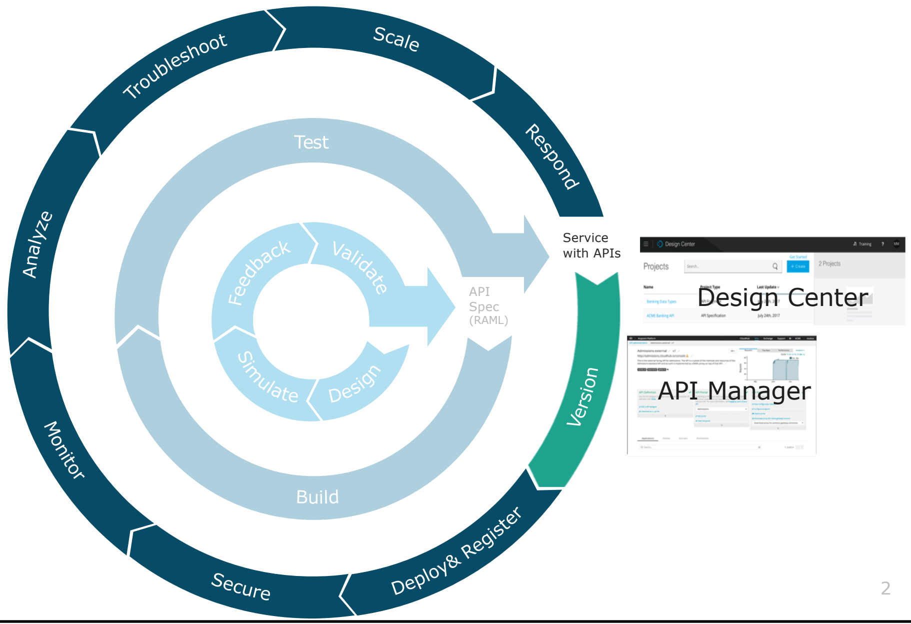
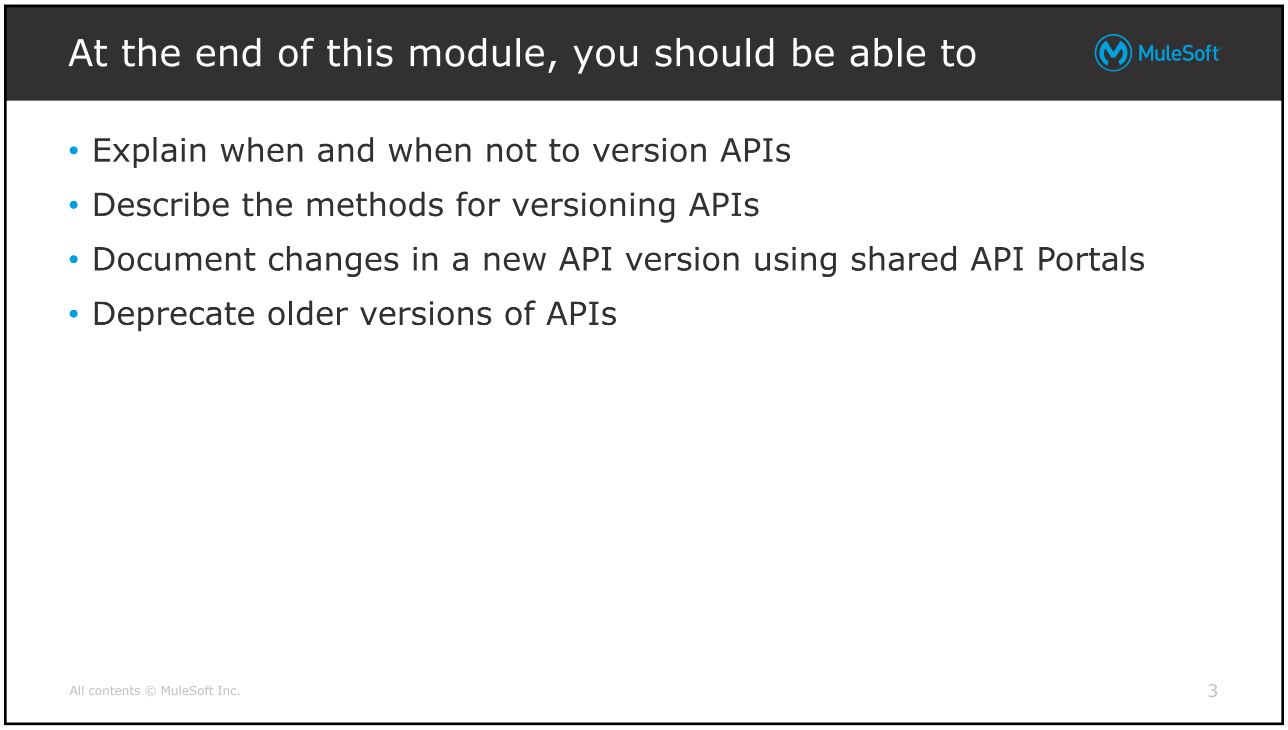
Module 13: Versioning APIs

Goal



2



At the end of this module, you should be able to

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

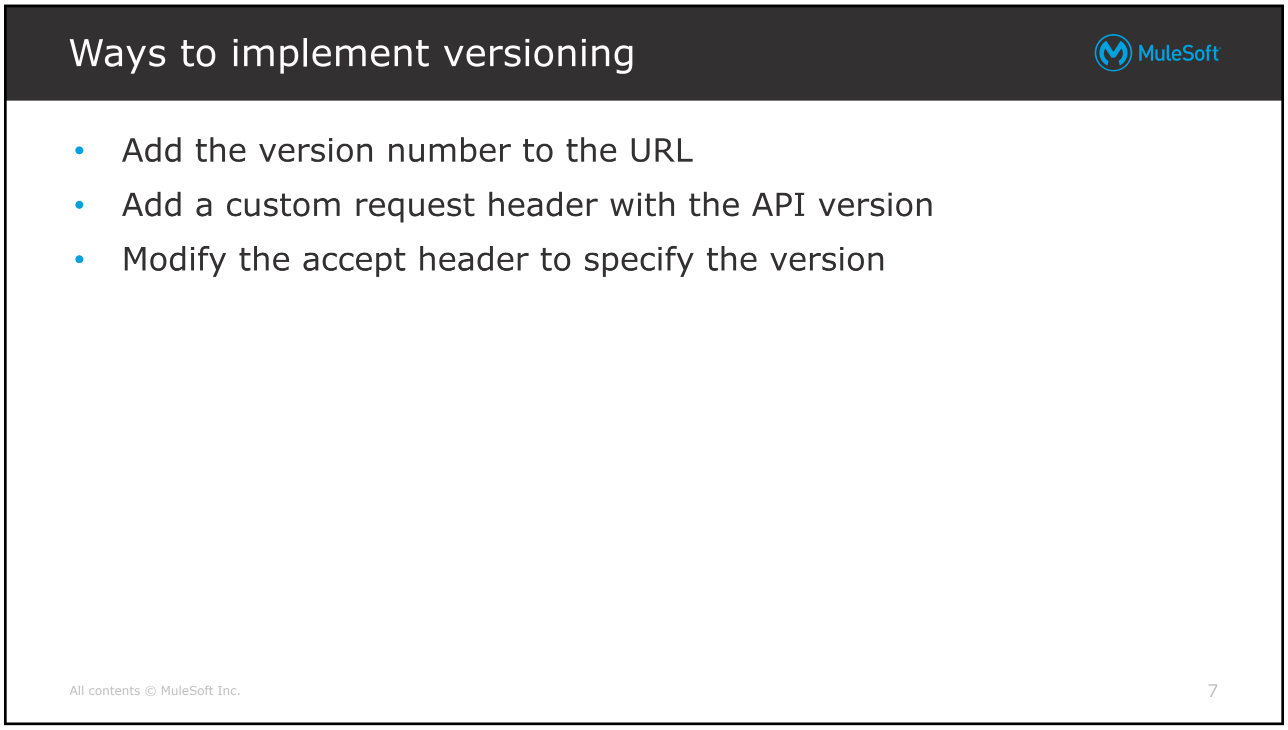
MuleSoft

- Version when additions or changes to an API break existing client code or changes the API interface rendering client code to fail
  - Does the client/server have to change the way in which they communicate with the API? Then, version your API
- Versioning helps to handle future changes even if you do not know what those changes are yet
- **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
    - Adding new resources/methods
    - Changing technologies in backend applications

Versioning throughout the API lifecycle

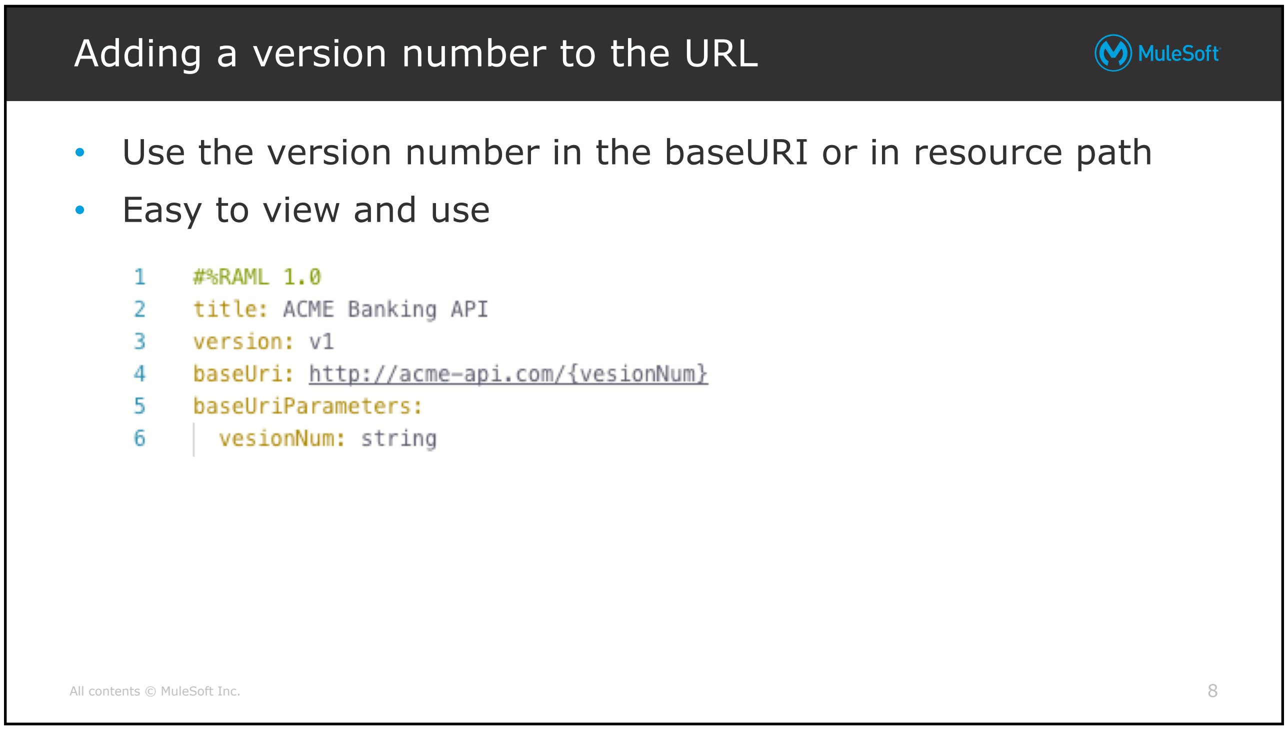
MuleSoft

- 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



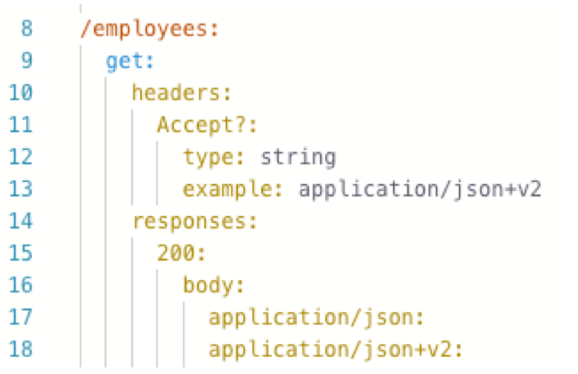
Adding a version number to the URL

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

```
#%RAML 1.0
title: ACME Banking API
version: v1
baseUri: http://acme-api.com/{vesionNum}
baseUriParameters:
  vesionNum: string
```

Specifying version in the Accept header

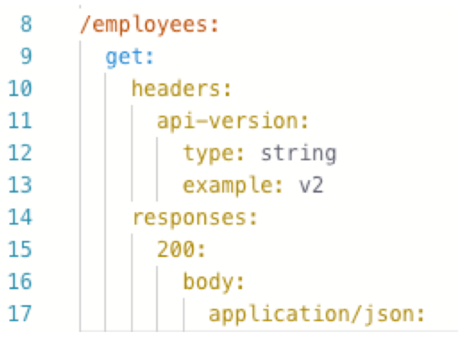
- Clients can specify the version in the accept header
  - Needs careful construction of the request with the right value for the header
  - Since the Accept header involves the type of the data returned, it might look like we are representing a different version of data versus the API



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

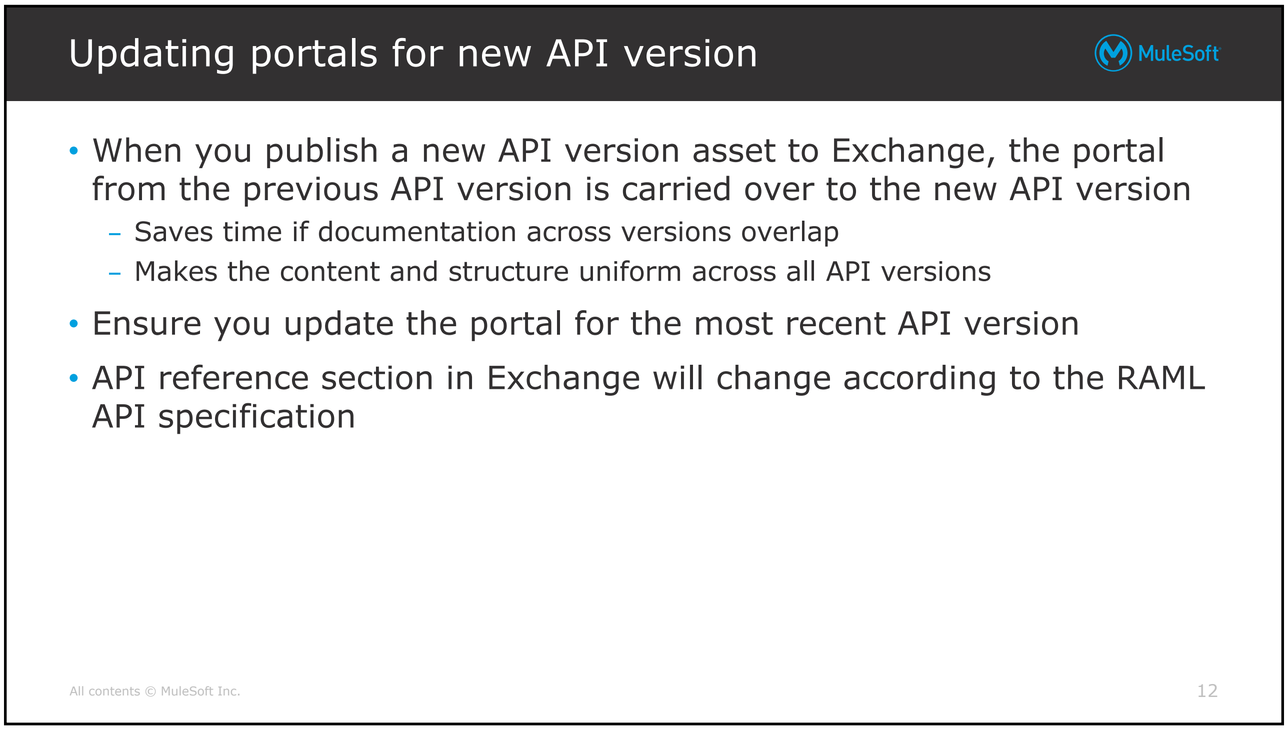
Specifying version in a custom request header

- 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
        example: v2
    responses:
      200:
        body:
          application/json:
```

Documenting changes in new API versions



Updating portals for new API version

- When you publish a new API version asset to Exchange, the portal from the previous API version is carried over to the new API version
  - Saves time if documentation across versions overlap
  - Makes the content and structure uniform across all API versions
- Ensure you update the portal for the most recent API version
- API reference section in Exchange will change according to the RAML API specification

Deprecating older versions of APIs

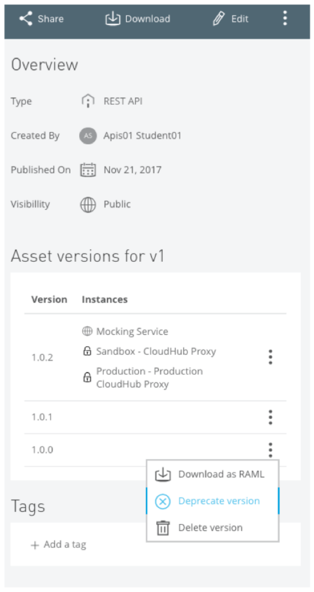
Before deprecating an old version

- 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

MuleSoft

- Deprecation needs to be carried out in Exchange
  - Deprecate each asset version that belongs to the old API version
- Set the old API asset version(s) 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


Share Download Edit

Overview

Type	REST API
Created By	Apis01 Student01
Published On	Nov 21, 2017
Visibility	Public

Asset versions for v1

Version	Instances
1.0.2	Mocking Service Sandbox - CloudHub Proxy Production - Production CloudHub Proxy
1.0.1	
1.0.0	

Download as RAML
Deprecate version
Delete version

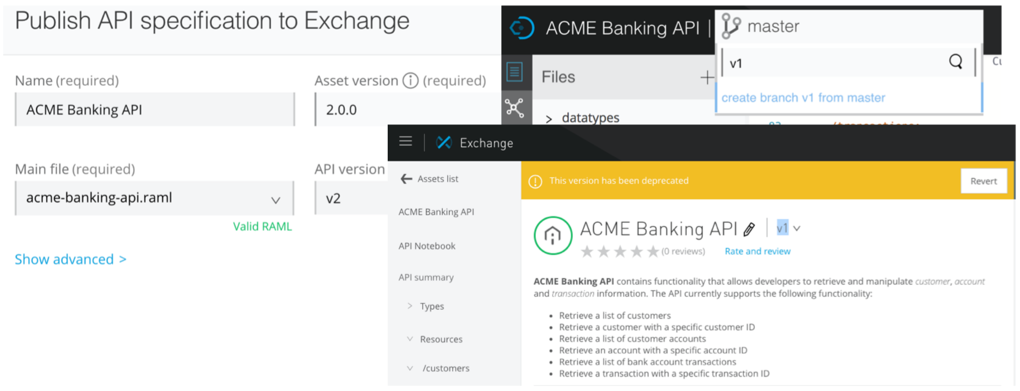
Tags

+ Add a tag

Walkthrough 13-1: Add a new API version

MuleSoft

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


Publish API specification to Exchange

Name (required)	ACME Banking API
Asset version (required)	2.0.0
Main file (required)	acme-banking-api.raml
API version	v2

Valid RAML

Show advanced >

ACME Banking API | master

Files

> datatypes

v1

create branch v1 from master

Exchange

Assets list
ACME Banking API
API Notebook
API summary
Types
Resources
/customers

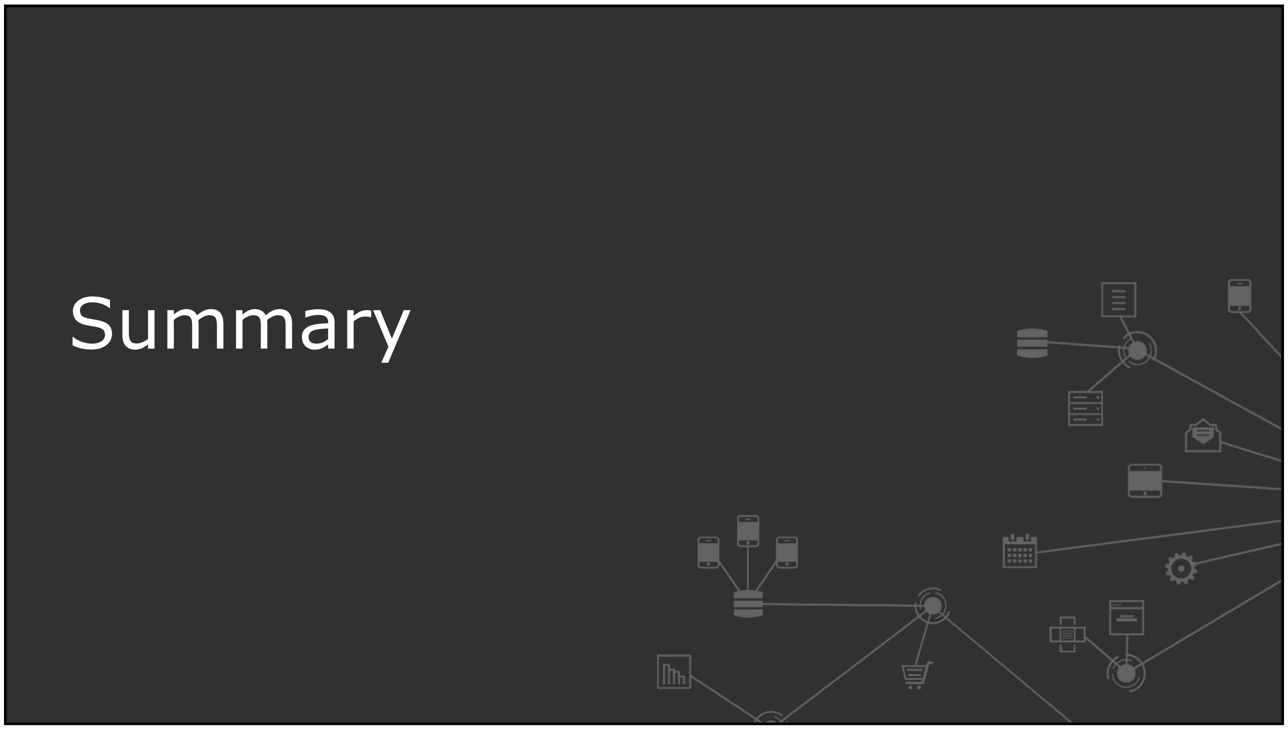
This version has been deprecated Revert

ACME Banking API | v1

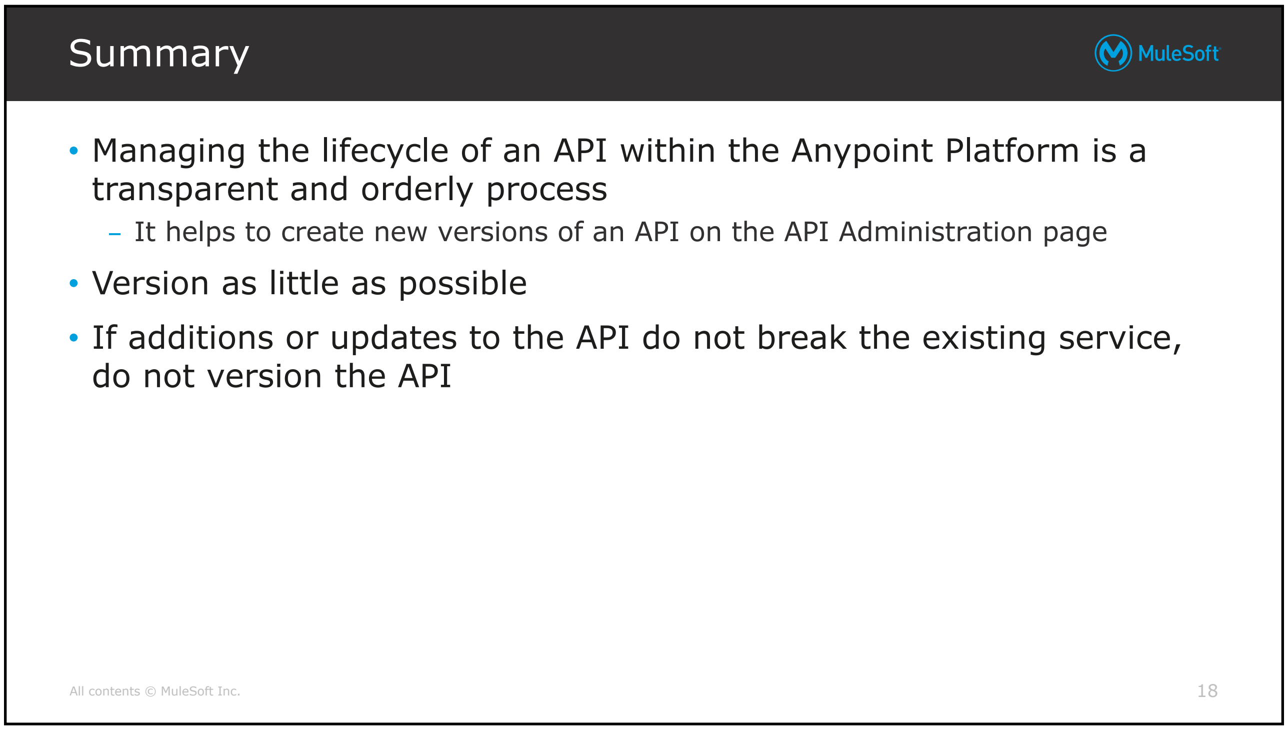
(0 reviews) Rate and review

ACME Banking API contains functionality that allows developers to retrieve and manipulate *customer*, *account* and *transaction* information. The API currently supports the following functionality:

- Retrieve a list of customers
- Retrieve a customer with a specific customer ID
- Retrieve a list of customer accounts
- Retrieve an account with a specific account ID
- Retrieve a list of bank account transactions
- Retrieve a transaction with a specific transaction ID



Summary



Summary

MuleSoft

- 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