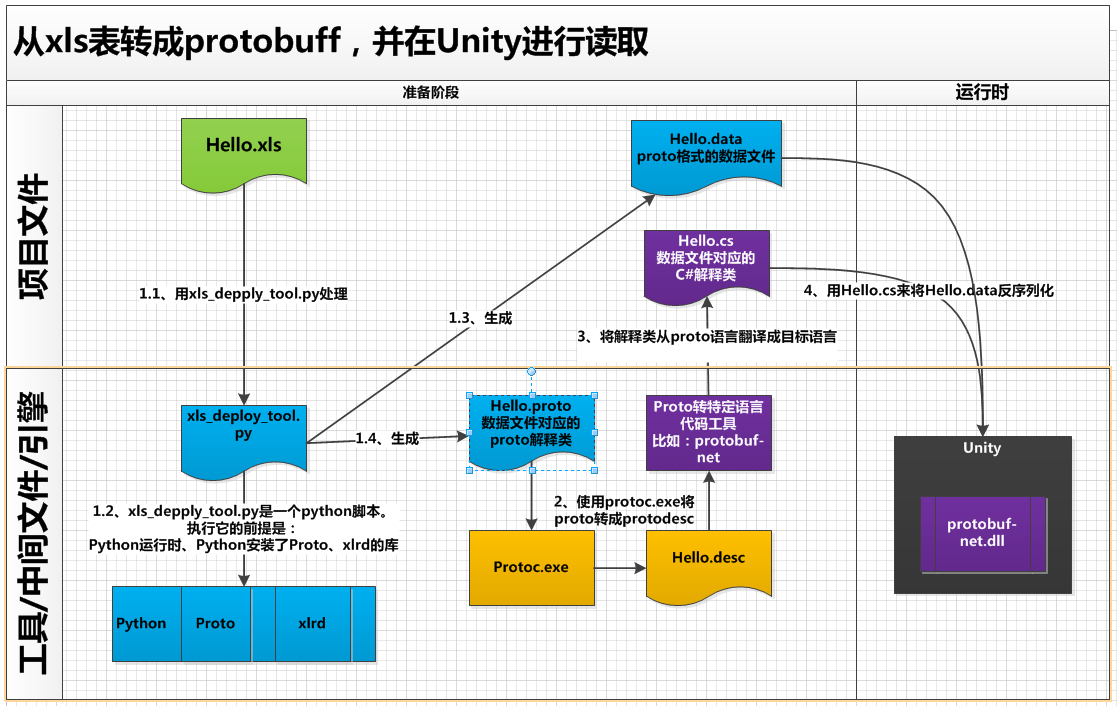
**将xls转成protobuf供Unity使用的流程**

**通用流程图**



*流程图*

**通用流程图简述**

1. 在python运行时下，依赖proto组件、xlrd的组件，使用xls\_deploy\_tool.py处理Hello.xls，生成Hello.bin数据文件、及其对应的Hello.proto解释类。
2. 在windows系统下，使用protoc.exe，将proto解释类转成中间格式Hello.desc。
3. 将中间格式Hello.desc用语言工具翻译成其他语言解释类比如c#解释类Hello.cs。
4. 游戏运行时引入protobuf的dll，使用c#解释类对data数据文件进行读取。

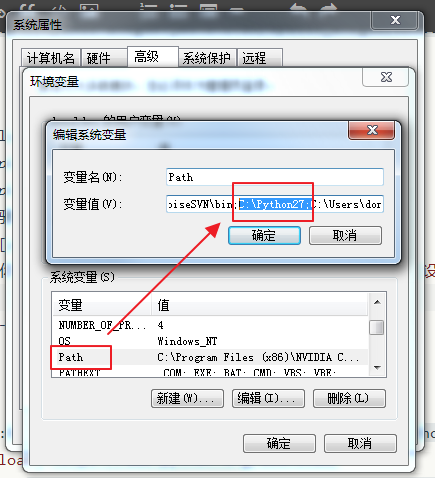
**Killer项目使用步骤**

**目录解释**

* 配置软件安装目录：KillerTool\DataConfigSetup。
* 配置表及工具目录：KillerProject\DataConfigBuild：
  + 配置表目录：KillerProject\DataConfigBuild\DataConfig。所有xls都在这里。它们是从策划SVN库里的DataConfig外链到这里的
* 导出数据目录：KillerProject\Assets\StreamingAssets\DataConfig
* 导出代码目录：KillerProject\Assets\Scripts\Killer\DataConfig\ProtoGen

**环境准备步骤**

* 进入配置软件安装目录：KillerTool\DataConfigSetup
* 如果你的本地电脑没安装python及其组件
  + 视乎你是32位还是64位操作系统，安装python-2.7.7.32.msi或python-2.7.7.amd64.msi。
  + 把你刚刚选择安装python的路径添加到本地[环境变量](http://baike.baidu.com/link?url=K46w-M08qur7chI9ghvDAhXcvhQjv0I-XM2HUYoMZ2DL_COL29g819IwHMKv1ntT#3)的path中。如下图



*环境变量path*

* + 命令行进入目录setuptools-5.1，执行命令python setup.py install
  + 命令行进入protobuf-2.5.0\python目录，执行命令python setup.py install
  + 命令行进入xlrd-0.9.3目录，执行命令python setup.py install

环境准备步骤到此结束。

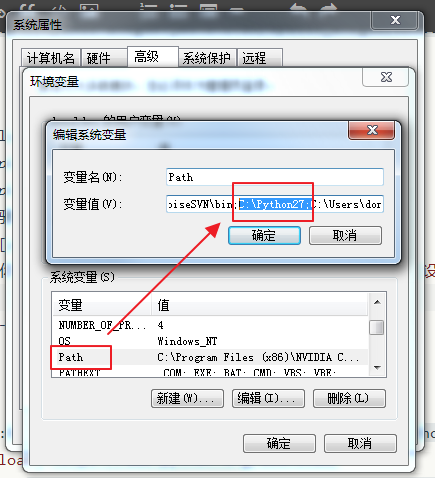
**使用方法**

* 如果你（一般是策划同学或程序同学）是修改已有xls文件，修改保存xls后，运行xls\_all.bat，并上传导出数据目录里受到修改的bin文件
* 如果你（一般是程序同学）是要添加新xls文件
  + 在配置表目录添加你的xls文件（注意xls文件头格式需要符合[格式](https://github.com/jameyli/tnt/blob/master/python/xls_deploy_tool.py)）
  + 修改xlsc\_all.bat，新增一行call xlsc.bat <表格文件名> <表格名>，比如call xlsc.bat gm GM\_CONF
  + 修改后运行xlsc\_all.bat
  + 上传导出数据目录里新增的bin文件
  + 上传导出代码目录里新增的cs文件
  + 修改导出代码目录里的DataConfigManager.cs文件，使用cs文件对对应的data文件进行读取。方法可参考那里已有的代码。
* 如果你（一般是程序同学）要删除已有xls文件
  + 请删除xlsc\_all.bat中对应的脚本代码
  + 请删除导出数据目录里对应的bin文件
  + 请删除导出代码目录里对应的cs文件和DataConfigManager.cs里对应的解释代码

**通用项目准备步骤**

如果你准备搭建新的项目环境，或者对环境准备的方法感兴趣，可继续阅读下面的通用项目准备步骤。

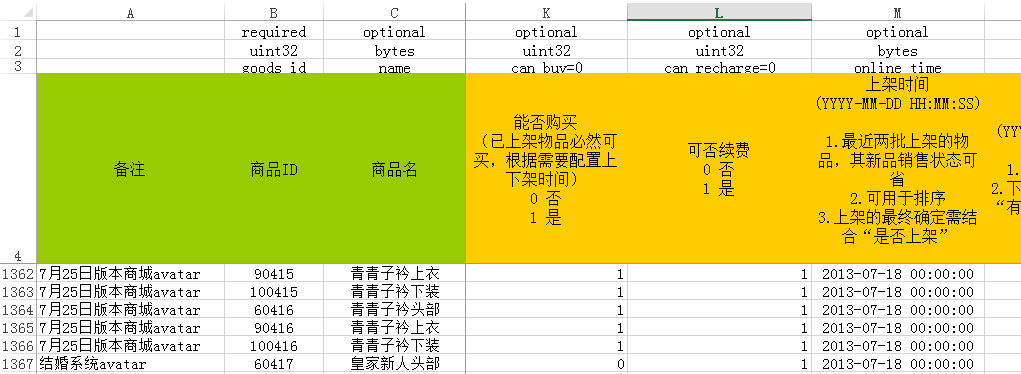
1. 准备[xls\_deploy\_tool.py](https://github.com/jameyli/tnt/tree/master/python)工具环境。xls\_deploy\_tool.py是一个python脚本，运行在python环境，并且python要安装了proto组件和xlrd组件
   * 安装python运行时
     + 在[python官网](https://www.python.org/download/releases/2.7.7/)下载并安装python2.7（不能装最新的python3，因为python3不向下兼容，而protobuff是用python2的）。
     + 把你刚刚选择安装python的路径添加到本地[环境变量](http://baike.baidu.com/link?url=K46w-M08qur7chI9ghvDAhXcvhQjv0I-XM2HUYoMZ2DL_COL29g819IwHMKv1ntT#3)的path中。如下图



*环境变量path*

* + 安装setuptool
    - python安装protobuff之前，需要先安装setuptool。setuptool类似于python的一个组件安装管理器
    - 在[setuptool官网](https://pypi.python.org/pypi/setuptools#downloads)的最下面点击下载安装文件，这里我们可以安装setuptools-5.1.zip。解压它。
    - 用命令行进入刚解压的setuptool目录，执行命令python setup.py install
  + 安装protobuff
    - 我们目前使用2.5.0的protobuff版本
    - 在[protobuff官网](https://code.google.com/p/protobuf/downloads/detail?name=protobuf-2.5.0.tar.bz2)下载protobuff的“全文件压缩包”（protobuf-2.5.0.tar.bz2）和“编译器压缩包”（protoc-2.5.0-win32.zip）
    - 将“全文件压缩包”解压
    - 在里面创建compiler文件夹，即protobuf-2.5.0\python\google\protobuf\compiler
    - 将“编译器压缩包”里的protoc.exe拷到protobuf-2.5.0\src
    - 用命令行进入刚解压的protobuf\python目录，执行命令python setup.py install
  + 安装xlrd（xls reader）
    - 在[xlrd官网](https://pypi.python.org/pypi/xlrd)下载并解压
    - 用命令行进入刚解压的xlrd目录，执行命令python setup.py install

1. xls\_deploy\_tool.py
   * [xls\_deploy\_tool.py](https://github.com/jameyli/tnt/tree/master/python)是腾讯魔方工作室[jameyli](https://github.com/jameyli)同学的作品
   * 使用方法是在命令行键入如下命令
     + python xls\_deploy\_tool.py <表格名> <xls文件名>
   * 它将“符合格式”的xls文件生成data文件和proto文件
   * “符合格式”指的是要求xls的每一列的前4行用于定义数据格式。例子如图：



* + - 详细格式文档可参考xls\_deploy\_tool.py 的文件注释。

1. xls\_deploy\_tool.py生成的文件
   * 生成的data文件是最终在游戏内进行读取的数据文件，以protobuff定义的格式进行压缩存储（或者明文存储，但xls\_deploy\_tool.py暂时只实现了压缩存储）
   * 生成的proto文件是用于解释上面这个data文件的解释类。
     + 但由于我们需要在我们游戏内对其进行读取，而游戏往往只能运行其他高级语言，所以需要将proto翻译成对应的语言
     + 这里以C#为例
2. 将proto语言翻译成C#语言
   * 需要分两步走：
     + 使用“编译器压缩包”（比如protoc-2.5.0-win32.zip）的protoc.exe将proto解释成“FileDescriptorSet ”中间格式
     + 使用[protobuf-net](https://code.google.com/p/protobuf-net/)的protogen.exe将“FileDescriptorSet ”翻译成cs文件
   * protoc.exe
     + 使用先前“编译器压缩包”（protoc-2.5.0-win32.zip）里的protoc.exe，命令行命令大概是
   * protogen.exe
     + 下载、解压[protobuf-net](https://code.google.com/p/protobuf-net/)
     + ProtoGen目录都需要用到，其中的protogen.exe负责将proto文件翻译成cs代码。
3. 在Unity读取data文件，并且用cs代码解释
   * 到[protobuf-net的github网站](https://github.com/mgravell/protobuf-net)，下载工程的zip包，解压，把里面的protobuf-net文件夹放置在Unity工程的Assets\Plugins目录下

注：不能使用protobuf-net\Full\unity\下的protobuf-net.dll拷到Unity工程的Assets\Plugins目录下的方法。  
因为部署到iOS设备，会发生JIT错误（ExecutionEngineException: Attempting to JIT compile method）。原因是因为iOS不允许JIT（Just In Time），只允许AOT（Ahead Of Time）。  
解决这个问题的方法之一是使用protobuf-net的precompile工具，但研究发现，过程会比较麻烦。具体可参阅precompile的help命令，或者谷歌“precompile protobuf net”。  
解决这个问题的方法之二，也是比较简单、证明过可行的，就是直接把源代码代替protobuf-net.dll，拷到Unity工程的Assets\Plugins目录下。

* + 引用导出的cs类，using ProtoBuf的命名空间，使用Serailizer.Deserialize<T>()全局函数，进行数据的解释，关键代码如下
  + private T ReadOneDataConfig<T>(string FileName)
  + {
  + FileStream fileStream;
  + fileStream = GetDataFileStream(FileName);
  + if (null != fileStream)
  + {
  + T t = Serializer.Deserialize<T>(fileStream);
  + fileStream.Close();
  + return t;
  + }
  + return default(T);
  + }
  + private FileStream GetDataFileStream(string fileName)
  + {
  + string filePath = GetDataConfigPath(fileName);
  + if (File.Exists(filePath))
  + {
  + FileStream fileStream = new FileStream(filePath, FileMode.Open);
  + return fileStream;
  + }
  + return null;
  + }
  + private string GetDataConfigPath(string fileName)
  + {
  + return Application.streamingAssetsPath + "/DataConfig/" + fileName + ".bin";

}

附Killer的bat文件

@echo off

set XLS\_NAME=%1

set SHEET\_NAME=%2

set DATA\_DEST=%3

echo.

echo =========Compilation of %XLS\_NAME%.xls=========

::---------------------------------------------------

::第一步，将xls经过xls\_deploy\_tool转成bin和proto

::---------------------------------------------------

set STEP1\_XLS2PROTO\_PATH=step1\_xls2proto

@echo on

cd %STEP1\_XLS2PROTO\_PATH%

@echo off

echo TRY TO DELETE TEMP FILES:

del \*\_pb2.py

del \*\_pb2.pyc

del \*.proto

del \*.data

del \*.log

del \*.txt

@echo on

python xls\_deploy\_tool.py %SHEET\_NAME% ..\DataConfig\%XLS\_NAME%.xls

::---------------------------------------------------

::第二步：把proto翻译成cs

::---------------------------------------------------

cd ..

set STEP2\_PROTO2CS\_PATH=.\step2\_proto2cs

set PROTO\_DESC=proto.protodesc

set SRC\_OUT=..\src

cd %STEP2\_PROTO2CS\_PATH%

@echo off

echo TRY TO DELETE TEMP FILES:

del \*.cs

del \*.protodesc

del \*.txt

@echo on

dir ..\%STEP1\_XLS2PROTO\_PATH%\\*.proto /b > protolist.txt

@echo on

for /f "delims=." %%i in (protolist.txt) do protoc --descriptor\_set\_out=%PROTO\_DESC% --proto\_path=..\%STEP1\_XLS2PROTO\_PATH% ..\%STEP1\_XLS2PROTO\_PATH%\\*.proto

for /f "delims=." %%i in (protolist.txt) do ProtoGen\protogen -i:%PROTO\_DESC% -o:%%i.cs

cd ..

::---------------------------------------------------

::第三步：将data和cs拷到Assets里

::---------------------------------------------------

@echo off

set OUT\_PATH=..\Assets

set DATA\_DEST=StreamingAssets\DataConfig

set CS\_DEST=Scripts\Killer\DataConfig\ProtoGen

@echo on

copy %STEP1\_XLS2PROTO\_PATH%\\*.data %OUT\_PATH%\%DATA\_DEST%

copy %STEP2\_PROTO2CS\_PATH%\\*.cs %OUT\_PATH%\%CS\_DEST%

::---------------------------------------------------

::第四步：清除中间文件

::---------------------------------------------------

@echo off

echo TRY TO DELETE TEMP FILES:

cd %STEP1\_XLS2PROTO\_PATH%

del \*\_pb2.py

del \*\_pb2.pyc

del \*.proto

del \*.bin

del \*.log

del \*.txt

cd ..

cd %STEP2\_PROTO2CS\_PATH%

del \*.cs

del \*.protodesc

del \*.txt

cd ..

::---------------------------------------------------

::第五步：结束

::---------------------------------------------------

cd ..

@echo on

附protoc.exe的参数文档：

E:\Project\ied\_kl\_rep\client\_proj\trunk\KillerProject\DataConfigBuild\step2\_prot

o2cs>protoc --help

Usage: protoc [OPTION] PROTO\_FILES

Parse PROTO\_FILES and generate output based on the options given:

-IPATH, --proto\_path=PATH Specify the directory in which to search for

imports. May be specified multiple times;

directories will be searched in order. If not

given, the current working directory is used.

--version Show version info and exit.

-h, --help Show this text and exit.

--encode=MESSAGE\_TYPE Read a text-format message of the given type

from standard input and write it in binary

to standard output. The message type must

be defined in PROTO\_FILES or their imports.

--decode=MESSAGE\_TYPE Read a binary message of the given type from

standard input and write it in text format

to standard output. The message type must

be defined in PROTO\_FILES or their imports.

--decode\_raw Read an arbitrary protocol message from

standard input and write the raw tag/value

pairs in text format to standard output. No

PROTO\_FILES should be given when using this

flag.

-oFILE, Writes a FileDescriptorSet (a protocol buffer,

--descriptor\_set\_out=FILE defined in descriptor.proto) containing all of

the input files to FILE.

--include\_imports When using --descriptor\_set\_out, also include

all dependencies of the input files in the

set, so that the set is self-contained.

--include\_source\_info When using --descriptor\_set\_out, do not strip

SourceCodeInfo from the FileDescriptorProto.

This results in vastly larger descriptors that

include information about the original

location of each decl in the source file as

well as surrounding comments.

--error\_format=FORMAT Set the format in which to print errors.

FORMAT may be 'gcc' (the default) or 'msvs'

(Microsoft Visual Studio format).

--plugin=EXECUTABLE Specifies a plugin executable to use.

Normally, protoc searches the PATH for

plugins, but you may specify additional

executables not in the path using this flag.

Additionally, EXECUTABLE may be of the form

NAME=PATH, in which case the given plugin name

is mapped to the given executable even if

the executable's own name differs.

--cpp\_out=OUT\_DIR Generate C++ header and source.

--java\_out=OUT\_DIR Generate Java source file.

--python\_out=OUT\_DIR Generate Python source file.

附protogen.exe参数列表

E:\Project\ied\_kl\_rep\client\_proj\trunk\KillerProject\DataConfigBuild\step2\_prot

o2cs\ProtoGen>protogen.exe --help

protobuf-net:protogen - code generator for .proto

usage: protogen -i:{infile2} [-i:{infile2}] [-o:{outfile}] [-t:{template}] [-p:{

prop}[=value]] [-q] [-d]

-i: Input file(s); proto definitions, either as text or pre-compiled binary (via

protoc)

-o: Output file; if none specified, writes to stdout

-t: Template to use; defaults to csharp

-p: Property for the template; value defaults to true; use -p:help to view avail

able options

-q: Quiet; suppresses header

-d: Include all dependencies of the input files in the set so the set is self-co

ntained.

-ns: Default namespace; used in code generation when no package is specified

Examples:

protogen -i:input.proto -o:output.cs

protogen -i:input.proto -o:output.xml -t:xml

protogen -i:input.proto -o:output.cs -p:datacontract -q

protogen -i:input.proto -o:output.cs -p:observable=true