Protobuf Manager User Guide

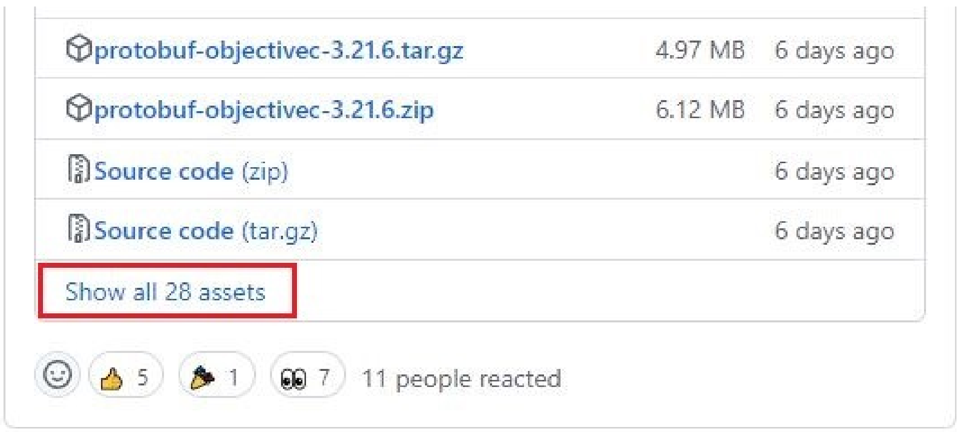
v1.1.2

1. Protobuf compiler integration

Due to the limitations of Unity Asset Store, this plugin doesn't include the Protobuf compiler, you need to complete the integration before starting to use. Don't worry, it's quite easy.

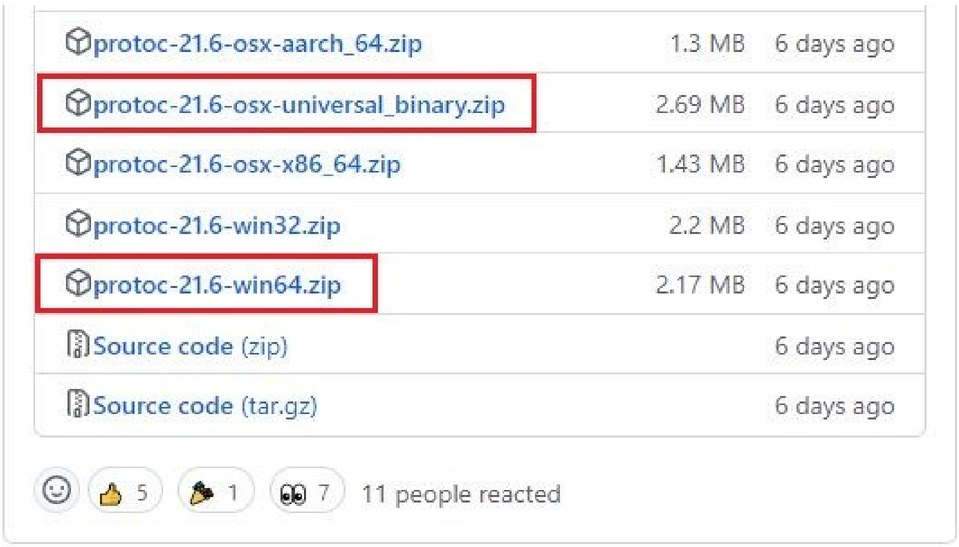
a) Open the compiler download page from "Tools - Protobuf Manager - Download Compiler".

b) For windows user, please download the "protoc-xxx-win64.zip". For mac user, please download the "protoc-xxx-osx-universal-binary.zip".



protobuf-objectivec-3.21.6.tar.gz	4.97 MB	6 days ago
protobuf-objectivec-3.21.6.zip	6.12 MB	6 days ago
Source code (zip)		6 days ago
Source code (tar.gz)		6 days ago
Show all 28 assets		

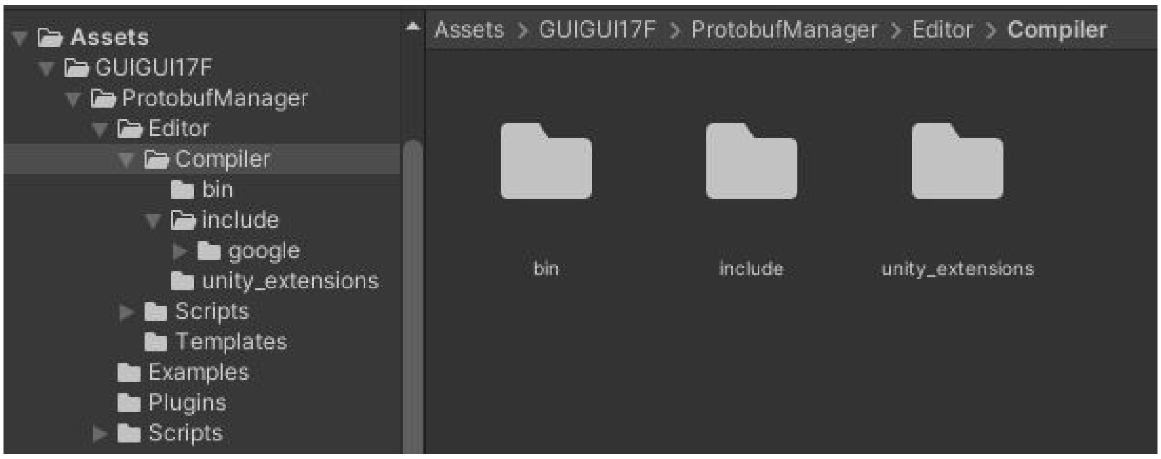
5 1 7 11 people reacted



protoc-21.6-osx-aarch_64.zip	1.3 MB	6 days ago
protoc-21.6-osx-universal_binary.zip	2.69 MB	6 days ago
protoc-21.6-osx-x86_64.zip	1.43 MB	6 days ago
protoc-21.6-win32.zip	2.2 MB	6 days ago
protoc-21.6-win64.zip	2.17 MB	6 days ago
Source code (zip)		6 days ago
Source code (tar.gz)		6 days ago

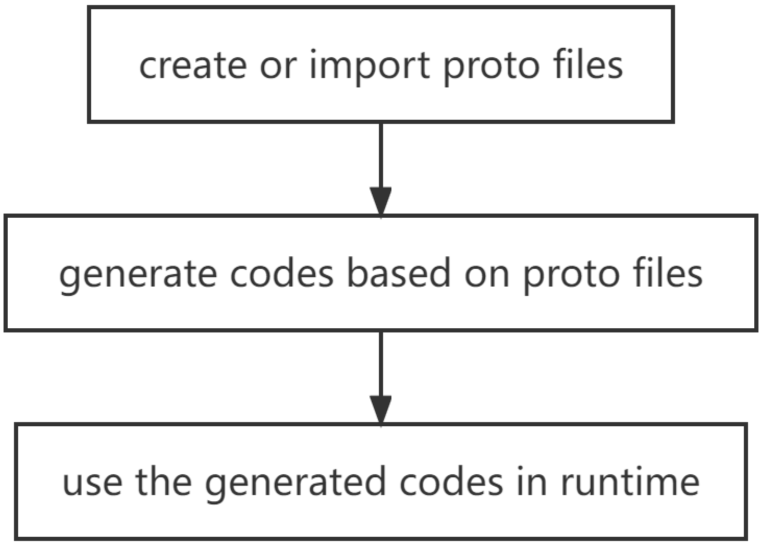
5 1 7 11 people reacted

c) Unzip the downloaded archive, put the “bin” and “include” folders into “ProtobufManager/Editor/Compiler”. The final result should be like this:

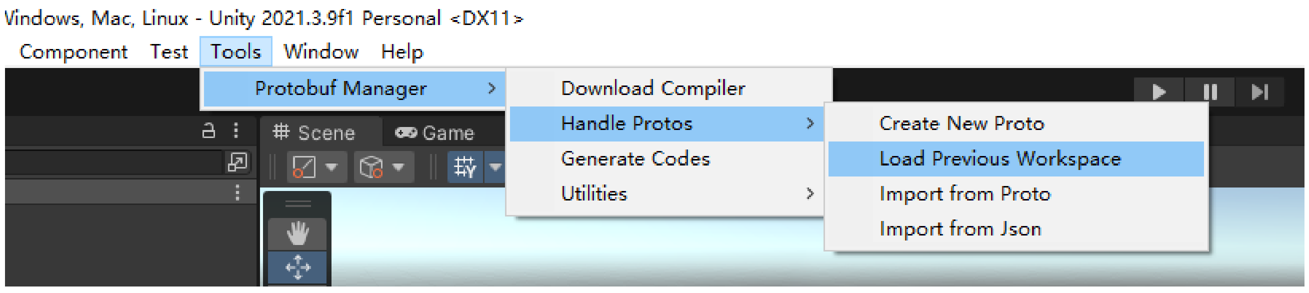


d) If the version of the compiler you downloaded is ahead of the Protobuf runtime library contains in this plugin, the generated codes may report errors during script compiling. Please check the FAQ to see how to solve this.

2. General Work Flow


create or import proto files
generate codes based on proto files
use the generated codes in runtime

3. Menu Bar



- **Download Compiler**: Open the Protobuf compiler download page.
- **Handle Protos – Create New Proto**: Open the Protobuf Edit Window and start to create a proto file from the beginning. See section 2 for the details about Protobuf Edit Window.
- **Handle Protos – Load Previous Workspace**: Open the Protobuf Edit Window and load the last cached workspace.
- **Handle Protos – Import from Proto**: Analyze the given proto file and import it as the Protobuf Edit Window workspace.

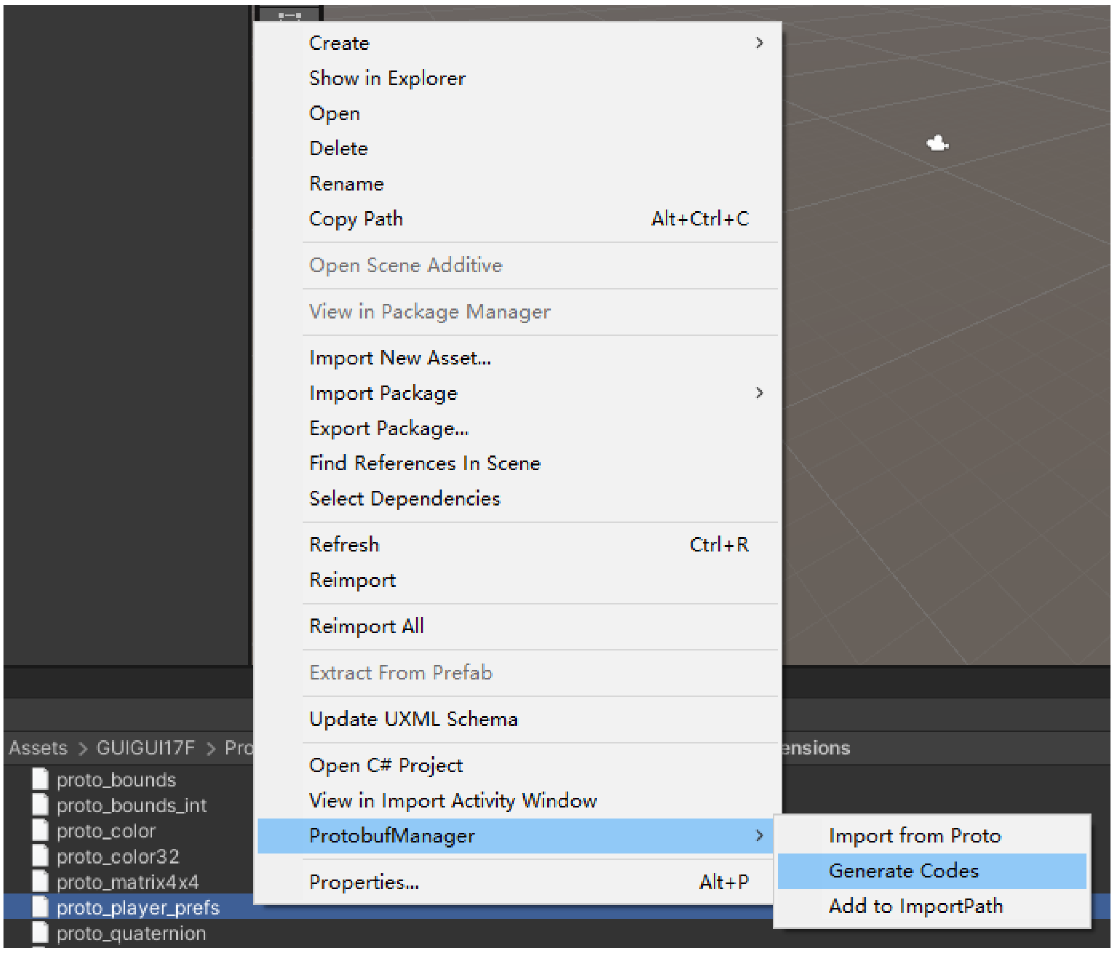
  Please notice this feature is only designed for importing protos created by Protobuf Manager, importing other protos may lead to loss of information.
- **Handle Protos – Import from Json**: Analyze the given json text and import it as the Protobuf Edit Window workspace.
- **Generate Codes**: Open the Code Generation Window and start to generate your codes with selected proto files. See section 3 for the details about Code Generation Window.
- **Utilities - Manage ImportPath**: Open the ImportPath Edit Window to review and edit the ImportPath settings for Protobuf compiler. ImportPath specifies a directory in which to look for proto files when resolving the “import” directive in a proto file,

see the Protobuf manual for more information

(https://developers.google.com/protocol-buffers/docs/proto3#importing_definitions).

- **Utilities - Open PersistentDataPath**: Open the persistentDataPath folder of current platform.

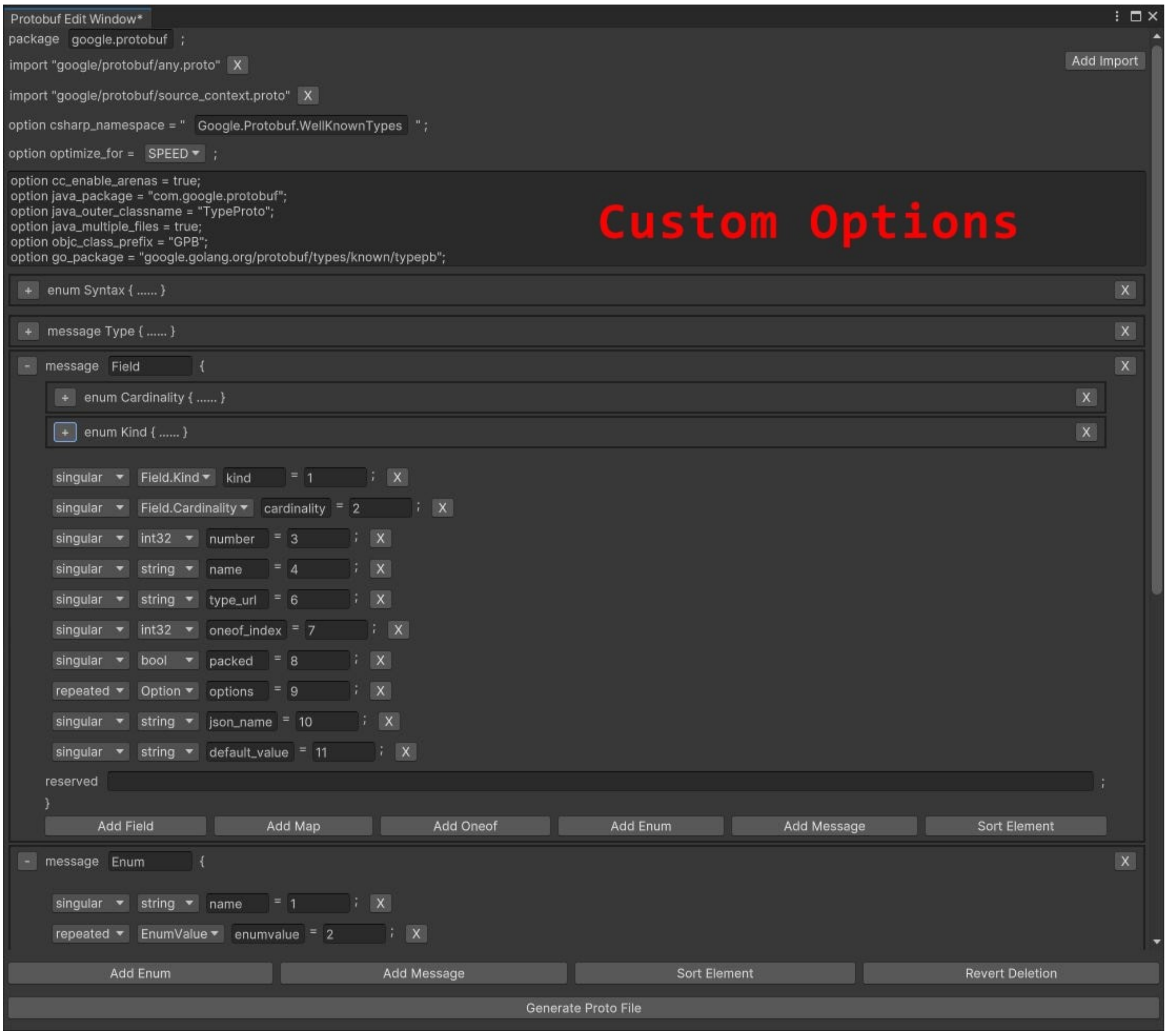
4. Assets Menu



- **Import from Proto**: Analyze the selected proto file and import it as the Protobuf Edit Window workspace.
- **Generate Codes**: Open the Code Generation Window and import the selected proto file as the generation source. If the selection is a folder, all proto files under this folder will be imported.

◆ **Add to ImportPath**: Open the ImportPath Edit Window and add the selected folder into the ImportPath list.

5. Protobuf Edit Window



```
Protobuf Edit Window*
package google.protobuf ;
import "google/protobuf/any.proto" X                                  Add Import
import "google/protobuf/source_context.proto" X
option csharp_namespace = " Google.Protobuf.WellKnownTypes " ;
option optimize_for = SPEED ;
option cc_enable_arenas = true;
option java_package = "com.google.protobuf";
option java_outer_classname = "TypeProto";
option java_multiple_files = true;
option objc_class_prefix = "GPB";
option go_package = "google.golang.org/protobuf/types/known/typepb";
                                                       Custom Options
+ enum Syntax { ...... }                                             X
+ message Type { ...... }                                            X
- message Field {                                                    X
    + enum Cardinality { ...... }                                    X
    + enum Kind { ...... }                                           X
    singular  Field.Kind         kind          = 1  ; X
    singular  Field.Cardinality  cardinality   = 2  ; X
    singular  int32              number        = 3  ; X
    singular  string             name          = 4  ; X
    singular  string             type_url      = 6  ; X
    singular  int32              oneof_index   = 7  ; X
    singular  bool               packed        = 8  ; X
    repeated  Option             options       = 9  ; X
    singular  string             json_name     = 10 ; X
    singular  string             default_value = 11 ; X
  reserved                                                           ;
  }
  Add Field  Add Map  Add Oneof  Add Enum  Add Message  Sort Element
- message Enum {                                                     X
    singular  string     name      = 1 ; X
    repeated  EnumValue  enumvalue = 2 ; X
Add Enum  Add Message  Sort Element  Revert Deletion
Generate Proto File
```

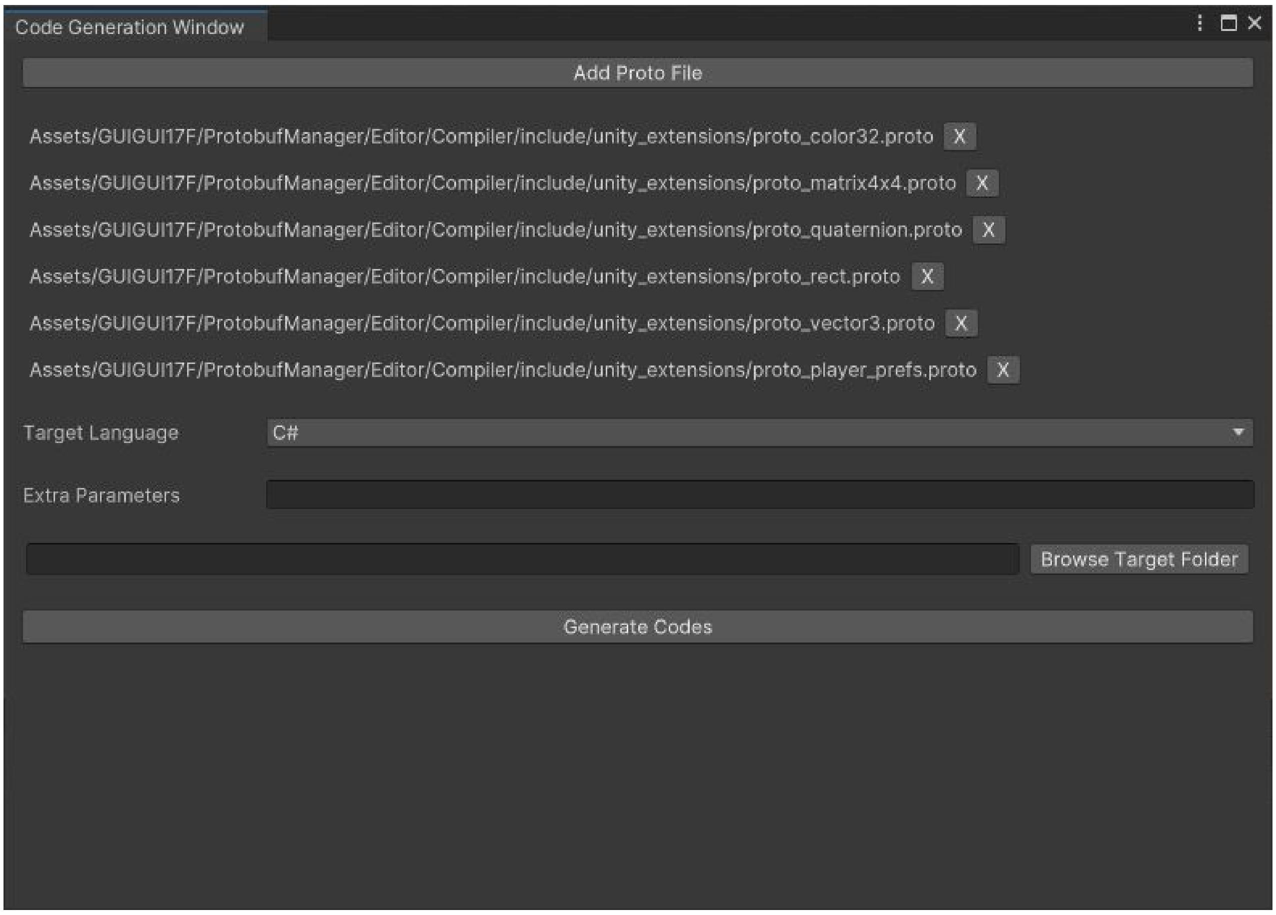
Use this window to create and edit a proto file. The visual elements are organized based on the Protobuf grammar, so you may find it's similar to the proto file itself, excepting more convenient.

The plugin already provides support for some most common options, such as C# namespace and optimize type. For other custom options, please fill them in the custom options field.

When you closing this window, you can choose to save current workspace as a cache, then you can continue your work by “Load Previous Workspace” later. Generating a proto file successful will auto save current workspace as well.

A feature worth to notice is the “Revert Deletion”. To make the work flow smooth, delete operation won’t ask for confirm. And if you delete anything by a mistake, just click the “Revert Deletion” to bring it back.

6. Code Generation Window



Assets/GUIGUI17F/ProtobufManager/Editor/Compiler/include/unity_extensions/proto_color32.proto
Assets/GUIGUI17F/ProtobufManager/Editor/Compiler/include/unity_extensions/proto_matrix4x4.proto
Assets/GUIGUI17F/ProtobufManager/Editor/Compiler/include/unity_extensions/proto_quaternion.proto
Assets/GUIGUI17F/ProtobufManager/Editor/Compiler/include/unity_extensions/proto_rect.proto
Assets/GUIGUI17F/ProtobufManager/Editor/Compiler/include/unity_extensions/proto_vector3.proto
Assets/GUIGUI17F/ProtobufManager/Editor/Compiler/include/unity_extensions/proto_player_prefs.proto

Target Language: C#

Extra Parameters

Use this window to generate the actual codes based on proto files.

Please notice the Protobuf compiler uses the ImportPath list in the ImportPath Edit Window as input parameters. Every proto file generated by Protobuf Manager use the project “Assets” folder as its ImportPath, if your proto file requires a custom one, make sure to add it in the ImportPath Edit Window (Tools - Utilities - Manage ImportPath).

7.Unity Extensions

The Unity extensions include two parts, one is the proto files under ProtobufManager/Editor/Compiler/include/unity_extensions, the other is the scripts under ProtobufManager/Scripts/UnityExtensions. The proto files are descriptions of many Unity built-in types, which you can import and use in your own proto files. The scripts are generated based on these proto files to provide runtime support for your generated codes.

There are other scripts under ProtobufManager/Scripts. The two “Extensions” scripts contain extension methods used for parse the Protobuf types to Unity built-in types and reverse. The ProtoPlayerPrefsWrapper is a simple custom data cache just like the Unity built-in one.

8. Examples

The example scene and scripts are under ProtobufManager/Examples. They demonstrate some simple usage of Protobuf generated codes, including read/write data from/to an array/file, and how to add custom logic such as encryption during serialization.

9. FAQ

Q: I downloaded the latest protoc compiler, but there are some errors in the generated codes.

A: It could be because the generation algorithm in the protoc compiler has upgraded, but we are still using the old version of the Protobuf runtime library. Follow the instruction below to upgrade it.

a) Open the download page from “Tools - Protobuf Manager - Download Compiler”.

b) Download the source code archive. The file name is “protobuf-XXX.zip” or “protobuf-XXX.tar.gz”, depends on which format you prefer.

c) Unzip the downloaded archive, refer to protobuf-XXX/csharp/src/Google.Protobuf.

d) Delete everything under the ProtobufManager/Scripts/Google.Protobuf folder **except the Google.Protobuf.asmdef** file. Some classes in the old library may conflict with the new version, so you do need to delete them first.

e) Copy every file and folder under protobuf-XXX/csharp/src/Google.Protobuf to ProtobufManager/Scripts/Google.Protobuf, the upgrade has done.

Q: After I changed the locations of my proto files, errors start to occur. How can I fix that?

A: There are two ways to synchronize the path changes. One is to open your proto files with any text editor, and update the “import” directives to new paths. But this isn’t the official recommended operation.

The recommended operation is to use the “import public” feature, see the Protobuf manual for details (https://developers.google.com/protocol-buffers/docs/proto3#importing_definitions).

Q: My generation process failed. How can I check what went wrong?

A: Every time the Protobuf Manager trying to generate some codes, the command line parameters passed to the Protobuf compiler will be printed into the Unity Editor console. If the process failed with any error message, the message will be printed as well. These logs will help you in most situations.

If you meet a failure when importing a proto file, you can try to uncomment the Debug.Log() in ProtoCompilerUtility line 155. This log may give you more information.

Q: Is Protobuf Manager support gRPC?

A: The support for gRPC in Unity engine is a little complicated. On one hand, the Grpc.Core implementation of gRPC for C# is in maintenance mode, and will be replaced by grpc-dotnet in the future. On the other hand, the grpc-dotnet needs the HTTP/2 feature which are not fully supported in most Unity Editor versions. So, it's recommended to use the Protobuf to generate binary data first, then pass them through a more suitable system.

But if you still want to give it a try, there's still a way. Please download the gRPC daily build package for Unity (https://packages.grpc.io/archive/2022/04/67538122780f8a081c774b66884289335c290cbe-f15a2c1c-582b-4c51-acf2-ab6d711d2c59/csharp/grpc_unity_package.2.47.0-dev202204190851.zip), and put the "Plugins" folder into your project. Please notice only the "Grpc.Core" and "Grpc.Core.Api" folders in the "Plugins" are needed, the assemblies in other folders are already included in the Protobuf Manager.

Although you can't create a service proto file using the Protobuf Edit Window, the Code Generation Window is still working. You can write your own service proto files and use the Code Generation Window to generate your service codes.

Q: Should I use encryption for the Protobuf serialization data?

A: It depends on your use cases. Technically, a Protobuf serialization data is a binary stream, which is hard to analyze without its origin proto file by nature. So, it’s no need to encrypt every data, as the encryption and decryption process take up extra CPU time and memories.

On the other hand, if your data needs a higher security, such as you need store passwords or financial data in it, then use encryption will be a more suitable choice.

10. Version Information

Protobuf Language Version: proto3