**Project – Digital Reality**

|  |  |  |
| --- | --- | --- |
| INDEX | | |
| Sr No. | Topic | Page No. |
| 1 | [Partner Role in Project](#_Partner_Role_in_1) | 2 |
| 2 | [Challenges](#_Challenges) | 2-3 |
| 3 | [Solutions w.r.t focal points](#_Solutions_w.r.t_below) | 3-4 |
| 4 | [How APIGEE helped to solve/address challenges](#_How_APIGEE_helped) | 3-4 |
| 5 | [Project Results](#_Project_Result) | 5 |
| 5 | [Customer Quote or Success Story Highlight](#_Customer_Quote_or) | 5 |

Project Start Date: 14th Mar 22

Project End Date: 31th Dec 23

## **Partner Role in Project**

~~Support Azure Spring Cloud, Apigee, Drupal Based developer portal and Integrated developer portal & DevOps related development activities to enhance Digital Realty APIs.~~

As a development partner, Intelliswift was responsible for APIs development on Apigee API Management, enhancement and support of Drupal based API developer Portal, design and implement Integrated developer Portal for sandbox users , implementation of Azure DevOps and microservices on Azure Spring Cloud

## **Challenges**

~~Development of Global Portal & API which allows Customer to easily consume and manage DLR’s products and services capabilities via a Single Global Experience reducing DLR support cost through self-service automation whilst promoting brand loyalty by addressing below Business & Technical challenges.~~

There were multiple region specific consumer-facing portals and segregated API catalogs which increases overall maintenance cost, disability to service with same experience to customer across portal, rolling out new product bundles seamlessly. The solution addresses following business and technical challenges by implementing a Global Portal to manage and publish all the DLR’s products and service capabilities

**Business Challenges:**

1. For Sales- difficult to market non-standard- (w.r.t Global Platform) Product / bundle Product (customized).
2. For Customer- difficult to estimate, evaluate and usage of API /Services as per their required need with high level of consistency.
3. Lack of adequate and standardized security, risk of unauthorized access /penetration in the System
4. Poor Customer experience, as they are forced to switch between multiple Portals to render services.

**Technical Challenges:**

1. ~~Due to inadequate validation risk of illegitimate incoming API call.~~
2. Risk of unauthorized access of APIs with non-standard security practices across the APIs and microservices ~~. Therefore, APIs has to be the most modern implementation from security practices perspectives.~~
3. Lack of protection against DDOS attacks and unwarranted traffic spikes ~~Risk of Rate limiting and throttling i.e., Unwanted traffic to APIs can cause DDOS attacks and brings the service down. Additionally, it incurs extra costs for the business~~.
4. Auditing and Accountability risk related to the user access and activities ~~If the User's access is not tracked in terms of service use, it could cause serious damage in-case some malicious actions are performed by the User~~.
5. Non-standardized Data Format, data types and Data Mediation to consumer data through services – ~~Change in data consumption & transformation model will bring down the Service due to the unsupported data types.~~
6. Lack of control and governance on API traffic ~~There is no control over the incoming traffic and also difficulty in removal or restricting incoming traffic.~~
7. Runtime scaling hesitancy – Difficult to make decisions about upscaling and downscaling of API runtimes, due to legit calls and not due to malicious calls for the popular API(s)
8. ~~Handling incompetent API challenge –Difficult to make decision in-order to decommission non-performance API(s) on the basis of which API getting maximum & least traffic (tracking analysis).~~
9. ~~Developer is focusing on core Service development within given development time frame and lacking focus on covering non-functional aspects like Security and performance testing like Spike test due to lack of E2E testing platform (architecture).~~
10. Non-availability of Federated and granular access controls across Internal & External users consuming APIs

### **Solutions w.r.t below focal points**

1. ~~Customer Focus~~
2. ~~Enable Growth and Product innovation~~
3. ~~Organizational excellence & profitability~~

~~Considering above solution strategy and in-order to address Business and Technical challenges DLR has introduced an additional level of layer in the form of Google APIGEE layer to build their services on Azure Spring cloud at Service Layer. Google APIGEE helps in APIs Gateway management for the API services like Monitoring, User & Rights Management, SLA Management, Monetization, Analytics, Scaling, Administration & Reporting. Implemented APIGEE layer as the customer facing layer and using Micro Services as the backend facing layer.~~

DLR partnered with Intelliswift and implemented Apigee X as API Management Platform , which manages and governs all the enterprise services hosted on Azure spring cloud , implemented security and traffic management features, API developer portal to document and market all the APIs as a global catalog. The solution strategy driven by customer focus, enablement of growth & product innovation ; and Organizational excellence & profitability

### **How APIGEE helped to solve/ address challenges:**

Apigee API Gateway platform solves

* Standardized API security by implementing OAuth 2.0 and JWT Token based authorization across all the APIs
* Enabled API traffic policies to control transaction spikes and implement quotas for customers
* Sandbox to enable customers to try out the APIs and helped DLR to market the new APIs effectively
* Insights driven through API analytics captured at API usage level to find the performing and lagging APIs across regions, popular APIs to plan for auto-scaling
* In combination with Cloud Armor secure against malicious and DDoS attack protection at the edge
* In conjunction with Google IAM, implement robust RBAC and federated user access control policies across external and internal consumers
* Faster Service development time, as the Micro Services developer has to focus only on the functionality of the service. All other aspects are taken care of by Apigee. Additionally Apigee provided analytics helped in finding incompetent or non-performing services easily, thus reduced the development time considerably

Apigee Developer Portal

* Implemented a Global API Portal to publish, market API catalogs
* Standardized documentation and convenient search of API bundles
* Promote and bundle APIs as API Products for specific regions or markets
* Insights on daily footfall on Portal, number of new developers addition, number of Consumer Apps and subscription added to streamline overall API Marketplace strategy
* ~~Inadequate validation: APIGEE has a lot of validation policies out of the box. E.g., OAuth v2, JWT, JWS, Access Control, additionally it is powered of Cloud Armor by google IAM. These policies ensure that the API calls are validated before they even enter the backend layer.~~
* ~~Unsatisfactory security: If case of any breach in security or if the admin is unsatisfied with the implemented security, multi layered security can be easily implemented over APIGEE layer without affecting the backend. Validation policies in APIGEE address almost 99.99 % of threats thus making the API secure from all aspects. Validation policy in Azure Spring cloud helps to track lots of security attacks, thus making Spring boot App services secure.~~
* ~~Use rate limiting and throttling: DDOS attacks are addressed by spike arrest policy which is out of box. Assigning no. of allowed calls to APIs is implemented by quota policy. Quota policy could be further utilized to monetize the apps.~~
* ~~Accountability: Accountability is a tricky question. Who really is accountable for what? This is achieved by RBAC, Google IAM service handles all the roles and permissions. Azure Spring cloud have authentication and authorization checks to validate who can access what information?~~
* ~~Data Formats and Data Mediation: Different data formats e.g., XML / JSON can be transformed easily without affecting backend.~~
* ~~API gateway and service mesh: APIGEE gateway will allow only authenticated traffic, as well as to control and analyse how APIs are used. Service mesh optimizes how all of these moving parts work together, including ensuring that proper authentication, access control, and other security measures are put into place. Azure spring cloud provides a one place solution to create microservice mesh, ranging from gateway, config server to Spring boot apps. Some of the features like config server gateways (in enterprise tier) are built in Spring cloud.~~
* ~~API utilization hesitancy: APIGEE Management plane has a lot of inbuilt reports. Tracking APIs usage is really simple with APIGEE. Azure Spring cloud has a support to track the usage of services, it has lots of flexible plans to use features like tracing, API usage stats, resource utilizations.~~
* ~~Handling incompetent API: Tracking a non-performing/over-performing API is cumbersome with APIGEE is really simple. Azure Spring cloud makes performance testing and tuning really simple for development team.~~
* ~~Going overboard with control: With all security, validations, data-transforms, mediations and APIGEE gateway makes onboarding process simple and secure.~~
* ~~Overall development Time: As the Micro Services developer has to focus only on the functionality of the service. All other aspects are taken care of by APIGEE. This reduces the development time considerably. Azure spring cloud makes App deployment fairly simple, features like tracing, config server, load balancing is taken care by Azure making developer to focus on core development. This helps to reduce the development time.~~
* ~~Access provisioning and granting privileges to end Users (Internal & External) as per Role: As the developer focuses only on the core functionality and APIGEE addresses all the security, mediation, traffic management.~~

### **Project Result**

A single Global Portal will make doing business with DLR effortless, empowering Customers to be self-sufficient in ordering & managing Products and Services while creatively positioning new offerings that could support their business, drive revenue and reduce operating costs thru DLR 's platform. Also helps Financial personnel and DLR personals who are providing support to Customer.

Target Audience benefitted with:

1. Ease usage: Having standardize APIs catalogue - End user easily select Product / bundle Products as per the need.
2. Security: Enhanced Security of APIs - As per end User Role easy to provision access and privilege’s as per Functionals Area/ selected Product.
3. Easy in Scalability & less effort in maintenance -Reducing Development /Customization effort by implementing Automation of Swaggers (save ~30% development effort); thereby reducing operational cost.
4. Sales & Marketing: Allows DLR Customers to easily consume and manage DLR's products and services capabilities via a single Global Experience; reducing support cost through self-service automation.

### **Customer Quote or Success Story Highlight:**

(Customer Feedback / review for us)