

SUMMIT

# API Governance

An Agile aproach

AppyThings

.Inspire .Innovate .Perform .Adapt



# Why API Governance ?

API {1<sup>st</sup>} is not just a new tool. It is about opening up processes via a simple stateless interface so people can easily integrate with them.

This only works when API's are simple, easy to understand, universal and consistent.

That is where API Governance comes in to play: To secure that we keep on the right track.



3

## People

Who's Doing Stuff

*Innovate*

*Scale*

WIN

*Automate*

## Process

How Stuff Is Done

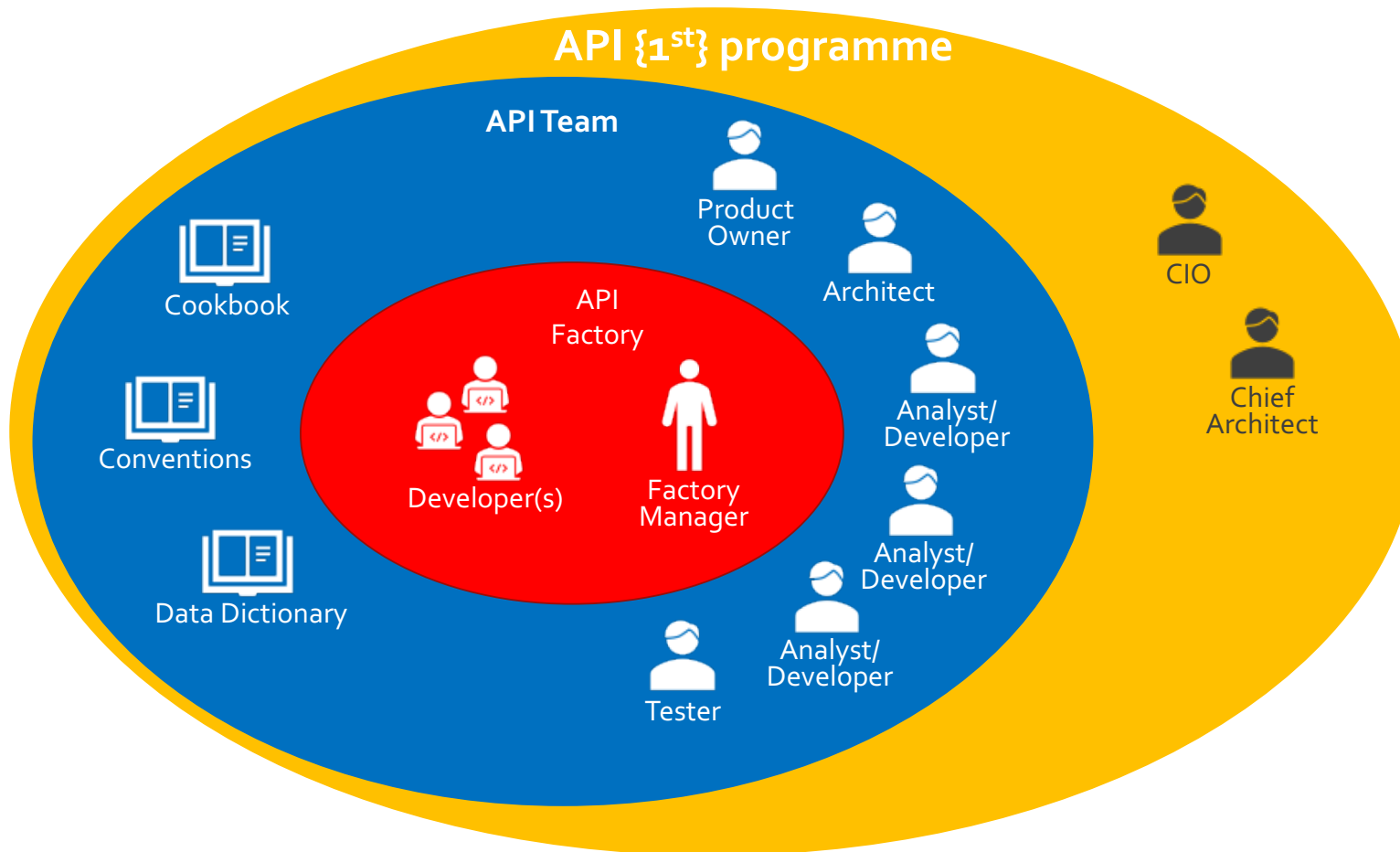
## Technology

What We Do Stuff With

# GOVERNANCE BUILDING BLOCKS



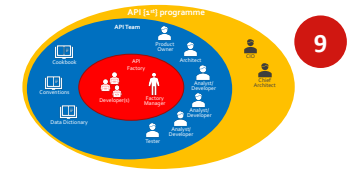
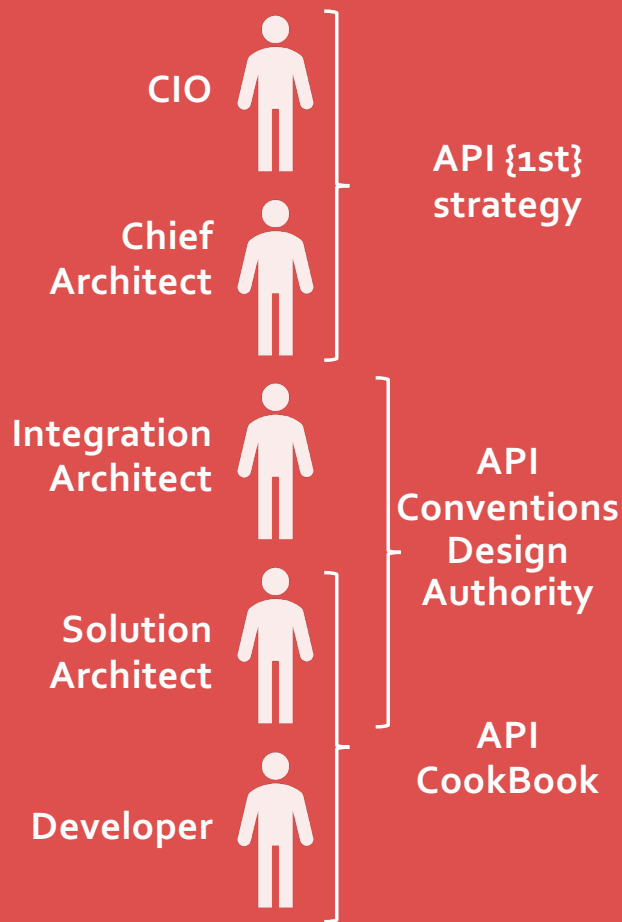
# The API Factory model



7



PEOPLE



Position	Role	Comment
CIO	Sponsor	The CIO mandates the strategic change
Chief Architect	Sponsor	The Chief Architect is the evangelist for the API {1 <sup>st</sup> } strategy.
Integration Architect	Design Authority	Responsible for the communication about Conventions, Cookbook, etc. within the organization
Solution Architect	Design Authority	Responsible for the exception handling and creation of new conventions / patterns.
Developer	API Team	Hands-on assistance to the Back-End teams for creating good API's

## 2 strategies on the API team

Depending on the culture and the maturity of the Dev/Ops teams one model might work better than the other.

### 1. Centralized API team

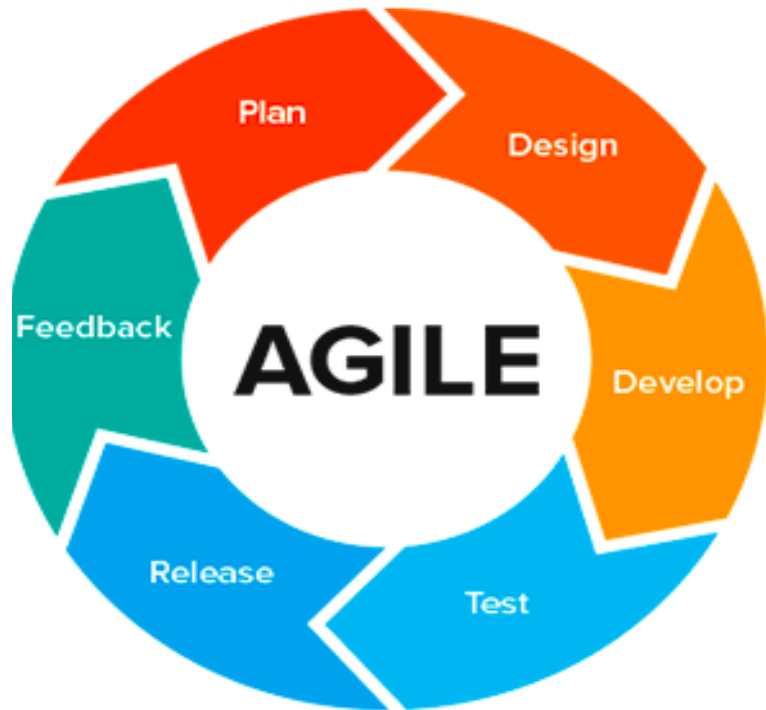
- ➕ The centralized API team has a built in objective to enforce the standards and policies that are stated in the three governance products (Conventions, Cookbook and Data Dictionary)
- ➖ The centralized API team might be too far away from the actual Dev/Ops team to work optimally

### 2. Integrated within Agile Teams

- ➕ Fits best with the philosophy of actual Dev/Ops
- ➖ The Dev/Ops team might prevail their own priority before the implementation of the standards and policies.
- ➖ The Dev/Ops team might not be able to keep the knowledge of the tool up to date because of the limited effort on the API in respect to the actual back end work

What most of our customers chose is *Model 1. Centralized API team* and they compensate the "distance" to the team by a so called wanderer. The API team member that works for the Dev/Ops team is temporarily part of the Dev/Ops team



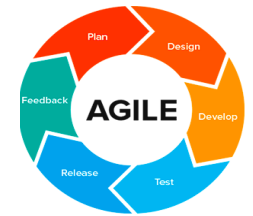


# PROCESS

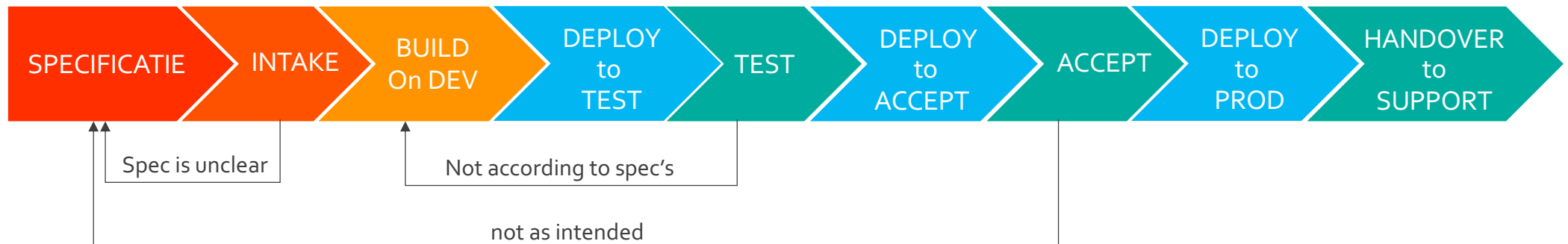


# IT build process (using AGILE/SCRUM)

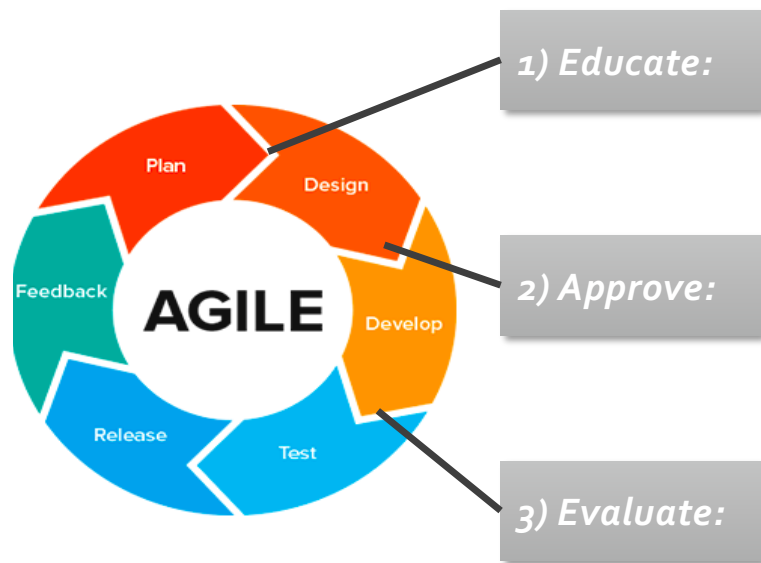
# 4 environment process steps



14

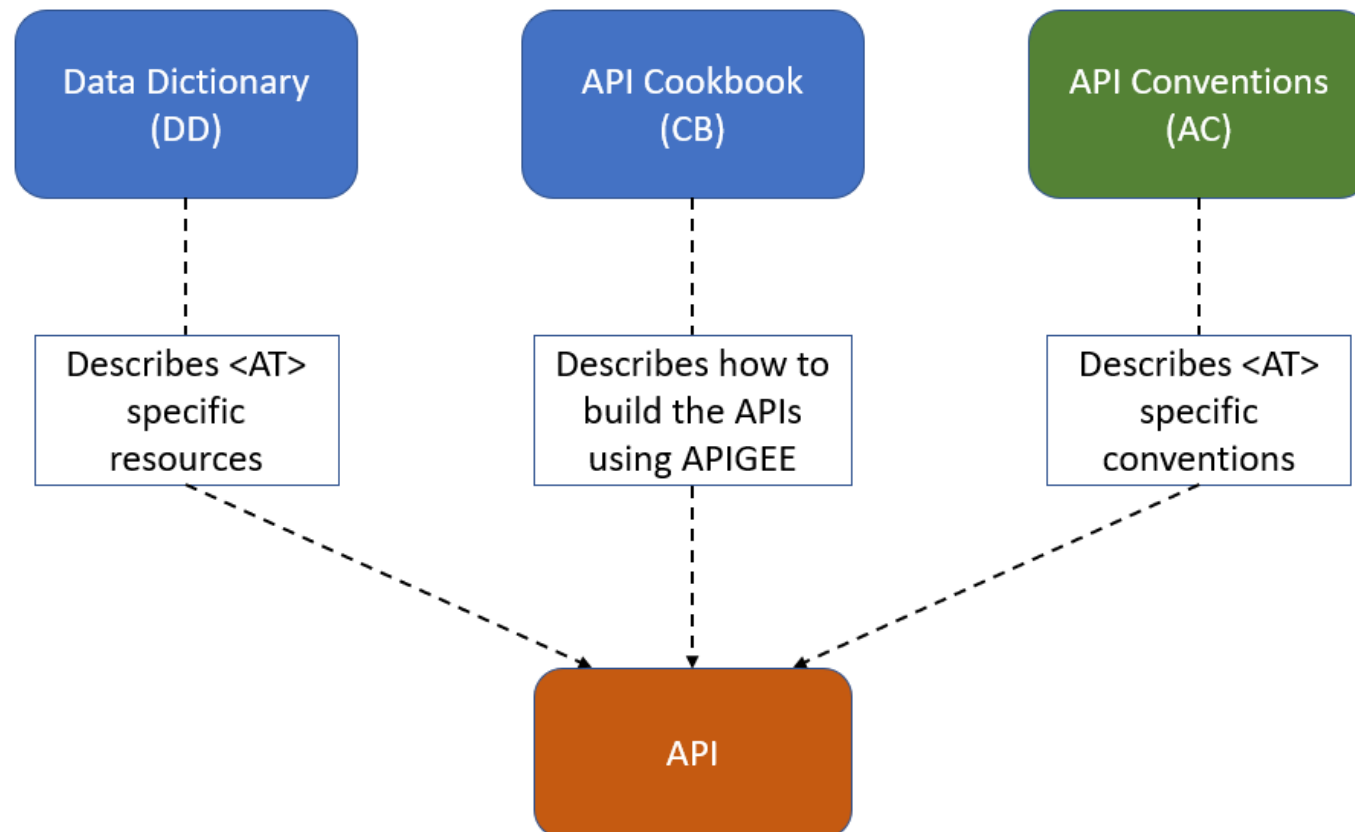


# API Governance process

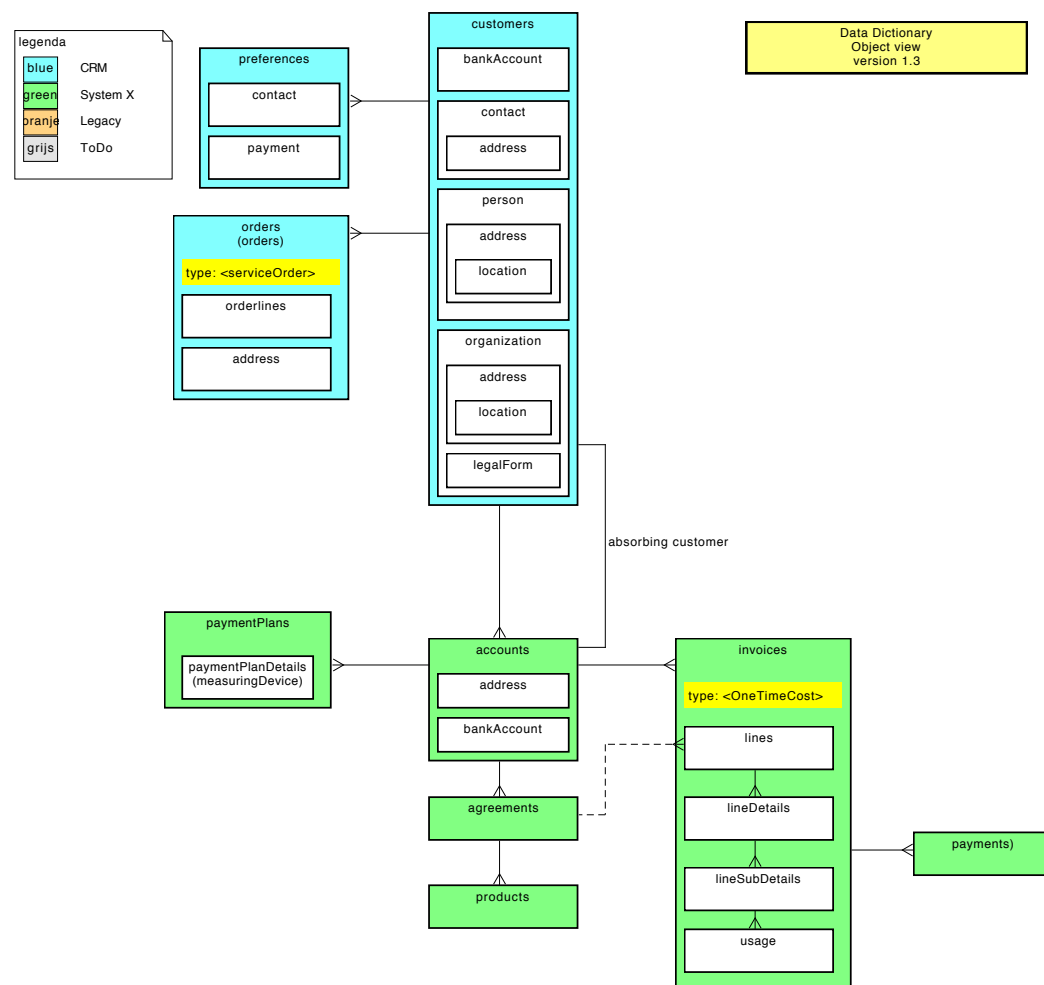


- 1) ***Educate:*** A main part of the Governance process is to consult and to communicate. We provide the standards like: Conventions, CookBooks and DataDictionary. If needed we organize trainings for people to get them up to speed.
- 2) ***Approve:*** We review each design to see if they implemented the Conventions, CookBook and DataDictionary. We'll mediate when exceptions are required. We'll help when new Conventions or Patterns are needed
- 3) ***Evaluate:*** We evaluate if everything has been build according to standards.

# Building Blocks of API Governance



# Content of API DataDictionary



# Content of API Cookbook

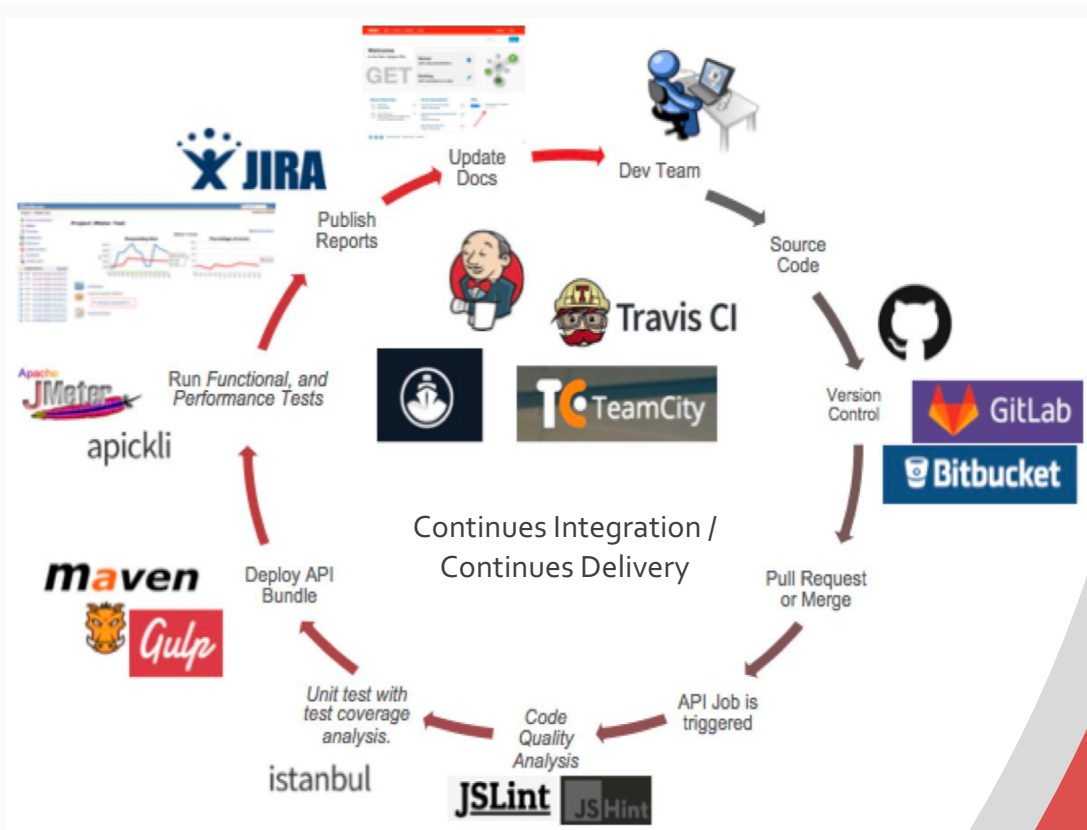
Introduction .....	6
<b>1 Structure and Flows .....</b>	<b>7</b>
1.1 What goes where .....	7
1.1.1 (A) Proxy Request quadrant .....	8
1.1.2 (B) Target Request Quadrant .....	8
1.1.3 (C) Target Response Quadrant .....	9
1.1.4 (D) Proxy Response Quadrant .....	9
1.2 Flows .....	10
1.2.1 Conditional Flows .....	10
1.2.2 Shared flows .....	11
1.2.3 Special flows .....	12
<b>2 Relations in Apigee .....</b>	<b>13</b>
2.1.1 Specification .....	13
2.1.2 The API product .....	14
2.1.3 The developer .....	14
2.1.4 The developer app .....	14
<b>3 APIGEE Naming Conventions .....</b>	<b>15</b>
3.1 API policies .....	15
3.2 Shared flows and flow hooks .....	15
3.3 Products .....	16
3.4 Specification .....	17
3.5 Developer app (application) .....	17
3.5.1 Test Application .....	17

3.6 Target Server .....	18
3.7 Key-value maps .....	18
<b>4 Policies .....</b>	<b>19</b>
4.1 What are Policies .....	19
4.2 Naming policies .....	19
4.3 Security Policies .....	19
4.3.1 Verify API key .....	19
4.3.2 Basic Authentication for back end .....	20
4.4 Traffic Management Policies .....	21
4.4.1 Spike Arrest .....	21
4.4.2 Quota .....	21
4.4.3 Cache .....	22
4.5 Mediation Policies .....	24
4.5.1 Raise Fault .....	24
4.5.2 Extract Variables .....	24
4.5.3 Assign Message .....	25
4.6 Extension Policies .....	26
4.6.1 JavaScript .....	26
4.6.2 Service Callout .....	27

4.7 Message Logging .....	28
4.7.1 Creating a Message logging policy .....	28
<b>5 Error Handling .....</b>	<b>30</b>
5.1 Flows .....	30
5.1.1 DefaultFaultRule .....	30
<b>6 Swagger / Specification .....</b>	<b>31</b>
6.1 CORS headers .....	31
<b>7 Environment Configuration .....</b>	<b>33</b>
7.1 Target Servers .....	33
<b>Appendix A: Quick API setup .....</b>	<b>34</b>
Step 1 – Generate the API .....	34
Step 2 – Standard configurations .....	37
Step 3 – Filling in the details .....	42

# Content of API Conventions

<b>1</b>	<b>Introduction .....</b>	
<b>2</b>	<b>Types of API's .....</b>	
<b>3</b>	<b>Security .....</b>	
	<b>3.1 South bound security .....</b>	
<b>4</b>	<b>Naming Conventions .....</b>	
	<b>4.1 Language .....</b>	
	<b>4.2 URLs .....</b>	
	4.2.1.1 Internal API's .....	
	4.2.2 Path parameters versus Query parameters .....	
	<b>4.3 API .....</b>	
	4.3.1 API-Naming .....	
	4.3.2 API-Versioning .....	
	<b>4.4 Caches .....</b>	
	<b>4.5 Objects, properties, JSON structures etc. ....</b>	
	<b>4.6 APIGEE specific naming conventions .....</b>	
<b>5</b>	<b>HTTP Verbs .....</b>	<b>14</b>
<b>6</b>	<b>HTTP Headers .....</b>	<b>15</b>
	6.1 Content-type versus Accept .....	15
	6.2 Apikey/ .....	15
	6.3 Authorization .....	15
<b>7</b>	<b>Error handling .....</b>	<b>16</b>
	7.1 HTTP codes .....	16
	7.2 Error messages .....	17
<b>8</b>	<b>JSON versus SOAP .....</b>	<b>18</b>



# TECHNOLOGY





**.Inspire .Innovate .Perform .Adapt**

MEMBER OF APPY ENTERPRISE