ICS 03.060
A11

JR

中华人民共和国金融行业标准

JR/T 0182—2020

轻量级实时STEP消息传输协议

Light weight realtime STEP message transfer protocol

2020-02-26 发布　　2020-02-26 实施

中国证券监督管理委员会　　发布

目　　次

前　言

本标准依据GB/T　1.1—2009给出的规则起草。

本标准由全国金融标准化技术委员会证券分技术委员会（SAC/TC180/SC4）提出。

本标准由全国金融标准化技术委员会（SAC/TC180）归口。

本标准起草单位：中国证券监督管理委员会信息中心、上海证券交易所、深圳证券交易所、中证信息技术服务有限责任公司。

本标准主要起草人：姚前、刘铁斌、周云晖、杜娟、朱立、王鹏飞、李向东。

引　言

FIX（Financial Information eXchange）协议即金融信息交换协议，是用于全球金融市场机构和参与者间的电子金融信息交互的协议，以下简称FIX协议。

在FIX标准化组织(FIX Trading Community)提出的FIX 5.0标准中，单独定义有一个会话层协议（FIX Session Protocol，FIXT），其版本是FIXT 1.1。JR/T 0022-2014《证券交易数据交换协议》（Securities Trading Exchange Protocol，简称STEP)是FIX协议的行业标准。本标准是对FIXT 1.1及JR/T 0022-2014中会话层协议的轻量化。

轻量级实时STEP消息传输协议

1 范围

本标准规定了轻量级实时STEP消息传输协议（Light-weight realtime STEP message transfer protocol）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、消息格式规范、数据字典等内容。

本标准适用于支持证券交易所与市场参与者和相关金融机构间的业务数据交换。本标准也可被推广用于支持证券期货行业各机构系统间的会话连接。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅所注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

JR/T 0022—2014 证券交易数据交换协议

ISO 10383:2012 证券及相关金融工具 交易所和市场识别码（Securities and related financial instruments-Codes for exchanges and market identification(MIC)）

Financial information exchange protocol(FIX) Version 5.0 Service Pack 2, August 2011

Financial information exchange protocol(FIX) FIX session protocol version1.1, March 2008

3 术语和定义

下列术语和定义适用于本文件。

3.1

证券交易数据交换协议 securities trading exchange protocol;STEP

证券交易所交易系统与市场参与者系统之间进行证券交易数据交换的行业信息标准。

3.2

金融信息交换协议 financial information exchange;FIX

进行证券交易信息电子交换的国际信息标准。

3.3

FIX会话层协议 fix session layer protocol;FIXT

独立于通信协议（如TCP、UDP等）和通信介质（如光缆、卫星等）的电子数据传输协议，FIX协议的重要子协议。

4 会话机制

4.1 关键概念的重要说明

4.1.1 会话层重传

4.1.1.1 本标准给出了市场参与者内部系统通过开放接口与证券交易所间的会话层连接标准。本标准是FIX标准化组织的FIXT 1.1标准的精简版，且消息标签的使用遵循FIX 5.0 SP2的规定。由于JR/T 0022-2014证券交易数据交换协议只是FIX协议的本地化版本，为凸显标准的直接来源，以下简称本标准为“LFIXT”。

4.1.1.2 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本标准而实现的应用程序或模块，以下简称为“LFIXT参与方”。基于FIXT开发的程序或模块（以下简称为“FIXT参与方”），若要和LFIXT参与方进行通信，也需遵循本标准的约束。

4.1.1.3 会话层重传是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FIXT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，采取的方式是给对方发送一条消息重传请求。本标准在事实上取消了FIXT会话层协议的会话层重传，只对外仍然表现为标准的FIXT会话层，且可以和对端的FIXT参与方进行互操作。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，虽然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标准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。

4.1.1.4 和FIXT协议一样，LFIXT协议本身并不限定所用的标准端口号。作为基于TCP的会话层协议，LFIXT使用的端口号建议在区间[1024,49151]内，具体取值由通信双方事先另行约定。

4.1.2 应用层重传

由于本标准的会话层恢复机制仅仅是为了与FIXT会话协议兼容，不能作为真正的消息恢复机制使用。因此，应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标准规定的范畴。

4.1.3 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。

4.1.4 会话发起方和接受方

4.1.4.1 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。

4.1.4.2 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。

4.1.4.3 FIXT协议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FIXT会话，而且由于FIXT中并不为每个FIXT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FIXT会话的唯一标识符是SenderCompID和TargetCompID字段的组合。LFIXT标准作为FIXT标准的精简版，也遵循此要求。

4.1.4.4 FIXT 协议中，单个 FIXT 参与方不能同时保有具有相同 SenderCompID+TargetCompID 的两个会话。在 FIXT 中推荐的做法是：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。另一方面，FIXT 协议并未明确不同的 FIXT 参与方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。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，FIXT 协议不作限定。若请求被拒绝，则拒绝方式将遵循 FIXT 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 标准作为 FIXT 标准的精简版，也遵循此要求。

4.1.4.5 本标准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。

4.1.5 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在大多数情况下被初始化为1，并在整个会话过程中连续递增，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。

上述的“连续递增”是针对通常情况而言。在下述情况下，接收方收到的消息的MsgSeqNum也可能出现倒流，但此时的倒流并非异常，而是被称为“正常倒流”：

a) 收到的消息是 SeqReset-Reset 消息且 PossDupFlag=Y，此时应忽略 MsgSeqNum，即使出现倒流也不是错误；
b) 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；
c) 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。

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(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：

a) 如果入向消息序号<NxtIn，且不属于正常倒流，表明发生了严重的错误，应立即结束会话，并开始进行人工干预；
b) 如果入向消息序号<NxtIn，且属于正常倒流，则不属于错误，应进行正常处理；
c) 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。

4.1.6 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。

4.1.7 有序消息处理

本标准采用TCP连接作为底层通信机制。在会话建立后，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，说明检测到了对方或通信连接发生了严重异常，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。

4.1.8 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：

a) 在发现入向消息序号缺口时，LFIXT 参与者不应发送重传请求，应回送 Logout 后直接断开连接；
b) 允许在入向消息中出现重传请求，比如基于 FIXT 的通信对手方虽然收到前面的消息但其自己没保存，并期望能按 FIXT 会话层协议的方式通过重传请求取回，对此 LFIXT 参与者将简单回送 SeqReset-Reset 消息予以回应；
c) 允许在入向消息中出现 PossDupFlag=Y 的消息，比如基于 FIXT 的通信对手方虽未收到本方发出的重传请求，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。

根据FIXT标准，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FIXT标准解释为“替代”而非“重发”。

4.1.9 可能的消息重复发送

在本标准中，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。由于互操作的对方应遵从同样的应用层协议，因此本标准将不会给出向消息打上任何PossibleResend标志。

本标准允许在入向消息头中出现PossibleResend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。

4.1.10 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。

4.1.11 混乱的消息

当至少出现以下情形之一时，一条消息被称为“混乱的”：

a) BeginString(tag8)不是消息的第一个标签，或不以 8=FIXT.n.m 的形式出现；
b) BodyLength(tag9)不是消息的第二个标签，或未包含正确的字节计数；
c) MsgType(tag35)不是消息的第三个标签；
d) CheckSum(tag10)不是最后的标签，或其取值不正确；
e) 若 MsgSeqNum(tag34)缺失，表明出现了严重的应用错误，应立刻终止 FIX 连接。

4.1.12 消息确认

由于本标准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，并在应用层接受或拒绝。

4.1.13 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，比如使用建构于TCP之上的TLS通信加密机制。

4.2 会话管理

4.2.1 会话和连接

本标准采用TCP连接作为底层通信机制。

若LFIXT参与方作为会话的主动发起方，应在每次新建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，此时会话和TCP连接是一一对应的。

虽然本标准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，因此本标准规定：对于单个会话而言，同时只使用一个全双工的TCP连接。

若LFIXT参与方作为会话的接收方，由于该会话的发起方可能是标准的FIXT，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。

4.2.2 建立会话

4.2.2.1 会话步骤

建立会话包含三个内部步骤：建立连接、身份认证、消息同步。

4.2.2.2 建立连接

会话的发起方与接受方应先建立 TCP 连接。LFIXT 参与方在 TCP 连接建立后，应当总是初始化 NxtIn=1，NxtOut=1。

4.2.2.3 身份认证

会话发起方发送登录消息（Logon），会话接受方认证发起方身份的合法性。处理逻辑如下：

a) 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。
b) 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。
c) 会话发起方应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。

d) 在发起方被认证后，接受方将立即回应一个确认Logon消息。发起方将把从接受方返回的Logon消息作为“一个会话已经建立”的确认。

4.2.2.4 消息同步

本标准并不提供真正的会话层重传机制，因此LFIXT参与方作为会话的发起方时，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参与方作为会话接受方时，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FIXT会话协议的消息同步的兼容，具体机制见4.3.2登录消息处理。

4.2.3 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”，本标准仅对管理消息进行描述。

4.2.4 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。注销不影响任何业务层消息的状态。所有有效的业务层消息都可在注销（Logout）之后执行。

4.3 恢复

4.3.1 会话恢复机制概述

本标准的会话层恢复机制是为了与FIXT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参与方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。

4.3.2 登录消息处理

LFIXT参与方作为会话接受方时，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。会话接受方不需要检查任何缺口，会话接受方也不会向发起方请求重传任何消息。如果会话发起方没有提供NextExpectedMsgSeqNum字段，则会话接受方的NxtOut设置为1。

4.3.3 重传请求消息处理

LFIXT参与方不会主动发送重传请求，但可能收到FIXT参与方发送的重传请求。当LFIXT参与方收到重传请求时，会使用SeqReset-Reset消息重置发送方序号，而不会提供历史消息的重传。

4.3.4 序号重设消息处理

LFIXT参与方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。

5 消息规范

5.1 消息结构

5.1.1 消息的组成部分

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。

5.1.2 消息头

会话双方所有交换的消息都具有标准的消息头，消息头所包含的各字段见表1所示。

每一个消息都由一个标准消息头开始。消息头规定了消息的类型、长度、目的地、顺序号、起始点和时间等数据域，均不进行会话层的加密传输。

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；当应用层重新发送消息时，应使用新的消息序号，并通知会话层将PossResend设置为Y。接收者应按以下方法处理上述消息：

a) 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。

b) PossResend=Y：不附带本标志地将消息传递给应用层，由应用层根据业务逻辑自行确定此前是否收到该消息。

表 1 消息头字段

Tag	域名	必选	字段描述
8	BeginString	Y	起始串 FIXT.1.1(总是不在会话层进行加密，应是消息的第一个域)。
9	BodyLength	Y	消息体长度(总是不在会话层进行加密，应是消息的第二个域)。
35	MsgType	Y	消息类型(总是不在会话层进行加密，应是消息的第三个域)。
49	SenderCompID	Y	发送方公司号。发送方不一定真是一个公司，因此应理解为发送方代码字符串(总是不在会话层进行加密)。由通信双方事先协商确定。
56	TargetCompID	Y	接收方公司号。接收方不一定真是一个公司，因此应理解为接收方代码字符串(总是不在会话层进行加密)。由通信双方事先协商确定。
34	MsgSeqNum	Y	消息序号，整数类型。
43	PossDupFlag	N	会话层可能重传标志；会话层重复传送时，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应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。

5.1.3 消息尾

会话双方所有交换的消息具有见表2所示标准的消息尾。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。其字段说明见表2。

表 2　消息尾字段

Tag	域名	必选	字段描述
10	CheckSum	Y	校验和，总是消息的最末域，总是不在会话层进行加密。

CheckSum计算方法对应的C语言代码参见附录A。

5.2　管理消息

5.2.1　管理消息和两种模式的关系

本标准支持FIXT的所有管理消息，但不会主动发送所有管理消息。

本标准分两种模式：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参与者时，可以采用精简模式，此时只需要支持见表3所示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。

表 3　精简模式：仅和 LFIXT 参与者通信时

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

当需要和基于FIXT的对手方进行通信时，应采用兼容模式。此时本标准实现者需要支持所有管理消息的接收，见表4所示，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。

表 4　兼容模式：和 FIXT 参与者通信时

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

例如，交易所端的LFIXT参与者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，也可以考虑采用兼容模式甚至是基于FIXT完成交易所接入。

精简模式的LFIXT协议实现简便，但不具备和FIXT协议互通的能力，使用者应全面衡量各系统的总体成本以确定实际使用的协议。不同协议实现者之间的兼容情况见表5。

表 5　协议兼容性矩阵

	FIXT	兼容模式 LFIXT	精简模式 LFIXT
FIXT	√	√	×
兼容模式 LFIXT		√	√
精简模式 LFIXT			√

5.2.2　Heartbeat 心跳消息（MsgType=0）

会话双方都可借助Heartbeat消息来检测当前使用的TCP连接的状态，因此当任何一方处于数据发送空闲期时，都需要定时发送Heartbeat消息以供检测链接的健康度，故会话双方都应支持心跳信号的主动发送，心跳消息字段见表6所示。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。

本标准不会主动发送TestRequest消息，但LFIXT参与方，为了与FIXT会话协议的兼容，如果收到了TestRequest请求，应按FIXT会话协议向发送方返回心跳响应。

如果上层应用约定有应用层心跳信息，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，永远也无法收到来自对方的会话层心跳消息，也不违反本标准。

附录B给出了帮助理解心跳和测试请求消息的例子。

表 6　心跳消息字段

Tag	域名	必选	字段描述
	StandardHeader	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。
	StandardTrailer	Y	

5.2.3　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，其消息格式规范见表7所示。

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）。LFIXT协议遵照FIXT协议的做法，本身不规定本字段的取值。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：应在收到返回的登录消息之后才实施正常的消息交换。

LFIXT参与方若作为会话发起方，其发送的Logon消息应将MsgSeqNum设置为1、ResetSeqNumFlag设置为Y、NextExpectedMsgSeqNum设置为1。

FIXT参与方若要向LFIXT参与方发起会话，且选择采用NextExpectedMsgSeqNum(789)来实现序号同步，应在Logon消息中将该字段填写为FIXT参与方已保存的入向消息的最大序号+1。比如FIXT参与方保存了入向消息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，故本标准对此予以特别限定。**

表 7　Logon 登录消息字段

Tag	域名	必选	字段描述
	StandardHeader	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。双方应用同一个值。
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，FIX 消息的缺省自定义应用版本。本标签是对 tag1137+tag1407 的进一步约束。
1408	DefaultCstmApplVerID	N	本字段填写数据协议版本，建议的格式是：STEP 版本号_市场代码_市场内部唯一的协议版本号。
	StandardTrailer	Y	

附录C给出了若干帮助理解正常登录、异常登录场景的例子。

5.2.4　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，其消息格式规范见表8所示。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参与方不会主动发送任何TestRequest消息。

表 8　测试请求消息

Tag	域名	必选	注释
	StandardHeader	Y	MsgType=1
112	TestReqID	N	测试请求标识符
	StandardTrailer	Y	

5.2.5　Resend 重发请求消息（MsgType=2）

重发请求消息由接收方发出，消息格式规范见表9所示，目的是向发送方申请某些消息重复发送。本标准不会主动发送该消息。

重发请求消息有以下三种表示方式：

a)　请求重发一条消息：BeginSeqNo=EndSeqNo；

b)　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；

c)　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参与方作为本消息的接收方时，只会通过SeqReset-Reset消息响应FIXT参与方发送的重传请求消息。

表 9　Resend 重发请求消息

Tag	域名	必选	注释
	StandardHeader	Y	MsgType=2
7	BeginSeqNo	Y	起始消息序号
16	EndSeqNo	Y	结束消息序号
	StandardTrailer	Y	

5.2.6　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，应发出Reject（拒绝）消息，拒绝消息的消息格式见表10所示，会话拒绝原因见表11所示。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=&）。

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，它应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FIXT会话协议不同之处在于：在本标准中，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FIXT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。

建议在本消息的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。

附录D给出了会话拒绝场景的例子。

表 10　Reject 消息

Tag	域名	必选	说明
	StandardHeader	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。
	StandardTrailer	Y	

表 11　会话拒绝原因

会话拒绝原因(SessionRejectReason)
0=无效域号
1=该消息中必选的域丢失
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。

5.2.7　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，序号重设消息的消息格式规范见表12。序号重设消息有两种模式：序号重设-缺口填补（SeqReset-GapFill）；序号重设-重设（SeqReset-Reset）。

在本标准中，只使用SeqReset-Reset消息回应ResendRequest消息。ResendRequest消息所请求的消息范围不应和接收方当前NxtOut矛盾，即当EndSeqNo>0时，应保证BeginSeqNo<=EndSeqNo<NxtOut；当EndSeqNo=0时，应保证BeginSeqNo<NxtOut。此SeqReset-Reset消息的MsgSeqNum按FIXT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则建议填写为本方当前NxtOut值，且作为NxtOut序号维护的特例，在发送完本SeqReset-Reset消息后发送者的NxtOut保持不变。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。

在本标准中，虽然不会主动发起重传请求，但仍然有可能收到FIXT参与方对手主动发出的SeqReset类消息。

当收到的序号重设消息为SeqReset-GapFill消息时，应确保NewSeqNo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。

附录E给出了若干帮助理解重传请求处理场景的例子。

表 12　序号重设消息

Tag	域名	必选	注释
	StandardHeader	Y	MsgType=4
123	GapFillFlag	N	缺口填补标志。布尔型。“Y”表明是 GapFill 模式；“N”或不存在本域，表明是 Reset 模式。
36	NewSeqNo	Y	新消息序号
	StandardTrailer	Y	

5.2.8　Logout 注销消息（MsgType=5）

注销消息是发起或确认会话终止的消息，消息格式规范见表13。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，注销的发起人应该在主动发送注销消息后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。本标准遵照FIXT协议的做法，本身不规定退登超时间隔的取值，而是应由连接双方另行约定。

存在一种边界情况，也就是会话双方在差不多相同的时刻都在没有收到对方Logout消息的前提下独立向对方发出了Logout注销消息。因此，其中一方A可能刚主动发出Logout消息就收到了对方B的Logout消息，虽然B的这条消息并不是针对A主动发起的那条Logout消息的应答，但对于A而言他是无法也无须区分这两种情况的，A仍然可以认为退登握手已经正常完成。

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。

一般而言，作为注销发起方的通信对手方，在收到注销发起方发来的Logout消息后，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：

a) 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，因此不应回送 Logout 消息以泄露 FIXT 版本、合法 SenderCompID、TargetCompID 等信息；

b) 在 FIXT 协议中，在收到登录请求时若在同一个 FIXT 若已存在一个合法的由相同 SenderCompID+TargetCompID 唯一标识的会话，则 FIXT 将直接断开后来的登录请求所欲发起的会话。

由于任何时刻都存在网络中断的可能，因此本标准的任何参与方都应对于未收到对方的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，但非必选。

对于收到报文中的各类异常，接收方通常处理的原则：

a) 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应能够处理 TCP 连接的中断，因此这也是缺省处理办法。
   在同一个 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，也视作恶意攻击，直接断开。

b) 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。

c) 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。

d) 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。

附录F给出了帮助理解正常注销场景的例子。

表 13　注销消息

Tag	域名	必选	字段描述
	StandardHeader	Y	MsgType=5
1409	SessionStatus	N	Logout 时的会话状态。根据 FIXT 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。
	StandardTrailer	Y	

6　数据字典

6.1　数据类型

本标准中使用的数据类型的说明及规范见表 14。

表 14　数据类型

数据类型	类型规范	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，FIXT 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

表 14　数据类型（续）

数据类型	类型规范	说明
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，类型规范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。类型规范Cx中，x表示字符串的最大长度。		

6.2　会话层域规范

会话层域规范见表 15。

表 15　会话层域规范

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 不在会话层加密，应是最后一个域。 校验算法见附录 F
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	发送方公司号。发送方不一定真是一个公司，因此应理解为发送方代码字符串(总是不加密)。由通信双方事先协商确定。
52	SendingTime	UTCTimestamp	消息发送时间，UTCTimestamp 类型。
56	TargetCompID	C32	接收方公司号。接收方不一定真是一个公司，因此应理解为接收方代码字符串(总是不加密)。由通信双方事先协商确定。
58	Text	C1024	文本

表 15　会话层域规范（续）

Tag	域名	类型	说明
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。
112	TestReqID	C32	测试请求标识符
141	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
371	RefTagID	N9	相关错误消息中，出现错误的 FIX 域号。
372	RefMsgType	C16	相关错误消息的 MsgType
373	SessionRejectReason	N9	会话拒绝原因编号，无符号整数。
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。若取值为'9'，意思是'FIX50SP2'。
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。若取值'124'，意思是'EP124'。
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 tag1137+tag1407 的进一步约束。 本字段填写数据协议版本，建议的格式是：STEP 版本号_市场代码_市场内部唯一的协议版本号。 本字段只需要对于会话双方而言本字段取值可供唯一地标识和区分数据协议版本，不需要全世界统一。为确保版本唯一性，建议市场代码填写 ISO 10383:2012 中定义的市场标识代码(MIC)。如果使用自定义市场代码，会话双方使用此类代码时需自行确保市场代码无歧义。
1409	SessionStatus	N4	本字段填写退登时的会话状态

附 录 A
（规范性附录）
计算校验和

以下为计算校验和的C语言代码段：

```
char*GenerateCheckSum(char*buf,longbufLen)
{
chartmpBuf[4];
    longidx;
    unsignedintcks;

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

附 录 B
（规范性附录）
处理心跳和测试请求

图 B.1 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。FIXT 协议的实现者可以不受限制地主动发起测试请求。图 B.1 中的 Server 是 LFIXT 参与方，不会主动发起测试请求。

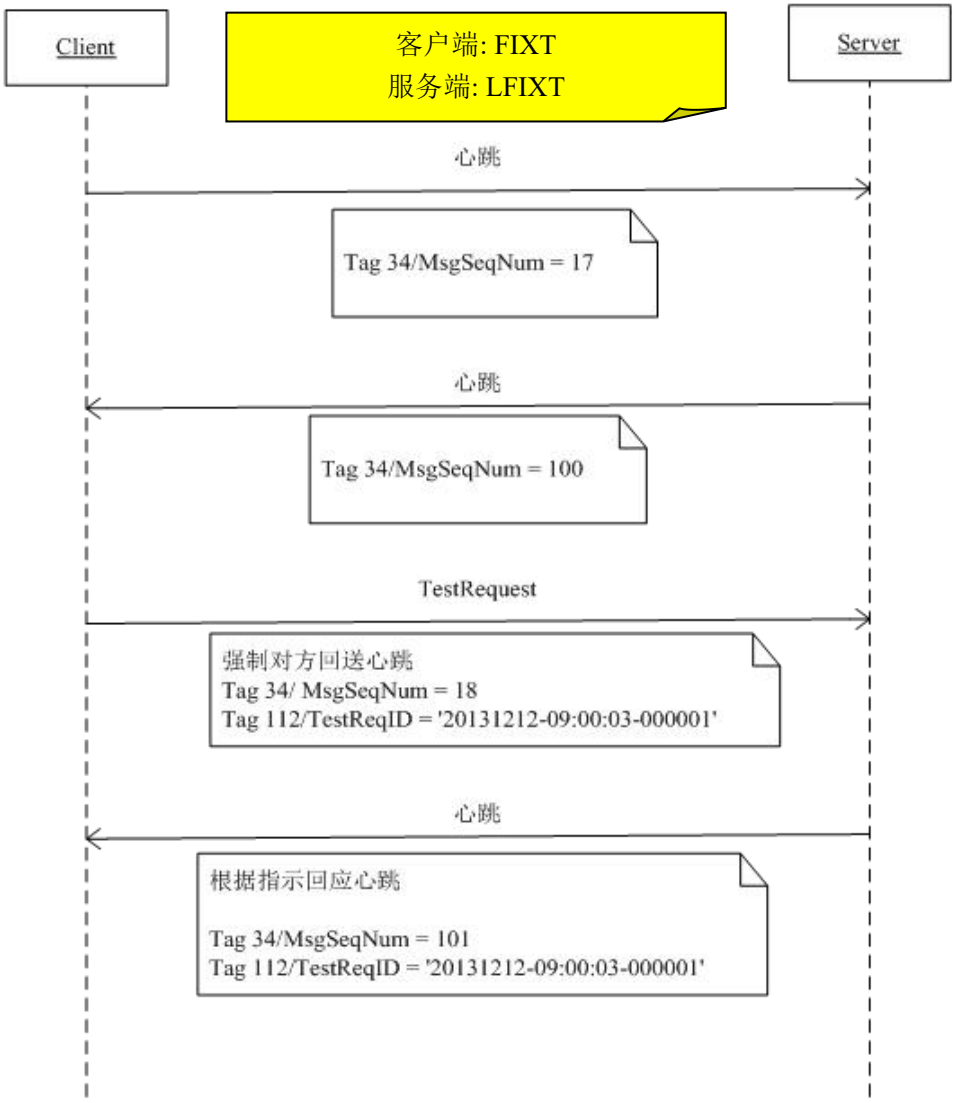


图 B.1　　处理心跳和测试请求

附 录 C
（规范性附录）
登录场景

C.1 正常登录场景一

LFIXT参与方作为登录发起方——Client,LFIXT参与方作为登录接受方——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。场景流程图如图C.1所示。

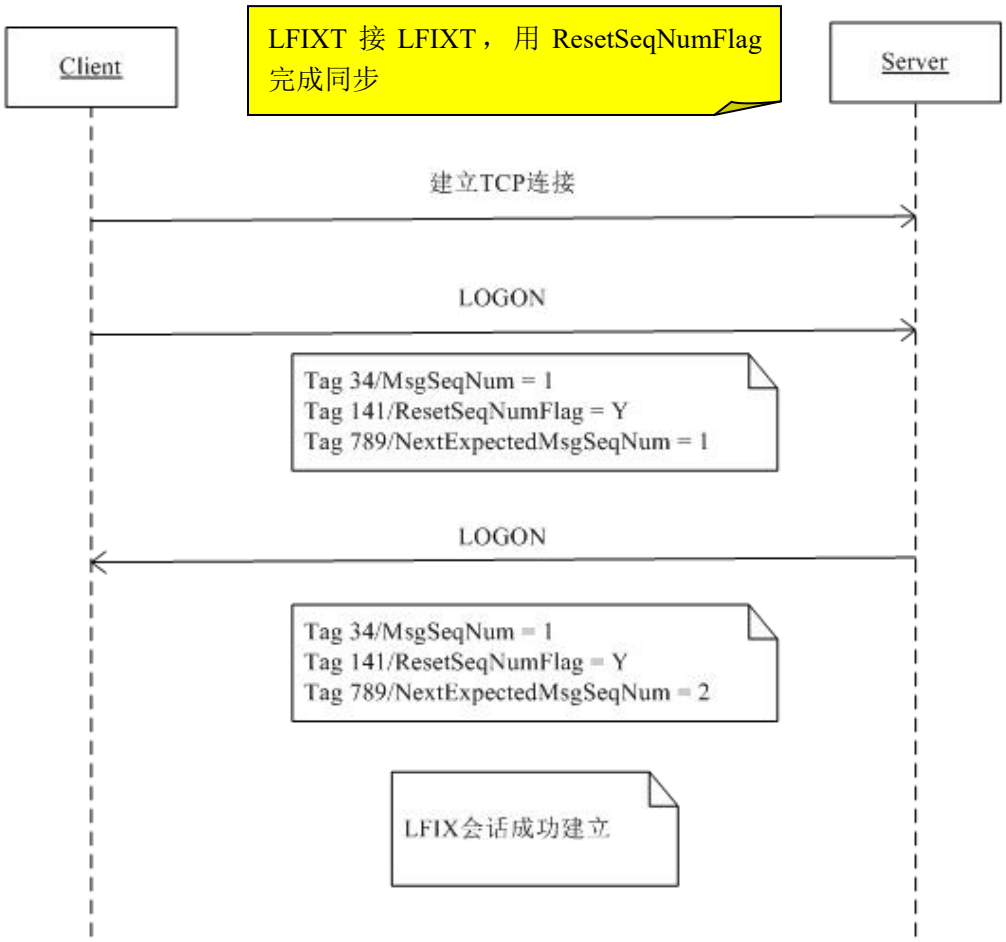


图 C.1 正常登录场景一

以图C.1中显示的场景为例，在会话建立后，将Client的NxtOut=2，NxtIn=2，Server的NxtOut=2，NxtIn=2。

C.2 正常登录场景二

FIXT参与方作为登录发起方——Client，LFIXT参与方作为登录接受方——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。

场景流程图如图C.2所示。

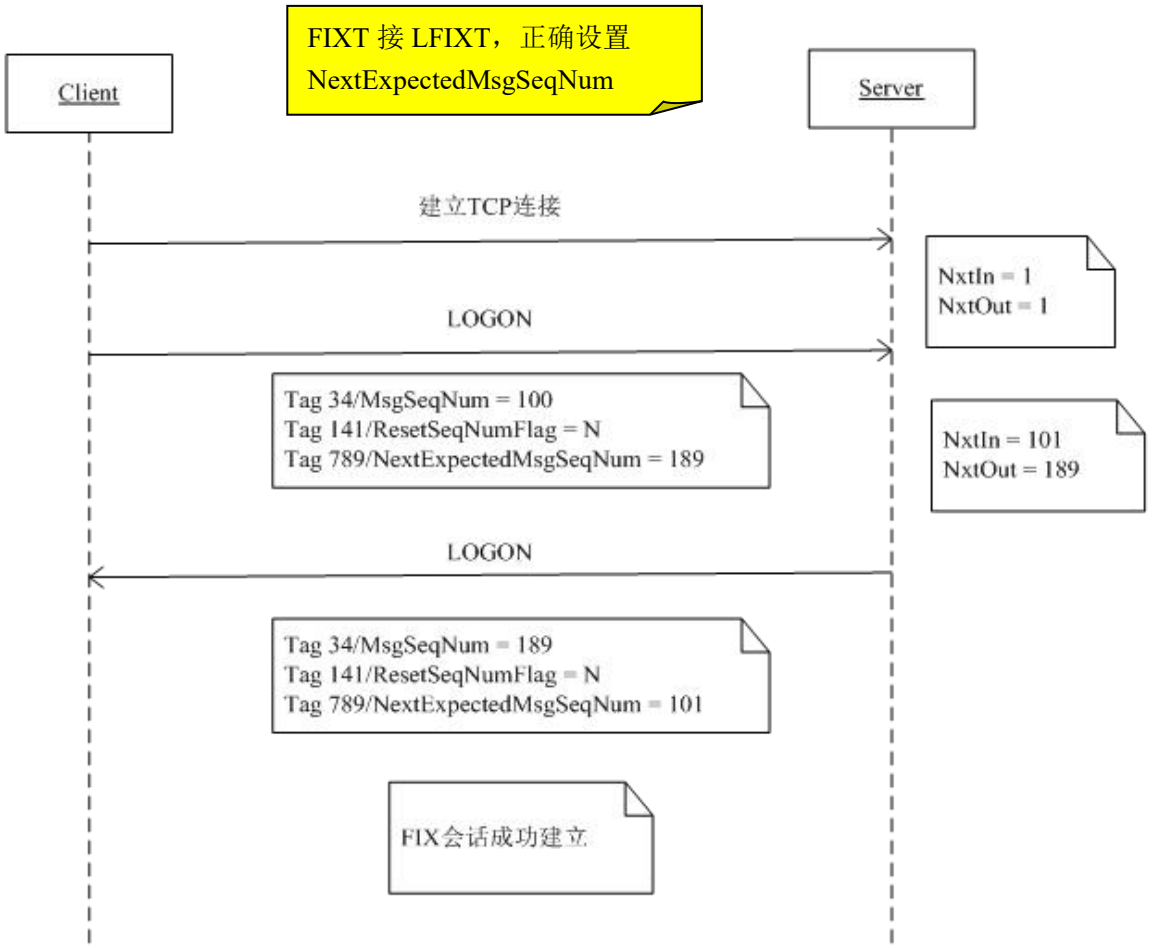


图 C.2 正常登录场景二

以图C.2中的场景为例，在会话建立后，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**。

C.3 正常登录场景三

为了说明在FIXT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FIXT协议。场景流程图如图C.3所示。

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，应使用新的序号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。

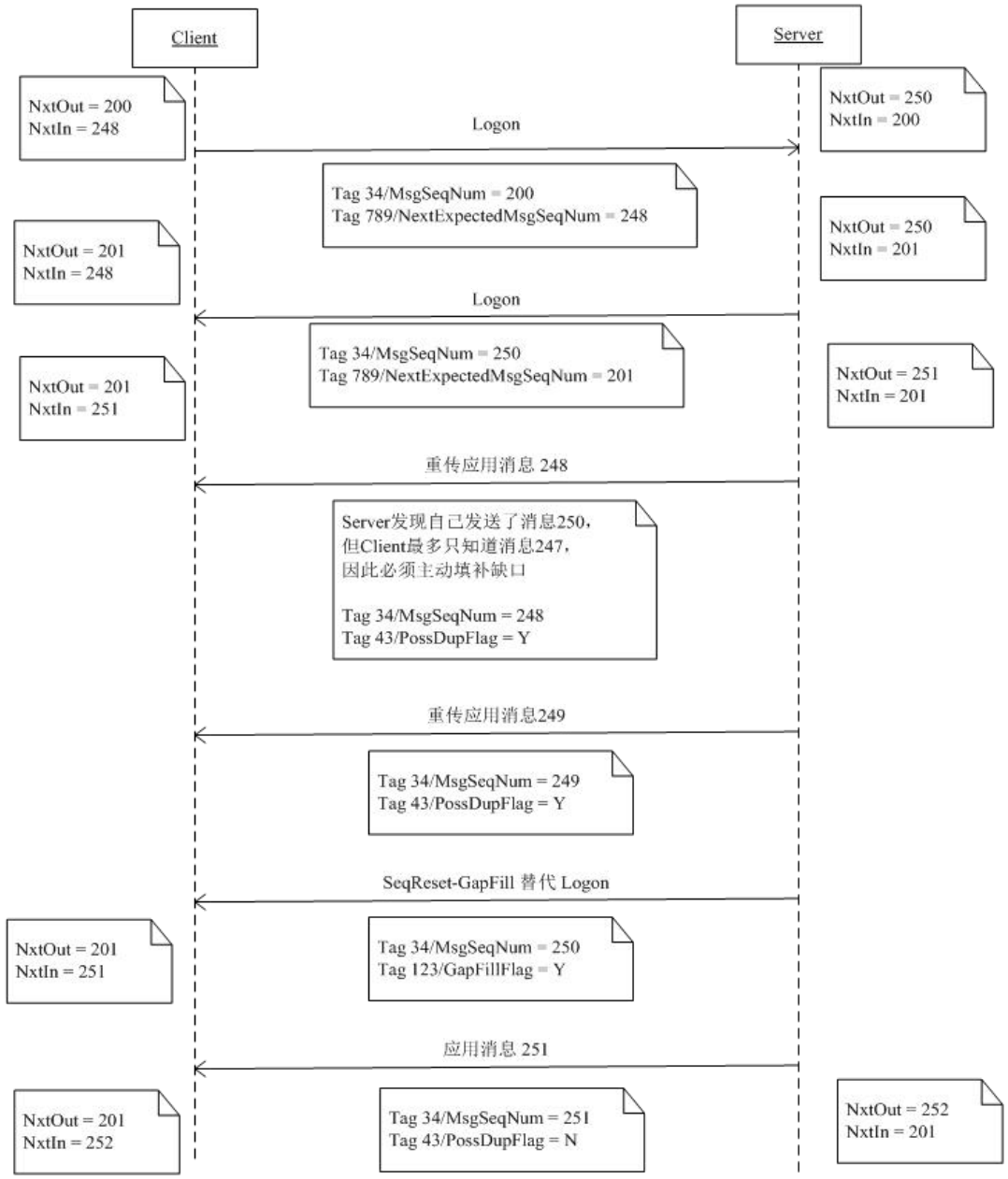


图 C.3　正常登录场景三

C.4 异常登录场景一

FIXT参与方作为登录发起方——Client，LFIXT参与方作为登录接受方——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。场景流程图如图C.4所示。

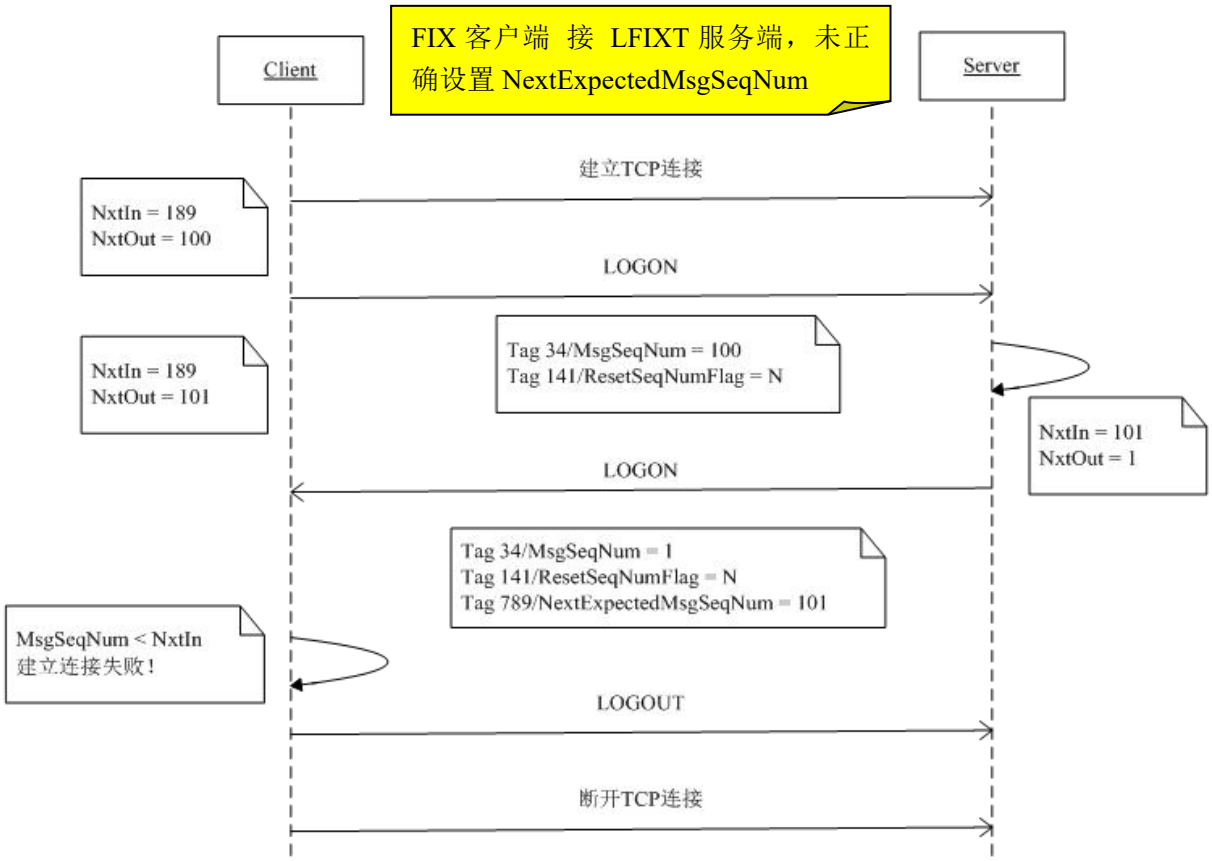


图 C.4 异常登录场景一

针对上图作一说明：由于FIXT作为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FIXT作为客户端认为出现了严重错误，在发送LOGOUT后断开连接。

C.5 异常登录场景二

LFIXT参与方作为登录发起方——Client，LFIXT参与方作为登录接受方——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。场景流程图如图C.5所示。

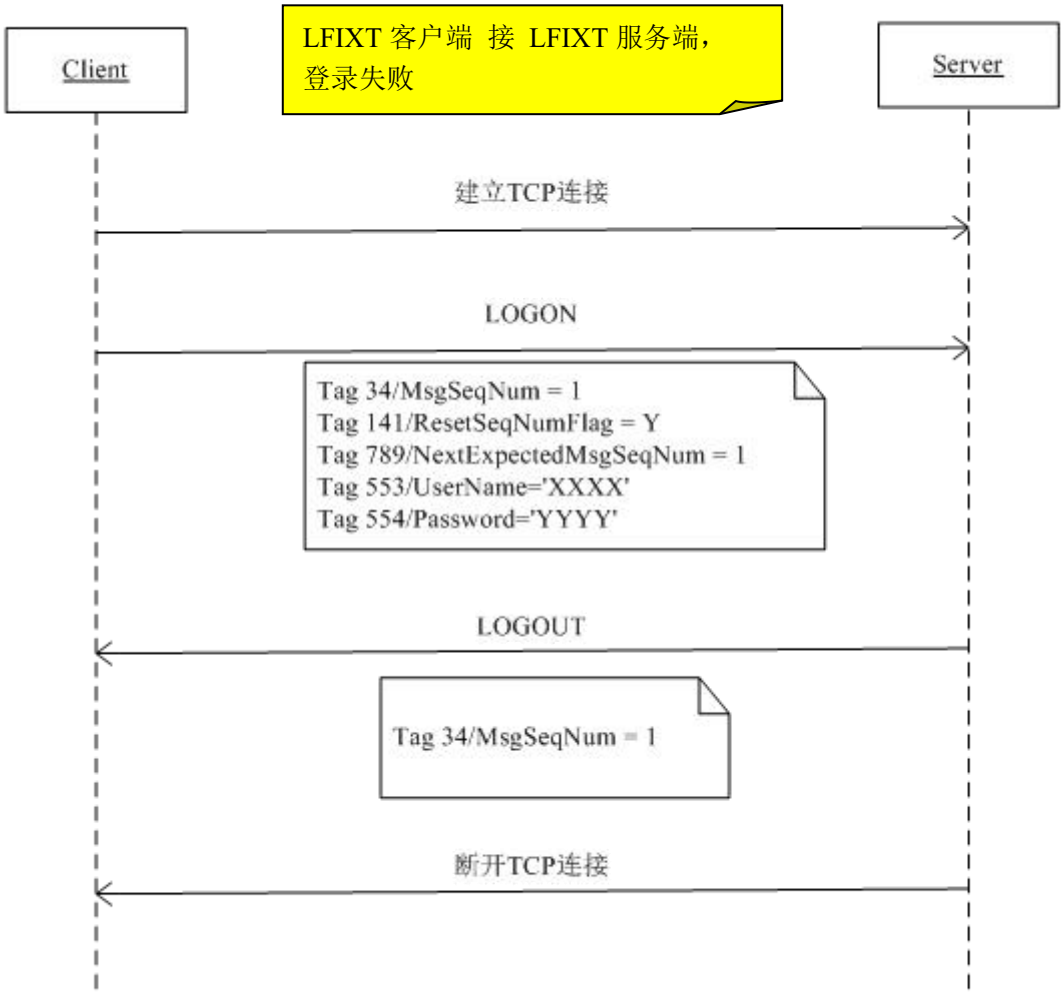

LFIXT 客户端 接 LFIXT 服务端，
登录失败
Client
Server
建立TCP连接
LOGON
Tag 34/MsgSeqNum = 1
Tag 141/ResetSeqNumFlag = Y
Tag 789/NextExpectedMsgSeqNum = 1
Tag 553/UserName='XXXX'
Tag 554/Password='YYYY'
LOGOUT
Tag 34/MsgSeqNum = 1
断开TCP连接

图 C.5　异常登录场景二

附 录 D
（规范性附录）
处理会话拒绝

LFIXT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（图D.1）中，Server是LFIXT参与方，Client是标准的FIXT参与方。

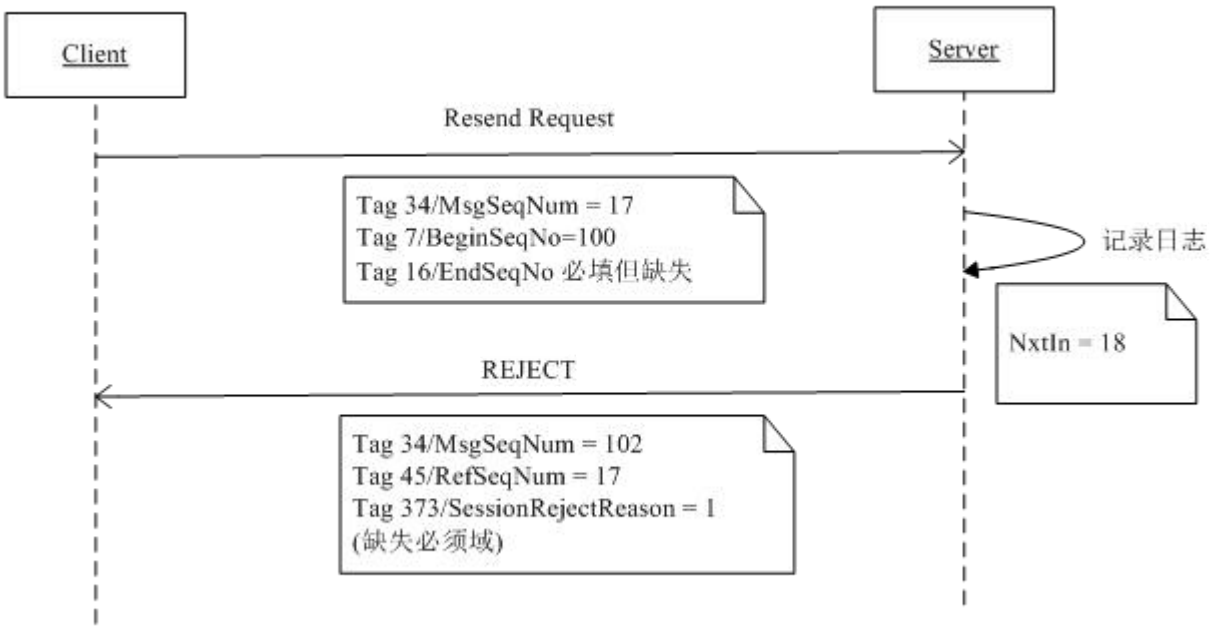


图 D.1 处理会话拒绝

附 录 E
（规范性附录）
处理重传请求场景

E.1 处理重传请求场景一

图E.1中，Client是FIXT参与方，Server是LFIXT参与方，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。

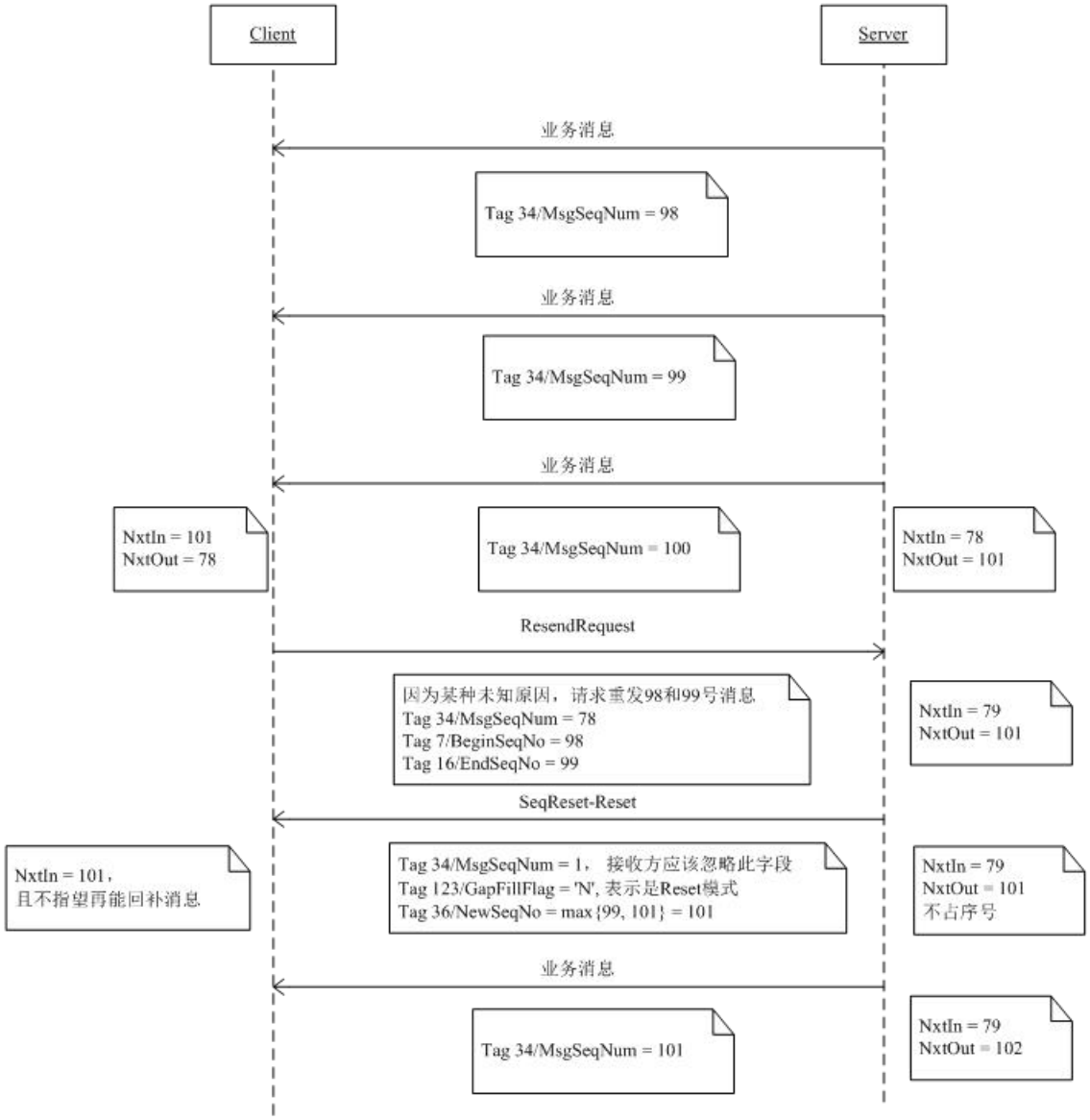
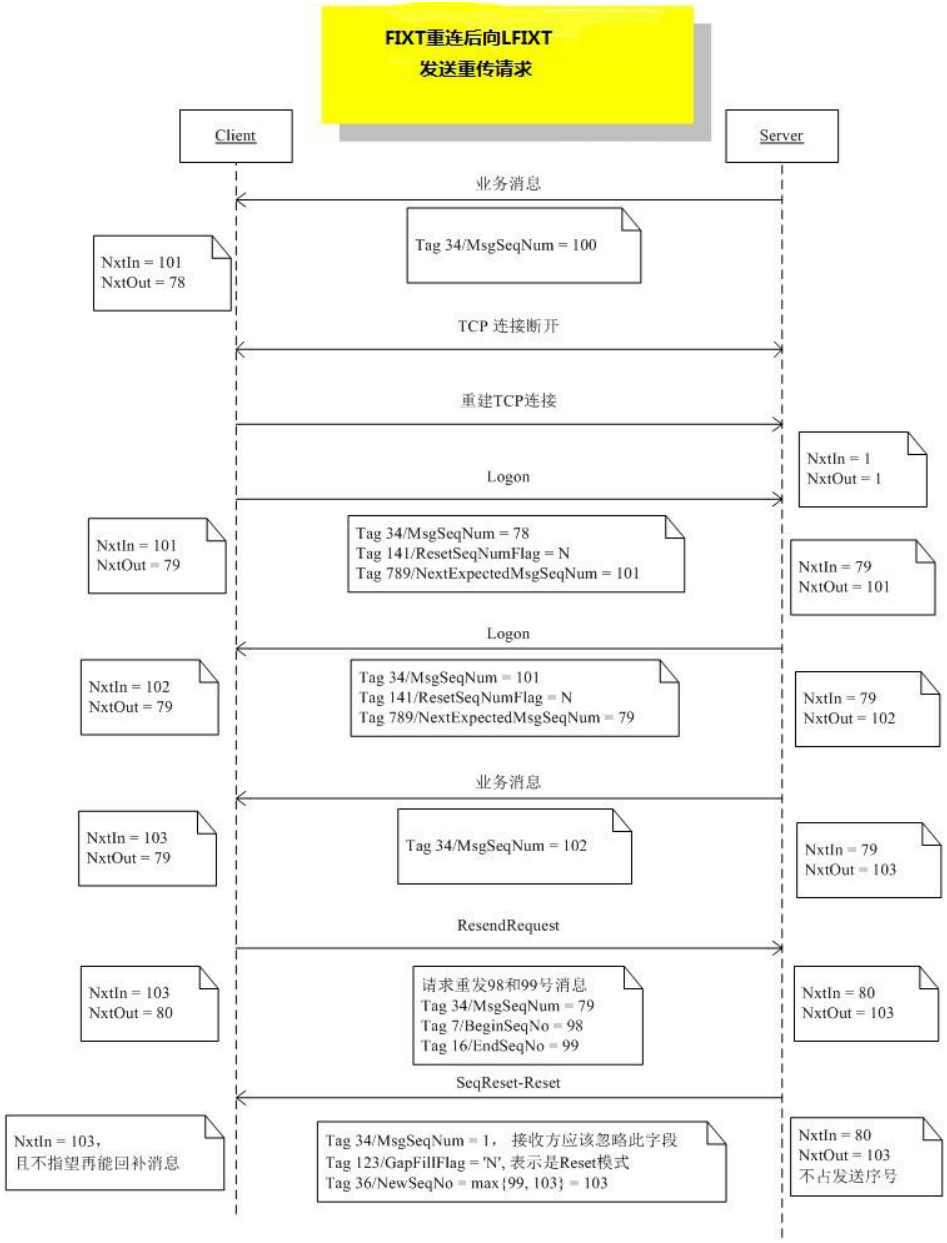


图 E.1 处理重传请求

E.2 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。

场景流程图如图E.2所示。



图E.2 处理重传请求

附 录 F
（规范性附录）
注销场景

图F.1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。

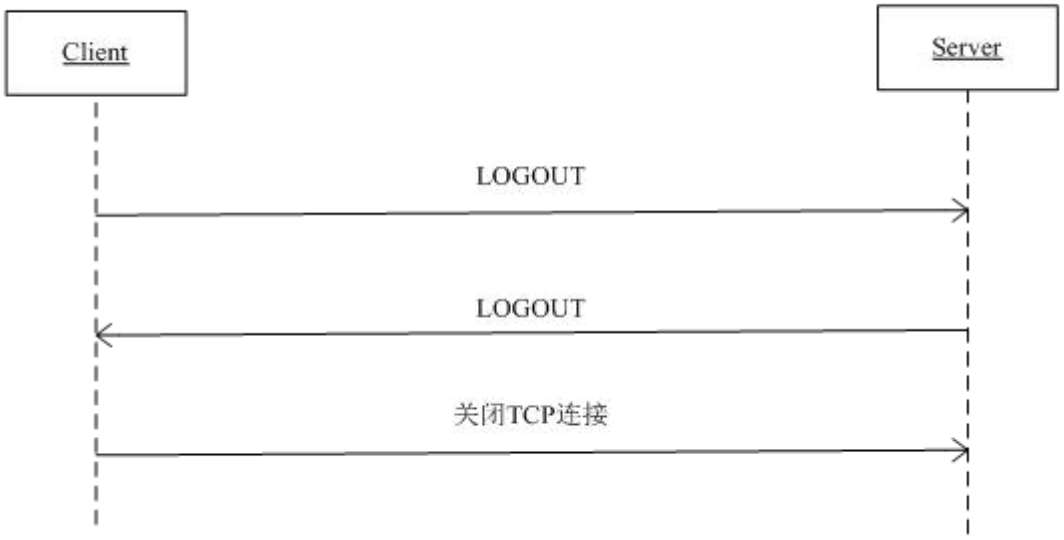


图 F.1　　正常注销场景