Inter-platform Interoperability for Zero Defects Platforms – API Gateway

Marc Dorchain[1], Jonas Schmitt [2], Klotilda Muca [3]

[1] Director Research Software AG, Altenkesselerstrasse 17, 66115 Saarbrücken, Germany [2] Master Student Research Software AG, Altenkesselerstrasse 17, 66115 Saarbrücken, Germany [3] Researcher Software AG, Altenkesselerstrasse 17, 66115 Saarbrücken, Germany

Abstract

Background: Zero Defects Manufacturing Platforms will be integrated with external Internet of-Things platforms of commercial manufacturers as well as other (research) projects. In doing so, the data sources available there should be usable as well as the integration of security measures to ensure controlled and frictionless cross-platform access for everybody. Objective: This paper describes an implementation approach of making use of open standards as an example of good design patterns and architecture and gives a short impression of the performance reachable by following these practices compared with other solutions. Method: Using a prototype implementation and conducting a systematic performance study. Results: A prototype implementation by making use of open standards (e.g. Swagger) and open-source tools (e.g. Spring framework) that have been evaluated.

Conclusion: The results provide a reference to developers to design a system to enable controlled and smooth cross-platform access in manufacturing environment.

Keywords

Interoperability, Standardization, API Management, I-ESA 2024, CEUR-WS

1. Introduction

The EU Horizon project ZDZW aims to enable interoperability in several ways. Not only interoperability with other external systems is meant, but also the possibility to connect to multiple instances of the ZDZW (Zero Defect Zero Waste) platform itself.

A standard setup of any platform / tool used not only in the manufacturing industry usually consist of a least one development system, followed by a test system and one or more system in a production environment (**Figure 1**).

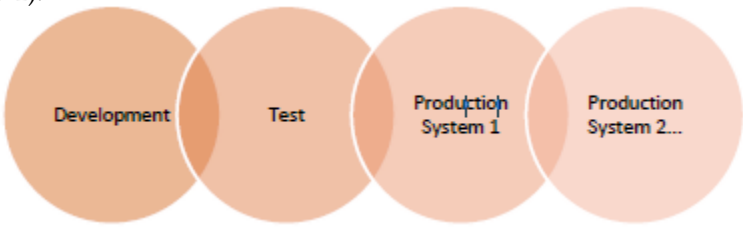

Figure 1 Simple system life cycle

Of course, this is a very simple landscape that is only meant to illustrate a distribution landscape where only one system is running in one place, e.g. one system responsible for one factory in a closed network.

In ZDZW, a dedicated API (application programming interface) management has been introduced to solve the described situation. API Management (see Figure 2) means that you have one single place where all APIs of the platform are exposed. Only those partners that are allowed may access an API, and it can be accessed only through an API gateway. The gateway has two main functions: acting as a kind of firewall blocking all unwanted access from outside and giving developers or admins one single place where all APIs can be found and managed including monitoring of the usage which might

Proceedings I-ESA 2024 12th International Conference on Interoperability for Enterprise Systems and Applications, April 10–12, 2024, Crete, Greece
EMAIL: marc.dorchain@softwareag.com (A. 1); jonas.schmitt@softwareag.com (A. 2);klotilda.muca@softwareag.com (A. 3)

be important from a business perspective of the platform owner if he can see how often an API is going to be called from a supplier or a customer outside of its organization.

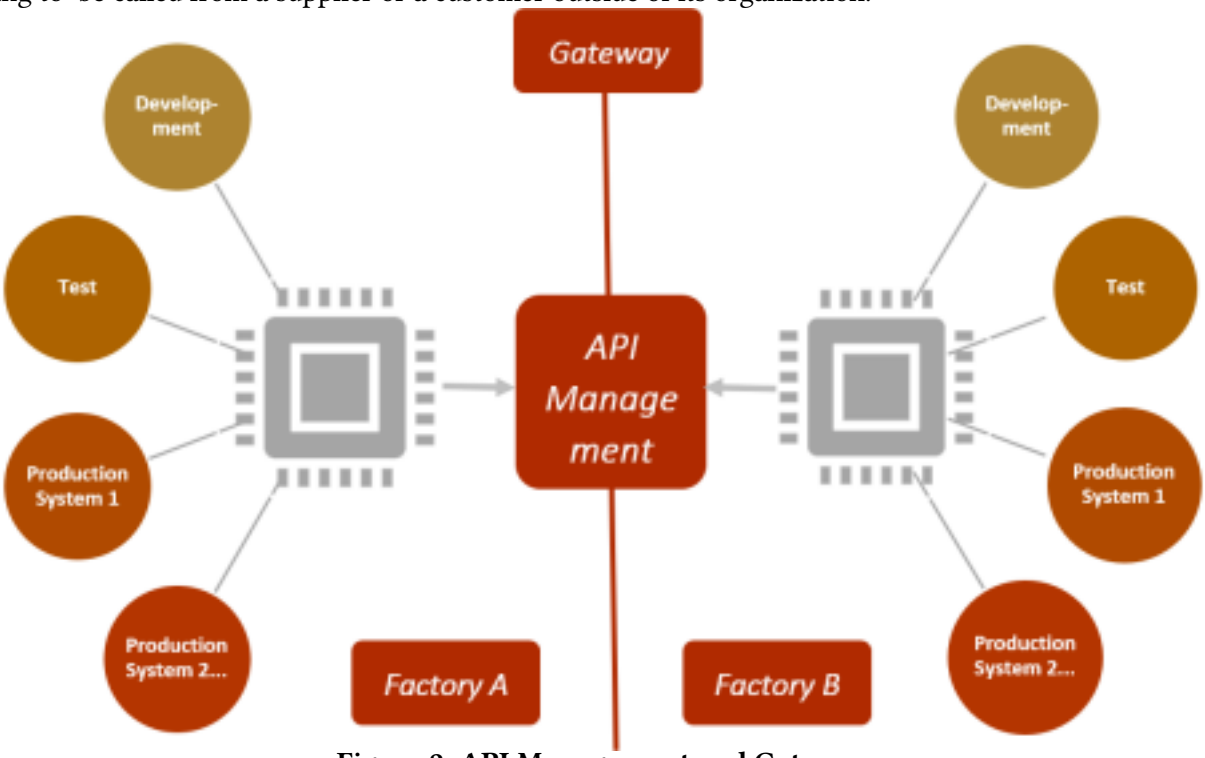

Figure 2: API Management and Gateway

It's an example of how an API Gateway proxy server can be implemented with the help of Spring Cloud Gateway and extended with individual functionalities. It is also covered how to establish a database connection with Spring Data and how to convert custom classes into specific database formats. With Spring Security and OAuth 2 helper libraries, it was possible to implement authentication and authorization of incoming requests on arbitrarily configurable authorization servers.

A Proof-of-Concept implementation has been developed and evaluation of the performance of implementation against some alternative solutions (commercial and non-commercial) has been made: • Direct call

- NGINX [8]
- Spring Cloud Gateway [9]
- Netflix Zuul [10]
- webMethods API Gateway (closed source/commercial product) [11]

The performance results are based measuring request latencies during increasement of the number of parallel accesses.

2. API Management and Gateway

A microservices-based integration architecture requires new patterns for event-based communication, micro gateways for policy enforcement, and network-level controls to be managed with a service mesh.

All examined tools and components more or less fulfill all aspects besides the direct calls which have been added as a reference point only. Finally, performance evaluation (Table 1) has been done against a commercial product.

2.1. Functional aspects

The functional aspects of a possible implementation do all follow the following sequence diagram, that shows the access of an external API by using an API gateway (***Figure 3***)**Figure 2**.

Assuming the use of OAuth 2 as a common protocol for authorization request as a de-facto standard is used the procedure of registering clients and/or APIs needs to be handled dynamically. The sequence is following the simple logical structure of an atomic functional requirement to technically receive a response and is not including any other functional requirements like specific user interaction designs vice versa:

- Request
- Validate access token e.g. via OAuth 2
- Transfer modified request
- Reply

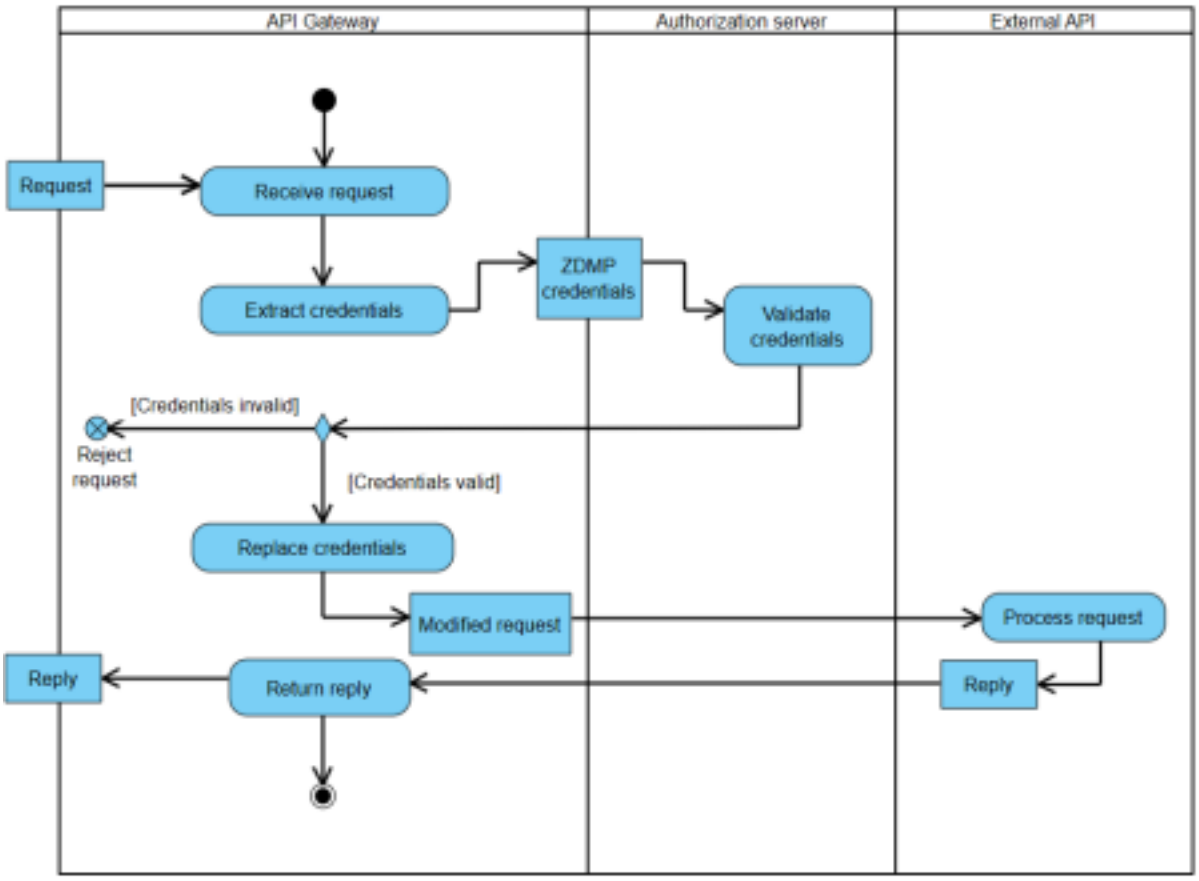

Figure 3: Access of an external API by using the API gateway

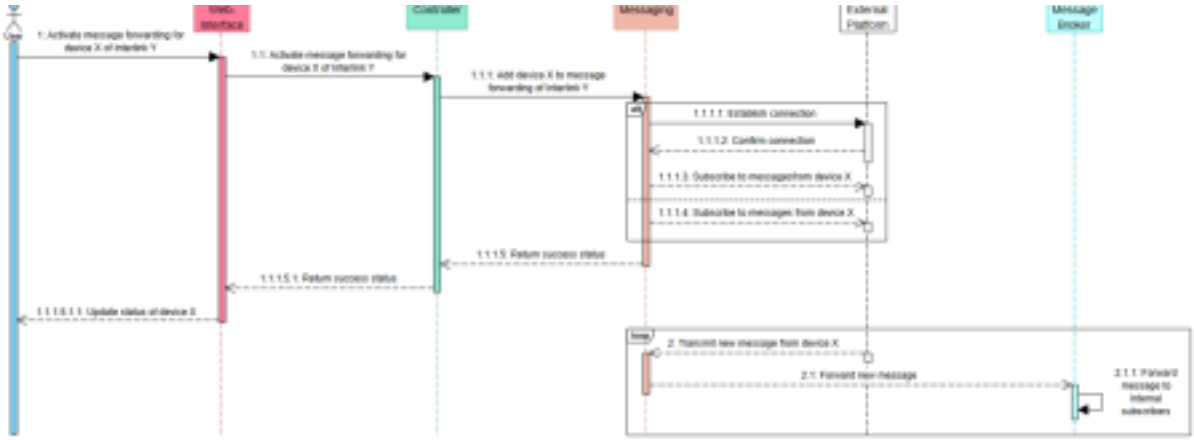

Figure 4: Activation of message forwarding

NGINX is a popular open-source proxy and web server software that can be used at the application level for HTTP and email, as well as at the transport level for TCP/UDP. It runs as a Linux or Windows service and is controlled via configuration files and process signals.

However, this type of configuration complicates the ability to dynamically add or delete new APIs or routes, as described in the concept. For that reason the use of NGINX [5] has been discarded and no reference implementation has been included in the performance evaluation.

Spring Cloud Gateway [9] is a project that provides a set of libraries for building an API Gateway. It aims to provide an effective way to route to APIs and provide support for security, monitoring/metrics, and resiliency. It is based on Spring Framework and Spring Boot and thus runs on Windows, Mac, Linux.

Netflix Zuul [10] is a gateway service that provides dynamic routing, monitoring, resiliency and stress testing, authentication, and security, and more. It is written in Java and thus runs on Windows, Mac, Linux.

webMethods API Gateway [11] is a commercial tool that provides capabilities to perform all the administration and API related tasks such as creating APIs, defining and activating policies, creating applications, and consuming APIs as well as security and policies.

2.2. Performance Evaluation

As a test API endpoint, the non-existent endpoint https://google.com/error has been used for the following reasons: it always returns a HTTP 404 error page; and is provided by the large internet company Google, so that a constant and reproducibility response time can be expected even under heavier loads.

As a test tool Apache Bench4 is used, a command-line program that measures connection and response times of HTTP servers, while maintaining parallel connections. A call for a measurement follows the following command line pattern:

```
ab -c <number of parallel calls> -n <total number of calls> \
-H "Authorization: Bearer <JWT>" "http://<Gateway URL>/error"
```

All measurements have been executed on a laptop with Intel i5 9th generation, 16 GB memory (OS Windows 10).

Table 1
Performance evaluation of API gateways

Implementation	Parallel requests	Total requests	time per request (ms) Average	Median	Standard Deviation	Transferrate (kb/s)	Delay by Gateway (average)
direct call	5	50	122	119	14,8	61,11	
	25	250	144	123	78,4	222,93	
	100	1000	226	122	370,1	476,69	
	250	2500	420	128	1028,7	567,79	
Spring Cloud Gateway	5	50	128	128	3,8	68,14	+6
	25	250	135	125	27,4	311,53	-9
	100	1000	143	127	49,9	1086,53	-83
	250	2500	175	15	57	2449,03	-245
Netflix Zuul	5	50	125	125	2,6	69,42	+3
	25	250	157	138	35	264,94	+13
	100	1000	54	473	483,4	298,72	+368
	250	2500	1435	1118	1063,4	311,03	+1015
webMethods API Gateway	5	5	140	138	7,2	22,8	+18
	25	25	251	165	287,2	56,59	+107
	100	1000	409	355	268,8	256,39	+183
	250	2500	1166	974	758,2	292,71	+746

The best results of a category and concurrency level are marked in green, the worst in red. It becomes clear that Spring Cloud Gateway takes the least amount of time per request at almost all stages. Spring Cloud Gateway provides a set of libraries for building an API Gateway on top of the Spring and Spring Boot Framework[9]. However, it must also be considered that some tools could possibly perform better in these areas if executed on better and more hardware resources, as some tools are designed for larger installations. But that has not been part of this study.

Acknowledgements

The ZDZW project has received funding from the European Union's Horizon Europe program under grant agreement No 101057404. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Platform technologies of the project ZDZW: https://www.zdzw-project.eu/outcomes/ SPRING Data: https://spring.io/projects/spring-data/
SPRING Security: https://spring.io/projects/spring-security/
SPRING Cloud Gateway: https://spring.io/projects/spring-cloud-gateway/
Swagger Parser: https://github.com/swagger-api/swagger-parser

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] Carnell, John, and Illary Huaylupo Sánchez. Spring microservices in action. Simon and Schuster, 2021.
[2] Tudose, Catalin, Christian Bauer, and Gavin King. Java persistence with spring data and hibernate. Simon and Schuster, 2023.

[3] Walls, Craig. Spring in action. Simon and Schuster, 2022.
[4] Spilca, Laurentiu. Spring security in action. Simon and Schuster, 2020.
[5] Nguyen, Quy, and Oras F. Baker. Applying Spring Security Framework and OAuth2 To Protect Microservice Architecture API. *J. Softw.* 14.6 (2019): 257-264.
[6] Trebichavský, Richard. "API Gateways and Microservice Architectures." 2021.
[7] Apache Bench 2024. URL: https://httpd.apache.org/docs/2.4/programs/ab.html
[8] Nginx. URL: https://en.wikipedia.org/wiki/Nginx
[9] SPRING Cloud Gateway. URL: https://spring.io/projects/spring-cloud-gateway/
[10] Netflix Zuul. URL: https://github.com/Netflix/zuul/wiki
[11] webMethods API Gateway 2024. URL: https://www.softwareag.cloud/site/product/webmethods api.html