# Create API Management Instance

1. From the Azure portal menu, Select **Create a resource**
2. ON the **Create a resource** page, select **Integration > API Management**
3. In the **Create API Management** page, enter the appropriate settings:

|  |  |
| --- | --- |
| Setting | Description |
| Subscription | The subscription under which the new service instance will be created. |
| Resource group | Select a new or existing resource group. A resource group is a logical container into which Azure resources are deployed and managed. |
| Region | Select a geographic region near you from the available API Management service locations. |
| Resource name | A unique name for your API Management service. The name cannot be changed later. The service name refers to both the service and the corresponding Azure resource.  The service name is used to generate a default domain name: *<name>.azure-api.net*. |
| Organization name | The name of your organization. This name is used in many places, including the title of the developer portal and sender of notification emails. |
| Administrator email | The email address to which all the notifications from **API Management** will be sent. |
| Pricing tier | Select **Developer** tier to evaluate the service. This tier is not for production use. |

1. Select **Review + create**.

# Mock API responses

## Create a test API

1. Sign in to the Azure portal, and then navigate to your API Management instance.
2. Select **APIs** > **+ Add API** > **HTTP** tile.
3. In the **Create an HTTP API** window, select **Full**.
4. Enter the *Test API* for **Display name**.
5. Select **Unlimited** for **Products**.
6. Ensure that **Managed** is selected for **Gateways**.
7. Select **Create**.

## Add an operation to the test API

1. Select the API you created in the previous step.
2. Select **+ Add Operation**.
3. In the **Frontend** window, enter the following values.

|  |  |  |
| --- | --- | --- |
| Setting | Value | Description |
| Display name | *Test call* | The name that is displayed in the developer portal. |
| URL (HTTP verb) | GET | Select one of the predefined HTTP verbs. |
| URL | */test* | A URL path for the API. |
| Description |  | Optional description of the operation, used to provide documentation in the developer portal to the developers using this API. |

1. Select the **Responses** tab. Enter settings on this tab to define response status codes, content types, examples, and schemas.
2. Select **+ Add response**, and select **200 OK** from the list.
3. Under the **Representations** heading on the right, select **+ Add representation**.
4. Enter *application/json* into the search box and select the **application/json** content type.
5. In the **Sample** text box, enter { “sampleField” : “test” }
6. Select **Save**.

## Enable response mocking

1. Select the API you created in Create a test API.
2. In the window on the right, ensure that the **Design** tab is selected.
3. Select the test operation that you added
4. In the **Inbound processing** window, select **+ Add policy**.
5. Select **Mock responses** from the gallery.
6. In the **API Management response** textbox, type **200 OK, application/json**. This selection indicates that your API should return the response sample you defined in the previous section.
7. Select **Save**.

## Test the mocked API

1. Select the API you created in Create a test API.
2. Select the **Test** tab.
3. Ensure that the **Test call** API is selected, and then select **Send** to make a test call.
4. The **HTTP response** displays the JSON provided as a sample in the first section of the tutorial.

# Create an publish a product

1. In the left navigation pane, select **Products** > **+ Add**.
2. In the **Add product** window, enter values described in the following table to create your product.

|  |  |
| --- | --- |
| Name | Description |
| Display name | The name as you want it to be shown in the developer portal. |
| Description | Provide information about the product such as its purpose, the APIs it provides access to, and other details. |
| State | Select **Published** if you want to publish the product. Before the APIs in a product can be called, the product must be published. By default, new products are unpublished, and are visible only to the **Administrators** group. |
| Requires subscription | Select if a user is required to subscribe to use the product (the product is *protected*) and a subscription key must be used to access the product’s APIs. If a subscription is not required (the product is *open*), a subscription key is not required to access the product’s APIs. |
| Requires approval | Select if you want an administrator to review and accept or reject subscription attempts to this product. If not selected, subscription attempts are auto-approved. |
| Subscription count limit | Optionally limit the count of multiple simultaneous subscriptions. |
| Legal terms | You can include the terms of use for the product which subscribers must accept in order to use the product. |
| APIs | Select one or more APIs. You can also add APIs after creating the product.  If the product is open (does not require a subscription), you can only add an API that is not associated with another open product. |

1. Select **Create** to create your new product.

# Import API

## Import and publish a backend API

1. In the left navigation of your API Management instance, select **APIs**.
2. Select the **Function App** title.
3. In the **Create from Function App** window, select **Full**.
4. Next to the **Function App** text box, click the **Browse** button
5. Next to the **Function App** text box, click the **Select** button
6. Select the appropriate Azure Function App and click the **Select** button
7. Enter the values from the following table.+

|  |  |  |
| --- | --- | --- |
| Setting | Value | Description |
| Display name | After you select the Azure Function App, API Management fills out this field based upon Azure Function App name. When importing an OpenAPI specification, the name comes from the JSON. | A unique name for the API. |
| Name | After you select the Azure Function App, API Management fills this field based upon the Azure Function App name. When importing an OpenAPI specification, the name comes from the JSON. | A unique name for the API. |
| Description | Afer you select the Azure Function App, API Management fills this field based upon the Azure Function App name. When importing an OpenAPI specification, it fills in based upon the JSON. | An optional description of the API. |
| API URL suffix | *Rquote* | The suffix appended to the base URL for the API Management service. API Management distinguishes APIs by their suffix, so the suffix must be unique for every API for a given publisher. |
| Tags |  | Tags for organizing APIs for searching, grouping, or filtering. |
| Products | **Unlimited** | Association of one or more APIs. Each API Management instance comes with two sample products: **Starter** and **Unlimited**. You publish an API by associating the API with a product, **Unlimited** in this example.  You can include several APIs in a product and offer them to developers though the developer portal. To add this API to another product, type or select the product name. Repeat this step to add the API to multiple products. You can also add APIs to products later from the **Settings** page. |
| Gateways | **Managed** | API gateways that expose the API. This field is only available in **Developer** and **Premium** tier services.  **Managed** indicates the gateway built into API Management service and hosted by Microsoft in Azure. Self-hosted gateways are available only in the Premium and Developer tiers. You can deploy them on-premises or in other clouds.  If no gateways are selected, the API will not be available and your API requests will not succeed. |
| Version this API? | Select or deslect |  |

1. Select **Create** to create you API.

## Test the new API in the Azure portal

1. In the left navigation of your API Management instance, select **APIs > rQuote**.
2. Select the **Test** tab, and then select **GetQuoteById**. The page shows **Query parameters** and **Headers**, if any. The **Ocp-Apim-Subscription-Key** is field in automatically for the subscription key associated with this API.
3. Enter a valid value for the **id** parameter (either a number 1 – 72 or random)
4. Select **Send**

The backend responds with **200 OK** and some data.

# Transform and protect your API

## Remove an unwanted response header value

Here we are going to hide HTTP headers that you do not want to show to your users. For example, the **SuperSecretValue** header probably should not be displayed to anonymous users.

1. Select **rQuote** > **Design** > **All Operations**.
2. In the **Outbound processing** section, select **+ Add policy**.
3. In the **Add outbound policy** window, select **Set headers**.
4. To configure the set headers policy, do the following:
   1. Under **Name**, enter **SuperSecretValue**. Under **Action**, select delete.
5. Select **Save**. A **set-header** policy elements appears in the **Outbound processing** section.

## Transform a content-location value to reflect the API Management URL

1. Select **rQuote (Admin)** > **Design** > **All operations**.
2. In the **Outbound processing** section, select the code editor (</>) icon.
3. Position the cursor inside the **<outbound>** element on a blank line.
4. Enter the script below.
5. Click the **Save** button. A **set-header** policy element appears in the **Outbound processing** section.

# Protect an API by adding rate limit policy (throttling)

1. Select **rQuote (Admin)** > **Design** > **All operations**.
2. In the **Inbound processing** section, select the code editor (</>) icon.
3. Position the cursor inside the **<inbound>** element on a blank line. Then, select **Show snippets** at the top-right corner of the screen.
4. In the right window, under **Access restriction policies**, select **Limit call rate per key**.
5. Modify your **<rate-limit-by-key />** code in the **<inbound>** element to the following code. Then select **Save**.

<rate-limit-by-key calls="3" renewal-period="15" counter-key="@(context.Request.IpAddress)" />

# Setup CORS

1. Select **rQuote (Admin)** > **Design** > **All operations**.
2. In the **Inbound processing** section, select the code editor (</>) icon.
3. Add the following to the **<inbound>** element on a blank line.
4. Click the **Save** button.

        <cors>

            <allowed-origins>

                <origin>\*</origin>

            </allowed-origins>

            <allowed-methods>

                <method>\*</method>

            </allowed-methods>

            <allowed-headers>

                <header>\*</header>

            </allowed-headers>

            <expose-headers>

                <header>\*</header>

            </expose-headers>

        </cors>

# Add Caching

1. Select **rQuote** > **Design** > **All operations**
2. In the **Inbound processing** section, select the code editor (</>) icon.
3. Add the following to the **<inbound>** element on a blank line.

        <cache-lookup vary-by-developer="false" vary-by-developer-groups="false" downstream-caching-type="none">

            <vary-by-header>Accept</vary-by-header>

            <vary-by-header>Accept-Charset</vary-by-header>

            <vary-by-header>Authorization</vary-by-header>

        </cache-lookup>

1. Add the following the **<outbound>** element on a blank line.

<cache-store duration="3600" />

# Monitor published APIs

## Debug your APIs

Corina