# 客户端代码规范

C#:

1.除非是需要变量只读，不要用属性（get;set;）

2.书写规范：

1）函数名大写：void Test();

2）共有成员变量，首字母小写，第二单词大写，public int testInt;

3）私有成员变量，在前面加下划线，private int \_testInt;

4) 局部变量，首字母小写，第二单词大写，int testInt;

3.大括号使用对称体：

void Test()

{

int test = 0;

}

4.常量首字母全大写 int const Test = 1;

5.枚举，名字首字母大写，元素首字母全大写，enum Test{ Test1 = 1, Test2= 2 };

6.不要忽略警告。

Lua:

1.定义及公共方法放到：Define.lua中。如有特殊需求，可以另起文件。

2.在语句末尾加：";"

3.书写规范：

1) 方法，首字母大写: function Test() end;

2) 本地变量，下划线+首字母小写：local \_test;

3) 全局变量，首字母大写：Test\_Value = 0;

4) 局部变量，首字母小写，第二单词大写，local testValue;

5) 全局常量，全部大写：HTTP\_LOGIN\_REQUEST = 1001;

6) 本地常量，下划线+首字母大写：\_Action\_Map = { 1, 2, 3 };

4功能模块类，全部用表限制命名空间：Test = {}; function Test.Func() end 如需要本地方法：function Test\_Func() end

1. 大部分功能能使用单例解决的用面向过程的编程方式。如需要多实例的地方用Class(“”)，来实现面向对象编程。

约束：全部UI功能使用面向过程的方式。

6不允许直接使用Lua高级特性：元表等。如需要面向对象编程方式使用：Class( “TestClass” );

**网络：**

命名：data为与服务器交互的数据，msg为反序列化的Lua Table

1.回包：

1) Socket:

监听：Event.AddListener( Res\_ResLogin, function() end );

解包：local msg = Res\_LoginMsg():Unpack( data );

2) Http:

监听：Event.AddListener( HTTP\_LOGIN, function() end );

2.发包：(嵌套时同理)

local msg = Req\_LoginMsg();

msg.secureCode = Login.SecureCode;

msg.version = "1.0.0";

Network.Send( msg );

**其他：**

1.起文件命名时，如果是一组同类型的，要求能从名字看出是什么类型。（包括C#，Lua）。比如DlgMain，CachePlayer。

1.SVN提交时，请写详细Log。

2.非调试日志的正式日志需要在前面加上”<>空格”，来表示日志类型，如：Debugger.Log( “<Boot> 游戏启动” );

**务必遵守，违者责令更改，并给大家发红包！**