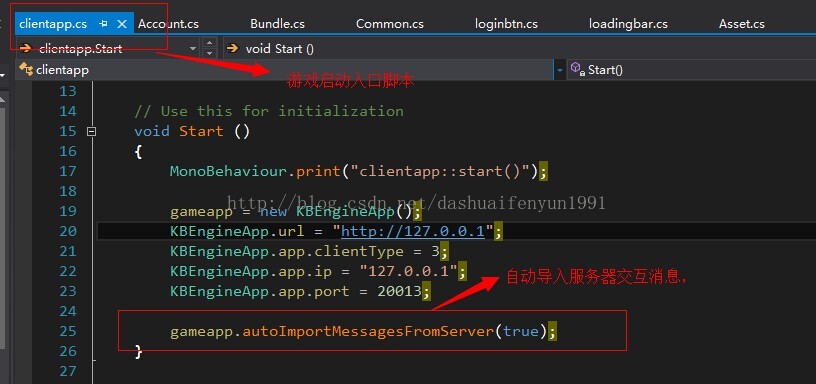
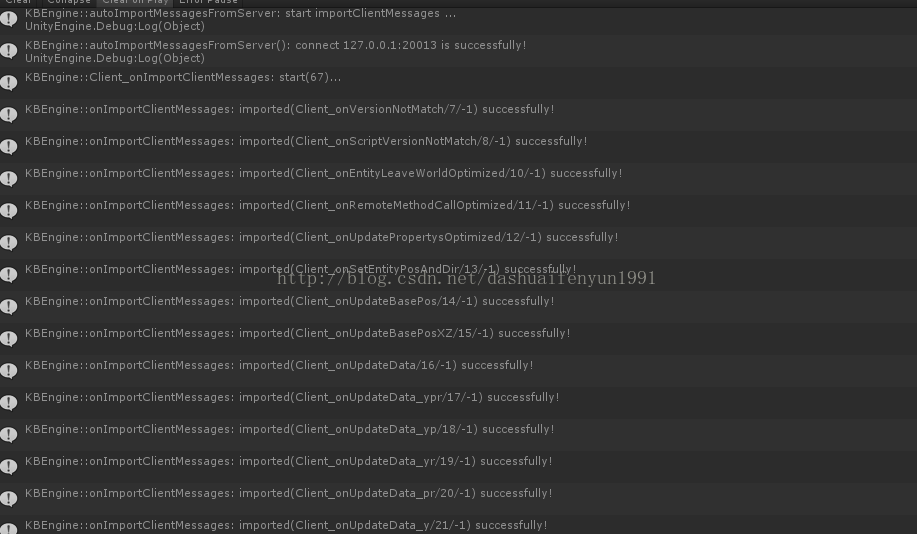
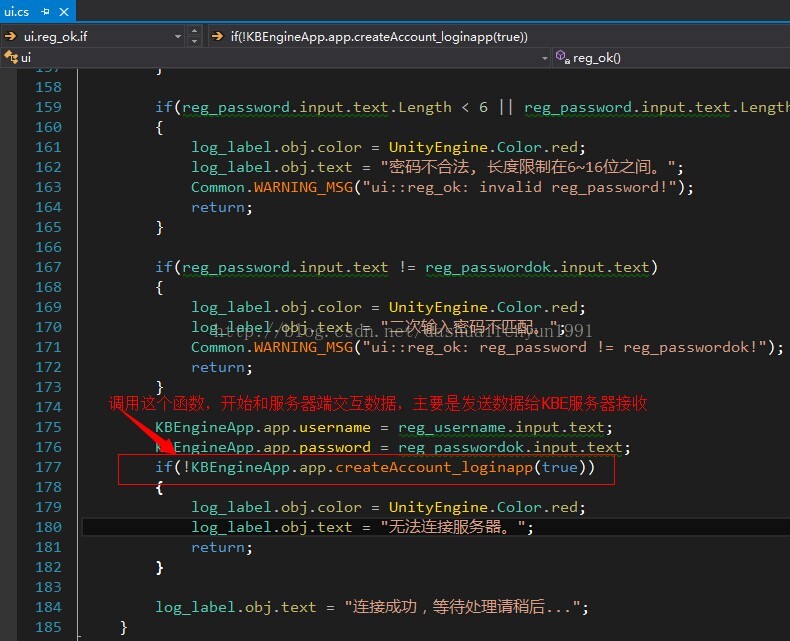
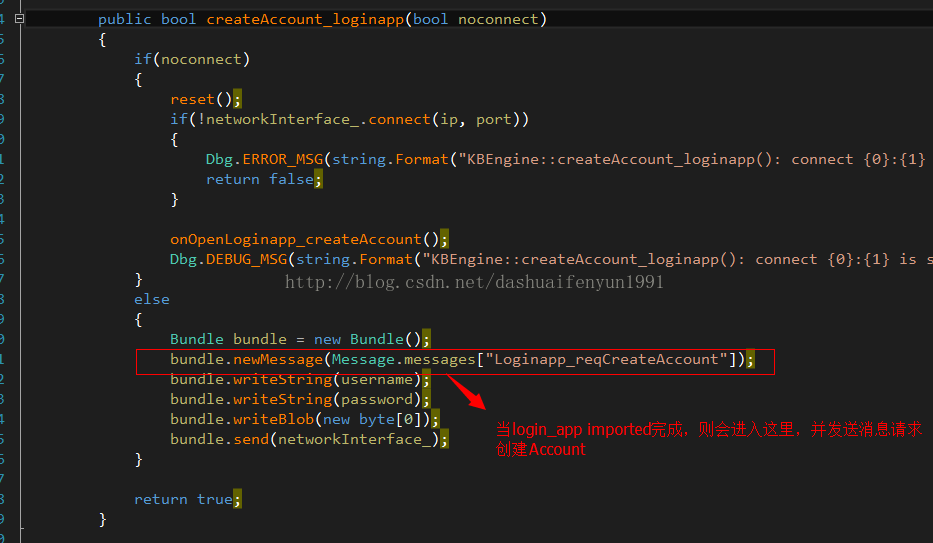
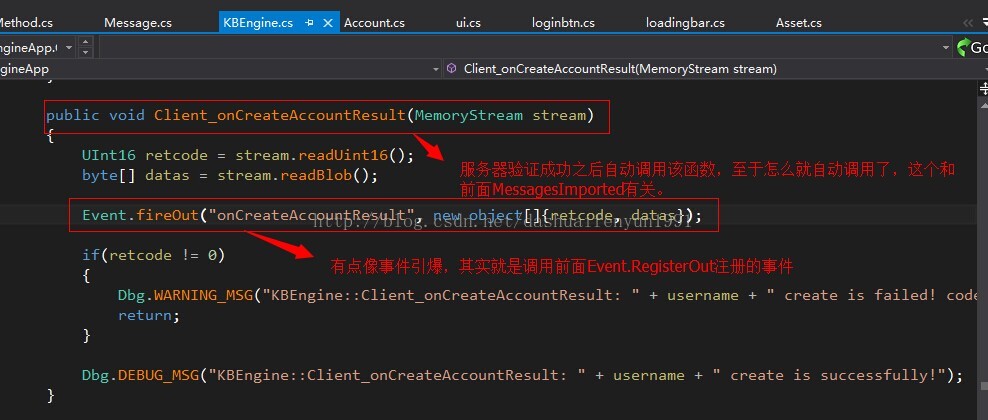
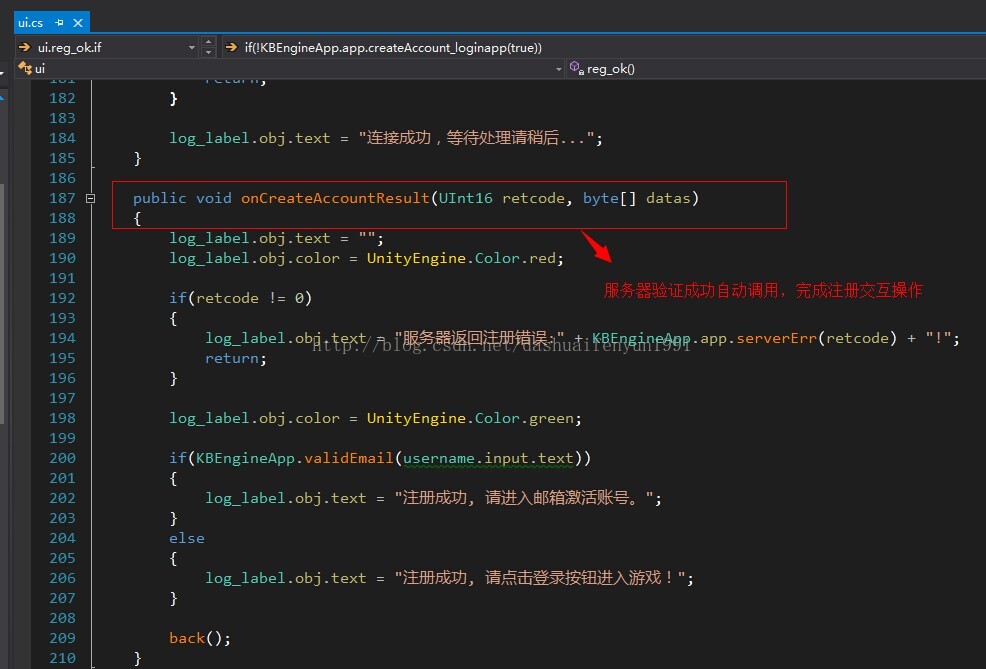
|  |  |
| --- | --- |
|  | [U3D + KBE Demo传输消息流程图文解析【客户端】](http://www.unitymanual.com/thread-37844-1-1.html) |

本文选择kbe中最简单直接的账户注册登录为实例。   
该流程测试例子为 kbengine\_unity3d\_warring-0.1.13 + kbengine-master.   
unity3d + kbengine demo搭建见文。   
客户端：   
1.   
  
该脚本clientapp是kbengine的客户端插件的入口脚本，通过挂载该脚本，start启动，连接服务器，导入基本的消息，以及loginapp相关消息为登录模块作准备。   
在这里，我们可以更加详细的进入kbengine插件内部去看下，在导入消息之前做了哪些工作？   
gameapp = new KBEngineApp();   
创建了一个KBEngineApp的实例，看下构造函数做了什么工作？   
 public KBEngineApp() { app = this; networkInterface\_ = new NetworkInterface(this); kbethread = new KBEThread(this); t\_ = new Thread(new ThreadStart(kbethread.run)); t\_.Start(); // 注册事件 installEvents(); }   
  
看样子是开启了一个线程。   
networkInterface和kbethread两个对象是干嘛用的？   
networkInterface:   
 public NetworkInterface(KBEngineApp app) { this.app\_ = app; bindMessage(); packets\_ = new List<MemoryStream>(); }   
  
    
 public void bindMessage() { if(Message.messages.Count == 0) { Message.messages["Loginapp\_importClientMessages"] = new Message(5, "importClientMessages", 0, 0, new List<Byte>(), null); Message.messages["Baseapp\_importClientMessages"] = new Message(207, "importClientMessages", 0, 0, new List<Byte>(), null); Message.messages["Baseapp\_importClientEntityDef"] = new Message(208, "importClientMessages", 0, 0, new List<Byte>(), null); Message.messages["Client\_onImportClientMessages"] = new Message(518, "Client\_onImportClientMessages", -1, -1, new List<Byte>(), this.app\_.GetType().GetMethod("Client\_onImportClientMessages")); Message.clientMessages[Message.messages["Client\_onImportClientMessages"].id] = Message.messages["Client\_onImportClientMessages"]; } }   
  
看到这里大概知道了，这里是定义了几个消息类型，初始化了一下需要服务器交互的消息。   
这就明白了gameapp.autoImportMessagesFromServer(true);通过这些消息，与服务器连通。   
kbethread:   
kbethread = new KBEThread(this);这个很简单，声明一个对象，并将KBEngine的实例给KBEthread。   
    
new ThreadStart(kbethread.run) 开启线程的时候调用了一个run函数，进去看下。   
  
到这里就不列举进去了，里面是通过process循环执行events，来保证能够及时处理服务器发到客户端的消息。   
 Monitor.Enter(events\_in); if(firedEvents\_in.Count > 0) { foreach(EventObj evt in firedEvents\_in) { doingEvents\_in.AddLast(evt); } firedEvents\_in.Clear(); } Monitor.Exit(events\_in);   
  
    
2.   
看到DEBUG时候，很疑惑为什么会有这多消息导入呢，貌似代码中并没有这么多处理的。   
经过分析，是服务器返回一个消息，调用Client\_onImportClientMessages该函数，这个消息可以上面的消息定义中找到。   
 public void Client\_onImportClientMessages(MemoryStream stream) { byte[] datas = new byte[stream.wpos - stream.rpos]; Array.Copy (stream.data (), stream.rpos, datas, 0, stream.wpos - stream.rpos); Event.fireAll ("onImportClientMessages", new object[]{currserver, datas}); onImportClientMessages (stream); }   
  
然后它就开始了它的消息导入之旅，也可以说是消息定义，这样这样服务器就可以通过这些消息与服务器进行通信了。   
  
当基本消息导入之后，就会默认进入处理loginapp消息处理部分。至此，客户端和服务端的通信已经联通，客户端可以通过注册消息，等待服务端消息返回处理结果。   
3.   
  
客户端，开始注册相关消息，并且实现回调处理服务器返回的结果。   
4.   
  
这是在u3d的UI上点击注册之后，通过该函数，发送数据给服务器。   
5.   
  
6.   
  
发送消息给服务器，服务器返回消息调用该函数，并在客户端的消息机制里面，调用UI面板注册的函数。   
7.   
  
    
最后到这里就宣告注册完成了！至于登陆，除了UI面板发送事件不一样，其他的都差不多。可以自行实践。   
下一篇将针对该文，分析一下服务器怎么处理注册的。   
以上为个人意见，有错请谅解！   
转载请注明出处。   
    
作者： 大帅纷纭   
微博：<http://weibo.com/2357191704/profile?topnav=1&wvr=6>   
博客：<http://blog.csdn.net/dashuaifenyun1991>   
邮箱：[bandit\_empire@163.com](mailto:bandit_empire@163.com)