# ESFramework开发手册 -- Android版

 ESFramework的Android客户端引擎的相关API与PC版保持一致的，所以，如果熟悉PC版的API，那么上手Android API就很容易了。

## 　第一章发送和处理消息

使用通信框架最基础的需求就是收发信息，ESFramework(Android)底层已经为我们封装好了所有与信息收发相关的操作，我们只要使用<com.oraycn.es.communicate.framework>下IBasicOutter接口中相关组件的API来发送信息，以及实现<com.oraycn.es.communicate.framework>下ICustomizeHandler接口来处理收到的信息就可以了。

### 1.客户端发送信息

　　客户端可以发送信息给服务端，也可以发送信息给其他在线用户。

　　客户端可以使用RapidPassiveEngine中的getCustomizeOutter()方法获取ICustomizeOutter接口的实例, 通过调用实例的方法来发送信息, ICustomizeOutter具体接口定义如下：

![C:\Users\ws\AppData\Roaming\Tencent\Users\69428484\QQ\WinTemp\RichOle\95HL(4O%K9UT]1ENR1NPZ%I.png]()



　发送消息有几种方式

#### 普通发送：

      调用send(String targetUserID, **int**informationType, **byte**[] info)方法进行普通发送，即将信息写入网络流后就立即返回。

      send方法的重载有个ActionTypeOnChannelIsBusy参数，用于指示当通道繁忙时所采取的动作：继续发送、或丢弃数据。在某些系统中，对于一些非重要非紧急信息的发送，可以为ActionTypeOnChannelIsBusy参数传入枚举值Discard（丢弃）。

#### 带ACK机制的发送：

      调用sendCertainly方法发送信息时会启用ACK机制，即将信息发送出去后，调用并不返回，而是要等到接收方的ACK后，才返回。ACK机制是由ESPlus底层实现的，我们直接使用，不需要做任何额外的其它工作。关于带ACK机制的信息发送的更多内容可以参见[ACK机制](http://blog.oraycn.com/ESFramework_09.aspx)。

#### 信息同步调用：

      调用query(**int** informationType, **byte**[] info)和query(String targetUserID, **int** informationType, **byte**[] info)方法可以发送请求信息，并返回接收方处理请求后的应答信息。就像方法调用一样 - - 使用参数调用方法并返回结果。从两个query方法的重载看到，信息同步调用的对象既可以是服务端、也可以是另外一个在线客户端。关于信息同步调用的更多内容可以参见[消息同步调用](http://blog.oraycn.com/ESFramework_09.aspx)。

#### 回复异步调用：

      重载的**void** query(String targetUserID, **int** informationType, **byte**[] info, CallbackHandler handler, Object tag)方法,在发送请求信息后，不会阻塞而继续向下执行，而框架在收到对应的回复信息时，会调用CallbackHandler 的execute方法体。由于调用线程与回复回调的线程不是同一个线程，所以称这种机制为回复异步调用。

#### 发送达数据块：

      调用sendBlob方法可以将大数据块信息发送给服务端或任何其他的在线用户。

### 2.处理信息

 　　客户端可以收到来自其它客户端或服务端的信息、大数据块、以及同步调用。服务端也可以收到来自客户端的信息（转发的信息除外）及同步调用。那么，我们如何处理这些接收到的信息了？无论是服务端，还是客户端，都只要实现ICustomizeHandler接口即可。



#### 接口说明

（1）在实现这个接口之后, 只需要实例化这个具体的实现类, 并 在RapidPassiveEngine的customizeHandler属性即可自动处理接收到的消息

（2）凡是sourceUserID参数为null的，都表示被处理的信息是来自服务端的；否则，表示被处理的信息是由其它在线客户端发出的。

（3）ICustomizeHandler接口的所有方法都是在后台线程中被调用的，所以如果这些方法的实现中不能直接修改UI线程

（4）在客户端，可以将ICustomizeHandler的实现类的实例传递给RapidPassiveEngine的initialize方法以挂接到框架。

### 3.更多说明

#### 信息发送模型

      信息发送可以使用同步模型或异步模型，在方法中通过bool型post参数体现出来。如果其值为true，表示使用异步模型（即发送方法的调用立即返回，不用等到信息发送完毕）；否则使用同步模型（阻塞调用线程，直到信息发送完毕）。

#### 信息处理

      客户端和服务端的ICustomizeHandler，我们称之为自定义信息处理器，或者业务处理器，表示其用于处理我们应用系统的具体业务逻辑。

（1）业务处理器将在后台线程中被调用，所以，实现业务处理器的方法中如果涉及到了UI操作，则必须将调用转发到UI线程。

（2）业务处理器的方法必须尽可能快地返回，否则，将不能及时地处理后续的消息。如果某个业务处理方法非常耗时，可以考虑使用异步方式。

#### 大数据块

      当发送大数据块时，发送方会将其拆分为许多连续的片段逐个发送，而在接收方会自动将接收到的片段重组起来构成一个完整的信息。而且无论是发送大数据块，还是普通信息，在接收方都是调用相同的方法（ICustomizeHandler的handleInformation方法）来处理的。

## 第二章 好友与组

IFriendOutter和IGroupOutter的功能已被IContactsOutter模块所取代

## 第三章 联系人

联系人（IContactsOutter）是ESFramework 6.0（必须与对应版本的服务端配合使用）新增的功能，此接口将之前的好友与组功能进行整合，并取而代之





#### 1.接口说明

（1）getGroupMembers 会返回某个组的所有成员，并将在线成员与不在线成员区分开来。

（2）当用户上线或下线时，框架会回调ContactsEventListener 接口的contactsConnected或contactsOffline事件以通知其所有相关联系人。

（3）可以通过Broadcast向任何一个组发送广播，目标组的每个在线成员都将会通过ContactsEventListener 的BroadcastReceived事件来获得广播内容。

#### 2.关注联系人的实时状态

 在类似IM的系统中，每个运行的客户端实例，在其运行的整个生命周期中，都需要清楚地知道与其相关每个联系人的实时状态，这个需求可以这样来实现：

（1）当某个客户端登陆成功后，就调用getContacts 方法和getAllOnlineContacts方法以获取联系人列表和所有的在线联系人列表。这样，就知道了所有联系人的初始状态。

（2）实现ContactsEventListener 事件接口，并通过setContactsEventListener方法预定事件，然后在运行的过程中，当contactsConnected和contactsOffline事件触发时，就修改对应联系人的状态。这样就保证我们的客户端可以实时地知道每个相关的联系人是否在线。

## 第四章 [在线用户管理、基础功能及状态通知](http://blog.oraycn.com/ESFramework_android_02.aspx)

      本文介绍[ESFramework 开发手册（00）－－概述](http://blog.oraycn.com/ESFramework_00.aspx)一文中提到的四大武器的第二个：在线用户管理、基础功能及状态通知。

      在解决了[发送信息和处理信息](http://blog.oraycn.com/ESFramework_android_01.aspx)之后，还有一些基础功能是很多分布式通信系统都需要用到的，比如，查询某个用户是否在线、获取在线用户列表、自己掉线时得到通知，等等。IBasicOutter下的组件，为我们解决了这些基础问题。

### 客户端

   客户端通过调用<com.oraycn.es.communicate.framework.IBasicOutter>接口对应的方法就可以完成基础功能或得到相关状态改变通知。 我们可以<RapidPassiveEngine>的getBasicOutter()方法来获取IBasicOutter引用。

![C:\Users\ws\AppData\Roaming\Tencent\Users\69428484\QQ\WinTemp\RichOle\~{$VEW3][YH61%A%PMYM%QJ.png]()



#### 状态改变事件通知

      首先，我们看看RapidPassiveEngine暴露的两个事件：

1. beingKickedOut 当自己被踢出时将触发该事件。
2. beingPushedOut 发生于当服务端将重登陆模式设置为ReplaceOld时，并且同名用户的成功登录，将会把老的在线用户挤掉而导致其下线。关于重登陆模式的更多内容可以参见[重登陆模式](http://blog.oraycn.com/ESFramework_08.aspx)。

#### 基础API

      接下来，我们简单看看BasicOutter的几个方法。

1. logon方法用于在登录时验证用户密码。该方法会在客户端Rapid引擎初始化时被引擎自动调用，所以，在使用Rapid引擎时，我们通常不需要手动调用它。如果有的系统需要验证除了密码之外更多的信息，那么可以通过systemToken参数进行传递这些额外信息。Logon方法返回类型为LogonResponse，其属性LogonResult表示了登录结果。如果LogonResult为0，表示登录成功；如果LogonResult为2，表示该账号已经在其它地方登录；如果LogonResult为1，则表示验证账号密码没有通过，没有通过的原因由LogonResponse的FailureCause属性指明。
2. getAllOnlineUsers用于获取所有在线用户，通常该方法仅仅用于demo，因为在正式的系统中，在线用户数可能是非常巨大的，这将导致GetAllOnlineUsers的返回消息非常大，甚至可能超过框架的最大消息尺寸的限制。
3. ping系列方法，用于获取当前客户端到服务端或到另一个在线客户端的消息来回的耗时，由于其是在应用层来模拟类似ICMP的ping，所以这个方法返回的值通常比ICMP的ping大一些。尽管如此，在一些应用中，该Ping的结果还是有一些参考价值的。
4. 有时，我们需要命令服务器将一些恶意的用户从服务端踢出（断开其连接），那么就可以调用kickOut方法，被踢出的客户端将会触发上述的beingKickedOut事件。
5. sendHeartBeatMessage方法用于向服务器发送心跳消息。如果我们使用的是Rapid引擎，那么框架会自动发送心跳消息，所以，我们通常不需要手动调用该方法。关于心跳消息的更多内容可以参见[心跳机制](http://blog.oraycn.com/ESFramework_07.aspx)。

#### **TCP连接状态**

<com.oraycn.es.communicate.framework.IBasicOutter>提供了一部分基础功能，还有另一部分很重要的基础功能需要涉及到客户端的Rapid引擎，我们在这里也一并介绍一下。客户端如何知道自己与服务器的TCP连接的状态及其变化了？<com.oraycn.es.communicate.framework.IRapidPassiveEngine>的几个事件来获取这些信息。



### 2.UserID的长度

　　Rapid引擎使用的就是ESPlus提供的基于二进制的消息头，这个消息头的默认长度是36字节，允许的UserID最大长度为11字节。但是，如果你的系统中需要用到的UserID长度超过了11字节，该怎么办了？我们可以通过调用GlobalUtil的setMaxLengthOfUserID静态方法来设定ESFramework允许的UserID的最大长度：

1. 注意，我们必须在Rapid引擎的Initialize方法执行之前调用setMaxLengthOfUserID方法。而且，客户端和服务端必须采用相同的设置，否则，就一定会导致服务端和客户端通信出现异常。如果你的客户端是使用的Silverlight，那么使用ESFramework.SL时也是如此。

       在ESFramework内部，组（Group）ID也采用与UserID相同的规则。
　　还要提醒的是，在能满足项目需求的情况下，尽可能使UserID的最大长度短一点，这样可以使得消息头更加短小，从而避免浪费本不需要的带宽。尤其在高性能、巨大并发的应用中，这点就更关键了。

### 3.消息的最大长度

   　Rapid引擎内部默认设置的消息的最大长度为100K（100\*1024），并且这个长度还包含了上述消息头的长度。如果您的应用需要发送的单个信息的长度超过了100k，就会被ESFramework认为是恶意的消息，ESFramework会丢弃该消息并关闭对应的连接。

      我们建议：在能同样满足项目的需求下，应该尽可能地使传送的消息小，这样不仅可以节省带宽，而且还有助于提升并发的性能。如果应用中确实有信息的长度超过最大限制，那么还可以通过customizeOutter的sendBlobToUser方法来将其作为大数据块进行发送。

      最后，大家可以查看最简单的那个demo的源码，并运行demo，来了解上述的状态改变通知及其它基础功能。