轻量级 STEP 会话层接口规范
（LFIXT 会话协议）
Version 1.00α

Lightweight Fix Session Layer Protocol

本规范由上海证券交易所和深圳证券交易所联合发布

2014-03-26

文档说明

文档名称	轻量级STEP会话层接口规范
内容描述	描述了轻量级的STEP会话协议相关内容。

修订历史		
日期	**版本**	**修订说明**
2014-01-20	0.80	创建
2014-3-26	1.00 α	根据会员反馈意见进行修订：1. 增加了REJECT消息 2.修正了一些文字错误 3. 将LFIXT协议细分为兼容模式和精简模式，并列明协议兼容性矩阵及典型应用场景;4. 为Logon消息增补了会话状态字段 5. 根据会员反馈意见增补了字段长度说明； 6.解释了什么叫做Garbled Message 7.根据正文的变更修订了附录

名词释义

词汇缩写	含义
STEP	Securities Trading Exchange Protocol 证券交易数据交换协议。
FIX	Financial Information Exchange 金融信息交换协议。

目录

轻量级 STEP 会话协议接口规范

一、 范围

本标准对 STEP 会话层协议（即 FIX 会话层协议 FIXT 1.1 版本）进行了裁剪和改造,确立了一个基于 TCP 的轻量化 STEP 会话层协议（记作：LFIXT 会话层协议），同时 LFIXT 标准仍然可以保持和标准 STEP/FIX 会话层协议的互操作性。

本标准规定了 LFIXT 会话层协议使用的会话机制、消息格式、安全与加密、数据完整性、扩展方式、消息定义、数据字典等内容。

如无特别说明，本标准中提及的接收方、发送方等通信参与方均特指遵循 LFIXT 会话层协议而实现的应用程序或模块。由于遵循本协议开发的程序或模块将可以同标准 FIX 引擎进行正常通信，因此若通信的一方是标准的 FIX 引擎，则除本文特别制订的少数额外约束外，其实现不受本文档约束。

二、 会话机制

2.1 术语和定义

2.1.1 会话层重传

会话层重传是标准FIX会话层协议所规定的一种重传机制，用来确保有序、无失地传输每一条会话层消息。在标准FIX会话层协议中，会话层重传由消息接收方在识别出消息序号缺口之际主动发起，采取的方式是发送一条消息重传请求给到对方。

LFIXT会话协议在事实上取消了标准FIX会话层协议的会话层重传，只对外仍然表现为标准的FIX会话层，且可以和对端的标准FIX会话层实现进行互操作。由于单个LFIXT会话使用单个TCP连接作为底层通信机制，因此在单个TCP连接内部，每一条消息将被有序、无失地传输。属于同一会话的、前后相继的若干次TCP连接之间，将可能存在会话层消息丢失，但收到的会话层消息将仍然具有有序接收的性质。

由于在LFIXT下会话层可能存在消息丢失，因此丢失的业务消息将只能通过应用层重传予以恢复。

2.1.2 应用层重传

由于LFIXT会话协议的会话层恢复机制仅仅是为了与标准FIX会话协议兼容，不能作为真正的消息恢复机制使用。因此，必须通过应用层商定的重传机制予以恢复。应用层重传的具体机制不属于本文档规定的范畴，请参见具体的应用层数据接口规范。

2.1.3 NxtIn 和 NxtOut

会话双方收发的每条消息都带有一个消息序号。参与通信的每一端都需要维护一对序号（NxtIn，NxtOut），NxtIn表示下一个期望的入向消息序号，NxtOut表示下一条出向消息将被赋予的序号。

2.1.4 会话发起方和接受方

会话的建立需要一个发起方，需要一个接受方。发起方是先发出Logon消息并希望对方响应以一个Logon消息的一方，接受方则是等待发起方首先发出Logon消息并响应以Logon消息的一方。

会话的发起方和接受方在会话建立后都可以双向地进行消息的发送和接收。不要将会话发起方（initiator）、会话接受方（acceptor）同某条特定消息的发送方（sender）和接收方（receiver）混为一谈。

类似于会话发起方和会话接受方，也定义有注销发起方和注销接受方、会话重置发起方和会话重置接受方的概念。

标准FIX协议原则上适用于各种不同的传输层协议（如UDP），因此不可能根据TCP socket等特定的传输层信息来区分哪些报文隶属于同一个FIX会话，而且由于FIX中并不为每个FIX会话定义有所谓“会话号”标签，且不是全部报文都具有username一类的标签，因此区分FIX会话的唯一标识符只能是SenderCompID和TargetCompID的组合。

标准FIX协议中，单个FIX引擎不能同时维护相同SenderCompID+TargetCompID的两个会话。标准所推荐的做法是：在已存在一个合法会话时，若一方试图以同样的SenderCompID + TargetCompID发起新的会话，对方将不发送任何Logout消息就直接终止新发起的会话，原先已存在的会话不应受到影响。

标准FIX协议并未明确不同的FIX引擎是否允许同时保有同样标识的会话。一般而言，同一台服务器上的同标识会话较易进行查重，针对不同服务器建立的同标识会话则较难实现查重，但也非无法做到。

LFIXT协议规定：在处理入向登录报文时，应利用SenderCompID和TargetCompID进行会话查重，之前已存在的同标识会话一定不受影响，但较晚收到的同标识会话请求可能被接受，但也可能被拒绝，协议不作限定。若请求被拒绝，则拒绝方式将遵循标准FIX协议的约定，不回送Logout消息，直接断开TCP连接。会话发起方应当对这种情形做好准备。

LFIXT协议只允许单个会话同时通过单个TCP连接进行全双工通信，因此在通过登录报文信息确定是否允许继续通信后，可以直接利用socket来区分报文所属会话，但对于从同一socket上收到的后续报文仍应检查其SenderCompID和TargetCompID是否和登录时一致。

2.1.5 消息序号

所有的消息都由一个唯一的会话层消息序号(即消息头中的 MsgSeqNum 字段)进行标识。消息序号在会话开始时[一般]被初始化为 1，并在整个会话过程中连续递增[1]，直到该会话过程全部结束。通过监视消息序号的连续性，通信双方可以识别消息缺口并做出反应，并可在同一会话的前后多个连接间进行同步[2]。

每次会话都会创建一套独立的入向及出向的序号序列，参与连接的任何一方都维护一套用于出向消息的序号序列(NxtOut)，同时也维护另一套独立的入向消息的序号序列

1. 这个说法是针对通常情况而言的。在下述情况下，接收方收到的消息的 MsgSeqNum 也可能出现倒流：1)收到的消息是 SeqReset-Reset 消息且 PossDupFlag=Y，此时 MsgSeqNum 应忽略，即使出现倒流也不是错误；2）除此以外，所收到消息的 PossDupFlag 是 Y，且此类消息确实允许出现 PossDupFlag=Y，表示这是会话层重传；3）通过 LOGON 消息进行会话序号重置时，收到的消息其 MsgSeqNum 一定是 1，因此也可能出现倒流。
2. LFIXT 会话协议在事实上取消了标准 FIX 会话层协议的会话层重传，这里的同步只是为了兼容标准 Fix 会话机制，并不进行真实的消息同步。

(NxtIn)，用以监视接收的消息序号，以保证消息缺口的发现和处理。

会话建立后，当 LFIXT 协议实现者接收到的消息序号不等于预期接收的消息序号(NxtIn)时，需要考虑进行修正处理。这里有几种情况：

1. 如果入向消息序号< NxtIn，且不属于前文脚注中注明的若干种情况之一时：**表明发生了严重的错误，必须立即结束会话**，并开始进行人工干预。
2. 如果入向消息序号< NxtIn，且属于前文脚注中注明的若干种情况之一，不属于错误，应进行正常处理。
3. 如果入向消息序号> NxtIn，那么表明有消息被遗漏。因为 LFIXT 使用 TCP 为传输协议，出现这种情况**说明发生了严重异常错误，应立刻终止当前会话**。

2.1.6 心跳

在消息交换的空闲期间，连接双方将以规定的时间间隔产生心跳消息。通过心跳消息可以监控通讯连接的状态并识别出入向消息序号的缺口。

心跳间隔时间由会话发起人通过登录消息的HeartBtInt字段确定。在传送了任何消息(而不仅仅是心跳消息)之后，都应立即重置心跳间隔计时器。心跳间隔时间应该得到连接双方的确认，由会话发起人给出，并得到会话接受方的确认。

连接双方应使用相同的心跳间隔时间。每个心跳消息都将占用一个MsgSeqNum消息序号。

2.1.7 有序消息处理

LFIXT会话协议采用TCP连接作为底层通信机制，会话建立后，在同一个TCP连接的延续期间，接收方在发现入向消息缺口时，说明发生了严重异常，建议接收方终止该会话并断开TCP连接。如果接收方为会话的发起方，则应根据需要重建会话。

2.1.8 可能的消息重复传送

本会话协议采用 TCP 连接作为底层通信机制，会话双方在建立 TCP 连接之后，通过 Logon 消息进行序号协商，其后则是基于 TCP 进行的连续通信，正常情况下，不应该出现前面消息丢失却收到后面消息的情形。所以，1) 在发现入向消息序号缺口时，LFIXT 会话协议的实现者不会发送重传请求，而是回送 Logout 后直接断开连接，但 2) 允许在入向消息中出现重传请求（比如基于标准 FIX 引擎的通信对手方虽然收到前面的消息但自己没保存,并期望能按标准 FIX 会话层协议通过重传请求取回),对此 LFIXT 会话协议实现者将简单回送 SeqReset-Reset 消息予以打发，3）允许在入向消息中出现 PossDupFlag = Y[3]的消息（比如基于标准 FIX 引擎的通信对手方虽未收到本方发出的重

3. 除了 REJECT 消息之外，其他管理消息理论上都不应被重发，而是通过发送带有同样消息序号的、带有 PossDupFlag 标志的 SeqReset-GapFill 消息对原消息进行替代。在此过程中，被替代的 SeqReset-GapFill

传请求，但仅仅因为怀疑本方可能错过某些消息，而向本方发送这类 PossDupFlag =Y 的消息[4])。

2.1.9 可能的消息重新发送

在LFIXT会话协议中，应用层重发的标志应在应用层协议中明确设置，而不应该体现在会话层消息的标志位上。由于互操作的对方必须遵从同样的应用层协议，因此LFIXT会话协议将不会给出向消息打上任何Possible Resend标志。

LFIXT会话协议允许在入向消息头中出现Possible Resend标志，但将忽略该标志，直接将不附带该标志的消息交由应用层处理。

2.1.10 消息完整性

消息数据内容的完整性可以用两种方式来验证：验证消息长度，及字符的简单校验和。

消息长度被包括在BodyLength字段中，可以通过清点消息之中跟在BodyLength字段之后、直至并包括直接先于CheckSum域号（”10=”）出现的那个域界定符<SOH>之间的字符来验证。

校验和的验证方法是：从消息头中‘8=’中的‘8’开始、直到并包括直接先于 CheckSum 域号‘10=’出现的<SOH>字符，将每个字符的二进制值加总后，将计算值的最低 8 位同 CheckSum 字段中的值进行比较。

2.1.11 混乱的消息(garbled message)

根据标准 FIXT 协议的附录，当至少出现以下情形之一时，一条消息被称为“混乱的”：

BeginString(tag 8) 不是消息的第一个标签，或不以 8=FIXT.n.m 的形式出现。

BodyLength(tag 9)不是消息的第二个标签，或未包含正确的字节计数

MsgType(tag 35) 不是消息的第三个标签

CheckSum(tag 10)不是最后的标签，或其取值不正确

若 MsgSeqNum(tag 34)缺失，必须立刻终止 FIX 连接，因为这表明出现了严重的应用错误，很可能只能通过修改软件来绕过。

消息本身虽然仍然以同样的消息序号、带上同样被置位的 PossDupFlag 标志出现，也被 FIX 标准解释为“替代”而非“重发”。

4. （PossDupFlag =Y）为 Y 的管理消息请参见具体的消息定义

2.1.12 消息确认

由于会话层协议是基于乐观的消息传输模式，通过监视消息序号发现缺口，不支持对每个消息收发的确认。但大量消息收发的确认可在应用层定义。在应用层接受和拒绝是允许的。

2.1.13 加密

LFIXT 会话层不对数据进行加密处理，会话双方可考虑使用通信层的加密机制。

2.2 会话管理

LFIXT 会话协议采用 TCP 连接作为底层通信机制。

若 LFIXT 会话协议的实现者作为会话的主动发起方，必须在每次新建 TCP 连接之后通过置位序号重设标志(ResetSeqNumFlag)的 Logon 消息来将起始消息序号重置回 1，因此，此时会话和 TCP 连接是一一对应的。

虽然 LFIXT 会话协议可以被设计成底层使用两个独立的 TCP 连接，每个连接都以单工模式工作，但由于在 TCP 连接上实现全双工的通信并不困难且维护简单，因此 LFIXT 会话协议规定：对于单个会话而言，同时只使用一个全双工的 TCP 连接。

若 LFIXT 会话协议的实现者作为会话的接收方,由于该会话的发起方可能是标准的 FIX 引擎，此时建立的会话可以跨越多个 TCP 连接。

在单次 TCP 连接内部，每个会话都分为三个部分：建立会话、消息交换、终止会话。

2.2.1 建立会话

建立会话包含三个步骤：建立连接（即为建立 TCP 连接）、身份认证、消息同步。

2.2.1.1 建立连接

LFIXT 会话的发起方与接受方建立 TCP 连接。LFIXT 会话协议的实现者在 TCP 连接建立后，应当总是初始化 NxtIn = 1, NxtOut = 1。

2.2.1.2 身份认证

1. 会话发起方发送登录消息（Logon），接受方认证发起方身份的合法性。
2. 如果发起方身份通过认证，则接受方发送一个登录消息作回应。
3. 如果认证失败，会话接受方则在可选地发送一个含失败说明的注销消息（Logout）后关闭连接。发送注销消息并非是必须的，因为这样做会消耗一个序号，在某些情况下可能会引起其他问题[5]。
4. 会话发起方必须等待来自接受方的确认 Logon 消息，方可向接受方发送其他消息。否则，接受方可能尚未准备好接收它们。
5. 在发起方被认证后，接受方将立即回应一个确认 Logon 消息。发起方将把从接

5. 这个问题和标准 FIX 引擎对于报文所属会话的认定方式有关，单在 LFIXT 引擎方面则并无不妥。

受方返回的 Logon 消息作为“一个会话已经建立”的确认。

2.2.1.3 消息同步

LFIXT 会话协议并不提供真正的会话层重传机制，因此 LFIXT 会话协议的实现者作为会话的发起方，可通过会话重置消息（即 ResetSeqNumFlag=Y 的 Logon 消息）将会话双方的消息序号重置，来完成会话层消息同步。

LFIXT 会话协议的实现者作为会话接受方，可以利用 Logon 消息中的 NextExpectedMsgSeqNum 来完成会话层消息同步。这种方式提供了对标准 FIX 会话协议的消息同步的兼容，具体机制参见“登录消息处理”一节。

2.2.2 消息交换

在建立会话之后，会话双方可以开始进行正常的消息交换。交换的消息包括“管理消息”和“应用消息”，本规范仅对管理消息进行描述。应用消息请参见具体的数据接口规范。

2.2.3 注销会话

LFIXT 会话的正常结束是通过连接双方互相发送注销消息（Logout），注销时不需要进行消息缺口检查。若结束时没有收到回送的注销消息（Logout），则把对方视作已注销。除此之外的其它方式的会话结束视为非正常，并应按错误来处理。

在结束会话之前，注销消息（Logout）的发起方应该等待对方回送的注销消息（Logout）。如果接收方在一定时间内没有答复，那么会话就可以立即中断[6]。

2.3 恢复

LFIXT 会话协议的会话层恢复机制是为了与标准 Fix 会话协议兼容，不能作为真正的消息恢复机制使用，会话对端应通过应用层的消息恢复机制来获得缺失的数据。

LFIXT 会话协议的实现者只在建立会话阶段存在消息序号同步，在会话持续期间不提供真正的消息恢复，而是简单地通过回应 SeqReset-Reset 消息来打发消息重传请求。

2.3.1 登录消息处理

LFIXT 会话协议的实现者作为会话接收方,只需将本方 NxtIn 设置为发起方 Logon 消息的 MsgSeqNum + 1，NxtOut 设置为发起方 Logon 消息中的 NextExpectedMsgSeqNum(789)即可。会话接收方不需要检查任何缺口，会话接收方也不会向发起方请求重传任何消息。如果发送方没有提供 NextExpectedMsgSeqNum 字段，则 NxtOut 设置为 1.

2.3.2 重传请求消息处理

作为 LFIXT 会话协议的实现者不会主动发送重传请求，但可能收到标准的 FIX 会话协议实现者发送的重传请求。当 LFIXT 会话协议的实现者收到重传请求时，会使用

6. 注销不影响任何订单的状况。所有有效的订单都可在注销（Logout）之后执行。

SeqReset-Reset 消息重置发送方序号，而不会提供历史消息的重传。

2.3.3 序号重设消息处理

LFIXT 会话协议的实现者收到序号重设消息时，会根据序号重设消息中的 NewSeqNo 来重置本方 NxtIn。

三、 消息定义

3.1 消息结构

每一条消息都由消息头、消息体、消息尾组成。消息总是由标准消息头开始，标准消息尾结束。

3.1.1 消息头

会话双方所有交换的消息具有如下标准的消息头。

每一个消息都由一个标准消息头开始。消息头定义了消息的类型，长度，目的地，顺序号，起始点和时间等数据域，均不加密传输。

其中有两个域用于消息重发。当作为会话级事件的结果而重复传送消息时，PossDupFlag 被设置为 Y，发送时沿用原来的消息序号；当[应用级]重新发送消息时，使用新的消息序号，并将 PossResend 设置为 Y。接收者应按以下方法处理上述消息：

PossDupFlag = Y：如果以前曾经收到过带有该消息序号的某条消息，则忽略本消息，如果不是，则按正常步骤处理。

PossResend = Y：不附带本标志地将消息传递给应用层，由其确定此前是否收到该消息

Tag	域名	必须	字段描述
8	BeginString	Y	起始串 **FIXT.1.1**(总是不加密，必须是消息的第一个域)
9	BodyLength	Y	消息体长度(总是不加密，必须是消息的第二个域)
35	MsgType	Y	消息类型(总是不加密，必须是消息的第三个域)
49	SenderCompID	Y	发送方代码字符串(总是不加密)
56	TargetCompID	Y	接收方代码字符串(总是不加密)
34	MsgSeqNum	Y	消息序号，整数类型
43	PossDupFlag	N	会话层可能重传标志，重复传送时，作此标记
97	PossResend	N	应用层可能重发标志。**LFIXT 会话层协议的实现者不会在出向消息中主动设置本字段，对于入向消息中出现的本字段将简单忽略**
52	SendingTime	Y	发送时间，UTCTimestamp 类型
347	MessageEncoding	N	消息中编码域的字符编码类型，固定为 GBK

3.1.2 消息尾

会话双方所有交换的消息具有如下标准的消息尾。每一个消息（管理或应用消息）

都用一个消息尾终止。消息尾可用于分隔多个消息，包含有 3 位数的校验和值。

Tag	域名	必须	字段描述
10	CheckSum	Y	校验和，总是消息的最末域，总是不加密

3.2 管理消息

LFIXT 会话协议支持标准 FIX 会话协议的所有管理消息，但不会主动发送所有管理消息。

LFIXT 会话协议分两种模式：精简模式和兼容模式，各自有不同的使用范围。

已知通信对手方为 LFIXT 会话协议实现者时，可以采用精简模式，此时只需要支持如下消息的接收和发送处理。此时由于本方不会主动发送 ResendRequest，因此根据协议也不会触发对方回应 SeqReset-Reset，因此不需要处理入向的 SeqReset-Reset 消息：

管理消息类型	来自对方	本方发送
Heartbeat	是	是
Logon	是	是
Reject	是	是
Logout	是	是

表1 精简模式：仅和 LFIXT 实现者通信时

当需要和基于标准 FIX 引擎的对手方进行通信时，必须采用兼容模式。此时 LFIXT 会话协议实现者需要支持所有管理消息的接收，但仍然不需要支持所有管理消息的发送。兼容模式和精简模式主要的区别只在于前者需要处理更多的入向管理消息。

管理消息类型	来自对方	本方发送
Heartbeat	是	是
Logon	是	是
TestRequest	是	否
ResendRequest	是	否
Reject	是	是
SeqReset-Reset	是	是
SeqReset-GapFill	是	否
Logout	是	是

表2 兼容模式：和标准 FIX 会话协议实现者通信时

一般而言，交易所端的 LFIXT 引擎应采用兼容模式以取得最大程度的协议兼容性。在已知交易所端采用兼容模式LFIXT协议的前提下,券商端可以选用精简模式的LFIXT协议，从而用最小的代价实现交易所接入，自然也可以考虑采用兼容模式甚至是标准FIX 会话层协议完成交易所接入。

必须注意：精简模式的 LFIXT 协议实现简便，但不具备和标准 FIX 协议互通的能力，券商必须全面衡量各系统的总体成本以确定实际使用的协议。

	标准 FIX 协议	兼容模式 LFIXT	精简模式 LFIXT
标准 FIX 协议	√	√	×
兼容模式 LFIXT		√	√
精简模式 LFIXT			√

表3 协议兼容性矩阵

3.2.1 Heartbeat 心跳消息（MsgType = 0）

会话双方使用 ***Heartbeat*** 消息来检测当前使用的 TCP 连接的状态，因此当一方处于数据发送空闲期时，需要定时发送 ***Heartbeat*** 消息以供检测链接的健康度。接收方如果在两倍的（[HeartBtInt]＋[合理传输时间] ）内没有收到来自对方的心跳消息，就认为连接失败，此时可以不发送 Logout 消息，立即关闭 TCP 连接。

会话协议不会主动发送 TestRequest 消息，但 LFIXT 会话协议的实现者，为了与标准 FIX 会话协议的兼容，如果收到了 TestRequest 请求，会按标准 FIX 会话协议向发送方返回心跳响应。

如果上层应用定义有应用层心跳信息，并以不低于 HeartBtInt 的间隔主动发送，则在事实上使得会话层心跳消息的发送条件永远得不到满足。在这种情况下，一方即使只收到应用层心跳消息，永远也收不到来自对方的会话层心跳消息，也不构成对本协议的违反。

Tag	域名	必须	字段描述
	Standard Header	Y	MsgType = 0
112	TestReqID	N	字符串。如是对 TestRequest 响应而发送的心跳消息，则应包含本域。本域的内容直接来自于触发本心跳消息的 TestRequest 消息的内容

	Standard Trailer	Y	

3.2.2 Logon 登录消息（MsgType = A）

登录消息应是请求建立一个会话的应用所发送的第一个消息。

HeartBtInt(108)域用来声明产生心跳的超时间隔（连接双方使用相同的值）。连接双方事先约定取值，由登录发起方产生，并得到登录接受方的确认响应。

当收到登录消息时，会话接受方将验证发起方身份的合法性，并且发出登录消息作为连接请求已被接受的确认。同样，确认的登录消息也可以被发起方用以验证连接是与正确的对方建立的。

会话接受方应在收到登录消息之后，立即作好开始消息处理的准备。

本标准规定：必须等到返回的登录消息收到之后才实施正常的消息交换。

LFIXT 会话协议实现着若作为会话发起方，其发送的 Logon 消息需将 MsgSeqNum 设置为 1、ResetSeqNumFlag 设置为 Y、NextExpectedMsgSeqNum 设置为 1。

标准 FIX 引擎若要向 LFIXT 协议实现者发起会话，且选择采用 NextExpectedMsgSeqNum(789)来实现序号同步，必须在 Logon 消息中将该字段填写为标准 FIX 引擎已保存的入向消息的最大序号 + 1。比如标准 FIX 引擎保存了入向消息 1-7，没有保存入向消息 8 和 9，但保存了入向消息 10，此时应该填写此字段为 11，而非 8。**这一点在 FIXT 协议中并未明确规定，故本文档对此予以特别限定。**

Tag	域名	必须	字段描述
	Standard Header	Y	MsgType = A
98	EncryptMethod	Y	加密方法 始终为 0，即不加密
108	HeartBtInt	Y	心跳间隔，单位是秒。双方必须用同一个值
141	ResetSeqNumFlag	N	双方序号重设回 1 的标志，布尔型
789	NextExpectedMsgSeqNum	N	接收方期望得到的下一条消息序号，可选。若不填写，认为取值为 1
553	Username	N	用户名
554	Password	N	密码
1137	DefaultApplVerID	Y	本次会话中使用的 FIX 消息的缺省版本。
1407	DefaultApplExtID	N	本次会话中使用的 FIX 消息[在 Tag1137 基础上]的缺省扩展包。

1408	DefaultCstmApplVerID	N	本次会话中，FIX 消息的缺省自定义应用版本。本标签是对 tag 1137 + tag 1407 的进一步约束。 本字段必须填写交易所发布的数据接口规范版本。如果接入方不提供该字段，接收方将不会允许接入方接入，并会通过 Logout 消息终止该会话。
	Standard Trailer	Y	

3.2.3 TestRequest 测试请求消息（MsgType = 1）

测试请求消息能强制对方发出心跳消息。会话协议不会主动发送该消息。测试请求消息的作用是检查对方消息序号和检查通信线路的状况。对方用带有测试请求标识符（TestReqID）的心跳作应答。LFIXT 协议的实现方不会主动发送任何 TestRequest 消息

Tag	域名	必须	注释
	Standard Header	Y	MsgType = 1
112	TestReqID	N	测试请求标识符
	Standard Trailer	Y	

3.2.4 Resend 重发请求消息（MsgType = 2）

重发请求消息由接收方发出，目的是向发送方申请某些消息重复发送。LFIXT 会话协议不会主动发送该消息。

重发请求消息有以下几种表示方式：

- 请求重发一条消息：BeginSeqNo= EndSeqNo
- 请求重发某个范围内的消息： BeginSeqNo=该范围中的第 1 条消息序号，EndSeqNo=该范围中的最后一条消息序号。注意请求重发一条消息只是本形式的特例。
- 请求重发某一特定消息之后的所有的消息：BeginSeqNo=该范围中的第 1 条消息， EndSeqNo=0（代表无穷大）

LFIXT 会话协议的实现者作为本消息的接收方时，只会通过 SeqReset-Reset 消息响应标准 FIX 会话协议实现者发送的重传请求消息。

Tag	域名	必须	注释
	Standard Header	Y	MsgType = 2
7	BeginSeqNo	Y	起始消息序号

16	EndSeqNo	Y	结束消息序号
	Standard Trailer	Y	

3.2.5 Reject 会话拒绝消息（MsgType=3）

当接收方收到一条消息，由于违反了**会话层规则**而不能适当地处理该消息时，应该发出 Reject（拒绝）消息。发出 Reject 消息可能是合适的一个例子是：收到了一条成功经过解码、校验和检查及 BodyLength 检查的消息，但该消息带有不合法的基本数据（比如：MsgType = &）。

规则要求：只要可能，消息应**尽可能**被转发给交易程序并通过业务层拒绝（而非会话层的 Reject 消息）来响应。若收到一条满足会话层规则的应用层消息，它必须在业务消息的层面被处理。若此处理过程发现了规则的违反，则应当产生一条业务层拒绝消息。许多业务层消息有特有的“拒绝”消息，应该使用这些消息。全部其他情况都可以通过“业务消息拒绝消息”进行拒绝。注意，即使收到了业务消息，且满足会话层规则，但当此消息不能被送达业务层处理系统的时候，也应该发送一条 BusinessRejectReason=“应用此刻不存在”的“业务消息拒绝”消息。

被拒绝的消息应该被记录日志，且入向序号应该增加。

和标准 FIX 会话协议不同之处在于：在 LFIXT 会话协议中，如果收到的消息是混乱的(garbled)，或者说：不能被解析或不能通过数据完整性检查时，应当记录日志后回送 Logout 消息并立刻断开连接（在标准 FIX 会话协议中建议忽略并丢弃本消息，以期望后续收到的正确格式的消息能触发一个消息缺口，从而能通过重传请求再度拿回这条混乱的消息）。同时也需要注意：这并非会话层 Reject 消息的使用场景。

产生和收到 Reject 消息，意味着出现了严重错误，可能发送方或接收方的应用存在逻辑错误。

如果发送方应用选择重新发送被拒绝的消息，应当使用新的消息序号并设置 PossResend = Y。因为 PossResend = Y 的语义是可能的应用层重发，因此这里被重新发送的一定是指应用层消息而非管理消息。

只要有可能，强烈推荐在本消息的 Text 字段中描述引起错误的原因。

当收到 Reject 消息本身时，建议记录日志，然后调整入向序号。

Tag	域名	必需	说明
	Standard Header	Y	MsgType=3

45	RefSeqNum	Y	关联消息的序号，即被拒绝消息(对方发送且己方收到)的序号。
371	RefTagID	N	相关错误消息中，出现错误的 FIX 域号
372	RefMsgType	N	相关错误消息的 MsgType
373	SessionRejectReason	N	会话拒绝原因编号
58	Text	N	文本，可解释拒绝的原因
	Standard Trailer	Y	

会话拒绝原因(SessionRejectReason)
0 =无效域号
1 = 该消息中必须的域丢失
2 = 该消息中出现未曾定义的域
3 = 未定义的域号
4 = 声明了域，但未赋值
5 = 此域的值错误（范围溢出）
6 = 取值格式错误
7 = 解密错误
8 = 签名错误
9 = CompID 错误
10 = 发送时间精度错误
11 = 无效的 MsgType
12 = XML 验证错误
13 =域多次出现（非重复组）
14 = 有序的域出现次序错误
15 = 重复组中，域次序错误
16 = 重复组中，NumInGroup 错误
17 = 非 data 数据域中，出现域界定符<SOH>
99 = 其他。注意可能存在其他的会话层规则违反，此时，“会话拒绝原因”99 可以被使用，进一步的信息应当包括在 Text 字段中。

表4 会话拒绝场景

3.2.6 SeqReset 序号重设消息（MsgType = 4）

序号重设消息由发送方发出，用于告知接收方下一个消息的消息序号。序号重设消息有两种模式：序号重设-缺口填补（SeqReset-Gap Fill）；序号重设-重设（SeqReset -Reset）。

在 LFIXT 会话协议中，只使用 SeqReset-Reset 消息回应 ResendRequest 消息，此 SeqReset-Reset 消息的 MsgSeqNum 按标准 FIX 协议规定可以任意填写且接收方不会检查，在 LFIXT 中规定固定填写 1，其中的 NewSeqNo 则建议填写为 max{ResendRequest 消息的 EndSeqNo 字段值，本方 NxtOut 值}。序号重设只能增加消息序号。如果收到的序号重设消息试图使下一个预期的消息序号变小，那么此消息应被拒绝接受，并被视作为严重错误。

在 LFIXT 会话协议中，虽然不会主动发起重传请求，但仍然有可能收到标准 FIX 协议的对手方主动发出的 SeqReset 类消息。

当收到的序号重设消息为 SeqReset-GapFill 消息时，必须确保 NewSeqNo 必须大于等于 SeqReset-GapFill 消息本身的 MsgSeqNum + 1 且不允许 NewSeqNo > NxtIn，此时本方 NxtIn 保持不变；否则，被视为严重错误。

当收到 SeqReset-Reset 消息时，则需要忽略 MsgSeqNum 的检查，同时令本方 NxtIn = NewSeqNo。

Tag	域名	必须	注释
	Standard Header	Y	MsgType = 4
123	GapFillFlag	N	缺口填补标志。布尔型。“Y”表明是 Gap Fill 模式;“N”或不存在本域，表明是 Reset 模式。
36	NewSeqNo	Y	新消息序号
	Standard Trailer	Y	

3.2.7 Logout 注销消息（MsgType = 5）

注销消息是发起或确认会话终止的消息。未经注销消息的交换而断开连接，一律视为非正常的断开。

在最后终止会话之前，注销的发起人应该等待连接对方确认注销消息。如果连接对方没有在适当的时间间隔里作回应，那么会话就可以终止。

注销发起人在发送注销消息之后不应发送任何消息，除非接收到连接对方发出的重传请求消息。

何时发送 Logout，何时直接断开

一般而言，在关闭连接前推荐的做法是回送一条 Logout 消息以利于对端诊断连接断开的原因。但此处可有一些例外：

1. 若收到的登录报文中，SenderCompID, TargetCompID 或者会话发起方的 IP 地址不合法，推荐做法是会话立即终止，不发送任何 Logout 消息。原因是此种登录尝试很可能属于未被授权的系统入侵行为，因此不应当回送 Logout 消息以泄露 FIX 版本、合法 SenderCompID、TargetCompID 等信息。

2. 在标准 FIXT 协议中，在收到登录请求时若在同一个 FIX 引擎若已存在一个合法的相同 KEY 的会话，则标准 FIX 引擎将直接断开后来的登录请求所欲发起的会话。

由于任何时刻总是存在网络中断的可能，因此 LFIXT 协议的任何参与方都应该对于没有收到对方的 Logout 消息但底层 TCP 连接却已关闭的情况做好准备。在断开连接前回送 Logout 消息是推荐行为，但非必须。

在此简要总结一下：对于收到报文中的各类异常接收方通常处理的原则。

1. TCP 通信异常/潜在的恶意攻击

推荐处理方式是直接断开 TCP 连接。由于在任何情况下通信参与者都必须能够处理 TCP 连接的中断，因此这也是缺省处理办法。

对于同一个 TCP 连接上，第一个收到的报文不是正确的 Logon 报文，或者在已经 Logon 成功的 TCP 连接上又收到一个普通的 Logon 消息，视作恶意攻击，直接断开。

2. 混乱的报文、入向消息序号出现缺口

推荐处理方式是回送 Logout 消息，然后断开 TCP 连接

3. 报文不混乱，但不满足会话层规则

推荐处理方式是发送 Reject 拒绝此消息，会话继续

4. 报文不混乱，满足会话层规则

交给应用层去处理。如果应用层发现报文违反了应用层规则，回送应用层拒绝消息，会话继续

Tag	域名	必须	字段描述
	Standard Header	Y	MsgType = 5

1409	SessionStatus	N	Logout 时的会话状态。根据 FIX 协议有如下取值: 0 = 会话活跃 1 = 会话口令已更改 2 = 将过期的会话口令 3 = 新会话口令不符合规范 4 = 会话退登完成 5 = 不合法的用户名或口令 6 = 账户锁定 7 = 当前时间不允许登录 8 = 口令过期 9 = 收到的 MsgSeqNum(34)太小 10 = 收到的 NextExpectedMsgSeqNum(789)太大. 100 及 100 以上的数字可供用户在双边认可的情况下自定义
58	Text	N	文本。注销原因的进一步补充说明。
	Standard Trailer	Y	

四、 数据字典

4.1 数据类型

数据类型	类型定义	说明
SeqNum	N18	消息序号 正整数
Boolean	C1	'Y' = Yes/True, 'N' = False/No
Length	N9	长度 表示字节为单位的数据长度，非负数
UTCTimeStamp	C21	UTC 时间戳，注意，FIX 协议允许使用 YYYYMMDD-HH:mm:SS, YYYY = 0000-9999, MM = 01-12, DD = 01-31, HH = 00-23, mm = 00-59, SS = 00-60（秒）， 或 YYYYMMDD-HH:mm:SS.sss(毫秒), YYYY = 0000-9999, MM = 01-12, DD = 01-31, HH = 00-23, mm = 00-59, SS = 00-60（秒）， sss = 000-999（毫秒） 之中的任何一种格式
LocalTimeStamp	C21	本地时间戳 YYYYMMDD-HH:mm:SS.sss(毫秒),

		YYYY = 0000-9999, MM = 01-12, DD = 01-31, HH = 00-23, mm = 00-59, SS = 00-60（秒）, sss = 000-999（毫秒）
NumInGroup	N9	重复数 表示重复组的项数，正数

注：除非特别声明，浮点数类型均有正负，类型定义 Nx(y)中，x 表示整数与小数总计位数的**最大值**，不包括小数点，y 表示固定的小数位数。类型定义 Cx 中，x 表示字符串的**最大**长度。

4.2 会话层域定义

Tag	域名	类型	说明
7	BeginSeqNo	SeqNum	起始消息序号
8	BeginString	C16	起始串 固定为 FIXT.1.1
9	BodyLength	Length	消息体长度
10	CheckSum	C3	校验和 不加密，必须是最后一个域 校验算法参见 STEP 协议
16	EndSeqNo	SeqNum	结束消息序号
34	MsgSeqNum	SeqNum	消息序号，由 1 开始
35	MsgType	C16	消息类型
36	NewSeqNo	SeqNum	新消息序号
43	PossDupFlag	Boolean	指示该消息序号的消息可能重复发送，取值: =Y: 可能重复 =N: 首次发送
49	SenderCompID	C32	接收方代码
52	SendingTime	UTCTimestamp	消息发送时间，UTCTimestamp 类型
56	TargetCompID	C32	发送方代码
58	Text	C1024	文本
97	PossResend	Boolean	可能重发标志 指示该消息可能发送过（使用不同的消息序号），取值: =Y: 可能重发 =N: 首次发送
98	EncryptMethod	N8	加密方法 =0: 即不加密
108	HeartBtInt	N8	心跳监测的时间间隔 为系统设定值,以秒为单位
114	ResetSeqNumFlag	Boolean	序号重设标志

123	GapFillFlag	Boolean	缺口填补标志 用于序号重设消息，指示是否填补缺口，取值： =Y: 序号重设-缺口填补消息，消息序号域有效 =N 或不存在序号重设-重设消息，消息序号域无效
347	MessageEncoding	C16	消息中编码域的字符编码类型 =GBK
383	MaxMessageSize	Length	最大消息长度，单条消息的最大字节数。 该字段暂未启用
553	UserName	C32	用户名
554	Password	C32	密码
789	NextExpectedMsgSeqNum	SeqNum	接收方期望得到的下一条消息序号
1137	DefaultApplVerID	C8	本次会话中使用的 FIX 消息的缺省版本。
1407	DefaultApplExtID	N8	本次会话中使用的 FIX 消息[在 Tag1137 基础上]的缺省扩展包。
1408	DefaultCstmApplVerID	C32	本次会话中，FIX 消息的缺省自定义应用版本。本标签是对 tag 1137 + tag 1407 的进一步约束。 本字段填写交易所发布的数据协议版本。建议的格式是：STEPn.xy_市场代码_市场内部唯一的协议版本号。市场代码是 SH 时，代表上海证券交易所；是 SZ 时，代表深圳证券交易所。市场内部唯一的协议版本号由各市场自行分配。
1409	SessionStatus	N4	本字段填写退登时的会话状态

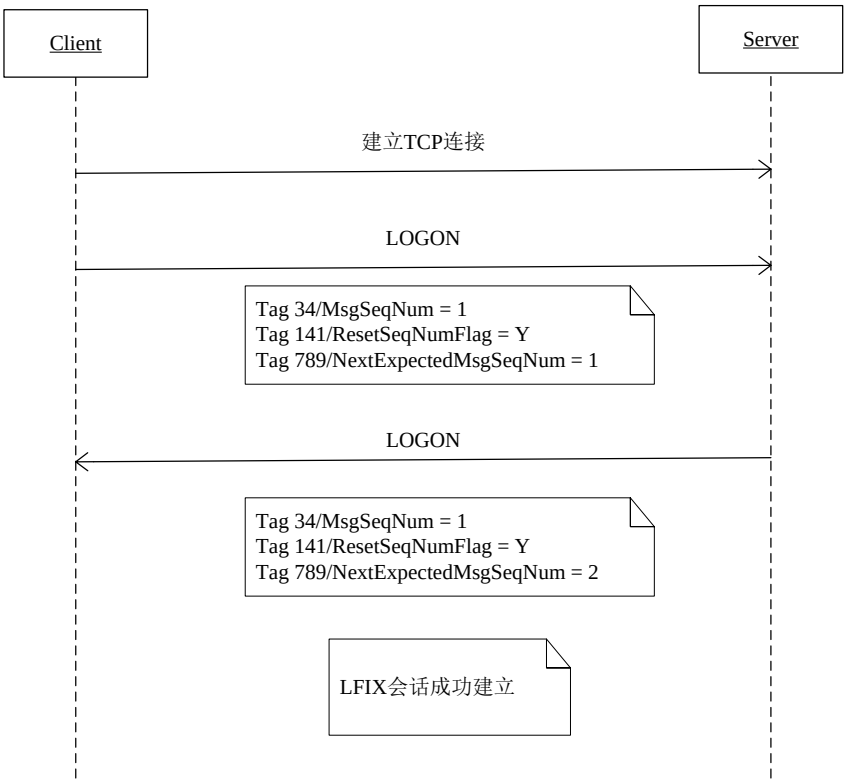
附 录 A

（登录场景）

正常登录场景一

LFIX作为登录发起方——Client，LFIX作为登录接受方——Server，日间正常登录。

场景开始时，TCP连接刚建立时，Client 的 NxtOut=1, NxtIn = 1，Server 的 NxtOut = 1, NxtIn = 1。



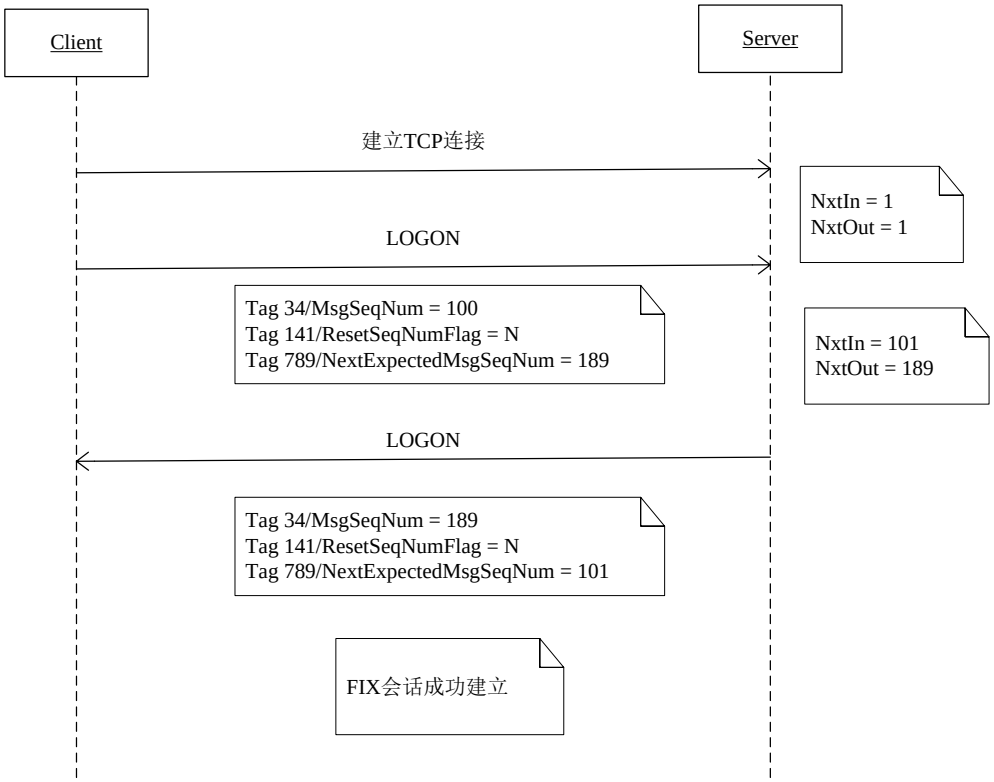
图A.1 正常登录场景一

FIX会话建立后，Client 的 NxtOut=2, NxtIn = 2，Server 的 NxtOut = 2, NxtIn = 2。

正常登录场景二

标准FIX作为登录发起方——Client，LFIX作为登录接受方——Server，日间非首次正常登录，且Client不设置ResetSeqNumFlag，设置NextExpectedMsgSeqNum为其在前一次连接断开时的NxtIn。

场景开始时，TCP连接刚建立后，Client 的 NxtOut=100, NxtIn = 189，Server 的 NxtOut = 1, NxtIn = 1。在此之前，Server曾经向Client发送过MsgSeqNum 为189， 190的业务消息，但因为通信故障，Client没有收到，所以Client端保存的NxtIn仍然是189，而非191。



图A.2　正常登录场景二

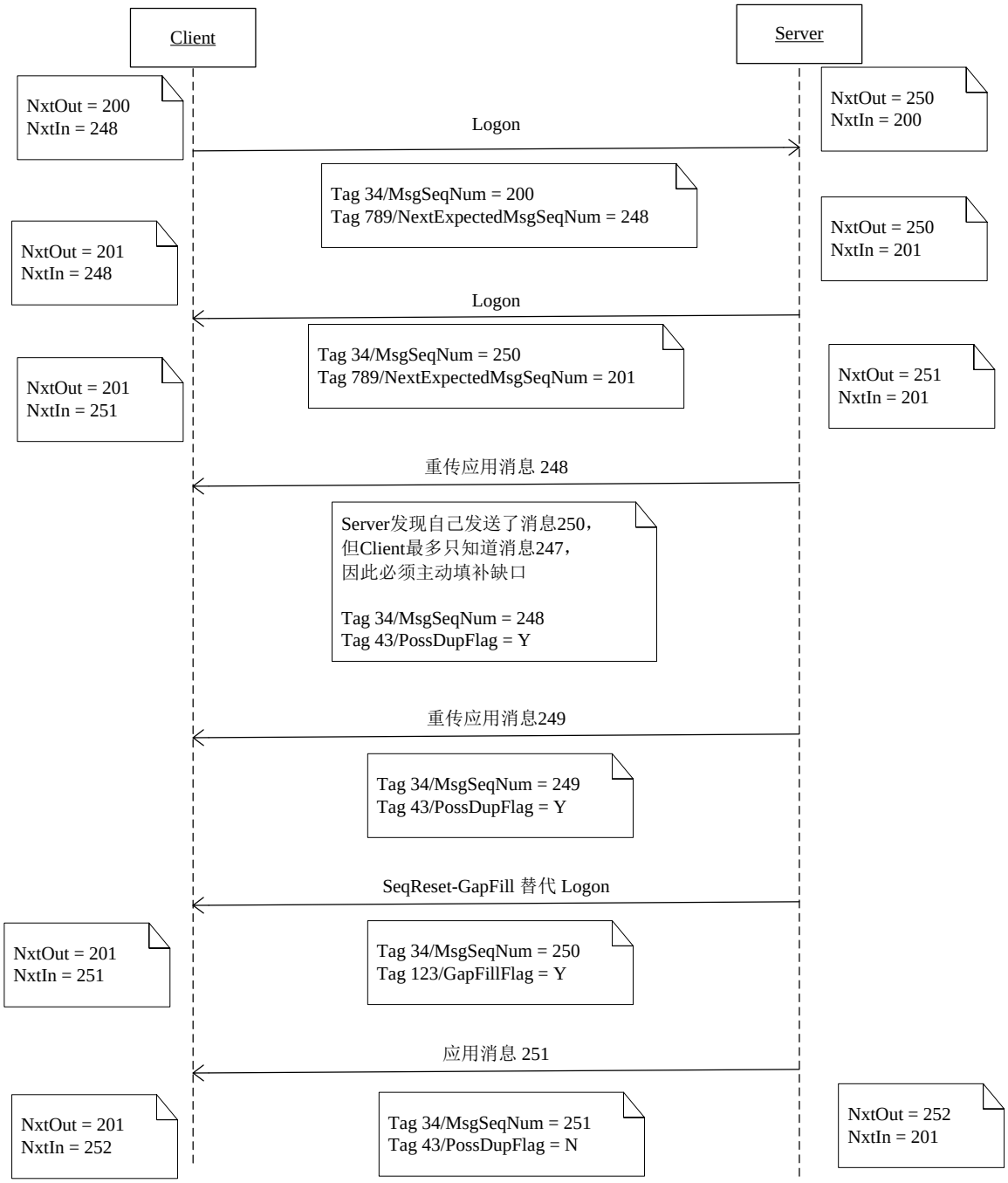
FIX会话建立后，Client 的 NxtOut=101, NxtIn = 190，Server 的 NxtOut = 190, NxtIn = 101。注意两次TCP连接上Server发送给Client的MsgSeqNum = 189的消息并不相同，但**因为Client没有保存之前收到的MsgSeqNum = 189 的业务消息，所以第二次收到MsgSeqNum = 189的Logon应答消息也不会发现存在不一致**。

正常登录场景三

为了说明在标准FIX协议中，NextExpectedMsgSeqNum相关的自动缺口填补的原理，撰写此场景。本场景中，通信双方都遵循标准FIX协议。

在场景开始的时候，Server的 NxtOut是250，NxtIn是200；由于Server和Client之间出现了网络中断，因此Client只收到了序号为247（含）以前的来自Server的消息,其NxtOut是200，NxtIn是248。

Server收到的来自Client的登录消息中，NextExpectedMsgSeqNum是248，但Server端内部下一个发送的应该是LOGON消息，必须使用新的序号250。但Server端也知道Client端出现了消息缺口，且根据协议Client不会为此缺口发送ResendRequest消息，而是需要Server端主动推送缺口部分的消息。因此Server端马上重复传送业务消息248，业务消息249，并用SeqReset-GapFill消息替代消息250——Logon消息，之后就可以开始正常传送业务消息251。

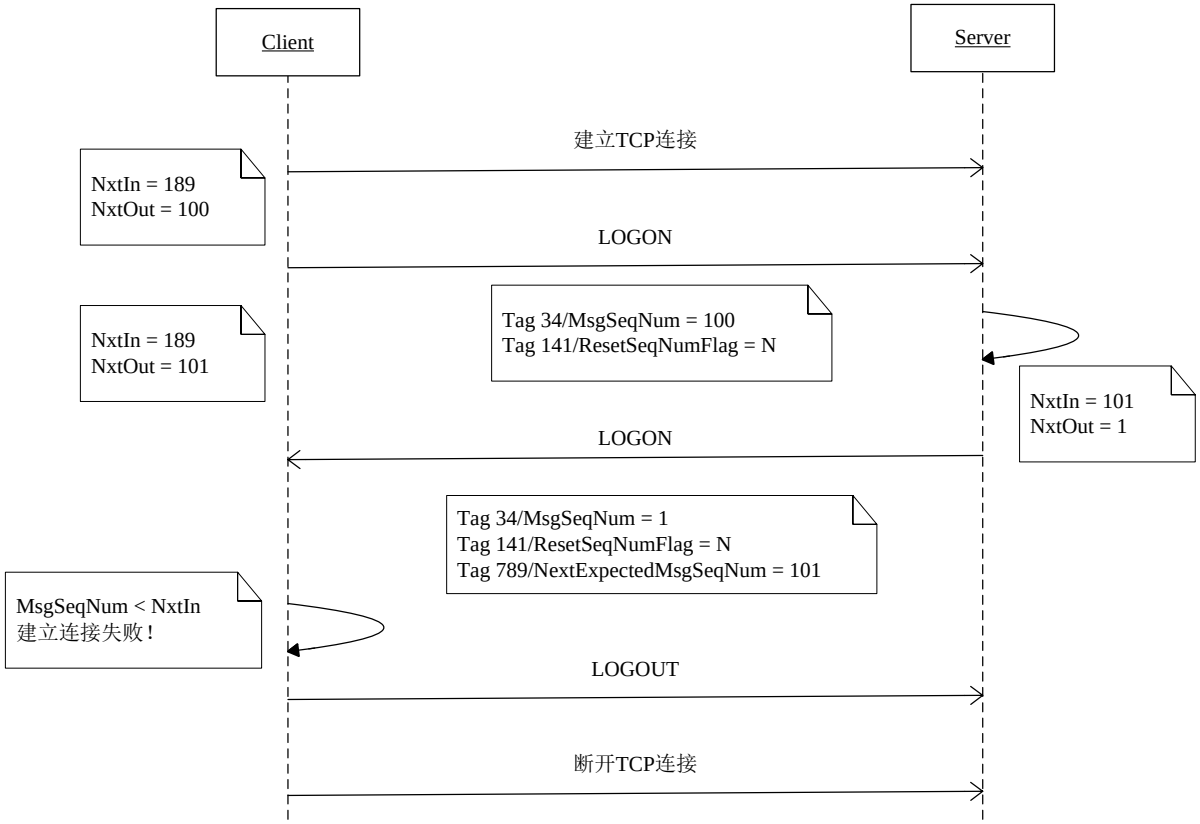


图A.3　正常登录场景三

异常登录场景一

标准FIX作为登录发起方——Client，LFIX作为登录接受方——Server，日间非首次正常登录，且Client不设置ResetSeqNumFlag，也没有设置NextExpectedMsgSeqNum为其在前一次连接断开时的NxtIn。

场景开始时，TCP连接刚建立后，Client 的 NxtOut=100, NxtIn = 189，Server 的 NxtOut = 1, NxtIn = 1。

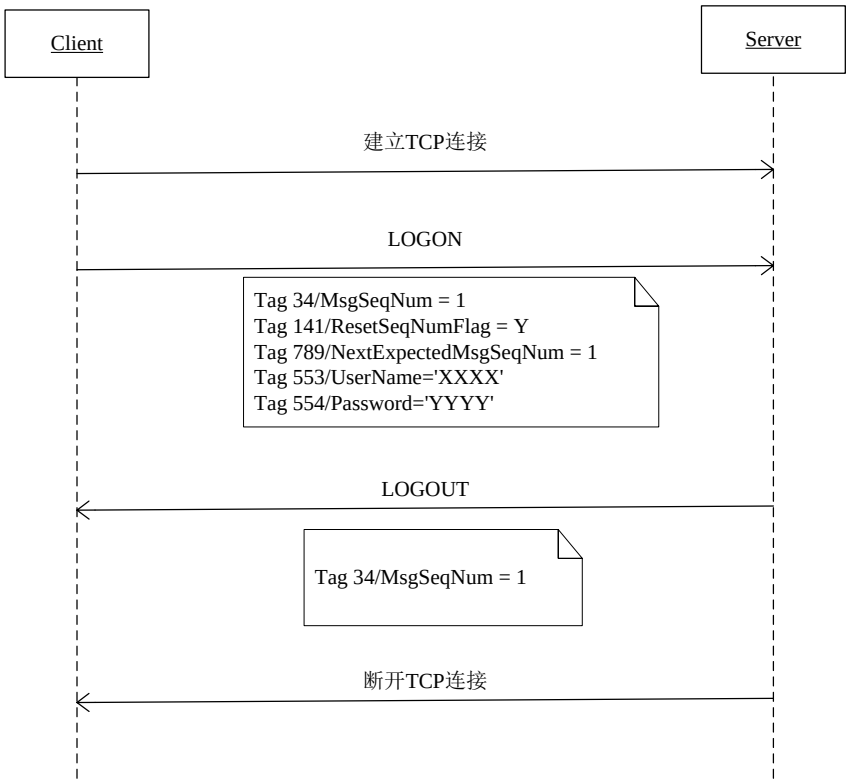


图A.4 异常登录场景一

由于标准FIX客户端没有正确设置NextExpectedMsgSeqNum，收到的LOGON响应报文的MsgSeqNum是1，小于预期的189，因此标准FIX客户端认为出现了严重错误，在发送LOGOUT后断开连接。

异常登录场景二

LFIX作为登录发起方——Client，LFIX作为登录接受方——Server，其余情况同本节场景一，唯一的区别在于Client在LOGON消息中提供的用户名和口令非法。



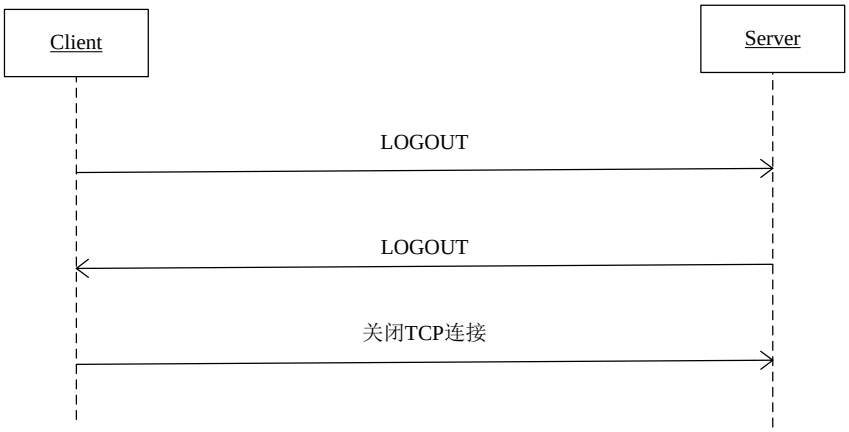
图A.5 异常登录场景二

附 录 B

（注销场景）

正常注销场景

下图显示注销会话的场景，申请注销后，服务方回送注销消息确认断开会话。



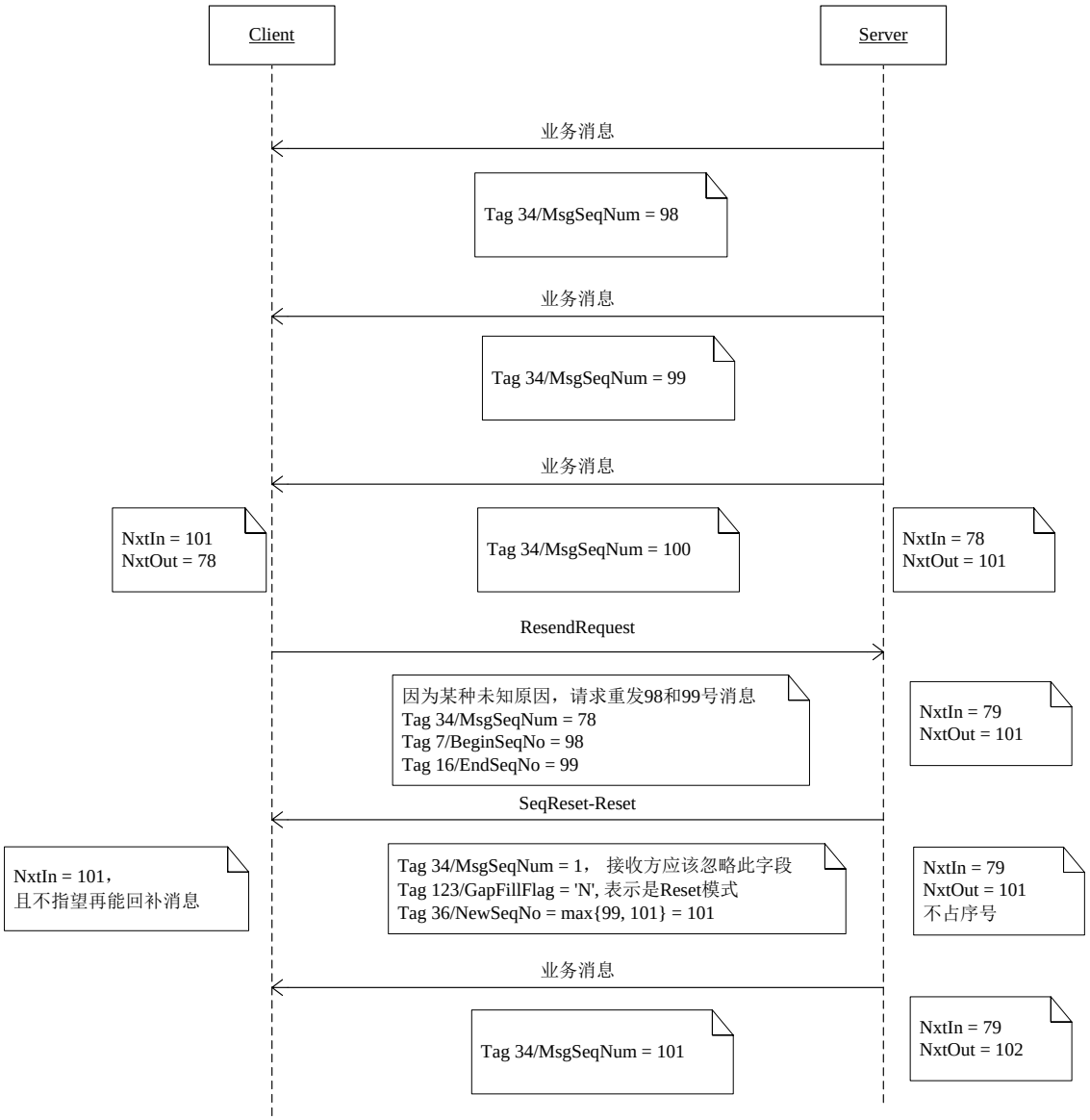
图B.1 正常注销场景

附 录 C

（处理重传请求场景）

处理重传请求场景一

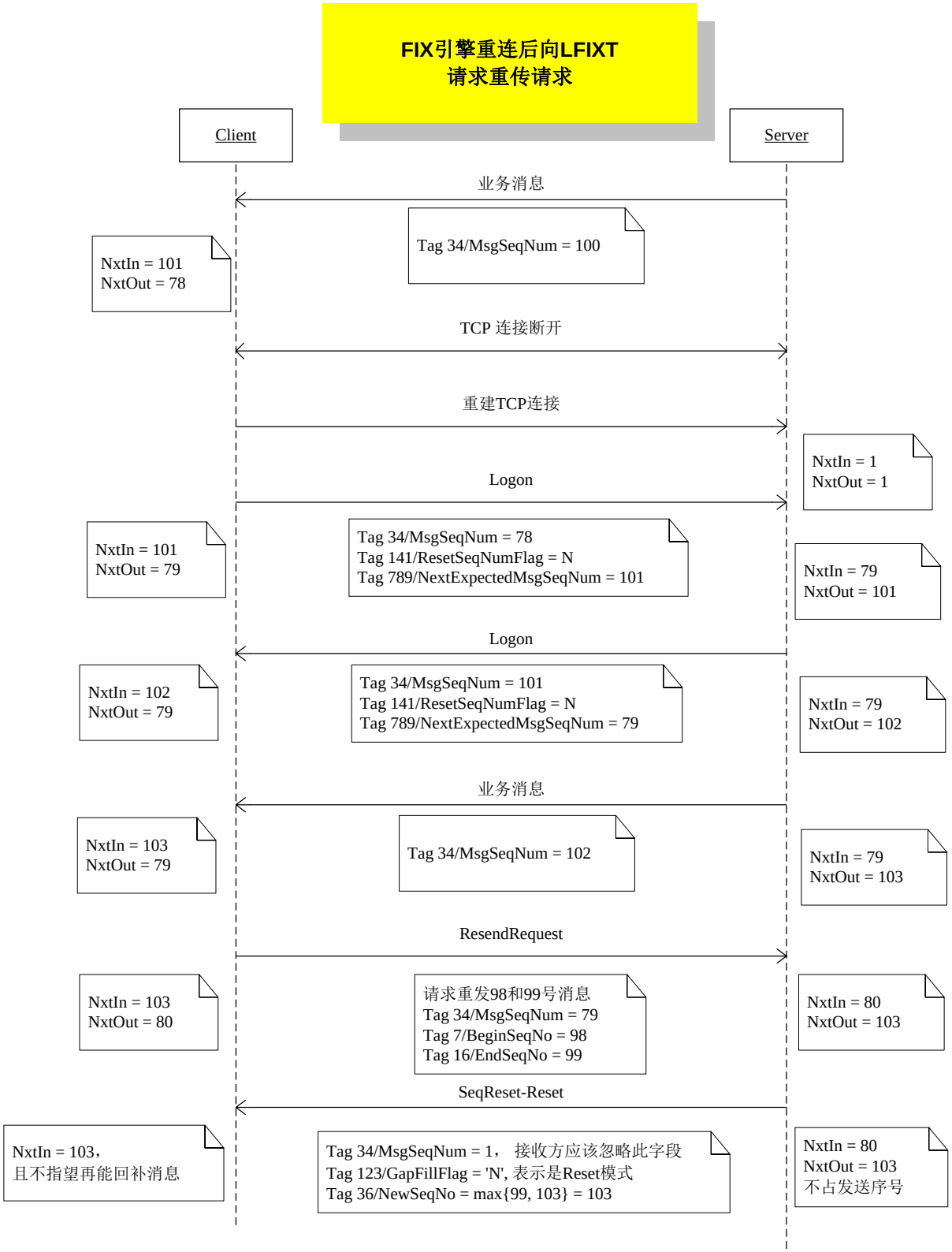
下图中，Client是标准FIX引擎，Server是LFIX引擎，且Client因为某种特殊的原因，比如虽然收到但因为自身错误而没有将消息98和99保留下来，但保存了消息100。



图C.1 处理重传请求一

处理重传请求场景二

和本节场景一前面部分都相同，但区别在于Client没有来得及发出ResendRequest，网络连接就断了，因此需要重新登录并后续发送ResendRequest补缺口。在登录时使用的NextExpectedMsgSeqNum 是根据登录时已知的最大消息序号确定，而非根据全部缺口消息中最小的序号来确定。


FIX引擎重连后向LFIXT
请求重传请求
Client
Server
业务消息
Tag 34/MsgSeqNum = 100
NxtIn = 101
NxtOut = 78
TCP 连接断开
重建TCP连接
NxtIn = 1
NxtOut = 1
Logon
NxtIn = 101
NxtOut = 79
Tag 34/MsgSeqNum = 78
Tag 141/ResetSeqNumFlag = N
Tag 789/NextExpectedMsgSeqNum = 101
NxtIn = 79
NxtOut = 101
Logon
NxtIn = 102
NxtOut = 79
Tag 34/MsgSeqNum = 101
Tag 141/ResetSeqNumFlag = N
Tag 789/NextExpectedMsgSeqNum = 79
NxtIn = 79
NxtOut = 102
业务消息
NxtIn = 103
NxtOut = 79
Tag 34/MsgSeqNum = 102
NxtIn = 79
NxtOut = 103
ResendRequest
NxtIn = 103
NxtOut = 80
请求重发98和99号消息
Tag 34/MsgSeqNum = 79
Tag 7/BeginSeqNo = 98
Tag 16/EndSeqNo = 99
NxtIn = 80
NxtOut = 103
SeqReset-Reset
NxtIn = 103，
且不指望再能回补消息
Tag 34/MsgSeqNum = 1， 接收方应该忽略此字段
Tag 123/GapFillFlag = 'N', 表示是Reset模式
Tag 36/NewSeqNo = max{99, 103} = 103
NxtIn = 80
NxtOut = 103
不占发送序号

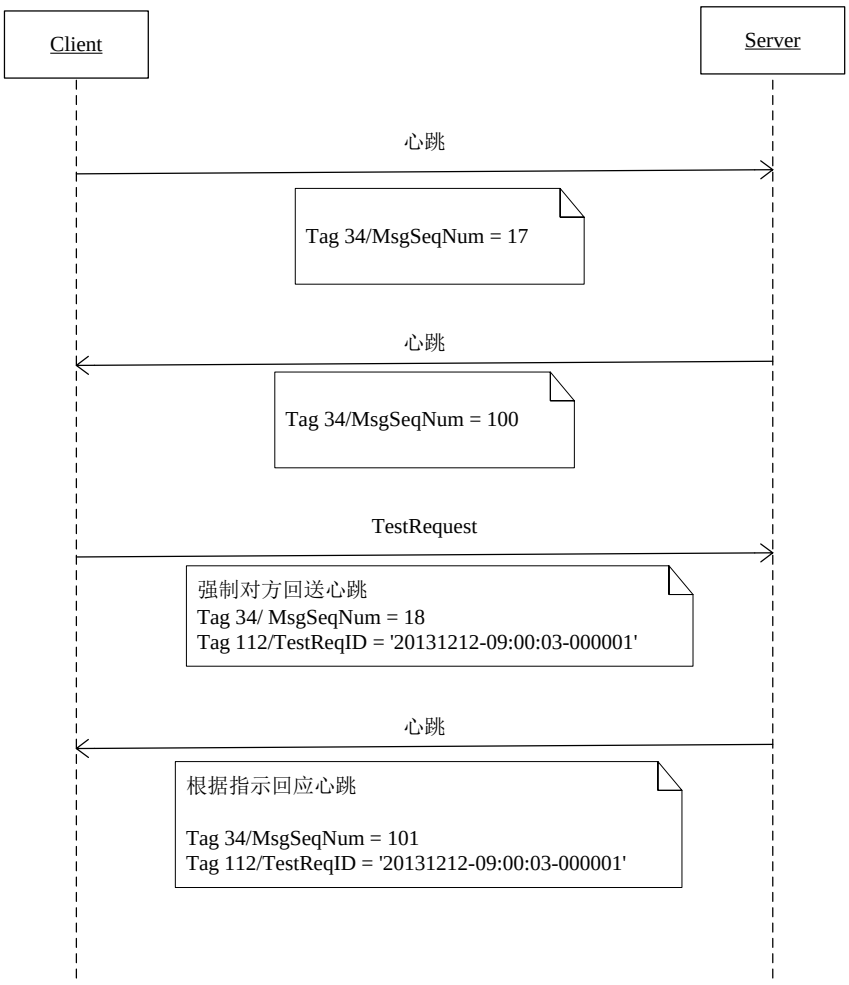
图C.2 处理重传请求二

附 录 D

（处理心跳和测试请求）

处理心跳和测试请求

下图是心跳和测试请求的场景，连接双方的空闲持续在经过一个约定的时间间隔后，连接双方根据规则都可以发送心跳。标准 FIX 协议的实现者可以不受限制地主动发起测试请求。下图中的 Server 是 LFIX 协议的实现者，不会主动发起测试请求。



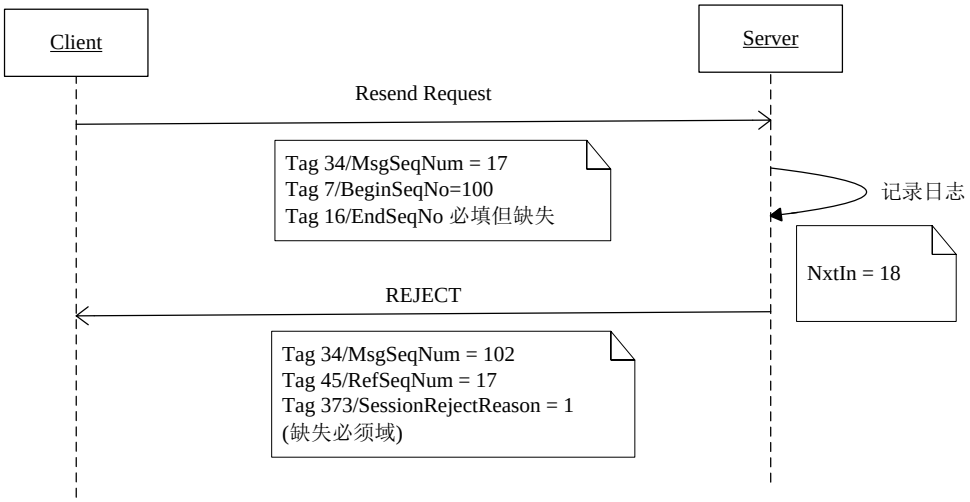
图D.1 处理心跳和测试请求

附 录 E

（处理会话拒绝）

处理会话拒绝

LFIX会话协议中，收到会话拒绝后应该记录日志，调整入向消息序号。下图中，Server是LFIX引擎，Client是标准的FIX引擎。



图E.1 处理会话拒绝

附 录 F

（计算校验和）

计算校验和

以下为计算校验和的代码段：

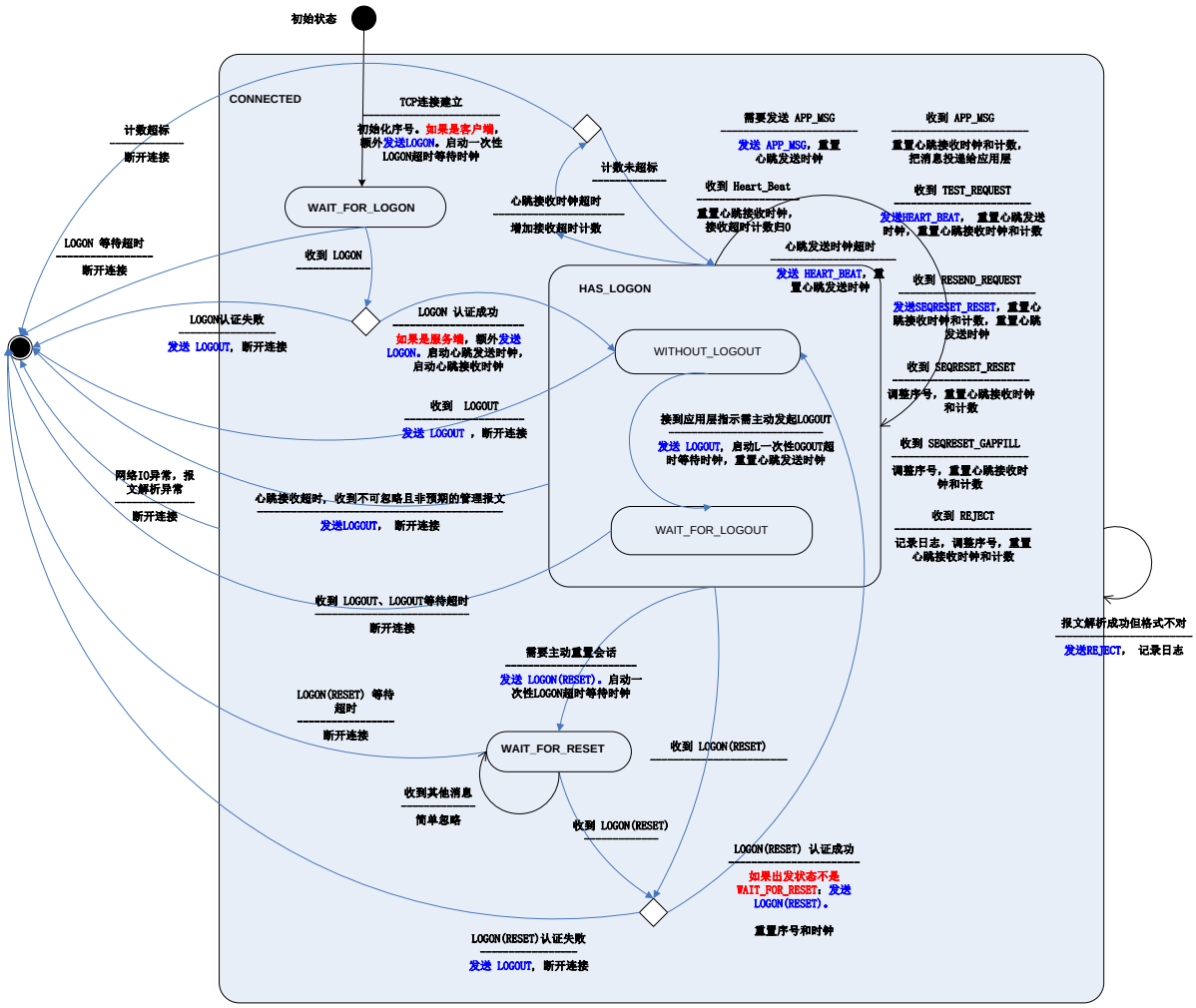
```
char *GenerateCheckSum( char *buf, long bufLen )
{
static char tmpBuf[ 4 ];
    long idx;
    unsigned int cks;

    for( idx = 0L, cks = 0; idx < bufLen; cks += (unsigned int)buf[ idx++ ] );
    sprintf( tmpBuf, "%03d", (unsigned int)( cks % 256 ) );
    return( tmpBuf );
}
```

附 录 G

（状态转换参考图）

LFIXT会话协议状态转换参考图



此图可以作为 LFIX 引擎实现的参考，但不能替代真实 LFIX 引擎开发的设计文档。若和正文有不一致，一律以协议正文为准。

附 录 H

（实现参考）

LFIXT会话协议实现参考

本参考若和协议正文不一致，一律以协议正文为准。

本参考按所收到的不同消息分别说明所需进行的最基本校验，以及在通过完整校验后的最基本处理。所谓“最基本校验”和“最基本处理”，并不意味着仅仅实现这些内容就足以构成一个正确的 LFIXT 引擎，而是说任何正确的 LFIXT 引擎都必须覆盖这些校验和处理。因此它们是正确的 LFIXT 引擎的必要但不充分条件。

根据标准 FIX 协议正文，有些管理消息的 PossDupFlag 不可以是 Y，但同时也规定了接收方要能够容忍对方错误地发出此种消息，但收到此种消息时只将其用于序号处理，而不管消息本身的类型和内容。本参考中，对于标注了“**PossDupFlag 不能是 Y**”的入向消息，如果真的收到了 PossDupFlag 被设置为 Y 的此类消息，约定先在内部将其转换为一条 PossDupFlag 是 Y 的 SeqReset-GapFill 入向消息后，再按 SeqReset-GapFill 消息进行处理。

除了已知的若干种特别情形以外，一律不允许入向消息的 MsgSeqNum 出现跳空缺口，而是必须连续递增。

1. Logon消息

收到消息类型： Logon

LFIX 引擎可以发出： 是

消息的最基本校验：

PossDupFlag 不能是 Y

如果是带 RESET 的 LOGON 消息—— ResetSeqNumFlag =Y

不需要校验 MsgSeqNum 是否出现逆转

如果该消息是 LOGON 的发起消息，必须 MsgSeqNum = 1, NextExpectedMsgSeqNum = 1

如果该消息是 LOGON 的响应消息，必须 MsgSeqNum = 1, NextExpectedMsgSeqNum = 2

否则

也不需要校验 MsgSeqNum 是否出现逆转，因为作为 TCP 建立后收到的第一个消息，

接收方的 NxtIn = 1, NxtOut = 1，因此 MsgSeqNum <NxtIn 不可能发生。

必须注意到：NextExpectedMsgSeqNum < NxtOut 同样也是不可能发生的。

消息的最基本处理：

如果是带 RESET 的 LOGON 消息，处理方式见正文。

对于其他 LOGON 消息，令 NxtOut = NextExpectedMsgSeqNum，NxtIn = MsgSeqNum + 1

2. Heartbeat消息

收到消息类型： Heartbeat

LFIX 引擎可以发出： 是

消息的最基本校验：

PossDupFlag 不能是 Y

MsgSeqNum <> NxtIn 意味着严重错误

消息的最基本处理：

NxtIn = MsgSeqNum + 1

清接收超时计时器和计数

3. TestRequest消息

收到消息类型： TestRequest

LFIX 引擎可以发出： 否

消息的最基本校验：

PossDupFlag 不能是 Y

MsgSeqNum <> NxtIn 意味着严重错误

消息的最基本处理：

NxtIn = MsgSeqNum + 1

回送 Heartbeat 消息，清发送超时计时器和计数，清发送超时计时器，

4. ResendRequest消息

收到消息类型： ResendRequest

LFIX 引擎可以发出： 否

消息的最基本校验：

PossDupFlag 不能是 Y

MsgSeqNum <> NxtIn 意味着严重错误

如果 EndSeqNo > 0

必须保证 BeginSeqNo <= EndSeqNo<NxtOut

如果 EndSeqNo == 0

必须保证 BeginSeqNo < NxtOut

消息的最基本处理：

NxtIn = MsgSeqNum + 1

发送 SeqReset_Reset 作为应答，其 MsgSeqNum 固定是 1，

NewSeqNo 设置为 NxtOut。

发送 SeqReset_Reset, NxtOut 保持不变

5. SeqReset-Reset消息

收到消息类型： SeqReset-Reset

LFIX 引擎可以发出： 是

消息的最基本校验：

PossDupFlag 必须是 Y

不需要校验 MsgSeqNum，但 NewSeqNo 必须>=接收方的 NxtIn 才可以，否则是严重错误

消息的最基本处理：

NxtIn = NewSeqNo

6. SeqReset-GapFill消息

收到消息类型： SeqReset-GapFill

LFIX 引擎可以发出： 否

消息的最基本校验：

PossDupFlag 必须是 Y

NewSeqNo 必须大于等于 SeqReset-GapFill 消息本身的 MsgSeqNum + 1，否则是严重错误。

NewSeqNo> NxtIn 是一个严重错误

消息的最基本处理：

NxtIn 保持不变

7. Reject消息

收到消息类型： Reject

LFIX 引擎可以发出： 是

消息的最基本校验：

若 MsgSeqNum < NxtIn 且 PossDupFlag != Y ， 意味着严重错误

MsgSeqNum < NxtIn 意味着严重错误

消息的最基本处理：

记录日志

NxtIn = MsgSeqNum + 1

8. Logout消息

收到消息类型： Logout

LFIX 引擎可以发出： 是

消息的最基本校验：

PossDupFlag 不能是 Y

MsgSeqNum < NxtIn 意味着严重错误

消息的最基本处理：

如果在本消息校验过程中发现错误：

如果本方是 Logout 过程的发起方

则马上断开连接

否则

在回应的 Logout 消息中告知对方校验错误

发送后立刻断开 TCP 连接

否则

如果本方是 Logout 过程的发起方

可以马上断开连接。

否则，

发送正常 Logout 消息后立即断开 TCP 连接

9. 应用消息

收到消息类型：应用消息

LFIX 引擎可以发出：是

消息的最基本校验：

若 MsgSeqNum < NxtIn 且 PossDupFlag != Y ，意味着严重错误

若 MsgSeqNum > NxtIn，意味着严重错误

消息的最基本处理：

NxtIn = MsgSeqNum + 1