Microsoft

Microsoft®
Terminology Service API

Last updated: January 2022

Microsoft Terminology Service API

The Microsoft Terminology Service API allows applications to look up terminology translations and user-interface translations from actual Microsoft products. The Terminology Service API is provided as a SOAP API, and it can be used with any technology capable of creating SOAP requests and parsing SOAP responses.

New features in version 2.0 (June 2014)

There are several new features provided in the version 2.0 release of the Microsoft Terminology Web Service. These features are:

- Support for “any to any” language translation searches. In previous versions of the API, searching was restricted to either en-us to a target language, or to en-us from a source language. You can now search from any language to any language.
- The ability to filter your search with string case and hotkey sensitivity. A new parameter in the GetTranslations method allows you to specify the sensitivity level.

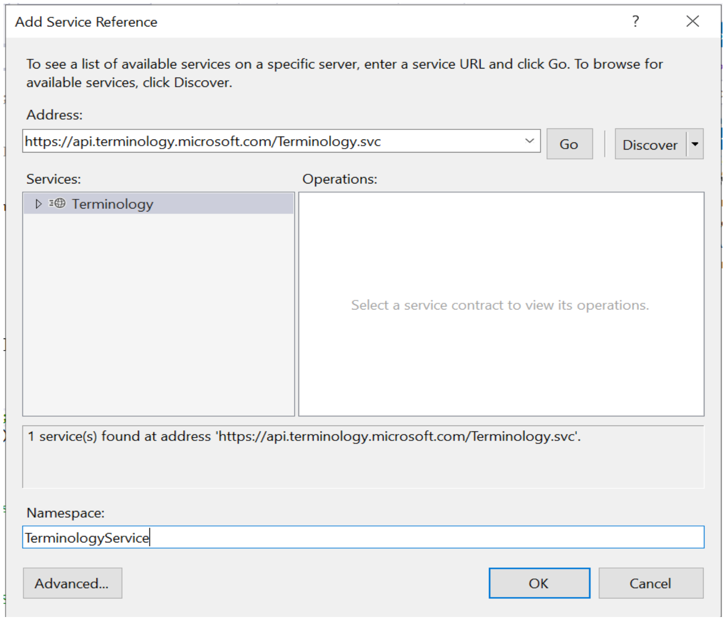
Although the GetTranslations method signature has an added parameter, existing client code will remain compatible and will not break existing programs. To take advantage of the new features, however, applications will need to regenerate their client proxy to retrieve the new method signature.

Using the Terminology Service API in Visual Studio

You can add the Terminology Service API as a Windows® Communication Foundation (WCF) service reference to a Microsoft Visual Studio® project so that SOAP requests and responses are handled automatically for your application. Here are the general steps for adding a service reference to a Visual Studio 2012 project:

1. Ensure that your Visual Studio project is configured to use the Microsoft .NET 3.5 Framework, .NET Framework 4, or .NET Framework 4.5. If not, you will have to add a web reference instead of a WCF service reference.
2. Right-click the project, and then select **Add Service Reference**.
3. Enter the URL *https://api.terminology.microsoft.com/Terminology.svc* into the **Address** field, and then click **Go**.A SOAP service should appear in the **Services** list.

4. Set the 'TerminologyService' in the **Namespace** box, and then click **OK**. Client proxy classes are added to the project.



To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address: https://api.terminology.microsoft.com/Terminology.svc

Select a service contract to view its operations.

1 service(s) found at address 'https://api.terminology.microsoft.com/Terminology.svc'.

Namespace: TerminologyService

5. Now you can create an instance of the **TerminologyClient** proxy class in the **TerminologyService** namespace to call methods of the SOAP API.

C#

```
TerminologyService.TerminologyClient service = new
TerminologyService.TerminologyClient();
```

6. After creating an instance of the TerminologyClient proxy class, you can call its GetTranslations method to retrieve translations. A simple example of calling GetTranslations is illustrated in the following code snippet:

C#

```
using System;
using SampleApplication.TerminologyService;

namespace SampleApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            // Create a collection to define the desired sources of translations
            TranslationSources translationSources = new TranslationSources() ↵
              {TranslationSource.Terms, TranslationSource.UiStrings};
            // Create the proxy for the Terminology Service SOAP client
            TerminologyClient service = new
TerminologyService.TerminologyClient();
            // Call GetTranslations to get the results
            Matches results = service.GetTranslations("start button", "en-us",
"es-es", ↲
              SearchOperator.Contains, translationSources, false, 20, true,
null);

            // Use the results
            foreach (Match match in results)
            {
                Console.WriteLine(match.OriginalText);
                Console.WriteLine(match.Translations[0].TranslatedText);
            }
        }
    }
}
```

Public Methods

The following public methods are available in the Terminology Service API:

- **GetCapability**
- **GetLanguages**
- **GetProducts**
- **GetTranslations**

Each method is covered in more detail on the subsequent pages.

GetCapability Method

The **GetCapability** method returns an object that defines the capabilities of the current version of the Terminology Service API. For example, it allows checking whether the API supports a reverse translation from a target language to English (United States).

Syntax

C#

```
Capability GetCapability();
```

Return Value

Type: TerminologyService.Capability

A **Capability** object with the following properties:

Property	Description
SupportsEnUsToAny	If true, **GetTranslations** supports translating en-us text to any other language returned by the **GetLanguages** method.
SupportsAnyToEnUs	If true, **GetTranslations** supports translating to en-us from any other language returned by the **GetLanguages** method.
SupportsAnyToAny	If true, **GetTranslations** supports translating from any language to any other language, provided both languages are returned by the **GetLanguages** method.

Examples

C#

```
// Create the proxy for the Terminology Service SOAP client
TerminologyClient service = new TerminologyService.TerminologyClient();
// Call GetCapability
Capability capability = service.GetCapability();
Console.WriteLine("Can the service translate from any language to any other language? "
+ capability.SupportsAnyToAny);
Console.WriteLine("Can the service translate from any language to en-us? " +
capability.SupportsAnyToEnUs);
Console.WriteLine("Can the service translate from en-us to any other language? " +
capability.SupportsEnUsToAny);
```

SOAP Request

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
      <s:Body>
         <GetCapability xmlns="http://api.terminology.microsoft.com/terminology"/>
      </s:Body>
</s:Envelope>
```

GetLanguages Method

The **GetLanguages** method returns the list of language codes supported by the Terminology Service API.

Syntax

C#

```
Languages GetLanguages();
```

Return Value

Type: TerminologyService.Languages

This method is a **Languages** collection of **Language** objects. A **Language** object consists of the following property:

Property	Description
Code	The language code (for example, en-us, es-es) that can be used as a "to" or "from" parameter on the **GetTranslations** method.

Examples

C#

```
// Create the proxy for the Terminology Service SOAP client
TerminologyClient service = new TerminologyService.TerminologyClient();
// Call GetLanguages
Languages languages = service.GetLanguages();
// Use the results
foreach (Language language in languages)
{
    Console.WriteLine(language.Code);
}
```

SOAP Request

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
      <s:Body>
             <GetLanguages xmlns="http://api.terminology.microsoft.com/terminology"/>
      </s:Body>
</s:Envelope>
```

GetProducts Method

The **GetProducts** method returns the list of Microsoft products and versions for which this Terminology Service API provides user-interface translations.

Syntax

C#

```
Products GetProducts();
```

Return Value

Type: TerminologyService.Products

This method is a **Products** collection of **Product** objects. A **Product** object consists of the following properties:

Property	Description
Name	The name of the Microsoft product (such as Windows).
Versions	A **Versions** collection of **Version** objects, representing the versions of the product for which the Terminology Service API provides user-interface translations.

A **Version** object consists of the following property:

Property	Description
Name	The name of the Microsoft product version (for example, Windows *7* or Windows *Vista*).

Examples

C#

```
// Create the proxy for the Terminology Service SOAP client
TerminologyClient service = new TerminologyService.TerminologyClient();
// Call GetProducts
Products products = service.GetProducts();
// Use the results
foreach (Product product in products)
{
    Console.WriteLine(product.Name);
    foreach (TerminologyService.Version version in product.Versions)
    {
        Console.WriteLine("\t" + version.Name);
    }
}
```

SOAP Request

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
   <GetProducts xmlns="http://api.terminology.microsoft.com/terminology"/>
  </s:Body>
</s:Envelope>
```

GetTranslations Method

The **GetTranslations** method returns a list of translations for the given text by searching actual Microsoft product translations and Microsoft terminology glossaries.

Syntax

C#

```
Matches GetTranslations(string text, string from, string to, SearchOperator
searchOperator, TranslationSources sources, bool? unique, int maxTranslations, bool
includeDefinitions, Products products);
```

Parameters

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td><b>text</b></td><td><b>Required.</b> A string representing the text to translate.</td></tr>
<tr><td><b>from</b></td><td><b>Required.</b> A string representing the language code of the provided text. The language codes used for this parameter must be a language code returned by <b>GetLanguages</b>.</td></tr>
<tr><td><b>to</b></td><td><b>Required.</b> A string representing the language code in to which to translate the text. The language codes used for this parameter must be a language code returned by <b>GetLanguages</b>.
In version 1.0 of the Terminology Service API, either the <b>from</b> or the <b>to</b> parameter must be “en-us.” In version 2.0, this restriction no longer applies.</td></tr>
<tr><td><b>sensitivity</b></td><td><b>New in version 2.0</b>
<b>Optional.</b> A SearchStringComparison enum value representing the sensitivity to filter results. The value can be one of the following:
<table>
<tr><td><b>CaseInsensitive</b></td><td>(default) Return translations in which the “from” text searched disregards the case of the text. A search for “Cat” would return both:
“Cat”
and
“cat”</td></tr>
<tr><td><b>CaseSensitive</b></td><td>Return translations in which the “from” text searched takes the case of the text into account. Only results matching the case of the “from” are returned. A search for “Cat” would return:
“Cat”
but not:
“cat”</td></tr>
<tr><td><b>HotKeyAndCaseSensitive</b></td><td>Return translations in which the “from” text searched takes the case of the text into account, along with any hotkeys in the string. Only results matching the case of the “from” are returned. A search for “&Cat” would return:
“&Cat”
but not:
“&cat”
or
“Cat”</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td><b>operator</b></td><td><b>Optional.</b> A SearchOperator enum value representing the type of matching operation to use. The value can be one of the following:
<table>
<tr><td><b>Exact</b></td><td>Return translations in which the provided text has an exact match to the translation pair’s “from” text (that is, Match.OriginalText).</td></tr>
<tr><td><b>Contains</b></td><td>Return translations in which the “from” text contains the provided translation text.</td></tr>
<tr><td><b>AnyWord</b></td><td>Return translations in which the “from” text contains any word in the provided translation text. This means that a search for:
“Lorem rutrum risus quis nulla ullamcorper”
Can even result in the hit:
“Lorem ipsum dolor sit amet, {0}, consectetur adipiscing elit”
Notice that there is only one word that matches. However, realize that results with more matching words will be ranked higher. A one word match isn’t likely to be in the top results.</td></tr>
</table>
<b>Note:</b> If this parameter is not provided, the <b>Exact</b> operator is used. Even though this parameter is not required in a SOAP request, a proxy class generated by Visual Studio will require it.</td></tr>
<tr><td><b>source</b></td><td><b>Required.</b> A TranslationSources type collection representing the sources in which to search for a translation. The collection must contain one or more of the following TerminlogySource enum values:
<table>
<tr><td><b>Terms</b></td><td>If provided, Microsoft terminology collections are searched.</td></tr>
<tr><td><b>UiStrings</b></td><td>If provided, Microsoft product strings are searched for translations.</td></tr>
</table></td></tr>
<tr><td><b>unique</b></td><td><b>Optional.</b> A nullable bool indicating whether or not only unique (that is, distinct) translations should be returned. If <b>true</b> is specified, the results are aggregated so that each distinct translation only appears once. If <b>false</b> is specified, the results are not aggregated, but each instance is returned. If null is provided or if this parameter is omitted from the SOAP request, the default value is <b>true</b>.</td></tr>
<tr><td><b>maxTranslations</b></td><td><b>Optional.</b> An int representing the maximum number of translations to return. The maximum allowed value is 20. If this parameter is omitted from the SOAP request, the default value, 1, is used.</td></tr>
<tr><td><b>includeDefinitions</b></td><td><b>Optional.</b> A bool indicating whether or not to include term definitions. If <b>true</b>, definitions are returned for the terms in the result set (if available in the data source). If <b>unique</b> is specified as <b>true</b>, the first definition for each unique set of translation pairs is used. If this parameter is omitted from the SOAP request, the default value, <b>false</b>, is used.</td></tr>
<tr><td><b>products</b></td><td><b>Optional.</b> A <b>Products</b> collection representing the products and product versions for which to filter the search results. Each <b>Product</b> entity in the collection may include a <b>Versions</b> collection to further restrict the results for that product. Valid products and versions are returned by the <b>GetProducts</b> method.
If this parameter is omitted from the SOAP request or if the parameter is set to null, results are not filtered by products and versions.
If the <b>Versions</b> collection for a product is null or empty, results matching the product are not filtered by version.
When the products parameter is provided, the search only includes items from the <b>UiStrings</b> source of translations.</td></tr>
</table>

Note: Even though the optional parameters listed are optional in the SOAP request, a WCF proxy generated by Visual Studio will not create these parameters as C# optional parameters.

Return Value

Type: TerminologyService.Matches

This method is a **Matches** collection of **Match** objects. A **Match** object consists of the following properties used to represent a translation pair, and to define where the translation pair comes from.

Property	Description
OriginalText	The **from** text version of the translation pair. If the **Contains** operator is used, this text may be different than the input translation text. Additionally, even if the **Exact** operator is used, there may be differences in punctuation and character casing.
Translations	A **Translations** type collection of **Translation** objects. In version 1, this collection only contains one entry. In future versions, the API may support returning more than one language translation at a time.
Count	This property specifies how many matches with the same **OriginalText** and **TranslatedText** are in the combined data sources.
ConfidenceLevel	An int representing the translation confidence level, between the value of 0 and 100. The value is set according to the following rules: • If the **OriginalText** is an exact match of the input text, this value is 100. Punctuation and text case are ignored when comparing the text. • If the **OriginalText** is not an exact match, but contains the input text, this value is 50. If the **operator** parameter is set to **Exact** there is never a result with the **ConfidenceLevel** of 50, so this is only relevant if the API is called with the **operator** parameter set to **Contains**.
Source	A **TranslationSource** enum value indicating the source of this match. This value is only provided if the **unique** parameter is false. It can be one of the following values: **Terms** Microsoft terminology collections. **UiStrings** Microsoft product string translations.
Product	A string representing the Microsoft product to which this translation belongs. This value is only provided if the **unique** parameter is false, and if the **Source** of the match is **UiStrings**.
ProductVersion	A string representing the Microsoft product version to which this translation belongs. This value is only provided if the **unique** parameter is false, and if the **Source** of the match is **UiStrings**.
Definition	If the **includeDefinitions** parameter is provided, this string represents the definition of the matching term.

A **Translation** object consists of the following properties:

Property	Description
Language	A string representing the language code of the translation.
TranslatedText	The translated text, in the language specified by the Language property.

Sorting

The sorting of the **Match** objects in the return value depends on the following rules:

- If the **unique** parameter is true, the **Match** objects are sorted in the following priority:
  - **ConfidenceLevel** (descending)

- **Count** (descending)

- If the **unique** parameter is false, the **Match** objects are sorted in the following priority:
  - **Source** (**UiStrings** first)
  - **Product** (ascending)
  - **ProductVersion** (descending)
  - **ConfidenceLevel** (descending)
  - **Count** (descending)

Note: The top 20 results are determined first by making **ConfidenceLevel** the highest priority. After the top 20 are determined, the results are sorted by **Source**, **Product**, **ProductVersion**, **ConfidenceLevel**, and then **Count**.

Examples

C#

```
// Create a collection to define the desired sources of translations
TranslationSources translationSources = new TranslationSources() {
TranslationSource.UiStrings };
// Create a collection to define the desired products and versions from which to get the
translations
Product word = new Product() { Name = "Windows" };
word.Versions = new Versions() { new TerminologyService.Version() { Name = "Vista" },
new TerminologyService.Version() { Name = "7" } };
Products products = new Products() { word };
// Create the proxy for the Terminology Service SOAP client
TerminologyClient service = new TerminologyService.TerminologyClient();
// Call GetTranslations to get the results
Matches results = service.GetTranslations("start", "en-us", "es-es",
SearchStringComparison.CaseInsensitive, SearchOperator.Contains, translationSources,
false, 20, true, products);
// Use the results
foreach (Match match in results)
{
    Console.WriteLine();
    Console.WriteLine("OriginalText: {0}", match.OriginalText);
    Console.WriteLine("ConfidenceLevel: {0}", match.ConfidenceLevel);
    Console.WriteLine("Definition: {0}", match.Definition);
    Console.WriteLine("Count: {0}", match.Count);
    Console.WriteLine("Source: {0}", match.Source);
    Console.WriteLine("Product: {0}", match.Product);
    Console.WriteLine("ProductVersion: {0}", match.ProductVersion);
    Console.WriteLine("Translations");
    Console.WriteLine("____________");
    foreach (Translation translation in match.Translations)
    {
        Console.WriteLine("{0}: {1}", translation.Language, translation.TranslatedText);
    }
}
```

SOAP Request

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
    <s:Body>
        <GetTranslations xmlns="http://api.terminology.microsoft.com/terminology">
            <text>1 hours ago</text>
            <from>en-us</from>
            <to>nl-nl</to>
            <sensitivity>CaseSensitive</sensitivity>
            <searchOperator>Exact</searchOperator>
            <sources xmlns:a="https://api.terminology.microsoft.com/terminology"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
                <a:TranslationSource>Terms</a:TranslationSource>
                <a:TranslationSource>UiStrings</a:TranslationSource>
            </sources>
            <unique>true</unique>
            <maxTranslations>4</maxTranslations>
            <includeDefinitions>true</includeDefinitions>
            <products i:nil="true"
xmlns:a="https://api.terminology.microsoft.com/terminology"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance"/>
        </GetTranslations>
    </s:Body>
    </s:Envelope>
```

Language and Locale Codes

The **GetLanguages** method returns the list of language codes supported by the Terminology Service API. The following table shows the names of the languages and locales represented by the codes.

Language Code	Language and country/region
af-ZA	Afrikaans (South Africa)
am-ET	Amharic (Ethiopia)
ar-EG	Arabic (Egypt)
ar-SA	Arabic (Saudi Arabia)
as-IN	Assamese (India)
az-Latn-AZ	Azerbaijani (Latin, Azerbaijan)
be-BY	Belarusian (Belarus)
bg-BG	Bulgarian (Bulgaria)
bn-BD	Bangla (Bangladesh)
bn-IN	Bangla (India)
bs-Cyrl-BA	Bosnian (Cyrillic, Bosnia and Herzegovina)
bs-Latn-BA	Bosnian (Latin, Bosnia and Herzegovina)
ca-ES	Catalan (Catalan)
ca-ES-valencia	Valencian
chr-cher-us	Cherokee (USA)
cs-CZ	Czech (Czech Republic)
cy-GB	Welsh (United Kingdom)
da-DK	Danish (Denmark)
de-AT	German (Austria)
de-CH	German (Switzerland)
de-DE	German (Germany)
el-GR	Greek (Greece)
en-AU	English (Australia)
en-CA	English (Canada)
en-GB	English (United Kingdom)
en-IE	English (Ireland)
en-MY	English (Malaysia)
en-NZ	English (New Zealand)
en-PH	English (Republic of the Philippines)
en-SG	English (Singapore)
en-US	English (United States)
en-ZA	English (South Africa)
es-ES	Spanish (Spain)
es-MX	Spanish (Mexico)
es-US	Spanish (United States)
et-EE	Estonian (Estonia)
eu-ES	Basque (Basque)
fa-IR	Persian
fi-FI	Finnish (Finland)
fil-PH	Filipino (Philippines)
fr-BE	French (Belgium)

fr-CA	French (Canada)
fr-CH	French (Switzerland)
fr-FR	French (France)
fr-LU	French (Luxembourg)
ga-IE	Irish (Ireland)
gd-GB	Scottish Gaelic (United Kingdom)
gl-ES	Galician (Galician)
gu-IN	Gujarati (India)
guc-VE	Wayuu
ha-Latn-NG	Hausa (Latin, Nigeria)
he-IL	Hebrew (Israel)
hi-IN	Hindi (India)
hr-HR	Croatian (Croatia)
hu-HU	Hungarian (Hungary)
hy-AM	Armenian (Armenia)
id-ID	Indonesian (Indonesia)
ig-NG	Igbo (Nigeria)
is-IS	Icelandic (Iceland)
it-CH	Italian (Switzerland)
it-IT	Italian (Italy)
iu-Latn-CA	Inuktitut (Latin, Canada)
ja-JP	Japanese (Japan)
ka-GE	Georgian (Georgia)
kk-KZ	Kazakh (Kazakhstan)
km-KH	Khmer (Cambodia)
kn-IN	Kannada (India)
kok-IN	Konkani (India)
ko-KR	Korean (Korea)
ku-arab-IQ	Central Kurdish
ky-KG	Kyrgyz (Kyrgyzstan)
lb-LU	Luxembourgish (Luxembourg)
lo-LA	Lao (Lao P.D.R.)
lt-LT	Lithuanian (Lithuania)
lv-LV	Latvian (Latvia)
mi-NZ	Maori (New Zealand)
mk-MK	Macedonian (Former Yugoslav Republic of Macedonia)
ml-IN	Malayalam (India)
mn-MN	Mongolian (Cyrillic, Mongolia)
mr-IN	Marathi (India)
ms-BN	Malay (Brunei Darussalam)
ms-MY	Malay (Malaysia)
mt-MT	Maltese (Malta)
nb-NO	Norwegian, Bokmål (Norway)
ne-NP	Nepali (Nepal)
nl-BE	Dutch (Belgium)
nl-NL	Dutch (Netherlands)
nn-NO	Norwegian, Nynorsk (Norway)
nso-ZA	Sesotho sa Leboa (South Africa)
or-IN	Oriya (India)

pa-arab-PK	Punjabi (Pakistan)
pa-IN	Punjabi (India)
pl-PL	Polish (Poland)
prs-AF	Dari (Afghanistan)
ps-AF	Pashto (Afghanistan)
pt-BR	Portuguese (Brazil)
pt-PT	Portuguese (Portugal)
qut-GT	K'iche' (Guatemala)
quz-PE	Quechua (Peru)
rm-CH	Romansh (Switzerland)
ro-RO	Romanian (Romania)
ru-RU	Russian (Russia)
rw-RW	Kinyarwanda (Rwanda)
sd-arab-pk	Sindhi (Pakistan)
si-LK	Sinhala (Sri Lanka)
sk-SK	Slovak (Slovakia)
sl-SI	Slovenian (Slovenia)
sq-AL	Albanian (Albania)
sr-Cyrl-BA	Serbian (Cyrillic, Bosnia and Herzegovina)
sr-Cyrl-rs	Serbian (Cyrillic, Serbia)
sr-Latn-rs	Serbian (Latin, Serbia)
sv-SE	Swedish (Sweden)
sw-KE	Kiswahili (Kenya)
ta-IN	Tamil (India)
te-IN	Telugu (India)
tg-Cyrl-TJ	Tajik (Cyrillic, Tajikistan)
th-TH	Thai (Thailand)
Ti-ET	Tigrinya (Ethiopia)
tk-TM	Turkmen (Turkmenistan)
tn-ZA	Setswana (South Africa)
tr-TR	Turkish (Turkey)
tt-RU	Tatar (Russia)
ug-CN	Uighur (PRC)
uk-UA	Ukrainian (Ukraine)
ur-PK	Urdu (Islamic Republic of Pakistan)
uz-Cyrl-UZ	Uzbek (Cyrillic, Uzbekistan)
uz-Latn-UZ	Uzbek (Latin, Uzbekistan)
vi-VN	Vietnamese (Vietnam)
wo-SN	Wolof (Senegal)
xh-ZA	isiXhosa (South Africa)
yo-NG	Yoruba (Nigeria)
zh-CN	Chinese Simplified (People's Republic of China)
zh-HK	Chinese (Hong Kong S.A.R.)
zh-TW	Chinese Traditional (Taiwan)
zu-ZA	isiZulu (South Africa)

Version History

Description	Last updated
Terminology Service API, original SDK document	August 2013
AnyWord operator added to list of SearchOperator enum values	March 2014
Version 2.0 with "any to any" translation service and the sensitivity operator added	June 2014
Added endpoint references and help images for HTTPS endpoint	August 2021

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for informational purposes only. Microsoft makes no warranties, express or implied, in this document.