Apigee Solution Architecture

<Customer>, <Project Name>

Author

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <date> | <version> | <description> | <author> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

Purpose 1

Scope and Context 1

Business Objectives and Constraints 1

Objectives 1

Constraints 1

Assumptions 1

Risks 1

Use Cases 2

<Use Case Name> 2

EXAMPLE: Authentication Using Oauth 2.0 3

Solution Overview 4

Architecture Guiding Principles 4

Solution Design 5

Solution Components 5

Apigee API Services (Edge) - Applications 5

Apigee Analytics Services 5

Apigee Monetization Services 5

Apigee Developer Services 5

Apigee App Services 5

<Component Name> 5

API Specification 5

Infrastructure Architecture 6

Platform Environment Specification 6

Apigee Environment Mappings 6

Apigee Virtual Host Definitions 6

Appendix 7

Key Concepts and Terminology 7

Supporting Documents 7

Purpose

This purpose of this document is to describe the overall Apigee solution at a high level. It states the problem definition(s) and outlines the design, architecture, and interactions of the Apigee solutions with other components. There are supporting documents which provide a greater level of detail for some sections within this document. You can find reference of these documents

Scope and Context

Business Objectives and Constraints

Objectives

[Summarize the challenge. What are the business objectives the solution addresses?]

Constraints

[List any know business or technical constraints considered on the solution]

Assumptions

[List any know business or technical constraints considered on the solution]

Risks

[List any know risk considered on the solution]

Use Cases

<Use Case Name>

[Describe the high-level interaction between actors and Apigee products. Describe scenarios and variance of given use cases. These are high-level descriptions from the functional point of view. An example would be overall use case and sequence diagram to achieve Oauth 2.0 – Authorization Code Grant Type, API Response Caching, REST to SOAP Transformation, etc.]

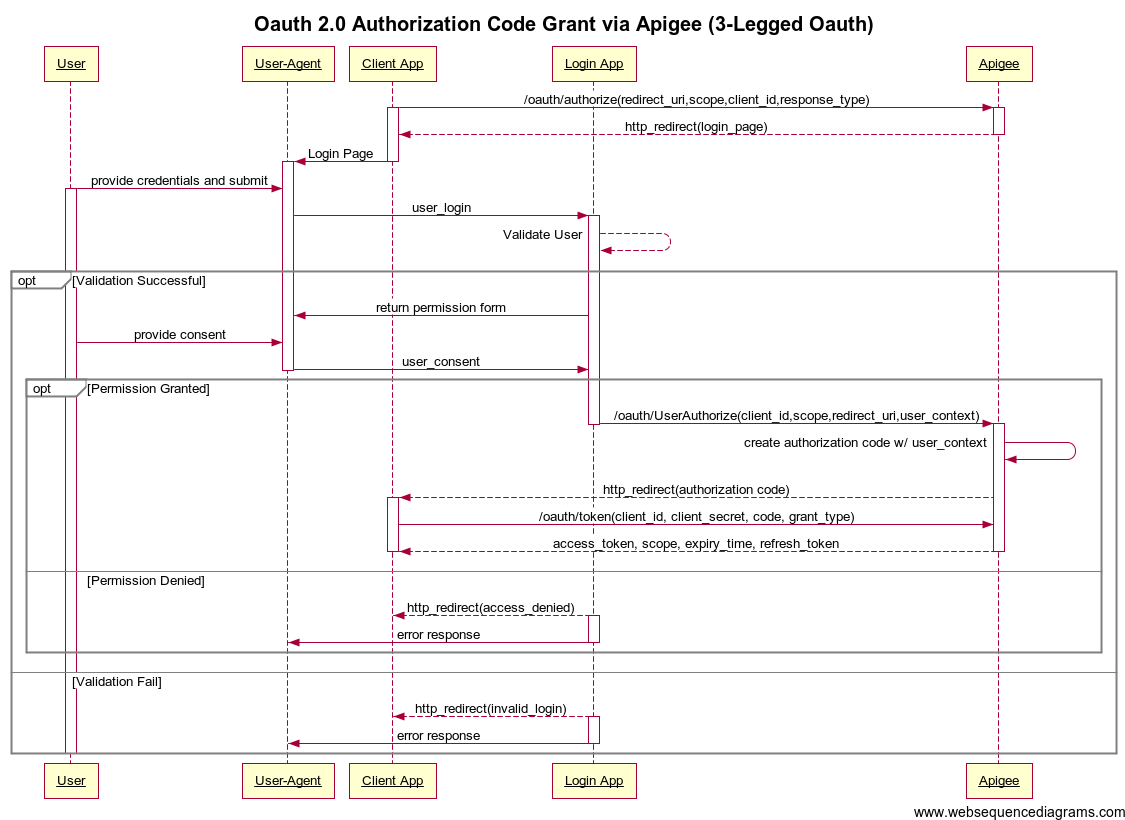
Figure 1. Use Case Diagram

Place holder for use case diagram

Show key use cases and actors. The diagram provides a visual representation of the use cases described on the tables below.

|  |  |
| --- | --- |
| Use Case Name | <Unique short descriptive name> |
| Actors | <List of the Actors who communicate with the Use Case> |
| Pre-Conditions | <List of conditions that must be true before the Use Case starts> |
| Success End Condition | <How do we know it works> |
| Failed End Condition | <How do we know it fail> |
| Post-Conditions | <List of conditions that must be true when the Use Case ends,  no matter which Scenario is executed. Example:   * Analytics data that must be captured.   State of data sources that needs to be updated.> |
| Basic Flow | 1. <> 2. <> 3. <> 4. <> 5. <> |
| Extension Points |  |

EXAMPLE: Oauth 2.0 – Authorization Code Grant Type



|  |  |
| --- | --- |
| Use Case Name | Oauth 2.0 – Authorization Code Grant Type |
| Actors | User-Agent, Client App, Login App, Apigee Edge |
| Pre-Conditions | Client App must have access to a user-agent and valid App credentials from Apigee Edge. |
| Success End Condition | Client App obtains an accesstoken. |
| Failed End Condition | Client App does not obtain an access token and receives an error response during the flow. |
| Post-Conditions | N/A |
| Basic Flow | 1. Client begins authentication process by sending a request to Apigee to authorize. 2. Apigee validates client\_id and redirects the client app to a login page. 3. Client app opens the login page via a User-Agent (web browser). 4. User logs in. 5. Login App validates user and, if successful, prompts for consent. 6. Upon user consent via User-Agent, the login app makes another request to Apigee with user information. This triggers the authorization code creation with user context as a stored attribute for the code. 7. Apigee sends a redirect to the registered redirect uri for the Client App. 8. Client App retrieves authorization code. 9. Client App makes request for access token using the authorization code from step 8. 10. Apigee validates the token request and generates a token, keeping the same user context from the authorization code as a stored attribute. 11. Apigee sends token response. 12. Client App receives access token. |
| Extension Points |  |

Solution Overview

This section describes the overall interaction between components such as: client applications, Apigee components, backend services, 3rd party services. High level diagrams illustrating the component interactions are provided below.

[Provide one or more architecture views of the solution(s).]

Placeholder for solution diagrams high level view.

Provide diagram showing the exposed API and consumed backend resources/services. Keep diagram focus on APIs and resources, center on the flow of information. Show relationship between Apigee products, customer and third party API/Services.

Architecture Guiding Principles

This section provides the key architecture principles used with the Apigee solution(s). This includes, but is not limited to: solution standards and protocols, design principal agreements.

[Provide any guiding principles here.]

Solution Design

Solution Components

This section describes the key elements of the Apigee solution. For each component, an overview of its role and integration with the complete solution is provided. Additional details regarding a component may be referenced in a supporting document.

Apigee API Services (Edge) - Applications

The detailed architecture of a developed Apigee application will be captured in the API Proxy Design and Architecture. That documentation lives with the source code as it is directly related to development. Additionally, abstracted information related to a solution’s architecture is present in the above use cases. Below is a list of all Apigee API Proxy applications for reference in other documentation:

|  |  |
| --- | --- |
| API Proxy | Solution Overview |
| <API Proxy Name> | [Provide a brief description of the API Proxy and the Apigee solution provided.] |
|  |  |
|  |  |
|  |  |

Apigee Analytics Services

This section provides an overview of any custom Apigee Analytics reports. It will describe the report’s measures and explain what is represented by the data.

|  |  |
| --- | --- |
| AX Report Name | Description |
| <AX Report Name> | [Provide a description of the reports measures and explain what’s represented by the data.] |
|  |  |
|  |  |
|  |  |

Apigee Monetization Services

[Provide an overview of how Monetization fits into the overall Apigee solution.]

Apigee Developer Services

[Provide an overview of how the Developer Portal fits into the overall Apigee solution.]

Apigee App Services

[Provide an overview of how App Services fits into the overall Apigee solution.]

<Component Name>

[Provide an overview of how <component> fits into the overall Apigee solution.]

API Specification

The specification for all APIs is captured in the API Specification Document. The solution described in this documents has been designed to support the implementation of those APIs. Refer to the Supporting Documents section of the Appendix for the location of the API Specification document.

<Add link or reference to the API Specification Document. Avoid copying the API Specification on the Solution Architecture document. The API Specification should be considered a relatively stable and controlled document since it defines contract interfaces consumed by internal and third party applications. For these and other reasons, the API Specification should remain as a separate deliverable>

Infrastructure Architecture

Platform Environment Specification

In the case of on-premise Apigee installations, the infrastructure architecture is captured in the Environment Specification document. Refer to the Supporting Documents section of the Appendix for the location of the Environment Specification document.

For solutions using the Apigee platform hosted in Apigee’s cloud, this document is absent since the cloud infrastructure is solely managed and operated by Apigee.

Apigee Environment Mappings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Environment | Apigee Organization | Apigee Environment | External Host | Apigee Host |
| Dev | customer-non-prod | dev | dev.api.example.com | apigee-router.example.internal:8443 |
| QA | customer-non-prod | qa | qa.api.example.com | apigee-router.example.internal:8443 |
| Prod | customer-prod | prod | api.example.com | apigee-router.example.internal:8443 |
|  |  |  |  |  |

Apigee Virtual Host Definitions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Environment | Virtual Host Name | Host Alias | Port | SSL (Y or N) |
| Dev | default | dev.api.example.com | 8443 | Y |
| QA | default | qa.api.example.com | 8443 | Y |
| Prod | default | api.example.com | 8443 | Y |
|  |  |  |  |  |

Appendix

Key Concepts and Terminology

|  |  |
| --- | --- |
| Term/Concept | Description |
| <Concept> | <Describe the meaning to facilitate understanding of audiences not familiar with Apigee terminology or specific terms adopted by the project> |
|  |  |
|  |  |

Supporting Documents

|  |  |  |
| --- | --- | --- |
| Document Title | Description | Location |
| Infrastructure Architecture | This document captures the Apigee platform infrastructure, including all environment specifications. | [Provide location of the document] |
| API Proxy Design and Architecture | This documentation details the specific API proxy solution architecture. It includes policy, routing, backend endpoint, and fault handling documentation. | This documentation is located with the Apigee source code in the Github repository. It is provided in markdown format. [Provide link to github repository here] |
| API Specification | This documentation details the API specifications, including all operations, error codes, service endpoint mappings. | [Provide location of the document] |
| Developer Portal Specification | This document provides details of the customizations used in Apigee’s developer portal. | [Provide location of the document] |
| <Add Additional Documents here> |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |