API intergration

Publish and manage your APIs with Azure API Management

Azure API Management (APIM) is a fully managed cloud service that you can use to publish, secure, transform, maintain, and monitor APIs. Azure API management is done by **azure api gateway**. An Azure API gateway is an instance of the Azure API management service.

why use api management

For developers, API Management provides a range of benefits.

- **API documentation**. Documentation of APIs enables calling clients to quickly integrate their solutions. API Management allows you to quickly expose the structure of your API to calling clients through modern standards like Open API. You can have more than one version of an API. With multiple versions, you can stage app updates as your consuming apps don't have to use the new version straight away.
- **Rate limiting access**. If your API could potentially access a large amount of data, its a good idea to limit the rate at which clients can request data. Rate limiting helps maintain optimal response times for every client. API Management let you set rate limits as a whole or for specific individual clients.
- **Health monitoring**. APIs are consumed by remote clients. So it can be difficult to identify potential problems or errors. API Management lets you view error responses and log files, and filter by types of responses.
- **Modern formats like JSON**. APIs have used many different data exchange formats over the years from XML to CSV and many more. API Management enables you to expose these formats using modern data models like JSON.
- **Connections to any API**. In many businesses, APIs are located across different countries and use different formats. API Management lets you add all of these disparate APIs into single modern interface.
- **Analytics**. As you develop your APIs, it's useful to see how often your APIs are being called and by which types of systems. API Management allows you to visualize this data within the Azure portal.
- **Security**. Security is paramount when dealing with system data. Unauthorized breaches can cost companies money, time lost in reworking code, and reputational loss. Security tools that you can use with Azure API management include OAuth 2.0 user authorization, and integration with Azure Active Directory.

Pricing tiers

When you create an Azure API management gateway, you must select from one of several pricing tiers:

- **Developer**. You use the Developer tier for evaluating the API management service. You shouldn't use this tier for production deployments.
- **Basic**. Entry level production use. 99.9% SLA. 1000 requests/sec. Two scale units.

> ⓘ Note
>
> A scale unit enables you to scale up a service. The more scale units you have, the more you can scale up the service.

- **Standard**. Medium level production use. 99.9% SLA. 2500 requests/sec. Four scale units.
- **Premium**. Multi region deployment. High volume use. 99.95% SLA. 4000 requests/sec. 10 scale units per region.
- **Consumption**. The serverless consumption tier plan lets you pay for what you use, rather than having dedicated resources. You can quickly set up ad-hoc testing, and you can scale up API access when demand increases. The consumption tier has built-in high availability and autoscaling. Because it is serverless, you can provision a consumption tier gateway in a lot less time than the other server-based tiers.

setup API management

To set up API management, you:

1. Create an API Management gateway. The name you use for the gateway must be globally unique within the ***.azure-api.net** namespace.
2. Register an existing Web API with the gateway. Registering the API makes the API available to consumers at the **<gateway>.azure-api.net** endpoint.
3. Call the existing API through the gateway. Consumers can now use the API at the endpoint URL, or through the developer portal.

API framework

API Management provides you with several options for importing APIs.

Type	Details
Blank API	You can import an API with a blank API definition. You then manually specify all the required parameters.
Open API	Open API is a specification that documents all the endpoints and operations for RESTful APIs, and all input and output parameters. OpenAPI was originally called Swagger.
WADL	Web Application Description Language is an XML description of HTTP-based web services. It is a simpler format and more lightweight than WSDL.
WSDL	Web Service Description Language is an XML description of any network service, not just HTTP.
Logic App	Logic apps are used to orchestrate and automate workflows and integrations with various data sources.
API App	An API hosted within an API app service in Azure.
Function App	Serverless code that can be called through triggers.

Policies

In Azure API Management, administrators can use policies to alter the behavior of APIs through configuration.

Popular configurations include:

- Conversion from XML to JSON
- Call rate limiting to restrict the number of incoming calls.
- Setting inbound and outbound headers
- Policies used for restricting access
- Policies for Authentication

When do policies execute?

- Inbound. These policies execute when a request is received from a client.
- Backend. These policies execute before a request is forwarded to a managed API.
- Outbound. These policies execute before a response is sent to a client.
- On-Error. These policies execute when an exception is raised.

Policy Scopes

- The global policy scope: Policies applied at the global scope affect all APIs within the instance of API Management.
- The product policy scope. Policies applied at the API scope affect only API within Product.
- The API policy scope: Policies applied at the API scope affect only a single API(apis from one import).
- Operation policy scope: Policies applied at the operation scope affect only one operation(function) within the API.

Which order are policies applied in

You can use the `<base />` tag to determine when policies from a higher scope are applied. For example, consider this policy, applied at the API scope:

XML Copy

```
<policies>
    <inbound>
        <base />
        <find-and-replace from="game" to="board game" />
    </inbound>
</policies>
```

Because the `<base>` tag appears above the `<find-and-replace>` tag, Azure applies policies from the global and product scopes first, and then executes the find-and-replace policy.

Product

Products let you group APIs, define terms of use, and runtime policies. API consumers can subscribe to a product on the developer portal to obtain a key to call your API. For all pricing tiers except consumption, there two default products: Starter and Unlimited. The Unlimited product is designed for production API management, as it has no restrictions on the number of attached APIs. You can create as many new products as you need. The starter product has a limit of five API calls/minute, and a maximum of 100 API calls/week.

Call an API with a subscription key

A subscription key is a unique auto-generated string which needs to be passed through in the headers of the client request. You could create different subscription and generate the key from it. When creating the subscription, the api(single api,all api, product) could be assossiated to the subscription. When calling the API, the Ocp-Apim-Subscription-Key head should be set with the subscription key, otherwise the api call would be rejected.

add caching to API management

- caching all response To set up a cache, you use an **outbound** policy named **cache-store** to store responses. You also use an **inbound** policy named **cache-lookup** to check if there is a cached response for the current request.

- caching part of the response Use the **cache-store-value** policy to add the value, with an identifying key. Retrieve the value from the cache by using the **cache-lookup-value** policy. If you want to remove a value before it expires, use the **cache-remove-value** policy

- Using vary-by tags Use the element within the policy to make sure that cache is used even some parameter veries.
- why using an external cache
    - You want to avoid the cache being cleared when the API Management service is updated.
    - You want to have greater control over the cache configuration than the internal cache allows.
    - You want to cache more data than can be store in the internal cache.

protect APIs on Azure API Management

remove the headers that reveals technical stack

e.g. removing the X-Powered-By filter using policy.

Transformation policy could be used the remove the important information which could be used by hacker.

Transform	Detail
Convert JSON to XML	Converts a request or response body from JSON to XML.
Convert XML to JSON	Converts a request or response body from XML to JSON.
Find and replace string in body	Finds a request or response substring and replaces it with a different substring.
Mask URLs in content	Rewrites links in the response body so that they point to the equivalent link through the gateway.
Set backend service	Changes the backend service for an incoming request.
Set body	Sets the message body for incoming and outgoing requests.
Set HTTP header	Assigns a value to an existing response or request header, or adds a new response or request header.
Set query string parameter	Adds, replaces the value of, or deletes a request query string parameter.
Rewrite URL	Converts a request URL from its public form to the form expected by the web service.
Transform XML using an XSLT	Applies an XSL transformation to the XML in the request or response body.

Throttle API requests(limit access to API endpoints)

- Limit by subscription throttling: Subscription throttling allows you to set the rate limits by a specific API operation of a subscription.
- Limit by key(key to identify client like ip) throttling: managing the rate limits as it applies the limit to a specified request key, often the client IP address.

Value	Detail
context.Request.IpAddress	Rates limited by client IP address
context.Subscription.Id	Rates limited by subscription ID
context.Request.Headers.GetValue("My-Custom-Header-Value")	Rates limited by a specified client request header value

The policy is not available when your API Management gateway is in the Consumption tier.

Control authentication for your APIs with Azure API Management

- use subscription key
- Use client certificates

Azure API Management for Azure Function

setup api management for azure function

1. create azure function and azure api management
2. in azure api management create api
3. choose azure function
4. in the popup dialog -> browse -> config function app -> select the function app.

Microservices architecture challenges

- Client apps are coupled to microservices. If you want to change the location or definition of the microservice, you may have to reconfigure or update the client app.
- Each microservice may be presented under different domain names or IP addresses. This presentation can give an impression of inconsistency to users and can negatively affect your branding.
- It can be difficult to enforce consistent API rules and standards across all microservices. For example, one team may prefer to respond with XML and another may prefer JSON.
- You're reliant on individual teams to implement security in their microservice correctly. It's difficult to impose these requirements centrally.

Benefit of using azure api management

- Client apps are coupled to the API expressing business logic, not the underlying technical implementation with individual microservices. You can change the location and definition of the services without necessarily reconfiguring or updating the client apps.
- API Management acts as an intermediary. It forwards requests to the right microservice, wherever it is located, and returns responses to users. Users never see the different URIs where microservices are hosted.
- You can use API Management policies to enforce consistent rules on all microservices in the product. For example, you can transform all XML responses into JSON, if that is your preferred format.
- Policies also enable you to enforce consistent security requirements.