



AZ-204T00A

# Learning Path 08: Implement API Management

# Agenda

---

- Explore API Management

# Module 1: Explore API Management



# Learning objectives

- Describe the components, and their function, of the API Management service.
- Explain how API gateways can help manage calls to your APIs.
- Secure access to APIs by using subscriptions and certificates.
- Create a backend API

# Introduction

- API Management helps organizations publish APIs to external, partner, and internal developers to unlock the potential of their data and services.

# Discover the API Management service (1 of 2)

## The role of API management

- API management provides core functions to ensure a successful API program through developer participation, business insight, analysis, security, and protection.
- Each API consists of one or more operations, and each API can be added to one or more products.

## The system is made up of the following components:

- API gateway
- Azure portal
- The Developer portal

# Discover the API Management service (2 of 2)

## Products

Products are how APIs are surfaced to developers.

## Groups

Groups are used to manage the visibility of products to developers.

## Developers

Developers represent the user accounts in an API Management service instance.

## Policies

Allow the Azure portal to change the behavior of the API through configuration.

## Developer portal

You can learn about your API, view and call operations, and subscribe to products.

# Explore API gateways (1 of 2)

## API gateway role

- An API gateway sits between clients and services.
- It acts as a reverse proxy, routing requests from clients to services.
- Applies policies and collects telemetry.
- Can also perform tasks such as authentication, SSL termination, and rate limiting.



# Explore API gateways (2 of 2)

## Potential issues when deploying an API without a gateway

- Lead to complex client code.
- Create coupling between the client and the backend.
- A single operation may need to call multiple services.
- Every public-facing service must be dealt with.
- The service must expose a client-friendly protocol.

## Functional design patterns

- Gateway routing: Use a gateway as a reverse proxy to route requests to one or more back-end services
- Gateway aggregation: Use a gateway to aggregate multiple individual requests into one request.
- Gateway Offloading: Use the gateway to offload functionality from individual services to the gateway, particularly cross-cutting concerns.

# Explore API Management policies (1 of 2)

## The role of policies

- Allows the publisher to change the behavior of the API through configuration.
- A collection of statements that are executed sequentially in response to requests or responses to the API.
- The policy is applied in the gateway between the API consumer and the managed API.
- Policies can apply changes to inbound requests and outbound responses.

## Policy configuration

- The policy definition is a simple XML document that describes a series of inbound and outbound statements.
- The configuration is divided into inbound, backend, outbound and error.
- If an error occurs during the processing of the request, any remaining steps will be skipped, and the statement that jumps to the error section will be executed.

# Explore API Management policies (2 of 2)

## Policy examples

```
<policies>
  <inbound>
    <!-- statements to be applied to the
         request go here -->
  </inbound>
  <backend>
    <!-- statements to be applied before the
         request is forwarded to
         the backend service go here -->
  </backend>
  <outbound>
    <!-- statements to be applied to the
         response go here -->
  </outbound>
  <on-error>
    <!-- statements to be applied if there
         is an error condition go here -->
  </on-error>
</policies>
```

```
<policies>
  <inbound>
    <cross-domain />
    <base />
    <find-and-replace from="xyz" to="abc" />
  </inbound>
</policies>
```

# Create advanced policies (1 of 2)

## API management policy

### Control flow

Conditionally applied based on the evaluation result of the Boolean expression.

### Forward request

Forward the request to the backend service.

### Limit concurrency

Prevent policies from executing more than the specified number of requests at a time.

### Log to Event Hub

Send the message in the specified format to the event center defined by the Logger entity.

### Mock response

The pipeline execution is aborted and the simulated response is returned directly to the caller.

### Retry

Retry the execution of the contained policy statement until the condition is met.

# Create advanced policies (2 of 2)

## Examples

```
<limit-concurrency key="expression" max-count="number">  
  <!-- nested policy statements -->  
</limit-concurrency>
```

```
<forward-request timeout="time in seconds" follow-redirects="true | false"/>
```

```
<log-to-eventhub logger-id="id of the logger entity" partition-id="index of the partition where  
messages are sent" partition-key="value used for partition assignment">  
  <!-- Expression returning a string to be logged -->  
</log-to-eventhub>
```

# Secure APIs by using subscriptions (1 of 3)

## Subscription key scopes

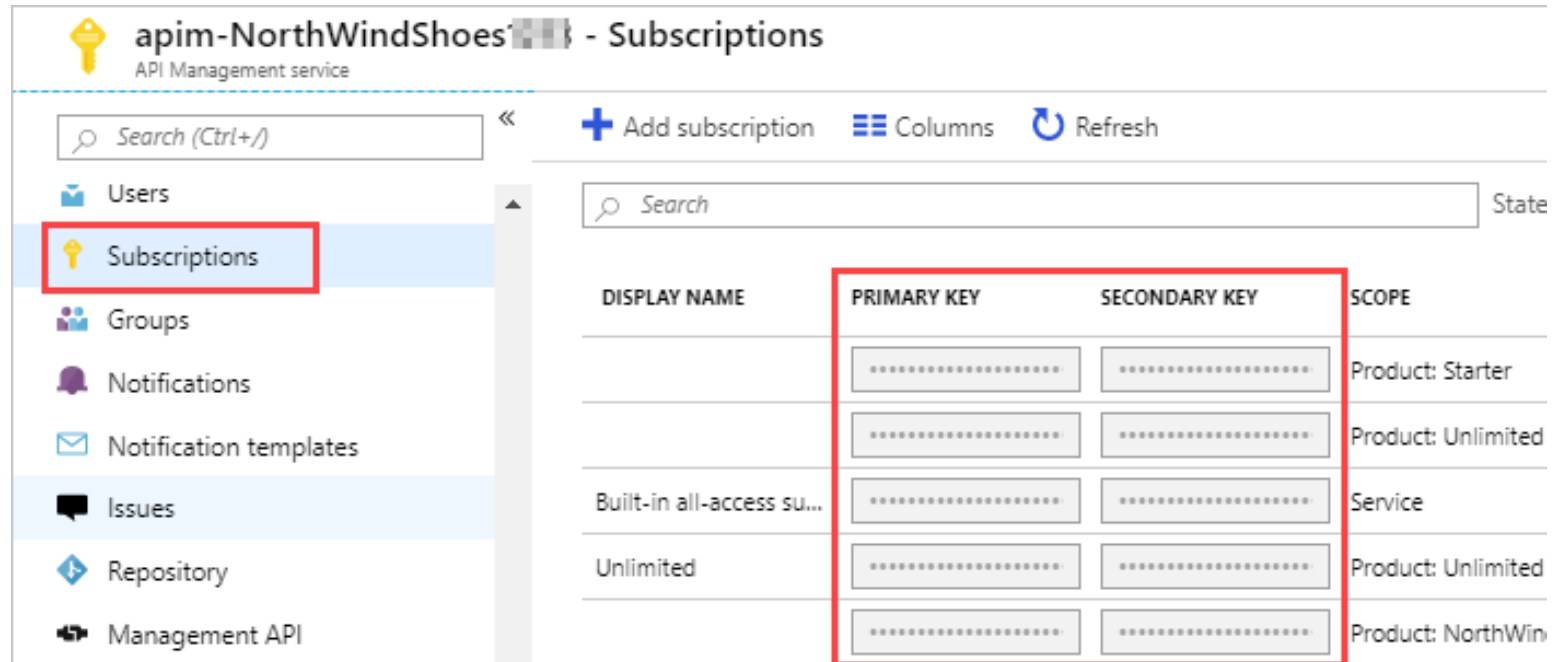
Scope	Details
All APIs	Applies to every API accessible from the gateway
Single API	This scope applies to a single imported API and all of its endpoints
Product	A product is a collection of one or more APIs that you configure in API Management. You can assign APIs to more than one product. Products can have different access rules, usage quotas, and terms of use.

**Note:** API Management also supports other mechanisms for securing access to APIs, including: OAuth2.0, Client certificates, and IP allow listing.

# Secure APIs by using subscriptions (2 of 3)

## Applications that call protected APIs

- Must include the key in every request
- You can regenerate these subscription keys at any time.
- Every subscription has two keys, a primary and a secondary.



apim-NorthWindShoes - Subscriptions  
API Management service

Search (Ctrl+/) Add subscription Columns Refresh

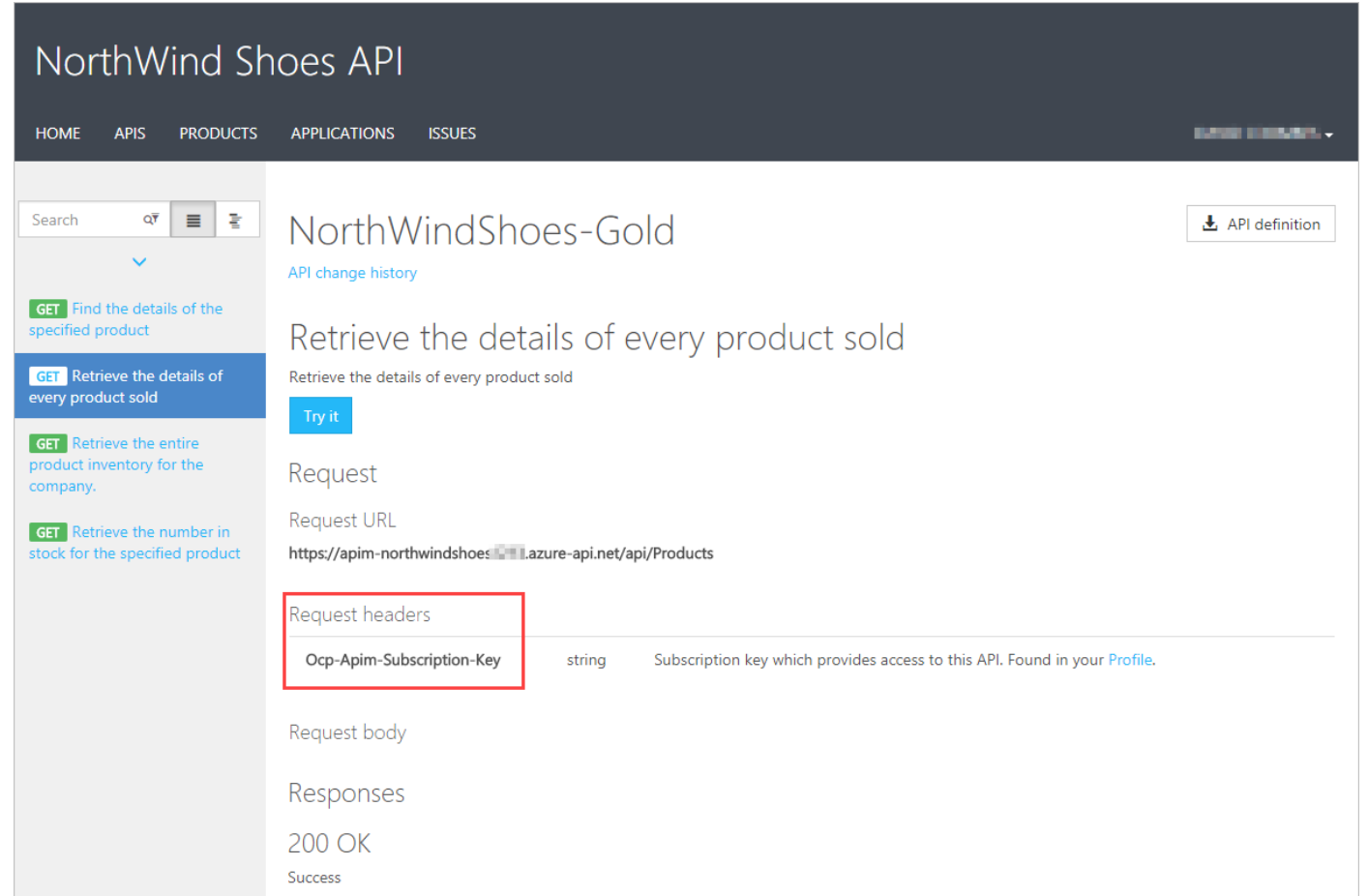
Search State

DISPLAY NAME	PRIMARY KEY	SECONDARY KEY	SCOPE
	.....	.....	Product: Starter
	.....	.....	Product: Unlimited
Built-in all-access su...	.....	.....	Service
Unlimited	.....	.....	Product: Unlimited
	.....	.....	Product: NorthWin

# Secure APIs by using subscriptions (3 of 3)

## Call an API with the subscription key

- Keys can be passed in the request header, or as a query string in the URL.
- The default header name is Ocp-Apim-Subscription-Key.
- Use the developer portal to test out API calls



NorthWind Shoes API

HOME APIS PRODUCTS APPLICATIONS ISSUES

Search

GET Find the details of the specified product

GET Retrieve the details of every product sold

GET Retrieve the entire product inventory for the company.

GET Retrieve the number in stock for the specified product

NorthWindShoes-Gold

API change history

Retrieve the details of every product sold

Retrieve the details of every product sold

Try it

Request

Request URL

https://apim-northwindshoes.azure-api.net/api/Products

Request headers

Ocp-Apim-Subscription-Key string Subscription key which provides access to this API. Found in your [Profile](#).

Request body

Responses

200 OK

Success



# Secure APIs by using certificates (1 of 4)

## Transport Layer Security client authentication

Property	Description
Certificate Authority (CA)	Only allow certificates signed by a particular CA
Thumbprint	Allow certificates containing a specified thumbprint
Subject	Only allow certificates with a specified subject
Expiration Date	Only allow certificates that have not expired

Two methods to verify a certificate:

- Check who issued the certificate.
- If the certificate is issued by the partner, verify that it came from them.

# Secure APIs by using certificates (2 of 4)

- Accepting client certificates in the Consumption tier
- Certificate Authorization Policies
- Check the thumbprint of a client certificate
- Check the thumbprint against certificates uploaded to API Management
- Check the issuer and subject of a client certificate

# Secure APIs by using certificates (3 of 4)

## Example

```
<!--Check the thumbprint of a client certificate-->
<choose>
  <when condition="@context.Request.Certificate == null || context.Request.Certificate.Thumbprint
  != "desired-thumbprint")" >
    <return-response>
      <set-status code="403" reason="Invalid client certificate" />
    </return-response>
  </when>
</choose>
```

# Secure APIs by using certificates (4 of 4)

## Example

```
<!--Check the issuer and subject of a client certificate-->
<choose>
  <when condition="@context.Request.Certificate == null || context.Request.Certificate.Issuer !=
    "trusted-issuer" || context.Request.Certificate.SubjectName.Name != "expected-subject-name")" >
    <return-response>
      <set-status code="403" reason="Invalid client certificate" />
    </return-response>
  </when>
</choose>
```

# Exercise: Create a backend API

In this exercise you learn how to implement an API Management instance and import and configure an API.

## Objectives

- Create an API Management instance
- Import a backend API
- Configure the backend settings
- Test the API
- Clean up resources

# Summary and knowledge check

In this module, you learned how to:

- Describe the components, and their function, of the API Management service.
- Explain how API gateways can help manage calls to your APIs.
- Secure access to APIs by using subscriptions and certificates.
- Create a backend API.

1

What component of the API Management service would a developer use if they need to create an account and subscribe to get API keys?

2

What API Management policy would you use to apply a policy based on a condition?

# Discussion and lab



# Group discussion questions

- The security team of Contoso Inc noticed a particular IP Address is firing thousands of the requests against your API. What would do?
- You've setup API Management and a 3rd party partner is receiving 401 when calling your API. What can be the issue?
- Contoso Inc is hosting APIs on App Service for 3rd party integration. What can you do to increase the security?



## Lab 08: Create a multi-tier solution by using Azure services

In this lab, you will create a containerized application to host a web app on Azure, as the source of information for the API. You will then build an API proxy using the Azure API Management capabilities to expose and test your APIs. Developers can query the APIs to test the service and validate its applicability.

<http://aka.ms/az204labs>

- Exercise 1: Create an Azure App Service resource by using a Docker container image
- Exercise 2: Build an API proxy tier by using Azure API Management

# End of presentation

