

Microsoft Translator Text API

Para utilizar el servicio de traductor o bot usaremos la Microsoft Translator Text API. El Microsoft Translator API se pueden integrar perfectamente en sus aplicaciones, sitios web, herramientas u otras soluciones para proporcionar experiencias de usuario en varios idiomas. Aprovechando los estándares de la industria, se puede usar en cualquier plataforma de hardware y con cualquier sistema operativo para realizar traducciones de idiomas y otras operaciones relacionadas con el lenguaje, como la detección de texto o texto a voz. [Haga clic aquí](#) para obtener más información acerca de la Microsoft Translator API

En esta demostración, voy a utilizar Microsoft Bot Framework con Cognitive Services para demostrar cómo crear un Bot que pueda hablar varios idiomas.

La API de texto de Microsoft Translator admite más de 50 idiomas y puede ser consumida fácilmente por Bot Framework.

Demo:

Crearemos una demo de Bot que responderá a las preguntas que puedan tener los clientes sobre un hotel en el que quieran alojarse. El usuario podrá hablar con el Bot en cualquiera de los idiomas compatibles y recibir una respuesta en ese mismo idioma.

Primeros pasos para usar la API de Microsoft Translator

Para acceder a la API de texto de Microsoft Translator, deberá registrarse en Microsoft Azure. Sigue estos pasos.

1. Regístrese para obtener una cuenta de Microsoft Azure en <http://azure.com>
2. Seleccione la opción + **Nuevo**.
3. Seleccione **AI + Cognitive Services** de la lista de servicios.
4. Haga clic en **Ver todo** en la parte superior derecha.
5. Seleccione la **API de texto del traductor**.
6. Seleccione el botón **Crear**.
7. Complete el resto del formulario.
8. En la sección **Nivel de precios**, seleccione el nivel de precios que se ajuste a sus necesidades.
9. Seleccione el botón **Crear**.

Create

Translator Text API

Name
TranslatorSubscriptionKey

Subscription
Visual Studio Enterprise

Pricing tier (View full pricing details)

Resource group
☒ Create new
 ☐ Use existing

TestBot

☒ I confirm I have read and understood the notice below.

Microsoft will use data you send to the Cognitive Services to improve Microsoft products and services. Where you send personal data to the Cognitive Services, you are responsible for obtaining sufficient consent from the data subjects. The General Privacy and Security Terms in the Online Services Terms do not apply to the Cognitive Services. Please refer to the Microsoft Cognitive Services section in the [Online Services Terms](#) for details. Microsoft offers policy controls that may be used to [disable new Cognitive Services deployments](#).

☐ Pin to dashboard

Create Automation options

Choose your pricing tier

Browse the available plans

The cost of your Cognitive Services depends on the actual usage and the options you choose below. [Learn more](#)

F0 Free	S1 Standard	S2 Standard
2M Up to 2M characters tr...	Pay as you go	250M Characters monthly c...
Translate supported la...	Translate supported la...	Translate supported I...
Full featured	Language Detection	Language Detection
Language Detection	Translation Customiza...	Translation Customiz...
Text to Speech	Text to Speech	Text to Speech
0,00 USD/MILLION CHARACTERS. NO O...	10,00 USD/MILLION CHARACTERS	2.055.00 USD/MONTH PLUS OVERAGE

S3 Standard	S4 Standard
1B Characters monthly co...	10B Characters monthly co...
Translate supported la...	Translate supported la...
Language Detection	Language Detection
Translation Customiza...	Translation Customiza...
Text to Speech	Text to Speech
6 USD/Million Overage Characters	4.5 USD/Million Overage Characters

Select

- Ahora está suscrito a Microsoft Translator.
- Vaya a **Todos los recursos** y seleccione la API de Microsoft Translator a la que se suscribió.
- Vaya a la opción **Llaves** y copie su clave de suscripción para acceder al servicio.

TranslatorSubscriptionKey - Keys

Cognitive Services

Search (Ctrl+/)

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems

RESOURCE MANAGEMENT

- Keys
- Quick start

Regenerate Key1 Regenerate Key2

Notice: It may take up to 10 minutes for the newly (re)generated keys to take effect

NAME
TranslatorSubscriptionKey
KEY 1 773e72d24e8d4375b577272094d5514f
KEY 2 d89c82222f1c4a2abb37b5f58a19aef1

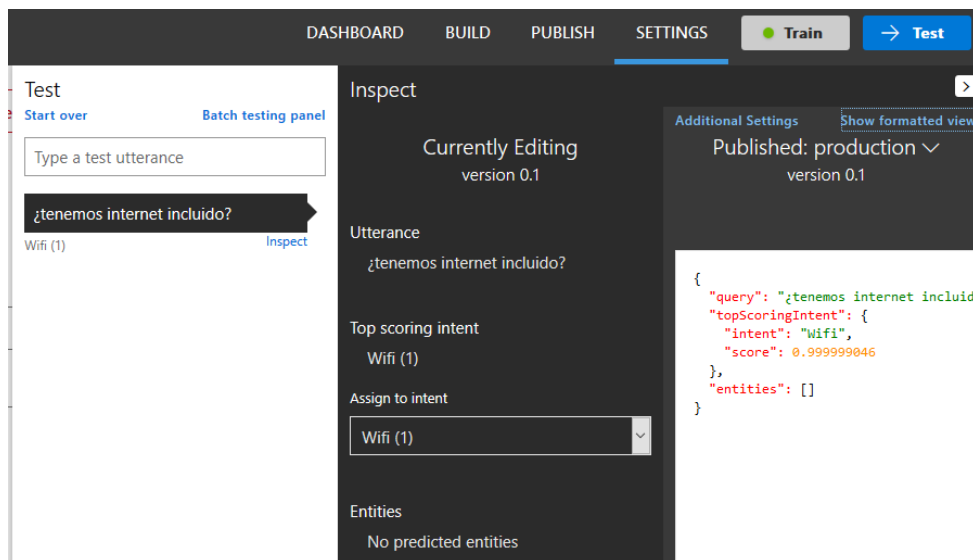
Utilizar la Translator Text API en el Bot

Ahora tenemos nuestra clave de suscripción, así que estamos listos para usar la API de traducción en nuestro Bot.

Entrenado el modelo LUIS para comprender muchas preguntas que un cliente puede tener sobre las instalaciones de un hotel, como restaurantes, piscina, ubicación, wifi.

El modelo LUIS está disponible para descargar en [Github](#).

Cualquier texto que enviemos al modelo LUIS debe estar en español.



The screenshot displays the Microsoft Bot Framework LUIS console interface. The top navigation bar includes 'DASHBOARD', 'BUILD', 'PUBLISH', and 'SETTINGS', along with 'Train' and 'Test' buttons. The 'Test' tab is active, showing a 'Test' panel on the left with a 'Type a test utterance' input field and a 'Batch testing panel' link. Below the input, a test utterance '¿tenemos internet incluido?' is shown with a 'Wifi (1)' entity prediction and an 'Inspect' link. The 'Inspect' panel is open, displaying the 'Currently Editing' version 0.1. It shows the 'Utterance' as '¿tenemos internet incluido?', the 'Top scoring intent' as 'Wifi (1)', and the 'Assign to intent' dropdown set to 'Wifi (1)'. The 'Entities' section shows 'No predicted entities'. On the right, the 'Additional Settings' panel shows 'Published: production' version 0.1 and a 'Show formatted view' link. The JSON output for the query is displayed in a code block:

```
{
  "query": "¿tenemos internet incluid",
  "topScoringIntent": {
    "intent": "Wifi",
    "score": 0.999999046
  },
  "entities": []
}
```

1. Obtenga el token de acceso para la autenticación de la API del traductor

Cada solicitud a la API del traductor debe contener un encabezado de Autorización, con un token de acceso enviado como un token de portador.

Headers | {Authorization: Bearer eyJ0eXAI0iJKV1QILCJhbGciOiJIUzI1NiJ9.eyJzY29wZSI6Imh0dHBzOi8v

Para obtener este token de acceso, enviamos nuestra clave de suscripción al servicio de token de Servicios Cognitivos. La clave de suscripción se envía en el encabezado **Ocp-Apim-Subscription-Key**.

El servicio identifica su clave como válida y le devuelve un token.

Ahora que comprendemos cómo funciona la Autorización para la API de traductor, echemos un vistazo al código.

El archivo web.config contiene el URI para el servicio de token de Servicios Cognitivos, junto con la clave de suscripción de Azure.

```
<appSettings>
  <add key="BotId" value="YourBotId" />
  <add key="MicrosoftAppId" value="" />
  <add key="MicrosoftAppPassword" value="" />
  <add key="SubscriptionKey" value="f751d..." />
  <add key="CognitiveServicesTokenUri" value="https://api.cognitive.microsoft.com/sts/v1.0/issueToken" />
  <add key="TranslatorUri" value="https://api.microsofttranslator.com/v2/Http.svc/" />
</appSettings>
```

Enviamos la solicitud del token de acceso en la clase **AzureAuthToken**.

El método **GetAccessTokenAsync** envía una solicitud HTTP POST al servicio de tokens de Servicios Cognitivos, con el **encabezado Ocp-Apim-Subscription-Key**

```
public async Task<string> GetAccessTokenAsync()
{
    if (string.IsNullOrWhiteSpace(this.SubscriptionKey))
    {
        return string.Empty;
    }

    // Return the cached token if it is still valid
    if ((DateTime.Now - _storedTokenTime) < TokenCacheDuration)
    {
        return _storedTokenValue;
    }

    try
    {
        using (var client = new HttpClient())
        using (var request = new HttpRequestMessage())
        {
            request.Method = HttpMethod.Post;
            request.RequestUri = ServiceUri;
            request.Content = new StringContent(string.Empty);
            request.Headers.TryAddWithoutValidation(OcpApimSubscriptionKeyHeader, this.SubscriptionKey);
            client.Timeout = TimeSpan.FromSeconds(180);
            var response = await client.SendAsync(request);
            this.RequestStatusCode = response.StatusCode;
            response.EnsureSuccessStatusCode();
            var token = await response.Content.ReadAsStringAsync();
            _storedTokenTime = DateTime.Now;
            _storedTokenValue = "Bearer " + token;
            return _storedTokenValue;
        }
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

request	{Method: POST, RequestUri: 'https://api.cognitive.microsoft.com/sts/v1.0/issueToken', Version: 1.1, ...}
Content	{System.Net.Http.StringContent}
Headers	{Ocp-Apim-Subscription-Key: f751d ... }
Method	{POST}
RequestUri	{https://api.cognitive.microsoft.com/sts/v1.0/issueToken}

Analizamos la respuesta para obtener el token, que está en formato JWT. Y lo guardamos en caché durante los próximos 5 minutos.

Tenga en cuenta que el tiempo de espera de un token de acceso es de 10 minutos.

```
public async Task<string> GetAccessTokenAsync()
{
    if (string.IsNullOrWhiteSpace(this.SubscriptionKey))
    {
        return string.Empty;
    }

    // Return the cached token if it is still valid
    if ((DateTime.Now - _storedTokenTime) < TokenCacheDuration)
    {
        return _storedTokenValue;
    }

    try
    {
        using (var client = new HttpClient())
        using (var request = new HttpRequestMessage())
        {
            request.Method = HttpMethod.Post;
            request.RequestUri = ServiceUrl;
            request.Content = new StringContent(string.Empty);
            request.Headers.TryAddWithoutValidation(OcpApimSubscriptionKeyHeader, this.SubscriptionKey);
            client.Timeout = TimeSpan.FromSeconds(180);
            var response = await client.SendAsync(request);
            this.RequestStatusCode = response.StatusCode;
            response.EnsureSuccessStatusCode();
            var token = await response.Content.ReadAsStringAsync();
            _storedTokenTime = DateTime.Now;
            _storedTokenValue = "Bearer " + token; //cached for 5 mins
            return _storedTokenValue;
        }
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

Este es el token que usaremos para detectar el idioma del texto enviado al bot, y traduciremos al inglés si es necesario.

Una vez que esté en inglés, lo enviaremos a LUIS para obtener el Intent.

2. Detecta el idioma del mensaje entrante

Ahora que tenemos Autenticación para la configuración de la API de texto del traductor, podemos detectar el idioma del mensaje entrante. Esto sucede en el método **POST** de la clase **MessageController**

```
public async Task<HttpResponseMessage> Post([FromBody]Activity activity)
{
    if (activity.Type == ActivityTypes.Message)
    {
        activity.Text = TranslationHandler.DetectAndTranslate(activity);
        await Conversation.SendAsync(activity, MakeRoot);
    }
    else
    {
        HandleSystemMessage(activity);
    }
    var response = Request.CreateResponse(HttpStatusCode.OK);
    return response;
}
```

La clase **TranslationHandler** maneja la detección y traducción del texto, utilizando el método **DetectAndTranslate**

```
private static string DoLanguageDetection(string input)
{
    var translator = new Translator();
    return translator.Detect(input);
}
```

Para hacer la detección de idioma, creamos una nueva instancia de la clase **Translator**.

Esta clase tiene un método llamado **Detect** que envía la solicitud a la API del traductor.

```
internal string Detect(string input)
{
    try
    {
        string uri = "https://api.microsofttranslator.com/v2/Http.svc/Detect?text=" + HttpUtility.UrlEncode(input);
        WebRequest translationWebRequest = WebRequest.Create(uri);
        translationWebRequest.Headers.Add("Authorization", Bearer);

        WebResponse response = null;
        response = translationWebRequest.GetResponse();
        Stream stream = response.GetResponseStream();
        Encoding encode = Encoding.GetEncoding("utf-8");

        StreamReader translatedStream = new StreamReader(stream, encode);
        XmlDocument xTranslation = new XmlDocument();
        xTranslation.LoadXml(translatedStream.ReadToEnd());

        return xTranslation.InnerText;
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

El URI para la Solicitud tiene el siguiente formato. Observe que el texto entrante se adjunta a la cadena de consulta.

RequestUri	{https://api.microsofttranslator.com/v2/Http.svc/Detect?text=tiene+wifi}
-------------------	--

3. Traduzca el mensaje entrante

Si el idioma detectado no es inglés, debemos traducirlo al inglés antes de reenviar el texto a LUIS.

La solicitud de traducción es casi idéntica a la solicitud de detección.

```
public string Translate(string inputText, string inputLocale, string outputLocale)
{
    try
    {
        string uri =
            "https://api.microsofttranslator.com/v2/Http.svc/" +
            $"Translate?text={HttpUtility.UrlEncode(inputText)}&from={inputLocale}&to={outputLocale}";

        WebRequest translationWebRequest = WebRequest.Create(uri);
        translationWebRequest.Headers.Add("Authorization", Bearer);

        WebResponse response = null;
        response = translationWebRequest.GetResponse();
        Stream stream = response.GetResponseStream();
        Encoding encode = Encoding.GetEncoding("utf-8");

        StreamReader translatedStream = new StreamReader(stream, encode);
        XmlDocument xTranslation = new XmlDocument();
        xTranslation.LoadXml(translatedStream.ReadToEnd());

        return xTranslation.InnerText;
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

4. Utilice Bot State para guardar el código de idioma del usuario

Utilizamos Bot State disponible en Bot Framework para permitirnos responder a cada usuario en el mismo idioma que su mensaje entrante.

Después de detectar el idioma, usamos la clase **StateHelper** para establecer el código de idioma **para ese usuario**.

El Servicio de Estado de Bot Framework nos permite guardar datos **por usuario**. Esta información puede ser utilizada en todas las conversaciones para ese usuario.

```
public static void SetUserLanguageCode(IDialogContext context, string languageCode)
{
    try
    {
        context.UserData.SetValue("LanguageCode", languageCode);
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

5. Responda al usuario en su LanguageCode

Cuando LUIS devuelve un Intento a la clase **ChatDialog**, usamos una respuesta *predefinida* o *enlatada* para enviar de vuelta al usuario. Todas las respuestas se almacenan en la clase **ChatResponse**.


```
[LuisIntent("Wifi")]
public async Task Wifi(IDialogContext context, LuisResult result)
{
    var response = ChatResponse.Wifi;

    await context.PostAsync(response.ToUserLocale(context));

    context.Wait(MessageReceived);
}
```

Estas respuestas se almacenan en inglés.

Usamos el método de extensión **ToUserLocale** para buscar el estado de Bot para el **código** de *idioma* del usuario; si el **código** de *idioma* no es inglés, traducimos la respuesta al **código** de *idioma*.

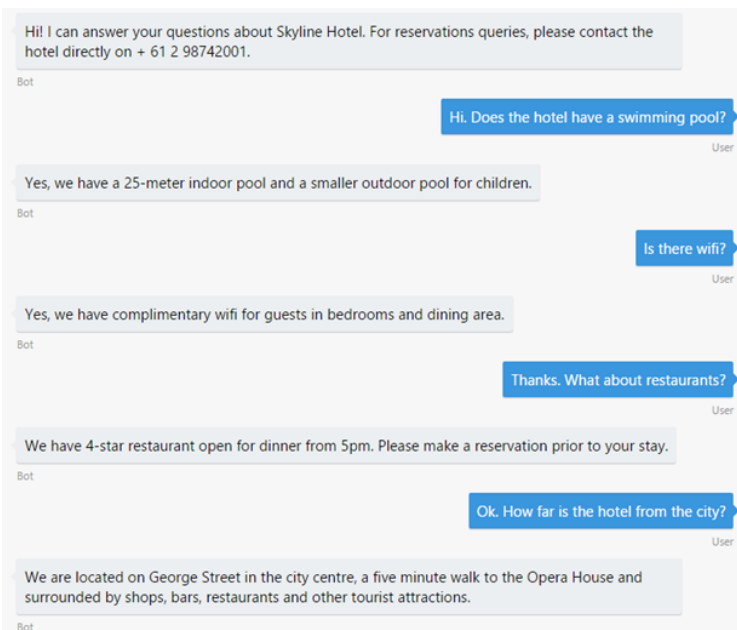
El texto traducido luego se devuelve al usuario.

```
public static string ToUserLocale(this string text, IDialogContext context)
{
    var userLanguageCode = StateHelper.GetUserLanguageCode(context);
    if (userLanguageCode != "en")
    {
        text = TranslationHandler.DoTranslation(text, "en", userLanguageCode);
    }
    return text;
}
```

Paso 3: el resultado: un bot de varios idiomas

Ahora ingresaremos texto en el [emulador de Framework Bot](#) - el texto estará en **inglés**, **español** y **chino (simplificado)** y el Bot responderá en el mismo idioma que el texto de entrada.

Inglés



The screenshot shows a chat window with a light gray background. On the left, the bot's responses are in white bubbles with gray borders. On the right, the user's questions are in blue bubbles with white borders. The conversation is as follows:

- Bot:** Hi! I can answer your questions about Skyline Hotel. For reservations queries, please contact the hotel directly on + 61 2 98742001.
- User:** Hi. Does the hotel have a swimming pool?
- Bot:** Yes, we have a 25-meter indoor pool and a smaller outdoor pool for children.
- User:** Is there wifi?
- Bot:** Yes, we have complimentary wifi for guests in bedrooms and dining area.
- User:** Thanks. What about restaurants?
- Bot:** We have 4-star restaurant open for dinner from 5pm. Please make a reservation prior to your stay.
- User:** Ok. How far is the hotel from the city?
- Bot:** We are located on George Street in the city centre, a five minute walk to the Opera House and surrounded by shops, bars, restaurants and other tourist attractions.

Español

Hi! I can answer your questions about Skyline Hotel. For reservations queries, please contact the hotel directly on + 61 2 98742001.

Bot

El hotel tiene una piscina?

User

Sí, tenemos una piscina cubierta de 25 metros y una piscina más pequeña para los niños.

Bot

tiene wifi?

User

Sí, tenemos wifi gratuito para los huéspedes en habitaciones y comedor.

Bot

Gracias. Y Restaurantes?

User

Dispone de un restaurante de 4 estrellas abierto para la cena de 17:00. Por favor, reservar antes de su estancia.

Bot

Bueno. A qué distancia de la ciudad es el hotel?

User

Estamos situados en George Street en el centro de la ciudad, a cinco minutos andando a la ópera y rodeados de tiendas, bares, restaurantes y otras atracciones turísticas.

Bot at 8:08:06 AM

Chino (Simplificado)

Hi! I can answer your questions about Skyline Hotel. For reservations queries, please contact the hotel directly on + 61 2 98742001.

Bot

你好。酒店里有一个游泳池吗？

User

是的我们有一个 25 米长的室内游泳池和一个较小的室外游泳池儿童。

Bot

你有互联网吗？

User

是的我们有提供免费 wifi 上网为客人在卧室和用餐区。

Bot

谢谢你。餐厅怎么样？

User

我们有 4 星级的餐厅，晚餐从 5 下午打开。请帮我预约之前往。

Bot

还行。有多远从城里的酒店？

User

我们是在乔治街位于城市中心，歌剧院有五分钟步行路程，周围的商店、酒吧、餐馆和其他旅游景点。

Bot at 8:10:16 AM

Nota:

Incluso se puede cambiar el idioma durante la conversación y el Bot entenderá.

Código fuente

El código fuente se puede descargar en: [Lab Localización](#)