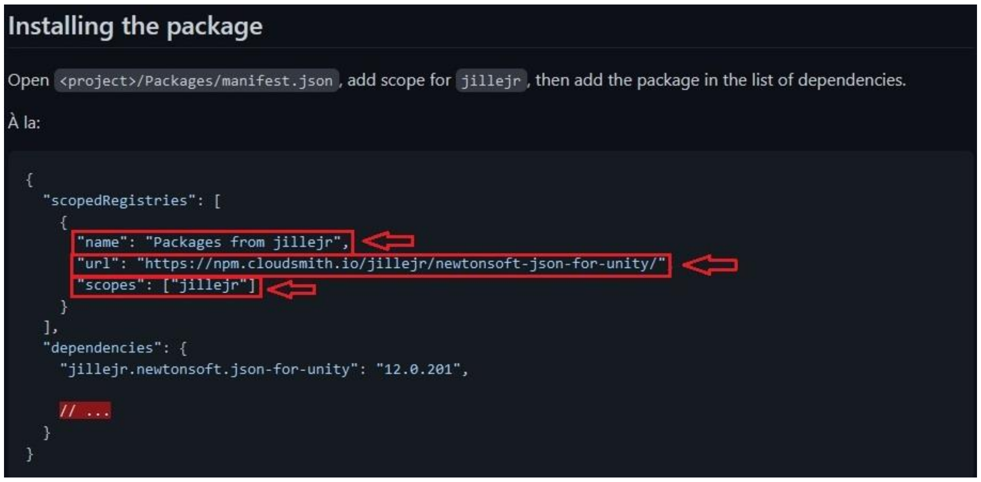
Uni-Gram

Overview

Uni-Gram is a package that was created to connect Unity to Telegram Messenger. You can send Message, Photo, Sticker, Audio, Voice, Video, Document, Animation, two kinds of Poll, Contact, Location & Venue with just two values, Telegram Token and Chat ID. In this version, the features are available to just Send, not Receive. But maybe Receive feature will be added in the future. We Simplified Telegram. Bot source code and migrated to Unity 3D. Uni-Gram tested on Android and PC, Mac & Linux Standalone, other platforms are not tested.

Installation

1- Import Uni-Gram package (You may face some problems. We will fix them in the next steps)
2- Remove Version Control package from Project (Unfortunately they have conflicts. to use Uni-Gram, you must remove it)
3- Install Json.NET v11.0.2 (Important)
To do that you must go to Edit -> Project Settings -> Package Manager
Then create a new Scope and fill the fields with data you got on this site



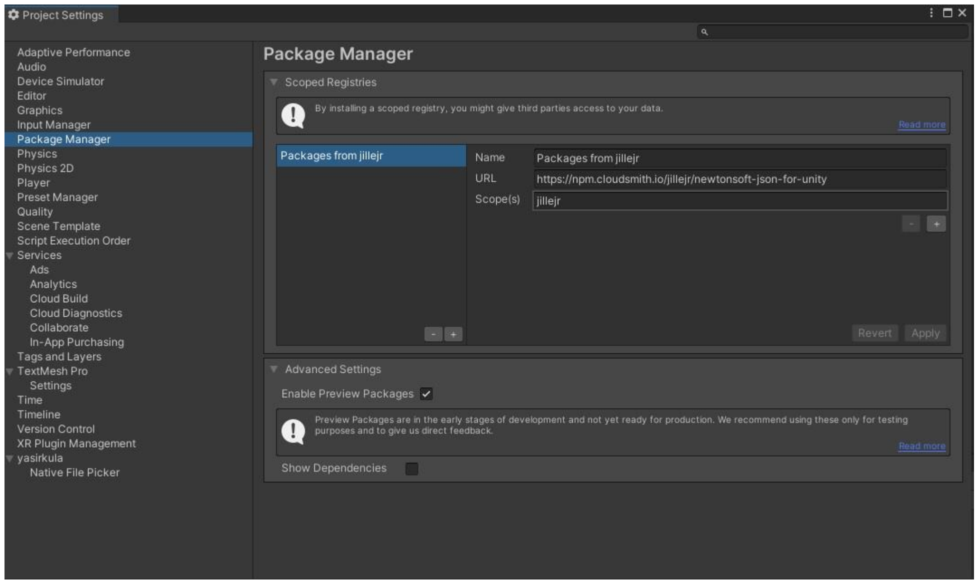
Installing the package

Open `<project>/Packages/manifest.json`, add scope for `jillejr`, then add the package in the list of dependencies.

À la:

```
{
  "scopedRegistries": [
    {
      "name": "Packages from jillejr",
      "url": "https://npm.cloudsmith.io/jillejr/newtonsoft-json-for-unity/",
      "scopes": ["jillejr"]
    }
  ],
  "dependencies": {
    "jillejr.newtonsoft.json-for-unity": "12.0.201",

    // ...
  }
}
```



By installing a scoped registry, you might give third parties access to your data.

Packages from jillejr

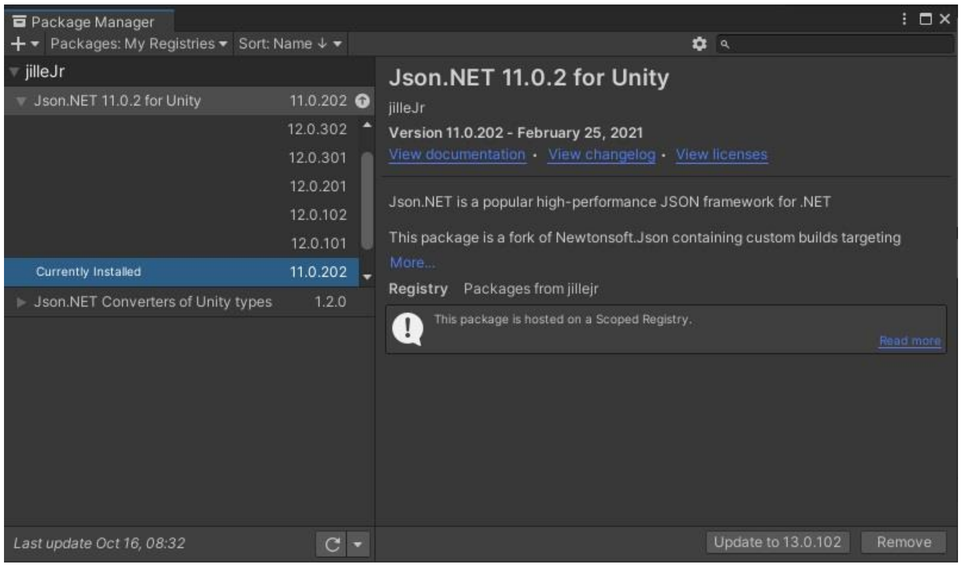
Name: Packages from jillejr
URL: https://npm.cloudsmith.io/jillejr/newtonsoft-json-for-unity
Scope(s): jillejr

Enable Preview Packages ✓

Preview Packages are in the early stages of development and not yet ready for production. We recommend using these only for testing purposes and to give us direct feedback.

Show Dependencies ☐

Then go to Window -> Package Manager and set Packages to My Registries.
On Json.NET, click on the arrow and click on "See other versions". Then select v11 and install it.



jilleJr

Json.NET 11.0.2 for Unity	11.0.202
	12.0.302
	12.0.301
	12.0.201
	12.0.102
	12.0.101
Currently Installed	11.0.202
Json.NET Converters of Unity types	1.2.0

Json.NET 11.0.2 for Unity

jilleJr

Version 11.0.202 - February 25, 2021

View documentation · View changelog · View licenses

Json.NET is a popular high-performance JSON framework for .NET

This package is a fork of Newtonsoft.Json containing custom builds targeting

More...

Registry Packages from jillejr

This package is hosted on a Scoped Registry. Read more

Last update Oct 16, 08:32

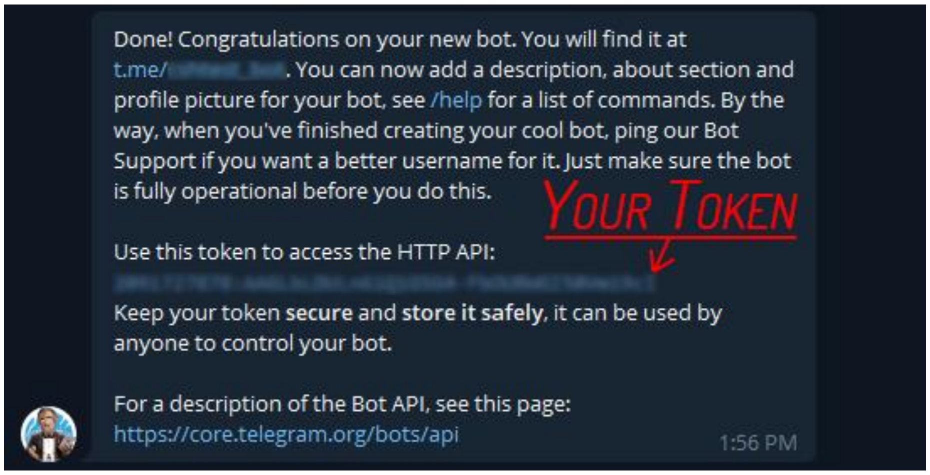
(Do not update it to the last version)

4- Package installed successfully!

Setup a Bot

Telegram Token

1- Open the Telegram and search for @BotFather then start it
2- Click on /newbot command and create a new bot by following the instructions
3- When the creating is done, BotFather will give you the Token


Done! Congratulations on your new bot. You will find it at t.me/[redacted]. You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

Use this token to access the HTTP API:
[redacted]
Keep your token **secure** and **store it safely**, it can be used by anyone to control your bot.

For a description of the Bot API, see this page:
https://core.telegram.org/bots/api

1:56 PM

Chat ID

1- Find @RawDataBot and start it normally or add it into your specific group then start it.
2- It will send you a message that contains some information. Your needed information is in **chat -> id**

Methods

Initialize (telegramToken, ChatID):

- Run it in the Start method to initialize the connection.
- Assign your Telegram Token to the first input and Chat ID to the second input.

- Example:

```
using UnityEngine;

public class ExampleCodes : MonoBehaviour
{
    public string telegramToken = "token";
    public long chatID = 123456789;

    private UniGram uniGram = new UniGram();

    private void Start()
    {
        uniGram.Initialize(telegramToken, chatID);
    }
}
```

SendMessage (text, parseMode):

- Use this method to send text messages. On success, the sent Message is returned.

- "text": Text of the message to be sent, 1-4096 characters after entities parsing.
- "parseMode": Mode for parsing entities in the message text.

- Example:

```
using System.Collections;
using System.Threading.Tasks;
using Telegram.Bot.Types;
using UnityEngine;
using Telegram.Bot.Types.Enums;

public class ExampleCodes : MonoBehaviour
{
    private UniGram uniGram = new UniGram();

    public void SubmitMessage()
    {
        StartCoroutine(SendMessage());
    }

    private IEnumerator SendMessage()
    {
        Task<Message> task = uniGram.SendMessage("example", ParseMode.Html);
        yield return new WaitUntil(() => task.IsCompleted);
        Debug.Log("Message sent!");
    }
}
```

SendPhoto (path, local, caption):

- Use this method to send photos. On success, the sent Message is returned.

- "path": Photo to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. The width and height ratio must be at most 20.
- "local": Is it on your local system or not.
- "caption": Photo caption, 0-1024 characters after entities parsing.

- Example:

```
using System.Collections;
using System.Threading.Tasks;
using Telegram.Bot.Types;
using UnityEngine;

public class ExampleCodes : MonoBehaviour
{
    private UniGram uniGram = new UniGram();

    public void SubmitPhoto()
    {
        StartCoroutine(SendPhoto());
    }

    private IEnumerator SendPhoto()
    {
        Task<Message> task = uniGram.SendPhoto(@"C:\Users\[YourUser]\Desktop\image.jpg",
true, "example");
        yield return new WaitUntil(() => task.IsCompleted);
        Debug.Log("Photo sent!");
    }
}
```

SendSticker (path, local):

- Use this method to send static .WEBP or animated .TGS stickers. On success, the sent Message is returned.

- "path": Sticker to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a .WEBP file from the Internet, or upload a new one using multipart/form-data.
- "local": Is it on your local system or not.

- Example:

```
using System.Collections;
using System.Threading.Tasks;
using Telegram.Bot.Types;
using UnityEngine;

public class ExampleCodes : MonoBehaviour
{
    private UniGram uniGram = new UniGram();

    public void SubmitPhoto()
    {
        StartCoroutine(SendSticker());
    }

    private IEnumerator SendSticker()
    {
        Task<Message> task =
uniGram.SendSticker(@"C:\Users\[YourUser]\Desktop\sticker.webp", true);
        yield return new WaitUntil(() => task.IsCompleted);
        Debug.Log("Sticker sent!");
    }
}
```

SendAudio (path, local, caption):

- Use this method to send audio files, if you want Telegram clients to display them in the music player. On success, the sent Message is returned. Your audio must be in .MP3 or .M4A format. Bots can currently send audio files of up to 50 MB in size.

- "path": Audio file to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an audio file from the Internet, or upload a new one using multipart/form-data.
- "local": Is it on your local system or not.
- "caption": Audio caption, 0-1024 characters after entities parsing.

- Example:

```
using System.Collections;
using System.Threading.Tasks;
using Telegram.Bot.Types;
using UnityEngine;

public class ExampleCodes : MonoBehaviour
{
    private UniGram uniGram = new UniGram();

    public void SubmitPhoto()
    {
        StartCoroutine(SendAudio());
    }

    private IEnumerator SendAudio()
    {
        Task<Message> task = uniGram.SendAudio(@"C:\Users\[YourUser]\Desktop\audio.mp3",
true,"example");
        yield return new WaitUntil(() => task.IsCompleted);
        Debug.Log("Audio sent!");
    }
}
```

SendVoice (path, local, caption):

- Use this method to send audio files, if you want Telegram clients to display the file as a playable voice message. For this to work, your audio must be in an .OGG file encoded with OPUS (other formats may be sent as Audio or Document). On success, the sent Message is returned. Bots can currently send voice messages of up to 50 MB in size.

- "path": Audio file to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data.
- "local": Is it on your local system or not.
- "caption": Voice message caption, 0-1024 characters after entities parsing.

- Example:

```
using System.Collections;
using System.Threading.Tasks;
using Telegram.Bot.Types;
using UnityEngine;

public class ExampleCodes : MonoBehaviour
{
    private UniGram uniGram = new UniGram();

    public void SubmitVoice()
    {
        StartCoroutine(SendVoice());
    }

    private IEnumerator SendVoice()
    {
        Task<Message> task = uniGram.SendVoice(@"C:\Users\[YourUser]\Desktop\voice.ogg",
true, "example");
        yield return new WaitUntil(() => task.IsCompleted);
        Debug.Log("Voice sent!");
    }
}
```

SendVideo (path, local, supportsStreaming, caption):

- Use this method to send video files, Telegram clients support mp4 videos (other formats may be sent as Document). On success, the sent Message is returned. Bots can currently send video files of up to 50 MB in size.

- "path": Video to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a video from the Internet, or upload a new video using multipart/form-data.
- "local": Is it on your local system or not.
- "caption": Video caption, 0-1024 characters after entities parsing.
- "supportsStreaming": Pass True, if the uploaded video is suitable for streaming.

- Example:

```
using System.Collections;
using System.Threading.Tasks;
using Telegram.Bot.Types;
using UnityEngine;

public class ExampleCodes : MonoBehaviour
{
    private UniGram uniGram = new UniGram();

    public void SubmitVideo()
    {
        StartCoroutine(SendVideo());
    }

    private IEnumerator SendVideo()
    {
        Task<Message> task = uniGram.SendVideo(@"C:\Users\[YourUser]\Desktop\video.mp4",
true, true, "example");
        yield return new WaitUntil(() => task.IsCompleted);
        Debug.Log("Video sent!");
    }
}
```

SendDocument (path, local, caption):

- Use this method to send general files. On success, the sent Message is returned. Bots can currently send files of any type of up to 50 MB in size.

- “path”: File to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data.
- “local”: Is it on your local system or not.
- “caption”: Document caption, 0-1024 characters after entities parsing.

- Example:

```
using System.Collections;
using System.Threading.Tasks;
using Telegram.Bot.Types;
using UnityEngine;

public class ExampleCodes : MonoBehaviour
{
    private UniGram uniGram = new UniGram();

    public void SubmitDocument()
    {
        StartCoroutine(SendDocument());
    }

    private IEnumerator SendDocument()
    {
        Task<Message> task = 
uniGram.SendDocument(@"C:\Users\[YourUser]\Desktop\document.rar", true, "example");
        yield return new WaitUntil(() => task.IsCompleted);
        Debug.Log("Document sent!");
    }
}
```

SendAnimation (path, local, caption):

- Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound). On success, the sent Message is returned. Bots can currently send animation files of up to 50 MB in size.

- "path": Animation to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an animation from the Internet, or upload a new animation using multipart/form-data.
- "local": Is it on your local system or not.
- "caption": Animation caption (may also be used when resending animation by file_id), 0-1024 characters after entities parsing.

- Example:

```
using System.Collections;
using System.Threading.Tasks;
using Telegram.Bot.Types;
using UnityEngine;

public class ExampleCodes : MonoBehaviour
{
    private UniGram uniGram = new UniGram();

    public void SubmitAnimation()
    {
        StartCoroutine(SendAnimation());
    }

    private IEnumerator SendAnimation()
    {
        Task<Message> task = 
uniGram.SendAnimation(@"C:\Users\[YourUser]\Desktop\animation.gif", true, "example");
        yield return new WaitUntil(() => task.IsCompleted);
        Debug.Log("Animation sent!");
    }
}
```

SendRegularPoll (question, options, isAnonymous, allowMultipleAnswers):

- Use this method to send a regular poll. On success, the sent Message is returned.

- "question": Poll question, 1-300 characters.
- "options": A JSON-serialized list of answer options, 2-10 strings 1-100 characters each.
- "isAnonymous": True, if the poll needs to be anonymous, defaults to True.
- "allowMultipleAnswers": True, if the poll allows multiple answers, ignored for polls in quiz mode, defaults to False.

- Example:

```
using System.Collections;
using System.Threading.Tasks;
using Telegram.Bot.Types;
using UnityEngine;

public class ExampleCodes : MonoBehaviour
{
    private UniGram uniGram = new UniGram();

    public void SubmitPoll()
    {
        StartCoroutine(SendRegularPoll());
    }

    private IEnumerator SendRegularPoll()
    {
        Task<Message> task = uniGram.SendRegularPoll("What's your favorite sports?",
new[] { "Soccer", "Tennis", "Basketball", "Golf" }, true, false);
        yield return new WaitUntil(() => task.IsCompleted);
        Debug.Log("Poll sent!");
    }
}
```

SendQuizPoll (question, options, isAnonymous, allowMultipleAnswers, correctOptionId, explanation, explanationParseMode):

- Use this method to send a quiz poll. On success, the sent Message is returned.

- "question": Poll question, 1-300 characters.
- "options": A JSON-serialized list of answer options, 2-10 strings 1-100 characters each.
- "isAnonymous": True, if the poll needs to be anonymous, defaults to True.
- "correctOptionId": 0-based identifier of the correct answer option, required for polls in quiz mode
- "Explanation": Text that is shown when a user chooses an incorrect answer or taps on the lamp icon in a quiz-style poll, 0-200 characters with at most two line feeds after entities parsing.
- "explanationParseMode": Mode for parsing entities in the explanation.

- Example:

```
using System.Collections;
using System.Threading.Tasks;
using Telegram.Bot.Types;
using UnityEngine;
using Telegram.Bot.Types.Enums;

public class ExampleCodes : MonoBehaviour
{
    private UniGram uniGram = new UniGram();

    public void SubmitPoll()
    {
        StartCoroutine(SendQuizPoll());
    }

    private IEnumerator SendQuizPoll()
    {
        Task<Message> task = uniGram.SendQuizPoll("1 + 1?", new[] { "2", "1", "0", "-1"
}, false, 0, "<i>1 + 1 = 2</i>", ParseMode.Html);
        yield return new WaitUntil(() => task.IsCompleted);
        Debug.Log("Poll sent!");
    }
}
```

StopPoll (messageID):

- Use this method to stop a poll that was sent by the bot. On success, the stopped Poll is returned.
- "messageID": Identifier of the original message with the poll.

- Example:

```
using System.Collections;
using System.Threading.Tasks;
using Telegram.Bot.Types;
using UnityEngine;
using Telegram.Bot.Types.Enums;

public class ExampleCodes : MonoBehaviour
{
    private int pollMessageID;
    private UniGram uniGram = new UniGram();

    public void SubmitPoll()
    {
        StartCoroutine(SendQuizPoll());
    }

    private IEnumerator SendQuizPoll()
    {
        Task<Message> task = uniGram.SendQuizPoll("1 + 1?", new[] { "2", "1", "0", "-1"
}, false, 0, "<i>1 + 1 = 2</i>", ParseMode.Html);
        pollMessageID = task.Result.MessageId;

        yield return new WaitUntil(() => task.IsCompleted);
        Debug.Log("Poll sent!");
    }

    public void StopPoll()
    {
        StartCoroutine(StopQuizPoll());
    }

    private IEnumerator StopQuizPoll()
    {
        Task<Poll> task = uniGram.StopPoll(pollMessageID);

        yield return new WaitUntil(() => task.IsCompleted);
        Debug.Log("Poll stopped!");
    }
}
```

SendContact (phoneNumber, firstName, lastName):

- Use this method to send phone contacts. On success, the sent Message is returned.

- "phoneNumber": Contact's phone number.
- "firstName": Contact's first name.
- "lastName": Contact's last name

- Example:

```
using System.Collections;
using System.Threading.Tasks;
using Telegram.Bot.Types;
using UnityEngine;

public class ExampleCodes : MonoBehaviour
{
    private UniGram uniGram = new UniGram();

    public void SubmitContact()
    {
        StartCoroutine(SendContact());
    }

    private IEnumerator SendContact()
    {
        Task<Message> task = uniGram.SendContact("+15552156980", "John", "Black");
        yield return new WaitUntil(() => task.IsCompleted);
        Debug.Log("Contact sent!");
    }
}
```

SendLocation (latitude, longitude):

- Use this method to send points on the map. On success, the sent Message is returned.

- “latitude”: Latitude of the location.
- “longitude”: Longitude of the location.

- Example:

```
using System.Collections;
using System.Threading.Tasks;
using Telegram.Bot.Types;
using UnityEngine;

public class ExampleCodes : MonoBehaviour
{
    private UniGram uniGram = new UniGram();

    public void SubmitLocation()
    {
        StartCoroutine(SendLocation());
    }

    private IEnumerator SendLocation()
    {
        Task<Message> task = uniGram.SendLocation(34.547711f, 50.802546f);
        yield return new WaitUntil(() => task.IsCompleted);
        Debug.Log("Location sent!");
    }
}
```

SendVenue (latitude, longitude, title, address):

- Use this method to send information about a venue. On success, the sent Message is returned.

- “latitude”: Latitude of the venue
- “longitude”: Longitude of the venue
- “title”: Name of the venue
- “address”: Address of the venue

- Example:

```
using System.Collections;
using System.Threading.Tasks;
using Telegram.Bot.Types;
using UnityEngine;

public class ExampleCodes : MonoBehaviour
{
    private UniGram uniGram = new UniGram();

    public void SubmitVenue()
    {
        StartCoroutine(SendVenue());
    }

    private IEnumerator SendVenue()
    {
        Task<Message> task = uniGram.SendVenue(34.547711f, 50.802546f, "example title",
"example address");
        yield return new WaitUntil(() => task.IsCompleted);
        Debug.Log("Venue sent!");
    }
}
```

Demos

Import TextMeshPro before using Demos.
@unitytelegram_bot Token has been assigned for demos. Assign your Chat ID and test it.