



Effectively Managing Mule API Versions and Portal Site

Yuan Meng – Sr. Solutions Architect, Perficient Inc.
Email: yuan.meng@perficient.com

05/08/2018 San Jose

- Powerful API Platform with reach features
- Leader in Gartner's top magic quadrant
- API Design Center, RAML editor
- Anypoint Exchange
- Mule API runs *In Cloud, on-Premise and Hybrid*

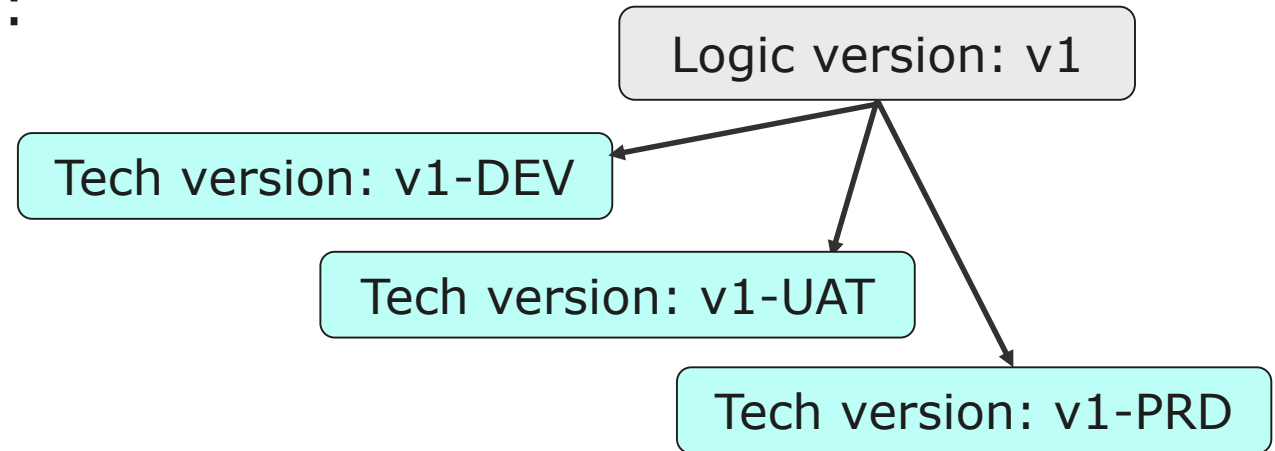
- API Manager 1.x (prior to Crowd Release)
 - The multitude of API versions create challenges to manage the APIs with 1.x
 - A systematic way to manage the versions with CI / CD is strongly desired.
- API Manager 2.x (Crowd Release and Mule 4 Release)
 - differentiates APIs per each environment
 - Starting API from Design Center and Exchange
- Knowledge of API Manager 1.x helps understanding and appreciation of API Manager 2.x!

challenge

goal

Every good API starts with a version:

```
RAML
  version: v1
  /foo:
    get:
  /bar:
    post:
```



This *logic version* in RAML declares the API interface (contract).

APIs can go through version changes. When the same API deploys to multiple environments, each environment has a different *technical version*, but they are just an incarnation of the same logic API version.

Technical versions should be hidden from API consumers & developers.

Two Ways to Deploy an API

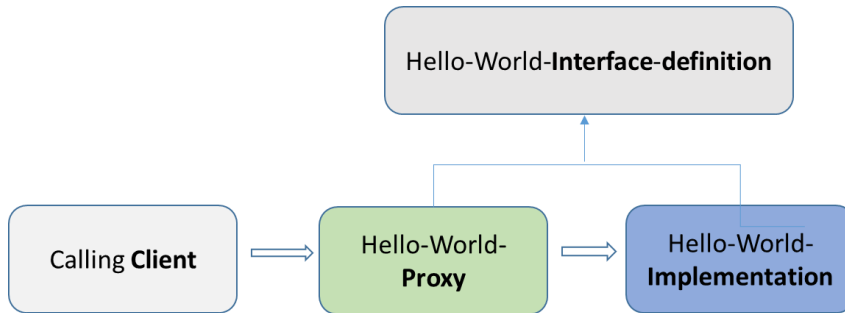


Fig. 1 – Proxy Routes to Implementation

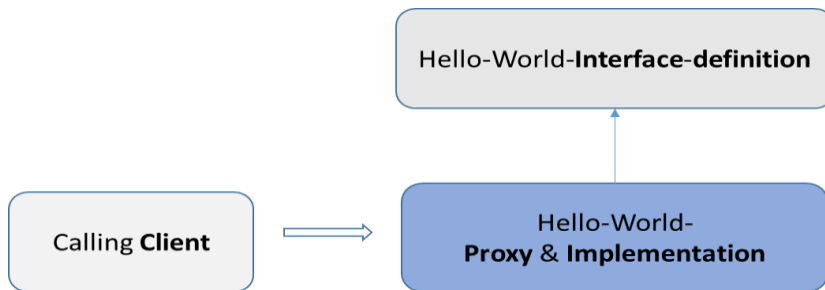


Fig. 2 – Merge Proxy & Implementation

- Modern Web API can still be viewed with the traditional Interface-Proxy-Implementation paradigm
- API RAML is the counter part of traditional interface definition (such as IDL, WSDL, or simply Java interface)
- API proxy and implementation still fit into the classic framework

Two Ways to Deploy an API:

- Endpoint with proxy – fig.1
- Basic endpoint – fig.2

Auto-discovery is based on 3 key factors:

1. API RAML (technically not required)

two key attributes: **apiName**, **version**

2. Auto Discovery Tag: `<api-platform-gw:api apiName="myHelloAPI" version="v1-
${env}" apikitRef="proxy-config" flowRef="proxy" doc:name="API Autodiscovery"/>`

3. Environment properties

```
anypoint.platform.analytics_base_uri=https://analytics-  
ingest.anypoint.mulesoft.com ...
```

```
anypoint.platform.client_id=75xxxxxxxxxxf410
```

```
anypoint.platform.client_secret=10xxxxxxxxxx1BF5
```

Cloudhub: defined under properties tab

OnPrem: defined in wrapper.conf

Mule API Manager relies on Auto-discovery to manage the APIs

- Endpoint with a Proxy: auto-discovery is inserted by the API manager
- Basic endpoint: need manually adding auto-discovery

API Version Naming Convention – 1.x

	myHelloAPI	version	<api-platform -gw:api> tag	Deployment Endpoint URL	
				On-Premise	CloudHub
1	Design Doc	v1			
2	RAML	v1			
3	DEV	v1-dev	v1- <code>{env}</code>	dev-hostname/myHelloAPI/v1	myHelloAPI-dev.cloudhub.io/v1
4	QA	v1-qa	v1- <code>{env}</code>	qa-hostname/myHelloAPI/v1	myHelloAPI-qa.cloudhub.io/v1
5	PROD	v1-prod	v1- <code>{env}</code>	prod-hostname/myHelloAPI/v1	myHelloAPI.cloudhub.io/v1

For the "version" column:

row #1 the version that appears in your design document. This is the **logic version**.

row #2 this is the RAML version, it is same logic version as in the API design doc.

row #3 - #5, these are the **technical versions**.

For column 2 and 3:

row #3 - #5: these are the technical versions in your API source code. They are supposed to be the same as the "version" column.

For "On-Premise" and "CloudHub" columns:

row #3 - #5: they should be named consistently as in the previous columns.

POM Snippet for Deployment

```
<plugin>
  <groupId>org.mule.tools.maven</groupId> <artifactId>mule-maven-plugin</artifactId> ...
  <configuration> <deploymentType>cloudhub</deploymentType>
    <muleVersion>3.8.5</muleVersion> <username>${user}</username> <password>${pwd}</password>
    <redeploy>true</redeploy><environment>${env}</environment><workerType>Micro</workerType>
    <applicationName>${artifactId}-${env}</applicationName>
    <properties>
      <env>${env}</env>
      <anypoint.platform.analytics_base_uri>...
      <anypoint.platform.platform_base_uri>...
      <anypoint.platform.coreservice_base_uri>...
      <anypoint.platform.client_id>7517bxxxx...
      <anypoint.platform.contracts_base_uri>...
      <anypoint.platform.client_secret>...
    </properties>
  </configuration>
  <executions> <execution>
    <id>deploy</id><phase>deploy</phase>
    <goals><goal>deploy</goal></goals>
  </execution></executions>
</plugin>
```


Viewing the Versions in API Manager 1.x Console

The screenshot displays the API Manager console interface. At the top, the header includes the API Manager logo, the user 'Perficient Inc', and a notification icon. The main content area is titled 'API Administration' and shows a list of API versions for 'hello-conn-api'. The versions listed are v1-PROD, v1-DEV, v1-UAT, and v1, all with an 'Active' status and a 'Public portal' link. A sidebar on the left contains navigation options like 'API Administration', 'Client Applications', and 'Alerts'. A right-hand panel provides details for the selected 'v1' version, including the owner 'Yuan Meng', a 'Download RAML files' button, and tabs for 'Applications', 'Policies', and 'SLA tiers'. A message at the bottom of the right panel states, 'There are no applications for this API version.'

Version	Status	Portal
v1-PROD	Active	Public portal
v1-DEV	Active	Public portal
v1-UAT	Active	Public portal
v1	Active	Public portal

With Maven, CI/CD and API auto-discovery, this version scheme is automated without manually tweaking the API versions from the API Manager console.

Each logic API version should have a unique portal site that defines the common interface

- API for each environment can potentially have its own portal site.
- It is best practice all technical versions (dev-qa-prod) of the same API version ("v1") share the same portal site.
- It is possible that this canonical API can have portal site, and without any deployment behind it (as shown in the next slide).
- Developers may choose to use one of the "technical" versions (such as "v1-prod") to establish the canonical portal site, other versions can share the same portal.

Viewing the Portals in API Manager 1.x Console

The screenshot displays the API Manager console interface. At the top, the header shows 'API Manager' and 'Perficient Inc'. The main content area is titled 'API Administration' and shows a list of API versions for 'hello-conn-api'. The versions listed are v1-PROD, v1-DEV, v1-UAT, and v1. Each version is marked as 'Active' and has a 'Public portal' link. Red arrows point from the 'Public portal' links of the v1-PROD, v1-DEV, and v1-UAT versions to the 'Public portal' link of the v1 version, indicating that all three versions share the same canonical portal site. The right sidebar shows details for the selected v1 version, including the owner 'Yuan Meng' and a 'Download RAML files' button. Below the sidebar, there are tabs for 'Applications', 'Policies', and 'SLA tiers', and a message stating 'There are no applications for this API version.'

Version	Status	Portal
v1-PROD	Active	Public portal
v1-DEV	Active	Public portal
v1-UAT	Active	Public portal
v1	Active	Public portal

Example: dev, uat and prod share the same canonical portal site

API Manager 2.x – Crowd Release / Mule 4 Release

API Administration (DEV) new-hello-api (v1) - Settings

DEV

API Administration

Alerts

Client Applications

Policies

SLA Tiers

Settings

new-hello-api v1

API Status: Unregistered Asset Version: 1.0.0 Type: RAML/OAS [+ Add consumer endpoint](#)

API Instance ⓘ

ID: 12713284

Label: [+ Add a label](#)

Autodiscovery ⓘ

API Name: groupId:318fd208-83be-4277-a95a-a8decc25f6ba:assetId:new-hello-api

API Version: v1:12713284

Mule 4

Crowd Release
API Mgr 2.x Runtime 3.x

API Configuration ▾

[ADD A TAG](#)

Managing type: Basic Endpoint Endpoint with Proxy

- <https://docs.mulesoft.com/api-manager/v/1.x/api-auto-discovery>
- <https://blogs.perficient.com/2017/01/27/technically-what-makes-a-mule-application-an-api/>
- <https://blogs.perficient.com/2017/10/04/what-is-in-a-mule-api-version/>

Q&A

Thank you!