Stefan Henningsson
sh.digi@cbs.dk

Jonas Hedman
jhe.digi@cbs.dk

Maximilian Schneider
maximilianschneider1
@outlook.com

In collaboration with:

Benny Boye Johansen
bjo@saxobank.com

CBS COPENHAGEN BUSINESS SCHOOL
HANDELSHØJSKOLEN

TABLE OF CONTENTS

INTRODUCTION: THE IMPORTANCE OF API PRODUCT QUALITY IN DIGITAL ECOSYSTEMS

In the realm of modern technology, Application Programming Interfaces (APIs) are becoming the cornerstone component of the digital ecosystem, resulting in the emergence of the API economy. This transformation is driven by firms leveraging APIs for cost reduction, broadening their audience reach, and monetizing services - enabling value creation through the exchange of information and services within and between organizations.

Following the proliferation of APIs, even traditional firms are evolving into software producers, and are now on a journey *from ad-hoc exploration* of APIs to establishing an organization-wide approach *to APIs at scale*. In this context, APIs increasingly become value-adding products. With their expanding role in businesses, the quality of APIs has become a crucial factor in enterprise success, demanding a systematic and exhaustive approach to API development. Thus, the **management of APIs as a value-creating product extends from a purely technical concern at the discretion of individual engineers to a managerial matter**.

While the practical importance of a thorough approach to API quality is widely recognized, it lacks resources that practitioners can use for guidance and reflection in their approach. To this end, we **created a guide towards API product quality**, which consist of two parts:

1) The report (this document) introduces the results of the research project and how they are conceptualized - providing guidance on how to read and understand the self-reflection model. The components of the model are introduced in the following chapters of this report.

2) The self-reflection guide (*see accompanying xls file*) unfolds the research findings in detail. It helps 1) understand, 2) assess, and 3) improve API product quality, which is broken down into different areas, capabilities, and practices each defined and complemented by links to external reports or research. In this way, the Excel model provides a tool for self-reflection that can be further extended and adjusted by practitioners to fit their specific needs.

The guide was created by a comprehensive review and synthesis of both academic and grey literature, and further evaluation and validation through an executive from a large financial service provider from Denmark. The resulting guide provides a comprehensive overview that lists, structures and relates considerations around API product quality.

UNDERSTANDING API PRODUCT QUALITY

KEY CONCEPTS

To effectively capture relations between the factors of API product quality, the framework is structured around **three building blocks: 1) focus areas, 2) capabilities, and 3) practices**. API product quality is characterized by a set of focus areas, each linked to a unique set of capabilities, which in turn consist of a set of practices that constitute a capability. A practice can be a recurring action, a way of doing things, or just a one-time event. Practices are broken down into two categories - *creating practices* and *controlling practices*. The former serve the purpose of creating or improving a capability, while the latter evaluate a capability to derive implications, but do not on their own impact the extent of a capability:

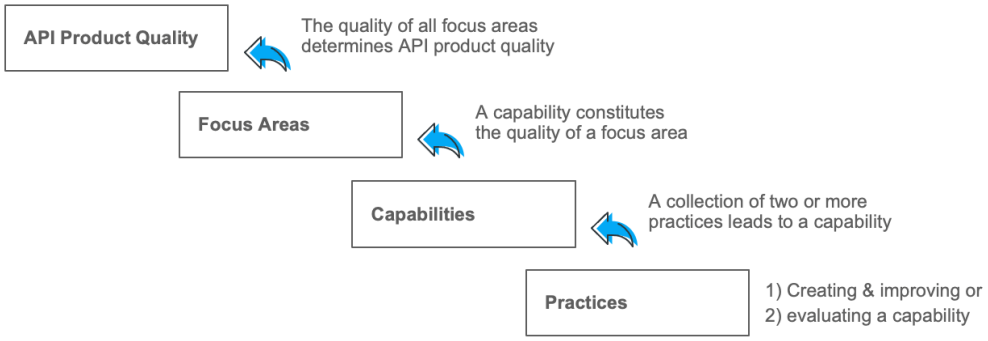


Figure 1 – Four levels of API product quality as key concepts for the framework.

We identified the **focus areas 1) security, 2) performance, 3) usability, and 4) architecture & design**, along with a total of 12 capabilities, 51 creating practices, and 30 controlling practices of API product quality. The wide range of API product quality factors suggests that the question "is an API any good?" can be answered in many ways – and often depends on the company-specific context it is embedded in. Hence, the following focus areas and related capabilities provide a way for practitioners to evaluate which areas of quality characteristics are important to them, and in which areas they currently do not - but should - follow a systematic and considerate approach.

1) **Security**: The API is secure due to sufficient authentication, authorization, encryption, protection, and threat detection.

2) **Performance**: The API handles request in a reliable and responsive manner.

3) **Usability**: The API is usable since it is accessible, learnable, evolvable, and effective.

4) **Architecture & Design**: The API's design is consistent, and the protocol fits to the API's purpose.

FOCUS AREAS AND THEIR CROSS-EFFECTS

During the development of the framework, it became apparent that different focus areas also impact each other. As a result, a collection of cross-effects emerged, capturing interdependencies between different focus areas. Following is an illustration of the four focus areas and identified cross-effects, which emphasize the importance of a holistic approach to API product quality.

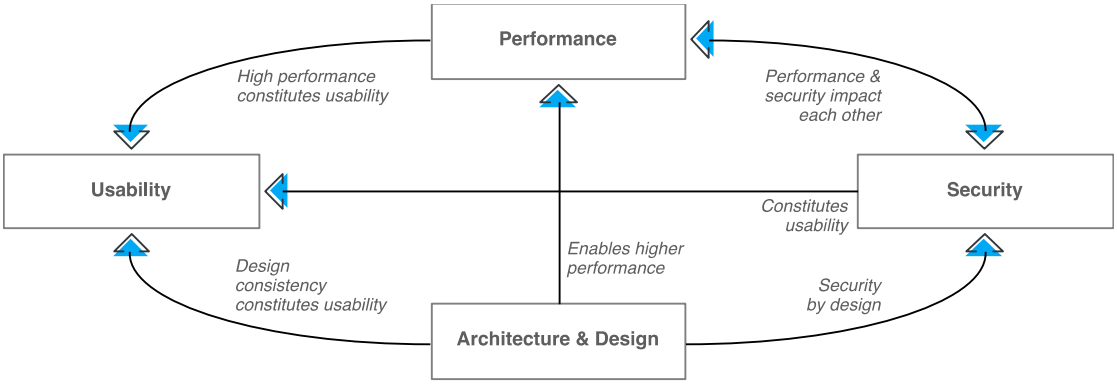


Figure 2 - Identified focus areas of API product quality and their cross-effects on each other.

Usability can be seen as a collection of several capabilities that indicate how easy it is for a developer to use an API. The different capabilities constituting the usability of an API are explored in the following section. Since this is rather an open-ended focus area, its quality is impacted by all other focus areas. Security and performance can be seen as focus areas distinct from usability, which serve their own purpose of being secure or performant while also indirectly affecting the usability of the API product, since an unsecure API that does not operate reliably is less usable.

Architecture & design as the fourth focus area has a notable overlap with usability, as design consistency can also be viewed as its own subpart of usability. However, the design of an API is a crucial factor for the understandability of an API and its quality in general. Thus, architecture & design is emphasized as its own focus area. The importance of the API's architecture & design is underscored by the fact that it has cross-effects on all other focus areas. Besides its impact on usability, architecture & design also constitutes security, as following certain principles during the design and development is imperative for a secure API. A less usable API design also invokes security risks for developers due to misuse of APIs. Moreover, the architecture & design also impact performance. A mismatch between the underlying protocol of an API and the needs of the applications using the API can hurt performance.

FOCUS AREAS: ENHANCING SECURITY, PERFORMANCE, USABILITY AND DESIGN

Emphasized by the cross-effects between the four focus areas, a deep understanding of each focus area is imperative to the quality of the product as a whole. Hence, the following sections introduce each focus area and the identified capabilities and practices to provide a holistic overview of what constitutes the quality of the respective focus area. Creating practices are further divided into practices to implement a feature of some kind, and practices to ensure conformance to certain standards or requirements.

PART 1: SECURITY

Since APIs enable value creation through the exchange of information and services within and between organizations, they expose valuable assets to the other end of a call, often users outside of the organization. Hence, exposed assets must be protected from unauthorized access and external threats. OWASP provides a widely recognized ranking of the top 10 web application security risks, with broken access control being the most common weakness. Other security risks include authentication failures and a lack of injection threat detection and protection. Protecting and securing internal assets from these risks is not trivial. Hence, a solid understanding of the security risks imposed by APIs, as well as mechanisms to defend against them is imperative to secure operations. A collection of respective capabilities and related mechanisms in form of practices is illustrated in the following table.

API Product Quality - Security

Capability	Creating-Practices to create a capability by ...	...implementing a feature	...ensuring conformance	Controlling-Practices to evaluate a capability
Authentication	Implement API authentication mechanisms	☑		Evaluate authentication methods fit
	Implement single sign-on authentication	☑		Conduct authentication testing
Authorization	Implement access models	☑		Log & audit user activity
	Implement token management	☑		Conduct authorization testing
	Implement authorization protocols	☑		Log & evaluate errors
Encryption	Implement TLS protocols	☑		Log & evaluate access attempts
	Implement certificate management	☑		Conduct security testing
	Implement encryption algorithms	☑		
Protection & Threat Detection	Implement allow & deny IP address lists	☑		
	Implement injection threat protection policies	☑		
	Implement DoS protection	☑		
	Conduct consumer side security reviews		☑	

For securing APIs, authentication and authorization are pivotal, related capabilities. Authentication confirms a user's identity through API keys, single sign-on or other mechanisms and methods, establishing a first line of defense. Subsequently, authorization determines the authenticated user's access rights, ensuring they perform only permissible actions. To further protect the data assets in transit between a client and the provided API from being accessed by anybody else, encryption is needed, for example through Transport Layers Security protocols

and further encryption algorithms. As the last capability needed to ensure sufficient security, threat detection mechanisms are essential to protect the API from further attacks, such as malicious code injection.

PART 2: PERFORMANCE

In order to satisfy demands of developers using an API and end-users consuming services built upon that API, it must be highly performant, both in terms of its reliability and responsiveness. As the cornerstone component of the digital ecosystem, APIs often experience increased demand. For example, especially time-sensitive operations, such as trading, rely on the respective APIs to be constantly available and respond quickly to requests. To ensure sufficient reliability, firms can follow several practices, such as managing their resources through load balancing and scaling to ensure reliability of services. Due to the increase and fluctuation in demand, firms are increasingly moving their API to the cloud. Besides scaling available resources to meet demand, firms can increase responsiveness of their API by managing the incoming traffic effectively through creating practices such as implementing request caching or rate throttling. For the performance, many controlling practices were identified that objectively quantify the performance of the provided service through metrics such as uptime, pass rate and latency. It is imperative to evaluate the capabilities along these controlling-practices, as many of these metrics are part of service level agreements as non-functional requirements. Thus, optimizing the capabilities along the controlling-practices contributes both to a higher quality API and in return also to success in terms of business.

API Product Quality - Performance

Capability	Creating-Practices to create a capability by ...	...implementing a feature	...ensuring conformance	Controlling-Practices to evaluate a capability
Reliability	Implement load balancing	☑		Evaluate uptime
	Implement (predictive) scaling	☑		Evaluate pass rate
	Implement failover policies	☑		Conduct robustness testing
	Implement rate limiting	☑		Evaluate latency
	Implement error handling	☑		Evaluate response time
Responsiveness	Implement timeout policies	☑		Evaluate throughput
	Implement request caching	☑		Evaluate CPU load
	Implement request rate throttling	☑		Evaluate memory usage
	Apply data volume limits	☑		Evaluate bandwith usage
	Implement traffic prioritization	☑		
	Implement filtering	☑		
	Implement pagination	☑		

PART 3: USABILITY

Besides the requirements of being secure and performant, there are several important capabilities that make an API usable. APIs that are difficult to use result in low usage growth and low retention of users, hampering the business potential of an API. The identified capabilities illustrated in the following table all constitute the API usability to ultimately enhance the experience of people using the API to develop their own applications. Hence, the identified capabilities follow a logical flow along the user journey. First of all, an API must be discoverable and accessible to internal and external developers that want to use its functionality in an easy way, for example through a well-curated developer portal. Second, providing consistent and in-depth documentation that follows certain standards contributes to the learnability of the API. Third, once developers use an API, the API provider must ensure a versioning strategy that copes with changing business requirements evolving over time. Lastly,

the effectiveness of the API defines to what extent the exposed service enables its users to complete their tasks correctly, measuring whether users succeed in achieving their goals when working with an API. Part of this is fulfilling (functional) requirements, and also adhering to regulatory standards, especially important in the banking (PSD) and healthcare sector (HL7). There is no golden path of getting there, but studying users' needs and acting on their feedback can lead to improvements.

API Product Quality - Usability

Capability	Creating-Practices to create a capability by ...	...implementing a feature	...ensuring conformance	Controlling-Practices to evaluate a capability
Accessibility	Provide a developer portal	☑		Evaluate API usage (growths)
	Provide an API catalog	☑		Evaluate unique customers
Learnability	Provide reference documentation	☑		Conduct expert reviews
	Provide example documentation	☑		Conduct task-based usability tests
	Provide tutorial documentation	☑		Conduct surveys
	Use standard for reference documentation		☑	Conduct regression testing
Evolvability	Implement an API versioning strategy	☑		Conduct user studies
	Implement API depreciation protocol	☑		Conduct system testing
	Decouple API & software versioning	☑		Evaluate API retention
	Decouple in- & external data model & format	☑		
	Decouple in- & external transport protocol	☑		
	Provide update notification to users		☑	
Effectiveness	Work backwards from customer use cases		☑	
	Implement user feedback on functionality		☑	
	Adhere to regulatory standards		☑	

PART 4: ARCHITECTURE & DESIGN

Even when an API provides the needed functionality and provides all capabilities mentioned under usability, the design of an API can make it practically unusable. Design consistency contributes to the understandability of an API. Inconsistent naming schemes and ill-defined standard methods, besides other factors, significantly decrease the usability of an API. As a response, firms must define and follow design guidelines, usually captured in a design guide. Organizations such as Google[1] and Zalando[2] provide design guides clearly showcasing consistent guidelines for all relevant design matters during the entire lifecycle of an API, from development to maintenance and depreciation. Moreover, following a unified principle of API development also contributes to higher design consistency. Zalando outlines the design-first principle in their design guide, dictating that an API must be designed before coding its implementation, opposed to a code-first approach. Another important capability of an API is having a good protocol purpose fit. Different protocols entail different trade-offs and serve different purposes, which must be evaluated against the specific user needs in mind.

[1] https://cloud.google.com/apis/design
[2] https://opensource.zalando.com/restful-api-guidelines/#

API Product Quality - Architecture & Design

Capability	Creating-Practices to create a capability by ...	...implementing a feature	...ensuring conformance	Controlling-Practices to evaluate a capability
Design Consistency	Define API developing principles		☑	Conduct expert reviews
	Define versioning and compatibility rules		☑	Conduct task-based usability tests
	Define naming schemes		☑	Conduct peer-reviews
	Define data formats		☑	Conduct automatic reviews
	Define standard & use of custom methods		☑	Evaluate targeted developers needs
	Define standard fields		☑	
	Define an error model		☑	
	Define common design patters for APIs		☑	
	Define event guidelines		☑	
Protocol Purpose Fit	Implement based on frequency of requests		☑	
	Implement based on data load of requests		☑	
	Implement based on target user system		☑	

LEVERAGING THE API QUALITY FRAMEWORK: GUIDELINES FOR SUCCESS

The findings around API product quality provide important managerial implications and practical guidance to enhance API product quality in organizations. First, the **framework can guide organizations that are just starting to explore APIs** as a new way to increase internal efficiencies or as a new revenue source. Due to the division in creating and controlling practices, the framework practically guides organizations, as it includes (one-time) actions that impact a capability and thus quality, and actions that evaluate the actual extent of the capability and resultant API product quality. Second, especially organizations with an existing API product portfolio can benefit from this study. While they already have an approach to API product quality, it might often be based solely on past experiences and learnings as a result from costly mistakes. This **framework provides a neutral, outside perspective on API quality that can be seen as a guide for self-reflection**. Practitioners can use it to evaluate their current approach, challenge their understanding of API product quality, and expand on it. By tailoring it to their specific needs and continuously adding their own learnings in the form of new practices, capabilities, and focus areas, the **framework serves as an evolving knowledge basis for organizations**. Beyond the technical guidance and self-reflection, the examined cross-effects underscore the important managerial implications of a well-considered approach to API product quality. Ensuring high quality for all focus areas requires significant orchestration of resources and processes. Since the quality of APIs has become a crucial factor in enterprise success, managers are required to constantly reflect on the practices conducted in their organization to ensure a high-quality API.

CONCLUSION

As organizations are moving from ad-hoc exploration to approaching API at scale there is a need to consider the qualities of a desirable API. This white paper intends to provide a foundation for a self-assessment with the purpose of initiating a reflection and potentially a correction and guidance to help make the API approach fit for purpose in a given organization.

We hope that you find this guide useful. If you find it useful, please let us know. If you miss something or have further input for us to consider then let us know. The technical possibilities and practical demands on API product management are changing rapidly. The nature of a report like this will always be a work in process and we look forward to updating it as we learn more about API product quality and beyond!

A NOTE ON THE RESEARCH

The research underlying this report was conducted at Copenhagen Business School, Department of Digitalization as part of a research project on API management. The research approach followed a systematic analysis and an iterative process of building and evaluating the framework with practitioners:

1) A wide range of factors around API product quality were identified through a systematic analysis of academic and grey literature, such as blog posts, white papers, and web articles. Including both academic and grey literature ensured a more exhaustive knowledge base upon which the framework is build.

2) Constructing the framework followed an iterative process of building and evaluating it. The objective of the building process was to construct a holistic, structured framework that enables practical guidance. As the literature lacks a holistic view of API product quality, there was no pertinent collection of focus areas and capabilities that could be directly. Hence, the framework was constructed with a bottom-up approach, where identified practices were categorized into capabilities and subsequently, capabilities were categorized into focus areas. Thus, the framework aggregates a vast collection of perspectives on API product quality to form a more holistic and comprehensive view on the topic.

3) To evaluate the framework, several iterations of expert evaluation were conducted, in which the respective version of the framework was discussed and refined with an executive from a Danish financial service firm. The objective of these discussions was to improve the quality of the framework through continuous evaluation and ensure its practical guidance. The received input provided significant contributions regarding the framework structure, purpose, and specific practices and capabilities that were either missing, misplaced, or obsolete.

Authors:
Ben Eaton (corresponding author)
be.digi@cbs.dk

Stefan Henningsson
sh.digi@cbs.dk

Jonas Hedman
jhe.digi@cbs.dk

Maximilian Schneider
maximilianschneider1@outlook.com

Department of Digitalization

Copenhagen Business School
Solbjerg Plads 3
2000 Frederiksberg
Denmark

cbs.dk

In collaboration with:

Benny Boye Johansen
bjo@saxobank.com

October 2024

DOI: 10.22439/DIGI.2024.01
https://doi.org/10.22439/DIGI.2024.01