Open Source Project Author:
JusterZhu

Document author: JusterZhu
(Zhu Zhen)

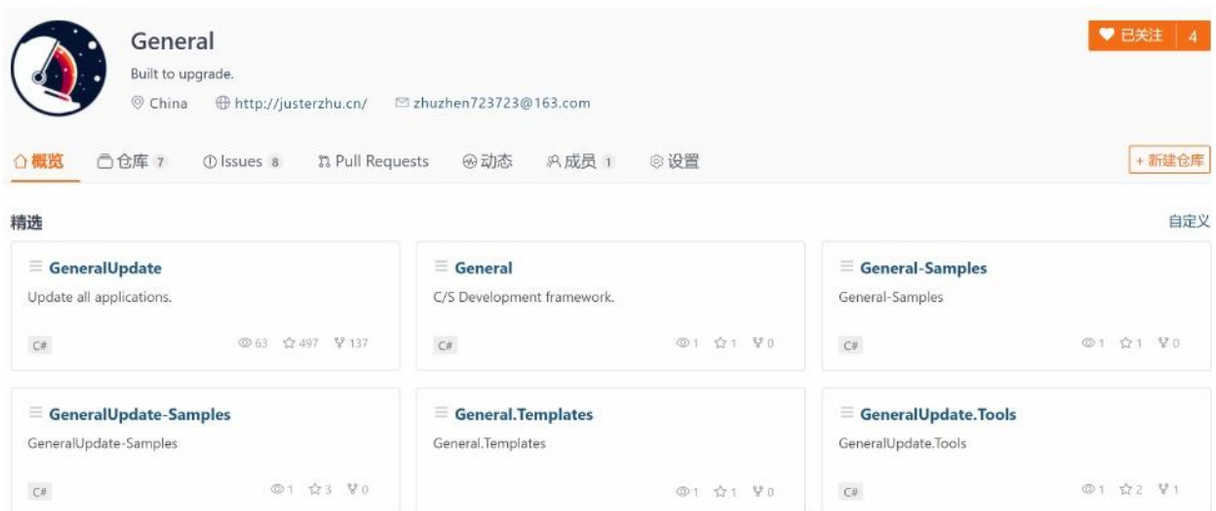
 Document date: 2023.4.22

1. Component Description

1.1 What is GeneralUpdate?

GeneralUpdate is an application automatic update component.

Open source address



- https://github.com/orgs/GeneralLibrary
- https://gitee.com/GeneralLibrary

The original github address has been abandoned to update, please developers move to the above open source warehouse address to re-star. https://github.com/WELL-E/AutoUpdater

GeneralUpdate.Tools The project address of the packaging tool

- [redacted]
- [redacted]

Sample item address GeneralUpdate-Samples

https://github.com/GeneralLibrary/GeneralUpdate-Samples
https://gitee.com/GeneralTeam/GeneralUpdate-Samples

Help documents

Official documentation:
http://justerzhu.cn/

Use the tutorial video:
https://www.bilibili.com/video/BV1FT4y1Y7hV

Development environment setup

Install the.net core SDK the latest edition of the
https://dotnet.microsoft.com/en-us/download

Install Visual Studio 2022 the latest edition of the
https://visualstudio.microsoft.com/zh-hans/vs/getting-started/ (Visual Studio 2022 Preview is best)

Get the code for this project https://gitee.com/Juster-zhu/GeneralUpdate

Clone code

```
//Gitee

git clone https://gitee.com/Juster-zhu/GeneralUpdate.git

//Github

git clone https://github.com/JusterZhu/GeneralUpdate.git
```

Branch instructions

master Stable Branch

dev Dev Function

branch contribute

1. Fork this project
2. Create a new branch of Feat_xxx
3. Submit the code
4. Create a Pull Request

communication

Free questions: https://github.com/JusterZhu/GeneralUpdate/issues

free technical communication: 341349660

Open source Project Usage
Discussion: 748744489

Wechat
public
number:

Business cooperation

Customized second development, technical consulting, open source project rewards. Paid consultation is required for one-to-one questions and answers about open source projects.

This open source project is currently open source under the MIT Open Source License and is free for commercial use. Questions from developers will be answered from time to time in the free communication channel. All community donations will be used for the development of the open source project. Code contributors will be rewarded with donated funds. Email: zhuzhen723723@outlook.com

A little star is your greatest support to me, so that I can keep the motivation of updating and maintenance.

Unifying Language (s)

There are some basic concepts in the GeneralUpdate that need to be understood before we begin to use the GeneralUpdate.

Client: Refers to your main application, which is the client that is updated. It can also be understood as the product of the company.

Upgrade: indicates the upgrade program, which will be a separate process. It must be stored in the same level directory as the Client. It cannot be associated with any business or design during use (or coding), and must be referenced independently.

Server: indicates that the server-side application (ASP.NET) will provide version update information interaction and version verification functions.

GeneralUpdate.Tool: GeneralUpdate.tool is a packaging tool provided by this open source project, which is used to generate update packages (7z and zip).

UpdatePacket: As the name implies, updatepacket is an update package. The update package contains patches (.pacth) or new files that have file content differences between the old and new versions.

SQL: Currently, sql scripts are generated based on the Mysql database.

Version number execution standard

Nuget version management reference standard: https://docs.microsoft.com/zh-cn/nuget/concepts/package-versioning application version management reference standard: https://docs.microsoft.com/zh-cn/dotnet/standard/assembly/versi oning (updated by the components of the client program, said popular point is your company's products; The operation of the component will be performed according to this standard. In addition to updating the version number in AssemblyInfo, each iteration will also need to change the version number of the DLL file.

What does GeneralUpdate provide?

Component Features

Features	Yes No support	Remarks
Breakpoint continuation	Holdup	Passive effective

Version-by-version update	Support	Passive effective
Binary differential update	Support	Passive effective
Incremental update function	Holdup	Passive effective
Keep the profile updated	the a	Currently refers to support depth of 1 json configuration file temporarily offline this function (passive effect)

Feat	is no the	rema
Force updates	a the a	Optional update can pop up a selection box for users to choose, mandatory update directly
Version verification, update credit support		when a product has multiple branches, you need to update the corresponding content according to different branches
ch the latest version	Pu	
Client program, service branch		. Verify whether the version needs to be updated
Mutual	Su pp	
	Th e mi ni st ry of	Linux, MacCatalyst, Windows.
Multil	Di ch the	You can also write this component as a console program as an update "script". For applications in other languages to call updates.
Blackl ist	the a th	Files and file extensions in the blacklist are skipped during the update process.
		Extremely simplified update, only need to place the version

Packaging tools

GeneralUpdate.PacketTool to use. A desktop side packaging program written by NET MAUI (.NET 6).

Featur es	Support	R e m a r k s
Differential update package generation	Support	Compare the previous version with the current version to find files that need to be updated or new files added.
Upload updates	Support	Automatically upload the generated differential update package to the

automatically		server.
Multi-platform support	Sections	Windows, Linux, MacCatalyst only.

What does GeneralUpdate support?

.NET Framework

Frame	Supporte
.NET Core 2.0	Supp
.NET 5 6 7 8	supp
.NET Framework 4.6.1	Supp

UI Framework

UI frame name	Supported or not
WPF	Support
UWP	Not updatable in Store mode (lindexi provides feedback)
MAUI	Support (currently windows only)
Avalonia	Support
WinUI	To be verified, waiting for feedback
Console	Support
Winform	Support

Server-side framework

Server-side framework	Supported or not
ASP.NET	Support
Minimal Api	Support

Operating system

Operating	Supporte
Windows	Supp
Linux	Supp
MacCatalyst	Supp
iOS	Not
Android	Not
Raspberry	To be

Operating	Supporte
Domestic operating systems	To be

Publicat
ion log

https://mp.weixin.qq.com/mp/appmsgalbum?__biz=MzI5MTg4NzIyNg==&action=getalbum&album_id=1785314218515218433&scene=173&from_msgid=2247483663&from_itemidx=1&count=3&nolastread=1#wechat_redirect

2.GeneralUpdate.AspNetCore

The server application (.NET Minimal Api) will provide version update information and version verification information to determine whether updates are required and the address contained in the update package. Provide Features Features

Function function	Yes No support	Remarks
Push-send	Holding	This function is based on Signal R to implement the latest version update push.
Newer	Support	Check whether it needs to be updated according to the information uploaded by the client, and return the information required for the update.
Upload	Brace	Used to upload update packages to GeneralUpdate.PacketTool. The implementation logic of storing the uploaded information into the repository and saving the compressed package to the file server needs to be implemented by the developer, which is reserved in the example structure.

Mysql Database
script

```
CREATE TABLE IF NOT EXISTS `updateversioninfo` (
  'MD5 'varchar(64) COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT 'update packet
md5'COMMENT
'MD5 code ',
  'PubTime' int(11) NOT NULL DEFAULT '0' COMMENT 'Version release
  time',
  'Name' varchar(64) COLLATE utf8mb4_unicode_ci DEFAULT 'version name' COMMENT
'Version
 name',
  'Url' varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT 'update
url'COMMENT
'Update packet file server address ',
  'Version' varchar(20) COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT 'last
versionnumber
' COMMENT 'Version number',
```

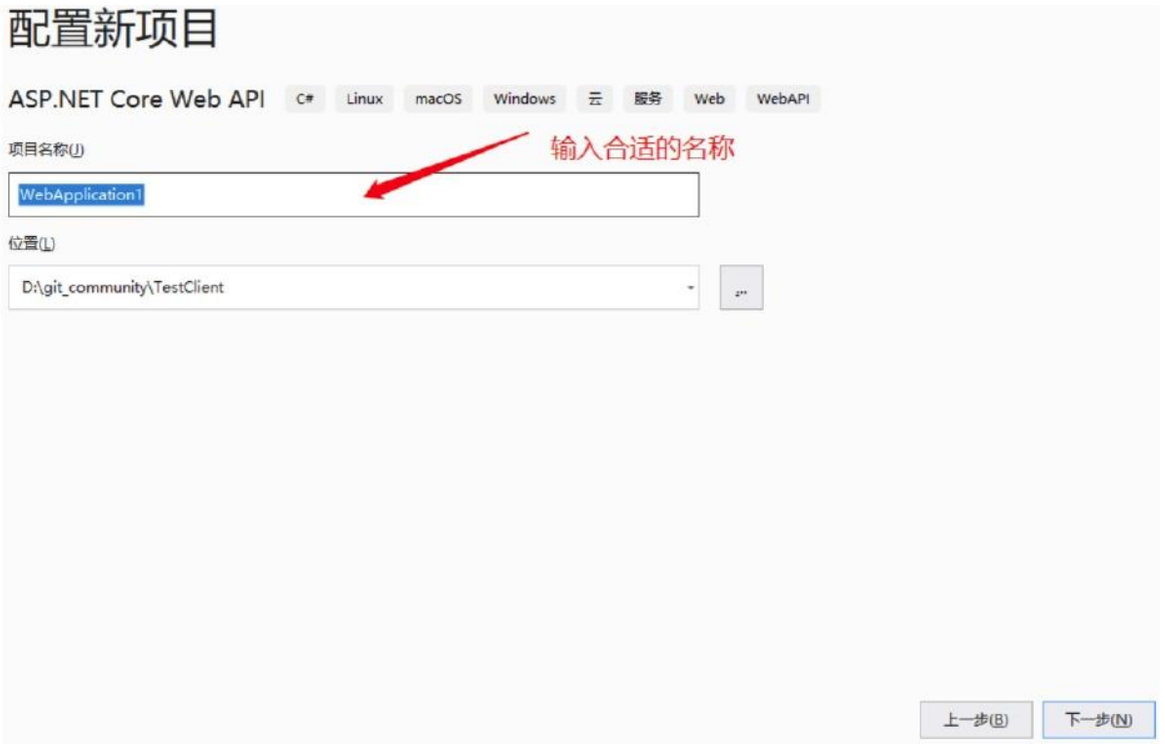
Example to use

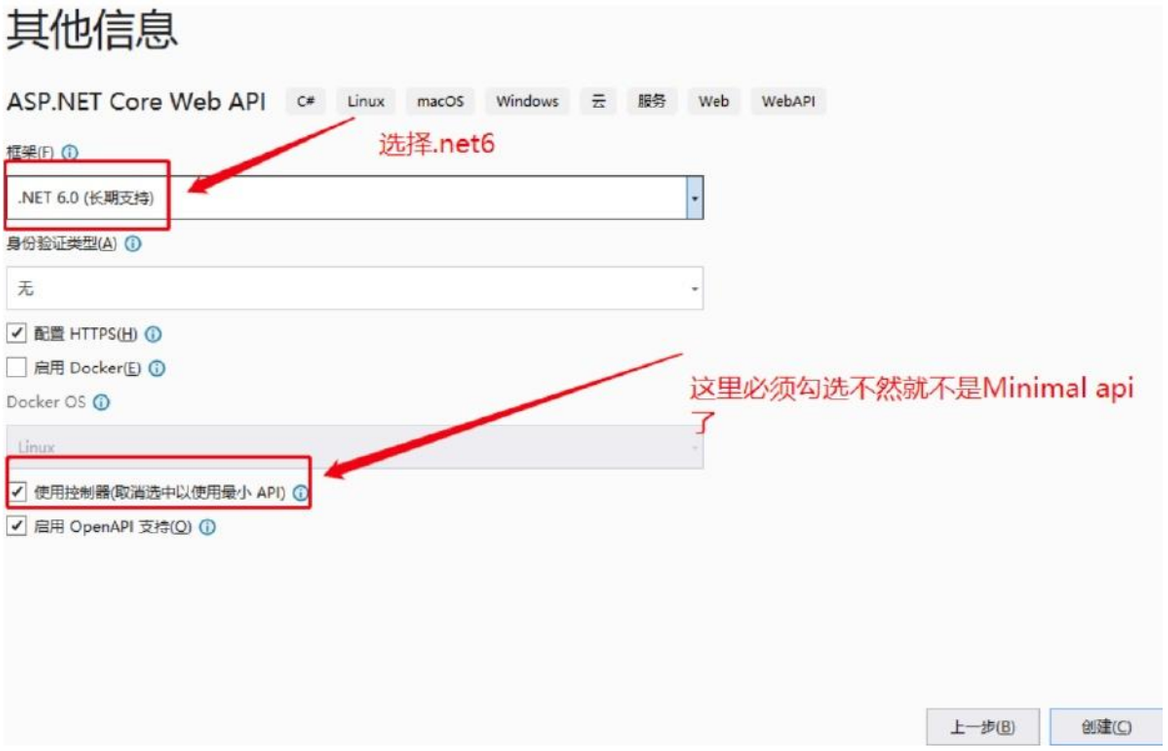
nuget Installation

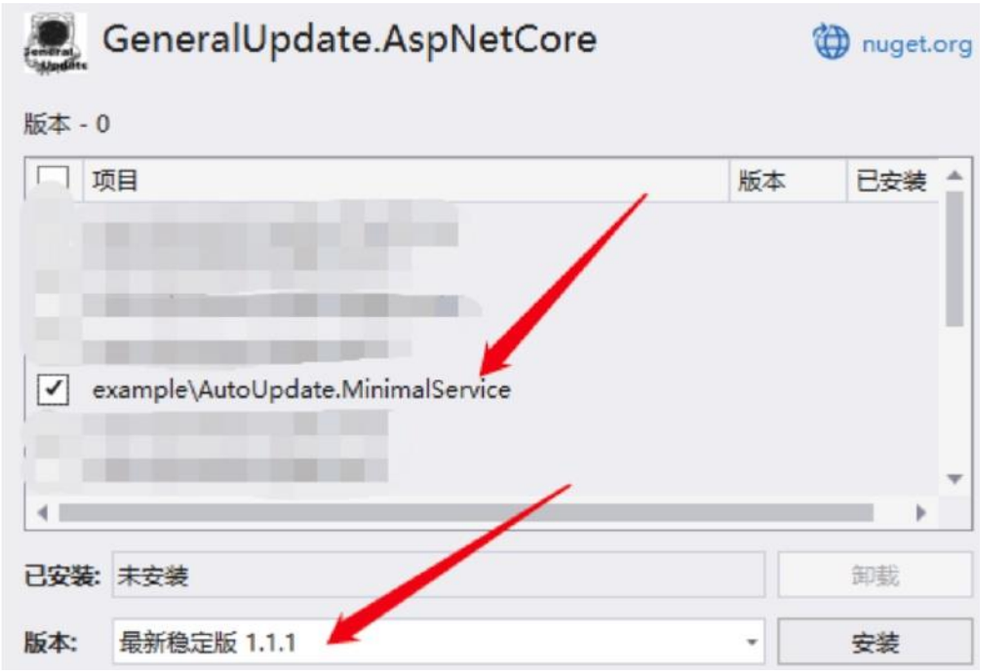
```
NuGet\Install-Package GeneralUpdate.AspNetCore-Version 1.1.1
```

The Minimal API is used as the service base.

After creation, the project structure is as follows:


配置新项目
ASP.NET Core Web API
C#
Linux
macOS
Windows
云
服务
Web
WebAPI
项目名称(J)
输入合适的名称
WebApplication1
位置(L)
D:\git_community\TestClient
...
上一步(B)
下一步(N)


其他信息
ASP.NET Core Web API
C#
Linux
macOS
Windows
云
服务
Web
WebAPI
选择.net6
框架(F)
.NET 6.0 (长期支持)
身份验证类型(A)
无
配置 HTTPS(H)
启用 Docker(E)
Docker OS
Linux
这里必须勾选不然就不是Minimal api了
使用控制器(取消选中以使用最小 API)
启用 OpenAPI 支持(O)
上一步(B)
创建(C)



Project structure

Search GeneralUpdate.AspNetCore installation in the nuget manager.

Sample code

The service side complete sample code: https://gitee.com/Juster-zhu/GeneralUpdate/blob/master/src/c%23/GeneralUpd ate. Api/Program. The cs

The Client complete sample code: https://gitee.com/Juster-zhu/GeneralUpdate/blob/master/src/c%23/GeneralUpd ate. Client/MainPage. Xaml. Cs

My SQL script: https://gitee.com/Juster-zhu/GeneralUpdate/blob/master/src/sql/update v2.0.sql

(1) Push function 1.1

Server-side

code

```
---
    builder.Services.AddSignalR();
```

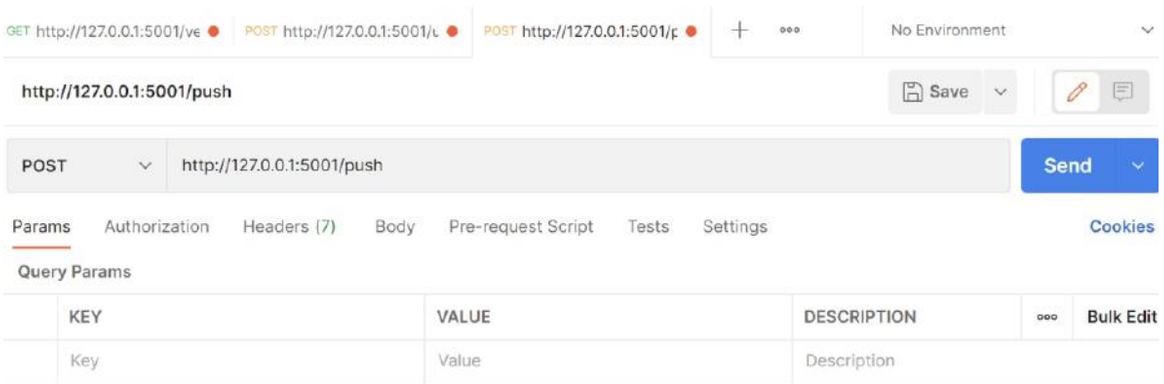
```
---

/ *
*

 * Push the latest version information in real time.
app.MapHub<VersionHub>("/versionhub");


app.MapPost("/push", async Task<string> (HttpContext context) =>
{
    try
    {
        var hubContext =
context.RequestServices.GetRequiredService<IHubContext<VersionHub>>();
        await hubContext.SendMessage
        ("TESTNAME", "123");
    }
    catch (Exception ex)
    {
        return ex.Message;
    }
    return "OK";
});
```

```
        private const string baseUrl = @"http://127.0.0.1:5001";
        private const string hubName = "versionhub";


        public MainPage()
        {
            InitializeComponent();
            MyButton.Clicked += OnClicked;
            Loaded += OnLoaded;
        }


        private void OnLoaded(object sender, EventArgs e)
        {
            VersionHub<string> Instance.Subscribe($"{baseUrl}/{hubName}"
"TESTNAME", new Action<string>(GetMessage));
        }


        private async void GetMessage(string msg)
        {
            var isUpdate = await Shell.Current.DisplayAlert("New Version",
"Thereare
 new version push messages !" , "update","cancel");
            if (isUpdate) Upgrade();
```

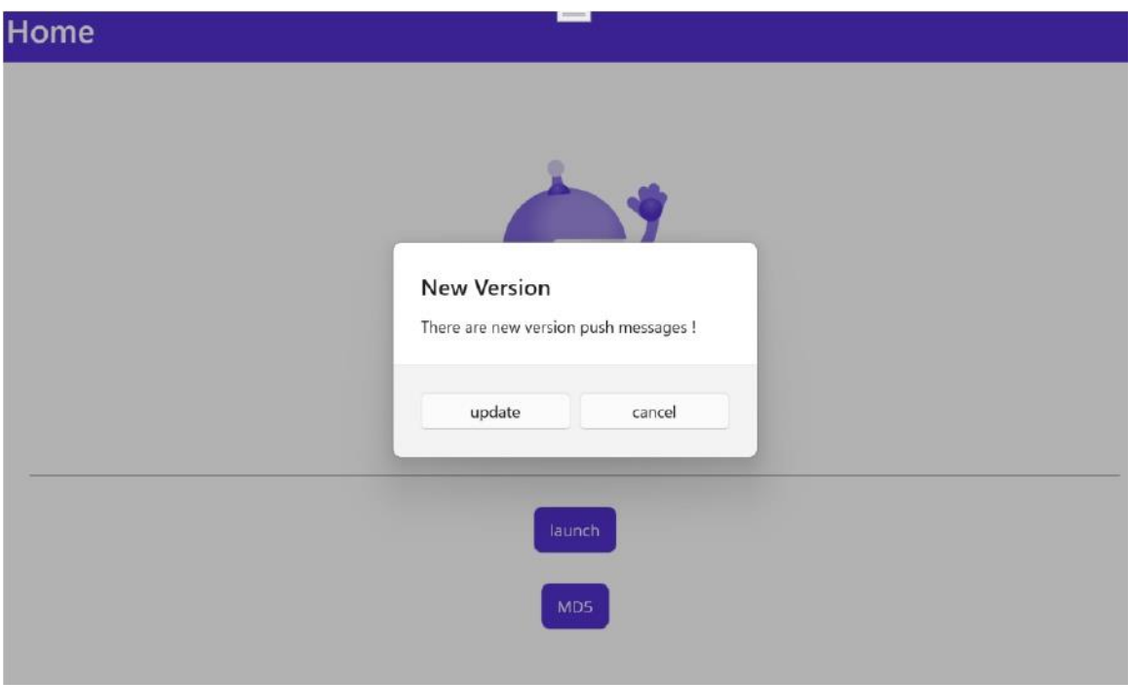
1.3 Launch the Minimal API

```
F:\git_project\GeneralUpdate\src\c#\GeneralUpdate.Api\bin\Debug\net6.0\GeneralUpdate.Api.exe
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://127.0.0.1:5001
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: F:\git_project\GeneralUpdate\src\c#\GeneralUpdate.Api\
```

1.4PostMan
Testing



1.5 Running
Effect



(2) Update features

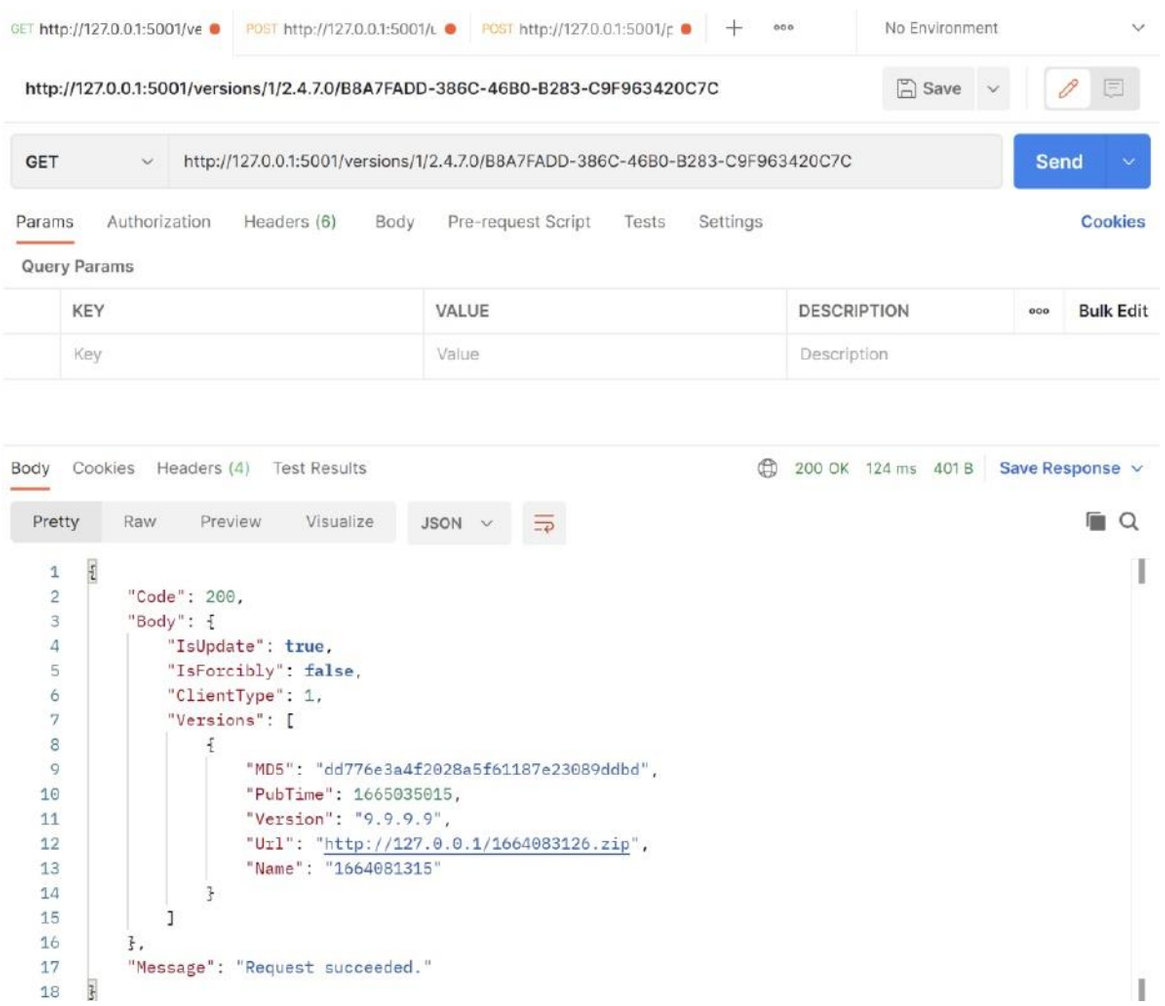
2.1 Server-side code

```
---
    builder.Services.AddSingleton<IUpdateService, GeneralUpdateService>();
---

/*
*
 * Check if an update is required.
 *
 */

{
    var versions = new List<VersionDTO>();
    var md5 = "dd776e3a4f2028a5f61187e23089ddbd";
    var pubTime = new
    DateTimeOffset(DateTime.UtcNow).ToUnixTimeSeconds();
    string version = null;
    {
        //client
        //version = "0.0.0.0";
        version = "9.9.9.9";
    }
    else if (clientType == AppType.UpgradeApp)
    {
        //upgrad
        //version = "0.0.0.0";
        version = "9.9.9.9";
    }
    Var = $url "http://127.0.0.1/1664083126.zip";
    var name = "1664081315";
    versions.Add(new VersionDTO(md5, pubTime, version, url, name));
    return updateService.Update(clientType, clientVersion, version, clientAppKey,
GetAppSecretKey(), false, versions);
});
```

Update definition:

```
        /// <summary>
        /// Verify whether the current version of the client needs to be updated.
        /// </summary>
        /// <param name="clientType">1:ClientApp 2:UpdateApp</param>
        /// <param name="clientVersion">Current version of the client</param>
        /// <param name="serverLastVersion">The latest version of the server.
</param>
        /// <param name="clientAppkey">The appkey agreed by the client and server.
</param>
        /// <param name="appSecretKey">Appkey is stored in the database.</param>
        /// <param name="isForce">Whether to force all versions to be updated.
</param>
        /// <param name="versions"></param>
        /// <returns>Json object.</returns>
        string Update(int clientType, string clientVersion, string
serverLastVersion, string clientAppkey,string appSecretKey, bool isForce,
List<VersionDTO> versions);
```

2.2PostMan
test

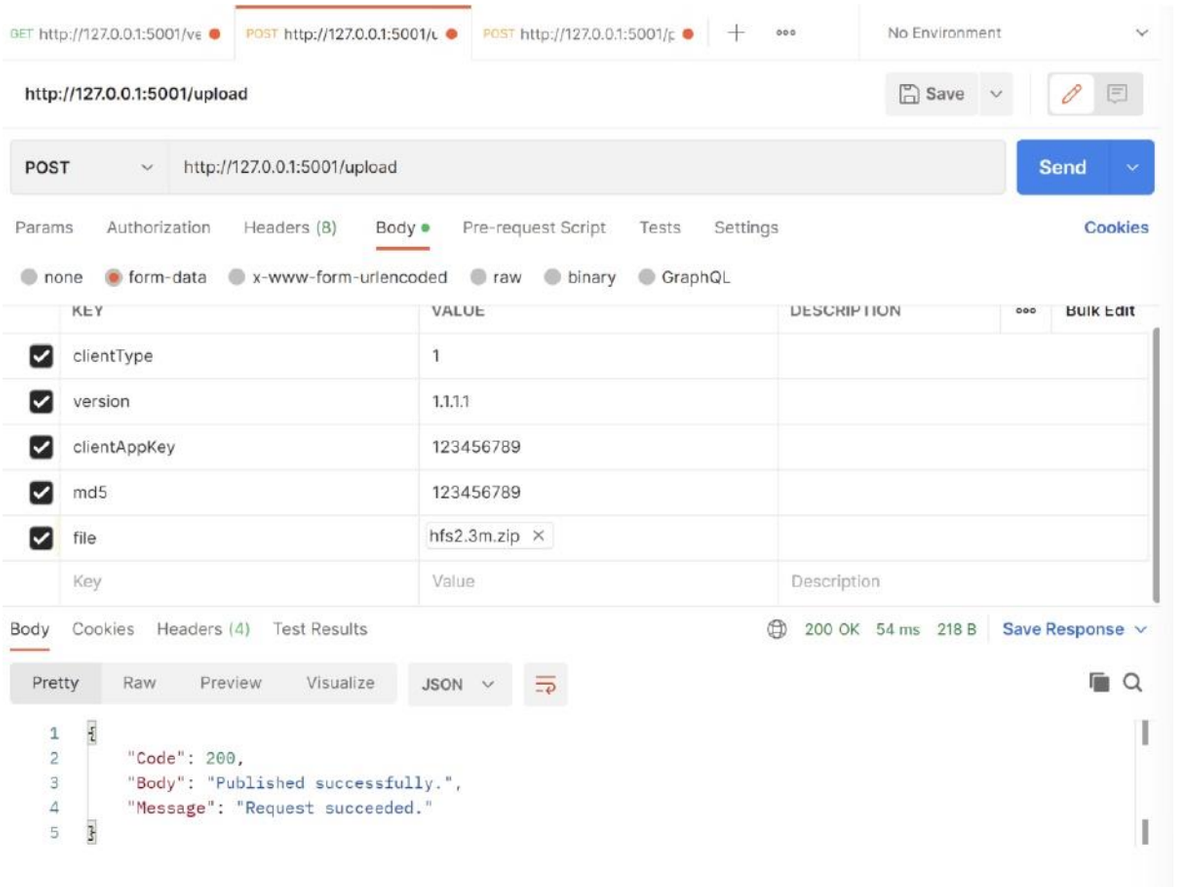


(3) Upload the update package

3.1 Server code

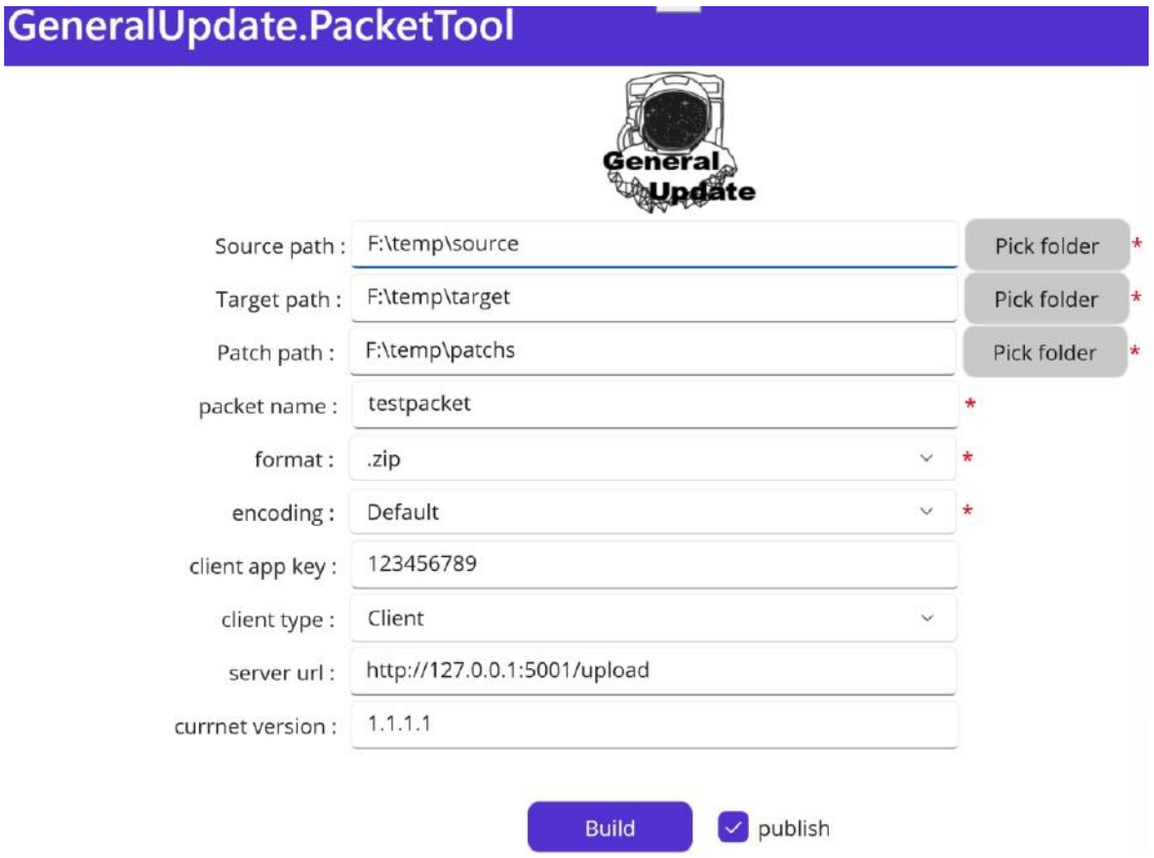
```
/*
 * Upload update package.
 *
 */
app.MapPost("/upload", async Task<string> (HttpContext context,HttpRequest request)
=>
{
    var uploadReapDTO = new UploadReapDTO();
    try
        var contextReq = context.Request;

        int.TryParse(contextReq.Form["clientType"], out int clientType);
        var version = contextReq.Form["clientType"].ToString();
        var clientAppKey = contextReq.Form["clientAppKey"].ToString();
        var md5 = contextReq.Form["md5"].ToString();


        if (! request.HasFormContentType) throw new Exception("ContentType was
notincluded
 in the request !" );


        var formFile = form.Files["file"];
        if (formFile is null || formFile.Length == 0) throw new
ArgumentNullException("Uploaded update package file not found !" );
        await using var stream = formFile.OpenReadStream();
        byte[] buffer = new byte[stream.Length];
        stream.Read(buffer, 0, buffer.Length);
        //TODO:save to file server.
        string localPath = $"E:\\{formFile.FileName}";
FileAccess.Write);
        fileStream.Write(buffer, 0, buffer.Length);

        //TODO: data persistence.To mysql , sqlserver....


        uploadReapDTO.Code = HttpStatus.OK;
        uploadReapDTO.Body = "Published successfully.";
        uploadReapDTO.Message =
        RespMessage.RequestSucceeded;
    }
    catch (Exception ex)
    {
        uploadReapDTO.Code = HttpStatus.BAD_REQUEST;
        uploadReapDTO.Body = $"Failed to publish ! Because : { ex.Message }";
        uploadReapDTO.Message = RespMessage.RequestFailed;
        return JsonConvert.SerializeObject(uploadReapDTO);
    }
});
```

3.2PostMan
Test

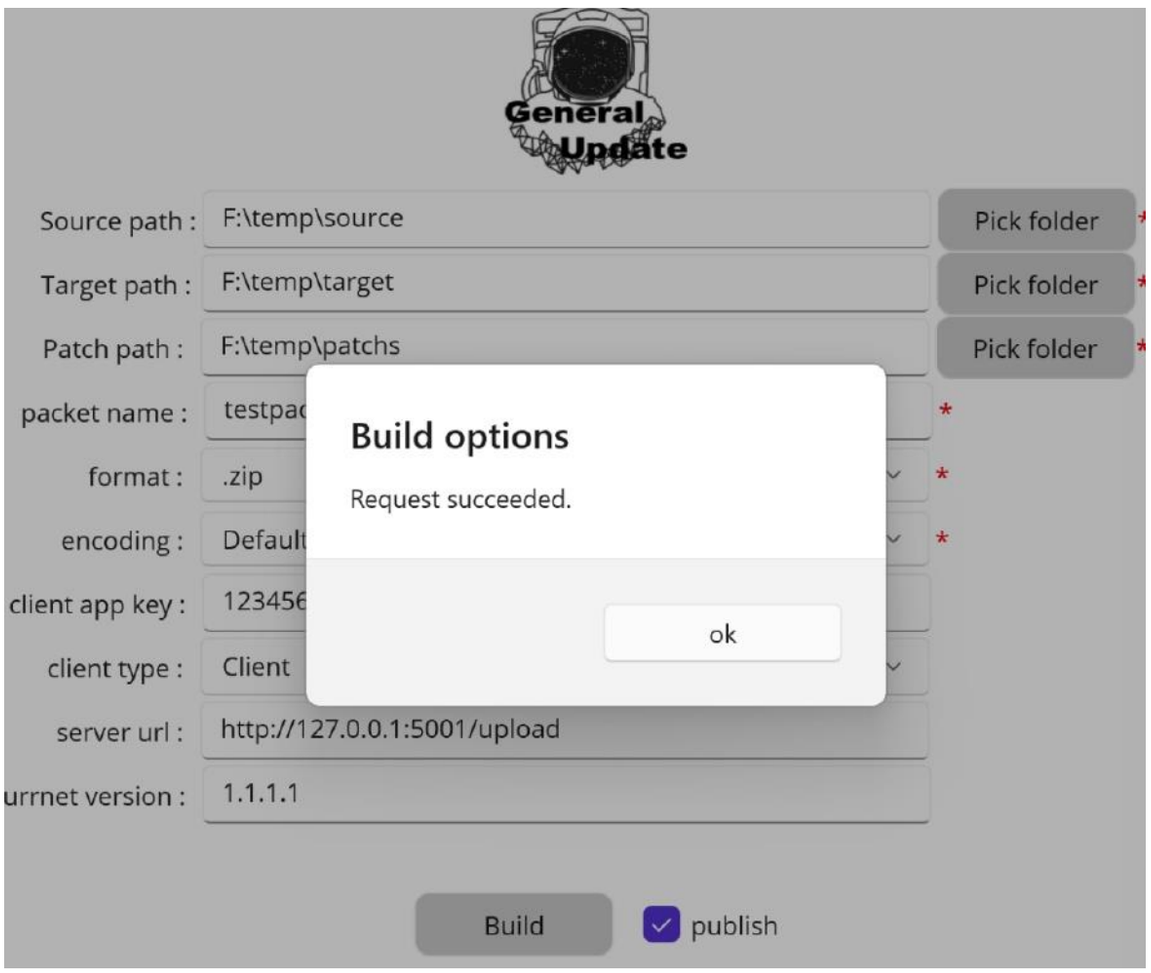


3.3 Running
Effect

Fill in the required parameters. The specific use of reference:
http://justerzhu.cn/index.php/generalupdate-packettool/



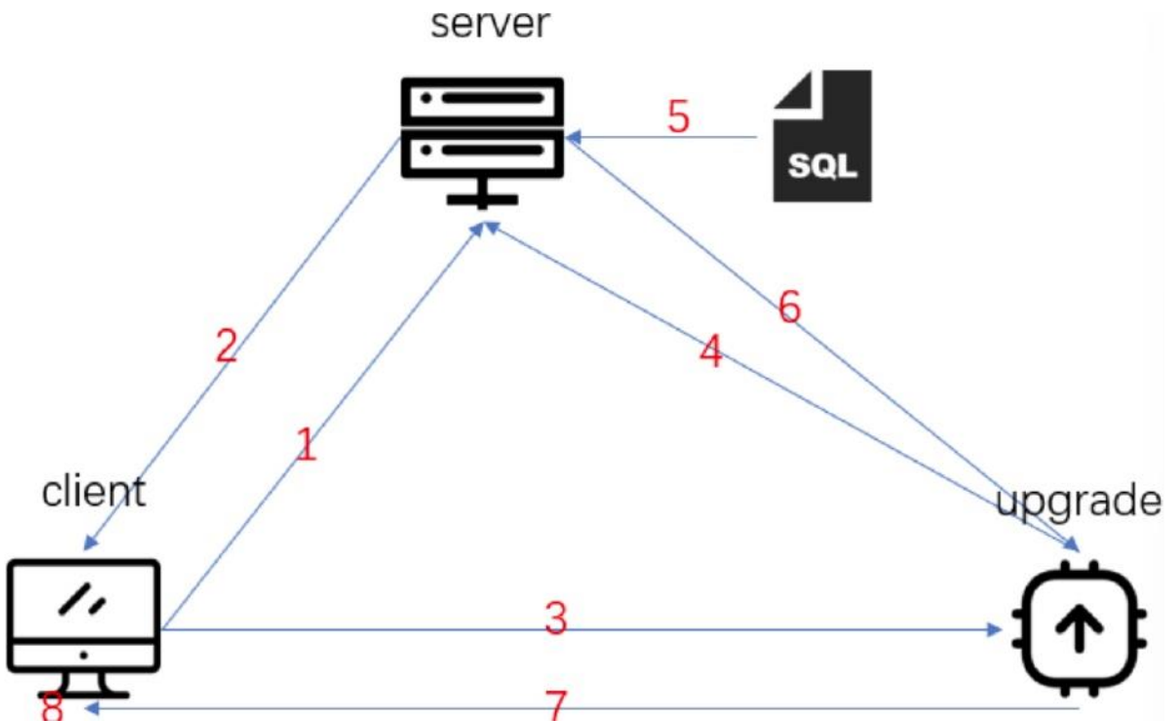
Generate the update package
and publish it to the server.



3.GeneralUpdate.ClientCore

Intr
oduc
tion

GeneralUpdate.ClientCore is one of the most core components and provides a number of major functions. ClientCore is used in the main program to update the update Assistant and then close the main program to start the Update Assistant. (Primary responsibility update Upgrade Assistant)



Step 1

After the client starts, it will send an http request to the server to confirm whether the upgrade needs to be updated.

Step 2

If upgrade needs to be updated, the upgrade package will be downloaded and updated.

Step 3

If the client finds that upgrade does not need to be updated or after the upgrade has been updated, it will start the upgrade independent application directly through the process. (That's why we need to keep the references separate.)

Step 4

After the upgrade is started, it automatically requests an update package from the client. It is used to update the contents of the client.

Step 5

The server will play a key role during client and upgrade requests for updates. Provide the version update information and version verification information to determine whether the update is required and the download address of the update package.

Step 6

After the server responds to upgrade's request, upgrade will

perform an update to the client. Step 7

After the update is complete, upgrade launches the client through the process.

Step 8

After the client is started, complete the update. (End of process)

Function provided

Feature

Features	Support or not	Remarks
Breakpoint continuation	Support	Passive effective
Version-by-version update	Support	Passive effective
Binary differential update	Support	Passive effective
Incremental update function	Support	Passive effective
The configuration file is kept newer	Support	It currently supports json configuration files with depth of 1 (temporarily offline, directly overwritten when updated)
Mandatory updates	Support	Optional update can pop up a selection box for users to choose from, while mandatory update is directly updated
Multi-branch Update	Support	When a product has multiple branches, the corresponding content needs to be updated according to the different branches
Latest version push	Support	Based on SignalR implementation

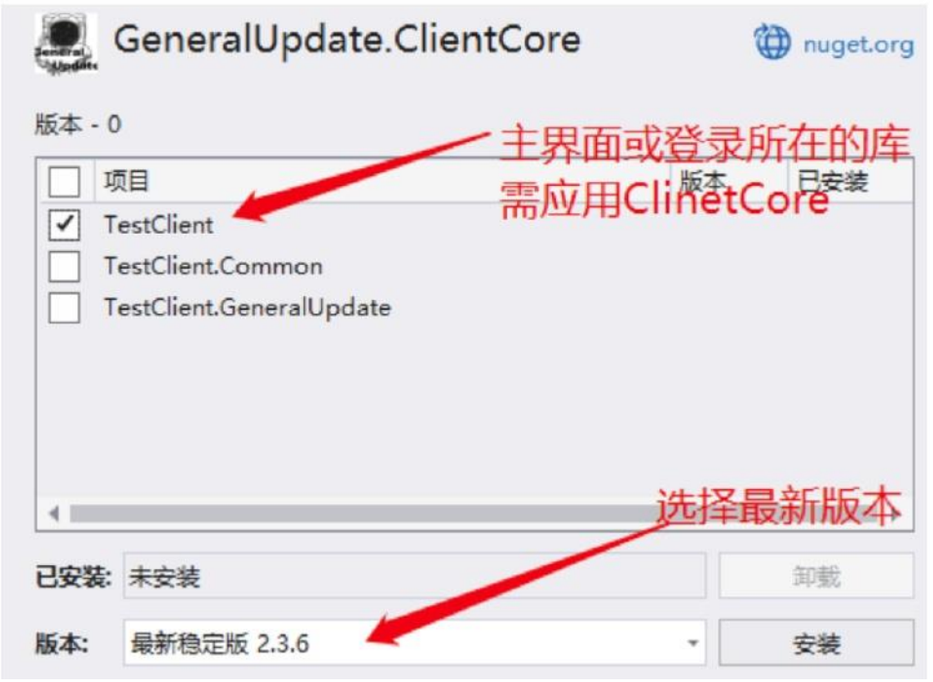
Name	cl as s	Definiti on
GeneralClientBootstrap	Class starts	
AddListenerMutiDownloadProgressChanged	pa rt y	
AddListenerMutiDownloadStatistics	Notification event for single or multiple update package download speed, remaining	
AddListenerMutiDownloadCompleted	pa rt y	
AddListenerMutiAllDownloadCompleted	pa rt y	

Name	class	Rema
AddListenerMutiDownloadError	type	Notification of an exception during the download process.
AddListenerException	party	Any problems that occur throughout the
Config	Meth	Update related content configuration parameters, url server address and port number, appSecretKey client
Option	party	
UpdateOption.DownloadTimeOut		Configure the setting method.
UpdateOption.Encoding	method	
UpdateOption.Format	gold	Set the download timeout configuration in seconds. for
SetCustomOption		Set the encoding format used for all file related processing
Strategy		, such as: compressed package decompression.
LaunchTaskAsync	gold	Format the zip pack.
LaunchAsync	for	Injects a user action method to let the user
VersionHub		skip the update when the new state is not forced .
VersionHub.Instance.Subscribe		Set the current update policy, for example: if it is a Windows party
SetBlacklist		For the platform, use the WindowsStrategy method

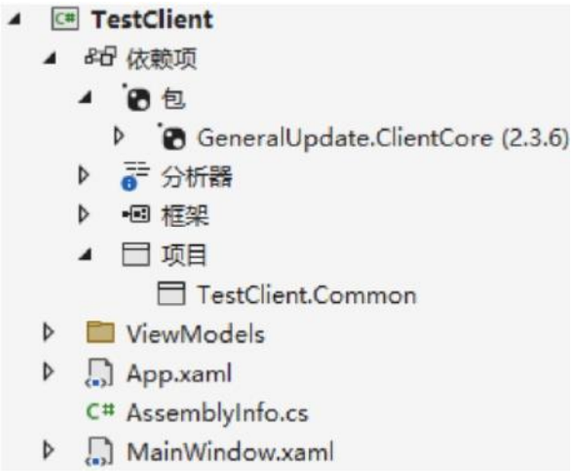
Example of use

nuget Installation

```
NuGet\Install-Package GeneralUpdate.ClientCore -Version 2.4.7
```



After the installation is complete, you will see these in the nuget package reference.



Sample code

Complete sample code: https://gitee.com/Juster-zhu/GeneralUpdate/blob/master/src/c%23/GeneralUpdate.Cli ent/MainPage. Xaml. Cs

(1) Client update code

```
 public class MainViewModel
{
    private const string baseUrl = @"http://127.0.0.1:5001";

    public MainViewModel()
```

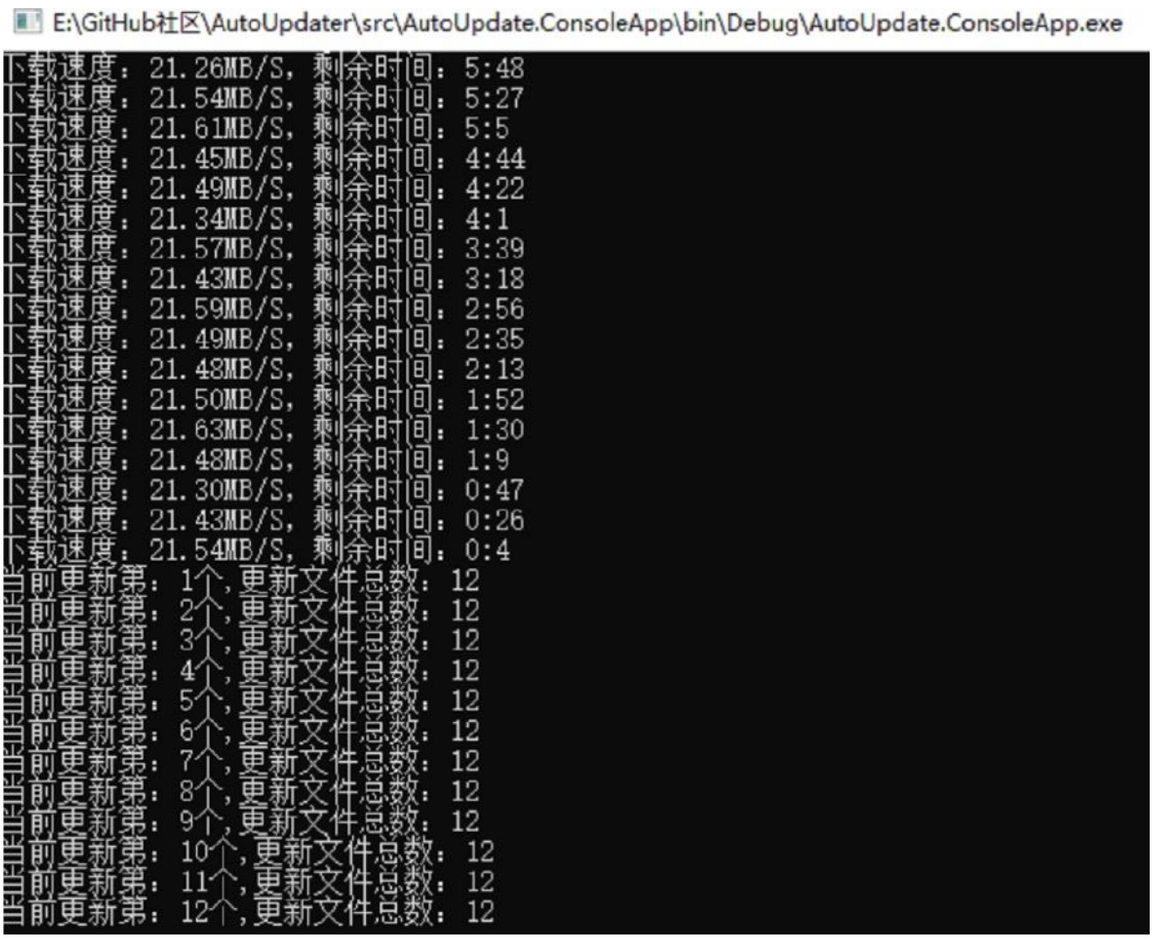
```
        {
            Upgrade();
        }


        private void Upgrade()
        {
            Task.Run(async () =>
            {
                var configinfo = GetWindowsConfiginfo();
                var generalClientBootstrap = await new GeneralClientBootstrap()
                // Single or multiple update package download notification
                events
                . AddListenerMultiDownloadProgress (OnMultiDownloadProgressChanged)
                // single or multiple update download speed, remaining download
                event notification events, the download version information
                . AddListenerMultiDownloadStatistics (OnMultiDownloadStatistics)
                // single or multiple updates the download is complete
                . AddListenerMultiDownloadCompleted (OnMultiDownloadCompleted)
                // download complete all the tasks
                . AddListenerMultiAllDownloadCompleted (OnMultiAllDownloadCompleted)
                // download process appeared abnormal notification
                .AddListenerMultiDownloadError(OnMultiDownloadError)
                // Any issues that occur throughout the update
                process will be notified via this event
                .AddListenerException(OnException)
                .Config(configinfo)
                .Option(UpdateOption.DownloadTimeOut, 60)
                .Option(UpdateOption.Encoding, Encoding.Default)
                .Option(UpdateOption.Format, Format.ZIP)
".patch", ".7z", ".zip", ".rar", ".tar" , ".json" }
                // If no extension is needed, you need to re-pass
                in the blacklist collection to overwrite.
                . SetBlacklist(GetBlackFiles(), GetBlackFormats())
            });
        }


        private void OnMutiDownloadStatistics(object sender,
MutiDownloadStatisticsEventArgs e)
        {
            var info = e.Version as VersionInfo;
            //info.Remaining Remaining
            download time
            //info.Speed Download
        }


        private void OnMutiDownloadProgressChanged(object sender,
MutiDownloadProgressChangedEventArgs e)
        {
            var info = e.Version as VersionInfo;
            // info. TotalBytesToReceive current updates need to
            download the total size
```

```
            The percentage of the / / info.
            ProgressPercentage current progress

            //info.Version Current downloaded
```

2.ProgressType.Donwload downloading the current version 3. ProgressType.Updatefile Updating the current version 4. ProgressType. Done
Updating the current version 5.ProgressType.Fail updating failed

```
        }


        private void OnException(object sender, ExceptionEventArgs e)
        {
            Debug.WriteLine(e.Exception.Message);
        }


        private void OnMutiAllDownloadCompleted(object sender,
MutiAllDownloadCompletedEventArgs e)
        {
            var info = e.Version as VersionInfo;

            //info.FailedVersions;                                                    D
Mid.
            ebug.WriteLine($"Is all download completed {e.I alldownloadcompleted) if the
}. ");

        }


        private void OnMutiDownloadCompleted(object sender,
MutiDownloadCompletedEventArgs e)
        {
            var info = e.Version as VersionInfo;

            //Debug.WriteLine($"{ e.Version.Name } download completed.");
        }

        private void OnMutiDownloadError(object sender, MutiDownloadErrorEventArgs e)
        {
            var info = e.Version as VersionInfo;
            //Debug.WriteLine($"{ e.Version.Name } error!" );
        }
    }
```

- Run



(2) Client subscription
push code

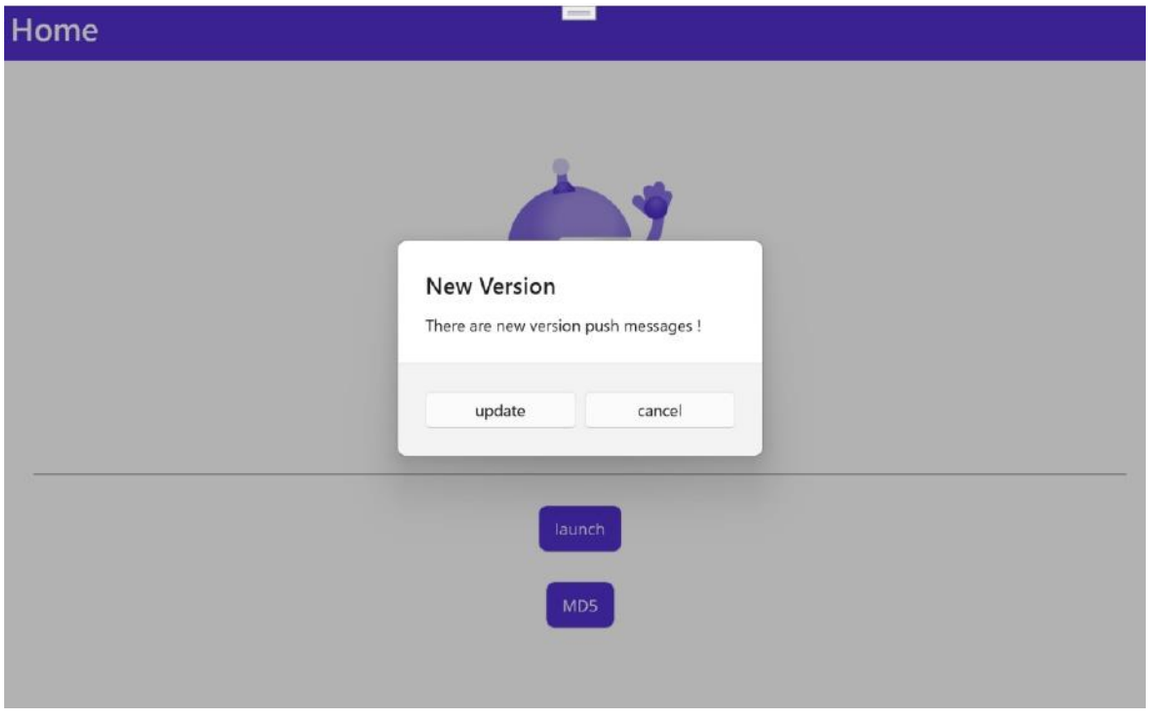
```
    private const string baseUrl = @"http://127.0.0.1:5001";
    private const string hubName = "versionhub";


    public MainPage()
    {
        InitializeComponent();

        MyButton.Clicked += OnClicked;
        Loaded += OnLoaded;

    }


    private void OnLoaded(object sender, EventArgs e)
    {

        /*
        *

        *Method
        parameters

        *url: Server
        address

        *name: Unique identifier of the client, which
        *

        VersionHub<string>.Instance.Subscribe($"{baseUrl}/{hubName}", "TESTNAME",
new Action<string>(GetMessage));

    }
```

```
    private async void GetMessage(string msg)
    {
        var isUpdate = await Shell.Current.DisplayAlert("New Version", "There are new version push messages !", "update","cancel");
        if (isUpdate) Upgrade();
```

Subscribe code definition:

```
        /// <summary>
        /// Subscribe to the latest version.
        /// </summary>
        /// <param name="url">Server address, For example: https://127.0.0.1:8080/versionhub</param>
        /// <param name="name">The name needs to be guaranteed to be unique.</param>
        /// <param name="receiveMessageCallback">Receive server push callback function, The caller needs to implement the update process.</param>
        /// <param name="onlineMessageCallback">Receive online and offline notification callback function.</param>
        /// <param name="reconnectedCallback">Reconnect notification callback function.</param>
        /// <exception cref="Exception">Subscribe exception.</exception>
        public void Subscribe(string url, string name, Action<TParameter> receiveMessageCallback, Action<string> onlineMessageCallback = null, Action<string> reconnectedCallback = null)
        {
            //CODE.....
        }
```

Run effect

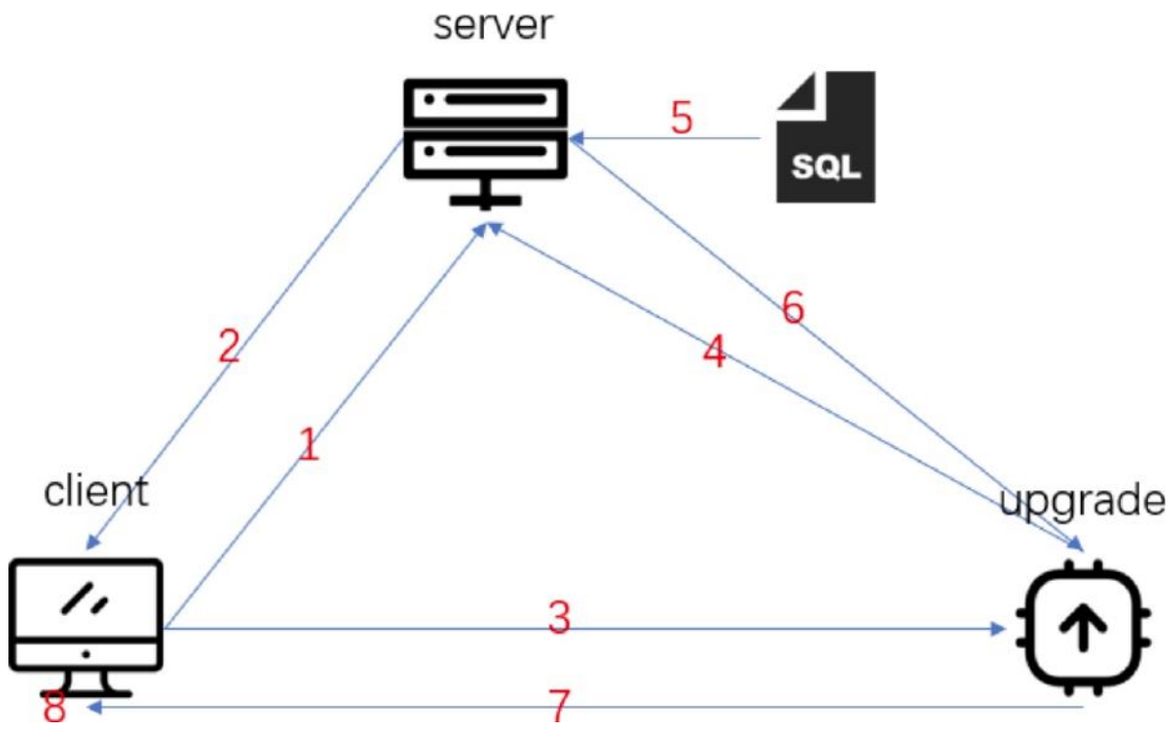


Dozens of clients can be updated at a single push

4.GeneralUpdate.Core

Intro.

GeneralUpdate.Core is one of the most core components and provides a large number of major functions. Generalupdate. core is invoked when a process starts and passes parameters to complete the main program upgrade. (Main responsibility to update the main program)



First step

After the client starts, it will send an http request to the server to confirm whether the upgrade needs to be updated.

Step 2

If upgrade needs to be updated, the upgrade package will be downloaded and updated.

Step 3

If the client finds that upgrade does not need to be updated or after the upgrade is updated, it will directly start the upgrade process independently

The application of the Upgrade process. (i.e., why you need to keep references separate above)

Step 4

After the upgrade is started, it automatically requests an update package from the client. It is used to update the contents of the client.

Step 5

The server will play a key role during client and upgrade requests for updates. Provide the version update information and version verification information to determine the YES

No update is required and the address for downloading the update package.

Step 6

After the server responds to upgrade's request, upgrade will perform an update to the client.

Step 7

After the update is complete, upgrade launches the client through the process.

Step 8

After the client is started, the update is complete. (End of process)

Supplied

Features

Features	Support or not	Remarks
Breakpoint continuation	Support	Passive effective
Version-by-version update	Support	Passive effective
Binary differential update	Support	Passive effective

Incremental update function	suppo rt	Pass ive effe ctiv e
The profile stays newer	Suppo rt	Currently refers to support for json profiles with depth of 1 (temporarily offline, directly overwrite when updated)
Mandatory updates	Suppo rt	Optional update can pop up a selection box for users to choose from, while mandatory update is directly updated
Multi-branch Update	Suppo rt	When a product has multiple branches, the corresponding content needs to be updated according to the different branches

Defi niti ons

Name	cl as s
GeneralClientBootstrap	Class starts

Name	Type	Remark
AddListenerMutiDownloadProgressChanged	party	
AddListenerMutiDownloadStatisticsAddListenerMutiDownloadCompletedAddListenerMutiAl		Single or multiple update packs download notification events. method Notification event for single or multiple update pack download speed, remaining download event, current download version information. party
		Single or multiple update package download completion events. method
lDownloadCompletedAddListenerMutiDownloadErrorException	party	
		Complete all download quest notifications. method party Notification of an exception in the download process. method Any problems that arise throughout the update process will be notified through this FA event.
Config		Update related content configuration parameters, url server address and side Port number, appSecretKey client unique identifier used for method
Option		Differentiate product branches.
UpdateOption.DownloadTimeOut	party	
UpdateOption.Encoding		Configure the setting method.
UpdateOption.Format	method	

Name	Class	Remark
Remote	Fang Fa	The component will serialize the update information object to json and serialize the image to base64 content for process information transfer.

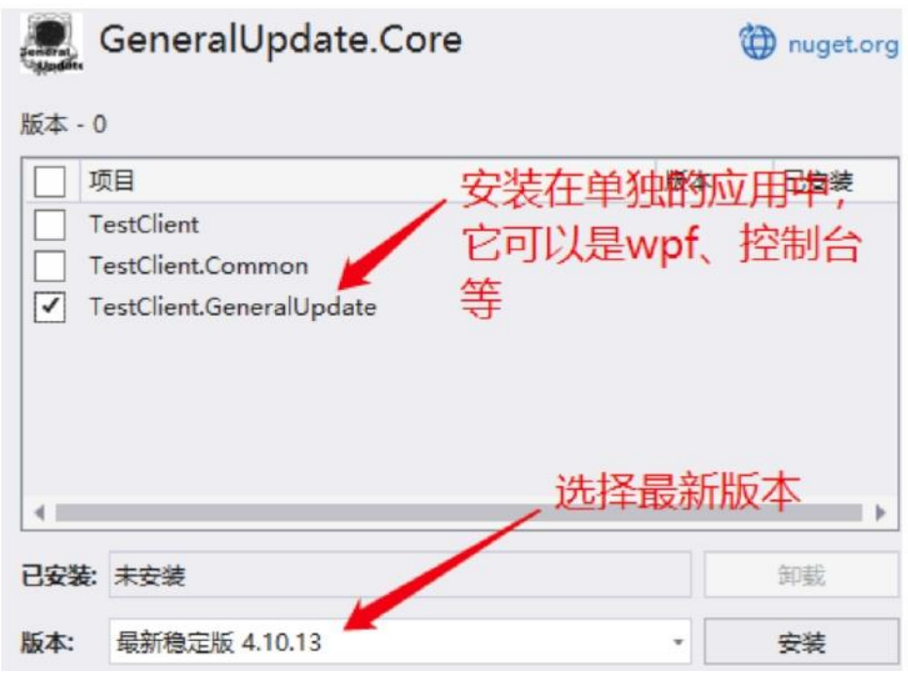
Sample base64 content:

```
eyJBcHBUeXBlIjoxLCJBcHBOYW1lIjoiR2VuZXJhbFVwZGF0ZS5DbGllbnQiLCJJbnN0YWxsUGF0aCI6IkY6XFxnaXRfcHJvamVjdFxcR2VuZXJhbFVwZGF0

ZVxcc3JjXFxjI1xcR2VuZXJhbFVwZGF0ZS5DbGllbnRcXGJpblxcRGVidWdcXG5ldDYuMC13aW5kb3dzMTAuMC4xOTA0MS4wXFx3aW4xMC14NjRcXEFwcFhc

XCIsIkN1cnJlbnRWZXJzaW9uIjoiMS4wLjAuMCIsIkxhc3RWZXJzaW9uIjoiOS45LjkuOSIsIkxvZ1VybCI6bnVsbCwiSXNVcGRhdGUiOmZhbHNlLCJDb21w
```

Example of use

nuget Installation

```
NuGet\Install-Package GeneralUpdate.Core-Version 4.10.13
```



After the installation is complete, you will see these in the nuget package reference.

Sample code

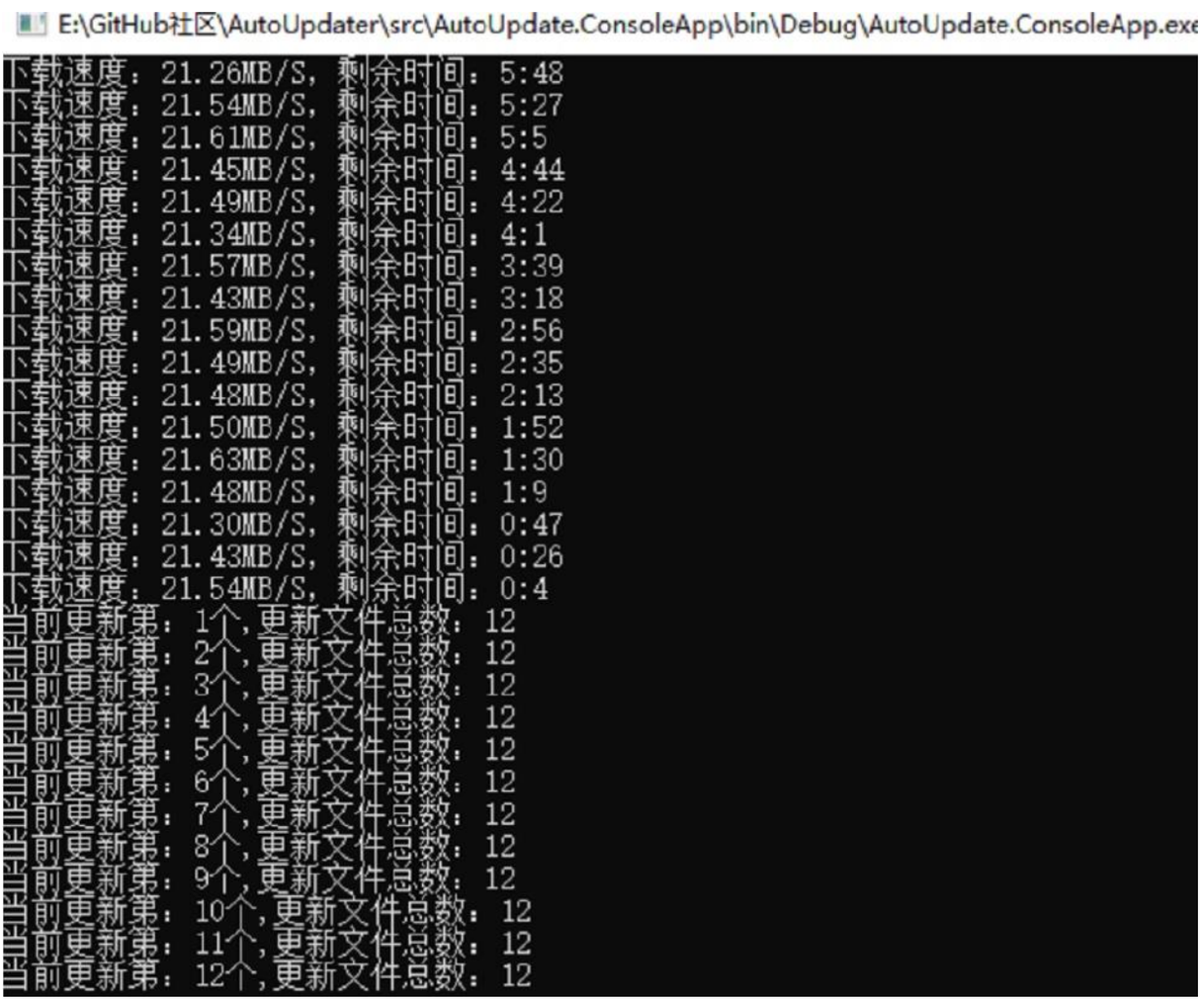
Complete sample code: https://gitee.com/Juster-zhu/GeneralUpdate/blob/master/src/c%23/GeneralUpdate.Upgrad/Program.The.cs

Upgrade Assistant

```
internal class Program
{
    static void Main(string[] args)
    {
        Task.Run(async () =>
        {
            var bootStrap = new GeneralUpdateBootstrap()
            // Single or multiple update
            pack download notification
            events
            . AddListenerMultiDownloadProgress (OnMultiDownloadProgressChanged)
            // single or multiple update download speed, remaining download
            event notification events, the download version information
            . AddListenerMultiDownloadStatistics (OnMultiDownloadStatistics)
            // single or multiple updates the download is complete
            . AddListenerMultiDownloadCompleted (OnMultiDownloadCompleted)
            // download complete all the tasks
            . AddListenerMultiAllDownloadCompleted (OnMultiAllDownloadCompleted)
            // download process appeared abnormal notification
            .AddListenerMultiDownloadError(OnMultiDownloadError)
            // Any issues that occur throughout the update
            process will be notified via this event
            .AddListenerException(OnException)
            .Strategy<WindowsStrategy>().
            Option(UpdateOption.Encoding, Encoding.Default).
        });
    }

    private void OnMutiDownloadStatistics(object sender,
MutiDownloadStatisticsEventArgs e)
    {
        var info = e.Version as VersionInfo;
        //info.Remaining
        download time //info.Speed
```

```
            //info.Version  Information about
        }


        private void OnMutiDownloadProgressChanged(object sender,
MutiDownloadProgressChangedEventArgs e)
        {
            var info = e.Version as VersionInfo;
            // info. TotalBytesToReceive current updates the total
            size of the need to download
            //info.ProgressValue The current
            progress value
            The percentage of the // info.
            ProgressPercentage current progress
2.ProgressType.Donwload downloading the current version 3. ProgressType.Updatefile
Updating the current version 4. ProgressType. Done
 Updating the current version 5.ProgressType.Fail updating failed
        }


        private void OnException(object sender, ExceptionEventArgs e)
        {
            Debug.WriteLine(e.Exception.Message);
        }


        private void OnMutiAllDownloadCompleted(object sender,
MutiAllDownloadCompletedEventArgs e)
        {
            var info = e.Version as VersionInfo;

            //info.FailedVersions;                                                    D
Mid.
            ebug.WriteLine($"Is all download completed {e.I alldownloadcompleted) if the
}. ");
        }


        private void OnMutiDownloadCompleted(object sender,
MutiDownloadCompletedEventArgs e)
        {
            var info = e.Version as VersionInfo;
            //Debug.WriteLine($"{ e.Version.Name } download completed.");
        }


        private void OnMutiDownloadError(object sender, MutiDownloadErrorEventArgs e)
        {
            var info = e.Version as VersionInfo;
            //Debug.WriteLine($"{ e.Version.Name } error!" );
        }
    }
```

- Run



5.GeneralUpdate.Differential

Intro.

GeneralUpdate.ClientCore is one of the most core components, providing differential detection and incremental discovery functions.

Features provided

The component will perform a recursive lookup (tree structure) to all files in the source folder directory, target folder directory. Then, the MD5 code and file name of each file are compared, the difference files in the two directories are extracted to generate binary difference files and the newly added files are directly copied to the update package. (Special note: The internal component will be based on target operations, so the source and target parameters can not be filled in the reverse order.)

Features

Features	Supported or not	Remar

		k s
Binary Differential patch	Suppo rt	Binary differential patch kb level for differences between old and new versions.
New file packaging	Suppo rt	Files in the new version that were not in the old version will be packaged directly into the update package.

Defi
niti
ons

Name	Type	rema
Clean	Meth	Clean out the files that
Drity	Meth	Update the contents of the

source: Old version folder directory.

target: New version folder directory.

patchs: Binary differential patch, new files, etc. generate results folder directory (compressed package in the target directory). Examples of usage

nuget Installation

```
NuGet \ Install - Package GeneralUpdate. Differential - Version
```

Example code key method definition:

Clean Definition

```
        /// <summary>
        /// Generate patch file [Cannot contain files with the same name but different extensions].
        /// </summary>
        /// <param name="appPath">Previous version folder path .</param>
        /// <param name="targetPath">Recent version folder path.</param>
        /// <param name="patchPath">Store discovered incremental update files in a temporary directory .</param>
        /// <param name="compressProgressCallback">Incremental package generation progress callback function.</param>
        /// <param name="type">7z or zip</param>
        /// <param name="encoding">Incremental packet encoding format .</param>
        /// <returns></returns>
= null, Action<object, BaseCompressProgressEventArgs> compressProgressCallback = null, OperationType type = OperationType.GZip, Encoding encoding = null,string name = null)
        {
          //CODE...
        }
```

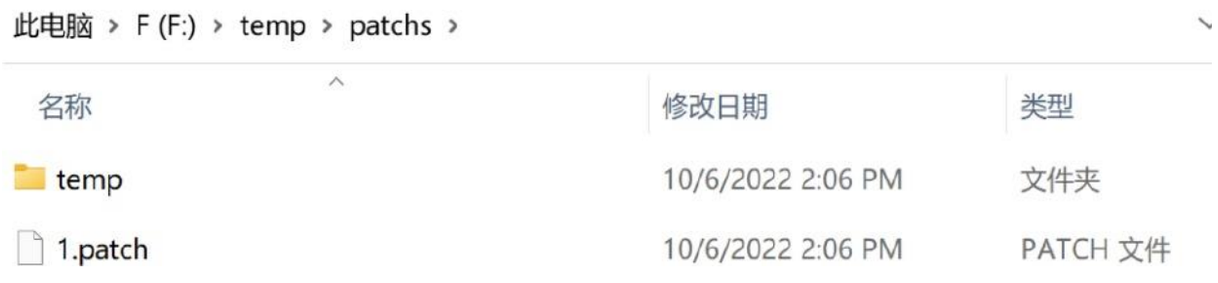
Drity
definiti
on

```
        /// <summary>
        /// Apply patch [Cannot contain files with the same name but different extensions] .
        /// </summary>
        /// <param name="appPath">Client application directory .</param>
        /// <param name="patchPath"></param>
        /// <returns></returns>
        /// <exception cref="Exception"></exception>
        public async Task Drity(string appPath, string patchPath)
        {
            //CODE...
        }
```

Different core features

```
// Clean up binary differential file patches and new files
private void BtnClean_Click(object sender, RoutedEventArgs e)
{
        Task.Run(async () =>
            var path1 = @"D:\TestCode\compare\source";
            var path2 = @"D:\TestCode\compare\target";
            var path3 = @"D:\TestCode\compare\patchs";
            await DifferentialCoreInstance.Clean(path1, path2, path3);
        });
}

// "Supplement" the binary diff file patch and the new file to the corresponding file to complete the update
private void BtnDrity_Click(object sender, RoutedEventArgs e)
        Task.Run(async () =>
        {
            var path1 = @"D:\TestCode\compare\source";
            var path3 = @"D:\TestCode\compare\patchs";
            await DifferentialCoreInstance.Drity(path1, path3);
        });
}
```

Run effect



6.GeneralUpdate.Single

Intro.

Component provides updaters to run singly, preventing repeated starting of updaters. The main purpose is to avoid the problem of multiple update assistants being started causing updates to be confused. (Currently only supports.NET Framework, which is the only component of this component that cannot be cross-platform. Compatibility will be considered later.)

Features Provided

Function prevents repeated startup of the update program.

Definition

Name	Type	Comm	Rema
ISingleInstanceApp	cl	Constraints, to	Implement
SingleInstance	cl	Register classes	
InitializeAsFirstInstance	Meth	Initial	Must
Cleanup	Meth	Reset	Must

Use the sample

nuget installation

```
NuGet\Install-Package GeneralUpdate.Single-Version 1.0.0
```

Sample code

```
class Program : Application, ISingleInstanceApp
{

    private const string AppId = "{7F280539-0814-4F9C-95BF-D2BB60023657}";

    [STAThread]

    static void Main(string[] args)
    {
        //ISingleInstanceApp Can only be used in the framework of .net framework .
        if (SingleInstance<Program>.InitializeAsFirstInstance(AppId))

        {

            var win = new MainWindow();

            var vm = new MainViewModel(resultArgs, win.Close);

            win.DataContext = vm;

            var application = new Program();

            application.Run(win);

        }
    }


    public bool SignalExternalCommandLineArgs(IList<string> args)
    {
```

```
            if (this.MainWindow.WindowState == WindowState.Minimized)
            {
                this.MainWindow.WindowState = WindowState.Normal;
            }
            this.MainWindow.Activate();

            return true;
        }
    }
```

7 GeneralUpdate.Zip

Introduction

This component provides the

furnish

- Features

Features	Whether supported or not	Remark
Compression	Support	

-

Name	Type	Remarks
GeneralZipFactory	class	Main startup class.
UnZipProgress	Events	Unpack progress.
CompressProgress	Events	Compress progress.
Completed	Events	Action complete notification.
UnZip	Method	Decompress.
CreatZip	Method	Create the zip package.
CreatefOperate	method	Compress the configuration.

Examples of use

nuget
Installa
tion

```
NuGet\Install-Package GeneralUpdate.Zip-Version 2.2.0
```

Sample code

Complete sample code: https://gitee.com/Juster-zhu/GeneralUpdate/blob/master/src/c%23/TestZIP/UnitTests.

Key method definitions:

```
        /// <summary>
        /// Configuring Compression.
        /// </summary>
        /// <param name="type">Enumeration, selects the compressed package format to operate on.(OperationType.GZip , OperationType.G7z)</param>
        /// <param name="name">Compressed package Name.</param>
        /// <param name="sourcePath">Source file path.</param>
        /// <param name="destinationPath">The target path.</param>
        /// <param name="includeBaseDirectory">Whether to include the root directory when packing.</param>
        /// <param name="encoding">Compressed package encoding format.</param>
        /// <returns></returns>
        /// <exception cref="ArgumentNullException"></exception>
        /// <exception cref="Exception"></exception>
        public void Compress(string type, string name, string sourcePath, string destinationPath, bool includeBaseDirectory = false, Encoding encoding = null)
        {
            //CODE...
        }
```

Uncompress

```
        [Test]
        public void UnZip()
        {
            try
            {
                string sourcePath = "D:\\Updatetest_hub\\Run_app\\1.zip";
                string destinationPath = "D:\\Updatetest_hub";
                string name = "testpacket.zip";
                var factory = new GeneralZipFactory();
                factory.UnZipProgress += (a,e) =>
                {
                    Console.WriteLine($"fileName:{e.Name},fileSize:
{e.Size},fileIndex:{e.Index},filePath:{e.Path},fileCount:{e.Count}");
                };
                factory.Completed += (a, e) =>
                {
                    Console.WriteLine($"IsCompleted:{e.IsCompleted}");
                };
                factory.CreatefOperate(OperationType.GZip, name,
sourcePath,
destinationPath, false, System.Text.Encoding.Default).
```

```
            catch (Exception ex)
            {
            }
        }
```

- Run

- To create the

```
        [Test]
        public void CreatZip()
        {
            try
            {
                string sourcePath = "D:\\Updatetest_hub\\Run_app";
                string destinationPath = "D:\\Updatetest_hub";
                string name = "testpacket.zip";
                var factory = new GeneralZipFactory();
                factory.CompressProgress += (a,e) =>
                {
                    Console.WriteLine($"fileName:{e.Name},fileSize:
{e.Size},fileIndex:{e.Index},filePath:{e.Path},fileCount:{e.Count}");
                };
                factory.Completed += (a,e) =>
                {
                    Console.WriteLine($"IsCompleted:{ e.IsCompleted }");
                };
                factory.CreatefOperate(OperationType.GZip, name,
sourcePath,
destinationPath,false,System.Text.Encoding.Default).
                    CreatZip();
            }
            catch
            {
                Assert.Fail();
            }
```

Run
effect

8.GeneralUpdate.PacketTool

Intro.

The tool is developed using.NET MAUI (.NET 6) and is currently only supported for desktop use. Help developers quickly generate update package content (new and old versions of the binary difference patch kb level, new files).

Features provided

Features

Features	Support	Remarks
Differential update packet	Support	Compare the previous version with the current version to find files that need to be updated or new files added.
Automatically upload the update package	Support	Automatically upload the generated differential update package to the server.
Multi-platform support	Support	Windows, Linux, MacCatalyst only supported.

Definition

Name	Non-empty	Remarks
Source path	no	Represents the folder path of the previous version.
Target path	no	Indicates the current version folder path.
Patch path	no	Indicates the path to the final generation of the subcontract.

Packet Name	no	Specify the update package name, or randomly generate it if not specified.
Format	no	zip package format, optional zip, 7z.
Encoding	no	Encoding format of the compressed package. If garbled characters occur, specify the encoding format.
Client app key	is	Can be empty when no upload publication is required to the server, used for product multi-branch tagging and management. Example value: {B13DAE4E-7881-4158-948C-24681CCA413F}

name	No n-	Rema '
Server Url	Yes	This parameter can be left blank when no upload is required to the server. Enter the full interface address of the apb ap
Current version	Is empty	when no upload is required to publish to the server. Upload the update p
Publish	HTTP	If checked then the generated good subcontract will be uploaded via and automatically inserted in the database of this published version is
Build		By recursively comparing all the project files in the Source path and Target path folders (DLL, exe... No, etc.), through the binary difference check, incremental check to analyze the need to update the file list

Example of use

Tool compilation

Source code to compile: https://gitee.com/Juster-zhu/GeneralUpdate/blob/master/src/c%23/GeneralUpdate.Tool.sln compile tools

With red "asterisk" items, are required fields. If you do not need to use the automatic Publish function, only fill in the red asterisk (you do not need to check the "Publish" box). If you want to use the automatic Publish function, you need to fill in all, and select the "Publish" check box.

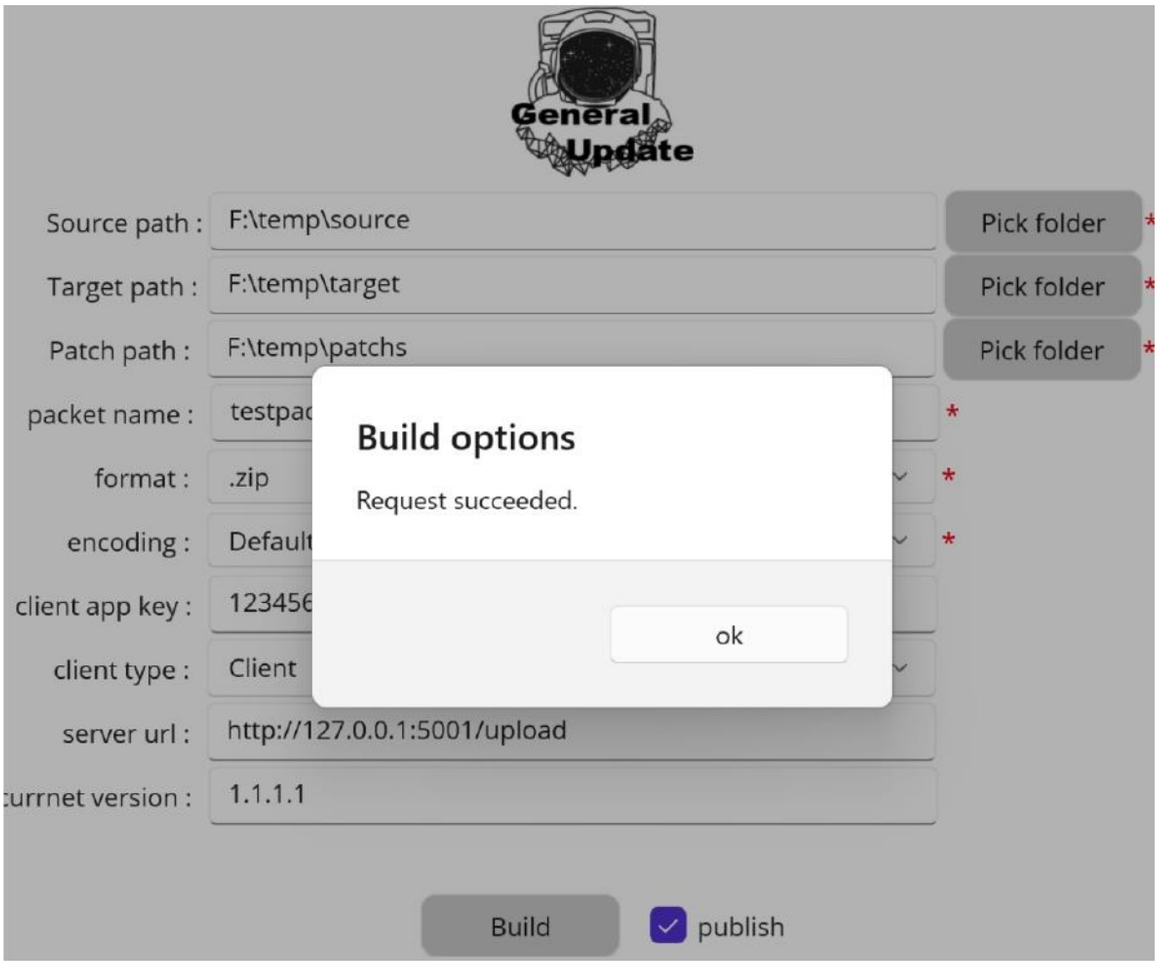
Running effect

GeneralUpdate.PacketTool

Source path :	F:\temp\source	Pick folder *
Target path :	F:\temp\target	Pick folder *
Patch path :	F:\temp\patchs	Pick folder *
packet name :	testpacket	*
format :	.zip	*
encoding :	Default	*
client app key :	123456789	
client type :	Client	
server url :	http://127.0.0.1:5001/upload	
currnet version :	1.1.1.1	

Build

publish



Generate
results

The tool will generate the update package in targetpath, default.zip format.

Update
package
contents

9.Minimal API

Introduction

Minimal API skills learning: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/minimal-apis/overview?view=aspnetcore-7.0

10.MAUI

Here share in.NET MAUI how to use GeneralUpdate, edited as required by the current version. If the.csproj file of the NET MAUI project is not configured, the GeneralUpdate cannot be opened using the process. NET MAUI written client application program.

The.net MAUI environment configuration: https://mp.weixin.qq.com/s/GycEDSpVlRCloTdvi1QgSA

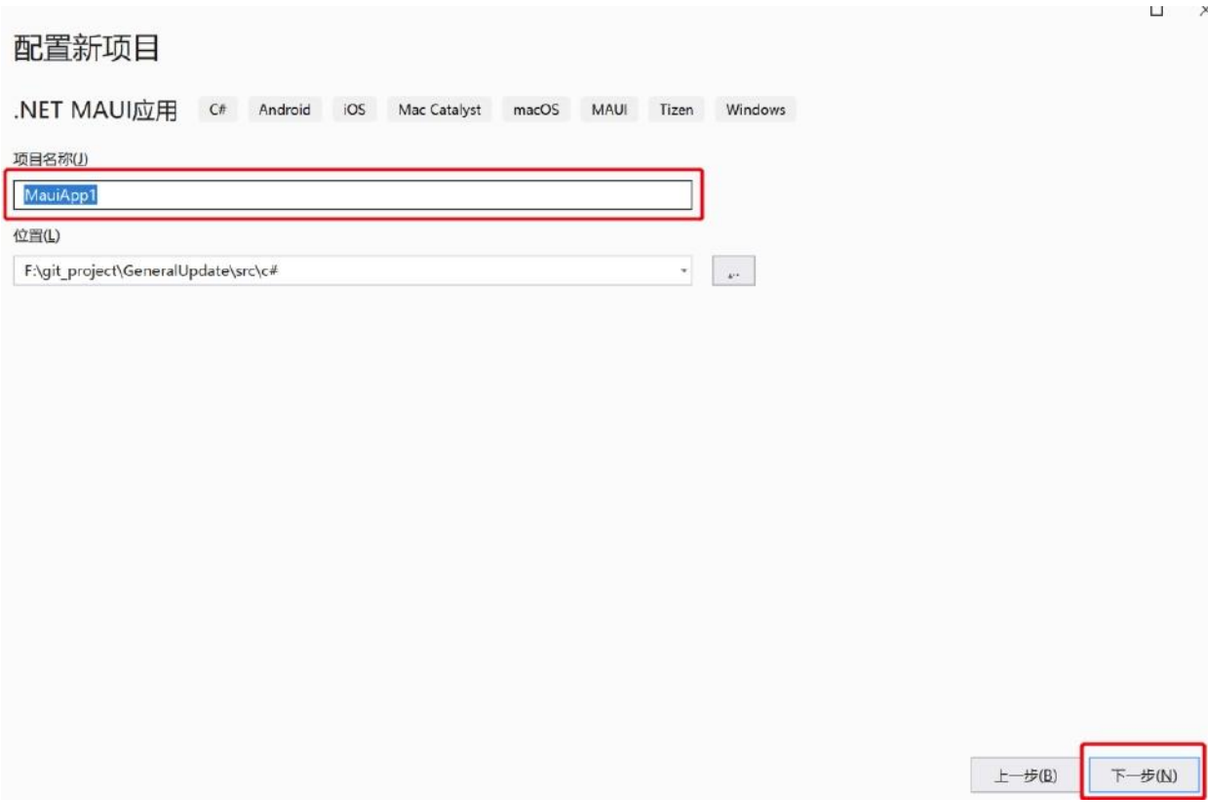
The.net MAUI skill learning: https://learn.microsoft.com/zh-cn/dotnet/maui/what-is-maui

using the sample complete code: https://gitee.com/Juster-zhu/GeneralUpdate/blob/master/src/c%23/GeneralUpdate.Client/MainPage.xaml.cs

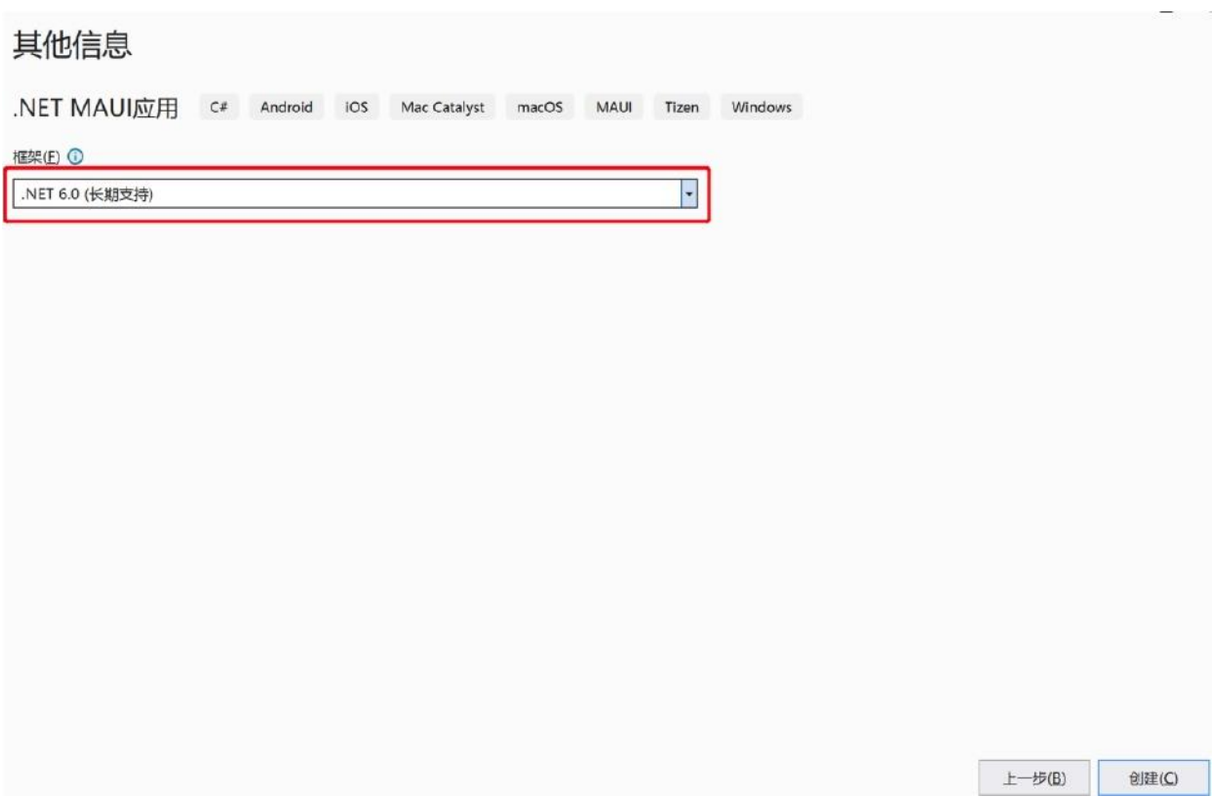
Compile MacCatalyst platform tutorial: https://mp.weixin.qq.com/s/1Lkzd7ZxRYGbjdAPQ2xl9A compile Windows platform tutorial: https://learn.microsoft.com/zh-cn/dotnet/maui/windows/deployment/overview

Steps 1 New project

2. Configure the project

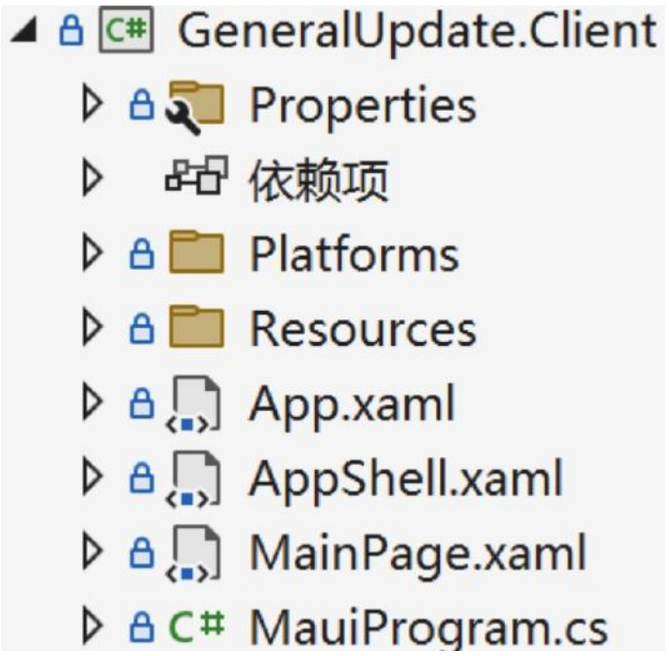


3. Select the.NET framework version

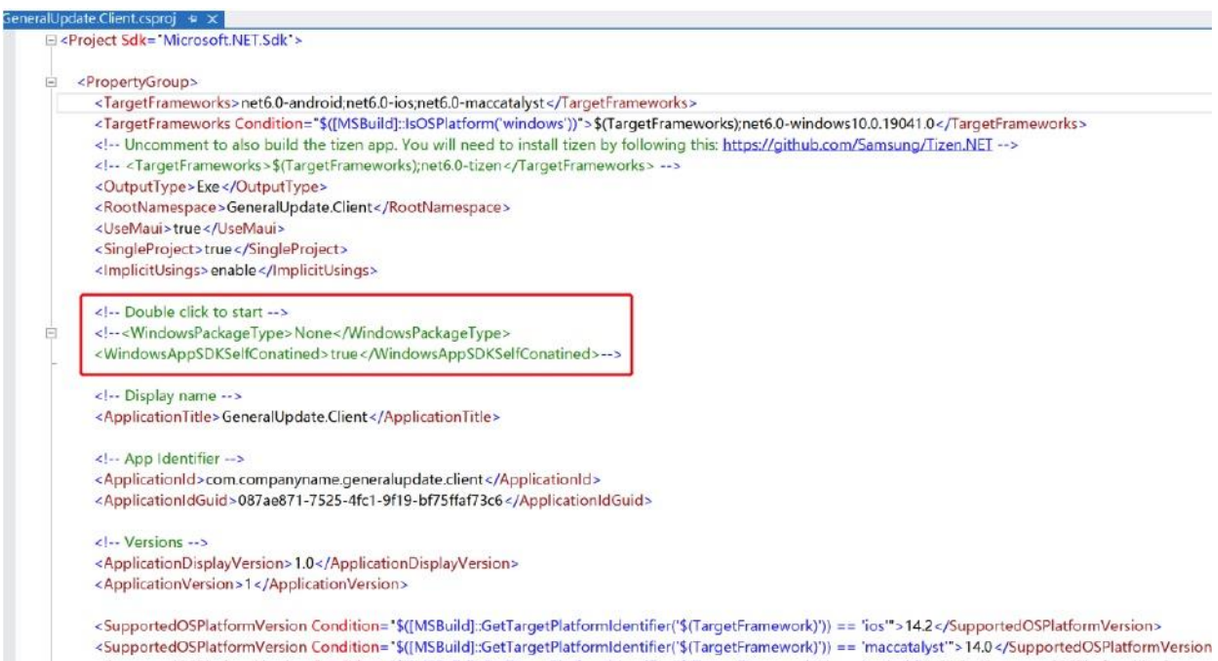


4 Locate the.csproj file

Find the.csproj of the.NET MAUI project and double-click it directly.



5 Edit the.csproj file

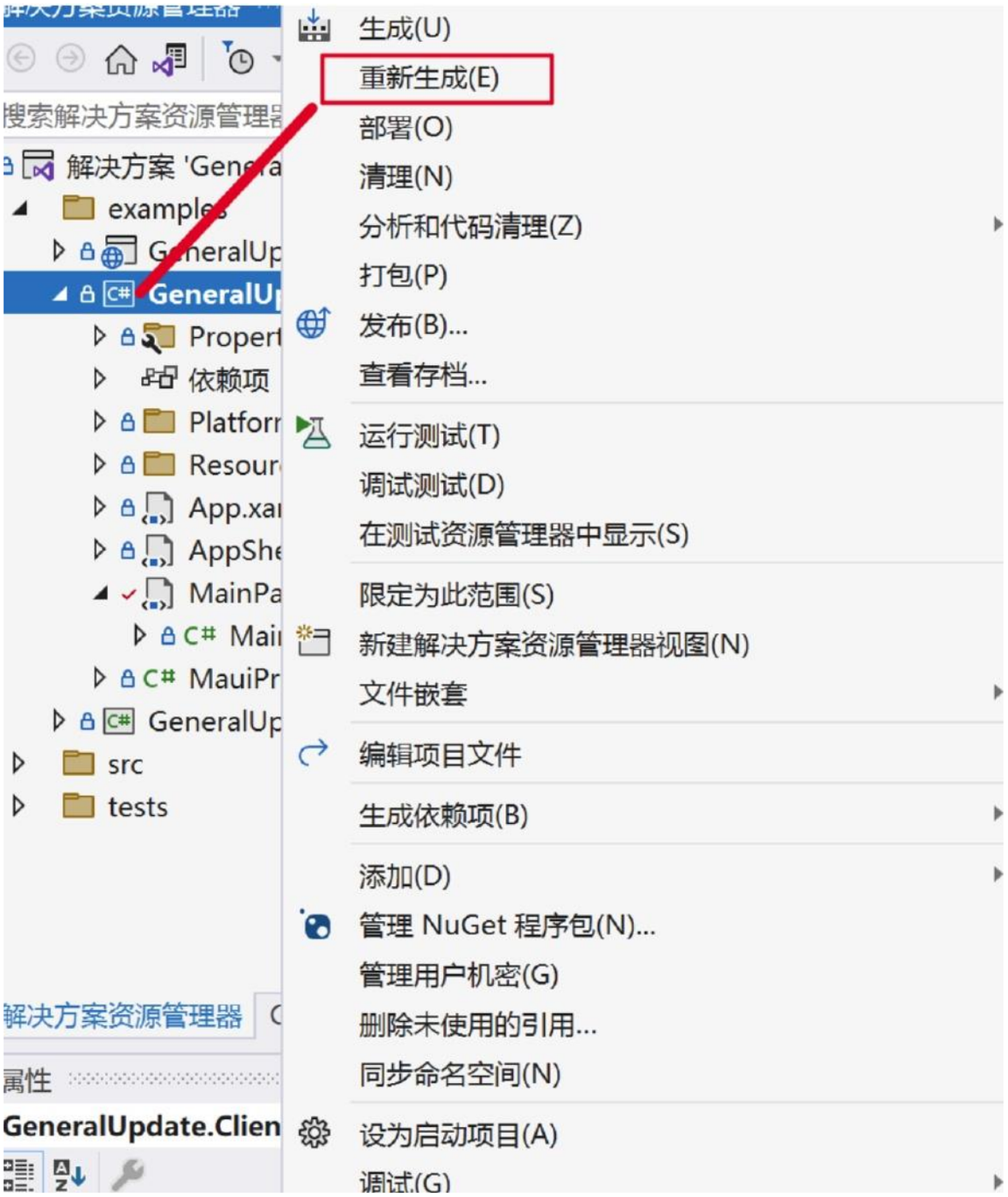


6 Add labels

```
<WindowsPackageType>None</WindowsPackageType>
<WindowsAppSDKSelfContained>true</WindowsAppSDKSelfContained>
```

7. Regenerate

You'll need to "regenerate" after you've added the label.



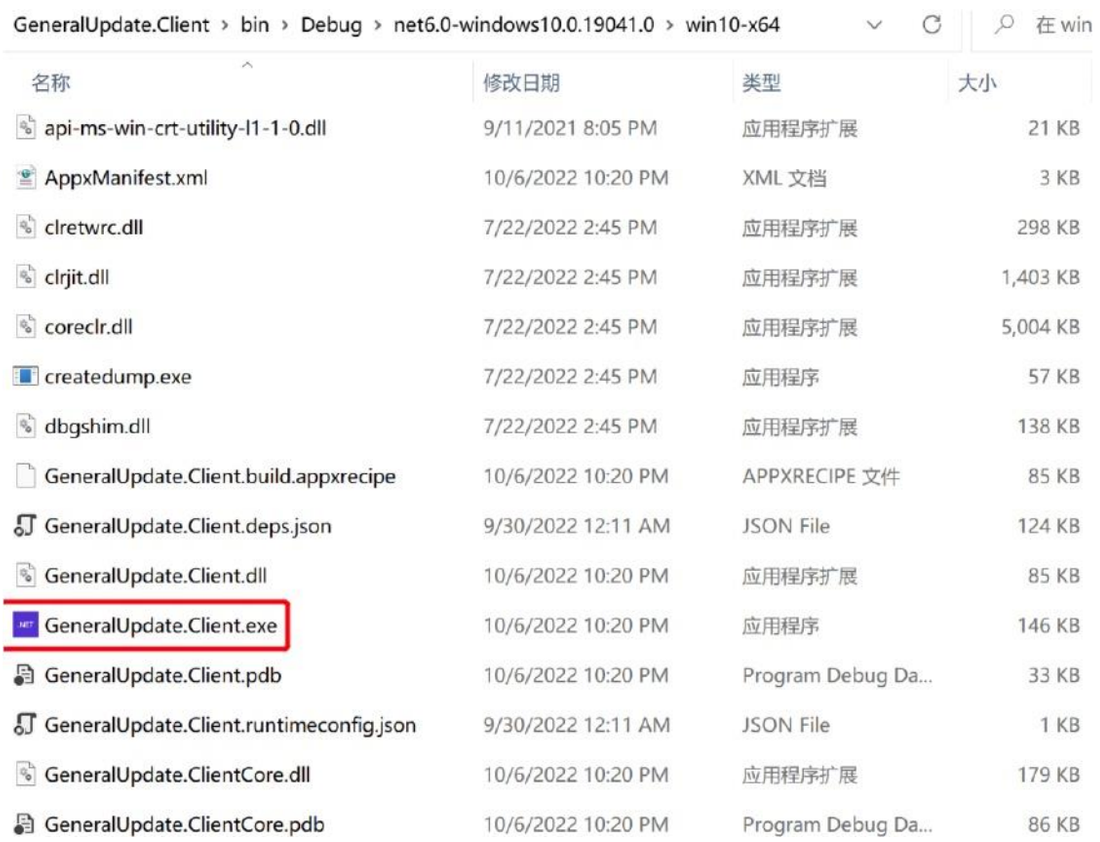
8 Run the effect

Double-click to open or use process startup to open.exe. NET MAUI program.

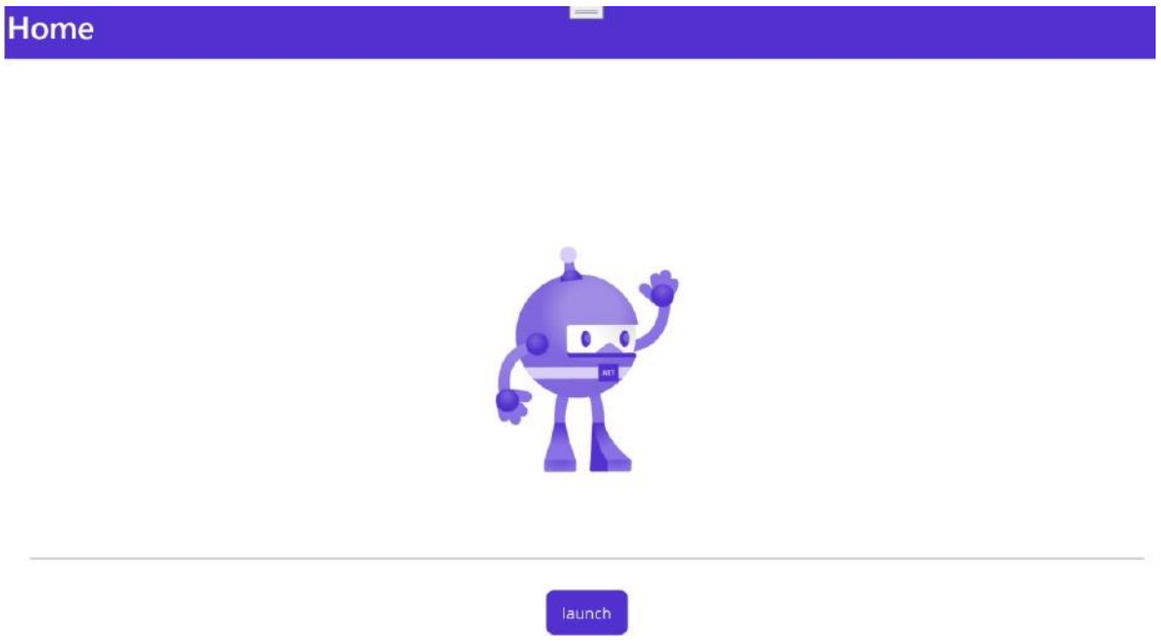
Reference path:

```
F: \ git_project \ GeneralUpdate \ SRC \ \ GeneralUpdate c # Client \ bin \ Debug \ net6.0
```

Directory:



Run effect



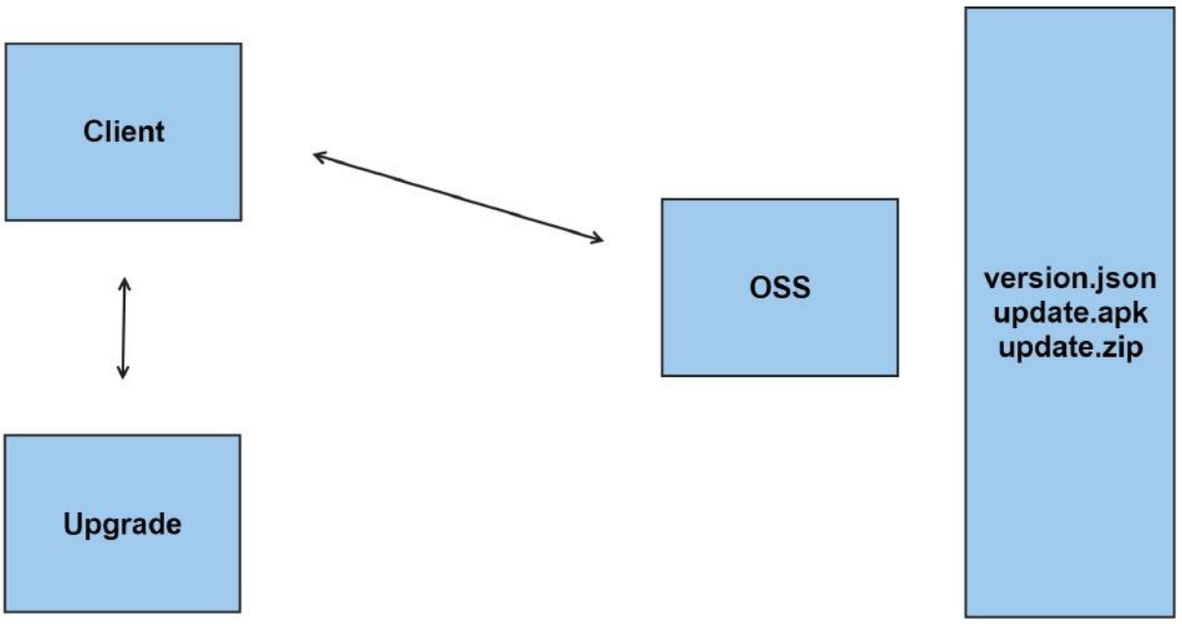
11.OSS

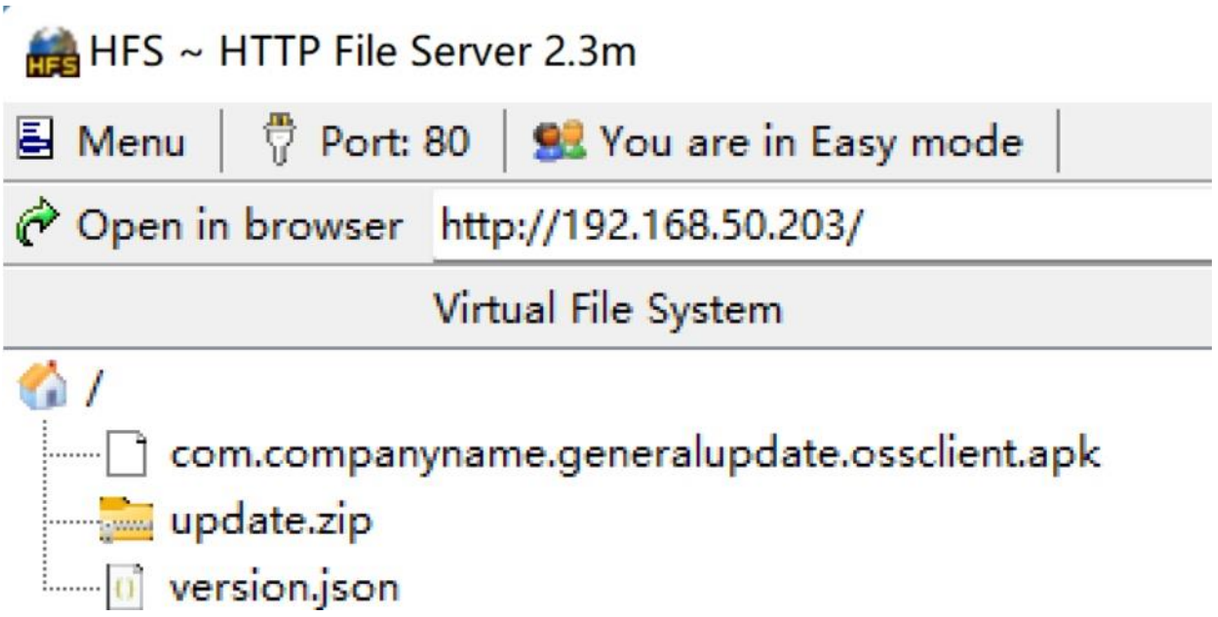
1. What is OSS?

The full name of OSS is Object Storage Service (Object Storage Service), do server-side technology stack development partners must be no stranger to this in the major cloud server vendors will provide similar services, say simply a file server. For example: Ali cloud OSS, Tencent cloud COS, Huawei cloud OBS, in fact, they are just not the same name function services are similar. Then the update component of the new function of the idea is very similar, then choose OSS to name the function, and convenient developers can get the function of this function at a glance (need to use components

GeneralUpdate.ClientCore, GeneralUpdate.Core).

2. Operating principle of GeneralUpdate.OSS


Client
OSS
version.json
update.apk
update.zip
Upgrade



json version information configuration file. update file (update.zip) The update file is packaged in the same way as before.

```
[
    {
        "PubTime": 1680443321,
        "Name" : "generalupdate  ossclient"
        , "MD5" : "9
        bf414990a67e74f11752d03f49b15d8"
        , "Version" : "1.0.4."
    }
    ,
        "PubTime": 1680444916,
        "Name": "generalupdate.ossclient",
        "MD5" :
        "JXC122DFXCZXZNMRFf11752d03f49b15d8"
        , "Version" : "1.0.5."
    }
]
```

2. When the Client starts, directly request the OSS server or file server and download the version.json file.

3. After downloading the version information to the local PC, analyze the version information to determine whether the Upgrade needs to be updated. If the information is transmitted through the process startup, the upgrade is transmitted (the Client automatically shuts down).

4. After the Upgrade is started, directly download update.zip, and then decompress the file to overwrite the local file.

5. After the Upgrade is updated,start the Client and close the client. The upgrade is complete.

GeneralUpdateOSS has a very low threshold of application compared to GeneralUpdateBootstrap. Generalupdateoss can be used if the company does not have high requirements for automatic updates. In a word, this function is to download version.json according to the contents of the file to download the update package version by version, after download directly decompress the update is over.

3. Quick boot

Client (Main client) Example code to use:

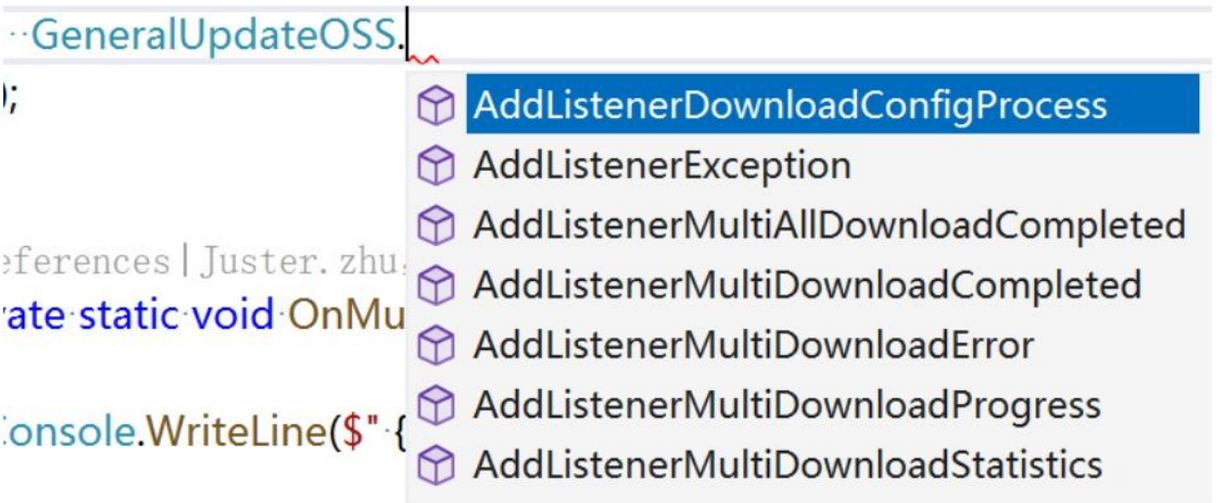
```
Task.Run(async () =>
{
        var url = "http://192.168.50.203";
        var appName = "GeneralUpdate.Client";
        var version = "1.0.0.0";
        var versionFileName = "version.json";
        ParamsOSS @params = new ParamsOSS(url, appName, version, versionFileName);
});
```

Example code to use for Upgrade:

```
private static void Main(string[] args)
{
    Task.Run(async () =>
    {
        //var url = "http://192.168.50.203";
        //var appName = "GeneralUpdate.Client";
        //var version = "1.0.0";
        //var versionFileName = "version.json";
        //SerializeUtil.Deserialize<ParamsOSS>(args[0]);
        //ParamsOSS @params = new ParamsOSS(url, appName, version, versionFileName);
        ParamsOSS @params = SerializeUtil.Deserialize<ParamsOSS>(args[0]);
    });
}
```

4 Subscribe to event notifications

During the OSS update process, the parameters of the update event are kept the same as before.

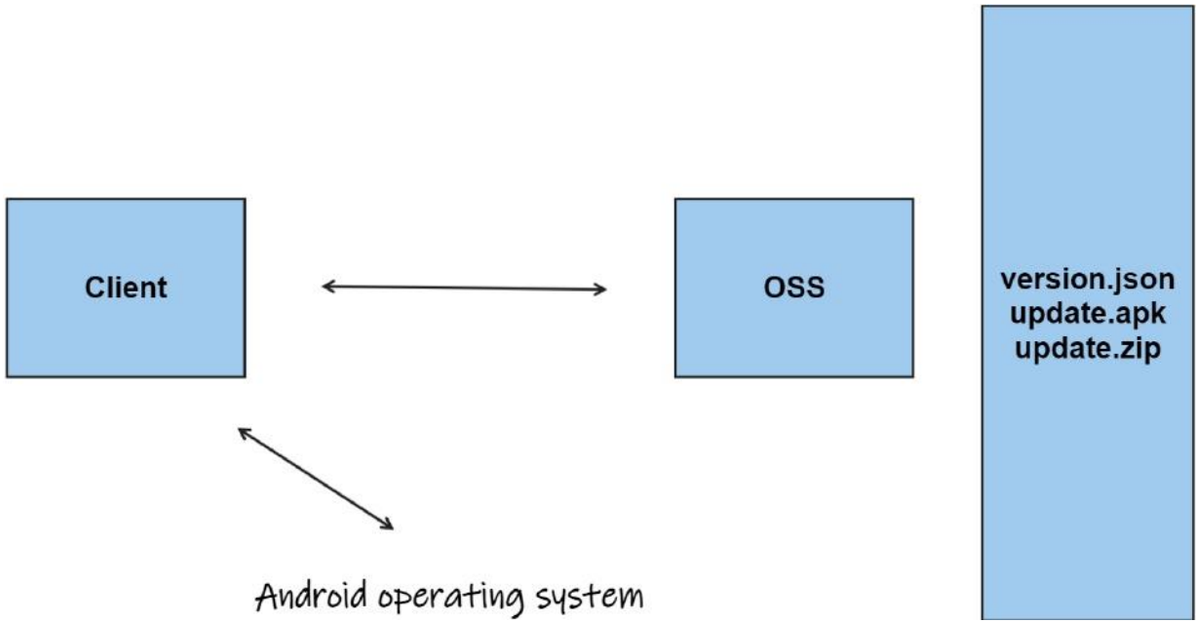


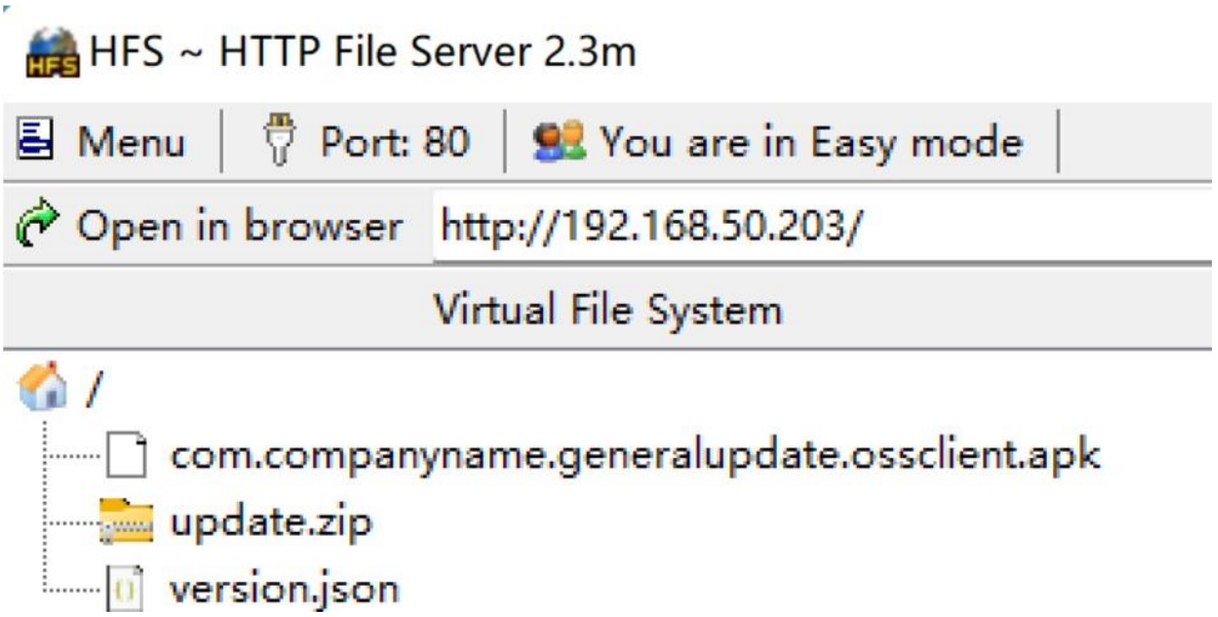

```
//code...
GeneralUpdateOSS.AddListenerMultiDownloadStatistics(OnMultiDownloadStatistics);

private static void OnMultiDownloadStatistics(object sender,
MultiDownloadStatisticsEventArgs e)
{
    Console.WriteLine($" {e.Speed} , {e.Remaining.ToShortTimeString()}");
}
```

12. .NET MAUI OSS

The MAUI OSS feature description is the same as in 11.OSS. However, it is an update written for.NET MAUI and uses the component library GeneralUpdate.Maui.OSS. Currently only automatic updates for the MAUI Andorid platform are implemented.

1. Operating principle of GeneralUpdate.Maui.OSS





json version information configuration file. update file (update.apk) The update file is the apk of the new version (or.abb).

```
{
    "PubTime": 1680444916,
    "Name": "com.companyname.generalupdate.ossclient",
    "MD5": "9bf414990a67e74f11752d03f49b15d8",
    "Version": "1.0.5",
}
```

2. When Client starts, directly request the OSS server or file server and download the version.json file.

3. After downloading the file to the local PC, parse the version information and determine whether to update it.

4. Download update.apk if you need to update it.

5. After the download is complete, install it. This step is handed over to the Android operating system. After execution, run the new version of app.

2. Start quickly

```
//
http://192.168.50.203/version.json
string url = "http://192.168.50.203";
string appName = "MainApplication.exe";
string currentVersion = "1.1.1.1";
string versionFileName = "versions.json";
GeneralUpdateOSS.AddListenerDownloadProcess(OnOSSDownload);
GeneralUpdateOSS.AddListenerException(OnException);
```

3. Event notification subscription

```
GeneralUpdateOSS.AddListenerDownloadProcess(OnOSSDownload
);


private void OnOSSDownload(object sender, OSSDownloadArgs e)
{
    Console.WriteLine($"{e.ReadLength},{e.TotalLength}");
}


private void OnException(object sender, ExceptionEventArgs exception)
{
    Console.WriteLine(exception.Exception.Message);
}
```

4..NET MAUI Android related question handling guide article

https://www.cnblogs.com/MASA/p/16612541.html

https://learn.microsoft.com/zh-cn/dotnet/maui/android/deployment/?view=net-maui-7.0

https://stackoverflow.com/questions/45940861/android-8-cleartext-http-traffic-not-permitted

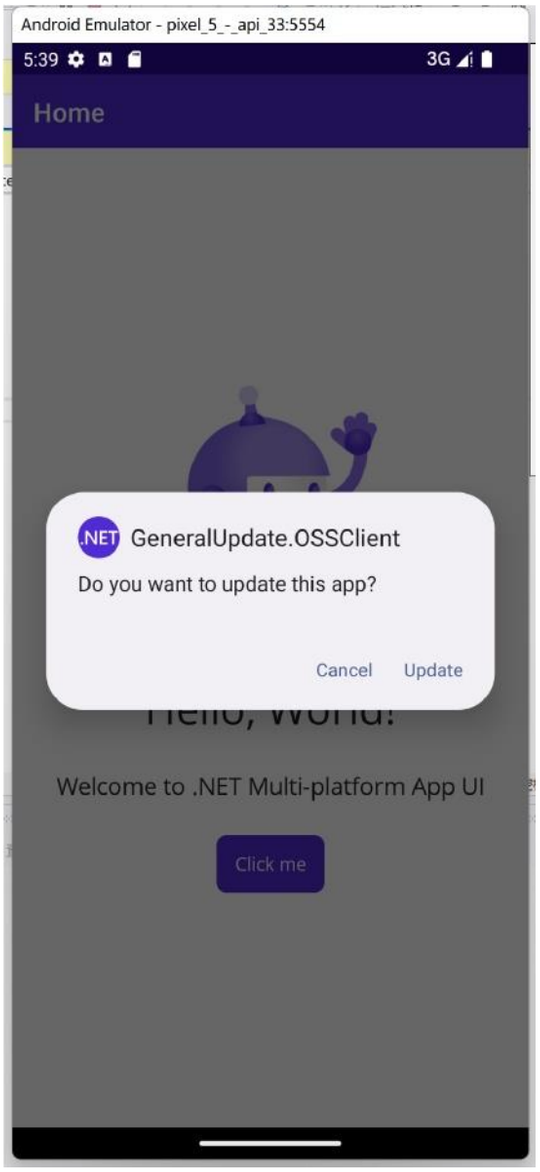
https://note.youdao.com/ynoteshare/mobile.html?id=5c5d5cf8fe1d67419b09024255ff239c

5 Run effect

At present, the test models and platforms have been run.

1. Huawei Honor Px30 non-HarmonyOS mobile phone can run.

2.Visual studio 2022 preview Pixe 5-API33 (Android 13.0-API 33) works.



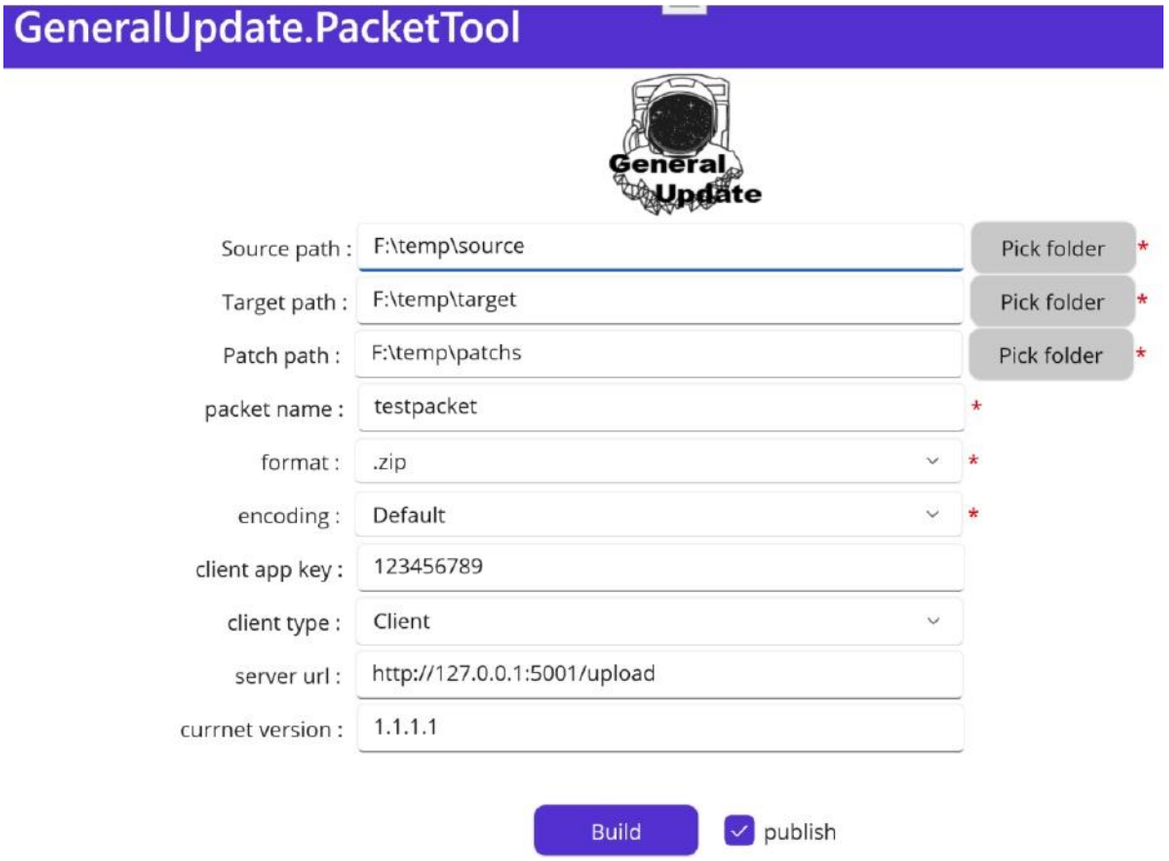
13. Quick start

Use the Flow

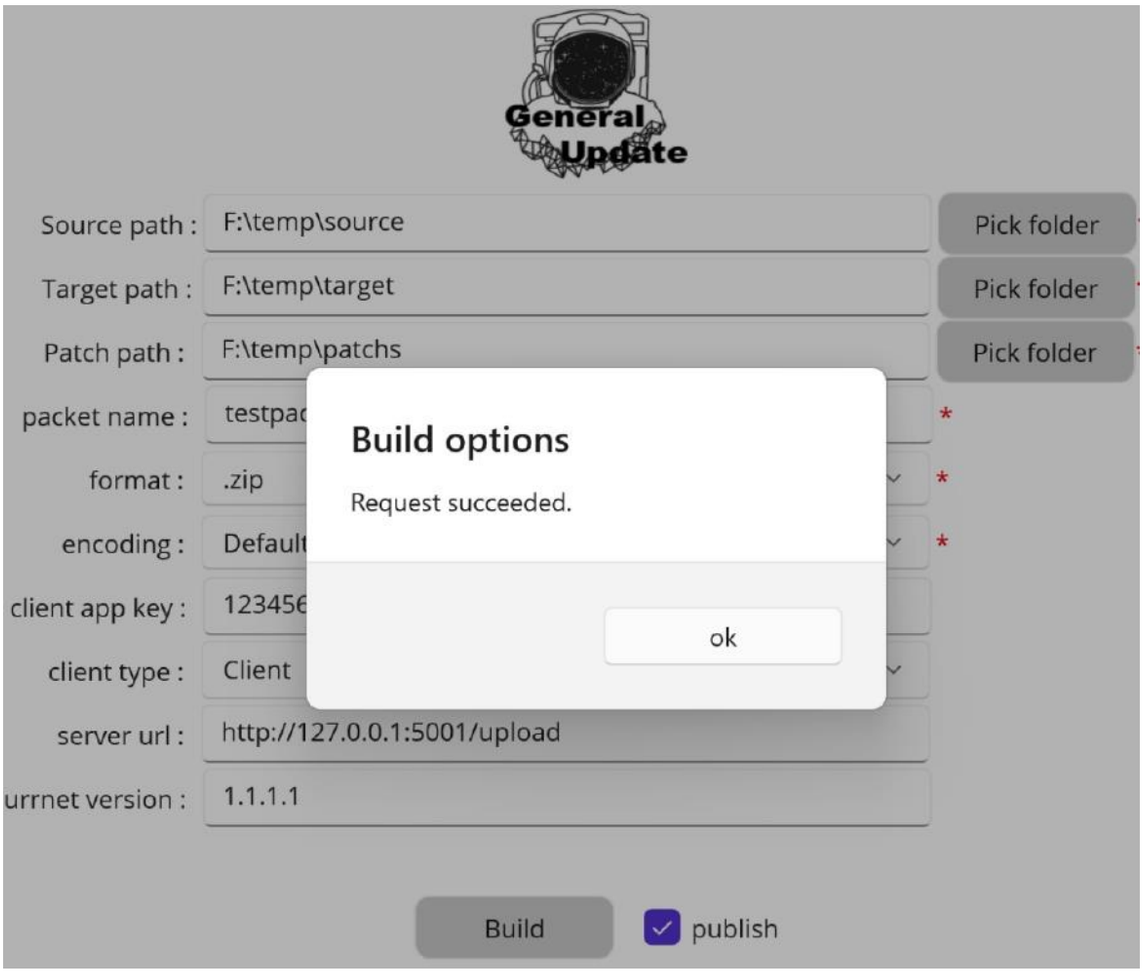
This section mainly explains how to use the GeneralUpdate component from 0. The point of this article is that you can use the GeneralUpdate component in the order you read it. (This article will not go into the details of the code because it has been described in other documentation. By default, you have built the basic code as required.)

1. Create an update package

Find PacketTool's project and just run it. (If not installed. NET MAUI environment partners need to install the environment in accordance with the relevant documents provided by this site). Continue to fill in the content required by the tool.



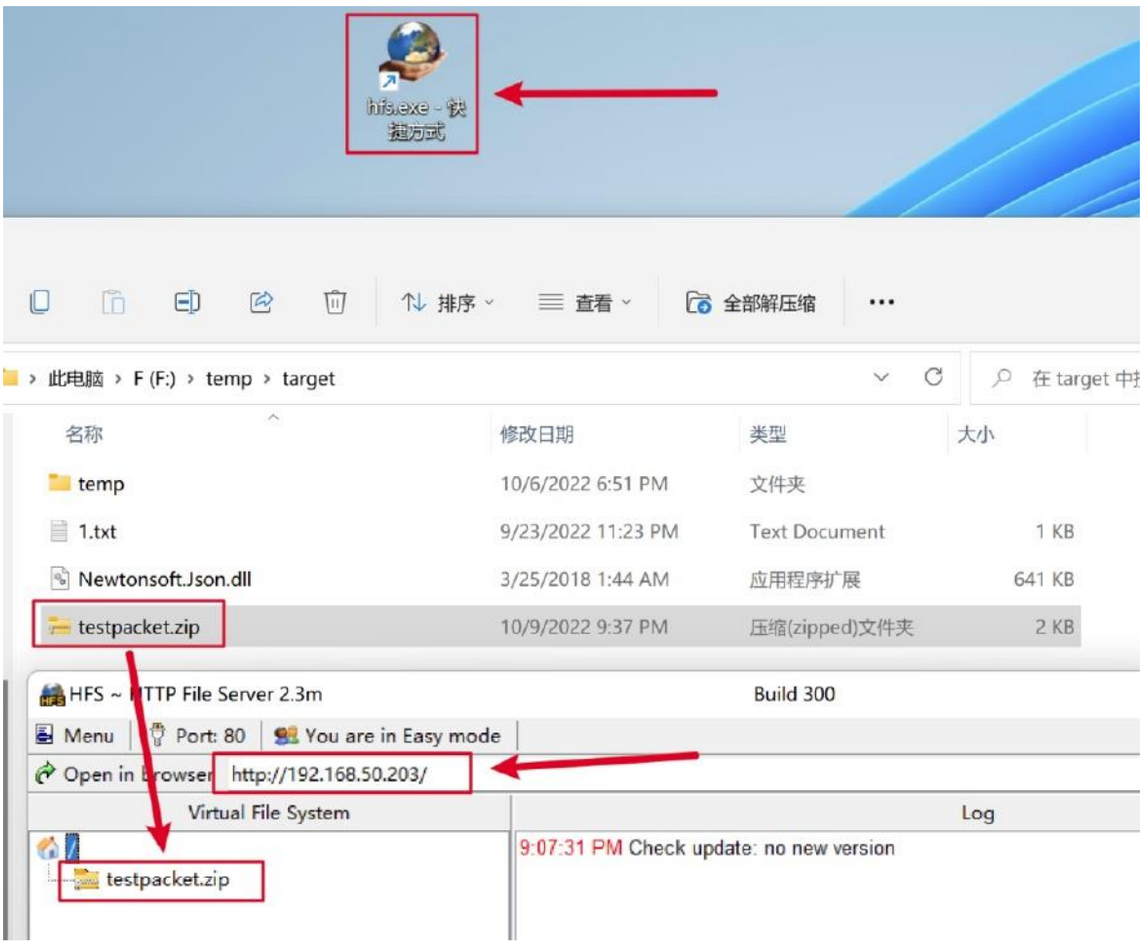
Click build. If you only need to update the package, do not fill in the content other than the red asterisk.



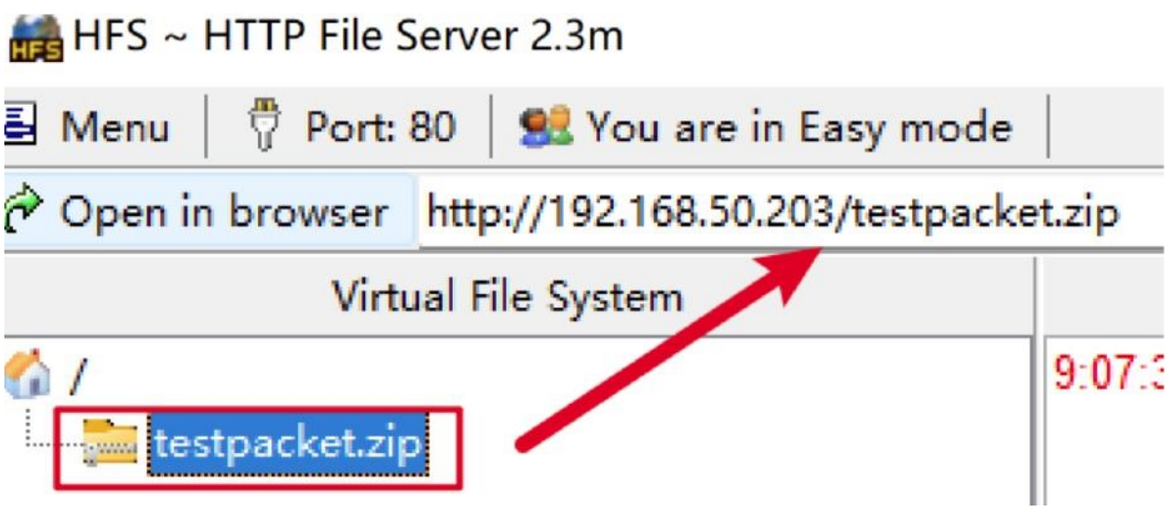
Then we get an update package.

2. Put the update package on the file server

Here I use the HFS tool to simulate the "file server", if it is a company production environment, please put it on the "file server". Find the content of the generated update pack and drag it to the file list on the left. http://192.168.50.203 is the address of the file server and you can change it to anything you want.



After you select the file, the HFS tool URL will change. This URL is the address we will provide for the client to download.



Next, you need to get the MD5 code of testpacket.zip. If you select publish and finish all the basic code construction of the server and write the information to the database, you can view the MD5 code of the file in the database. Or directly obtain the code from [src/c#/TestMD5/UnitTest1.cs · Juster.zhu/ GeneralUpdate-Gitee.com](), as follows:

```
"b03d52c279faf003965c46041f2037f9"
```

Getting MD5 is used in order to fake data without building a full server code, if there is a full server build can be skipped.

3. Server setup

The server side example code is GeneralUpdate.Api, at this point we find the code:

```
/*
*
 * Check if an update is required.
 *
 */
app.MapGet("/versions/{clientType}/{clientVersion}/{clientAppKey}", (int clientType, string clientVersion, string clientAppKey, IUpdateService updateService) =>
{
The client needs to use this to verify whether it is available
    var pubTime = new DateTimeOffset(DateTime.UtcNow).ToUnixTimeSeconds();
    string version = null;
    if (clientType == AppType.ClientApp)
    {
        //client
    }
    else if (clientType == AppType.UpgradeApp)
    {
        //upgrad
        //version = "0.0.0.0";
    }
    Var = $url "http://192.168.50.203/testpacket.zip"; // the download address of the update package
    var name = "testpacket";
    versions.Add(new VersionDTO(md5, pubTime, version, url, name));
GetAppSecretKey(), false, versions);
});
```

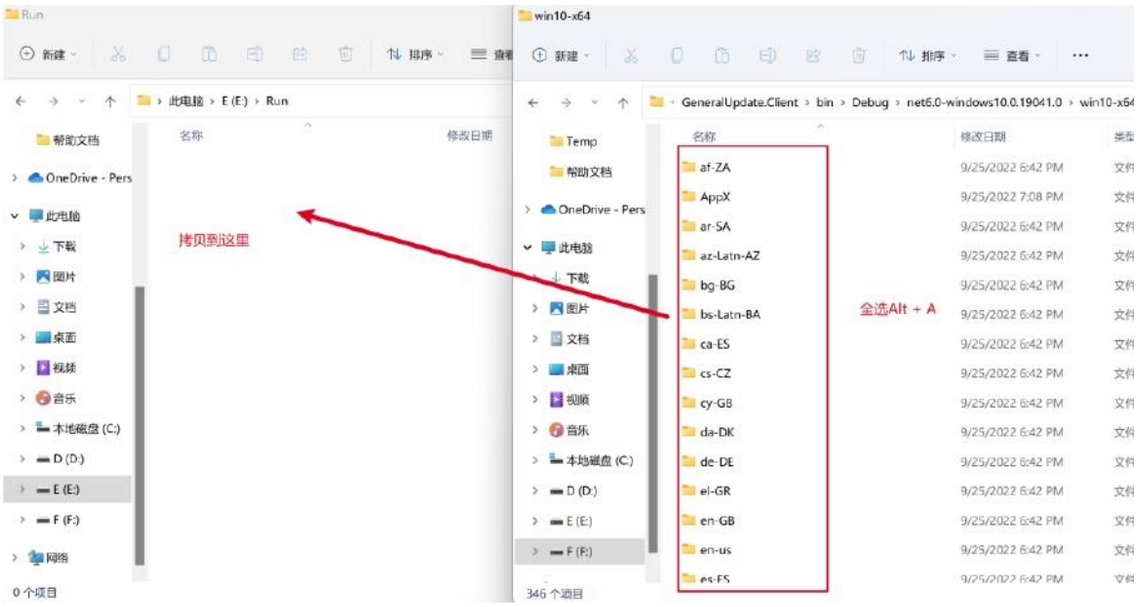
Then assign the MD5 value we just got, and the rest of the content can be faked according to the above code.

```
var md5 = "b03d52c279faf003965c46041f2037f9";
```

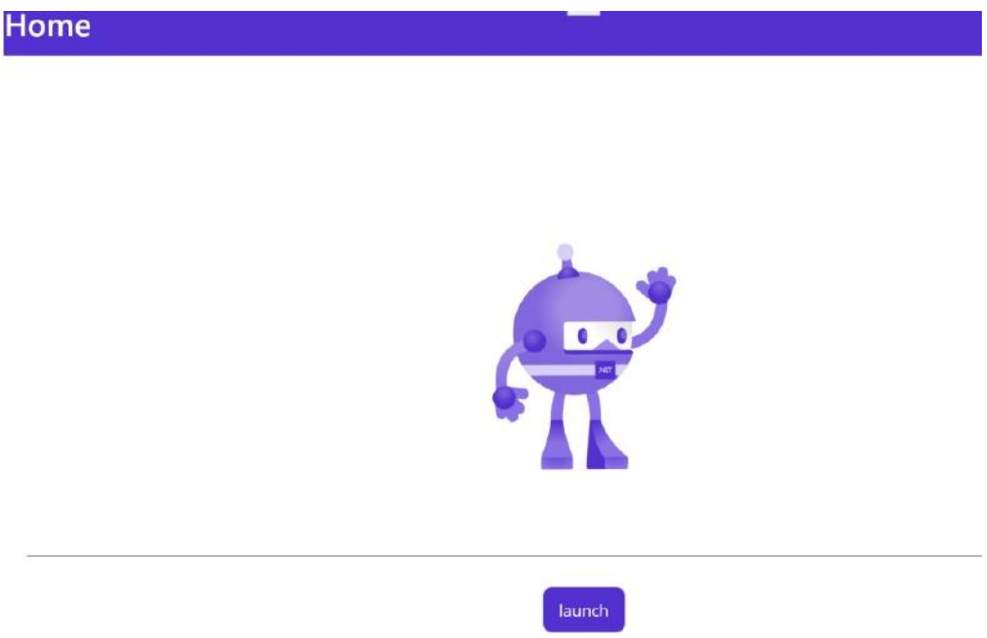
4. Client Settings

Start by creating an empty folder Run and copying the list of files in the Bin directory of both GeneralUpdate.Client and GeneralUpdate.Upgrad to Run. You need to check that the GeneralUpdate.Client has added labels before copying. Refer to: [.NET MAUI -- GeneralUpdate Doc (justerzhum.cn)]

This is after the copy is done.

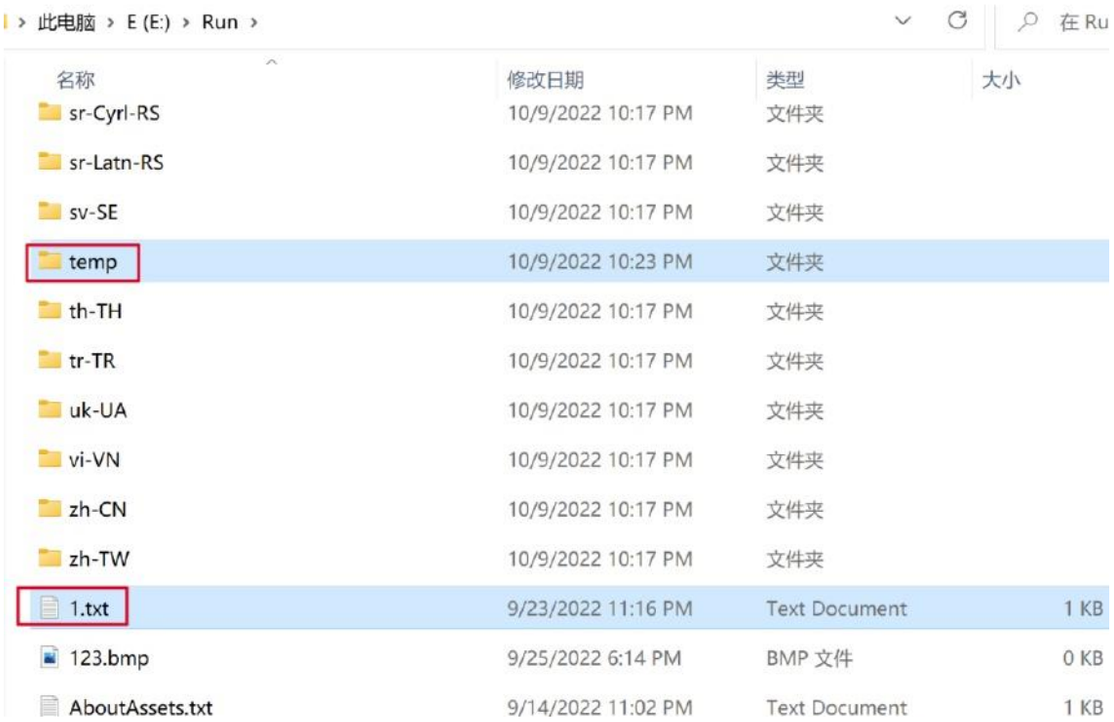


Next double click GeneralUpdate.Client.exe to run it and click launch (normal here is fine).

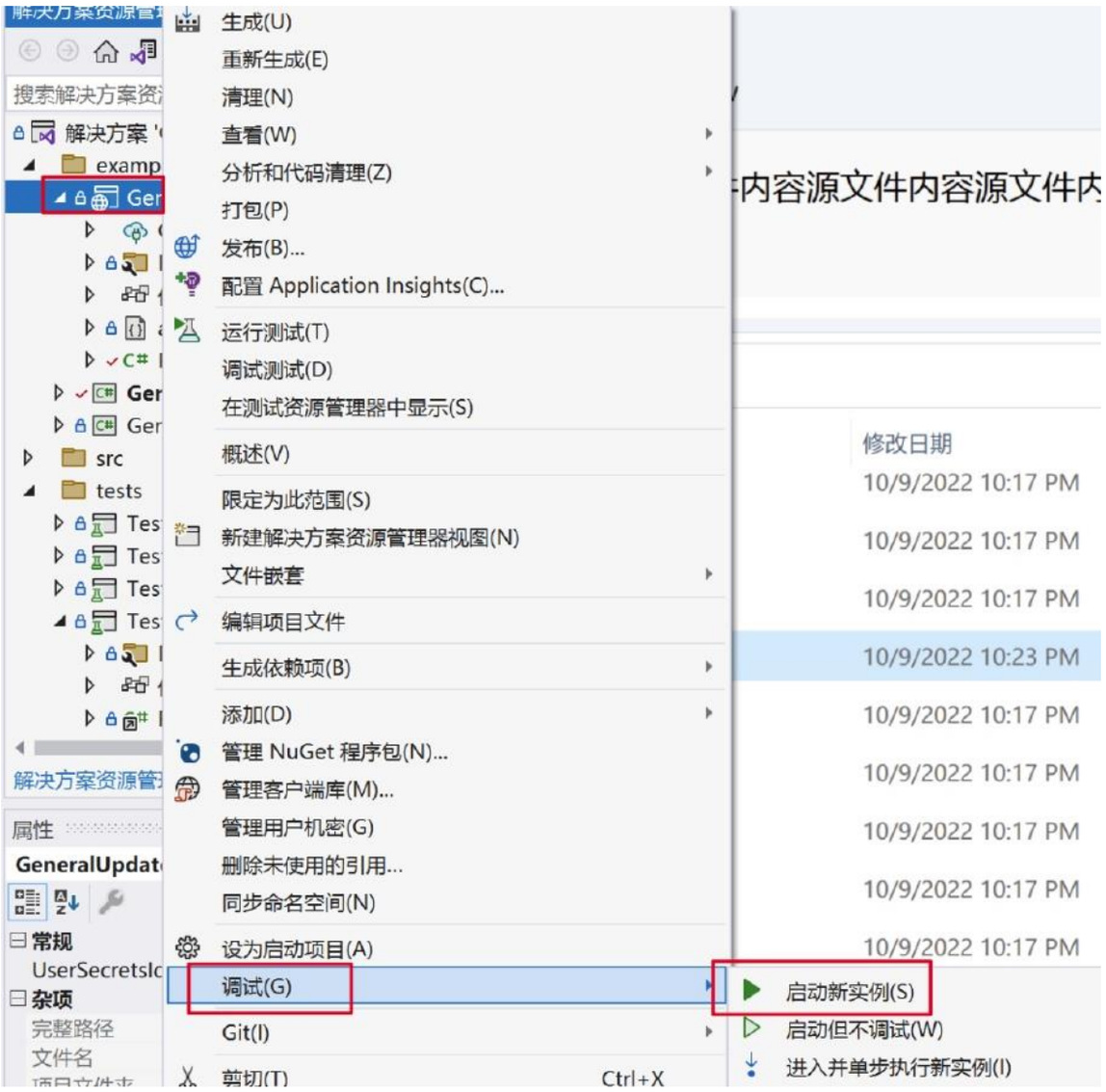


However, in order to let everyone feel the effect of the update, the content of the update package is packaged with some txt text files (the actual use can skip the following link, just to demonstrate the update effect).

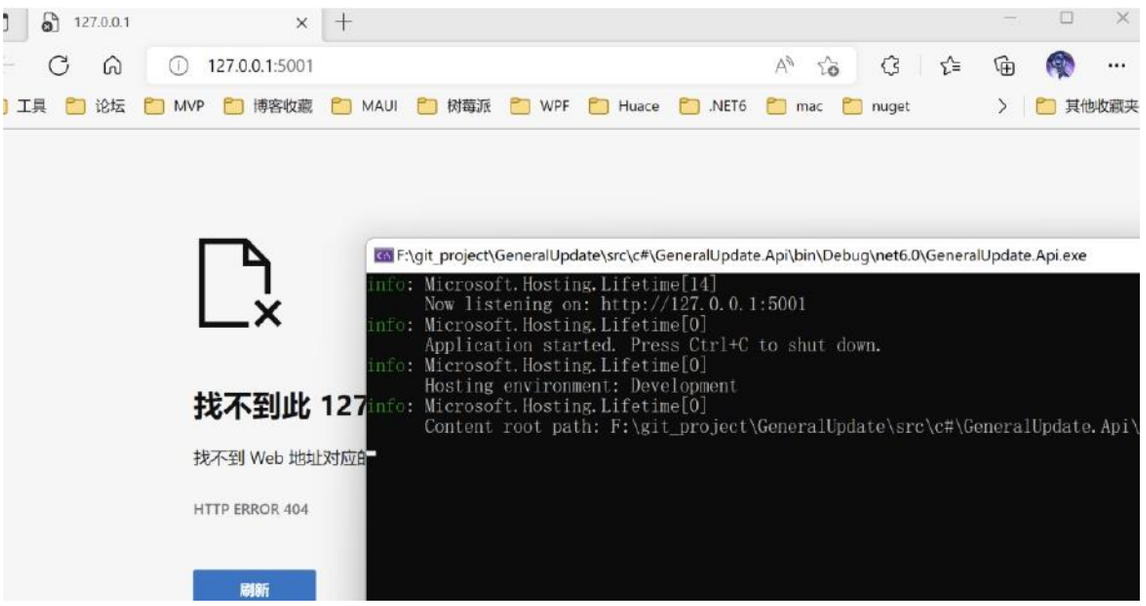
Also need to copy all the contents of the file here to the Run directory.



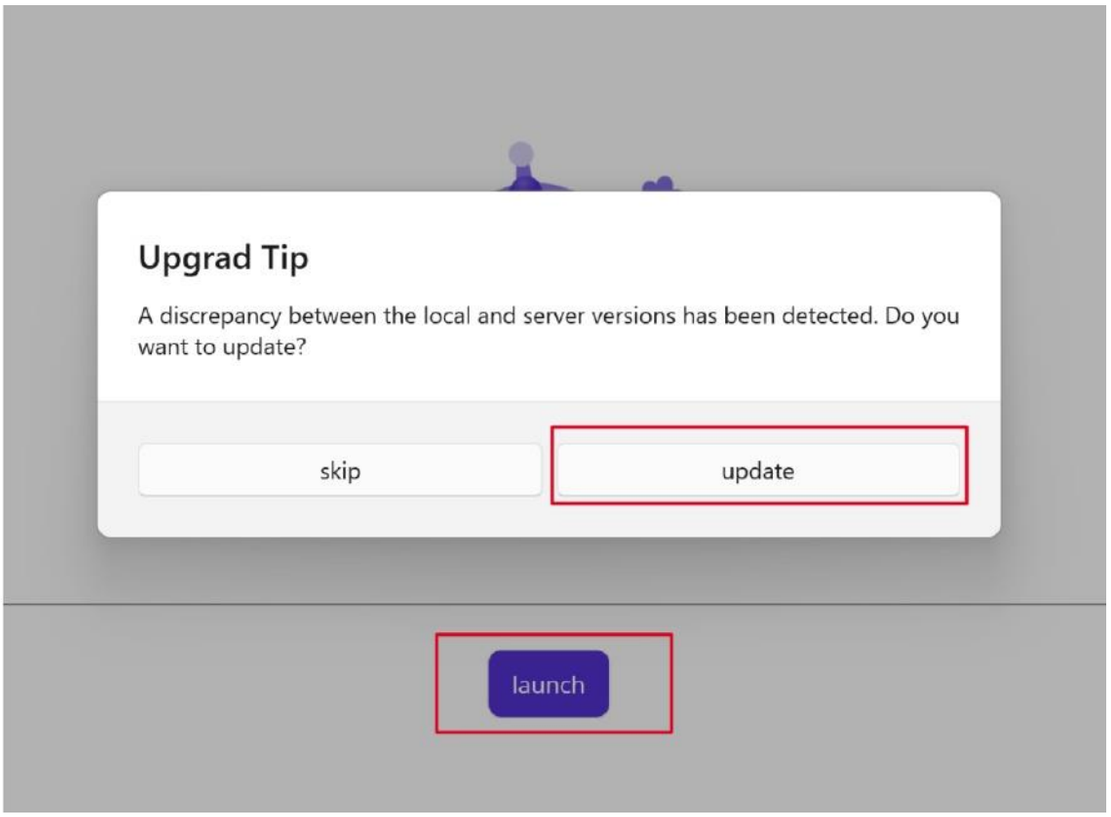
At this point, you can start the server.



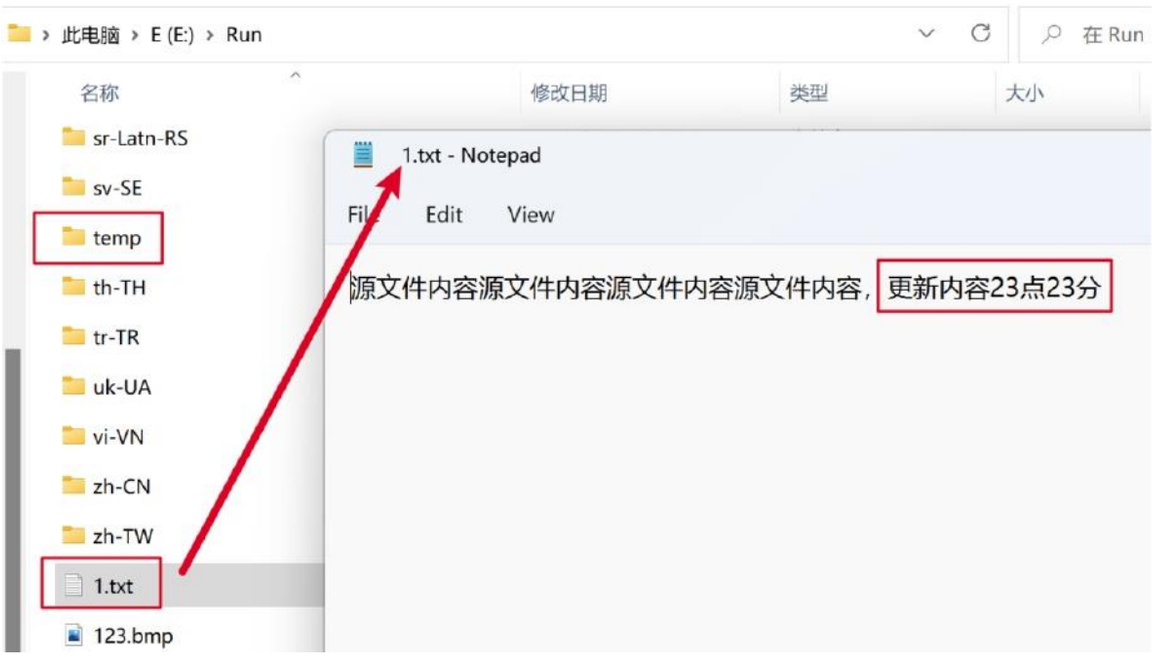
After a successful startup.



Click GeneralUpdate.Client.exe on launch.



As you can see from here, the GeneralUpdate.exe will be closed and launched. Here is the success of the update, we will verify the update effect next.



Then see if the contents of the multi-layer nesting of the temp folder have been updated.

The contents of the file can be seen to add some, see here
congratulations you learned to use the basics.

14.Q&A

1. A comprehensive Q&A reference

Address: [Issues · Juster.zhu/ GeneralUpdate-Gitee.com]

Keywords: C/S, WPF, MAUI, Winfrom, Avalonia, Console App, UWP, WinUI, Linux, Windows, MacOS, automatic update, automatic upgrade, update, push.