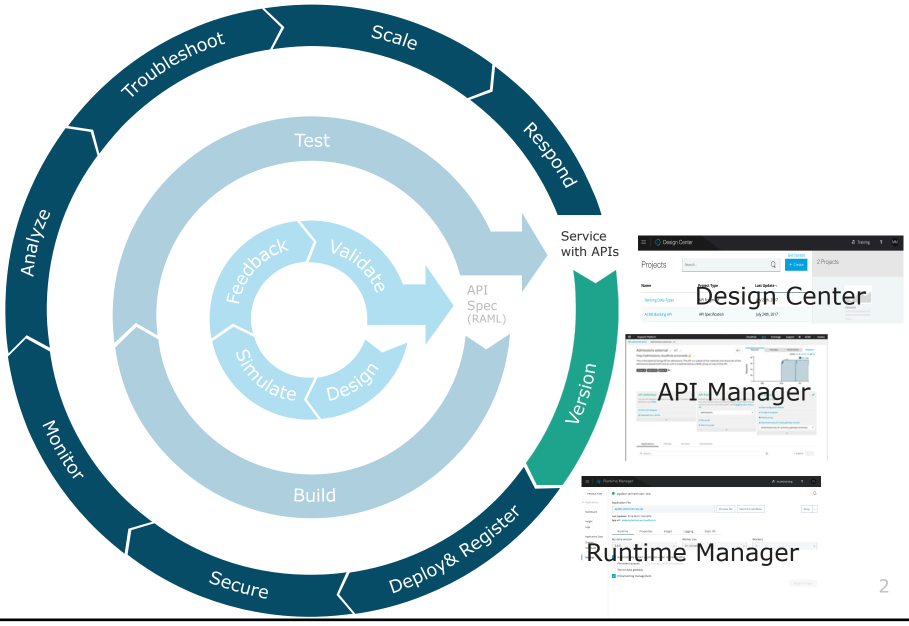
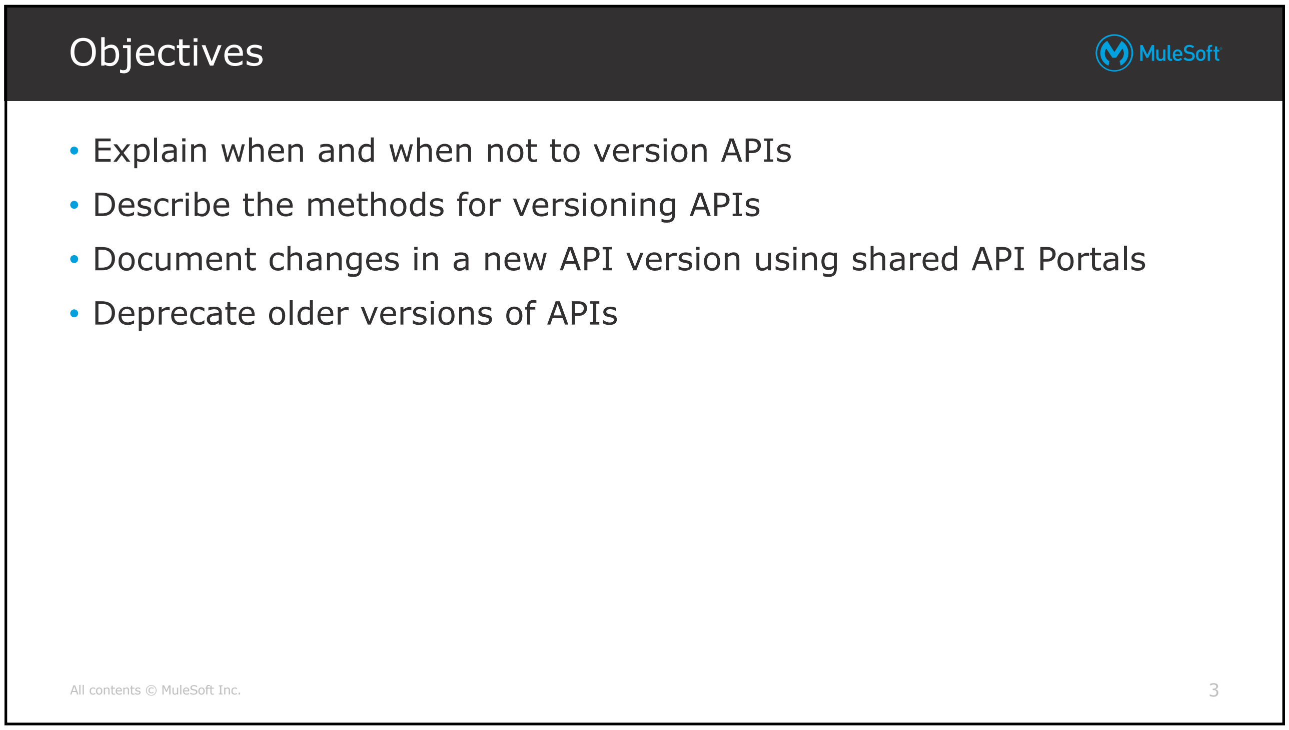
Module 13: Versioning APIs

Goal



2



Objectives

MuleSoft

- Explain when and when not to version APIs
- Describe the methods for versioning APIs
- Document changes in a new API version using shared API Portals
- Deprecate older versions of APIs

Versioning APIs

Determine when and if to version an API

- Version when additions or changes to an API break existing client code or changes the API interface rendering client code to fail
- Versioning helps to handle future changes even if you do not know what those changes are yet
- **The best practice is to version as little as possible**
  - When possible, add to the existing service in a non-breaking manner
  - Don't version APIs for a basic underlying data model change

Versioning throughout the API lifecycle

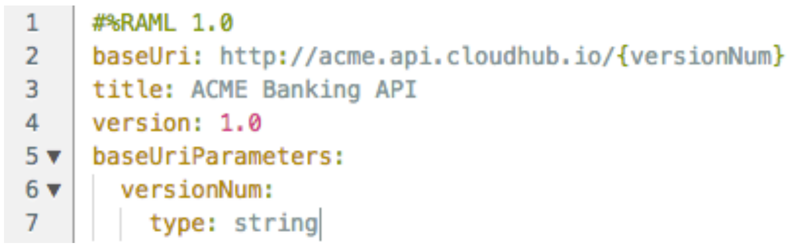
- During development
  - You will likely have to make adjustments to the RAML API definition as you deal with the realities of backend changes
  - Versioning is not the answer if an API is still under development
- While updating/deleting existing resources and methods
  - Does the flow need to change? If yes, alter the existing flow
  - Remove resource from RAML definition and flows in the implementation, while deleting resources
  - Do not version the API if updating/deleting resources does not change the API interface rendering the client code to fail

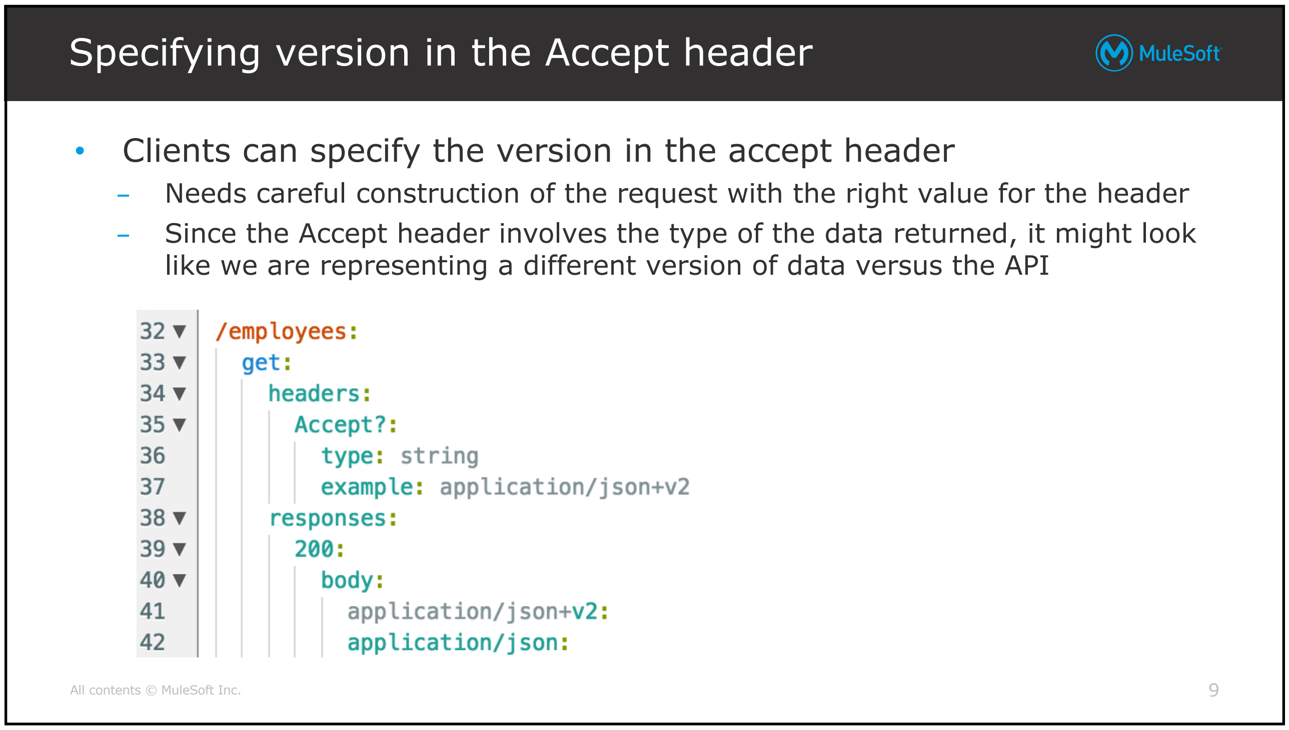
Ways to implement versioning

- Add the version number to the URL
- Add a custom request header with the API version
- Modify the accept header to specify the version

Adding a version number to the URL

- Use the version number in the baseURI or in resource path
- Easy to view and use


```
#%RAML 1.0
baseUri: http://acme.api.cloudhub.io/{versionNum}
title: ACME Banking API
version: 1.0
baseUriParameters:
  versionNum:
    type: string
```

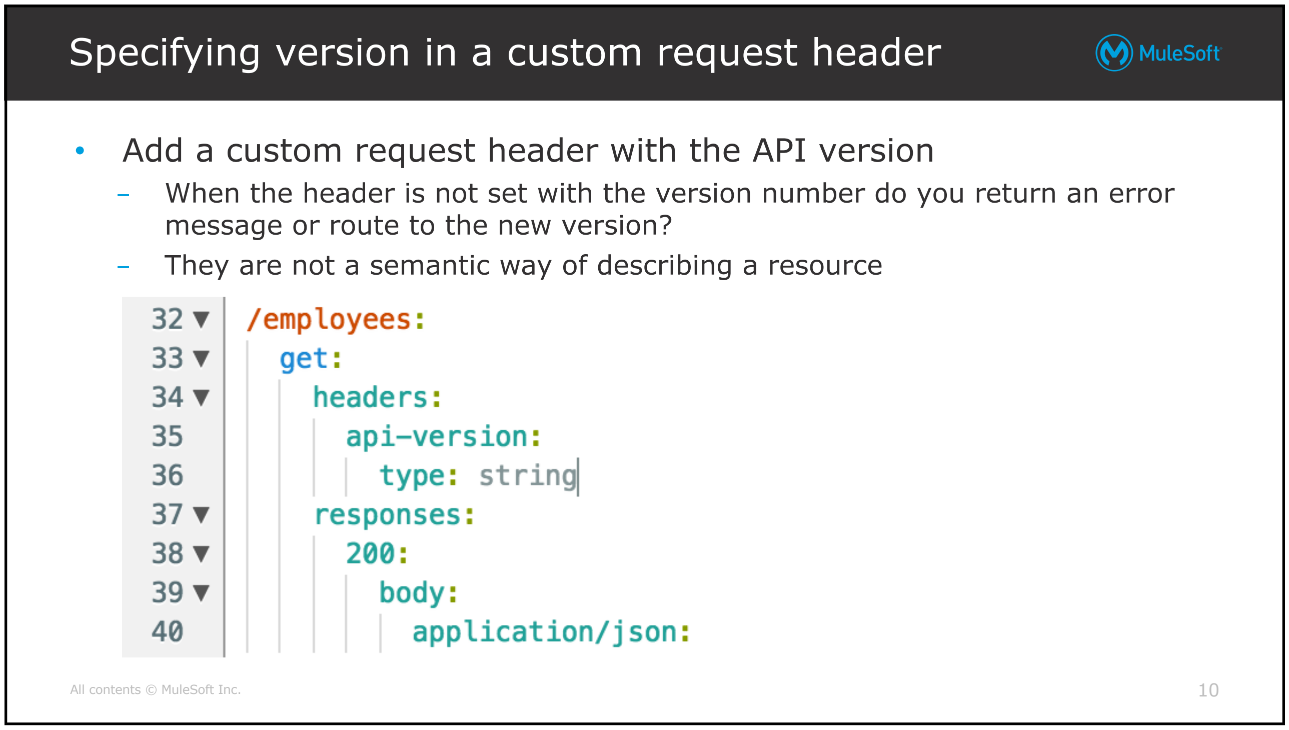


Specifying version in the Accept header

MuleSoft

- Clients can specify the version in the accept header
  - Needs careful construction of the request with the right value for the header
  - Since the Accept header involves the type of the data returned, it might look like we are representing a different version of data versus the API

```
/employees:
  get:
    headers:
      Accept?:
        type: string
        example: application/json+v2
    responses:
      200:
        body:
          application/json+v2:
          application/json:
```



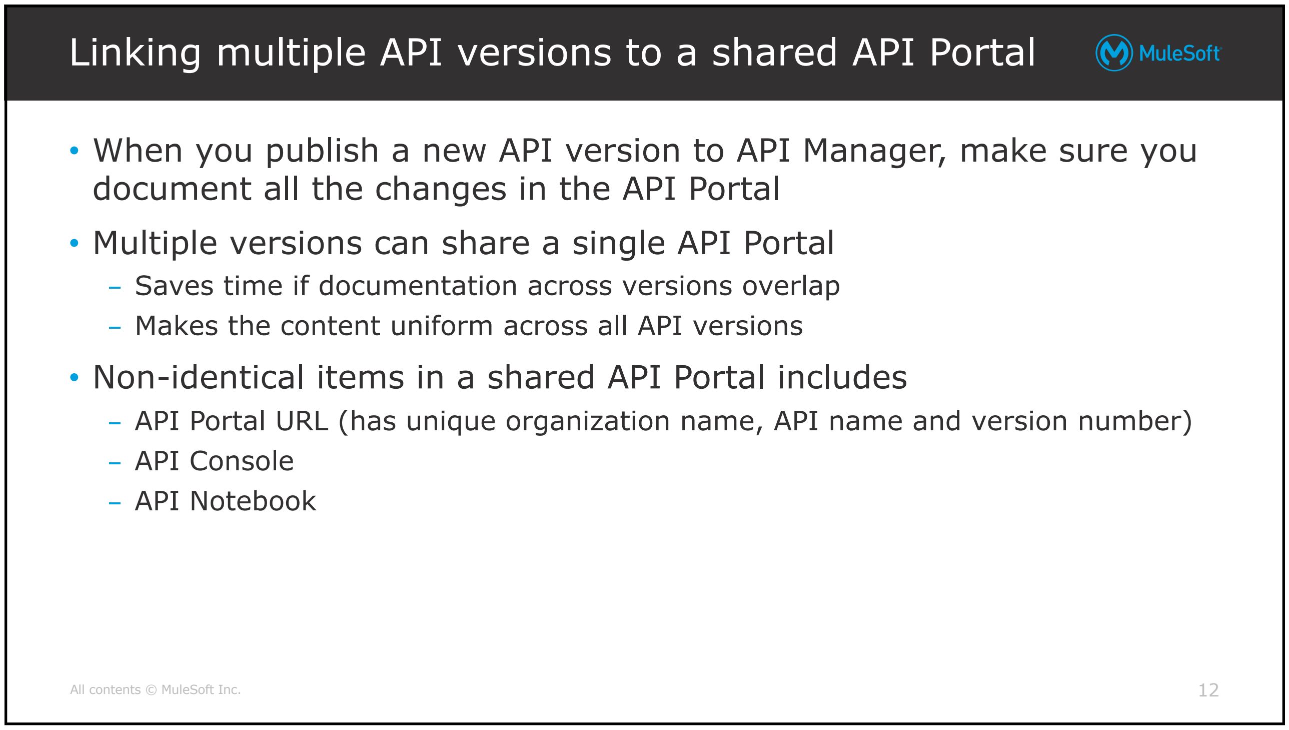
Specifying version in a custom request header

MuleSoft

- Add a custom request header with the API version
  - When the header is not set with the version number do you return an error message or route to the new version?
  - They are not a semantic way of describing a resource

```
/employees:
  get:
    headers:
      api-version:
        type: string
    responses:
      200:
        body:
          application/json:
```

Documenting changes in new API versions

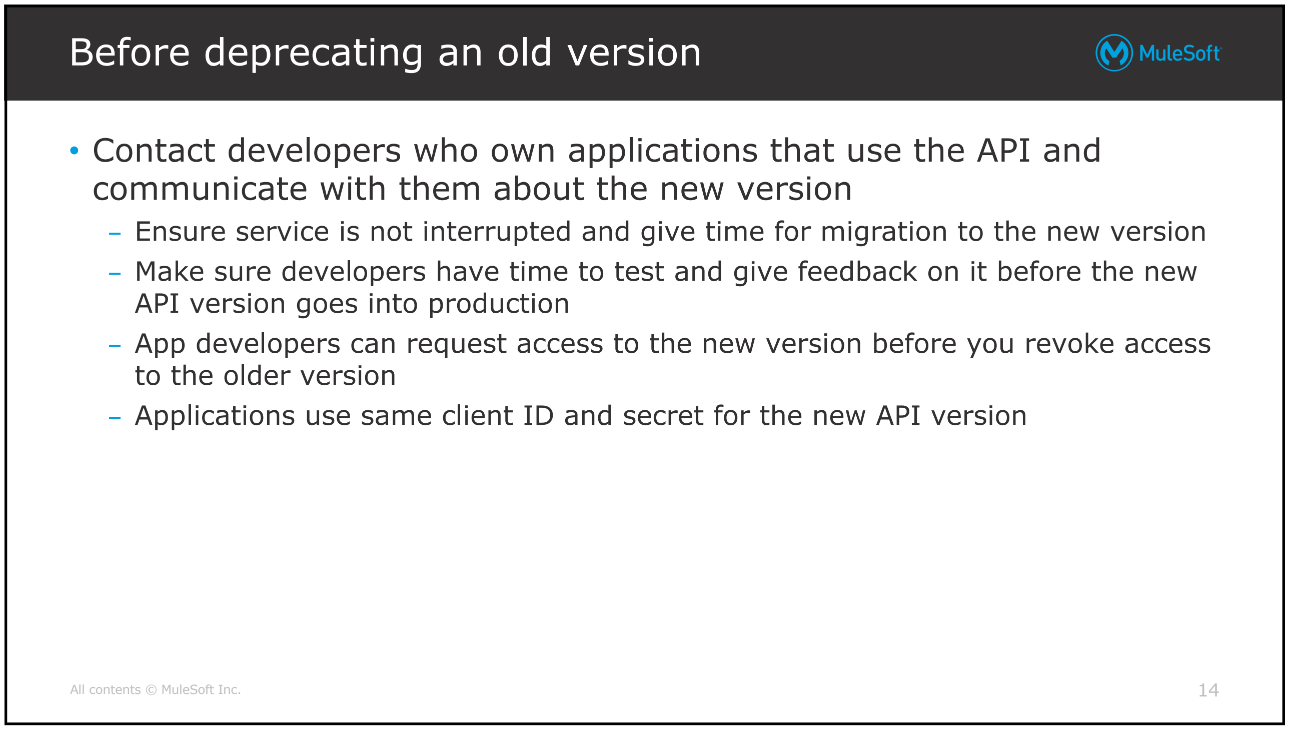


Linking multiple API versions to a shared API Portal

MuleSoft

- When you publish a new API version to API Manager, make sure you document all the changes in the API Portal
- Multiple versions can share a single API Portal
  - Saves time if documentation across versions overlap
  - Makes the content uniform across all API versions
- Non-identical items in a shared API Portal includes
  - API Portal URL (has unique organization name, API name and version number)
  - API Console
  - API Notebook

Deprecating older versions of APIs



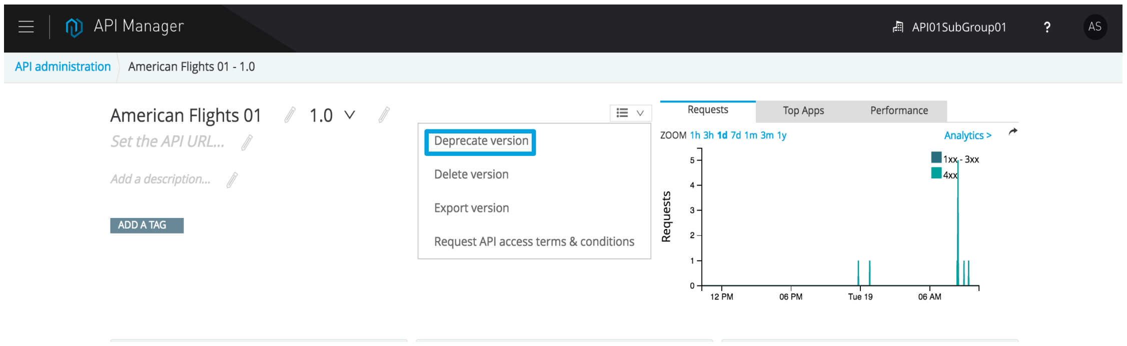
Before deprecating an old version

MuleSoft

- Contact developers who own applications that use the API and communicate with them about the new version
  - Ensure service is not interrupted and give time for migration to the new version
  - Make sure developers have time to test and give feedback on it before the new API version goes into production
  - App developers can request access to the new version before you revoke access to the older version
  - Applications use same client ID and secret for the new API version

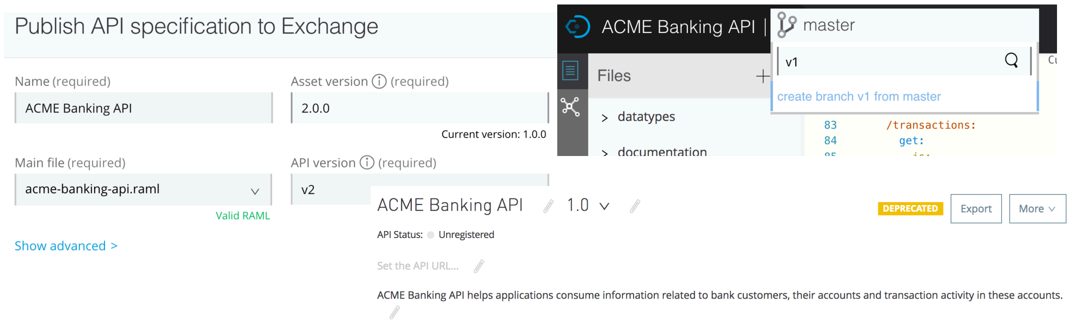
Deprecating old API versions

- Set the old API version to **Deprecated** to prevent developers from signing up for access to your old API version
- Provide API calls for a finite amount of time until deprecation cut-off date occurs



Walkthrough 13-1: Add a new API version

- Create a new API version in Anypoint Design Center
- Learn how to publish the new version to Exchange and API Manager
- Deprecate the old version of the API


Publish API specification to Exchange

Name (required): ACME Banking API

Asset version (required): 2.0.0

Current version: 1.0.0

Main file (required): acme-banking-api.raml

Valid RAML

API version (required): v2

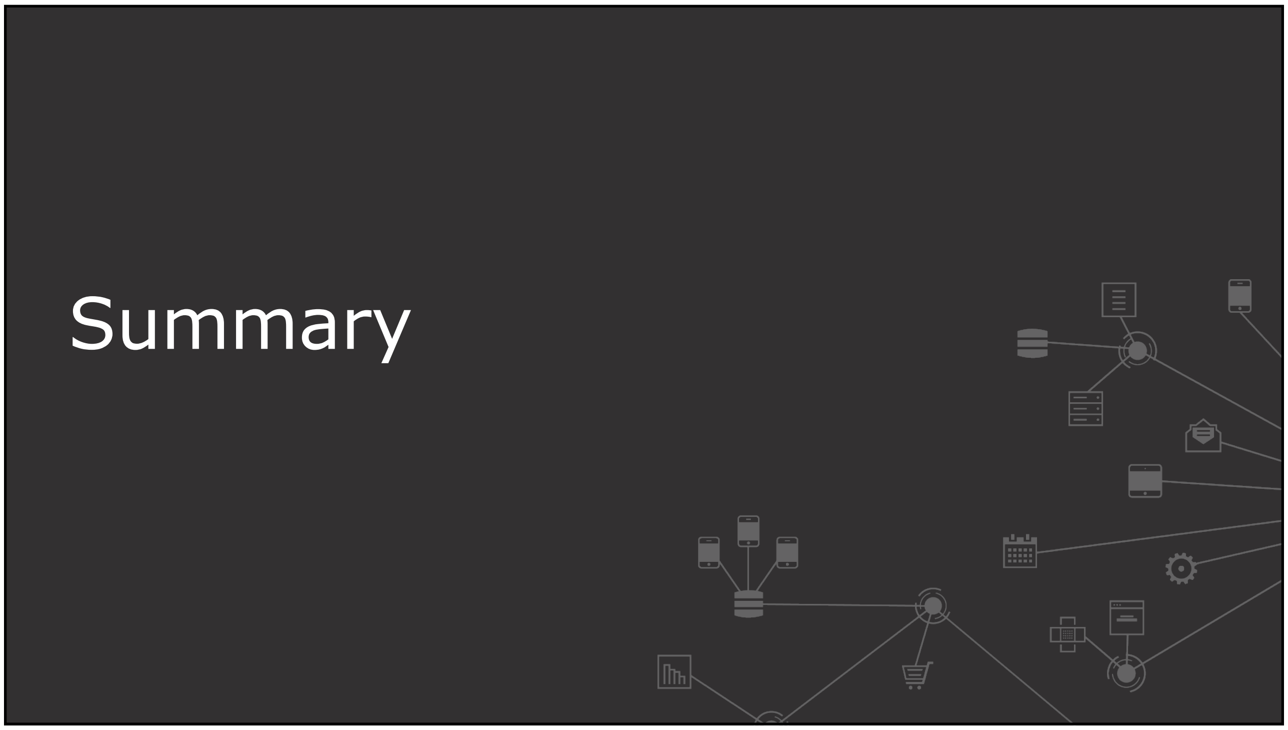
Show advanced >

```
83        /transactions:
84          get:
```

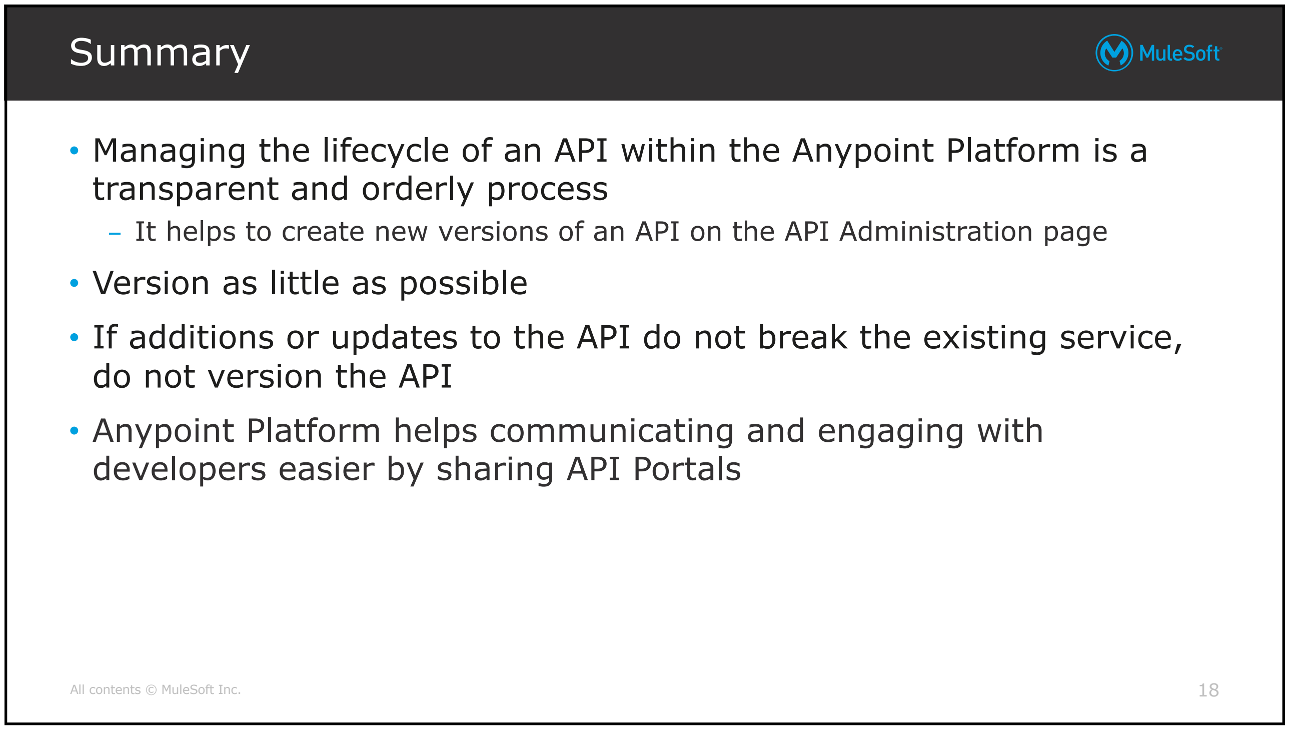
API Status: Unregistered

Set the API URL...

ACME Banking API helps applications consume information related to bank customers, their accounts and transaction activity in these accounts.



Summary



Summary

- Managing the lifecycle of an API within the Anypoint Platform is a transparent and orderly process
  - It helps to create new versions of an API on the API Administration page
- Version as little as possible
- If additions or updates to the API do not break the existing service, do not version the API
- Anypoint Platform helps communicating and engaging with developers easier by sharing API Portals