Towards SLA-Driven API Gateways

Antonio Gámez-Díaz, Pablo Fernández-Montes, and Antonio Ruiz-Cortés

University of Sevilla [⋆]
{agamez2,pablofm,aruiz}@us.es

Abstract. As APIs are becoming popular to build *Service-Based Applications* (SBA), API Gateways are being increasingly used to facilitate API features management. They offer API management functionalities such as pricing plans support, user authentication, API versioning or response caching. Some parts of the information that an API Gateway needs are already included into a *Service Level Agreement* (SLA), that providers use to describe the rights and the obligations of involved parties in the service. Unfortunately, current API Gateways do not use any SLA representation model nor SLA underlying technology, thereby missing potential opportunities. In this paper we analyze the state of the art to justify the current situation and we identify some research challenges so as to achieve SLA-Driven API Gateways.

1 Introduction

As software architecture is shifting form a *product paradigm* to a *service paradigm* (i.e. *software as a service* (SaaS)), the traditional models, where users buy a perpetual-use license (e.g. *Microsoft Office*) are moving to service-oriented models, where users have to buy a subscription from the service provider (e.g. *Zoho*). Moreover, in the current *Service-Oriented Computing* paradigm, *Service-Based Applications* (SBA) are becoming increasingly important, especially nowadays with the popularization of cloud platforms.

Furthermore, we can find a large API marketplace where providers get paid for their clients API usage. In this context, most APIs have different pricing plans in order to fit with the customer's needs and they have to determinate the proper service level for each plan. These decisions need to be managed and aligned with business goals.

Traditional service-oriented industries define contractual documents known as *Software Level Agreement* (SLA) which stipulates and commits the provider to a required level of service according their business model. We can extend SLA model to APIs providers so that we get SLA managed services where both customer and provider agree certain service properties, such as service level objectives, service features, up-time guarantees or number of requests.

[⋆] This work has been partially supported by the European Commission (FEDER), the Spanish and the Andalusian R&D&I programs (grants TIN201232273 (TAPAS), TIC5906 (THEOS) and P12TIC-1867 (COPAS)) and TIN2014-53986-REDT.

Since most API Gateways do not have any associated SLA, the business model is coupled with the API Gateway logic, thus there exists a rigid and complex evolution, giving as a result a code difficult to maintain. Therefore, generated solutions will not be generic, scalable nor domain-independent ones. On the contrary, by having an implicit SLA, open, scalable, flexible and domain-independent solutions can be built.

When API providers try to align technology with their business goals they can use an API Gateway solution, i.e. a third-party solution that try to resolve some API management common tasks such as user authentication, establishing a certain request limit per user or blocking resources depending upon the user who called the API. This solution is uncoupled from the API code. Nevertheless, the service provider depends on a certain third-party gateway, which offers limited features that may not fit with the business needs.

In this work we analyze APIs and API Gateways to identify requirements for a future framework that supports SLA-driven API Gateways solutions. We analyze a few APIs and study what they have in common. Next, we focus on some API Gateway solutions existing in the market and we detail what they really offer. Finally, with this information, we could establish research goals so that we continue working on this line. Figure 1 shows a context diagram that represents the relationships between the elements we describe on the paper.

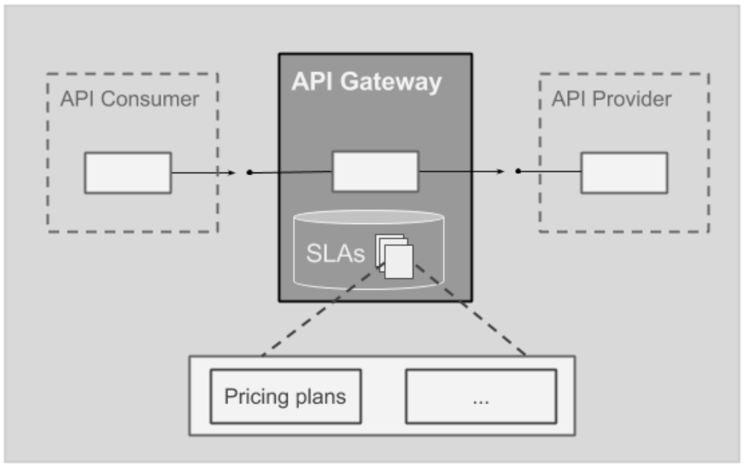


Fig. 1. Context diagram.

This paper is organized as follows: Section 2 describes the current situation of APIs by giving examples of real ones and analyzing its features. In Section 3, we study a few examples of API gateways focus on its relationship with SLA management. Next, in Section 4 we try to identify new research challenges that can be used as requirements for a further design and development. Finally, Section 5 shows some remarks and conclusions.

2 Pricing plans in APIs

In this section we show the relationship between the properties that API providers are used to monitor and we analyze some real world APIs in order to highlight different pricing plans.

A Service Level Agreement (SLA) can be seen as containers of the functional and non-functional properties that both parties (the service consumer and the service provider) agree specifying its rights and obligations during the interaction. SLAs are widely used in the industry in situations where consumers and providers need or desire to explicitly express certain guarantees over the service transaction. There exists several purposes to describe SLAs, for instance, WS-Agreement [1], purposed by *Open Grid Forum*. It is one of the firsts specification which counts with well-defined validity criteria [3] and an environment to edit, validate and monitor SLAs[1] [6].

In connection with SLAs, APIs often monitor properties for each user or each request (e.g. the number of requests, the called endpoint, user authentication data, etc.). According to these properties and their own business model, providers can design pricing plans and get paid for their service usage. An important challenge for API service providers is managing different plans with their customers while satisfying their business objectives. These monitorable properties could be introduced into the existent SLA between the consumer and the provider. For this reason, it is possible to establish a relationship among SLAs and pricing plans.

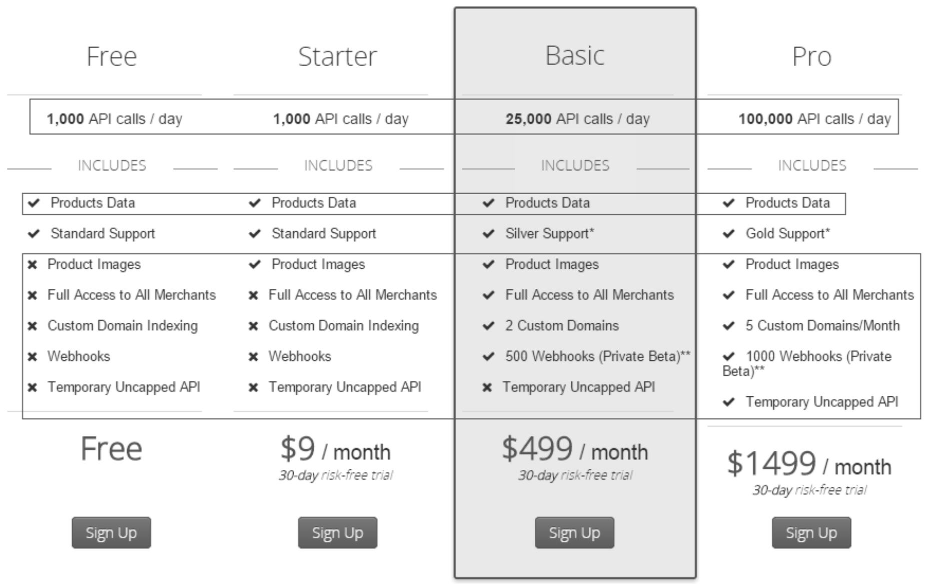
In order to illustrate the variability of existing API pricing plans we show and detail a real example in Figure 2.

In Figure 2 we show the pricing plan of Semantic3 company, whose API provides access to a database of normalized e-commerce products, UPC, and pricing data. There exists 4 different plans which are primarily based upon daily requests amount. It also offers diverse features and support level for each plan. For instance, in *free plan* we are not able to access the full API data nor indexing custom domains. Definitely, most APIs have to control user requests and the features they offer to each client.

API pricing plans range from completely free to paid plans. In this context, in order to capture the variability in the different scenarios we have studied 9 different APIs.

- **Freemium (P_1)**, when there exists a limited basic plan and users can overtake the limits depending on they pay for the service, e.g. *AlchemyAPI has free plan of 1 thousand transactions per day but subsequent ones are paid.*
- **Tiered (P_2)**, when each tier has its own set of services and allowances for access to API resources and it is fixed a prize for each tier, e.g. *Semantics3 has 3 different paid plans.*
- **Unit-based (P_3)**, when pricing is based upon a pre-defined unit of measurement, such as API calls or the storage used on disk, e.g. *GroupDocs establishes a base price and \$0.15 is billed for each extra operation.*

[1] This environment is deployed at isa.us.es/IDEAS.



Free	Starter	Basic	Pro
1,000 API calls / day	**1,000** API calls / day	**25,000** API calls / day	**100,000** API calls / day
INCLUDES	INCLUDES	INCLUDES	INCLUDES
✓ Products Data	✓ Products Data	✓ Products Data	✓ Products Data
✓ Standard Support	✓ Standard Support	✓ Silver Support*	✓ Gold Support*
✗ Product Images	✓ Product Images	✓ Product Images	✓ Product Images
✗ Full Access to All Merchants	✗ Full Access to All Merchants	✓ Full Access to All Merchants	✓ Full Access to All Merchants
✗ Custom Domain Indexing	✗ Custom Domain Indexing	✓ 2 Custom Domains	✓ 5 Custom Domains/Month
✗ Webhooks	✗ Webhooks	✓ 500 Webhooks (Private Beta)**	✓ 1000 Webhooks (Private Beta)**
✗ Temporary Uncapped API	✗ Temporary Uncapped API	✗ Temporary Uncapped API	✓ Temporary Uncapped API
Free	$9 / month 30-day risk-free trial	$499 / month 30-day risk-free trial	$1499 / month 30-day risk-free trial
Sign Up	Sign Up	Sign Up	Sign Up

Fig. 2. Semantics3 Pricing Plan.

- **Pay-as-you-go (P_4)**, when a customer accumulates a monthly bill basing on what is its use of API resources, e.g *Stupeflix defines a cost per each API resource (such as $0.01 per minute of input video), thus the bill depends upon the user usage.*
- **Enterprise pricing (P_5)**, when pricing is part of a larger sales process, it is a common way for large companies to price products and services based upon customer's needs, e.g. *in AlchemyAPI, clients with over 1 billion requests per month are eligible for a custom plan.*
- **Volume pricing (P_6)**, when pricing is based upon volume purchases by buying in bulk, e.g. *MailGun defines an initial price of $0.00050 for first 500,000 sent emails and $0.00035 for the next 1,000,000.*

We study some selected APIs in terms of their pricing plan they offer and additional features. In Figure 3 we illustrate that comparison[2], where a cell C_{ij} marked with an x means that API i supports the property j. Note there is not any property that could be considered as a particular case from another. Next, we detail an example extracted from the same table.

A large part of studied APIs have a unit-based (P_3) plan, probably due to it is easy to implement and is enough for many API providers. As an example, *AlchemyAPI* has two different pricing plans. On the one hand, it has 3 plans only based on the daily request volume (between 90.000 and 3 million) and a

[2] The reference of all studied APIs can be found at: goo.gl/0n8cPr

API Study						
API	P_1	P_2	P_3	P_4	P_5	P_6
AlchemyAPI	x	x	-	x	x	-
Algorithmia	-	-	-	x	x	-
FlightStats	-	-	-	-	x	x
Groupdocs	x	x	x	-	x	-
Kraken	-	x	x	-	-	-
Mailgun	x	-	-	x	x	x
OpenWeatherMap	x	x	-	-	-	-
Semantics3	x	x	-	-	x	-
Stupeflix	-	-	-	x	-	x

Fig. 3. APIs comparative.

free tier of 1000 requests is also available. On the other hand, we could select a pay-as-you-use plan so that we would be billed only for the requests made.

It is important to separate how the total cost is calculated from how the client is billed. For instance, *Algorithmia* has a credit-based plan, where it is necessary to add funds to one's account to be able to continue making API requests. Nevertheless, it is only a particular case of a unit-based plan.

As another example of unit-based plan, *FlightStats* has a plan based upon the requests you made, although it distinguishes according to the resource type you call in the request (e.g. is more expensive to serve a request that implies more CPU or memory consumption). *Groupdocs*, whose pricing plan is not based upon the requests amount but a pre-defined unit of measurement based their own business logic.

3 Pricing plans in API Gateways

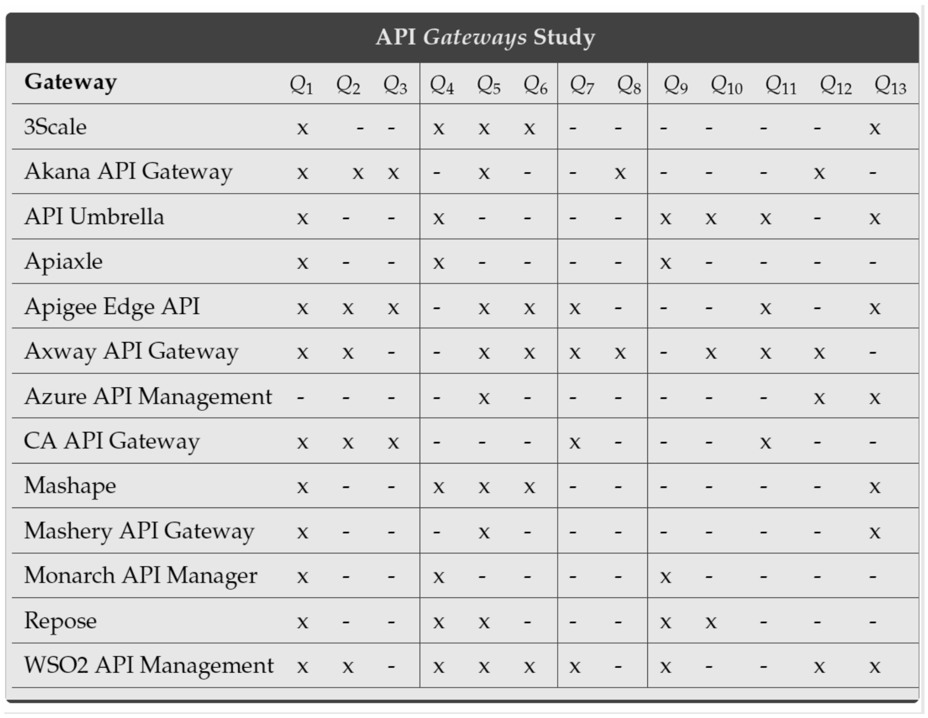
In this section we discuss about the concept of *API gateways* (other authors already use this term: [10], [2]) and study how they could be related with pricing plans and SLA-Driven APIs by analyzing what they really offer.

An API gateway offers API management features, such as security (e.g. access control, DoS attacks blocking), pricing plans support, API analytics, request monitoring, API lifecycle control (e.g. service orchestration, govern data flows, API design support) or response transformation (e.g. JSON to SOAP). Furthermore, API Gateways are usually implemented as virtual appliances, virtual machine images, SaaS or reverse proxies.

From the 13 API Gateways studied we have selected 13 criteria in order to compare all of them. In Figure 4 we illustrate that comparison[3], where a cell C_{ij} marked with an x means that platform i has the property j, and we provide some examples extracted from the same table.

- **Security:**
  - **Basic Authentication** (Q_1): it supports basic authorization methods, e.g. *Mashape let us configure authentication by a HTTP header, a query parameter and an API key.*
  - **Advanced Authentication** (Q_2): it supports advanced authorization methods, e.g. *Apigee offers OAuth and LDAP authentications.*
  - **Attacks protection** (Q_3): it includes a way to protect itself from attacks, e.g. *CA prevents our API from DoS attacks.*
- **Pricing plans support:**
  - **Free Tier** (Q_4): it has a free plan with basic features or limited by somehow, e.g. *3Scale offers a basic tier with 50 thousand calls per day for free.*
  - **Requests limit** (Q_5): there exists a way to establish quotas per user, e.g. *with Azure it is possible to limit API calls by user to a custom value.*
  - **Enterprise Plans** (Q_6): it offers customized plans for enterprise clients, e.g. *3Scale let enterprise clients configure unlimited APIs with a guaranteed availability.*
- **Lifecycle control:**
  - **API versions** (Q_7): it supports versioning e.g. *Axway let us set different versions from the same API.*
  - **Orchestration** (Q_8): there exists a way to orchestrate RESTful services in terms of SOA governance, e.g. *Akana let us control complete API lifecycle and orchestrate all services involved.*
- **Other features:**
  - **OpenSource** (Q_9): the code is available to anyone (e.g. *WSO2 and API Umbrella are both Open Source solutions*).
  - **Load balancing** (Q_{10}): it makes possible to do load balancing between various API servers, e.g. *API Umbrella balance the requests between our backend.*
  - **Cache** (Q_{11}): it includes request caching in order to reduce the effective API calls amount, e.g. *Apigee save the API responses in order to serve the same data if there is no changes.*
  - **Response conversion** (Q_{12}): it gives the possibility of converting API response into another language, e.g. *Akana perform a JSON to SOAP conversion.*
  - **Documentation** (Q_{13}): it generates a developer portal with API documentation, e.g. *Mashape generate an interactive documentation and a playground where developers can make requests.*

[3] The reference of all studied API Gateways can also be found at: goo.gl/0n8cPr



API *Gateways* Study													
Gateway	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	Q_{11}	Q_{12}	Q_{13}
3Scale	x	-	-	x	x	x	-	-	-	-	-	-	x
Akana API Gateway	x	x	x	-	x	-	-	x	-	-	-	x	-
API Umbrella	x	-	-	x	-	-	-	-	x	x	x	-	x
Apiaxle	x	-	-	x	-	-	-	-	x	-	-	-	-
Apigee Edge API	x	x	x	-	x	x	x	-	-	-	x	-	x
Axway API Gateway	x	x	-	-	x	x	x	x	-	x	x	x	-
Azure API Management	-	-	-	-	x	-	-	-	-	-	-	x	x
CA API Gateway	x	x	x	-	-	-	x	-	-	-	x	-	-
Mashape	x	-	-	x	x	x	-	-	-	-	-	-	x
Mashery API Gateway	x	-	-	-	x	-	-	-	-	-	-	-	x
Monarch API Manager	x	-	-	x	-	-	-	-	x	-	-	-	-
Repose	x	-	-	x	x	-	-	-	x	x	-	-	-
WSO2 API Management	x	x	-	x	x	x	x	-	x	-	-	x	x

Fig. 4. API Gateways comparative.

3Scale is an API gateway service provider who let us connect up to 3 different APIs with a daily traffic close to one million requests. It is possible to establish requests filtering policies based upon trusted API keys, black and whitelisting or OAuth authentication. In order to create billing plans, we could limit the requests amount per user and create a billing for that use or simply a billing plan based upon the requests that a user has made; it is also possible to create trial plans. Besides these core features, it also offers several statistics, a developer portal, documentation generation and different ways to deploy the reverse proxy towards *3Scale*.

Almost all API gateway studied services offer the same core functionality: request authorization and quota establishment so that various billing plans could be created. It might be enough for few APIs that do not need a more complex business billing logic, but does not allow to introduce all business logic we may need.

4 Research Challenges

We propose API Gateways based on SLAs. We have analyzed API properties and billing needs and how third-party API Gateways solutions could be useful to solve this problem. Nevertheless, there are some API requirements that are not satisfied by the previous solution, so the only possibility is to code it manually into the API code. This implies an increasing error probability and an API code coupled itself with management one. Previously we have shown the importance of considering SLAs in our design, therefore we focus on a SLA-Driven solution.

In this section we analyze these requirements and we try to identify other ones that could appear in a different scenario. Finally, we propose several research challenges in order to establish a road-map to focus our researches on.

We can identity new requirements that could be met by a theoretical API that use a SLA based pricing plan.

- **Challenge 1**: supporting temporarily in pricing plans. As current API gateways are static, if an API pricing plan depends on the moment that the API is called, it would be necessary to code this logic manually. It is possible to find more examples, such as support plan life-time (i.e. when it starts and expires) or time periods for guarantee terms (e.g. availability level is guaranteed only during Monday-Friday working hours). In this way, we have dynamics and elastic pricing plans, regulated according to demand cycles. In [7], [9] and [8] authors analyze formal temporarily in SLAs.
- **Challenge 2**: adding a more fine-grained configuration by supporting other features and metrics than the requests amount. An API could define custom metrics that represent the cost for the business. For instance, metrics such as CPU time consumption, memory or storage usage.
- **Challenge 3**: adding support to compensations (penalties and rewards) according to the under-fulfillment or over-fulfillment of SLA service level objectives (SLO). For instance, API provider could be committed to serve requests in an established time; if there is a percentage of these request served out of time, provider should compensate client by certain way (e.g. claiming a refund). In [5] and [4] authors show the formal specification of a SLA with compensations.

5 Conclusions

In this paper we have shown an analysis of the state of art in API and API Gateway paradigm and propose a SLA-Driven solution in an API Gateway design. The API and SaaS management is a difficult task and there is no solution yet that help providers to automatize simple things like pricing management, feature limitation, billing based upon custom parameters, etc. Currently, service providers have to code this logic by hand or they ought to resort to third-party platforms, which involves an extra cost. We try to define the main research lines so that we could continue working on and, in the future, we could give a framework that permits APIs to be managed by Service Level Agreements.

References

1. Alain Andrieux, Karl Czajkowski, Asit Dan, Kate Keahey, Heiko Ludwig, Toshiyuki Nakata, Jim Pruyne, John Rofrano, Steve Tuecke, and Ming Xu. *Web Services Agreement Specification (WS-Agreement).* 2007.
2. Mingyu Li, Qian Zhang, Hanyue Chu, Xiaohui Hu, and Fanjiang Xu. Conha: An soa-based api gateway for consolidating heterogeneous ha clusters. In *2013 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 1–3. IEEE, 9 2013.
3. Carlos Muller. *On the Automated Analysis of WS-Agreement Documents.* PhD thesis, 2013.
4. Carlos Muller, Antonio Manuel Gutierrez, Octavio Martin-Diaz, Manuel Resinas, Pablo Fernandez, and Antonio Ruiz-Cortes. Towards a formal specification of slas with compensations. In *On the Move to Meaningful Internet Systems: OTM 2014 Conferences*, volume 8841 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2014.
5. Carlos Muller, Antonio Manuel Gutierrez, Manuel Resinas, Pablo Fernandez, and Antonio Ruiz-Cortes. Towards compensable slas. In *4th International Workshop on Adaptive Services for the Future Internet (WAS4FI in ESOCC14)*, 2014.
6. Carlos Muller, Antonio Manuel Gutierrez, Manuel Resinas, Pablo Fernandez, Antonio Ruiz-Cortes, Antonio Manuel Gutierrez, and Manuel Resinas. iagree studio: A platform to edit and validate ws-agreement documents. In *11th Intl. Conf. on Service Oriented Computing (ICSOC)*, volume 8274 LNCS, pages 696–699, 2013.
7. Carlos Muller, Octavio Martin-Diaz, Manuel Resinas, Pablo Fernandez, and Antonio Ruiz-Cortes. A ws-agreement extension for specifying temporal properties in slas. In *III Jornadas Cientíico-Técnicas en Servicios Web y SOA*, pages 77–84, 9 2007.
8. Carlos Muller, Octavio Martin-Diaz, Antonio Ruiz-Cortes, Manuel Resinas, and Pablo Fernandez. Improving temporal-awareness of ws-agreement. In *5th. Intl. Conf. on Service Oriented Computing (ICSOC)*, volume 4749 of *LNCS*, pages 193–206. Springer Verlag, 9 2007.
9. Carlos Muller, Antonio Ruiz-Cortes, and Pablo Fernandez. Temporal-awareness in slas. why should we be concerned? In *Service-Oriented Computing - ICSOC 2007 Workshops: 1st. Non-Functional Properties and Service Level Agreements in Service Oriented Computing Workshop (NFPSLA)*, volume 4907 of *LNCS*, pages 166–173. Springer Verlag, 9 2007.
10. Qian Zhang, Hanyue Chu, Mingyu Li, and Xiaohui Hu. A unified api gateway for high availability clusters. In *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*, pages 2321–2325. IEEE, 12 2013.