

Azure API Management Tutorial

Creating, configuring, and accessing APIs

Revisions

Date	Version	Editor	Brief Description
2018-04-19	0.1 (Draft)	Kyle James Karwatski	Creation of document.
2018-04-27	1.0	Kyle James Karwatski	Refined tutorial procedure and naming. Published document to Azure Bootcamp Group.
2022-05-17	1.1	Kyle James Karwatski	Revised company name and contact info. Added links to additional resources. Added screenshots for reference. Removed Logic App components as they are unnecessary. Added extended Function App configuration.
2022-05-19	1.2	Kyle James Karwatski	Added Naming Conventions details. Revised image size and placement. Added more Alt text to images. Modified styling.
2022-06-13	1.3	Kyle James Karwatski	Applied 3Cloud branding (Colors and Fonts). Added comments for storage and app service plans. Updated links to samples in GitHub.

Contents

Overview.....	3	Frontend.....	21
Prerequisites.....	4	Testing.....	23
Technology	4	Internal	23
Recommended Experience	4	External.....	23
Our Model.....	5	Mocking.....	23
Endpoints.....	5	Caching.....	24
Publishing APIs.....	6	Policies	24
Setting up API Management.....	6	Common Policies.....	25
A Note on Naming Conventions.....	6	Named Values/Properties	25
Resource Group.....	6	Simulating Further Development	25
API Management	7	Versions.....	26
Scale and Pricing	10	Revisions.....	26
Groups and Users.....	10	Change Log	27
Creating a Product.....	10	Current Revision	27
Description.....	11	Configuring the POST Operation	28
Approval.....	11	Export/Import	29
Subscription Count Limit.....	11	Export	29
Legal Terms.....	12	Import.....	29
APIs	12	Notifications	29
API URL Suffix	13	Notification Templates.....	29
Version Set.....	13	Developer Portal.....	30
A GET Operation	14	Accessing APIs from your apps.....	30
The Backend	15	C# Console Application.....	30
Prerequisite Resources.....	15	JavaScript Application	31
Function App	16	Contacts.....	32
Configuring the GET Operation	20	3Cloud Solutions.....	32
API Endpoint.....	20	Resources	33

Overview

Azure makes implementing APIs and managing their use simple. Generally, organizations will create Products which are collections of APIs that serve a specific purpose. Those Products and APIs may be used internally by their own developers or externally by other consumers. The usage of these Products and APIs is important to administrate and manage for both technical and business reasons.

General management of APIs is done through the API Management resource in the Azure Portal. Developers can consume your Products and APIs via the Developer Portal.

Please note that the API Management features in the Publisher Portal are now deprecated so only use the Azure Portal itself for administration.

Warning: The procedures demonstrated in this tutorial will incur costs in Azure. Working with API Management will cost some amount even at the lowest pricing tier.

Prerequisites

Technology

To complete this tutorial, you will need the following:

- An account with Microsoft Azure.
- An active Azure subscription.
 - *(Optional)* Developers can obtain an MPN Subscription that provides Azure credits monthly at <https://my.visualstudio.com/benefits>.
- *(Optional)* Visual Studio

Recommended Experience

No formal experience is required for this tutorial; however, knowledge of the following will be helpful:

- General use of Microsoft Azure
- Creating resources in Azure
- APIs and HTTP request/responses
- Usage of JSON formatting

Our Model

This tutorial will create an API that produces [Lorem Ipsum](#) text and converts a length of text to an equivalent Lorem Ipsum representation. For this tutorial, our examples will follow this model:

Resource	Product	API	Version	Operation	Operation Name	Parameter/Header/Body
apim-demo-eastus-dev (API Management)	Texty	Lorem Ipsum	1	GET	GETSomeLoremIpsum	Parameters
						length int; required
						unit string; required [words, characters]
			1 Rev2	GET	GETSomeLoremIpsum	Parameters
						length (int; required)
						unit string; required [words, characters]
				POST	POSTConvertToLoremIpsum	Body
						data string
						Headers
						Content-Type string [application/json]
			2	GET	GETSomeLoremIpsum	Parameters
						length int; required
						unit string; required [words, characters]
				POST	POSTConvertToLoremIpsum	Body
						data string
						Header
						Content-Type string [application/json]

Note: All operations require a subscription-key query string parameter or Ocp-Apim-Subscription-Key header.

Endpoints

Version	Operation	Endpoint
1	GETSomeLoremIpsum	https://{APIM}.azure-api.net/{API}/v1/?{parameters}
1 Rev2	GETSomeLoremIpsum	https://{APIM}.azure-api.net/{API}/v1;rev=2/?{parameters}
1 Rev2	POSTConvertToLoremIpsum*	https://{APIM}.azure-api.net/{API}/v1;rev=2/?{parameters}
2	GETSomeLoremIpsum	https://{APIM}.azure-api.net/{API}/v2/?{parameters}
2	POSTConvertToLoremIpsum*	https://{APIM}.azure-api.net/{API}/v2/?{parameters}

Note: Query string parameters for these POST operations are an optional alternative to request body values.

Publishing APIs

Setting up API Management

The foundation for your APIs will be your API Management resource in Azure.

A Note on Naming Conventions

Strong and consistent naming conventions are always advisable when working with Azure resources. In this tutorial we are flexible in some areas on the naming conventions. Please adhere to your organizations policies on naming conventions and governance when creating resources.

For cases where there are no set rules established, you may want to set a strict convention for resources within the Resource Group you are using.

See <https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-naming> for guidance.

Resource Group

It is a best practice to keep resources related to your APIs in the same Resource Group; however, resource security or networking considerations may determine if exceptions should be made. For this tutorial, we will keep all resources in the same group.

Use an existing Resource Group or create a new one.

1. To create a new **Resource Group**:
 - A. Assign it to a **Subscription**.
 - B. **Name** the resource group **APIManagementDemo**.
 - I. If you are creating this under your organizations subscription you may want to consider using a different name to comply with naming conventions.
 - C. Select a **Location**. In our case we will use **East US**. Use a region close to you for better performance.

Note: It is ideal to use the same location for all related resources.
 - D. Apply **Tags** as needed. <https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/tag-resources?tabs=json>

[Home](#) > [Create a resource](#) > [Resource group](#) >

Create a resource group ...

Basics Tags Review + create

Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. [Learn more](#) ↗

Project details

Subscription * ⓘ	Visual Studio Enterprise Subscription – MPN	▼
Resource group * ⓘ	APIManagementDemo	✓

Resource details

Region * ⓘ	(US) East US	▼
------------	--------------	---

API Management

An API Management resource will host your APIs.

1. Create a new **API Management** resource.
 - A. Assign it to a **Subscription**.
 - B. Select an appropriate **Resource Group**.
 - C. Set the **Location** to the same as your resource group.
 - D. **Name** the resource something that can broadly encompass the APIs that you will build. Consider the business or structural purpose of the instance to help determine the name. For this tutorial we will use **apim-demo-eastus-dev**.
Note: The resource name must be globally unique. The one above will not be available for you to use. See the [A Note on Naming Conventions](#) section for best practices.
 - E. Set the **Organization Name**.
 - F. Select the **Developer Pricing Tier**. See [Scale and Pricing](#) for more information.
*Note: This could take tens of minutes to complete. You may view the progress/time in the **Deployments** section for the resource group.*
 - G. For this tutorial we will leave the remaining options set to default settings in the other tabs. Please note that the configuration for these can be changed at any time; however, it is often ideal to configure them at the time of resource creation.

[Home](#) > [Resource groups](#) > [APIManagementDemo](#) > [Create a resource](#) > [API Management](#) >

Install API Management gateway ...

API Management service

Basics Monitoring Scale Managed identity Virtual network Protocol settings Tags Review + install

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ	<input type="text" value="Visual Studio Enterprise Subscription – MPN"/>
Resource group * ⓘ	<input type="text" value="APIManagementDemo"/>


[Create new](#)

Instance details

Region * ⓘ	<input type="text" value="(US) East US"/>
Resource name *	<input type="text" value="apim-demo-eastus-dev"/>
Organization name * ⓘ	<input type="text" value="3Cloud Solutions"/>
Administrator email * ⓘ	<input type="text" value="kkarwatski@3cloudsolutions.com"/>

Pricing tier

API Management pricing tiers vary in computing capacity per unit and the offered feature set - for example, support for virtual networks, multi-regional deployments, or self-hosted gateways. To accommodate more API requests, consider adding API Management service units instead. [Learn more](#)

 The Developer tier of API Management does not include SLA and should not be used for production purposes. Your service may experience intermittent outages, for example during upgrades. [Learn more](#)

Pricing tier ⓘ	<input type="text" value="Developer (no SLA)"/>
----------------	---

[Home](#) > [Resource groups](#) > [APIManagementDemo](#) > [Create a resource](#) > [API Management](#) >

Install API Management gateway ...

API Management service

[Basics](#) [Monitoring](#) [Scale](#) [Managed identity](#) [Virtual network](#) [Protocol settings](#) [Tags](#) [Review + install](#)

TERMS

By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. See the [Azure Marketplace Terms](#) for additional details.

Basics

Subscription	Visual Studio Enterprise Subscription – MPN
Resource group	APIManagementDemo
Region	East US
Resource name	apim-demo-eastus-dev
Organization name	3Cloud Solutions
Administrator email	kkanwatski@3cloudsolutions.com
Pricing tier	Developer

Monitoring

Application Insights	Disabled
Application Insights instance	-

Scale

Unit(s)	1
---------	---

Managed identity

Identity type	None
---------------	------

Network

Connectivity type	None
-------------------	------

Protocol settings

Triple DES (3DES)	Disabled
HTTP/2	Disabled
TLS 1.1 (HTTP/1.x only)	Disabled
TLS 1.0 (HTTP/1.x only)	Disabled
SSL 3.0 (HTTP/1.x only)	Disabled
TLS 1.1	Disabled
TLS 1.0	Disabled
SSL 3.0	Disabled

After about an hour this instance was fully deployed.

[Home](#) >

 **628286e00b5f77b71ecb41f8** | Overview  ...

Deployment

<<  Delete  Cancel  Redeploy  Refresh


 Overview


 Inputs

 Outputs


 Template

Your deployment is complete

 Deployment name: 628286e00b5f77b71ecb41f8
Subscription: [Visual Studio Enterprise Subscription – MPN](#)
Resource group: [APIManagementDemo](#)

Start time: 5/16/2022, 1:16:17 PM
Correlation ID: ffe03376-e12d-48ca-afe9-5bba4a8e9dcf 

Deployment details [\(Download\)](#)

Resource	Type	Status	Operation details
 apim-demo-eastus-dev	Microsoft.ApiManagement/service	OK	Operation details

Next steps

[Go to resource](#)

Scale and Pricing

Currently, there are 4 pricing tiers: *Developer*, *Basic*, *Standard*, and *Premium*. The Developer tier should specifically be used for development purposes and may not be used for production.

More details can be found at <https://azure.microsoft.com/en-us/pricing/details/api-management/>.

Groups and Users

You can control groups of users for consuming, developing, and administering your API Management.

Creating a Product

Now that our foundation is in place, we can create Products. A Product here is what your subscribing developers will consume. APIs will be associated with these products.

After APIM deployment there are two Products available. For better control over the usage of your APIs it is not recommended to use the "Starter" or "Unlimited" Products and instead create your own. For detailed instructions on Product creation see <https://docs.microsoft.com/en-us/azure/api-management/api-management-howto-add-products?tabs=azure-portal>.

1. In your API Management, select **Products** and add a new one.
 - A. For the **Display Name**, we will call this product **Texty**. This will populate the **ID** as well.
 - B. Add a relevant **Description**.
 - C. Choose **Publish** to make the Product immediately available to developers. This option can be left unselected and publishing can take place later.
 - D. Ensure the checkbox for **Require Subscription** is selected. This means that all requests that come in must have a subscription key included for access to the Product's APIs; otherwise, consumer will receive unauthorized errors.
 - E. We will leave **Requires Approval** unchecked. This option can be selected if you want to manually approve developers subscription requests which is necessary for strict management of the Product access. Since ours will be unselected consumers can request subscriptions and automatically acquire their access without further review.
 - F. We will skip the rest of the settings for now. The **Subscription Count Limit** and **Legal Terms** are optional and may be filled out as you desire. The **APIs** will be added later.

[Home](#) > [APIManagementDemo](#) > [apim-demo-eastus-dev](#) >

Add product ...

API Management service

Display name *

Texty ✓

Id * ⓘ

texty ✓

Description *

A series of APIs dedicated to working with Lorem Ipsum representations. ✓

Published



Requires subscription



Requires approval



Subscription count limit

Legal terms

Unlimited use is granted. We may revoke subscriptions for any reason at any time without notice.

APIs



Description

The description provided for a Product is visible in the Developer Portal. HTML can be used in this field. A concise description that covers a single business or technical purpose is recommended. It is better to be broad in the description so future development efforts are not confounded.

Approval

It is advised to require approval on all Products intended to be used for real consumption of APIs. Any case beyond development purposes should have this extra layer of management to ensure that APIs are not abused and so that resources are not overused, incurring costs.

Subscription Count Limit

A limit of active subscriptions can be imposed. This hard limit will prevent unlimited subscriptions from being created. If the intent is for a small development team to use the Product then a small limit should be imposed to avoid an abundance of neglected subscriptions that could be a security risk. Similarly, impose a limit based on business cases as applicable to better control access.

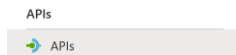
Legal Terms

Legal Terms are optional and also show on the Developer Portal. Use this section to cover limits of use and liability disclosures. This is most relevant for organizations intending to monetize API usage through a Product.

APIs

There are several ways to create or import APIs. All methods share most of the same configuration options.

1. In **API Management**, select **APIs** in the left navigation blade under the APIs section.



2. Several templates will appear. We'll select a **Define a new API > HTTP**. Import options are available here along with templates for Azure resources like Logic Apps. These Templates will prepopulate several aspects of your API, saving you time on configuration.
 - A. Select the Full form toggle to show more fields.

Create an HTTP API

A screenshot of two toggle buttons labeled 'Basic' and 'Full'. The 'Full' button is highlighted with a blue background and a white checkmark, indicating it is the selected option.
 - B. For the **Display Name**, we want to use something that will ultimately describe the categorization and purpose of the API. We'll call this one **Lorem Ipsum**. The **Name** will autofill. There are recommended naming conventions for larger scale operations; however, for the sake of this tutorial, we'll avoid those.
 - C. Write a meaningful **Description**.
 - D. The **Web Service URL** setting will specify where to forward requests to. Typically, this is the location of your API's backend. We will leave this field blank until later in the section **API Endpoint**.
 - E. For **URL scheme** we should always use **HTTPS**. To prevent common attacks, we always want to use the secure option. Really... No exceptions.
 - F. The **API URL suffix** specifies the naming for the API and will be appended to the end of the path.
 - G. The **Base URL** will be generated.
 - H. Select the **Product** that we created above, **Texty**.
 - I. Check the box to enable **Versioning**. More options will appear.
 - I. Select **Path** for **Versioning Scheme**. This defines where the version detail will be specified in the request. In our case, we will use the path.
 - II. In **Version Identifier** type **v1**. This will define how we will refer to our first version.
 - III. By establishing versioning, the API will be organized within a Version Set and will appear grouped in the APIs interface.

J. Click **Create** to complete the process.

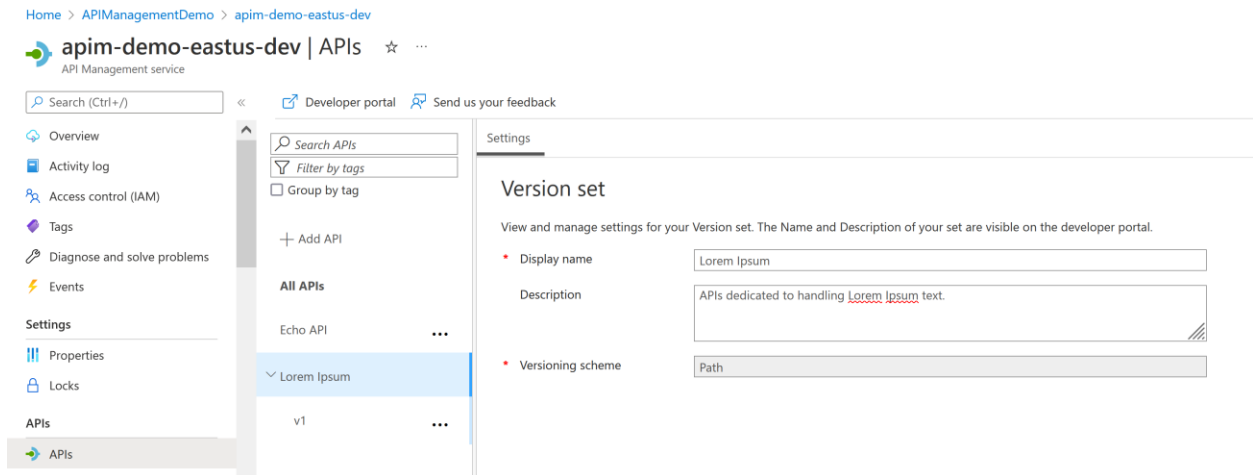
The screenshot shows the Azure API Management console interface. On the left is a navigation pane with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, Settings, Properties, Locks, and APIS. The main area is titled 'Define a new API' and offers three ways to create an API: 'Manually define an HTTP API', 'Create from definition' (with OpenAPI and WADL options), and 'Create from Azure resource' (with Logic App and App Service options). A modal window titled 'Create an HTTP API' is open on the right, showing the 'Basic' tab. The form includes fields for Display name, Name, Description, Web service URL, URL scheme (HTTP, HTTPS, Both), API URL suffix, Base URL, Tags, Products, Gateways, Version this API?, Version identifier, and Versioning scheme. The 'Create' button is at the bottom right of the modal.

API URL Suffix

It is important to specify the organization of your URL structure through components like the URL Suffix. More detailed URLs allow for user/developer URL navigation and interpretation so that patterns and the site or API structure can be inferred. Leverage this based on the expected usage of the APIs. These should NOT match Products or fluid company organization structures.

Version Set

After the API is created with versioning enabled, you can access the Version Set by clicking on the API name that appears in the APIs section of the interface. You'll notice that there is an area for another description. While we previously added a description for the API, we can also add a more encompassing description here. Consider the API description from before as pertaining to the individual version of the API when first created. If, over time, the API matures then maybe that description will change in later versions. The description on the Version Set should cover this possibility.



A GET Operation

Now let's add an operation to our API.

1. In our **Lorem Ipsum API**, add a new operation. This will reveal the Frontend configuration for a new operation.

+ Add operation

- A. Set the **Display Name** for this operation **GETSomeLoremIpsum**. A general practice is to include the operation verb (GET, POST, etc.) at the beginning of the name. The **Name** field will autofill.
- B. In the **URL**, specify the **GET** operation and input a **/**, indicating that our operation will be accessible at the root of the API (and version).
Note: You can add complexity to the URL to meet your API categorization needs.
- C. Fill out a **Description**.

The screenshot shows the Azure API Management console interface. On the left is a navigation pane with categories like Overview, Settings, APIs, and Monitoring. The main area displays the 'Add operation' form for a new API endpoint. The form includes fields for Display name, Name, URL, and Description. The URL field is set to 'GET /'. The Description field contains the text 'Acquire Lorem Ipsum text.' Below the form are tabs for Template, Query, Headers, Request, and Responses. The Template tab is active, showing a section for Template parameters. At the bottom, there are 'Save' and 'Discard' buttons.

For now, let's save and come back after we create the backend.

The Backend

Now that we have the structure of our API in place, it is time to make a meaningful service.

Prerequisite Resources

Before setting up the Function App, we should set up a few resources that it will reference. This is optional because these resources can be created during the creation of the Function App; however, we want to maintain deliberate control over the resources we make and adhere to any governance that has been established.

You have the option to create new storage or use existing. The choice varies depending on the situation. Sometimes grouping storage items is convenient while other times keeping them separate better respects governance and patterns.

The plan used by the function app is often Consumption for smaller traffic setups. In a large scale configuration, using an App Service Plan is advisable. Use this documentation for all of the specifics and determine which plan is necessary in your own projects

<https://docs.microsoft.com/en-us/azure/app-service/overview-hosting-plans>.

Function App

You can create a realistic endpoint in Azure leveraging Function Apps for your backend code.

1. Create a new **Function App** resource.
 - A. Assign it to a **Subscription**.
 - B. Select the **Resource Group** from earlier.
 - C. **Name** the resource something unique and relevant to its purpose. We'll name ours **func-loremipsum-eastus-dev**.
 - D. **Publish** should be set to **Code**.
 - E. The **Runtime Stack** will need to be **Node.js**, the latest version, for this tutorial. However, the JavaScript later doesn't depend on Node.js itself.
 - F. Set the **Region** to the same as your resource group.
 - G. For the **OS**, we'll select **Windows**.
 - H. Our **Hosting Plan** will be a **Consumption Plan**. An App Service Plan exists for more predictable and scalable use.
 - I. Create a new **Storage** space, which is required for a function app. You may elect to use an existing storage space. If you have difficulties creating a Storage Account through this interface, create one as you would typically in Azure and then come back to create the Function App. See <https://docs.microsoft.com/en-us/azure/media-services/latest/storage-create-how-to?tabs=portal> for details.
Note: It may take several minutes for the resource to be created.
 - J. **Monitoring** can be set up later. Select No for Application Insights.

[Home](#) > [APIManagementDemo](#) > [Create a resource](#) > [Function App](#) >


Create Function App


[Basics](#) [Hosting](#) [Networking](#) [Monitoring](#) [Tags](#) [Review + create](#)

Create a function app, which lets you group functions as a logical unit for easier management, deployment and sharing of resources. Functions lets you execute your code in a serverless environment without having to first create a VM or publish a web application.

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * 

Resource Group *  [Create new](#)

Instance Details

Function App name * .azurewebsites.net

Publish * ☒ Code ☐ Docker Container

Runtime stack *

Version *

Region *

Operating system

The Operating System has been recommended for you based on your selection of runtime stack.

Operating System * ☐ Linux ☒ Windows

Plan

The plan you choose dictates how your app scales, what features are enabled, and how it is priced. [Learn more](#)

Plan type * 

[Home](#) > [APIManagementDemo](#) > [Create a resource](#) >

Create Function App

[Basics](#) [Hosting](#) [Networking](#) [Monitoring](#) [Tags](#) [Review + create](#)

Storage

When creating a function app, you must create or link to a general-purpose Azure Storage account that supports Blobs, Queue, and Table storage.

Storage account * [Create new](#)

[Home](#) > [APIManagementDemo](#) > [Create a resource](#) >

Create Function App ...

Basics Hosting Networking Monitoring Tags Review + create

Summary



Details

Subscription	92b59909-4af2-4a7d-afad-7726b3cbfe82
Resource Group	APIManagementDemo
Name	func-loremipsum-eastus-dev
Runtime stack	Node.js 16 LTS

Hosting

Storage

Storage account	saapimdemoeastusdev
-----------------	---------------------

Plan (New)

Plan type	Consumption (Serverless)
Name	ASP-APIManagementDemo-8711
Operating System	Windows
Region	East US
SKU	Dynamic

Monitoring

Application Insights	Not enabled
----------------------	-------------

2. Within that Function App, add a new **Function** called **ProduceLoremIpsum**. This function will generate sample text for our GETSomeLoremIpsum operation in our Lorem Ipsum API.
 - A. The **Development Environment** is most convenient when set to **Develop in Portal**.
 - B. Select **HTTP trigger** template.
 - C. Set the function name to **ProduceLoremIpsum**.
 - D. The **Authorization Level** is best scoped at the lowest level. Set to **Function**.

Create function >

Select development environment
Instructions will vary based on your development environment. [Learn more](#)

Development environ... Develop in portal

Select a template
Use a template to create a function. Triggers describe the type of events that invoke your functions. [Learn more](#)

Filter

Template	Description
HTTP trigger	A function that will be run whenever it receives an HTTP request, responding based on data in body or query string
Timer trigger	A function that will be run on a specified schedule
Azure Queue Storage trigger	A function that will be run whenever a message is added to a specified Azure Storage queue
Azure Service Bus Queue trigger	A function that will be run whenever a message is added to a specified Service Bus queue
Azure Service Bus Topic trigger	A function that will be run whenever a message is added to the specified Service Bus topic
Azure Blob Storage trigger	A function that will be run whenever a blob is added to a specified container

Template details
We need more information to create the HTTP trigger function. [Learn more](#)

New Function* ProduceLoremIpsum

Authorization level Function

3. For a more secure and manageable experience with our newly created Function App, we want to make a new, unique, Function Key. Within the Function navigate to **Function Keys**.

- A. The name we will give this key should be contextually relevant to its purpose. We'll call ours "**apimdemokey**".
- B. We will use the generated key later.

Home > APIManagementDemo > func-loremipsum-eastus-dev > ProduceLoremIpsum

ProduceLoremIpsum | Function Keys

Function

Search (Ctrl+F) Refresh

Overview

Developer

Code + Test

Integration

Monitor

Function Keys

Function Keys
Function keys are scoped to this function and can be used to access this function.

+ New function key Show values

Filter functions keys

Name	Value
default	Hidden value. Click to show value

Add key

Name apimdemokey

Value Leave empty to auto-generate a key

4. Navigate to **Code + Test**. Replace the code in this function with the backend code **ProduceLoremIpsum.js** code in the **Resources** section at the end of this document. Save the code.

Home > APIManagementDemo > func-loremipsum-eastus-dev > ProduceLoremIpsum

ProduceLoremIpsum | Code + Test ...

Function

Search (Ctrl+/) << Save Discard Refresh Test/Run Upload Get function URL

Overview

Developer

- Code + Test
- Integration
- Monitor
- Function Keys

func-loremipsum-eastus-dev \ ProduceLoremIpsum \ index.js

```

10 module.exports = function (context, req) {
11   ... context.log('JavaScript HTTP trigger function processed a request.');
```

- Test to see that the code works as expected. Expand the **Test** tab and place parameters in the **Query** section for Unit and Length. The content of the request body is not applicable or required.

A. The code provided accepts the query string parameters for "length" and "unit".

Input Output

Provide parameters to test the HTTP request. Results can be found in the Output tab.

HTTP method

POST

Key

apimdemokey (Function key)

Query

Name	Value
length	100
unit	words

+ Add parameter

Headers

+ Add header

Body

1

Input Output

HTTP response code

200 OK

HTTP response content

Lorem Ipsum aenean et tortor at. Massa tempor nec feugiat nisl pretium fusce id velit. Etiam tempor orci eu lobortis elementum nibh tellus molestie. Convallis posuere morbi leo urna. Magna fermentum iaculis eu non. Augue neque gravida in fermentum et sollicitudin ac orci phasellus. Quisque id diam vel quam elementum. Quam nulla porttitor massa id neque aliquam vestibulum morbi blandit. Dictumst vestibulum rhoncus est pellentesque elit ullamcorper. Leo integer malesuada nunc vel risus commodo viverra. Turpis egestas integer eget aliquet nibh praesent tristique magna sit. Leo integer malesuada nunc vel risus commodo. Egestas fringilla phasellus faucibus scelerisque eleifend donec pretium.

Configuring the GET Operation

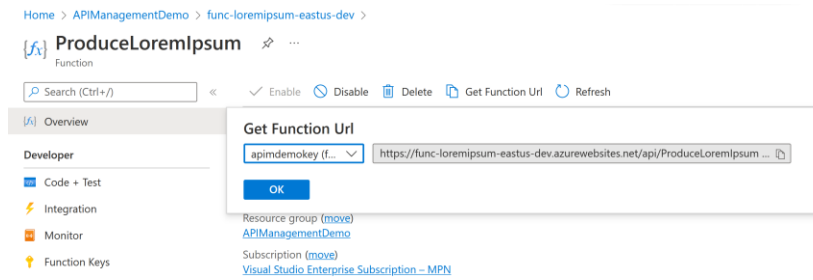
API Endpoint

Your API isn't valuable without a functional service. The endpoints of your API or Operations can be an Azure resource or external endpoint.

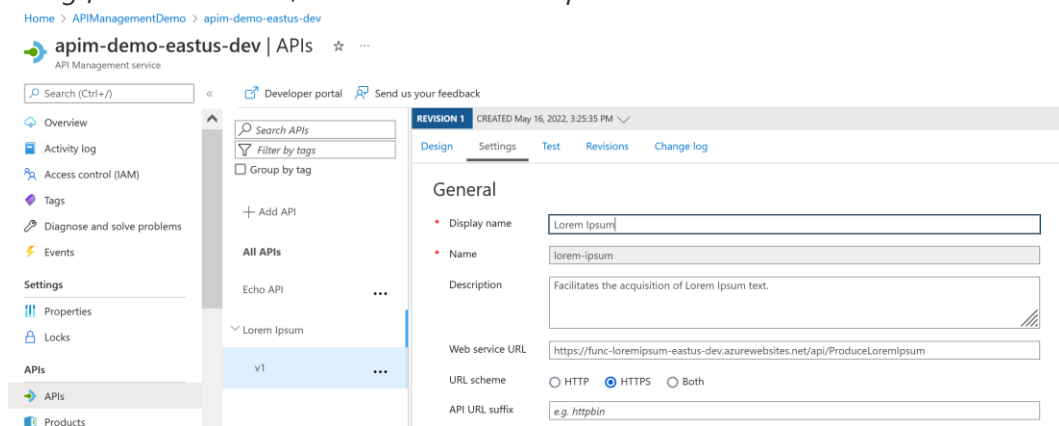
- Acquire the **Function URL** from the Function App.
 - Within the Function select the **Get Function Url** button.

Get Function Url

- The URL generated will contain the Service URL and the Function Key. We only want the Service URL (i.e. everything before the "?").



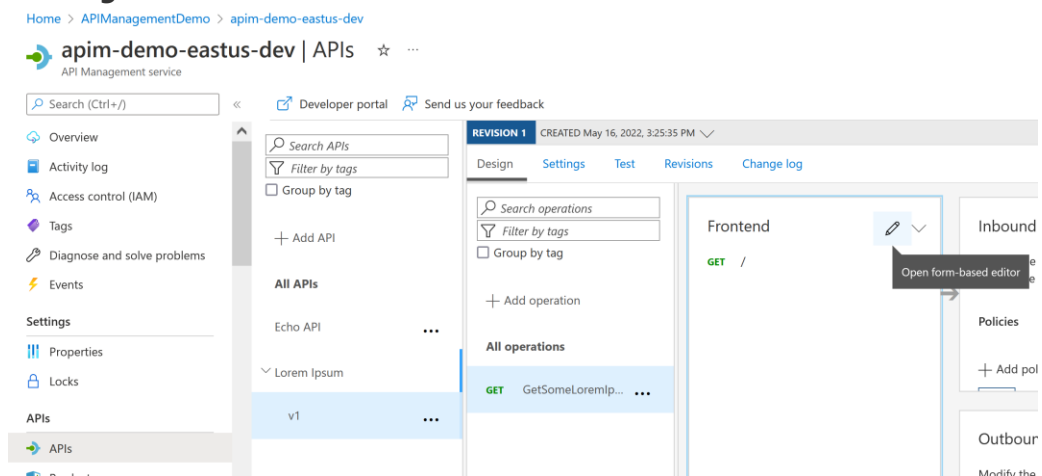
2. Set the **Web Service URL** in the API.
 - A. Within the API version we previously created, go to **Settings**.
 - B. Here we can populate the Web Service URL with the value we obtained above.
Note: Even though it may seem intuitive to include the Function Key in the query string for the URL here, it will not work as expected. We will work on that next.



Frontend

The Frontend is where template parameters and other specification related features are configurable.

1. Now we must set the parameter that controls access to our backend.
 - A. In **Design**, select the **Edit** icon in the **Frontend** section.



- B. Go to the **Query** tab and select **Add parameter**.

- I. Populate the **Name** with "**code**", which is the query string key for the Function Key.
- II. Add the **Description** to show that this is the Function Key.
- III. Set the **Type** to "**string**"
- IV. For the **Value** we must add a default designation. Select the radio button next to the empty text box and fill the text value in with the Function Key value that we previously generated.
- V. Select the checkbox for **Required**. Requests can only be authorized if this key is sent along with them so we must require it.

The screenshot shows the Azure API Management console for the 'apim-demo-eastus-dev' instance. The left sidebar contains navigation options like Overview, Activity log, Access control (IAM), Tags, and Settings. The main pane displays the 'Design' tab for the 'GET /getSomeLoremIpsum' operation. Under the 'Query parameters' section, a table lists the parameters:

NAME	DESCRIPTION	TYPE	VALUES	REQUIRED	DELETE
code	Azure Function Key value	string	<input checked="" type="radio"/> No default <input type="radio"/> r4wlr3v5aDRzpdYWoYehHs	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Buttons for 'Save' and 'Discard' are at the bottom.

- c. Add additional parameters for "length" and "unit" as defined in **Our Model**.
 - I. We set a few **Values** here. For **length** we want a default value of 100. For **unit** we want "words" and "characters" to be values. "words" is the default.

This screenshot shows the same console after adding two more parameters. The 'Query parameters' table now includes:

NAME	DESCRIPTION	TYPE	VALUES	REQUIRED	DELETE
code	Azure Function Key value	string	<input checked="" type="radio"/> No default <input type="radio"/> r4wlr3v5aDRzpdYWoYehHs	<input checked="" type="checkbox"/>	<input type="checkbox"/>
length	The length of the desired output text.	integer	<input checked="" type="radio"/> 100 <input type="radio"/> 200	<input checked="" type="checkbox"/>	<input type="checkbox"/>
unit	The unit in which the length is specified. Words or Characters are available.	string	<input checked="" type="radio"/> words.characters <input type="radio"/> characters	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Now, our API is ready for testing.

Testing

It is important to test the API to make sure that all of the configuration has been properly applied.

Internal

With our parameters (code, length, unit) specified, we can just go to the **Test** tab in and send a request.

The screenshot shows the Azure API Management console for the 'apim-demo-eastus-dev' environment. The 'Test' tab is selected for the 'GetSomeLoremIpsum' operation. The 'Query parameters' section contains three parameters: 'code' (string, value: 'r4w13v5aDRpdeYWoYehbQmz_Xg4u3Uu'), 'length' (integer, value: '100'), and 'unit' (string, value: 'words'). The 'Request URL' is 'https://apim-demo-eastus-dev.azure-api.net/v1/code=r4w13v5aDRpdeYWoYehbQmz_Xg4u3Uu?length=100&unit=words'. The 'HTTP request' shows a GET request with the same URL and headers. The 'HTTP response' shows a 200 OK status with a JSON body containing Lorem Ipsum text.

External

We can also test our APIs externally as well using a browser or other REST application.

Mocking

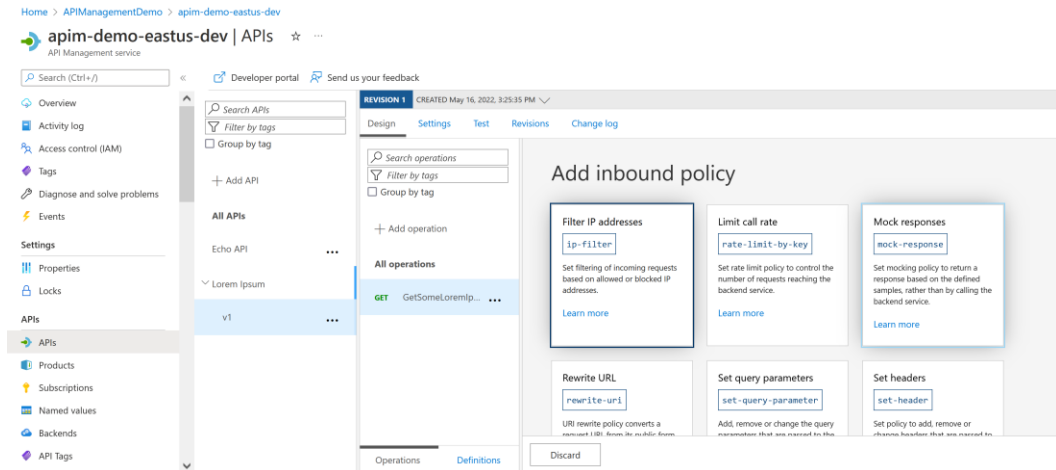
If you have just set up your service and want to test it out without having your backend in place, you can emulate responses through mocking. Once configured, static responses will be returned as defined, regardless of the backend. This is useful when testing your API without having your backend in place.

1. In your API, select your operation or all operations, and edit **Inbound Processing**.

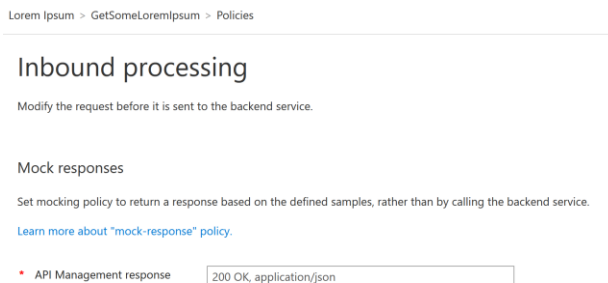
- A. Add a policy provides multiple UI features so that the policy XML does not have to be manually edited.

+ Add policy

- B. Select **Mock Responses**.



- C. 200 will be specified by default.



Caching

To improve response time, you can configure caching. Because of the more random nature of our functions, caching is not going to be practical. You would use caching if the same response is frequently generated. See <https://docs.microsoft.com/en-us/azure/api-management/api-management-caching-policies> for more information.

Policies

General API security is governed by Policies. Policies can be applied at a Product, all APIs, a single API, and an Operation level, each having a respective scope.

Policies can be defined in the interface or code view.

This reference is helpful for getting familiar with policies: <https://docs.microsoft.com/en-us/azure/api-management/api-management-policies>.

Common Policies

- **IP Filtering:** Restrict access to the API based on a range of (or a single) IP addresses.
- **Rate Limit:** Restrict the number of requests per unit of time by subscriber.
- **Quota:** Set a bandwidth quota by subscriber.
- **CORS:** Set the support for Cross-Domain referencing. This may be required for web applications to use your API.

Named Values/Properties

In the code view, you may notice IDs for API components referenced instead of just the resource name. Every resource has one of these "Named Values". They can be modified in API Management > Named Values.

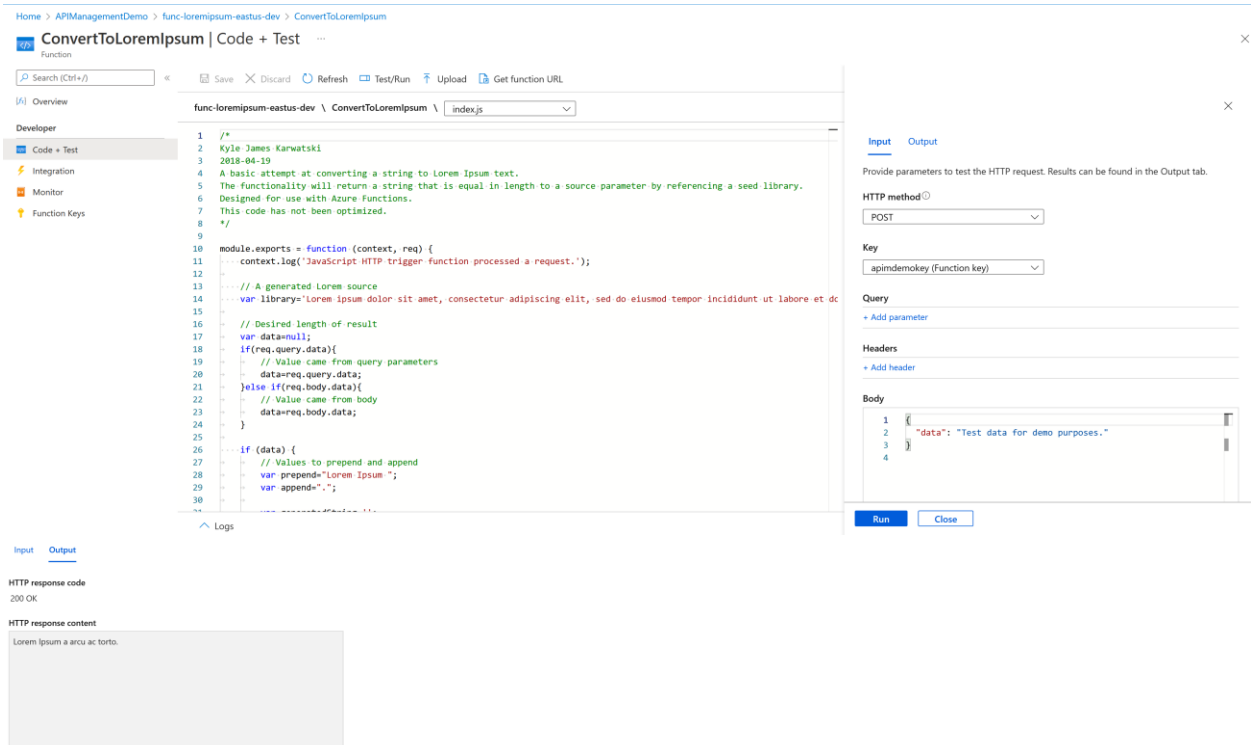
Simulating Further Development

Now we can make our system a little more robust by adding another set of functionality. Our next operation is going to take some text string and return a Lorem Ipsum string of similar length.

For this one, we'll take the same Function App approach.

1. In the same Function App as before, let's add a new HTTP trigger **Function** called **ConvertToLoremIpsum**.
2. Create a new **Function Key** like before. You can name this one the same as the other (**apimdemokey**) because it is in the context of the new function.
3. Replace the code in this function with the **ConvertToLoremIpsum.js** code in the **Resources** section at the end of this document.
4. Test to ensure that the code works as expected. Add a JSON object to the **Request Body** section with one property called Data and a value with text of any length.


```
{
  "data": "Test data for demo purposes."
}
```



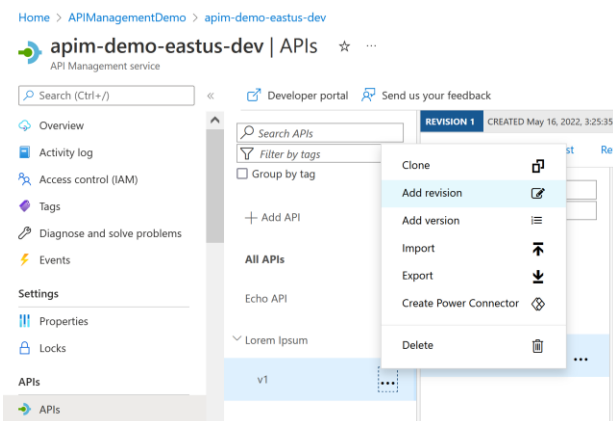
Versions

Ideally, you will make revisions where applicable and then break out into versions. Because we are simulating a development process, we will make this new operation a part of a new revision before we version it.

Revisions

In common practice, revisions should contain changes to an API. We are adding support for a new operation for our API in this tutorial so we will create a revision from our API.

1. In **API Management**, go to **APIs**. Click the **Ellipsis** next to **v1** of our **Lorem Ipsum API**. Select **Add Revision**.



- A. A dialog will appear. Fill out the **Revision Description** with relevant information about this revision. This description should differ from the API's description and should be specific enough about your expected changes to the first revision.

Create a new revision of Lorem Ipsum-v1

Create a new revision of the **Lorem Ipsum-v1** API. A revision is a copy of your API that you can use to model and test changes. When you are ready, you can make this new revision current and replace your current API.

Revision description

Facilitates the acquisition and conversion of Lorem Ipsum text.

Create

Cancel

You may toggle between the active revisions in the APIs interface near the top where it displays the revision and update time.

REVISION 2	CREATED May 17, 2022, 4:06:59 PM	
Revision 2 - Created May 17, 2022, 4:06:59 PM		
Revision 1 - Created May 16, 2022, 3:25:35 PM		

Change Log

The Change Log tab in the APIs interface is where you can provide some detailed notes (more documentation) for your subscribers about the history of the version of the API. These notes are version specific.

Current Revision

After making versions and revisions you can then set one to "Current". Current indicates to subscribers that this is the stable/preferred version. This feature is available in the revisions tab of the APIs interface.

We'll want to leave our first revision current for now. Once we set up the next operation we can designate the next revision as current.

Home > APIManagementDemo > apim-demo-eastus-dev

apim-demo-eastus-dev | APIs

API Management service

Search (Ctrl+/)

Developer portal Send us your feedback

Overview Activity log Access control (IAM) Tags Diagnose and solve problems Events Settings Properties Locks APIs

Search APIs Filter by tags Group by tag

+ Add API

All APIs

Echo API

▼ Lorem Ipsum

v1

REVISION 2 CREATED May 17, 2022, 4:06:59 PM

Design Settings Test Revisions Change log

Revisions

ID	CREATED	DESCRIPTION	URL	ONLINE	CURRENT
2	May 17, 2022, 4:06:59 PM	Facilitates the acquisition and conversion of Lorem Ipsum text.	//v1,rev=2	✓	...
1	May 16, 2022, 3:25:35 PM		//v1	✓	✓

+ Add revision

Configuring the POST Operation

In the **Lorem Ipsum API**, add an operation called **POSTConvertToLoremIpsum**. Configure this just as in the

Configuring the GET Operation section except with different parameters as defined in the **Our Model** section.

The URL will be set to "/" just like our GET operation. The design of our API is to allow for requests to be sent to the same endpoint and the operation verb will differentiate the service.

It will be necessary to edit the Web Service URL after creating the operation.

REVISION 2 CREATED May 17, 2022, 4:06:59 PM

Design Settings Test Revisions Change log

Search operations
Filter by tags
Group by tag

+ Add operation

All operations

GET GetSomeLoremIp... ..

POST POSTConvertToL... ..

Lorem Ipsum > POSTConvertToLoremIpsum > Frontend [OpenAPI specification V](#)

Frontend

- Display name: POSTConvertToLoremIpsum
- Name: postconverttoloremipsum
- URL: POST /
- Description: Converts the provided data to a Lorem Ipsum representation.
- Tags: e.g. Booking

REVISION 2 CREATED May 17, 2022, 4:06:59 PM

Design Settings Test Revisions Change log

General

- Display name: Lorem Ipsum
- Name: lorem-ipsum;rev=2
- Description: Facilitates the acquisition of Lorem Ipsum text.
- Web service URL: https://func-loremipsum-eastus-dev.azurewebsites.net/api/ConvertToLoremIpsum
- URL scheme: ☐ HTTP ☒ HTTPS ☐ Both
- API URL suffix: e.g. httpbin
- Base URL: https://apim-demo-eastus-dev.azure-api.net

Now, our API is complete.

Export/Import

APIs can be exported from and imported into an API Management. These features make it convenient to replicate or backup APIs. The features are granular enough to cover individual revisions.

Export

1. In **API Management > APIs**, select the **Ellipsis** next to the API [or version] that you wish to export. Click **Export**.
2. Various export options will appear. Select the **Revision** that you wish to export.
3. Chose a **Format** to export. Typically, the **OpenAPI Specification** (formerly Swagger) is preferred. This will prompt a download dialog.

Import

You may import into a new API or to an existing one.

1. For a new API:
 - A. In **API Management > APIs**, select the **Add API**.
 - B. Various options will appear. Choose the **Format** that you will import.
 - C. A dialog will appear. You may import from a **File** or **URL**.
 - D. Other options may be available depending on the format you select.
2. For an existing API:
 - A. In **API Management > APIs**, select the **Ellipsis** next to the API [or version] that you wish to export. Click **Import**.
 - B. Various import options will appear. Select the **Revision** that you wish to import into.
 - C. Choose the **Format** that you will import.
 - D. A dialog will appear. You may import from a **File** or **URL**.
 - E. Other options may be available depending on the format you select.

Notifications

When subscribing developers interact with your products via the Developer Portal notifications are produced. You can set up notifications to send emails based on specific actions.

Notification Templates

Templates for the notification emails are available for edit for all notification scenarios. You may modify these to address your needs.

Developer Portal

The Developer Portal and its appropriate use cases warrant their own tutorial. See the documentation for more information <https://docs.microsoft.com/en-us/azure/api-management/api-management-howto-developer-portal>.

Accessing APIs from your apps

1. In the Developer Portal, go to the APIs tab.
2. Select the API or version of an API. In this case, select **Lorem Ipsum > v1**.
3. Select the operation of interest. Choose **GETSomeLoremIpsum**.
4. At the bottom of the page in the **Code Samples** section there will be samples of preconfigured code for use within your applications.

In practice, developers will interact with the Developer Portal to acquire subscriptions and keys and then add those to an external application or other Azure resource to allow interactions with APIM. Below are a couple of examples demonstrating the possibilities of API integration with custom apps.

C# Console Application

Sample available at

<https://github.com/kylekarwatski/APIMTutorial/tree/master/API%20Samples/API%20Service%20Sample%20CSharp>.

This app will send a GET request to the API > GETSomeLoremIpsum.

1. Copy the code from the **C#** sample and paste it into a new console application (.NET Framework) project in Visual Studio.
 - A. You may have to splice the source code with your project template (i.e. only copy over the references and the contents of the Program class).
 - B. It may be necessary to add references for System, System.Net.Http, or System.Web to your project. The other references may not be necessary and can be deleted.
2. Add the following line of code to the end of the MakeRequest() method. This code will print out the response from the **GETSomeLoremIpsum** operation.
 - A. `Console.WriteLine(response.Content.ReadAsStringAsync().Result);`
3. Replace the subscription key value ({subscription key}) with the **Texty** Product's subscription key. You can acquire this key from your subscription in the Developer's Portal > Products > Texty and selecting the Texty subscription which will bring you to a list of subscription keys. Clicking **Show** will reveal the key value. For you, as an administrator, this key will be the admin key which is available in the Azure Portal as well.

A developer will have their own unique subscription key once they subscribe to a Product.

4. In the code, alter the **Request Parameters** as you see fit.
5. In Visual Studio, select **Start** to run your program. If the API and console application has been properly configured, you will see the result in the console window.

JavaScript Application

Sample available at

<https://github.com/kylekarwatski/APIMTutorial/tree/master/API%20Samples/API%20Service%20Sample%20JavaScript>.

This app will send a POST request to the API > POSTConvertToLoremIpsum.

1. Copy the code from the **JavaScript** sample and paste it into a new HTML file.
Note: This can be a plain text file or a project in an IDE if you'd like.
2. Replace the subscription key value ({subscription key}) with the **Texty** Product's subscription key. You can acquire this key from your subscription in the Developer's Portal > Products > Texty and selecting the Texty subscription which will bring you to a list of subscription keys. Clicking **Show** will reveal the key value. For you, an administrator, this key will be the admin key which is available in the Azure Portal as well. A developer will have their own unique subscription key once they subscribe to a Product.
3. In the code, alter the **Request Parameters** as you see fit.
4. You can execute the script by opening it locally in a web browser. You will see a popup with "Success" or "Failure" to show you the status of your script. If the script is successful, you can view the result in the developer's tools provided by the browser or you could alter the code to display it on the page. If you receive a failure, you may have to set a CORS policy at the API level to allow the application to run from a different domain, shown below.

```
<cors>
  <allowed-origins>
    <!-- allow any -->
    <origin>*</origin>
  </allowed-origins>
  <allowed-methods>
    <!-- allow any -->
    <method>*</method>
  </allowed-methods>
  <allowed-headers>
    <!-- allow any -->
    <header>*</header>
  </allowed-headers>
</cors>
```


Contacts

3Cloud Solutions

Kyle James Karwatski, *Senior Application Engineer*

KKarwatski@3CloudSolutions.com

(724) 624-3832 (M)

Resources

1. Azure API Management Documentation
<https://docs.microsoft.com/en-us/azure/api-management/>
2. Produce Lorem Ipsum JavaScript code
<https://github.com/kylekarwatski/APIMTutorial/blob/master/ProduceLoremIpsum.js>
3. Convert to Lorem Ipsum JavaScript code
<https://github.com/kylekarwatski/APIMTutorial/blob/master/ConvertToLoremIpsum.js>
4. Naming Conventions Reference
<https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-naming>
5. Azure Resource Tags Reference
<https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/tag-resources?tabs=json>
6. APIM Pricing Details
<https://azure.microsoft.com/en-us/pricing/details/api-management/>
7. Create a Storage Account
<https://docs.microsoft.com/en-us/azure/media-services/latest/storage-create-how-to?tabs=portal>
8. App Service Plans
<https://docs.microsoft.com/en-us/azure/app-service/overview-hosting-plans>
9. Developer Portal
<https://docs.microsoft.com/en-us/azure/api-management/api-management-howto-developer-portal>
10. APIM Caching Policies
<https://docs.microsoft.com/en-us/azure/api-management/api-management-caching-policies>
11. C# API Sample
<https://github.com/kylekarwatski/APIMTutorial/tree/master/API%20Samples/API%20Service%20Sample%20CSharp>
12. JavaScript API Sample
<https://github.com/kylekarwatski/APIMTutorial/tree/master/API%20Samples/API%20Service%20Sample%20JavaScript>