**AMQP v1.0（修订版 0）**

2011 年 10 月 07 日

决赛

URL： <http://svn.amqp.org/svn/amqp/trunk>

修订： 1350

**版权声明**

(c) 版权所有 Bank of America, N.A.、Barclays Bank PLC、Cisco Systems、Credit Suisse、Deutsche Boerse、Envoy Technologies Inc.、Goldman Sachs、HCL Technologies Ltd、IIT Software GmbH、iMatix Corporation、INETCO Systems Limited、Informatica Corporation、JPMorgan Chase & Co.、Kaazing Corporation, N.A.、Microsoft Corporation、

my-Channels、Novell、Progress Software、Red Hat Inc.、Software AG、Solace Systems Inc.、StormMQ Ltd.、Tervela Inc.、TWIST Process Innovations Ltd、VMware、Inc. 和 WS02 Inc.

2006-2011.保留所有权利。

**许可证**

美国银行、巴克莱银行、思科系统公司、瑞士信贷银行、德意志交易所、高盛集团、HCL 技术有限公司、IIT 软件有限公司、INETCO 系统有限公司、Informatica 公司、摩根大通公司、

Kaazing Corporation, N.A, Microsoft Corporation, my-Channels, Novell, Progress Software, Red Hat Inc、和WS02 Inc.（以下统称为 "作者"）特此分别向您授予全球范围内的、永久的、免版税的、不可转让的、非独占的许可，允许您(i)复制、显示、分发和实施高级消息队列协议（"AMQP"）规范和(ii)作者持有的许可权利要求，所有这些都是为了实施高级消息队列协议规范。如果您对任何作者提起与高级消息队列协议规范有关的任何索赔、诉讼、要求或行动，您在本协议项下的许可和任何权利将立即终止，而无需任何作者的通知。一旦终止，你应销毁你所拥有或控制的《高级消息队列协议规范》的所有副本。

在本协议中使用的 "许可权利要求 "是指作者或其关联公司现在或将来任何时候在全世界范围内拥有或控制的专利或专利申请中的权利要求，不包括外观设计专利和外观设计注册，或者可以按照本协议的要求无偿再许可的权利要求，这些权利要求必然会因高级报文队列协议规范的实施而被侵权。只有在不可能避免侵权的情况下，某项权利要求才必然被侵权，因为没有任何合理的非侵权替代方案来实施《高级邮件队列协议规范》的必要部分。尽管有上述规定，被许可权利要求不得包括除上述规定以外的任何权利要求，即使这些权利要求与被许可权利要求包含在同一专利中；也不得包括仅基于《高级邮件队列协议规范》任何部分的任何实施的权利要求，这些实施并非《高级邮件队列协议规范》所要求的，或者如果被许可，许可人需要向非关联第三方支付使用费的权利要求。此外，被许可权利要求不应包括：(i) 任何制造或使用被许可产品所必需的、但未在《高级报文队列协议规范》中明确规定的使能技术（例如：半导体制造技术、编译器技术）、半导体制造技术、编译器技术、面向对象技术、网络技术、操作系统技术等）；或(ii)其他地方制定的、仅在高级信息队列协议规范中提及的其他已发布标准的实施；或(iii)其目的或功能并非为符合高级信息队列协议规范所必需的任何许可产品及其任何组合。就本定义而言，高级消息队列协议规范应被视为包括对互操作性至关重要的架构和互连要求，并且还可能包括支持源代码的人工制品，如果这些架构、互连要求和源代码人工制品被明确认定为实现符合高级消息队列协议规范的要求或文件。

在本协议中，"许可产品 "仅指实施并符合《高级报文队列协议规范》所有相关部分的产品（硬件、软件或其组合）的特定部分。

以下免责声明也适用于对高级报文队列协议规范的任何使用：

高级报文队列协议规范 "按原样 "提供，作者不作任何明示或暗示的陈述或保证，包括但不 限于对适销性、特定用途的适用性、非侵权性或所有权的保证；不保证高级报文队列协议规 范的内容适用于任何用途；也不保证高级报文队列协议规范的实施不会侵犯任何第三方的专 利、版权、商标或其他权利。

对于因使用、执行或分发高级报文队列协议规范而引起的或与之有关的任何直接、间接、特 殊、附带或间接损害，作者概不负责。

未经事先书面许可，不得以任何方式使用作者的姓名和商标，包括与《高级报文队列协议规范》或其内容有关的广告或宣传。高级消息队列协议规范》的版权归作者所有。

不得以暗示、禁止反言或其他方式授予其他权利。

在本协议项下的许可或权利终止后，您应销毁您所拥有或控制的《高级报文队列协议规范》的所有副本。

**商标**

"摩根大通"、"摩根大通"、"大通"、摩根大通标志和八角形符号是摩根大通公司的商标。

RED HAT 是 Red Hat 公司在美国和其他国家的注册商标。

其他公司、产品或服务名称可能是他人的商标或服务标志。

**链接至完整的 AMQP 规范**

[http://www.amqp.org/confluence/display/AMQP/AMQP+ 规格](http://www.amqp.org/confluence/display/AMQP/AMQP%2BSpecification)

**目录**

[导言](#_bookmark0) 6

1. [类型](#_bookmark1) 9
   1. [类型系统](#_bookmark2) 9
      1. [原始类型](#_bookmark3) 9
      2. [描述类型](#_bookmark4) 10
      3. [描述符值](#_bookmark5) 10
   2. [类型编码](#_bookmark6) 10
      1. [固定宽度](#_bookmark7) 13
      2. [可变宽度](#_bookmark26) 17
      3. [复合物](#_bookmark30) 18
      4. [阵列](#_bookmark33) 19
      5. [编码清单](#_bookmark35) 20
   3. [复合材料类型](#_bookmark36) 21
      1. [列表编码](#_bookmark37) 22
2. [运输](#_bookmark38) 23
   1. [运输](#_bookmark39) 23
   2. [版本谈判](#_bookmark40) 25
   3. [框架](#_bookmark41) 27
      1. [框架布局](#_bookmark42) 27
      2. [AMQP 帧](#_bookmark43) 28
   4. [连接](#_bookmark44) 29
      1. [开启连接](#_bookmark45) 30
      2. [流水线开放式](#_bookmark46) 30
      3. [关闭连接](#_bookmark47) 31
      4. [同时关闭](#_bookmark48) 31
      5. [断开连接的空闲时间](#_bookmark49) 31
      6. [连接状态](#_bookmark50) 32
      7. [连接状态图](#_bookmark51) 33
   5. [会期](#_bookmark52) 34
      1. [建立会议](#_bookmark53) 34
      2. [结束会议](#_bookmark54) 35
      3. [同时结束](#_bookmark55) 35
      4. [会议错误](#_bookmark56) 36
      5. [届会国家](#_bookmark57) 36
      6. [会话流量控制](#_bookmark58) 37
   6. [链接](#_bookmark59) 38
      1. [命名链接](#_bookmark60) 39
      2. [链接手柄](#_bookmark61) 39
      3. [建立或恢复链接](#_bookmark62) 40
      4. [拆卸和重新连接链接](#_bookmark63) 42
      5. [链接错误](#_bookmark64) 43
      6. [关闭链接](#_bookmark65) 44
      7. [流量控制](#_bookmark66) 44
      8. [同步获取](#_bookmark67) 46
      9. [异步通知](#_bookmark68) 47
      10. [停止链接](#_bookmark69) 47
      11. [信息](#_bookmark70) 48
      12. [传输信息](#_bookmark71) 48
      13. [恢复交付](#_bookmark72) 52
      14. [传输大容量信息](#_bookmark73) 52
   7. [表演节目](#_bookmark74) 53
      1. [开放](#_bookmark75) 53
      2. [开始](#_bookmark76) 55
      3. [附件](#_bookmark77) 56
      4. [流量](#_bookmark78) 58
      5. [转让](#_bookmark79) 60
      6. [处置](#_bookmark80) 62
      7. [分离](#_bookmark81) 63
      8. [结束语](#_bookmark82) 64
      9. [关闭](#_bookmark83) 64
   8. [定义](#_bookmark84) 65
      1. [作用](#_bookmark85) 65
      2. [发送器设置模式](#_bookmark86) 65
      3. [接收器设置模式](#_bookmark88) 65
      4. [手柄](#_bookmark90) 66
      5. [秒数](#_bookmark91) 66
      6. [毫秒](#_bookmark92) 66
      7. [送货标签](#_bookmark93) 66
      8. [交货编号](#_bookmark94) 66
      9. [转账编号](#_bookmark95) 66
      10. [序列号](#_bookmark96) 67
      11. [信息格式](#_bookmark97) 67
      12. [IETF 语言标记](#_bookmark98) 67
      13. [领域](#_bookmark99) 67
      14. [错误](#_bookmark100) 67
      15. [AMQP 错误](#_bookmark101) 68
      16. [连接错误](#_bookmark103) 69
      17. [会话错误](#_bookmark104) 70
      18. [链接错误](#_bookmark106) 70
      19. [常量定义](#_bookmark108) 71
3. [信息传递](#_bookmark110) 73
   1. [导言](#_bookmark111) 73
   2. [信息格式](#_bookmark112) 73
      1. [页眉](#_bookmark113) 74
      2. [交付注释](#_bookmark114) 76
      3. [消息注释](#_bookmark115) 76
      4. [属性](#_bookmark116) 76
      5. [应用属性](#_bookmark117) 78
      6. [数据](#_bookmark118) 79
      7. [AMQP 序列](#_bookmark119) 79
      8. [AMQP 值](#_bookmark120) 79
      9. [页脚](#_bookmark121) 79
      10. [注释](#_bookmark122) 79
      11. [信息 ID ULong](#_bookmark123) 80

|  |  |  |
| --- | --- | --- |
|  | [3.2.12 报文 ID UUID](#_bookmark124) . | 80 |
| [3.2.13 报文 ID 二进制](#_bookmark125) .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 80 |
| [3.2.14 报文 ID 字符串](#_bookmark126) . | 80 |
| [3.2.15 地址字符串](#_bookmark127) ............................................................................... | 80 |
| [3.2.16 消息格式](#_bookmark128) ...................................................................... | 80 |
| [3.3](#_bookmark129) | [分发节点](#_bookmark129) | 80 |
|  | [3.3.1 报文状态](#_bookmark130) .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 80 |
| [3.4](#_bookmark131) | [交货状态](#_bookmark131) | 81 |
|  | [3.4.1 接收到的信息](#_bookmark132) | 81 |
|  | [3.4.2 接受](#_bookmark133)。 | 82 |
|  | [3.4.3 拒绝](#_bookmark134)。 | 82 |
|  | [3.4.4 已发布](#_bookmark135)。 | 83 |
|  | [3.4.5 已修改](#_bookmark136)。 | 83 |
|  | [3.4.6 使用递送状态恢复递送](#_bookmark137) .. . . . . . . . . . . . . . . . . . . . | 84 |
| [3.5](#_bookmark138) | [来源和目标](#_bookmark138) | 88 |
|  | [3.5.1 过滤报文](#_bookmark139) . | 88 |
|  | [3.5.2 分配模式](#_bookmark140) . | 88 |
|  | [3.5.3 Source](#_bookmark141) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 89 |
|  | [3.5.4 目标](#_bookmark142)。 | 91 |
|  | [3.5.5 总站耐久性](#_bookmark143) | 92 |
|  | [3.5.6 总站到期政策](#_bookmark144) .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 92 |
|  | [3.5.7 标准分布模式](#_bookmark145) .. . . . . . . . . . . . . . . . . . . . . . . . . . . . | 93 |
|  | [3.5.8 过滤器套件](#_bookmark146) . | 93 |
|  | [3.5.9 节点属性](#_bookmark147) . | 93 |
|  | [3.5.10 关闭时删除](#_bookmark148) ...................................................................... | 94 |
|  | [3.5.11 无链接时删除](#_bookmark149) . | 94 |
|  | [3.5.12 在无信息时删除](#_bookmark150) . | 94 |
|  | [3.5.13 无链接或信息时删除](#_bookmark151) . | 94 |

1. [**交易**](#_bookmark152) **95**
   1. [事务性信息传递](#_bookmark153) 95
   2. [申报交易](#_bookmark154) 95
   3. [解除交易](#_bookmark155) 96
   4. [事务性工作](#_bookmark156) 97
      1. [事务性发布](#_bookmark157) 98
      2. [事务性退休](#_bookmark158) 98
      3. [交易性收购](#_bookmark159) 99
      4. [结算与交易的互动](#_bookmark160) 100
         1. [事务性发布](#_bookmark161) 100
         2. [事务性退休](#_bookmark162) 101
         3. [交易性收购](#_bookmark163) 101
   5. [协调](#_bookmark164) 102
      1. [协调员](#_bookmark165) 102
      2. [宣布](#_bookmark166) 102
      3. [卸货](#_bookmark167) 102
      4. [交易 ID](#_bookmark168) 103
      5. [已宣布](#_bookmark169) 103
      6. [交易状态](#_bookmark170) 103
      7. [交易能力](#_bookmark171) 104
      8. [交易错误](#_bookmark173) 104
2. [**安全**](#_bookmark175) **106**
   1. [安全层](#_bookmark176) 106
   2. [TLS](#_bookmark177) 106
      1. [另类机构](#_bookmark178) 107
      2. [常量定义](#_bookmark179) 107
   3. [SASL](#_bookmark181) 107
      1. [SASL 框架](#_bookmark182) 108
      2. [SASL 谈判](#_bookmark183) 109
      3. [安全框架机身](#_bookmark184) 109
         1. [SASL 机制](#_bookmark185) 109
         2. [SASL Init](#_bookmark186) 110
         3. [SASL 挑战赛](#_bookmark187) 110
         4. [SASL 答复](#_bookmark188) 111
         5. [SASL 成果](#_bookmark189) 111
         6. [SASL 编码](#_bookmark190) 111
      4. [常量定义](#_bookmark192) 112

# 导言

**概述**

高级信息队列协议是一种开放的互联网商务信息传输协议。

AMQP 分成不同的层。在最底层，我们定义了一种高效的二进制点对点协议，用于在网络上传输两个进程之间的信息。其次，我们定义了一种具有具体标准编码的抽象信息格式。每个符合标准的 AMQP 进程都必须能够以这种标准编码发送和接收信息。

**理由和用例**

一个商业消息传递用户社区根据他们构建和运行网络信息处理系统的经验，定义了 AMQP 的要求。

AMQP 工作组根据协议满足这些要求的程度来衡量 AMQP 的成功与否，概述如下。

*普遍性*

1.0 开放式互联网协议标准，支持不受限制的(a) 使用、(b) 实施和(c) 扩展

在互联网基础设施内为业务信息路由和传输提供清晰明确的核心功能--使业务信息传输由基础设施而非集成专家提供

理解、使用和实施的门槛低

1.0 以实用的方式融入现有的企业信息应用环境

*安全*

* 1. 安全可信的全球交易网络基础设施
     + 由防篡改的商业信息组成
     + 支持独立于接收器连接的信息耐用性，以及
     + 信息传递不受技术故障影响

1.0 支持传输任何财务价值的业务交易的业务需求

未来 发送方和接收方是共同商定的对等方 - 不可能注入垃圾邮件

*保真度*

1.0 完善的信息队列和传递语义包括：最多一次；至少一次；一次且仅有一次，又称 "可靠"。

1.0 阐述清晰的信息排序语义，描述发送方可期望(a)接收方观察到的信息和(b)队列管理器观察到的信息

1.0 明确说明可靠的故障语义，以便管理所有异常情况

*适用性*

正如 TCP 包含了网络的所有技术特性一样，我们希望 AMQP 成为企业普遍采用的业务消息传递技术（工具），以便随着使用的增加，投资回报率提高，总体拥有成本降低。

1.0 任何 AMQP 客户端都可以通过 TCP 与任何 AMQP 代理启动通信，然后再与之通信

未来 任何 AMQP 客户端都可以请求与任何 AMQP 代理进行通信，并在支持的情况下协商使用替代传输协议（如 SCTP、UDP/Multicast）。

1.0 通过单个可管理协议提供一套核心消息传递模式：异步定向消息传递、请求/回复、发布/订阅、存储和转发

1.0 支持业务边界内和业务边界间的 Hub & Spoke 消息传递拓扑 未来 支持通过 enactment 实现业务边界间的 Hub to Hub 消息中继

代理机构之间的明确协议 支持在任何网络上进行点对点信息传递

*互操作性*

1.0 多个稳定和互操作的代理实现，每个实现都有完全独立的出处，包括设计、架构、代码和所有权

1.0 每个代理实现都符合规范的所有强制性报文交换和队列功能，包括保真语义

1.0 实施方案可由任何公众成员免费独立测试和验证

1.0 稳定的核心（客户端--代理服务器）线路协议，使代理服务器在运行过程中无需升级

1.x 功能演变：如果 y >= x，任何 1.x 客户端都能与任何 1.y 代理商协同工作

未来稳定的扩展（代理-代理）线协议，使代理在 1.x 功能演进过程中无需升级：任何两个版本为 1.x、1.y 的代理服务器都可以使用协议 1.x 进行通信，条件是 x<y

1.0 分层架构，因此功能和网络传输可由不同的使用社区独立扩展，无需与 AMQP 工作组协调即可与其他系统进行业务整合

*可管理性*

1.0 二进制 WIRE 协议，使其无处不在、快速、嵌入式（可在其上分层添加 XML），使封装系统（如操作系统、中间件、电话）能够提供管理功能

* 1. 具有可扩展性，因此可作为高性能容错无损信息传输基础设施的基础，即无需其他信息传输技术

未来可与信息传送系统进行互动，足以与使用管理标准管理信息传送系统的现行业务运作相结合。

未来中介：支持路由和中继管理、流量管理和服务质量管理

未来的分散部署与独立的地方治理

未来全球寻址在任何网络范围内实现端到端交付标准化

**如何阅读标准**

AMQP 标准分为几本书，分别定义了标准的各个部分。根据您感兴趣的领域，您可能希望从某一本书开始阅读，并使用其他书籍作为参考。

第一册 定义 AMQP 类型系统 第 II 册 定义 AMQP 传输层 第 III 册 定义 AMQP 消息层 第 IV 册 定义 AMQP 事务层 第 V 册 定义 AMQP 安全层

**第一册**

**类型**

* 1. **类型系统**

AMQP 类型系统定义了一组用于互操作数据表示的常用基本类型。AMQP 值可注释原始类型相关信息之外的其他语义信息。这样就可以将 AMQP 值与外部类型关联起来，而外部类型并不作为 AMQP 原始类型存在。例如，URL 通常表示为字符串，但并非所有字符串都是有效的 URL，许多编程语言和/或应用程序都定义了特定类型来表示 URL。AMQP 类型系统允许定义一个代码，当字符串的值要表示 URL 时，可以用它来注释字符串。

## 原始类型

定义了以下基本类型

[空值](#_bookmark8) 表示空值 [布尔值](#_bookmark9) 表示真值或假值

[ubyte](#_bookmark10) 范围在 0 到 2 之间的整数8 - 1（含） [ushort](#_bookmark11) 范围在 0 到 2 之间的整数16 - 1（含） [uint](#_bookmark12) 范围在 0 到 2 之间的整数32 - 1（包含） [ulong](#_bookmark13) 范围在 0 到 2 之间的整数64 - 1（包含）[字节](#_bookmark14) 范围在 *-*(27 ) 至 27 - 1（包含）之间的整数

[短](#_bookmark15) 范围在 *-*(215 ) 至 215 - 1（含）之间的整数 [int](#_bookmark16) *范围在 -*(2 ) 至 23131 - 1（包括 1）之间的整数 [长](#_bookmark17) 范围在 -(263 ) 至 263 - 1（包括 2 ）之间的整数

[浮点数](#_bookmark18) 32 位浮点数（IEEE 754-2008 二进制 32） [double](#_bookmark19) 64 位浮点数（IEEE 754-2008 二进制 64） [decimal32](#_bookmark20) 32 位十进制数（IEEE 754-2008 decimal32） [decimal64](#_bookmark21) 64 位十进制数（IEEE 754-2008 decimal64） [decimal128](#_bookmark22) 128 位十进制数（IEEE 754-2008 decimal128） [char](#_bookmark23) 单个 unicode 字符

[时间戳](#_bookmark24) 绝对时间点

[uuid](#_bookmark25) RFC-4122 第 4.1.2 节定义的通用唯一 ID [二进制](#_bookmark27) 八进制数序列

[字符串](#_bookmark28) 一串统一码字符

[符号](#_bookmark29) 受限域[列表](#_bookmark31)中的符号值 多态值序列

[映射](#_bookmark32) 从不同键到值的多态映射

[数组](#_bookmark34) 一个类型的数值序列

## 描述类型

AMQP 定义的基本类型可直接表示大多数流行编程语言中的许多基本类型，因此可用于交换基本数据。但实际上，即使是最简单的应用程序也有自己的一套自定义类型，用于为应用程序领域内的概念建模，而对于消息传递应用程序，这些自定义类型需要外部化以进行传输。

AMQP 允许任何 AMQP 类型注释*描述符，*从而提供了实现这一目标的方法。*描述符*在自定义类型和 AMQP 类型之间形成关联。这种关联表明，AMQP 类型实际上是自定义类型的*代表*。AMQP 类型及其描述符的组合称为*描述类型*。

描述类型包含两种不同的类型信息。它既能识别 AMQP 类型，也能识别自定义类型（以及它们之间的关系），因此可以在两个不同的层次上理解。对给定领域有深入了解的应用程序可以将描述类型理解为它们所代表的自定义类型，从而根据该领域的完整语义对它们进行解码和处理。而不具备相关知识的应用程序仍可将所描述的类型理解为 AMQP 类型，并据此进行解码和处理。

## 描述符值

符号（[symbol](#_bookmark29)）或数值（[ulong](#_bookmark13)）以外的描述符值，虽然在语法上不是无效的，但都是保留的，这包括 [ulong](#_bookmark13) 以外的数值类型。为了使类型系统的用户能够定义自己的描述符，而不会造成描述符值的冲突，下文给出了符号和数值描述符的赋值策略。

符号和数字描述符的命名空间被划分为不同的域。每个域都有一个由 AMQP 工作组分配的定义符号和/或 4 字节数字 id。对于数字 id，分配的域 id 将等于请求组织 [(](http://www.iana.org/assignments/enterprise-numbers)http://www.iana.org/assignments/enterprise-numbers) 的 IANA 私有企业编号 (PEN)，域 id 0 将保留给 AMQP 规范中定义的描述符。

然后根据以下规则在每个领域内分配描述符：

**符号描述符**

*<域>:<名*

**数字描述符**

(*domain-id <<* 32) *| descriptor-id*

* 1. **类型编码**

AMQP 编码数据流由带有嵌入式构造函数的未键入字节组成。嵌入式构造函数指示如何解释后面的未键入字节。构造函数可被视为从开放式字节流中消耗非类型化字节并构造类型化值的函数。AMQP 编码数据流总是以构造函数开始。

构造字节

| |

+--+ +-----------------+ +

| | | |

... 0xA10x1E "Hello Glorious Messaging World" ...

|| | | |

||utf8 字节 |

|| | |

| 数据字节数 |

| | |

| +-----------------+ +

| |

|字符串值按照

| 到 str8-utf8 编码

|

str8-utf8 编码的原始格式代码

图 1.1：原始格式代码（字符串）

AMQP 构造函数由原始格式代码或描述格式代码组成。原始格式代码是 AMQP 原始类型的构造函数。描述格式代码由描述符和原始格式代码组成。描述符定义了如何从 AMQP 原始值生成特定域类型。

构造字节

| |

+-----------+-----------+ +-----------------+ +

| | | |

...0x00 0xA1 0x03 "URL" 0xA10x1E ["http://example.org/hello-world"](http://example.org/hello-world) ...

|| | | |

+------+------+ | | |

| | | |

描述符 | +------------------+ +

| |

|字符串值按照

| 到 str8-utf8 编码

|

str8-utf8 编码的原始格式代码

(注：本例显示的是字符串类型的描述符，应视为保留描述符）

图 1.2：描述格式代码（URL）

描述格式代码的描述符部分本身就是任何有效的 AMQP 编码值，包括其他描述值。构造函数的正式 BNF 如下。

构造函数 = 格式代码

/ %x00 描述符构造函数

格式代码 = 固定/可变/复合/数组

固定 = 空 / 固定一 / 固定二 / 固定四

/ 固定-8 / 固定-16 可变 = 可变-1 / 可变-4 复合 = 复合-1 / 复合-4

array = array-one / array-four

描述符 = 值

value = 构造函数 untyped-bytes

untyped-bytes = \*OCTET ; 这实际上不是 \*OCTET，而是\*OCTET。

有效字节序列受到限制

；由构造函数

固定宽度格式代码

空 = %x40-4E / %x4F %x00-FF 固定一 = %x50-5E / %x5F %x00-FF 固定二 = %x60-6E / %x6F %x00-FF 固定四 = %x70-7E / %x7F %x00-FF 固定八 = %x80-8E / %x8F %x00-FF 固定十六 = %x90-9E / %x9F %x00-FF

可变宽度格式代码 可变一 = %xA0-AE / %xAF %x00-FF 可变四 = %xB0-BE / %xBF %x00-FF

复合格式代码

化合物一 = %xC0-CE / %xCF %x00-FF 化合物四 = %xD0-DE / %xDF %x00-FF

数组格式代码

array-one = %xE0-EE / %xEF %x00-FF array-four = %xF0-FE / %xFF %x00-FF

图 1.3：构造函数 BNF

格式代码映射到四个不同类别之一：固定宽度、可变宽度、复合宽度和数组。每个类别中的编码值都具有相同的基本结构，以宽度为参数。格式代码中的子类别同时标识了类别和宽度。

**固定宽度** 固定宽度数据的大小完全根据固定宽度值的格式代码子类别确定。

**可变宽度** 可变宽度数据的大小是根据预先固定数据的编码大小确定的。编码大小的宽度由可变宽度值的格式代码子类别决定。

**复合数据** 复合数据以大小和计数的形式编码，后面是*计数*组成值的多态序列。每个组成值前面都有一个构造函数，用于指示后面数据的语义和编码。大小和计数的宽度由复合值格式代码的子类别决定。

**数组** 数组数据编码为大小和计数，然后是数组元素构造函数，最后是根据所提供的数组元素构造函数编码的单态值序列。大小和计数的宽度由数组格式代码的子类别决定。

格式代码中的位可按以下布局进行解释：

位： 7 6 5 4 3 2 1 0

+------------------------------------+ + +

| 子类别 | 子类

+------------------------------------+ + +

1 个八位字节 1 个八位字节

| |

+ +

| 格式代码

ext-type：仅在子类型为 0xF 时出现

下表描述了格式代码的子类别：

子类别 类别 格式

==============================================================================

0x4 固定 宽度零八位位组数据。

0x5 固定 宽度一个八位位组数据。

0x6 固定宽度 两个八位位组的数据。

0x7 固定宽度 四个八位位组数据。0x8 固定 八个八位位组的数据。0x9 固定宽度 八位位组数据。

0xAVariable Width（变量宽度） 一个八位位组的大小，0-255 个八位位组的数据。

0xB 变量宽度 四个八位位组的大小，0-4294967295 个八位位组的数据。

0xC 复合大小和计数各一个八位位组，0-255 各不相同

类型值。

0xD 复合大小和计数各四个八进制数，0-4294967295

不同类型的值。

0xE 数组大小和计数各一个八位位组，0-255 一致

类型值。

0xF 数组大小和计数各 四个八进制数，0-4294967295

统一类型的值。

请注意，除非另有说明，AMQP 对所有数值都使用网络字节序。

## 固定宽度

特定固定宽度编码的宽度可根据固定宽度值的格式代码子类别计算得出：

n OCTET

+ +

| 数据 |

+ +

子类别 n

=================

|  |  |
| --- | --- |
| 0x4 | 0 |
| 0x5 | 1 |
| 0x6 | 2 |
| 0x7 | 4 |
| 0x8 | 8 |
| 0x9 | 16 |

**类型：**空

<type name="null" class="原始"/>。

**编码 编码 类别 说明**

0x40 固定宽度，0 字节值 *空值*

**类型：**布尔

<type name="boolean" class="primitive"/> 类型

|  |  |  |  |
| --- | --- | --- | --- |
| **编码** | **代码** | **类别** | **说明** |
|  | 0x56 | 固定宽度，1 字节值 | *布尔型，八位位组 0x00 为假，八位位组 0x01 为真* |
| 真 | 0x41 | 固定宽度，0 字节值 | *布尔值 true* |
| 错误 | 0x42 | 固定宽度，0 字节值 | *布尔值 false* |

**类型：** ubyte

<type name="ubyte" class="primitive"/>

**编码 编码 类别 说明**

0x50 固定宽度，1 字节值 *8 位无符号整数*

**类型：**短裤

<type name="ushort" class="原始"/>类型

**编码 编码 类别 说明**

0x60 固定宽度，2 字节值 *按网络字节顺序排列的 16 位无符号整数*

**类型：** uint

<type name="uint" class="原始"/>类型

|  |  |  |  |
| --- | --- | --- | --- |
| **编码** | **代码** | **类别** | **说明** |
|  | 0x70 | 固定宽度，4 字节值 | *按网络字节顺序排列的 32 位无符号整数* |
| 小燧石 | 0x52 | 固定宽度，1 字节值 | *范围在 0 至 255（含 255）之间的无符号整数值* |
| uint0 | 0x43 | 固定宽度，0 字节值 | *uint 值 0* |

**类型：**乌龙

<type name="ulong" class="原始"/>类型

|  |  |  |  |
| --- | --- | --- | --- |
| **编码** | **代码** | **类别** | **说明** |
|  | 0x80 | 固定宽度，8 字节值 | *按网络字节顺序排列的 64 位无符号整数* |
| 小龙 | 0x53 | 固定宽度，1 字节值 | *范围在 0 至 255 之间的无符号长值* |
| ulong0 | 0x44 | 固定宽度，0 字节值 | *ulong 值 0* |

**类型：**字节

<type name="byte" class="primitive"/>

**编码 编码 类别 说明**

0x51 固定宽度，1 字节值 *8 位二进制整数*

**类型：**短

<type name="short" class="primitive"/>

**编码 编码 类别 说明**

0x61 固定宽度，2 字节值 *按网络字节顺序排列的 16 位两位补码整数*

**类型：**int

<type name="int" class="primitive"/>

|  |  |  |  |
| --- | --- | --- | --- |
| **编码 编码** | **类别** | **说明** |  |
| 0x71 | 固定宽度，4 字节值 | *32 位二元补整数字节顺序* | *在网络中* |
| 小整数 0x54 | 固定宽度，1 字节值 | *范围内的带符号整数值* | *-128至127* |

**类型：**长

<type name="long" class="primitive"/>

|  |  |  |
| --- | --- | --- |
| **编码 编码** | **类别** | **说明** |
| 0x81 | 固定宽度，8 字节值 | *按网络字节顺序排列的 64 位两位补码整数* |
| smalllong 0x55 | 固定宽度，1 字节值 | *范围在 -128 至 127 之间的带符号长数值。* |

**类型：**浮点

<type name="float" class="primitive"/>

**编码 编码 类别 说明**

ieee-754 0x72固定宽度 ，4 字节值 *IEEE 754-2008 二进制 32*

**类型：**双

<type name="double" class="primitive"/>

**编码 编码 类别 说明**

ieee-754 0x82固定宽度 ，8 字节值 *IEEE 754-2008 二进制 64*

**类型：**十进制 32

<type name="decimal32" class="primitive"/>

**编码 编码 类别 说明**

ieee-754 0x74固定宽度 ，4 字节值 *IEEE 754-2008 decimal32 使用二进制*

*整数 十进制编码*

**类型：**十进制64

<type name="decimal64" class="primitive"/>

**编码 编码 类别 说明**

ieee-754 0x84固定宽度 ，8 字节值 *使用二进制的 IEEE 754-2008 decimal64*

*整数 十进制编码*

**类型：**十进制 128

<type name="decimal128" class="primitive"/>

**编码 编码 类别 说明**

ieee-754 0x94固定宽度 ，16 字节值 *使用二进制的 IEEE 754-2008 十进制128*

*整数 十进制编码*

**类型：**字符

<type name="char" class="原始"/>类型

**编码代码 类别 说明**

utf32 0x73 固定宽度、4 字节值 *UTF-32BE 编码的 unicode 字符*

**类型：**时间戳

<type name="timestamp" class="primitive"/>

**编码代码 类别 说明**

ms64 0x83 固定宽度、8 字节值 *64 位带符号整数，表示毫秒数*

*自 unix 时代以来*

表示使用 UTC 的 Unix 时间 t 编码的近似时间点，但精度为毫秒。例如，1311704463521 表示 2011-07-26T18:21:03.521Z 时刻。

**类型：**uuid

<type name="uuid" class="primitive"/>

**编码代码 类别 说明**

0x98 固定宽度，16 字节值 *RFC-4122 第 4.1.2 节中定义的 UUID*

## 可变宽度

所有可变宽度编码都由以八位位组为单位的大小和编码数据的八位位组*大小*组成。特定可变宽度编码的大小宽度可根据格式代码的子类别计算得出：

n OCTETssize OCTETs

+----------+ +

| 大小 | 值 |

+----------+ +

子类别 n

================= 0xA 1

0xB 4

**类型：**二进制

<type name="binary" class="primitive"/>

|  |  |  |  |
| --- | --- | --- | --- |
| **编码** | **代码** | **类别** | **说明** |
| vbin8 | 0xa0 | 可变宽度，1 字节大小 | *最多* 28 *- 1 个八进制数据字节* |
| vbin32 | 0xb0 | 可变宽度，4 字节大小 | *最多* 232 *- 1 个八进制数据字节* |

**类型：**字符串

<type name="string" class="primitive"/>

字符串表示 Unicode V6.0.0 标准定义的 unicode 字符序列（见 [http://www.unicode.org/versions/Unicode6.0.0）。](http://www.unicode.org/versions/Unicode6.0.0))

**编码 编码 类别 说明**

str8-utf8 0xa1 变宽 ，1 字节大小 *最多* 28 *- 1 个八进制字节的 UTF-8 统一字符编码*

*(无字节序号）*

str32-utf8 0xb1 可变宽度，4 字节大小 *最多* 232 *- 1 个八进制值的 UTF-8 统一码*

*(无字节序号）*

**类型：**符号

<type name="symbol" class="primitive"/> 类型名称为 "符号"。

符号是来自受限域的值。虽然可能的域集是开放的，但在任何特定应用中，符号的数量和大小通常都很小，例如，小到足以缓存所有不同的值。

|  |  |  |  |
| --- | --- | --- | --- |
| **编码** | **代码** | **类别** | **说明** |
| sym8 | 0xa3 | 可变宽度，1 字节大小 | *最多* 2 个8 *- 1 个七位 ASCII 字符，代表"......"。* |

*发送符号值*

sym32 0xb3 可变宽度，4 字节大小 *最多* 232 *- 1 个七位 ASCII 字符代表*

*象征意义*

## 化合物

所有复合编码都由大小和计数组成，后面是*计数*编码项。特定复合编码的大小和计数宽度可根据格式代码的类别计算得出：

+----------= count items = +

| |

n OCTETsn OCTETs |

+----------+----------+--------------+------------+ +

| size| count | ... /| 项目 |\ |

+----------+----------+------------/ +------------+ \ +

/ /\ \

/ /\ \

/ /\ \

+-------------+ +

| 构造函数 数据 |

+-------------+ +

子类别 n

================= 0xC 1

0xD 4

**类型：**列表

<type name="list" class="primitive"/>

|  |  |  |  |
| --- | --- | --- | --- |
| **编码** | **代码** | **类别** | **说明** |
| 列表0 | 0x45 | 固定宽度，0 字节值 | *空列表（即没有元素的列表）* |

list8 0xc0 可变宽度，1 字节大小 *最多* 2 个8 *- 1 个列表元素，总大小小于*

*超过* 28 *个八位字节*

list32 0xd0 可变宽度，4 字节大小 *最多* 2 个32 *- 1 个列表元素，总大小小于*

*超过* 232 *个八位字节*

**类型：**地图

<type name="map" class="primitive"/>

地图以复合值的形式编码，其中的组成元素形成交替的键值对。

项目 0 项目 1 项目 n-1 项目 n

+-------+-------+----+---------+ +

| 键 1 | 值 1 | ...| 键 n/2 | 值 n/2 | .

+-------+-------+----+---------+ +

映射编码必须包含偶数个项目（即键和值的数量相等）。存在两个相同键值的映射是无效的。除非已知情况并非如此，否则必须将地图视为有序的--即键值对的顺序在语义上非常重要，两个仅在键值对编码顺序上不同的地图是不相等的。

|  |  |  |  |
| --- | --- | --- | --- |
| **编码** | **代码** | **类别** | **说明** |
| 地图8 | 0xc1 | 可变宽度，1 字节大小 | *最多* 28 *- 1 个八进制编码地图数据* |
| 地图32 | 0xd1 | 可变宽度，4 字节大小 | *最多* 232 *- 1 个八进制编码地图数据* |

## 阵列

所有数组编码都由大小、计数、元素构造器和按元素构造器要求格式化的编码数据的*计数*元素组成：

+--= 计数元素 =--+

| |

n OCTETsn OCTETs | |

+----------+----------+---------------------+-------+------+ +

| size| count| element-constructor | ...| 数据 |

+----------+----------+---------------------+-------+------+ +

子类别 n

================= 0xE 1

0xF 4

**类型：**数组

<type name="array" class="primitive"/>

|  |  |  |  |
| --- | --- | --- | --- |
| **编码** | **代码** | **类别** | **说明** |
| 数组8 | 0xe0 | 可变宽度，1 字节大小 | *最多* 28 *- 1 个总大小小于* 28 *八进制数的数组元素* |
| 数组32 | 0xf0 | 可变宽度，4 字节大小 | *最多* 232 *- 1 个总大小小于* 232 *八进制数的数组元素* |

## 编码列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类型** | **编码** | **代码** | **类别** | **说明** |
| [无效](#_bookmark8) |  | 0x40 | 固定/0 | 空值 |
| [布尔](#_bookmark9) |  | 0x56 | 固定/1 | 布尔值，八位位组 0x00 为假，而 |
|  |  |  |  | 八进制 0x01 为真 |
| [布尔](#_bookmark9) | 真 | 0x41 | 固定/0 | 布尔值 true |
| [布尔](#_bookmark9) | 错误 | 0x42 | 固定/0 | 布尔值 false |
| [宇字节](#_bookmark10) |  | 0x50 | 固定/1 | 8 位无符号整数 |
| [短裤](#_bookmark11) |  | 0x60 | 固定/2 | 按网络字节顺序排列的 16 位无符号整数 |
| [整数](#_bookmark12) |  | 0x70 | 固定/4 | 按网络字节顺序排列的 32 位无符号整数 |
| [二进制](#_bookmark12) | 小燧石 | 0x52 | 固定/1 | 范围在 0 至 255 之间的无符号整数值 |
|  |  |  |  | 包容性 |
| [二进制](#_bookmark12) | uint0 | 0x43 | 固定/0 | uint 值 0 |
| [乌龙](#_bookmark13) |  | 0x80 | 固定/8 | 按网络字节顺序排列的 64 位无符号整数 |
| [乌龙](#_bookmark13) | 小龙 | 0x53 | 固定/1 | 范围在 0 至 255 之间的无符号 long 值。 |
|  |  |  |  | 阉割 |
| [乌龙](#_bookmark13) | ulong0 | 0x44 | 固定/0 | ulong 值 0 |
| [字节](#_bookmark14) |  | 0x51 | 固定/1 | 8 位二进制整数 |
| [短](#_bookmark15) |  | 0x61 | 固定/2 | 网络中的 16 位二元补整数 |
|  |  |  |  | 字节序 |
| [int](#_bookmark16) |  | 0x71 | 固定/4 | 网络中的 32 位二元补整数 |
|  |  |  |  | 字节序 |
| [int](#_bookmark16) | 小数点 | 0x54 | 固定/1 | 范围在 -128 至 127 之间的带符号整数值 |
|  |  |  |  | 包容性 |
| [长](#_bookmark17) |  | 0x81 | 固定/8 | 网络中的 64 位二元补整数 |
|  |  |  |  | 字节序 |
| [长](#_bookmark17) | 小龙 | 0x55 | 固定/1 | 范围在 -128 至 127 之间的带符号 long 值。 |
|  |  |  |  | 阉割 |
| [浮动](#_bookmark18) | IEEE-754 | 0x72 | 固定/4 | IEEE 754-2008 二进制 32 |
| [双人](#_bookmark19) | IEEE-754 | 0x82 | 固定/8 | IEEE 754-2008 二进制 64 |
| [小数32](#_bookmark20) | IEEE-754 | 0x74 | 固定/4 | IEEE 754-2008十进制32使用二进制输入法。 |
|  |  |  |  | teger 十进制编码 |
| [小数64](#_bookmark21) | IEEE-754 | 0x84 | 固定/8 | IEEE 754-2008十进制64使用二进制输入法。 |
|  |  |  |  | teger 十进制编码 |
| [小数128](#_bookmark22) | IEEE-754 | 0x94 | 固定/16 | IEEE 754-2008 十进制128 使用二进制 |
|  |  |  |  | 整数 十进制编码 |
| [烧掉](#_bookmark23) | utf32 | 0x73 | 固定/4 | UTF-32BE编码的统一码字符 |
| [时戳](#_bookmark24) | ms64 | 0x83 | 固定/8 | 代表毫秒的 64 位有符号整数 |
|  |  |  |  | 自 unix 时代以来 |
| [uuid](#_bookmark25)  [二进制](#_bookmark27) | vbin8 | 0x98  0xa0 | 固定/16  变量/1 | RFC-4122 第 4.1.2 节中定义的 UUID  最多 28 - 1 个八进制数据字节 |
| [二进制](#_bookmark27) | vbin32 | 0xb0 | 变量/4 | 最多 232 - 1 个八进制数据字节 |

[字符串](#_bookmark28) str8-utf8 0xa1 变量/1 最多 28 - 1 个八进制数值的 UTF-8 unicode

(无字节序号）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类型** | **编码** | **代码** | **类别** | **说明** |
| [字符串](#_bookmark28) | str32-utf8 | 0xb1 | 变量/4 | 最多 232 - 1 个八进制数值的 UTF-8 统一码 |
| [符号](#_bookmark29) | sym8 | 0xa3 | 变量/1 | (无字节序号）  最多 28 - 1 个七位 ASCII 字符，代表"......"。 |
| [符号](#_bookmark29) | sym32 | 0xb3 | 变量/4 | 发送符号值  多达 232 - 1 个七位 ASCII 字符代表 |
|  |  |  |  | 象征意义 |
| [清单](#_bookmark31) | 列表0 | 0x45 | 固定/0 | 空列表（即没有元素的列表） |
| [清单](#_bookmark31) | 列表8 | 0xc0 | 复式/1 | 最多 2 个8 - 1 个列表元素，总大小小于 |
| [清单](#_bookmark31) | 列表32 | 0xd0 | 复式/4 | 超过 28 个八位字节  最多 2 个32 - 1 个列表元素，总大小小于 |
| [地图](#_bookmark32) | 地图8 | 0xc1 | 复式/1 | 超过 232 个八位字节  最多 28 - 1 个八进制编码地图数据 |
| [地图](#_bookmark32) | 地图32 | 0xd1 | 复式/4 | 最多 232 - 1 个八进制编码地图数据 |
| [矩阵](#_bookmark34) | 数组8 | 0xe0 | array/1 | 最多 2 个8 - 1 个数组元素，总大小小于 |
| [矩阵](#_bookmark34) | 数组32 | 0xf0 | 阵列/4 | 超过 28 个八位字节  最多 2 个32 - 1 个数组元素，总大小小于 |
|  |  |  |  | 超过 232 个八位字节 |

* 1. **复合类型**

AMQP 定义了许多*复合类型*，用于编码帧体等结构化数据。复合类型描述一个复合值，其中每个组成值都由一个已知的命名*字段*标识。每个复合类型定义都包含一个有序的字段序列，每个字段都有指定的名称、类型和多重性。复合类型定义还包括一个或多个描述符（符号和/或数字），用于标识其定义的表示形式。

本规范随附的 XML 文档中正式定义了复合类型。使用以下符号来定义它们：

<type class="composite" name="book" label="示例复合类型">。

<doc

<p>复合类型示例。

</doc

<descriptor name="example:book:list" code="0x00000003:0x00000002"/>

<field name="title" type="string" mandatory="true" label="书名"/>。

<field name="authors" type="string" multiple="true"/>

<field name="isbn" type="string" label="图书的 ISBN 代码"/>"ISBN "字段

</类型

图 1.4：复合类型示例

字段描述的*强制*属性控制着是否允许在表示中使用空元素值。

字段描述的 *multiple* 属性控制着是否允许在表示中使用多个元素值。字段描述中指定类型的单个元素始终是允许的。多元素值通过使用数组表示，数组中元素的类型是字段定义中定义的类型。请注意，空值和零长数组（其元素的类型正确）都表示没有值，在语义上应视为相同。

定义为多重和必填的字段必须至少包含一个值（即对于此类字段，*空值*和无条目数组均无效）。

## 列表编码

AMQP 复合值以描述列表的形式编码。列表中的每个元素都与复合类型定义中列出的字段位置相关。允许的元素值由类型规范和相应字段定义的多重性决定。当列表表示的尾部元素为空时，可以省略。列表描述符表示所代表的特定复合类型。

下面描述的列表是上文定义的*图书*类型的复合值示例。示例中描述了一个与没有 ISBN 值相对应的尾部空元素，但可根据编码规则选择省略。

一本书的 建构表

| |

+-----------------+-------------------+ +-------------+ +

|| | |

0x00 0xA3 0x11 "example:book:list" 0xC0 0x40 0x03 title authors isbn

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | |  确定复合类型 | |  | | |  | | |  | |
| | | | | | | 0x40 | |
| sym8 | + + | | | | | |
| (符号) | |  +--------------+ +  | |  0xA1 0x15 "傻瓜版 AMQP" | | | 空值  |  |  | | |

|

+------------------------------------------------------------+ +

| |

0xE0 0x25 0x02 0xA1 0x0E "Rob J. Godfrey" 0x13 "Rafael H. Schloming"

| | | || 大小 | | +---------+---------+ +-----------+------------+

| | | |

计数 | 第一元素 第二元素

|

元素构造函数

图 1.5：综合值示例

**第二册**

**运输**

* 1. **运输**

AMQP 网络由通过*链接*连接的*节点*组成。节点是负责安全存储和/或*传递信息*的命名实体。信息可以从节点发出、在节点终止或由节点转发。

链接是两个节点之间的单向路由。链接在*终点站*连接到节点。总站有两种：*源*和*目标*。总站负责跟踪特定传入或传出信息流的状态。源跟踪发出的信息，目标跟踪接收的信息。信息只有在符合源站的输入标准时，才能沿着链接传输。

当消息在 AMQP 网络中传输时，安全存储和传递消息的责任会在遇到的节点之间转移。链接协议（在[第 2.6 节 "链接](#_bookmark59) "中定义）管理源和目标之间的责任转移。

+------------+ + +

/ 节点 A \ / 节点 B \

+----------------+ +- 过滤器 + +

| | / | |

| msg\_3 <msg\_1> | \_/\_ |msg\_1 | \_/\_ |msg\_1

| |(\_)------------------>(\_)| |

| <msg\_2> msg\_4 | | | msg\_2 |

|| | 链接(Src,Tgt) |||。 |

+----------------+ || + +

| |

来源 Tgt

键：<MSG\_n> = MSG\_n 的旧位置

节点存在于*容器*中，每个容器可容纳多个节点。AMQP 节点的例子包括生产者（Producers）、消费者（Consumer）和队列（Queues）。生产者和消费者是客户端应用程序中生成和处理信息的元素。队列是 Broker 中存储和转发消息的实体。代理和客户端应用程序就是容器的例子。

+---------------+ + +

| <<Container>> | 1... 10..n | <<Node>> | 1... 10..n

|---------------|<>-------------------->| |

|  |  |
| --- | --- |
| | container-id |  + + | | 名称 |  + + |
| /\_\ | /\_\ |
| | | | |
| | | | |
| +-----+ + | +----------+ + |
| | | | | | | |
| | | | | | | |
| +--------+ + + | +----------+ +----------+ + + |
| | 经纪人 | | 客户 | | | 生产者队列 |
| |--------| | | | |----------| |----------| | | |

|| | | || || | |

+--------+ +--------++----------+ +----------+ + +

AMQP 传输规范（AMQP Transport Specification）定义了在 AMQP 网络中节点间传输消息的点对点协议。该规范的这一部分不涉及任何节点的内部运作，只涉及将信息从一个节点明确传输到另一个节点的机制。

容器通过*连接进行*通信。AMQP 连接由全双工、可靠有序的*帧*序列组成。对连接的精确要求是，如果th 第 n 个帧到达，则 n 个之前的所有帧也必须到达。假设 "连接 "是瞬时的，可能会因各种原因而失败，导致丢失未知数量的帧，但它们仍须遵守上述有序可靠性标准。这与 TCP 或 SCTP 为字节流提供的保证类似，本规范定义了一个成帧系统，用于将字节流解析为用于建立 AMQP 连接的帧序列（见[第 2.3 节 "成帧](#_bookmark41)"）。

一个 AMQP 连接被分成若干个协商好的独立单向*通道*。每个 "帧 "都标有表示其父通道的通道号，每个通道的 "帧 "序列被复用为连接的单个 "帧 "序列。

AMQP *会话*将两个单向通道关联起来，形成两个容器之间的双向顺序转换。单个连接可同时有多个独立的会话处于活动状态，但最多不超过协商的通道限制。连接和会话都被每个对等*端点*建模为*端点*，用于存储有关连接或会话的本地状态和最后已知的远程状态。

会议<------+ + >会话

(ICH=1, OCH=1) | |(ICH=1，OCH=1)

\|/ \|/

Session<--> Connection <---------> Connection <-->Session (ICH=2, OCH=3) /|\\ /|\ (ICH=3, OCH=2)

| |

会议<------+ + >会话

(ICH=3, OCH=2) (ICH=2, OCH=3)

按键ICH -> 输入通道，OCH -> 输出通道

图 2.1：会话和连接端点

会话为 "源 "和 "目标 "之间的通信提供了上下文。*链接端点*将终端与*会话端点*关联起来。在会话中，链接协议（定义见[第 2.6 节 "链接](#_bookmark59)"）用于在源和目标之间建立链接，并在它们之间传输信息。一个会话可同时与任意数量的链接相关联。

+ +

| 链接

+-------------+ （节点到节点）

| 名称 |

| 来源 |

| 目标 |

| 超时 |

+ +

/|\ 0..n

|

|

|

\|/ 0..1

+ +

| 帧传输

+------------+ （集装箱到集装箱）

| 名称 |

+ +

/|\ 0..n

|

|

|

\|/ 1..1

+ +

| 连接 | 帧传输

+------------+ （集装箱到集装箱）

| 校长

+ +

帧是有线传输的工作单位。连接有一个协商的最大帧大小，允许将字节流轻松碎片化为完整的帧体，代表第 [2.7 节中](#_bookmark74)正式定义的可独立解析[的](#_bookmark74)单元。下表列出了所有帧体，并定义了处理这些帧体的端点。

帧连接会话链接

========================================

打开 H

开始 I H

附上 I H

流动 I H

转让 I H

我 I H

脱离 I H

结束 I H

关闭 H

钥匙

H：由终端处理

I：拦截（端点检查帧，但委托其他端点进一步处理）

* 1. **版本协商**

在连接上发送任何帧之前，每个对等方必须先发送一个协议标头，标明连接上使用的协议版本。协议头由大写 ASCII 字母 "AMQP"（AMQP）和一个协议 ID（0）组成，后跟三个无符号字节，分别代表协议版本的主版本、次版本和修订版本（目前为 [1（主](#_bookmark109)版本）、0（次版本）、[0（修订版本](#_bookmark109)））。这总共是一个 8 字节的序列：

4 个 八位字节1 个八位字节 1 个八位字节 1 个八位字节

+----------+---------+---------+---------+ +

| "AMQP" | "AMQP %d0| 主版本 | 次版本 | 修订版本 | 主版本 | 次版本 | 修订版本

+----------+---------+---------+---------+ +

出现在协议标头之外的任何数据都必须与协议标头指示的版本相匹配。如果传入和传出的协议头不匹配，两个对等设备都必须关闭其传出数据流，并应读取传入数据流，直至其终止。

扮演 TCP 客户端角色的 AMQP 对等体（即打开连接的对等体）必须在建立 TCP 会话时立即发送其传出协议头。扮演 TCP 服务器角色的 AMQP 对等体可以选择等待，直到收到传入协议标头后再发送自己的传出协议标头。

两个 AMQP 对等方商定的协议版本如下（这里的 "客户端 "和 "服务器 "指的是对等方在 TCP 连接级别扮演的角色）：

* 当客户端打开与服务器的新套接字连接时，必须发送一个包含客户端首选协议版本的协议头。
* 如果请求的协议版本受支持，服务器必须向套接字发送自己的协议头和请求的版本，然后根据协议定义继续执行。
* 如果请求的协议版本**不受**支持，服务器必须发送一个**受支持**协议版本的协议头，然后关闭套接字。
* 在选择响应的协议版本时，服务器应选择小于或等于请求版本的最高支持版本。如果不存在此类版本，服务器应使用最高支持版本进行响应。
* 如果服务器无法解析协议标头，服务器必须发送一个有效的协议标头和一个受支持的协议版本，然后关闭套接字。

根据这种行为，客户端可以通过尝试连接服务器支持的最高版本，并以小于或等于服务器返回的版本重新连接，从而发现服务器支持的协议版本。

TCP 客户端TCP 服务器

====================================================== AMQP%d0.1.0.0 ------------->

<------------- AMQP%d0.1.0.0 (1)

... \*继续\*

AMQP%d0.1.1.0 >

<------------- AMQP%d0.1.0.0 (2)

\*TCP CLOSE\*

HTTP >

<------------- AMQP%d0.1.0.0 (3)

\*TCP CLOSE\*

1. 服务器接受以下连接AMQP, protocol=0, major=1, minor=0, revision=0
2. 服务器拒绝连接：AMQP，protocol=0，major=1，minor=1，revision=0，服务器响应支持：AMQP、协议=0、主协议=1、次协议=0、修订版=0
3. 服务器拒绝连接：HTTP。服务器回应支持AMQP, protocol=0, major=1, minor=0, revision=0

图 2.2：版本协商示例

请注意，上述示例对非字母数字值使用了 RFC 2234 中定义的文字符号。

协议 id 不是协议版本的一部分，因此上述关于最高支持版本的规则并不适用。客户端可能会请求使用服务器无法接受的协议 ID，例如，当服务器配置为需要 TLS 或 SASL 安全层时，客户端可能会请求使用原始 AMQP 连接（参见[第 5.1 节 "安全层](#_bookmark176)"）。在这种情况下，服务器必须发送带有**可接受协议** ID（和版本）的协议头，然后关闭套接字。服务器可以选择任何协议 ID。

TCP 客户端TCP 服务器

====================================================== AMQP%d0.1.0.0 ------------->

<------------- AMQP%d3.1.0.0

\*TCP CLOSE\*

服务器拒绝连接：AMQP，protocol=0，major=1，minor=0，revision=0，服务器响应需要：SASL 安全层，协议=3，major=1，minor=0，修订版=0

图 2.3：协议 ID 拒绝示例

* 1. **框架**

帧分为三个不同的区域：固定宽度的帧头，可变宽度的扩展帧头和可变宽度的帧体。

必要 可选 可选

+--------------+-----------------+ +

| 帧头 | 扩展帧头 | 帧体

+--------------+-----------------+ +

8 个字节 \*变量 \*变量

**帧头** 帧头是一个固定大小（8 字节）的结构，位于每个帧之前。帧头包含解析帧其余部分所需的必填信息，包括大小和类型信息。

**扩展页眉** 扩展页眉是框架主体之前的一个宽度可变的区域。这是一个为未来扩展而定义的扩展点。该区域的处理方式取决于帧类型。

**帧体** 帧主体是宽度可变的字节序列，其格式取决于帧类型。

## 框架布局

下图显示了所有机架类型的一般机架布局细节。

+0 +1 +2 +3

+-----------------------------------+ -.

0 | SIZE| |

+-----------------------------------+ |---> 帧头

4 | DOFF | TYPE | <TYPE-SPECIFIC> | | （8 字节）

+-----------------------------------+ -’

+-----------------------------------+ -.

8 | ... | |

. .|---> 扩展标头

. <特定类型 >.| (DOFF \* 4 - 8) 字节

| ... | |

+-----------------------------------+ -’

+-----------------------------------+ -.

4\*DOFF || | |

. .|

. .|

. .|

. <特定类型 >.|---> 帧主体

. .| (SIZE - DOFF \* 4) 字节

. .|

. .|

. | |

| ... | |

+--------------------------+ -'

**大小** 帧头的 0-3 字节包含帧大小。这是一个无符号 32 位整数，必须包含帧头、扩展帧头和帧主体的总帧大小。如果大小小于所需的帧头大小（8 字节），则帧为畸形。

**DOFF** 帧头的第 4 个字节是数据偏移量。它给出了主体在帧中的位置。数据偏移值是无符号 8 位整数，指定 4 字节字的计数。由于帧头必须为 8 字节，如果该值小于 2，帧就会变形。

**类型** 帧头的第 5 个字节是类型代码。类型代码表示帧的格式和目的。根据帧的类型，帧头中的后续字节可能会有不同的解释。类型代码为 0x00 表示帧是 AMQP 帧。(类型代码为 0x01 表示帧是 SASL 帧，参见[第 5.3 节 SASL](#_bookmark181)）。

## AMQP 帧

AMQP 帧的第 6 和第 7 字节包含通道编号（见[第 2.1 节 "传输](#_bookmark39)"）。帧主体被定义为一个*执行符，*其后是一个不透明的*有效载荷*。执行符必须是[第 2.7 节中](#_bookmark74)定义的[执行](#_bookmark74)符之一，并在 AMQP 类型系统中编码为描述类型。帧主体中的剩余字节构成该帧的有效载荷。有效载荷的存在和格式由给定执行符的语义定义。

类型：0x00 - AMQP 帧

+0 +1 +2 +3

+-----------------------------------+ -.

0 | SIZE| |

+-----------------------------------+ |---> 帧头

4 | doff | type | CHANNEL| | （8 字节）

+-----------------------------------+ -’

+-----------------------------------+ -.

8 | ... | |

. .|---> 扩展标头

. <IGNORED >.| (DOFF \* 4 - 8) 字节

| ... | |

+-----------------------------------+ -’

+-----------------------------------+ -.4\*DOFF | | performative: | | |

. Open / Begin / Attach .|

. Flow / Transfer / Disposition .|

. Detach / End / Close .|

|-----------------------------------| |

. .|---> 车架主体

. .| (SIZE - DOFF \* 4) 字节

. PAYLOAD .|

. .|

. | |

| ... | |

+--------------------------+ -'

无主体的 AMQP 帧可用于产生所需的人工流量，以满足任何协商的空闲超时时间间隔。请参阅 [2.4.5 连接的空闲超时](#_bookmark49)。

* 1. **连接**

AMQP 连接分为多个单向通道。一个连接端点包含两种通道端点：传入和传出。连接端点会根据传入通道的编号，将[开放](#_bookmark75)和[关闭](#_bookmark83)以外的传入帧映射到传入通道端点，并中继由传出通道端点产生的帧，在发送前用相关的传出通道编号标记这些帧。

这要求连接端点包含两个映射。一个是从传入通道编号到传入通道端点，另一个是从传出通道端点到传出通道编号。

+-------OCHE X: 1

|

+-------OCHE Y: 7

|

<=== 帧[CH=1]，帧[CH=7] <===+

===> 帧[CH=0]，帧[CH=1] ===>+

|

+------>0: ICHE A

|

+------>1：ICHE B

OCHE：传出通道端点 ICHE：传入通道端点

通道是单向的，因此在每个连接端点，输入和输出通道是完全不同的。通道编号的作用域是相对于方向而言的，因此输入和输出通道之间没有因果关系，而这些通道恰好由 "相同 "的编号标识。这意味着，如果一个双向端点是由一个传入通道端点和一个传出通道端点构成的，则用于传入帧的通道编号不一定与用于传出帧的通道编号相同。

+-------BIDI/O：7

|

<=== 帧[CH=1]，帧[CH=7] <===+

===> 帧[CH=0]，帧[CH=1] ===>+

|

+------>1：BIDI/I

BIDI/I： 单个双向端点传入的一半 BIDI/O：单个双向端点传出的一半

虽然严格来说，[开始](#_bookmark76)和[结束](#_bookmark82)帧并不是针对连接端点的，但连接端点可以截获这些帧，因为这些帧是会话在特定通道上标记通信开始和结束的方式（请参阅[第 2.5 节会话](#_bookmark52)）。

## 打开连接

每个 AMQP 连接都以交换能力和限制（包括最大帧大小）开始。在进行任何显式协商之前，最大帧大小为 [512（MIN-MAX- FRAME-SIZE）](#_bookmark109)，最大通道数为 0。 在建立或接受 TCP 连接并发送协议头之后，每个对等方必须在发送任何其他帧之前发送一个[开放](#_bookmark75)帧。[开放](#_bookmark75)帧描述了该对等方的能力和限制。[开放](#_bookmark75)帧只能在信道 0 上发送。发送[开放](#_bookmark75)帧后，每个对等端都必须读取其伙伴的[开放](#_bookmark75)帧，并且必须在双方都能接受的限制范围内运行。

TCP 客户端TCP 服务器

================================== TCP-CONNECT TCP-ACCEPT

PROTO-HDR PROTO-HDR OPEN ---++--- 打开

\ /

等待 x 等待

/ \

继续 <--++--> 继续

...

## 流水线开放式

对于使用许多短时连接的应用程序来说，可能需要将连接协商过程管道化。对等方可在收到对方的 "连接 "标头或[开放](#_bookmark75)帧之前开始发送后续帧。只要事先知道流水线帧符合对方的能力和限制，就可以这样做。例如，这可以通过将 Connection 的使用保持在[开放](#_bookmark75)帧规范所定义的所有 AMQP 实现的能力和限制范围内来实现。

TCP 客户端TCP 服务器

============================================= TCP-CONNECT TCP-ACCEPT

PROTO-HDR PROTO-HDR

开放 ---++--- 开放

\ /

流水线框架 xpipelined frame

/ \

继续 <--++--> 继续

...

只要流水线帧符合对等方的能力和限制，对等方的伙伴就无法区分对等方使用的流水线帧和非流水线帧。

## 关闭连接

在关闭连接之前，每个对等端都必须写入一个[关闭](#_bookmark83)帧，并用代码说明关闭的原因。该帧必须是最后写入连接的内容。写入此帧后，对等方应继续从连接中读取数据，直到收到伙伴的[关闭](#_bookmark83)帧为止（为了防止伙伴错误或恶意执行，对等方应实施超时，在放弃并简单关闭底层传输机制之前，给伙伴一个合理的时间来接收和处理关闭帧）。[关闭](#_bookmark83)帧可在任何信道上接收，但不得超过开放时协商的最大信道数。但是，实现应在信道 0 上发送，如果与相应的 [open](#_bookmark75) 一起以流水线方式发送，则必须在信道 0 上发送。

TCP 客户端TCP 服务器

=============================

...

关闭 >

+-- 关闭

/ tcp-close tcp-close <--+

在交换了[关闭](#_bookmark83)执行符后，实现不应期望能够重新使用打开的 TCP 套接字。不要求实现在收到[关闭](#_bookmark83)指令后从套接字中读取数据。

## 同时关闭

通常情况下，一个对等端点会启动连接关闭，而另一个对等端点会发送关闭作为回应。不过，由于两个端点可能出于各自独立的原因同时选择关闭连接，因此有可能出现同时关闭的情况。在这种情况下，从每个端点的角度来看，唯一可观察到的潜在差异就是表示关闭原因的代码。

TCP 客户端TCP 服务器

================================

...

关闭 ---+ + 关闭

\ / x

/ \

tcp-close <--++--> tcp-close

## 连接闲置时间

连接受空闲超时阈值的限制。当超过阈值后没有收到任何帧时，本地对等设备就会触发超时。空闲超时以毫秒为单位，从接收到最后一个帧开始计算。如果超过阈值，对等端应尝试使用[关闭](#_bookmark83)帧优雅地关闭连接，并说明错误原因。如果远程对等端没有在阈值内对此作出优雅的响应，那么对等端就可以关闭 TCP 套接字。

每个对等节点都有自己的（独立的）空闲超时时间。在连接打开时，每个对等端都会告知它希望从其伙伴处获得的连接活动（帧）的最长间隔时间。为此，[打开](#_bookmark75)帧会携带空闲超时字段。为避免出现虚假超时，空闲超时值应为对等方实际超时阈值的一半。

如果对等方因某种原因无法支持建议的空闲超时，则应使用[关闭](#_bookmark83)帧关闭连接，并说明错误原因。不要求对等设备支持任意长短的空闲超时。

使用空闲超时是对任何网络协议级控制的补充。实施过程中应尽可能使用 TCP keep-alive，做个好公民。

如果对等设备需要发送流量以防止空闲超时，但又没有东西可发送，则可发送空帧，即仅有帧头而无帧体的帧。该帧的信道可以是信道最大值以内的任何有效信道，否则将被忽略。实现者应使用通道 0 来处理空帧，如果尚未协商通道最大值（即在接收到[开放](#_bookmark75)帧之前），则必须使用通道 0。除此以外，空帧没有任何意义。

空帧只能在[打开](#_bookmark75)帧发送后发送。由于空帧是一个帧，因此不应在[关闭](#_bookmark83)帧发送后再发送。

作为使用空帧防止空闲超时的替代方法，如果连接处于允许状态，实施可以选择为有效会话发送流量帧。

如果对等端在运行过程中超过了远程对等端的空闲超时阈值（例如，由于负载过重），对等端应通过使用带有错误说明的[关闭](#_bookmark83)帧来优雅地关闭连接。

## 连接国家

**开始** 在这种状态下，存在一个连接，但没有发送或接收任何信息。这是执行套接字连接或套接字接受后的状态。

**HDR RCVD** 在这种状态下，连接头已从对等方收到，但我们尚未发送任何内容。

**已发送 HDR** 在此状态下，连接头已发送给对等方，但我们尚未收到任何内容。

**开放管道** 在这种状态下，我们已经发送了连接头和[开放](#_bookmark75)帧，但还没有收到任何信息。

**OC 管道** 在这种状态下，我们已经发送了连接头、[打开](#_bookmark75)帧、任何流水线连接流量和[关闭](#_bookmark83)帧，但还没有收到任何信息。

**OPEN RCVD** 在此状态下，我们发送和接收了连接头，并收到了一个

但尚未发送[开放](#_bookmark75)帧。

**已发送** 在这种状态下，我们发送并接收了连接头，并发送了[打开](#_bookmark75)

但尚未收到[开放式](#_bookmark75)框架。

**关闭管道** 在这种状态下，我们已经发送并接收了连接头、发送了[开放](#_bookmark75)帧、任何管道连接流量和[关闭](#_bookmark83)帧，但还没有接收到[开放](#_bookmark75)帧。

**已打开** 在这种状态下，连接头和[开放](#_bookmark75)帧都已发送和接收。

**CLOSE RCVD** 在这种状态下，我们收到了一个[关闭](#_bookmark83)帧，表明我们的伙伴已启动关闭。这意味着我们再也不必从该连接中读取任何内容，但我们可以继续向该连接写入帧。如果需要，可以在此时执行 TCP 半关闭，关闭连接的读取端。

**CLOSE SENT（关闭已发送）** 在此状态下，我们已向对方发送了一个[关闭](#_bookmark83)帧。再向连接写入任何内容都是非法的，但仍可能有帧传入。如果需要，可以在此时执行 TCP 半关闭，关闭连接的写入端。

DISCARDING DISCARDING 状态是由错误触发[关闭](#_bookmark83)的 CLOSE SENT 状态的一种变体。在这种情况下，连接上的任何传入帧都必须被静默丢弃，直到收到对等方的[关闭](#_bookmark83)帧。

**结束** 在这种状态下，任一端点都不能再向连接写入任何内容。连接可以安全关闭并丢弃。

## 连接状态图

下图描述了每个端点的完整状态图。方框代表状态，箭头代表状态转换。每个箭头都标有触发该特定转换的操作。

R:HDR @=======@ S:HDR R:HDR[!=S:HDR]

+--------| START |-----+ + +

| @=======@ | | |

\|/ \|/ | |

|  |  |  |
| --- | --- | --- |
| @==========@ | @==========@ S:OPEN | | |
| +----| HDR\_RCVD | | | HDR\_SENT | + | | |
| | @==========@ | @==========@ | | r:hdr[!=s:hdr] | r:hdr[! |
| |S :HDR | | R:HDR | | + + |
| | + + | +------+ | | | | |
| | \|/ | \|/ \|/ | | | |

| @==========@ +-----------+ S:CLOSE |

|| hdr\_exch || open\_pipe |----+ |

| @==========@ +-----------+ | |

|r :open ||s: open|r:hdr | |

| +--------+ +------+ +-------+ | |

| \|/ \|/ \|/ \|/ |

| @===========@ @===========@ S: CLOSE+---------+ | |

|| open\_rcvd || open\_sent |-----+| oc\_pipe |--+

| @===========@ @===========@ |+---------+ |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | S:OPEN |  | | | R:OPEN  @========@ | | \|/  + + | | R:HDR  | | |  | |
| | | +------>| opened |<----+| close\_pipe |<--+ | | | | | |
| | | @========@ + + | | | | | |
| | | r:close || s: close| r:open | | | | | |
| | | +---------+ +-------+ | | | | | | |
| | | \|/ \|/ | | | | | | |
| | | @============@ @=============@ | | | | | | |
| | | | close\_rcvd || close\_sent\* |< + | | | | | |
| | | @============@ @=============@ | | | | | |
| | | s:close || r:close | | | | | |
| | | | @=====@ | | | | | | |
| | | +-------->| END |< + | | | | | |
| | | @=====@ | | | | | |

| /|\ |

| S:HDR[！=R:HDR］ | R:HDR[！=S:HDR］ |

+----------------------+ +

R:<CTRL> = 接收 <CTRL> S:<CTRL> = 发送 <CTRL>

\* 如果错误条件触发了 CLOSE，也可能是 DISCARDING。

状态法律 发送法律 接收法律连接行动

=======================================================================

TCP 关闭写入 TCP 关闭写入

|  |  |  |
| --- | --- | --- |
| 开始 | HDR | HDR |
| HDR\_RCVD | HDR | 开放 |
| HDR\_SENT | 开放 | HDR |
| HDR\_EXCH | 开放 | 开放 |
| OPEN\_RCVD | 开放 | \* |
| OPEN\_SENT | \*\* | 开放 |
| OPEN\_PIPE | \*\* | HDR |
| 关闭管道 | - | 开放 |
| OC\_PIPE | - | HDR |
| 已开启 | \* | \* |
| CLOSE\_RCVD | \* | - |
| 关闭发送 | - | \* |
| 废弃 | - | \* |
| 结束 | - | - |

读 TCP 关闭 写 TCP 关闭 写 TCP 关闭

\* = 任何帧

- = 无框架

\*\* = 事先已知符合对等设备能力和限制的任何帧

* 1. **会议**

会话是两个容器之间的双向顺序对话，为相关链接提供分组。会话是链接通信的上下文。一个给定的会话可以*连接*任意数量、任意方向的链接。不过，一个链接一次最多只能连接到一个会话。

链接 A-------+ +------> 链接 A

| |

\|/ (attached) |

链接 B<--- 会话 <--------------> 会话 <--- 链接 B

链接 C------>\* (脱离) \*------> 链接 C

链路上传输的信息在会话中按顺序标识。会话可被视为链路流量的多路复用，就像连接对会话流量的多路复用一样。不过，与连接上的会话不同，会话上的链路并非完全独立，因为它们共享一个适用于会话的共同传输序列。这种共同序列允许端点有效地引用交付集，而不考虑源链接。当一个应用程序通过大量不同的链接接收信息时，这一点尤为重要。在这种情况下，会话可将原本独立的链路*聚合*为一个流，接收应用程序可有效地确认该流。

## 建立会议

建立会话的方法是创建一个会话端点，将其分配给一个未使用的信道编号，然后发送一个 [begin](#_bookmark76)，宣布会话端点与传出信道的关联。合作伙伴收到 [begin](#_bookmark76) 后，会检查远程信道字段，发现该字段为空。这表明 begin 指的是远程发起的会话。因此，伙伴将为远程发起的会话分配一个未使用的传出信道，并通过发送自己的 [begin](#_bookmark76) 将远程信道字段设置为远程发起的会话的传入信道来表明这一点。

为了更容易监控 AMQP 会话，建议实施时始终分配可用的最低未用通道号。

对于本地发起的会话，[begin](#_bookmark76) 帧的远程信道字段必须为空，而在宣布远程发起的会话所创建的端点时，必须设置远程信道字段。

端点 终点

===================================================================== [CH3] BEGIN(name=... ,--------->

remote-channel=null)

<---+

+-- [CH7] BEGIN（name=.....、

/ remote-channel=3)

/

...

## 结束会议

会话在连接关闭或中断时自动结束。会话在任一端点选择结束会话时明确结束。当会话明确结束时，会发送一个[结束](#_bookmark82)帧，宣布端点与其传出信道解除关联，并在相关情况下携带错误信息。

端点 A 端点 B

====================================================================

...

[CH3] 结束（错误=...） ---------> (1)

+-- [CH7] 结束（错误=...）

/

/

(2) <---+

...

* + - 1. 此时，会话端点将与 A 的传出通道和 B 的传入通道分离。
      2. 此时，会话端点与 B 的传出信道和 A 的传入信道分离。

## 同时结束

由于会话可能是异步的，因此两个对等方有可能同时决定结束会话。如果出现这种情况，在每个对等方看来，其伙伴自发启动的[结束](#_bookmark82)帧实际上是对等方初始[结束](#_bookmark82)帧的应答。

端点 A 端点 B

=================================================================

...

[CH3] END（错误=...） --++-- [CH7] END（错误=...）

(1)\ / (2)

x

/ \

(3) <-+ +-> (4)

...

1. 此时，A 不再发送任何帧。
2. 此时，B 将不再发送任何帧。
3. 至此，A 端点完全结束。
4. 至此，B 端点完全结束。

## 会议错误

当会话无法处理输入时，它必须发出带有适当[错误信息](#_bookmark100)的结束帧（END）来说明问题的原因。然后，会话必须丢弃所有从远程端点传入的帧，直到听到远程端点相应的[结束](#_bookmark82)帧。

端点 终点

================================================ FRAME 1---------->

框架 2 >

帧 3 ---++--- 结束（错误=. )

\ / x

/ \

<--++--> \*弃权\*

结束 >

...

================================================

## 届会国家

**UNMAPPED** 在 UNMAPPED 状态下，会话端点不会映射到连接端点上的任何传入或传出通道。在这种状态下，端点不能发送或接收帧。

**BEGIN SENT（开始发送**） 在 "开始发送 "状态下，会话端点被分配了一个传出信道编号，但在传入信道映射中没有条目。在这种状态下，端点可以发送帧，但不能接收帧。

**BEGIN RCVD** 在 BEGIN RCVD 状态下，会话端点在传入信道映射中有一个条目，但尚未分配一个传出信道编号。端点可以接收帧，但不能发送帧。

**MAPPED** 在 MAPPED 状态下，会话端点既有传出信道编号，也有传入信道映射中的条目。端点既可以发送也可以接收帧。

**结束发送** 在 END SENT 状态下，会话端点在传入信道映射中有一个条目，但不再分配传出信道编号。端点可以接收帧，但不能发送帧。

**RCVD 结束** 在 END RCVD 状态下，会话端点被分配了一个传出信道号，但传入信道映射中没有条目。端点可以发送帧，但不能接收帧。

丢弃丢弃（DISCARDING**）**状态是[结束](#_bookmark82)发送（END SENT）状态的一种变体，[结束](#_bookmark82)是由错误触发的。在这种情况下，会话中的任何传入帧都必须被静默丢弃，直到收到对等方的[结束](#_bookmark82)帧。

解压缩< +

| |

+-------+-------+ | s:begin || r:begin |

|  |  |  |
| --- | --- | --- |
| | | | | | |
| \|/ | \|/ | | |
| 开始发送 | BEGIN\_RCVD | | |
| | | | | | |
| | | | | | |
| :begin || s:begin |  +-------+-------+ | | | |
| | | | | |
| \|/ | | | |
| 地图 | | | |
| | | | | |
| +-------------+ + | | | |

R

S:END(error) |S :END ||R:END |

|  |  |  |  |
| --- | --- | --- | --- |
| | | | | | | | |
| \|/ | \|/ | \|/ | | |
| 废弃 | 结束\_发送 | END\_RCVD | | |
| | | | | | | | |
| | | | | | | | |

r:end |r :end ||s:end |

+-------------+-------------+ |

| |

| |

+ +

图 2.4：状态转换

当会话端点处于 UNMAPPED 状态时，没有义务保留该会话端点，即 UNMAPPED 状态等同于不存在状态。

## 会话流量控制

会话端点从会话作用域序列中为每个传出[传输](#_bookmark79)帧分配一个隐式*传输标识*。每个会话端点维护以下状态，以管理传入和传出的[传输](#_bookmark79)帧：

*next-incoming-id 下*一个接收的隐式传输标识。

[转接](#_bookmark79)架。

**来电窗口**

*incoming-window* 定义了端点当前可接收的最大[传入传输](#_bookmark79)帧数。它确定了当前最大传入传输 ID，该 ID 可通过从 *incoming-window* 和 *next-incoming-id 的*总和中减去 1 计算得出。

**next-outgoing-id** *next-outgoing-id* 用于为给定会话中所有传出的传输帧分配一个唯一的传输标识。*next-outgoing-id* 可以初始化为任意值，并在每次连续[传输](#_bookmark79)后根据 RFC-1982 序号运算递增。

**outgoing-window** *输出窗口*定义端点当前可发送的最大输出[传输](#_bookmark79)帧数。这标识了当前最大传出

的总和中减去一个，就可以计算出正在进行的转移标识。

*outgoing-window* 和 *next-outgoing-id*。

**远程来电窗口**

*远程传入窗口（remote-incoming-window*）反映了在不超过远程端点传入窗口的情况下可发送的最大传出传输数。该值必须在每个[传输](#_bookmark79)帧发送后递减，并在获知远程会话端点状态后重新计算。

**远程输出窗口**

*远程传出窗口（remote-*outgoing-*window*）反映了在不超过远程端点传出窗口的情况下可能到达的传入传输帧的最大数量。该值必须在每次收到传入[传输](#_bookmark79)帧后递减，并在获悉远程会话端点状态后重新计算。当该窗口缩小时，表明有未完成的传输。解决未完成的传输可能会导致窗口增大。

初始化后，该状态会根据会话及其相关链接生命周期中发生的各种事件进行更新：

**发送转账**

发送传输时，发送端点将递增其下一个发送 ID，递减其远程接收窗口，并可能（根据策略）递减其发送窗口。

**接转**

接收传输时，接收端点将递增下一个传入 ID，使其与传入传输的隐式传输 ID 加一相匹配，同时递减远程传出窗口，并可能（根据策略）递减其传入窗口。

**接收流量** 当端点从其对等设备接收到[流量](#_bookmark78)帧时，必须直接从帧的 *next-outgoing-id* 更新 *next-incoming-id，并*直接从帧的 *outgoing-window* 复制 *remote-outgoing-window 。*

*远程进入窗口的*计算方法如下：

*下一个入站标识流* + *入站窗口流* - *下一个出站标识端点*

如果流帧的 *next-incoming-id* 字段未设置，则*远程入站窗口的*计算方法如下：

*初始出站标识符* + *入站窗口流* - *下一个出站标识符*

* 1. **链接**

链接为信息源和目标之间的信息提供单向传输。信源或目标（终点站）的主要职责是保存每次活动传送尝试的状态记录，直到可以安全遗忘为止。这些被称为*未结算*交付。当终点站遗忘了与某个递送标记相关的状态时，该递送即被视为*已解决*。每个递送尝试都会在源端分配一个唯一的*递送标记*。活动递送尝试的状态称为递送的递送*状态。*

链接端点是终端和会话端点之间的接口，并维护用于本地端点和远程端点之间主动通信的附加状态。因此，链接端点有两种类型：*发送端点*和*接收端点*。当发送应用程序向发送方提交信息以进行传输时，它还会提供源用来跟踪交付状态的交付标记。

链接端点从会话范围序列中为每个报文分配一个唯一的*交付标识*。这些交付标识用于有效引用会话中未完成交付的子集。

终端可能存在于其关联的链路端点之外，因此可能出现会话终止而终端仍然存在的情况。如果终端节点存在，但没有关联的链路端点，则称链路*暂停。*将新的链接端点与现有端点关联并重新建立通信的过程称为*恢复*链接。

恢复链路不需要原始链路端点状态。恢复链路时只需要在终端站维护未结算的传送状态，而且无需直接存储。交付标记的形式有意保持开放，以便在需要时可以根据应用状态（重新）构建标记及其相关的交付状态，从而最大限度地减少或消除为恢复链路而保留额外协议特定状态的需要。

## 命名链接

对链接进行命名是为了在通信中断时恢复链接。链接名称必须在两个参与容器之间的所有同方向链接中唯一标识该链接。链路名称只在连接链路时使用，因此可任意加长，而不会造成严重影响。

链接的名称唯一标识了从源节点容器到目标节点容器的链接，例如，如果源节点的容器是 A，目标节点的容器是 B，那么链接可以用（有序）元组 *(A,B,<name>)* 全局标识。

因此，链接一次只能在一个连接中处于活动状态。如果在链路未暂停的情况下尝试连接该链路，那么该链路就会被 "窃取"，即第二次连接成功后，第一次连接必须关闭，并显示链路错误为 "[被窃取](#_bookmark107)"。这种行为可确保在发生连接失败并被一方注意到的情况下，重新建立连接能达到预期效果。

## 链接手柄

每个链路端点都会被分配一个数字句柄，由对等设备在所有引用该链路的帧中（[附加](#_bookmark77)、[分离](#_bookmark81)、[流动](#_bookmark78)、[传输](#_bookmark79)、[处置](#_bookmark80)）用作引用该链路的速记符号。该句柄由初始[附加](#_bookmark77)帧分配，并一直使用到链路分离为止。两个端点无需使用相同的句柄。这意味着，当链路端点与会话关联时，对等端点可以自由选择其句柄。本地选择的句柄称为*输出句柄*。远程选择的句柄称为*输入句柄*。

在一个端点上，当链接端点存在并在活动会话端点上分配了输入和输出句柄时，链接被视为已*连接*。当链接端点存在，但未分配输入或输出句柄时，链接被视为*分离。当*只分配了一个输入或输出句柄时，链接可视为*半连接*（或*半分离*）。

+-------------------+ + +

|name : Link\_1 | |name : Link\_1 |

| 句柄: i | 句柄： j |

|-------------------| | |

|角色 ： 接收者 |角色 ：发送者 |

| 来源：A |<--++--->| 源代码：A |

| 目标：B | | || 目标： B |

+-------------------+ | | + +

| |

| +---------+ |

... <---+--->| Session |<---+---> ...

| +---------+ |

| |

+-------------------+ | | + +

|名称 ：Link\_N | | | |名称 : Link\_N |

| 句柄： k |<--++--->| 句柄： l |

|-------------------| | |

|角色 ：发件人 | |角色 ： 接收者

| 来源：C | 源：C |

| 目标：D | target：D |

+-------------------+ + +

## 建立或恢复链接

建立和/或恢复链接的方法是创建一个与本地终端相关联的链接端点，将其分配给一个未使用的句柄，并发送一个[附加](#_bookmark77)帧。该帧包含新创建链接端点的状态，包括本地和远程终端，一个是源端，一个是目标端，这取决于链接端点的方向性。收到[连接](#_bookmark77)后，远程会话端点会创建一个相应的链路端点，并将连接链路通知其应用程序。应用程序会尝试查找先前与链路关联的终点站。该终端与链接端点相关联，如果其属性与远程链接端点发送的属性不匹配，则可能会被更新。如果不存在这样的终端，应用程序可以选择使用远程链接端点提供的属性创建一个终端。然后，链接端点会被映射到一个未使用的句柄，并发出一个[附着](#_bookmark77)帧（[attach](#_bookmark77) Frame），其中包含新创建端点的状态。需要注意的是，如果应用程序选择不创建终端，会话端点仍将创建一个链接端点，并发出一个[附件](#_bookmark77)，表明该链接端点没有关联的本地终端。在这种情况下，会话端点必须立即分离新创建的链接端点。

同行 合作伙伴

================================================================

\*创建链接端点\*

ATTACH(name=N, handle=1, ----------> \*创建链接端点\*role=sender, +--- ATTACH(name=N, handle=2、

source=A、 / 角色=接收者、

target=B) / source=A、

/ target=B)

<--+

...

图 2.5：建立链接

如果没有预先存在的终点站，而对等方又不想创建新的终点站，则可将本地终点站（视情况为源或目标）设置为空。

同行 合作伙伴

================================================================

\*创建链接端点\*

ATTACH(name=N, handle=1, ---------->\*创建链接端点\* (1) role=sender, +--- ATTACH(name=N, handle=2、

source=A、 / 角色=接收者、

target=B) / source=A、

/ target=-)

(2) <--+

<--+

+--- DETACH(handle=2、

/ closed=True)

/

/

DETACH(handle=1、 >

closed=True)

...

* + - 1. 链接端点已创建，但未创建目标。
      2. 此时，链接已经建立，但链接的目标并不存在。

图 2.6：拒绝链接

如果链路的任一端已与一个终点站关联，则[附加](#_bookmark77)帧必须包括其未结算的传送状态。

同行 合作伙伴

================================================================

\*现有来源\*

ATTACH(name=N, handle=1, ----------> \*发现现有目标\*role=sender, +--- ATTACH(name=N, handle=2, (1)

source=X、 / 角色=接收者、

target=Y、 / source=X、

不安=......) / target=Y、

(2) <--+ 不安=......)

...

1. 目标已经存在，其属性符合同行的预期。
2. 此时重新建立链接，源=X，目标=Y。

图 2.7：恢复链接

请注意，预期的终点属性不一定总是与远程端点报告的实际终点属性一致。在这种情况下，链接总是被认为是在发送方描述的 "源 "和接收方描述的 "目标 "之间。这种情况在建立和恢复链接时都可能发生。

建立链接时，端点可能不具备创建终端所需的所有功能，无法完全满足对等方的期望。如果出现这种情况，端点可以调整属性，以便成功创建终端。在这种情况下，端点必须报告所创建的终端的实际属性。

恢复链接时，源和目标属性可能在链接暂停时发生了变化。在这种情况下，[附件](#_bookmark77)帧源字段和目标字段中通信的 Termini 属性可能会发生冲突。在这种情况下，发送方被认为持有源属性的权威版本，接收方被认为持有目标属性的权威版本。如上所述，由此产生的链路是在发送方描述的 "源 "和接收方描述的 "目标 "之间构建的。一旦链接恢复，如果更新后的属性可以接受，任一对等方都可以继续使用，如果不能接受，则[分离](#_bookmark81)。

请注意，对等方必须负责验证远程终端是否满足其要求。 远程对等方不应试图预先判断总站是否符合其要求。

其合作伙伴的要求。这对创建和恢复链接同样适用。

同行 合作伙伴

================================================================

\*现有来源\*

ATTACH(name=N, handle=1, ----------> \*发现现有目标\*role=sender, +--- ATTACH(name=N, handle=2, (1)

source=A、 / 角色=接收者、

target=B、 / source=A、

不安=......) / target=C、

(2) <--+ 不安=......)

...

1. 终点站已经存在，但其状态与 Peer 的终点不匹配。
2. 此时，链接已建立，源=A，目标=C。

图 2.8：恢复已更改的链接

即使其中一个终点站几乎失去了所有状态，也可以恢复链路。所需要的只是链路名称和方向。这就是所谓的*恢复*链接。具体做法是创建一个新的链接端点，其输入或输出链接的源或目标分别为空。一旦链接建立，就会根据另一个端点提供的权威源或目标构建完整的链接状态。如果远端对等端点没有链接记录，那么将无法找到任何终端，[附件](#_bookmark77)帧中的本地终端（源或目标，视情况而定）字段将为空。

同行 合作伙伴

================================================================

\*创建链接端点\*

ATTACH(name=N, handle=1, ----------> \*发现现有目标\*role=sender, +--- ATTACH(name=N, handle=2, (1)

来源=X / 角色=接收者、

target=-) / source=X、

(2) <---+ target=Y)

...

1. 目标已存在，其属性具有权威性。
2. 此时重新建立链接，源=X，目标=Y。

图 2.9：恢复链接

## 拆卸和重新连接链接

会话端点可选择取消映射链接的输出句柄。在这种情况下，端点必须发送一个[分离](#_bookmark81)帧，通知远程对等端点该句柄已不再连接到链接端点。如果两个端点都这样做，链路可能会返回到完全分离状态。请注意，在这种情况下，链路端点仍可通过会话进行间接通信，因为通过 delivery-id 引用的链路上可能有活动交付。

同行 合作伙伴

=============================================================

\*创建链接端点\*

ATTACH(name=N, handle=1 ----------> \*创建链接端点\*role=sender, +--- ATTACH(name=N, handle=2、

source=A、 / 角色=接收者、

target=B) / source=A、

/ target=B)

<--+

...

\*使用链接\* <---------> \*使用链接\*

...

DETACH(handle=1 )----------> \*detach输入句柄\*

1. \*卸下输出手柄\* < DETACH(handle=2)

...

* 1. 此时，两个端点都已分离。

当链路端点的状态发生变化时，可通过分离链路端点，然后在[连接](#_bookmark77)帧上用更新的状态重新连接的方式进行通信。这可用于更新链路端点的属性或更新终端的属性。

同行 合作伙伴

=============================================================

...

DETACH(handle=1) +

\

\

\

\*修改链接端点\* \

+--> \*取下输入句柄\* ATTACH(name=N, handle=1---+) + DETACH(handle=2)

role=sender ,\ /

source=A'、 \/

target=B') /\

/ \

\*删除输入句柄\* <--++--> \*断开输入句柄 \*

\*修改链接端点\*

+--- ATTACH(name=N, handle=2

/ role=receiver、

/ source=A'、

/ target=B')

/

1. \* 重新启动输入手柄\* <--+

...

\*使用链接\* <---------> \*使用链接\*

...

* 1. 此时，链接已更新并附上。

## 链接错误

当链路端点发生错误时，必须分离该端点，并在[分离](#_bookmark81)帧的错误字段中提供适当的错误信息。然后必须销毁链路端点。如果通过输入句柄或交付标识接收到与端点相关的任何输入（非分离），会话必须以 [Errant-link](#_bookmark105) 会话错误信息终止。由于链接端点已被破坏，对等方无法重新连接，必须恢复链接才能恢复通信。为了区分恢复请求和管道式重新连接，恢复[连接](#_bookmark77)执行符必须在其未结算字段中包含一个非空值。收到流水线[连接](#_bookmark77)后，会话必须以 "[勘误链路](#_bookmark105)会话错误"（[errant-link](#_bookmark105) session-error）终止。

## 关闭链接

对等节点关闭链路的方法是发送带有指定链路句柄的[分离](#_bookmark81)帧，并将关闭标记设为 true。伙伴将销毁相应的链接端点，并回复自己的[分离](#_bookmark81)帧，同时将关闭标记设为 true。

同行 合作伙伴

=============================================================

\*创建链接端点\*

ATTACH(name=N, handle=1 ----------> \*创建链接端点\*role=sender, +--- ATTACH(name=N, handle=2、

source=A、 / 角色=接收者、

target=B) / source=A、

/ target=B)

<--+

...

\*使用链接\* <---------> \*使用链接\*

...

DETACH(handle=1 ,----------> \*销毁链接端点\* closed=True)

1. \*销毁链接端点\* < DETACH(handle=2、

closed=True)

* 1. 此时，两个端点都会被销毁。

图 2.10：关闭链接

请注意，一个对等节点可能会在其伙伴发送非关闭分离时发送关闭分离。在这种情况下，对等方必须通过重新连接然后发送关闭脱离（closing detach）来发出它已关闭链路的信号。

## 流量控制

链接一旦连接，就必须对报文传输进行流量控制。链接端点会保持以下流量控制状态。该状态定义了在已连接的链接上发送传输何时是合法的，并指示何时会出现某些值得注意的情况，如报文不足以消耗当前可用的*链接点数，*或*链接点数*不足以发送可用报文：

**delivery-count 发送**方在创建链接端点时初始化交付计数，并在发送（发送方）或接收（接收方）信息时递增。只有发送方可以独立修改该字段。接收方的值是根据发送方的最后已知值和链接上收到的任何后续信息计算得出的。

**链接信用** *link-credit* 变量定义了当前可增加*传送次数*的最大法定金额。将 *link-credit* 与*传送次数*相加可计算出一个传送*限制*。

只有接收方可以独立选择该字段的值。发送方的值必须始终保持与接收方确定的*交付限制*相匹配。这意味着，当接收方提供流量信息时，发送方的链路信用变量必须根据此公式进行设置：

link-creditsnd := *交货-计数*rcv + link-creditrcv - 交货*-计数nd*。

如果接收方还不知道发送*次数*，即*发送次数* rcv 未指定，则发送方必须假定发送次数 rcv 已被指定。

是发送端向接收端发送的第一个发送计数，即发送端向接收端发送的初始[附加](#_bookmark77)帧所携带的流状态中指定的*发送*计数。

此外，每当发送方增加*传送次数*时，它必须减少相同数量的*链接信用，以*维持接收方确定的*传送限制*。

**可用** *可用*变量由发送方控制，向接收方表明发送方可以使用指定的*链接点*数。只有发送方可以独立修改该字段。接收方的值是根据发送方的最后已知值和随后收到的任何传入信息计算得出的。即使可用变量为零，发送方也可以传输信息。如果出现这种情况，接收方在计算可用值时必须保持零下限。

**耗尽** 耗尽标志表示当可用信息不足以消耗当前链路信用时，发送方应如何处理。如果设置了该标志，发送方将（在发送完所有可用报文后）尽可能提前发送计数，消耗所有链路信用，并将流状态发送给接收方。只有接收方可以独立修改该字段。发送方的值始终是接收方显示的最后一个已知值。

如果链路信用（link-credit）小于或等于零，即传送次数等于或大于传送限制（delivery-limit），则发送更多信息是非法的。如果接收方在传输过程中减少了链路信用，接收方可以正常处理多余的信息，或使用传输限制超限错误代码分离链路。

+----------+ + +

| 发送方 |---------------transfer------------>| 接收方 |---------------transfer------------>.

+----------+ + +

\ / <----------------flow--------------- \ /

+------+ + +

|

|

|

如果 link-credit <= 0 则暂停

如果发送方的耗尽标志被设置，并且没有可用的报文，发送方必须将其发送计数（delivery-count）提前，直到链路信用（link-credit）为零，并将其更新的[流](#_bookmark78)状态发送给接收方。

传送次数是一个绝对值。虽然该值本身在概念上是无限制的，但它被编码为一个 32 位整数，并根据 RFC-1982 序列号算术进行环绕和比较。

链路端点的初始流量状态确定如下。*链路信用*和*可用*变量初始化为零。*耗尽*标志初始化为 False。发送方可选择一个任意点来初始化*传送计数*。该值在初始[附加](#_bookmark77)帧中传达。接收方在收到发送方的[附件](#_bookmark77)后初始化其*传送次数*。

流动状态

|

| 修改

+------------------+ | + +

| 发件人 | .----------------------. | 收件人 |

+------------------+ 附加 、转移、流动 + +

| 交付数量 |------------------------------->| delivery-count |

| 链接-荣誉 | link-credit |

| 可用 |<-------------------------------| available |

| 排水 | 流量 |

+------------------+ '-----' + +

|

| 修改

| 流动状态

本节定义的流量控制语义提供了实现各种流量控制策略所需的基元。此外，通过操作链路信用和耗尽标志，接收器可以提供各种对应用程序非常有用的高级行为，包括同步阻塞获取、带超时的同步获取、异步通知和停止/暂停。

+----------+ + +

| 接收方 |<--------------transfer-------------| 发送方 |<--------------transfer-------------| 接收方

+----------+ + +

\ / -----------------flow--------------> \ /

+------+ + +

|

|

|

sync-get: flow(link-credit=1, ...) >

timed-get: flow(link-credit=1、 ),

\*等等

flow(drain=True, ...) >

async-notify: flow(link-credit=delta, ...) >

stop: flow(link-credit=0, ...) >

## 同步获取

从链路同步获取报文的方法是递增链路信用，发送更新的[流](#_bookmark78)状态，然后无限期地等待[传输](#_bookmark79)到达。

接收器 发送器

=================================================================

...

flow(link-credit=1) >

+---- transfer(. )

\*阻塞，直到传输到达\* /

<---+

...

超时同步获取是通过递增链路信用、发送更新的[流](#_bookmark78)状态并等待链路信用消耗来实现的。当所需时间过去后，接收器会设置耗尽标志，并再次发送新更新的[流](#_bookmark78)状态，同时继续等待链路信用消耗完。即使没有信息可用，也会因为耗尽标志而迅速满足这一条件。一旦链路信用消耗完毕，接收方就能明确判断是否有信息到达或操作是否超时。

接收器 发送器

=================================================================

...

flow(link-credit=1) >

\* 等待链接积分 <= 0\*

flow(drain=True) ---++--- transfer(. )

\ / x

/ \

(1) <--+ +-->

(2 )<---------- flow(. )

...

1. 如果在超时时间内有信息，则会在此时到达。
2. 如果在超时时间内没有信息，则

耗尽标志将确保发件人及时推进发送次数，直到链接积分耗尽为止。

## 异步通知

异步通知可通过以下方式实现。接收方为该链路保留一个目标链路信用额度。当传输到达链路时，发送方的链路信用会随着传送次数的增加而减少。当发送方的链路信用低于临界值时，可发送[流量](#_bookmark78)状态，将发送方的链路信用提高到所需的目标值。

接收器 发送器

=====================================================================

...

<---------- transfer(. )

<---------- transfer(. )

flow(link-credit=delta) ---+ +--- transfer(. )

\ / x

/ \

<--+ +-->

<---------- transfer(. )

<---------- transfer(. )

flow(link-credit=delta) ---+ +--- transfer(. )

\ / x

/ \

<--+ +-->

...

链路的传入信息速率受限于接收方通过发放链路积分更新交付限制的速率。

## 停止链接

通过将链路信用值更新为零并发送更新的[流量](#_bookmark78)状态，就可以停止给定链路上的传输。发送[流](#_bookmark78)状态时，有些传输可能还在飞行中，因此传入的传输可能仍会到达链路。流帧的回声字段可用于请求发送方的[流](#_bookmark78)状态回声。这可用于确定链路何时最终静止。

接收器 发送器

================================================================

...

<---------- transfer(...)

flow(..、 ---++--- transfer(...) link-credit=0,\ /

echo=True) x

/ \

(1) <--+ +-->

(2 )<---------- 流量(...)

...

1. 在发送方更新流量状态之前，飞行中的传输仍可能到达。
2. 至此，将不会再有转运人员抵达。

## 信息

传输层对信息的假设尽可能少，并允许在其上分层显示其他信息。信息数据作为有效载荷在包含[传输](#_bookmark79)执行符的帧中传输。信息可在多个[传输](#_bookmark79)帧中分片，[传输](#_bookmark79)执行符中的 "更多 "标志表示了这一点。

## 传输信息

应用程序启动信息传输时，会分配一个交付标记，用于跟踪信息传输过程中的交付状态。从发送/接收点开始，直至发送/接收应用程序将其*结算*为止，在发送方/接收方处，交付都被视为*未结算*。每个递送必须由发送应用程序选择的递送标记来标识。在所有可被链路任一端视为未结算的递送中，递送标记必须是唯一的。

启动传输时，应用程序将向发送链路端点（发送端）提供信息数据及其相关的传送标记。发送端将在其未设置映射中创建一个条目，并发送一个包括传送标记、初始状态和相关信息数据的传送帧。为了简化传输过程，传送标记还与会话分配的传送标识相关联。在该会话的所有后续交互中，交付-ID 都会被用来引用交付-标记。

为简单起见，以下图表中省略了传送 ID，直接使用传送标记。这些图表还假定这种交互是在一条已建立的链路上进行的，因此省略了实际中存在于线路上的其他细节，如信道号、链路句柄、分片标志等，只关注信息传输的基本方面。

+ +

/ 发件人 \

+ +

| 不稳定： | transfer(delivery-tag=DT, settled=False、

| ... | state=S\_0、 )

|DT -> (local：S\_0, | >

|remote : ?)|

| ... |

+ +

图 2.11：初始转移

接收传输后，接收链路端点（接收器）将在自己的未结算映射中创建一个条目，并将传输的信息数据提供给应用程序处理。

+ +

/ 接收器 \

+ +

transfer(delivery-tag=DT, settled=False, unsettled: state=S\_0, ...) | ... |

----------------------------------------------->|DT -> (local:S\_1, |

|remote ：S\_0)|

| ... |

+ +

图 2.12：初始接收

一旦收到信息数据通知，应用程序就会处理信息，并根据需要向链路端点显示更新后的传送状态。应用程序可能希望将传送状态分为*终端状态*和*非终端*状态，这取决于端点在到达后是否会进一步更新状态。在某些情况下（如大型信息或事务），接收应用程序可能希望向发送方指明非终端传送状态。这可以通过[处置](#_bookmark80)帧来实现。

+ +

/ 接收器 \

+ +

| 不稳定： |

| ... |

<-----------------------------------------------|DT -> (local:S\_2, |

disp(role=receiver, ..., delivery-tag=DT, |remote : S\_0)| settled=False, state=S\_2, ...) | ... |

+ +

图 2.13：非终端状态指示

一旦接收应用程序完成了信息处理，它就会向链路端点指示一个*终端*交付状态，该状态反映了应用程序的处理结果（成功与否），因此也是接收方希望在发送方出现的结果。这一状态将通过处置帧传回发送端。

+ +

/ 接收器 \

+ +

| 不稳定： |

| ... |

<-----------------------------------------------|DT -> (local:T\_0, |

disp(role=receiver, ..., delivery-tag=DT, |remote : S\_0)| settled=False, state=T\_0, ...) | ... |

+ +

图 2.14：推定终点状态指示

发送方在收到接收方更新的发送状态后，如果尚未自发达到终端状态，则会更新其状态视图，并将此反馈给发送应用程序。

+ +

/ 发件人 \

+ +

| 不稳定： |

| ... |

| DT -> (local：S\_0, |<-----------------------------------------------

|remote ：T\_0)| disp(role=receiver, ..., delivery-tag=DT、

| ... | settled=False, state=T\_0、 )

+ +

图 2.15：接收终端状态

然后，发送应用程序通常会根据该终端状态执行一些操作，然后结算交付，使发送方从其未结算地图中删除交付标记。然后，发件人将发送其最终的交付状态，并表明该交付已在发件人处结算。需要注意的是，这相当于发件人宣布它将永远忘记有关投递标记的一切，因此，这种宣布只能进行一次，因为在发件人忘记之后，它就不可能再想起来进行宣布了。如果该帧因通信中断而丢失，接收方将在链路恢复后发现发送方已完成了投递。当发送方重新连接后，接收方将检查发送方的未结算状态（即未被遗忘的内容），并由此推断出有关的发送已经结算（因为其标签将不处于未结算状态）。

+ +

/ 发件人 \

+ +

| 不稳定： | disp(role=sender, ..., delivery-tag=DT、

| ... | settled=True, state=T\_1、 )

|- -> - | >

| ... |

+ +

图 2.16：结算指示

当接收方发现发送方已完成交付时，接收方将更新其远程状态视图以表明这一点，然后通知接收应用程序。

+ +

/ 接收器 \

+ +

disp(role=sender, ..., delivery-tag=DT, unsettled: settled=True, state=T\_1, ...) | ... |

----------------------------------------------->|DT -> (local:S\_2, |

|remote : - )|

| ... |

+ +

图 2.17：结算收据

然后，应用程序可能会执行一些最终操作，例如从为去重而保留的集合中移除交付标记，然后通知接收方交付已结算。然后，接收方将从其未结算映射中移除交付标记。需要注意的是，由于接收方知道发送方已经结算了递送，因此它不会通知其他端点它正在结算递送。

+ +

/ 接收器 \

+ +

| 不稳定： |

| ... |

<-----------------------------------------------|- -> - |

| ... |

+ +

图 2.18：最终结算

如上所述，发送应用程序有可能在发送端自发地（即不是由于从接收方接收到处置的结果）将交付过渡到终端状态。在这种情况下，发送方应向接收方发送处置，但在接收方通过相反方向的处置确认已更新其终端状态之前，发送方不得进行结算。

这组交换说明了信息传输的基本原理。当传送处于未定状态时，端点交换传送的当前状态。最终，两个端点都会达到应用程序指示的终端状态。这将触发另一个应用程序采取某些最终行动并结算交付，一旦一个端点结算，通常会触发另一个端点的应用程序结算。

这种基本模式可以通过多种方式进行修改，以实现不同的保证，例如，如果发送应用程序*在*发送*前*结算了递送，就会产生 "*最多一次 "*保证。发送方在首次发送时已经表明，他已经忘记了有关此邮件的一切，因此不会再尝试发送。如果该邮件发送到了接收方，接收方显然没有义务对接收方的邮件发送状态进行更新，因为这些更新没有任何意义，会被发送方忽略。

+ +

/ 发件人 \

+ +

| 不稳定： | transfer(delivery-tag=DT, settled=True、

| ... | state=T\_0、 )

|- -> - | >

| ... |

+ +

图 2.19：最多一次

同样，如果我们修改基本方案，使接收应用程序选择在处理报文时立即结算，而不是等待发送方先结算，我们就能得到*至少一次的*保证。如果下面显示的处置帧丢失，那么在链路恢复时，发送方将不会在接收方的未结算映射中看到发送标记，因此会认为发送丢失并重新发送，从而导致接收方重复处理报文。

+ +

/ 接收器 \

+ +

| 不稳定： |

| ... |

<-----------------------------------------------|- -> - |

disp(role=receiver, ..., delivery-tag=DT, |) ... | settled=True, state=T\_0, ...) | |

+ +

图 2.20：最少运行一次

正如人们所猜测的那样，最初出现的情况是，当

如果接收方达到终端状态，而发送方结算时接收方应用程序也结算，则可保证*完全一致*。更一般地说，如果接收方在发送方之前结算，就有可能出现重复信息，除非发送方在其初始传输之前结算。同样，如果发送方在接收方到达终端状态之前结算，信息也有可能丢失。

有关结算的发送方和接收方策略既可以为整个链路预先配置，从而优化端点选择，也可以为每次传送临时确定。应用程序也可以选择在端点进行结算，而不考虑其交付状态，例如，发送应用程序可以选择结算因信息 ttl 过期而产生的交付，而不管接收方是否已达到终端状态。

## 恢复交付

当有未结算交货的暂停链路恢复时，[附加](#_bookmark77)帧中的*未结算*字段将包含发出链路端点认为未结算的所有交货的交货标记和交货状态。两个端点的未结算映射中包含的一系列交付标记和交付状态可分为三类：

**只有来源方认为未结算的交货**

发送应用程序可自行决定恢复此类发送。如果发送应用程序将重发尝试标记为恢复发送，则接收方必须忽略它。(这就允许发送方在发送方没有重复风险的情况下进行管道恢复）。

**只有目标公司认为未结算的交货**

发件人必须忽略此类邮件，收件人必须将其视为已结算邮件。

**来源国和目标国都认为未结算的交货**

发件人必须恢复此类邮件的发送。

请注意，在端点表示未结算图不完整的情况下，未结算图中没有条目并不表示已结算。在这种情况下，两个端点必须通过发送方恢复和/或结算接收方认为未结算的传输，尽可能降低未结算状态的水平。状态减少完成后，双方必须暂停并重新尝试恢复链路。只有当双方都有完整的未结算地图时，才能通过发送非恢复传输来创建新的未结算状态。

恢复传送的方式与最初传送的方式大致相同，但有以下例外：

* 恢复传送时，[传送](#_bookmark79)帧的恢复标志必须设为 true。
* 当接收方的发送状态表明没有必要进行重传时，发送方可以省略报文数据。

请注意，未结算的递送标记在恢复之前不会关联任何有效的递送标识，因为其原始链接端点的递送标识对新链接端点毫无意义。

## 传输大容量信息

每个[传输](#_bookmark79)帧可携带任意数量的报文数据，但不得超过最大帧大小的限制。如果报文过大，无法容纳在最大帧大小内，则可通过在除最后一个传输帧外的所有[传输](#_bookmark79)帧上设置更多标记，在更多[传输](#_bookmark79)帧中传输额外数据。当信息以这种方式被分割成多个[传输](#_bookmark79)帧时、

沿不同链路传输的报文可以交错。但是，沿单一链路传输的报文不得交错。

发送方可通过在最后一次[传输](#_bookmark79)中设置中止标志来表明发送信息的尝试已中止。在这种情况下，接收方必须丢弃中止前传输的报文数据。

+------------+ s:xfr(m=1,a=0)

+------| NOT\_SENT | +

| +------------+ |

| |

| s:xfr(m=0,a=0) |

|  |  |
| --- | --- |
| | | | S:XFR(M=1,A=0) |
| | | | + + |
| | | | | | |
| | | \|/ \|/ | |
| | | +------------+ | |
| | | +----------------| 发送 | + |
| | | | s:xfr(m=0,a=0) + + |
| | | | | |
| | | | | |
| | | | s:xfr(m=0,a=1) |
| | | | | |

\|/ \|/ \|/

+------------+ + +

| 已发送 | 错误 |

+------------+ + +

关键字：S:XFR(M=?,A=?) --> 发送 TRANSFER(more=?, aborted=?)

图 2.21：传出分片状态图

+------------+ r:xfr(m=1,a=0)

+------| NOT\_RCVD | +

| +------------+ |

| |

| r:xfr(m=0,a=0) |

| | R:XFR(M=1,A=0)

| | + +

| | | |

| \|/ \|/ |

| +------------+ |

|+----------------| 接收 | +

| r:xfr(m=0,a=0) + +

| | |

| | |

| | r:xfr(m=0,a=1)

| | |

\|/ \|/ \|/

+------------+ + +

| 收到 | | 中止 |

+------------+ + +

键：R:XFR(M=?,A=?) --> 接收 TRANSFER(more=?, aborted=?)

图 2.22：传入碎片状态图

* 1. **表演节目**

## 开放

协商连接参数。

<type name="open" class="composite" source="list" provides="frame">

<descriptor name="amqp:open:list" code="0x00000000:0x00000010"/>

<field name="container-id" type="string" mandatory="true"/>

<field name="hostname" type="string"/>

<field name="max-frame-size" type="uint" default="4294967295"/>

<field name="channel-max" type="ushort" default="65535"/>

<field name="idle-time-out" type="milliseconds"/> 字段名称为 "空闲时间-输出"。

<field name="outgoing-locales" type="ietf-language-tag" multiple="true"/>

<field name="incoming-locales" type="ietf-language-tag" multiple="true"/>

<field name="offered-capabilities" type="symbol" multiple="true"/>

<field name="desired-capabilities" type="symbol" multiple="true"/>

<field name="properties" type="fields"/>

</类型

在任一方向的连接上发送的第一个帧都必须包含 Open 主体。(请注意，在连接上首先发送的连接头并不是\*\*帧）。这些字段表示发送对等方的能力和限制。

**实地详情**

container-id *源容器的 id*

主机名 *目标主机的名称*

发送对等程序正在连接的主机的 dns 名称（全称或相对名称）。并非必须提供主机名。如果未提供主机名，接收对等方应根据自身配置选择默认主机名。AMQP 代理可使用此字段确定要将客户端连接到的正确后端服务。

如果使用 SASL 层，该字段可能已由 sasl-init 框架指定；如果使用 TLS 层，该字段可能已由 RFC-4366 中描述的服务器名称指示扩展指定。目前尚不清楚不同于这些特定值的含义。

max-frame-size *建议的最大帧大小*

发送对等端在此连接上可接受的最大帧大小。如果未设置此字段，则表示对等方未规定任何特定限制。对等端不得发送大于其伙伴所能处理的帧。收到超大帧的对等端必须使用成帧错误（framing-error）错误代码关闭连接。

两个对等端都必须接受最大 [512 (MIN-MAX-FRAME-SIZE)](#_bookmark109) 个八进制字节的帧。

channel-max *连接上可使用的最大通道数*

通道最大值是连接上可使用的最高通道编号。该值加一就是连接上可同时激活的最大会话数。对等端不得使用超出其合作伙伴可处理范围的通道号。收到超出支持范围的信道号的对等端必须使用成帧错误（framing-error）错误代码关闭连接。

空闲时间*空闲超时*

发送方要求的空闲超时时间。空闲超时值为 0 与未设置空闲超时值相同。如果接收方无法或不愿意支持空闲超时，则应关闭连接，并给出错误解释（例如，因为超时太短）。如果未设置空闲超时值，则发送方没有空闲超时值。不过，发送方在这样做时应注意，实现者可能会选择使用内部默认值来有效管理对等方的资源。

outgoing-locales *发送文本可用的本地语言*

对等设备支持发送信息文本的本地语言列表。其中包括连接、会话和链路错误描述。对等方必须至少支持 *en-US* 本地[语言](#_bookmark98)（见[第 2.8.12 小节 IETF 语言标记](#_bookmark98)）。由于始终支持该值，因此无需在 outgoing-locales 中提供该值。空值或空列表表示只支持 *en-US。*

incoming-locales *按优先程度递减的所需接收文本的本地化语言*

发送端点允许接收信息文本的本地语言列表。该列表按优先级递减排列。接收伙伴将从其支持的语言中选择第一个（最优先）传入语言。如果所请求的本地语言都不支持，则选择 *en-US。请*注意，无需在此列表中提供 *en-US，*因为它始终是备用语言。对等方可通过检查 outgoing-locales 字段中指定的伙伴支持的本地语言，来确定选择哪个允许的传入本地语言。空值或空列表表示只支持 *en-US。*

offered-capabilities *发送方支持的扩展功能*

如果所提供功能的接收方需要的扩展功能不在所提供功能列表中，则接收方必须关闭连接。

通常定义的连接功能及其含义列表可在此处找到[：](http://www.amqp.org/specification/1.0/connection-capabilities) http://www.amqp.org/specification/1.0/connection-capabilities。

desired-capabilities *如果接收方支持，发送方可使用的扩展功能*

所需的功能列表（desired-capability list）定义了如果接收方提供哪些扩展功能，发送方可以使用这些功能（即这些功能在所需功能发送方收到的所提供功能列表中）。如果所需功能的接收方提供的扩展功能不在其收到的所需功能列表中，则可以确定这些（不需要的）功能不会在连接上使用。

属性 *连接属性*

属性映射包含一组字段，用于显示连接及其容器的相关信息。

常用定义的连接属性及其含义列表可在此处找到[： http://www.amqp.org/specification/1.0/connection-properties](http://www.amqp.org/specification/1.0/connection-properties)

## 开始

在频道上开始会话。

<type name="begin" class="composite" source="list" provides="frame">

<descriptor name="amqp:begin:list" code="0x00000000:0x00000011"/>

<field name="remote-channel" type="ushort"/>

<field name="next-outgoing-id" type="transfer-number" mandatory="true"/>

<field name="incoming-window" type="uint" mandatory="true"/>

<field name="outgoing-window" type="uint" mandatory="true"/>

<field name="handle-max" type="handle" default="4294967295"/>

<field name="offered-capabilities" type="symbol" multiple="true"/>

<field name="desired-capabilities" type="symbol" multiple="true"/>

<field name="properties" type="fields"/>

</类型

表示频道上的会话已经开始。

**实地详情**

远程通道 *此会话的远程通道*

如果会话是本地发起的，则远程信道不得设置。当端点响应远程发起的会话时，远程信道必须设置为远程会话发送开始的信道。

next-outgoing-id 发送*方将发送的第一个传输 ID 的传输 ID*

请参阅 [2.5.6 会话流量控制](#_bookmark58)。

接收窗口*（*incoming-window*） 发件人的初始接收窗口*

请参阅 [2.5.6 会话流量控制](#_bookmark58)。

发件窗口*（*outgoing-*window）发件人的初始发件窗口*

请参阅 [2.5.6 会话流量控制](#_bookmark58)。

句柄最大值 *会话中可能使用的最大句柄值*

句柄最大值是会话中可使用的最高句柄值。对等方不得使用超出其伙伴可处理范围的句柄值尝试连接链路。收到超出支持范围的句柄的对等端必须使用成帧错误代码（framing-error error-code）关闭连接。

offered-capabilities *发送方支持的扩展功能*

常用会话功能及其含义列表可在此处找到：

http://www.amqp.org/specification/1.0/session-capabilities。

desired-capabilities *如果接收方支持，发送方可使用的扩展功能*

属性 *会话属性*

属性映射包含一组字段，用于显示会话及其容器的相关信息。

常用会话属性及其含义列表可在此处找到：

http://www.amqp.org/specification/1.0/session-properties。

## 附件

将链接附加到会话。

<type name="attach" class="composite" source="list" provides="frame">

<descriptor name="amqp:attach:list" code="0x00000000:0x00000012"/>

<field name="name" type="string" mandatory="true"/>

<field name="handle" type="handle" mandatory="true"/>

<field name="role" type="role" mandatory="true"/>

<field name="snd-settle-mode" type="sender-settle-mode" default="mixed"/>

<field name="rcv-settle-mode" type="receiver-settle-mode" default="first"/>

<field name="source" type="\*" requires="source"/>

<field name="target" type="\*" requires="target"/>

<field name="unsettled" type="map"/>

<field name="incomplete-unsettled" type="boolean" default="false"/>

<field name="initial-delivery-count" type="sequence-no"/>

<field name="max-message-size" type="ulong"/>

<field name="offered-capabilities" type="symbol" multiple="true"/>

<field name="desired-capabilities" type="symbol" multiple="true"/>

<field name="properties" type="fields"/>

</类型

[附加](#_bookmark77)帧表示链接端点已附加到会话。打开标志用于表示链接端点是新创建的。

**实地详情**

名称 *链接名称*

例如，如果源节点的容器是 A*，*而目标节点的容器是 B，那么链接可以用（有序）元组 *(A,B,<name>)* 进行全局标识。

处理

该句柄不得用于其他打开的链接。如果尝试使用已与链接关联的句柄进行连接，则必须立即[关闭](#_bookmark83)，并附上 "[句柄](#_bookmark104)使用[中会话错误](#_bookmark104)"（Handle-in-use [session-error](#_bookmark104)）。

为便于监控 AMQP 链路附加帧，建议实施时始终为该字段分配可用的最低句柄。

角色 *链接端点的角色*

*snd-settle-mode 发送端结算模式*

决定在发件人处发送的交货的结算政策。在收件人处设置时，这表示发件人处结算模式的期望值。在发件人处设置时，这表示实际使用的结算模式。

rcv-settle-mode *接收机的结算模式*

决定收件人收到的未结算交货的结算政策。在发件人处设置时，这表示收件人处结算模式的期望值。在收件人处设置时，这表示实际使用的结算模式。

消息来源 *消息来源*

如果出站链接上未指定消息来源，则表示该链接当前没有连接消息来源。没有来源的链接永远不会产生发出的信息。

目标 *信息的目标*

如果没有在传入链接上指定目标，则表示当前没有目标连接到该链接。没有目标的链接永远不会允许信息传入。

未结算 *未结算交割状态*

当暂停的链路恢复时，它用于指示任何未定的交付状态。该映射以递送标记为键值，其值表示递送状态。必须比较给定递送标记的本地和远程递送状态，以解决任何有疑问的递送。如有必要，可根据比较结果重新发送或恢复发送。请参阅 [2.6.13 恢复递送](#_bookmark72)。

如果本地未设定地图太大，无法在约定的最大帧大小的帧内编码，那么会话可能会以帧大小太小错误（见 [amqp-error](#_bookmark101)）结束。端点应利用发送不完整未设定映射的功能（见下文）来避免发送错误。

未设定映射不得包含空值键。

重新连接（而不是恢复）时，未设置的地图必须为空。

不完整-不安

如果设置为 "true"，则表示所提供的未结算地图不完整。当地图不完整时，地图接收方不能将地图中没有派送标记作为结算的证据。在收到不完整的未结算地图时，发送端点不得向其伙伴发送任何新的传送（即恢复标志未设置为 true 的传送）（发送不完整的未结算地图的接收端点在收到未将恢复标志设置为 true 的传送时，必须以错误的方式脱离）。

初始交付次数

如果角色是发送方，该值不得为空；如果角色是接收方，该值将被忽略。参见

[2.6.7 流量控制](#_bookmark66)。

max-message-size *链路端点支持的最大报文大小*

该字段表示链路端点支持的最大报文大小。任何试图传送大于此大小的报文的行为都会导致报文大小超出[链接错误](#_bookmark106)。如果该字段为零或未设置，则表示链路端点没有规定最大报文大小。

offered-capabilities *发送方支持的扩展功能*

常用会话功能及其含义列表可在此处找到：

http://www.amqp.org/specification/1.0/link-capabilities。

所需的功能 *如果接收方支持，发送方可使用的扩展功能*

属性 *链接属性*

属性图包含一组字段，用于显示链接及其容器的相关信息。

常用链接属性及其含义列表可在此处找到：

<http://www.amqp.org/specification/1.0/link-properties>

## 流量

更新链接状态。

<type name="flow" class="composite" source="list" provides="frame">

<descriptor name="amqp:flow:list" code="0x00000000:0x00000013"/>

<field name="next-incoming-id" type="transfer-number"/>

<field name="incoming-window" type="uint" mandatory="true"/>

<field name="next-outgoing-id" type="transfer-number" mandatory="true"/>

<field name="outgoing-window" type="uint" mandatory="true"/>

<field name="handle" type="handle"/>

<field name="delivery-count" type="sequence-no"/>

<field name="link-credit" type="uint"/>

<field name="available" type="uint"/>

<field name="drain" type="boolean" default="false"/>

<field name="echo" type="boolean" default="false"/>

<field name="properties" type="fields"/>

</类型

更新指定链接的流量状态。

**实地详情**

next-incoming-id

标识下一个传入[传输](#_bookmark79)帧的预期传输 ID。只有当发送方尚未收到会话的[开始](#_bookmark76)帧时，才会设置此值。详见 [2.5.6 会话流量控制。](#_bookmark58)

来电窗口

定义端点当前可接收的最大[传入传输](#_bookmark79)帧数。详见 [2.5.6 会话流量控制。](#_bookmark58)

下一个去向标识

将分配给下一个传出[传输](#_bookmark79)帧的传输 ID。详情请参阅 [2.5.6 会话流量控制。](#_bookmark58)

outgoing-window

定义端点当前可能发送的最大传出[传输](#_bookmark79)帧数，前提是它不受对等体传入窗口的限制。详见 [2.5.6 会话流量控制。](#_bookmark58)

处理

如果设置，则表示流帧携带与给定句柄关联的本地链接端点的流状态信息。如果未设置，则流帧只携带与会话端点有关的信息。

如果设置为当前未与所附链接关联的句柄，接收方必须[以未](#_bookmark105)附加句柄会话错误结束会话。

*端点的*交付*次数*

未设置句柄字段时，不得设置该字段。

当句柄确定流状态正从发送链路端点发送至接收链路端点时，该字段必须设置为链路端点的当前发送计数。

当流量状态从接收端点发送到发送端点时，该字段必须设置为相应发送端点的最后已知值。如果接收链路端点尚未看到来自发送端点的初始[附加](#_bookmark77)帧，则不得设置该字段。

详见 [2.6.7 流量控制。](#_bookmark66)

link-credit *当前可接收信息的最大数量*

链路接收端点当前可处理的最大信息数量。只有接收端点可以独立设置此值。发送端点将其设置为从接收端点看到的最后一个已知值。参见

[2.6.7 流量控制](#_bookmark66)，了解更多详情。

未设置句柄字段时，不得设置该字段。

*可用信息的数量*

链接发送端点等待计入的信息数量。只有发送方可独立设置此值。接收方将其设置为从发送方看到的最后已知值。详见 [2.6.7 流量控制。](#_bookmark66)

未设置句柄字段时，不得设置该字段。

排水 *表示排水模式*

当流量状态从发送端发送至接收端时，该字段包含发送端的实际耗尽模式。当流量状态从接收方发送至发送方时，该字段包含接收方所需的耗尽模式。详见 [2.6.7 流量控制。](#_bookmark66)

未设置句柄字段时，不得设置该字段。

回声 *从其他端点请求链路状态*

属性 *链接状态属性*

常用链接状态属性及其含义的列表可在此处找到[： http://www.amqp.org/specification/1.0/link-state-properties](http://www.amqp.org/specification/1.0/link-state-properties)

## 转让

传递信息。

<type name="transfer" class="composite" source="list" provides="frame">

<descriptor name="amqp:transfer:list" code="0x00000000:0x00000014"/>

<field name="handle" type="handle" mandatory="true"/>

<field name="delivery-id" type="delivery-number"/>

<field name="delivery-tag" type="delivery-tag"/>

<field name="message-format" type="message-format"/>

<field name="settled" type="boolean"/>

<field name="more" type="boolean" default="false"/>

<field name="rcv-settle-mode" type="receiver-settle-mode"/>

<field name="state" type="\*" requires="delivery-state"/>

<field name="resume" type="boolean" default="false"/>

<field name="aborted" type="boolean" default="false"/>

<field name="batchable" type="boolean" default="false"/>

</类型

传输帧用于在链路上发送信息。信息可通过单个传输帧传输，但不得超过连接的最大协商帧大小。较大的报文可分割成多个传输帧。

**实地详情**

处理

指定传送报文的链接。

delivery-id *作为 delivery-tag 的别名*

在多次传送的第一次传送中，必须提供传送 ID。在继续转移时，可以省略交货标识。如果续转中的递送标识与首次递送中的递送标识不同，则属于错误。

交货标签

唯一标识此链接上给定报文的传送尝试。多传送报文的第一次传送必须指定此字段，只有继续传送时才可省略。

message-format *表示信息格式*

多路传送报文的第一次传送必须指定此字段，只有在继续传送时才可省略。

稳定的

如果在第一次（或唯一一次）传送时未设置，则必须将结算标记解释为假。对于随后的传输，如果未设置已结算标记，则必须将其解释为 "真"，前提是且仅当之前的任何传输中已结算标记的值为 "真 "时；如果之前的任何传输都未将已结算标记设置为 "真"，则未设置时的值必须视为 "假"。

如果连接时协商的 snd-settle-mode 值为[结算](#_bookmark87)，则该字段

必须至少在一个传送帧中为真（即在传送完毕时，必须在发件人处结算）。

如果连接时协商的 snd-settle-mode 值为[未结算](#_bookmark87)，则该字段

必须在每个传送帧上为 false（或取消设置）（除非传送被中止）。

更多 *表示信息有更多内容*

请注意，如果 more 字段和 aborted 字段都设置为 true，则以 aborted 标志优先。也就是说，如果传输被标记为中止，接收方应忽略 more 字段的值。如果发送方同时将中止标志设置为 true，则不应将 more 标志设置为 true。

rcv-settle-mode

如果[为先](#_bookmark89)，则表示收件人必须在快递到达后立即结算，而无需等待发件人先结算。

如果是[第二种](#_bookmark89)情况，则表示接收方在向发送方发送其处置并从发送方收到结算处置之前，不得进行结算。

如果未设置，该值默认为链路连接时协商的值。

如果协商的链接值为[第一](#_bookmark89)，则将该字段设置为[第二](#_bookmark89)是非法的。

如果报文是由发件人结算发送的，则该字段的值将被忽略。该字段（隐式或显式）的值不构成传输状态的一部分，在链路暂停后又恢复时也不会保留。

状态 *发件人的交货状态*

设置后，接收方会了解发件人的交货状态。这在恢复链接后恢复未结算货件的转运时特别有用。在传输中设置状态可视为等同于在[传输](#_bookmark79)执行语句之前发送一个处置语句，也就是说，它是发送帧时存在的交付状态（而不是传输状态）。需要注意的是，如果[传输](#_bookmark79)执行语（或提及该传送的早期[处置](#_bookmark80)执行语）表明该传送已达到终端状态，那么发送方今后发送的任何[传输](#_bookmark79)或[处置](#_bookmark80)都不能改变该终端状态。

恢复 *表示恢复投递*

如果为 "true"，则恢复标志表示传输用于重新关联已解散链接端点的未结算交付。详情请参阅 [2.6.13 恢复交付。](#_bookmark72)

接收方必须忽略不在其本地未结算图中的恢复递送。发送方不得为不在其本地未结算映射中的递送发送恢复递送。

如果恢复的传送跨越一个以上的传送执行器，则必须在恢复的传送的第一次传送中将恢复标记设置为 "true"。对于同一传输的后续传输，恢复标志可以设为 true，也可以省略。如果未结算映射交换清楚地表明所有报文数据都已成功传送到接收方，只需传送发送方的最终状态（可能还有结算），那么恢复的传送可以不带有效载荷，而只作为传送发送方的最终状态的载体。

aborted *表示信息已终止*

收件人应丢弃中止的报文（必须忽略携带执行内容的帧内的任何有效载荷）。中止的报文是隐式设置的。

可批处理 *可批处理提示*

如果为 "true"，则发行方暗示对等方无需紧急通报更新的交付状态。此提示可用于人为地增加实现在通信交付状态时使用的批处理量，从而节省带宽。

如果传送的信息太大，无法在单个帧内容纳，那么在传送的任何[传输](#_bookmark79)执行器上将可批处理设置为 true，就等同于在传送的所有[传输](#_bookmark79)执行器上将可批处理设置为 true。

可批处理值不构成传输状态的一部分，在链路暂停后又恢复时也不会保留。

## 处置

向远端对等设备通知发送状态的变化。

<type name="disposition" class="composite" source="list" provides="frame">

<descriptor name="amqp:disposition:list" code="0x00000000:0x00000015"/>

<field name="role" type="role" mandatory="true"/>

<field name="first" type="delivery-number" mandatory="true"/>

<field name="last" type="delivery-number"/>

<field name="settled" type="boolean" default="false"/>

<field name="state" type="\*" requires="delivery-state"/>

<field name="batchable" type="boolean" default="false"/>

</类型

处置帧用于向远端对等设备通报本地传送状态的变化。尽管所有链接都必须具有指定*角色*所指示的方向性，但处置帧可以引用与会话相关联的许多不同链接的交付。

请注意，从发送方发送到接收方的处置有可能指的是尚未完成的传送（即分散在多个帧中的传送，且并非所有帧都已发送）。我们不鼓励使用这种交错方式，而是将修改后的状态带入下一个[传送](#_bookmark79)执行器。

处置执行符可指不再连接的链接上的交付。只要链接没有被关闭或因错误而脱离，那么这些交货就仍然是 "有效 "的，并且必须应用更新的状态。

**实地详情**

作用 *处置方向性*

该角色可确定处置帧是否包含有关*发送*的信息。

链路端点或*接收*链路端点。

第一 *交付下限*

确定该集合中交货标识的下限。

最后 *交货上限*

确定此集合中交付货品的交付货品 ID 上限。如果未设置，则视为与*第一个*相同。

已结算 *表示交货已结算*

如果为 "true"，表示签发端点认为所引用的交付已结算。

状态 *表示交货状态*

传达该处置所引用的所有交付的状态。

可批处理 *可批处理提示*

如果为 "true"，则发行方暗示对等方无需紧急通报更新后的交付状态的影响。此提示可用于人为地增加实现在发送状态通信时使用的批处理量，从而节省带宽。

## 脱离

将链接端点从会话中分离。

<type name="detach" class="composite" source="list" provides="frame">

<descriptor name="amqp:detach:list" code="0x00000000:0x00000016"/>

<field name="handle" type="handle" mandatory="true"/>

<field name="closed" type="boolean" default="false"/>

<field name="error" type="error"/>

</类型

从会话中分离链接端点。这将取消句柄的映射，使其可供其他链接使用。

**实地详情**

*要分离的链接的本地句柄*

closed *如果为 true，则表示发件人已关闭链接*

请参阅 [2.6.6 关闭链接](#_bookmark65)。

错误 *导致分离的错误*

如果设置了该字段，则表示链接因错误条件而被分离。该字段值应包含错误原因的详细信息。

## 结束

结束会议。

<type name="end" class="composite" source="list" provides="frame">

<descriptor name="amqp:end:list" code="0x00000000:0x00000017"/>

<field name="error" type="error"/>

</类型

表示会话已结束。

**实地详情**

错误*错误导致结束*

如果设置了该字段，则表示会话因出错而结束。该字段值应包含错误原因的详细信息。

## 关闭

发出连接关闭信号。

<type name="close" class="composite" source="list" provides="frame">

<descriptor name="amqp:close:list" code="0x00000000:0x00000018"/>

<field name="error" type="error"/>

</类型

发送关闭信号表示发送方将不再在连接上发送任何帧（或任何其他类型的字节）。有序关闭要求发送方必须写入此帧。发送关闭帧后，再发送任何帧（或任何其他类型的字节）都是非法的。

**实地详情**

*导致关闭的错误*

如果设置了该字段，则表示连接因错误条件而关闭。该字段值应包含错误原因的详细信息。

* 1. **定义**

## 角色

链接端点角色。

<type name="role" class="restricted" source="boolean">

<选择项 name="sender" value="false"/>

<choice name="receiver" value="true"/>

</类型

**有效值**

**错误** 发件人

**真** 接收器

## 发送端设置模式

发件人的结算政策。

<type name="sender-settle-mode" class="restricted" source="ubyte">

<选择项 name="unsettled" value="0"/>

<choice name="settled" value="1"/>

<choice name="mixed" value="2"/>

</类型

**有效值**

1. 发件人将把所有最初未结算的交货发送给收件人。
2. 发件人将把所有已结清的交货发送给收件人。
3. 发件人可以向收件人发送已交收和未交收的混合货件。

## 接收器设置模式

破产管理人的结算政策。

<type name="receiver-settle-mode" class="restricted" source="ubyte">

<choice name="first" value="0"/>

<choice name="second" value="1"/>

</类型

**有效值**

1. 接收器将自动结算所有转入的转账。
2. 收件人只有在向发件人发送了处理结果并收到发件人的[处理结果](#_bookmark80)后，才会进行结算。

## 手柄

林克的手柄

<type name="handle" class="restricted" source="uint"/> 返回值

由[连接](#_bookmark77)帧建立的别名，随后被端点用作在所有传出帧中引用链路的简称。两个端点有可能使用不同的句柄来指代同一链路。一旦链路关闭，发送和接收都可重复使用链路句柄。

## 秒数

以秒为单位的持续时间。

<type name="seconds" class="restricted" source="uint"/>

## 毫秒

以毫秒为单位的持续时间。

<type name="milliseconds" class="restricted" source="uint"/> 类型名称="毫秒" class="restricted" source="uint"/> 限制条件

## 送货标签

<type name="delivery-tag" class="restricted" source="binary"/>

一个发送标记最多可以是 32 个八进制二进制数据。

## 交货编号

<type name="delivery-number" class="restricted" source="sequence-no"/>

## 转移编号

<type name="transfer-number" class="restricted" source="sequence-no"/> 类型名称="传输编号" class="restricted" 来源="序列号"/>

## 序列号

32 位 RFC-1982 序列号。

<type name="sequence-no" class="restricted" source="uint"/> <type name="sequence-no" class="restricted" source="uint"/>

序列号编码 RFC-1982 中定义的序列号。这些数字的算术和运算符由 RFC-1982 定义。

## 信息格式

32 位报文格式代码。

<type name="message-format" class="restricted" source="uint"/> 消息格式

报文格式代码的前三个八位位组用于标识特定的报文格式。最低八位位组表示所述报文格式的版本。格式的任何给定版本都与所有更高的版本向前兼容。

1. 八位组 1

+----------------+ +

| 信息格式

+----------------+ +

| |

msb lsb

## IETF 语言标记

BCP 47 定义的 IETF 语言标记。

<type name="ietf-language-tag" class="restricted" source="symbol"/>

IETF 语言标记是 IETF 当前最佳实践 [BCP-47](http://www.rfc-editor.org/rfc/bcp/bcp47.txt) [(http://www.rfc-editor.org/rfc/bcp/bcp47.txt)](http://www.rfc-editor.org/rfc/bcp/bcp47.txt)) 中定义的缩写语言代码（包含 [RFC-5646](http://www.rfc-editor.org/rfc/rfc5646.txt) (http://www.rfc- editor.org/rfc/rfc5646.txt)）。已注册的子标记列表保存在 [IANA 语言子标记注册中心](http://www.iana.org/assignments/language-subtag-registry) [(http://www.iana.org/assignments/language-subtag-registry) 中。](http://www.iana.org/assignments/language-subtag-registry))

所有 AMQP 实现至少应能理解 IETF 语言标记 *en-US*（注意，它使用的是连字符分隔符，而不是下划线）。

## 字段

从字段名到值的映射。

<type name="fields" class="restricted" source="map"/>

*字段*类型是一个映射，其中的键被限制为符号类型（这排除了键为空的可能性）。*字段*类型对条目允许的值或允许的键集没有进一步的限制。

## 错误

错误详情。

<type name="error" class="composite" source="list">

<descriptor name="amqp:error:list" code="0x00000000:0x0000001d"/>

<字段 name="condition" type="symbol" requires="error-condition" mandatory="true"/>

<field name="description" type="string"/>

<field name="info" type="fields"/>

</类型

**实地详情**

条件 *错误条件*

表示错误条件的符号值。

description *关于错误条件的描述性文本*

此文本提供条件字段未指明的任何补充细节。此文本可作为解决问题的辅助记录。

信息 *包含错误信息的映射*

## AMQP 错误

共享错误条件。

<type name="amqp-error" class="restricted" source="symbol" provides="错误条件">。

<choice name="internal-error" value="amqp:internal-error"/>

<choice name="not-found" value="amqp:not-found"/>

<choice name="unauthorized-access" value="amqp:unauthorized-access"/>

<choice name="decode-error" value="amqp:decode-error"/>

<choice name="resource-limit-exceeded" value="amqp:resource-limit-exceeded"/>

<choice name="not-allowed" value="amqp:not-allowed"/>

<choice name="invalid-field" value="amqp:invalid-field"/>

<choice name="not-implemented" value="amqp:not-implemented"/>

<choice name="resource-locked" value="amqp:resource-locked"/>

<choice name="precondition-failed" value="amqp:precondition-failed"/>

<choice name="resource-deleted" value="amqp:resource-deleted"/>

<choice name="illegal-state" value="amqp:illegal-state"/>

<choice name="frame-size-too-small" value="amqp:frame-size-too-small"/>

</类型

**有效值** **amqp:internal-error**

发生内部错误。可能需要操作员干预才能恢复正常运行。

**amqp:not-found**

一个对等设备试图与一个不存在的远程实体协同工作。

**amqp:未经授权的访问**

一个对等设备试图与一个远程实体协同工作，但由于安全设置原因，它无法访问该远程实体。

**amqp:decode-error**

数据无法解码。

**amqp:resource-limit-exceeded**

一个对等点超出了资源分配。

**amqp:not-allowed**

对等方试图以与规范中定义的语义不一致的方式使用帧。

**amqp:invalid-field**

帧体中传递了一个无效字段，操作无法继续。

**amqp:not-implemented**

同行试图使用其合作伙伴未实现的功能。

**amqp:resource-locked**

客户机试图处理其无法访问的服务器实体，因为另一个客户机正在处理该实体。

**amqp:precondition-failed**

客户提出的请求因某些前提条件失败而不被允许。

**amqp:resource-deleted**

客户端正在使用的服务器实体已被删除。

**amqp：非法状态**

对等方发送了会话当前状态下不允许的帧。

**amqp:frame-size-too-small**

对等方无法发送帧，因为使用当前有效值的最小执行编码太大，无法容纳在约定的最大帧大小的帧内。在传输报文时，报文数据可以分多个[传输](#_bookmark79)帧发送，从而避免了这种错误。同样，当连接一个有较大未确定映射的链接时，端点可以使用未完成-未确定标记来避免使用过大的帧。

## 连接错误

用于表示连接错误条件的符号。

<type name="connection-error" class="restricted" source="symbol" provides="错误条件">。

<choice name="connection-forced" value="amqp:connection:forced"/>

<choice name="framing-error" value="amqp:connection:framing-error"/>

<choice name="redirect" value="amqp:connection:redirect"/>

</类型

**有效值 amqp:connection:forced**

操作员出于某种原因干预关闭了连接。客户端可以稍后重试。

**amqp:connection:framing-error**

传入的字节流无法形成有效的帧头。

**amqp:connection:redirect**

当前连接上的容器已不可用。对等方应使用 info map 中提供的详细信息尝试重新连接容器。 **主机名** 容器的主机名。这是应

在[开放](#_bookmark75)框架的*主机名*字段中提供，然后

SASL 和 TLS 协商（如果使用）。

**网络主机** 托管容器的机器的 DNS 主机名或 IP 地址。

**端口** 托管容器的机器上的端口号。

## 会话错误

用于表示会话错误条件的符号。

<type name="session-error" class="restricted" source="symbol" provides="error-condition">

<choice name="window-violation" value="amqp:session:window-violation"/>

<choice name="errant-link" value="amqp:session:errant-link"/>

<choice name="handle-in-use" value="amqp:session:handle-in-use"/>

<choice name="unattached-handle" value="amqp:session:unattached-handle"/>

</类型

**有效值** **amqp:session:window-violation**

会话的对等方违规接收窗口。

**amqp:session:errant-link**

已收到链接的输入，但该链接已分离，且出现错误。

**amqp:session:handle-in-use(使用中的句柄)**

收到的连接使用的句柄已被连接的链接使用。

**amqp:session:unattached-handle**

收到一个帧（非附加），该帧引用的句柄当前未用于已连接的链接。

## 链接错误

用于表示链路错误条件的符号。

<type name="link-error" class="restricted" source="symbol" provides="错误条件">。

<choice name="detach-forced" value="amqp:link:detach-forced"/>

<choice name="transfer-limit-exceeded" value="amqp:link:transfer-limit-exceeded"/>

<choice name="message-size-exceeded" value="amqp:link:message-size-exceeded"/>

<choice name="redirect" value="amqp:link:redirect"/>

<choice name="stolen" value="amqp:link:stolen"/>

</类型

**有效值**

**amqp:link:detach-forced**

操作员出于某种原因干预了分离。

**amqp:link:transfer-limit-exceeded**

对等设备发送的信息传输量超过了链路当前允许的传输量。

**amqp:link:message-size-exceeded**

对等方发送的报文大于链路支持的范围。

**amqp:link:redirect**

所提供的地址无法解析为当前容器中的总站。信息地图可能包含以下信息，以便客户端定位到总站的附件。

**主机名** 托管终端的容器的主机名。该值应在[开放](#_bookmark75)框架的*主机名*字段中提供，并在 SASL 和 TLS 协商过程中提供（如果使用）。

**网络主机** 托管容器的机器的 DNS 主机名或 IP 地址。

**端口** 托管容器的机器上的端口号。

**地址** 集装箱终点站的地址。

**amqp:link:stolen**

链接已附加到其他地方，导致现有附件被强制关闭。

## 常量定义

|  |  |  |
| --- | --- | --- |
| 港口 | 5672 | 是 IANA 为 AMQP 分配的端口号。  IANA 为 TCP、UDP 和 SCTP 分配的标准 AMQP 端口号。 |
|  |  | 目前没有为 UDP 或 SCTP 定义映射。  AMQP。端口号是为将来与这些协议的传输映射保留的。 |
| 安全端口 | 5671 | IANA 为安全 AMQP (amqps) 分配的端口号。IANA 为使用 TLS 的安全 TCP 分配的标准 AMQP 端口号。  监听此端口的实施程序不应期望在协商 TLS 之前进行协议握手。 |
| 重大 | 1 | 主要协议版本。 |
| MINOR | 0 | 次要协议版本。 |
| 修订 | 0 | 协议修订。 |

最小-最大-帧大小 512 商定的最大帧大小的下限（以

字节）。

在初始连接协商期间，两个对等设备必须就最大帧大小达成一致。该常数限定了可设置最大帧大小的最小值。通过定义该值，对等方可保证在就该 Connec- tion 的最大帧大小达成一致之前，可以发送最大为该值的帧。

**第三册**

**信息传递**

* 1. **导言**

消息层建立在第一册和第二册所述概念的基础之上。传输层提供了许多适用于各种不同消息传递应用的扩展点。消息层规定了这些扩展点的标准化使用，以提供可互操作的消息传递功能。本标准包括

* 信息格式
  + 裸信息的属性
  + 裸电文中结构化和非结构化部分的格式
  + 注释邮件的页眉和页脚
* 节点间信息的传递状态
* 配送节点
  + 存储在分发节点的信息的状态
* 来源和目标
  + 转让的默认处置
  + 支持的成果
  + 从一个节点过滤信息
  + 分配模式，用于访问存储在分配节点的信息
  + 按需创建节点
  1. **信息格式**

在报文传送领域，报文一词有多种含义。发送方可能喜欢将消息视为一个不可变的有效负载，交给消息基础架构进行传递。接收方通常认为，信息不仅是发送方的不可变有效载荷，还包括信息基础架构在传递过程中提供的各种注释。为了避免混淆，我们将术语 "*裸信息 "*正式定义为发送方提供的信息，而术语 "*注释信息 "则*指接收方看到的信息。

*带注释的报文*由原始报文和原始报文头部和尾部的注释部分组成。注释分为两类：一类是无限期随信息传播的注释，另一类是被下一个节点消耗的注释。

*裸报文*由标准属性、应用程序属性和应用程序数据（正文）部分组成。

赤裸裸的信息

|

.---------------------+ --.

| |

+--------+-------------+-------------+------------+--------------+--------------+ +

| 标题 | 发送- | 消息- | 属性 | 应用程序- | 应用程序- | 页脚

|| 注释 | 注释 | 属性 | 数据 | |

+--------+-------------+-------------+------------+--------------+--------------+ +

| |

’-------------------------------------------+ '

| 注释信息

裸信息在 AMQP 网络中是不可变的。也就是说，作为 AMQP 中间人的任何节点都不能更改其中的任何部分。如果裸报文的某个部分被省略，中间节点也不能插入该部分。裸报文各部分的精确编码不得修改。这将保留基于裸报文二进制编码的报文哈希值、HMAC 和签名。

报文的确切结构及其编码由报文格式定义。本文件定义了报文格式 [0（MESSAGE-FORMAT）](#_bookmark128)的结构和语义。总的来说，报文由以下部分组成：

* 零或一个[标头](#_bookmark113)。
* 零或一个[交付说明](#_bookmark114)。
* 零个或一个[信息注释](#_bookmark115)。
* 零或一个[属性](#_bookmark116)。
* 零或一个[应用程序属性](#_bookmark117)。
* 主体包括：一个或多个[数据](#_bookmark118)部分、一个或多个 [amqp-sequence](#_bookmark119) 部分或一个 [amqp-value](#_bookmark120) 部分。
* 零或单[页脚](#_bookmark121)。

## 页眉

信息的传输标头。

<type name="header" class="composite" source="list" provides="section">

<descriptor name="amqp:header:list" code="0x00000000:0x00000070"/>

<field name="durable" type="boolean"/>

<field name="priority" type="ubyte"/>

<field name="ttl" type="milliseconds"/>

<field name="first-acquirer" type="boolean"/>

<field name="delivery-count" type="uint"/>

</类型

报文头（header）部分包含通过 AMQP 网络传输报文的标准传送细节。如果省略了报文头部分，接收方必须为[报文头中](#_bookmark113)的字段设置适当的默认值，除非已设置了其他目标或节点特定的默认值。

**实地详情**

耐久性 *规定耐久性要求*

即使中间人意外被删除或重启，持久信息也不会丢失。 不能满足这一保证的目标机不得接受持久性标头设为 true 的信息：如果信源允许被[拒绝的](#_bookmark134)结果，那么信息应被拒绝，并出现[前提条件失败的](#_bookmark102)错误，否则接收方必须分离链接，并出现同样的错误。

优先权*相对 信息优先权*

该字段包含相对的信息优先级。数字越大表示信息的优先级越高。优先级较高的信息可能会在优先级较低的信息之前发送。

实施不同优先级的 AMQP 中介必须按以下方式进行：

* 如果执行了 n 个不同的优先级，且 n 小于 10 - 优先级 0 至 (5 - ceiling(n/2))必须等同处理，且必须是最低有效优先级。优先级（4 + floor(n/2)）及以上必须等同处理，并且必须是最高有效优先级。优先级（5 - 上限（n/2））至（4 + 下限（n/2））必须被视为不同的优先级。
* 如果实施了 n 个不同的优先级，且 n 为 10 或更大，则优先级 0 至 (n - 1) 必须是不同的，优先级 n 及以上必须等同于优先级 (n - 1)。

因此，举例来说，如果有 2 个不同的优先级，那么 0 至 4 级是等价的，5 至 9 级是等价的，4 和 5 级是不同的。如果有 3 个不同的优先级，那么 0 至 3 级是等价的，5 至 9 级是等价的，3、4 和 5 级是不同的。

这种方案确保，如果一个服务器有 m 个不同的优先级，那么对于有 n 个不同优先级（n *>* m）的服务器来说，这两个优先级也是不同的。

ttl *毫秒生活时间*

信息被视为 "实时 "的持续时间（毫秒）。如果设置了该值，则将根据信息到达中介的时间计算信息过期时间。超过过期时间的报文将被丢弃（或死字）。当中间人传送带有 ttl 的信息时，传送的信息头应包含一个 ttl，该 ttl 是根据当前时间与之前计算的信息过期时间（即缩短的 ttl）之间的差值计算得出的，因此，如果信息最终陷入传送循环，它们最终也会死亡。

第一收购人

如果该值为 true，则表示该信息未被任何其他链接获取。如果该值为假，则该信息可能已被另一个或多个链接获取。

投递--计算*之前不成功的投递尝试次数*

之前尝试递送此信息失败的次数。如果该值不为零，则表明该信息可能是重复发送。首次递送时，该值为零。根据为每种结果定义的规则，在发件人结算结果时，该值会递增。

## 交付注释

<type name="delivery-annotations" class="restricted" source="annotations" provides="section">

<descriptor name="amqp:delivery-annotations:map" code="0x00000000:0x00000071"/>

</类型

递送注释部分用于在报文头部显示特定于递送的非标准属性。传递注释将信息从发送对等节点传递给接收对等节点。如果收件人不理解注释，就无法对其采取行动，也无法对其效果（如任何隐含的传播）采取行动。注解可以是一种实现的特定注解，也可以是多种实现的共同注解。在链接附加以及[源](#_bookmark141)和[目标](#_bookmark142)上协商确定的功能应用于确定对等方支持哪些注释。已定义注解及其含义的注册表可在此处找到[：http://www.amqp.org/specification/1.0/ delivery-annotations](http://www.amqp.org/specification/1.0/delivery-annotations)。

如果省略了 "交付-注释 "部分，则等同于 "交付-注释 "部分包含一个空的注释地图。

## 信息注释

<type name="message-annotations" class="restricted" source="annotations" provides="section">

<descriptor name="amqp:message-annotations:map" code="0x00000000:0x00000072"/>

</类型

信息注释部分用于说明信息的属性，这些属性针对的是基础结构，并应在每个传送步骤中传播。报文注释传递有关报文的信息。中间人必须传播注释，除非注释被明确增强或修改（如使用[修改](#_bookmark136)后的结果）。

在链接附加以及[源](#_bookmark141)和[目标](#_bookmark142)上协商的能力可用于确定对等方能理解哪些注释，但在 AMQP 中介网络中，可能无法知道是否每个中介都能理解注释。需要注意的是，对于某些注释，中间人可能没有必要了解它们的用途--它们可能纯粹被用作可以过滤的属性。

已定义注释及其含义的注册表可在此处找到[：http://www.amqp.org/ specification/1.0/message-annotations](http://www.amqp.org/specification/1.0/message-annotations)。

如果省略报文注释部分，则等同于报文注释部分包含一个空的注释映射。

## 属性

信息的不变属性。

<type name="properties" class="composite" source="list" provides="section">

<descriptor name="amqp:properties:list" code="0x00000000:0x00000073"/>

<field name="message-id" type="\*" requires="message-id"/>

<field name="user-id" type="binary"/>

<field name="to" type="\*" requires="address"/>

<field name="subject" type="string"/>

<field name="reply-to" type="\*" requires="address"/>

<field name="correlation-id" type="\*" requires="message-id"/>

<field name="content-type" type="symbol"/>

<field name="content-encoding" type="symbol"/>

<field name="absolute-expiry-time" type="timestamp"/>

<field name="creation-time" type="timestamp"/>

<field name="group-id" type="string"/>

<field name="group-sequence" type="sequence-no"/>

<field name="reply-to-group-id" type="string"/>

</类型

属性部分用于定义电文的一系列标准属性。属性部分是裸报文的一部分，因此在中间人重新传输时必须保持完全不变。

**实地详情**

message-id *应用程序 消息标识符*

Message-id 是一个可选属性，用于在 Message 系统中唯一标识 Message。报文生产者通常负责设置报文 id，以确保它是全局唯一的。如果 Message-id 的值与先前接收到的发送到同一节点的 Message 的值相匹配，则代理可以将该 Message 作为重复的 Message 丢弃。

user-id *创建用户 ID*

负责生成信息的用户的身份。该值由客户端设置，可由中间人验证。

到 *信息目的地节点的地址*

收件人（to）字段标识了作为报文预定目的地的节点。在任何给定的传输中，这可能不是链路接收端的节点。

主题 *邮件主题*

信息内容和目的摘要信息的常用字段。

回复到 *节点发送回复*

要发送回复的节点地址。

correlation-id *应用程序相关标识符*

这是一个特定于客户机的 ID，可用于标记或识别客户机之间的信息。

内容类型 *MIME 内容类型*

RFC-2046 MIME 类型，用于报文的应用程序数据部分（正文）。

根据 RFC-2046，这可能包含一个定义所用字符编码的 charset 参数：例如 "text/plain; charset="utf-8""。为清楚起见，真正不透明二进制部分的正确 MIME 类型是 application/octet-stream。

在使用具有除*数据*以外的部分代码的应用程序数据部分时，如果设置了 content- 类型，则应将其设置为 message/x-amqp+? 的 MIME 类型，其中的"? "表示数据、映射或列表。

内容编码 *MIME 内容类型*

Content-Encoding 属性用作内容类型的修饰符。当该属性存在时，其值表示应用数据应用了哪些额外的内容编码，因此必须应用哪些解码机制才能获得内容类型标头字段所引用的媒体类型。

内容编码（Content-Encoding）主要用于对文档进行压缩而不丢失其基本内容类型的特性。

内容编码应根据 RFC 2616 第 3.5 节进行解释。有效的内容编码在 IANA 注册为 "超文本传输协议（HTTP）参数" (http://www.iana.org/assignments/http-parameters/http- parameters.xml)。

当应用程序数据部分不是"...... "时，不得设置 Content-Encoding。

*数据*

实现不得使用*身份*编码。相反，实现不应设置此属性。实施不应使用*压缩编码*，除非是为了与最初使用其他协议（如 HTTP 或 SMTP）发送的报文保持兼容。

实施不应指定多个内容编码值，除非是为了与最初使用其他协议（如 HTTP 或 SMTP）发送的信息兼容。

绝对*过期*时间（absolute-expiry*-*time） *该信息被视为过期的时间*

该信息被视为过期的绝对时间。

创建时间 *该信息的创建时间*

创建此信息的绝对时间。

group-id *该信息所属的组*

标识信息所属的组。

组序列 *该组中该报文的序列号*

该信息在其组中的相对位置。

reply-to-group-id *回复信息所属的组*

这是一个特定于客户端的 ID，用于客户端向特定组发送对该信息的回复。

## 应用特性

<type name="application-properties" class="restricted" source="map" provides="section">

<descriptor name="amqp:application-properties:map" code="0x00000000:0x00000074"/>

</类型

应用属性部分是裸报文的一部分，用于结构化应用数据。中间人可使用此结构中的数据进行过滤或路由选择。

该映射的键被限制为[字符串](#_bookmark28)类型（排除了键为空的可能性），值被限制为简单类型，即（不包括[映射](#_bookmark32)、[列表](#_bookmark31)和[数组](#_bookmark34)类型）。

## 数据

<type name="data" class="restricted" source="binary" provides="section">

<descriptor name="amqp:data:binary" code="0x00000000:0x00000075"/>

</类型

数据部分包含不透明的二进制数据。

## AMQP 序列

<type name="amqp-sequence" class="restricted" source="list" provides="section">

<descriptor name="amqp:amqp-sequence:list" code="0x00000000:0x00000076"/>

</类型

序列部分包含任意数量的结构化数据元素。

## AMQP 值

<type name="amqp-value" class="restricted" source="\*" provides="section">

<descriptor name="amqp:amqp-value:\*" code="0x00000000:0x00000077"/>

</类型

一个 amqp-value 部分包含一个 AMQP 值。

## 页脚

传送信息的页脚。

<type name="footer" class="restricted" source="annotations" provides="section">

<descriptor name="amqp:footer:map" code="0x00000000:0x00000078"/>

</类型

页脚部分用于放置有关信息或传送的详细信息，这些信息或传送只有在构建或查看整个裸信息后才能计算或评估（例如信息哈希值、HMAC、签名和加密详细信息）。

已定义页脚及其含义的登记表可在此处找到[： http://www.amqp.org/specification/ 1.0/footer](http://www.amqp.org/specification/1.0/footer)。

## 注释

<type name="annotations" class="restricted" source="map"/>

*注释*类型是一个映射，其中的键被限制为[符号](#_bookmark29)类型或 [ulong](#_bookmark13) 类型。所有 ulong 键和以 "x-"开头的符号键均被保留。AMQP 容器在收到包含它无法识别的键或值的注释映射时，如果键不是以字符串 "x-opt-"开头，则必须使用未实现的 amqp-error 来分离链接。

## 信息 ID ULong

<type name="message-id-ulong" class="restricted" source="ulong" provides="message-id"/>

* + 1. **信息 ID UUID**

<type name="message-id-uuid" class="restricted" source="uuid" provides="message-id"/>

* + 1. **信息 ID 二进制**

<type name="message-id-binary" class="restricted" source="binary" provides="message-id"/>

* + 1. **信息 ID 字符串**

<type name="message-id-string" class="restricted" source="string" provides="message-id"/>

* + 1. **地址字符串**

节点地址。

<type name="address-string" class="restricted" source="string" provides="地址"/>类型

* + 1. **信息格式**

MESSAGE-FORMAT：本文件定义的报文格式+修订版 0

在传输本文所定义格式的报文时，该值将进入传输帧的报文格式字段。

* 1. **配送节点**
     1. **留言国**

报文传送层为存储在*分发节点*的报文定义了一组状态。并非所有节点都存储信息用于分发，但这些定义允许与存储信息的节点进行一些标准化交互。这些状态之间的转换受控于信息向/从分发节点的传输以及由此产生的终端交付状态。请注意，信息在一个分发节点的状态不会影响同一信息在另一个节点的状态。

默认情况下，报文将从 AVAILABLE（可用）状态开始。在启动*获取*传输之前，报文将过渡到 ACQUIRED（获取）状态。一旦进入 ACQUIRED（获取）状态，信息就没有资格向任何其他链接进行*获取传输。*

报文将在分发节点保持 ACQUIRED 状态，直到传输结算。接收方的传送状态决定了传送结算时报文的转换方式。如果接收方的传送状态尚不可知（例如，链路端点在恢复之前被摧毁），则使用源的*默认结果。*

状态转换也可能在分发节点自发发生。例如，如果带 ttl 的报文过期，过期的影响可能是（取决于特定类型和配置的

分发节点）自发地从 AVAILABLE 状态进入 ARCHIVED 状态。在这种情况下，无论接收方处于何种状态，信息的任何传输都会过渡到分发节点的终端结果。

+ +

+->| 可用 |

| + +

| |

| |

最终结果： | |

发布/修改 | TRANSFER（获取）

| |

| |

| \|/

| + +

+--| 获得

+ +

|

|

|终端结果：

| 接受/拒绝

|

|

\|/

+ +

| 存档

+ +

图 3.1：报文状态转换

* 1. **交货状态**

报文传送层定义了一组具体的传送状态，这些状态可（通过[处置](#_bookmark80)帧）用于指示接收方的报文状态。传送状态可以是终端状态，也可以是非终端状态。一旦递送达到终端递送状态，该递送的状态将不再改变。终端传送状态被称为*结果*。

以下结果由报文传送层正式定义，表示接收方的处理结果：

* [接受](#_bookmark133)：表示接收方处理成功
* [rejected（已拒绝](#_bookmark134)）：表示报文无效且无法处理
* 已[释放](#_bookmark135)：表示未处理（也不会处理）信息
* [已](#_bookmark136)修改：表示电文已被修改，但未被处理

以下非终端传送状态由报文层正式定义，用于链路恢复期间，允许发送方在不重传所有报文数据的情况下恢复大容量报文的传送：

* [收到](#_bookmark132)：表示接收者看到的部分信息数据以及恢复传输的起点
  + 1. **已收到**

<type name="received" class="composite" source="list" provides="delivery-state">

<descriptor name="amqp:received:list" code="0x00000000:0x00000023"/>

<field name="section-number" type="uint" mandatory="true"/>

<field name="section-offset" type="ulong" mandatory="true"/>

</类型

在目标端，[接收](#_bookmark132)状态表示如果链路恢复，目标端无需重新发送的报文有效载荷的最远点。在源端，[接收](#_bookmark132)状态表示发送方在链路恢复时能够恢复传输的有效载荷的最早点。在恢复传送时，如果此状态在第一次[传输](#_bookmark79)执行时被设置，则表示第一次恢复传送开始的有效载荷中的偏移量。发送方不得在[传输](#_bookmark79)或[处置](#_bookmark80)执行符上发送[接收](#_bookmark132)状态，除非是在恢复传送的第一个[传输](#_bookmark79)执行符上。

**实地详情**

节号

由发件人发送时，这表示可以重新发送数据的第一个报文部分（部分编号 0 为第一个部分）。给定部分之前各部分的数据不能重新传输。

由接收方发送时，表示可能尚未收到所有数据的信息的第一部分。

段落偏移

发送端发送时，表示可以重新发送数据的段落编号所给段落编码数据的第一个字节（段落偏移 0 为第一个字节）。在给定偏移段之前的同一段中的字节不能重新传输。

由接收方发送时，表示尚未收到的给定部分的第一个字节。需要注意的是，如果接收方已经接收了 X 节的全部数据（包含 N 个字节的数据），但没有接收 X+1 节的数据，那么接收方可以发送 Received(section-number=X, section-offset=N) 或 Received(section-number=X+1, section-offset=0)来表示。Received(section- number=0, section-offset=0)状态表示没有传送任何报文数据。

* + 1. **已接受**

公认的结果。

<type name="accepted" class="composite" source="list" provides="delivery-state, outcome">

<descriptor name="amqp:accepted:list" code="0x00000000:0x00000024"/>

</类型

在源端，接受状态意味着报文已从节点退役，如果链路暂停，将无法恢复有效载荷数据的传输。即使在所有传输帧都已发送之前，源端也可能已接受传输，但这并不意味着将不发送传输的剩余传输--只有[传输](#_bookmark79)执行器上的中止标志才可用于指示传输的提前终止。

在目标端，接受结果用于表示传入的报文已被成功处理，报文的接收方期待发送方将传送转换为源端接受状态。

已接受的结果不会增加已接受报文标头中的*传送次数。*

* + 1. **拒绝**

被拒绝的结果。

<type name="rejected" class="composite" source="list" provides="delivery-state, outcome">

<descriptor name="amqp:rejected:list" code="0x00000000:0x00000025"/>

<field name="error" type="error"/>

</类型

在目标处，拒收结果用于表示收到的报文无效，因此无法处理。当拒收结果应用到报文时，会导致被拒收报文头部的*传送次数*递增。

在源端，拒收结果意味着目标已通知源端该信息被拒收，并且源端已采取必要的措施。在源端，传送不应自发达到被拒状态。

**实地详情**

*导致拒收邮件的错误*

在此字段中提供的值将放在与符号键 "rejected "相关联的被拒收报文的递送注释中。

* + 1. **已发布**

发布的结果。

<type name="released" class="composite" source="list" provides="delivery-state, outcome">

<descriptor name="amqp:released:list" code="0x00000000:0x00000026"/>

</类型

在源端，发布结果意味着接收方不再获取报文，并可将报文（重新）发送给从该节点接收的相同或其他目标。在节点处，报文保持不变（即已发布报文头部的*发送计数*不得递增）。由于释放是一种终端结果，因此如果链路暂停，有效载荷数据的传输将无法恢复。即使在所有传输帧都已发送之前，源也可能会释放一个传输，但这并不意味着将不发送该传输的剩余传输。信源可能会自发获得信息的已释放结果（例如，信源可能会实施某种有时间限制的获取锁定，在锁定后，节点对信息的获取将被撤销，以便将信息传送给其他消费者）。

在目标机上，释放的结果用于表示没有也不会对某项转移采取行动。

* + 1. **改装**

修改后的结果。

<type name="modified" class="composite" source="list" provides="delivery-state, outcome">

<descriptor name="amqp:modified:list" code="0x00000000:0x00000027"/>

<field name="delivery-failed" type="boolean"/>

<field name="undeliverable-here" type="boolean"/>

<field name="message-annotations" type="fields"/>

</类型

在信源处，修改后的结果意味着接收方不再获取该信息，并可将其发送（重新）给从该节点接收信息的同一或其他目标。在节点处，电文已按结果字段所示方式进行了修改。由于修改是终端结果，因此如果链路出现以下情况，将无法恢复有效载荷数据的传输

暂停。即使在所有传输帧都已发送之前，源端也可能修改了传输，但这并不意味着将不发送该传输的剩余传输。信源可自发实现修改后的报文结果（例如，信源可实施某种有时间限制的获取锁定，在锁定后，节点对报文的获取将被取消，以便将报文交付给其他消费者，并以某种方式修改报文，以表示之前的失败，如将交付失败设为 true）。

在目标处，修改后的结果用于表示某项传输没有也不会被执行，并表示节点应按指定方式修改报文。

**实地详情**

传送失败 *将转账视为一次不成功的交付尝试*

如果设置了 "发送失败 "标志，任何被修改的信息都必须增加其发送次数。

此处无法投递 *防止重新投递*

如果设置了 undeliverable-here（无法投递-此处），则发布的任何信息都不得重新投递到修改链接端点。

消息注释 *消息属性*

包含属性的映射，与信息标题部分中的现有*信息注释*相结合。如果信息的现有信息注释中包含与该字段中的条目具有相同关键字的条目，则该字段中与该关键字相关的值将取代现有标头中的值；如果现有信息注释中没有该值，则将添加该映射中的值。

* + 1. **使用交付状态恢复交付**

[2.6.13 恢复传送](#_bookmark72)中提供了链路恢复后两个端点应如何重建状态的一般方案。通过上文定义的具体交付状态，可以更全面地举例说明链路恢复。

同行 合作伙伴

=======================================================================

ATTACH(name=N, handle=1, --++-- ATTACH(name=N, handle=2, role=sender、 \ / role=receiver、

source=X, \ / source=X、

target=Y、 x target=Y

unsettled=/\ unsettled=

{ 1 -> 空、 / \{ 2 -> Received(3,0)、

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 -> | | 无效、 | <-+ | +-> | 3 | -> | 已接受、 |
| 3 -> | | 无效、 |  |  | 4 | -> | 无效、 |
| 4 | -> | 无效、 | | 6 | | -> | 收到（2,0）、 |
| 5 | -> | 收到（0,200）、 | | 7 | | -> | 收到（0,100）、 |
| 6 | -> | 收到(1,150)、 | | 8 | | -> | 已接受、 |
| 7 | -> | 收到（0,500）、 | | 9 | | -> | 无效、 |
| 8 | -> | 收到（3,5）、 | | 11 | | -> | 收到(1,2000)、 |
| 9 | -> | 收到（2,0）、 | | 12 | | -> | 已接受、 |
| 10 | -> | 已接受、 | | 13 | | -> | 已发布、 |
| 11 | -> | 已接受、 | | 14 | | -> | null } |
| 12 | -> | 已接受、 | |  | |  |  |
| 13 | -> | 已接受、 | |  | |  |  |
| 14 | -> | 接受 } | |  | |  |  |

钥匙

Received(x,y) 指 Received(节号=x, 节偏移=y)

在本例中，对于包含 1 至 4 的发送标记，发件人表示可以从信息的起点开始继续发送。

对于递送标记 1，接收方没有递送记录。为了保证 "至少一次 "或 "最多一次 "的递送，发送方必须重新发送信息，但递送并没有恢复（因为接收方不记得递送标记），因此传输时不得将恢复标记设为 true。如果发送方将传输标记为恢复，那么接收方将忽略这些传输。

对于发送标记 2，接收方保留了构成信息的部分数据，但不是全部数据。为了完成发送，发送方必须从接收方期待的下一个位置或之前的某个位置继续发送。

TRANSFER(delivery-id=1 ,----------> \*\* 将未 \* \* 的 邮件数据\*\* delivery-tag=2, \*\* 添加 到递送中 \*\*

(1)state=Received(3,0), state. \*\* resume=true)

{ \*\* 有效载荷 \*\* }

* + - 1. 状态可能是
         1. 空，这意味着传输从第 0 段的第一个字节开始恢复（接收器必须忽略其之前未接收到的第一个位置之前的所有数据）。
         2. 接收时带有段号 0、1 或 2 和偏移量，表示恢复发送的第一帧有效载荷数据从给定的点开始，接收机必须忽略之前未接收到的第一个位置之前的所有数据。
         3. Received(3,0)表示从接收方之前没有接收到的第一个点开始恢复。

对于递送标记 3，接收方表示它已将递送处理到需要终端结果（在本例中为[接受](#_bookmark133)）的地步。在这种情况下，发件人要么在源端应用该结果，要么在无法应用该结果的罕见情况下，指出已应用的终端结果。为此，发件人必须发送恢复传输，将交付标记 3 与交付标识关联起来。在此传送中，发件人应在源端设置传送状态。如果这与接收方的结果相同，则发送方也可以发送已解决的恢复转移。

TRANSFER(delivery-id=2, ----------> \*\* 处理 \*\* 交货标记=3, \*\* 已被接受的确认，并结算。\*\* 结算=true、

more=false, state=Accepted、

resume=true)

对于递送标记 4，接收方表示它知道递送已经开始，但没有提供任何迹象表明除了递送存在这一事实外，它还保留了有关递送的任何数据。为了保证 "至少一次 "或 "最多一次 "的传送，发送方必须重新发送整个信息。与递送标记 1 的情况不同，重新发送的递送必须将恢复标记设为 true，并将递送标记设为 4（虽然在接收者映射中使用 null 是有效的，但我们不鼓励这样做。建议接收方不要在其映射中保留这样的条目，在这种情况下，情况与本例中的递送标记 1 相同）。

TRANSFER(delivery-id=3 ,----------> \*\* 处理方式与 \* \* delivery-tag=4, \*\* 一样，我们将对非 \*\*

(1)state=null, resumed delivery. \*\* resume=true)

{ \*\* 有效载荷 \*\* }

(1) 或者（也可以等价），状态可以是 Received（段落编号=0，段落偏移=0）

对于发送标记 5 至 9（含 9），发送方表示可以在信息数据开始后的某一点恢复发送。这通常表明，接收方先前已确认接收到指定点的信息数据，因此发送方没有责任在恢复链接后保留重新发送该数据的能力。发送方仍可保留以新的传送方式重新发送信息的能力（即发送方可能尚未完全丢弃生成原始传送的数据）。

对于递送标记 5，接收方没有递送记录。为了保证 "至少一次 "或 "最多一次 "的递送，发送方必须重新发送信息，但递送并没有恢复（因为接收方不记得递送标记），因此传输时不得将恢复标记设为 true。如果发送方没有足够的数据来重新发送信息，那么发送方可以采取一些措施来表明它认为有可能发生了信息丢失。

对于发送标记 6，接收方保留了组成报文的部分数据，但不是全部数据。接收方没有收到的信息的第一个位置是在发送方可以继续发送的第一个位置之后。为了完成发送，发送方必须在接收方期待的下一个位置之前的某个位置继续发送。

TRANSFER(delivery-id=4 ,----------> \*\* 将未 \* \* 的 邮件数据\*\* delivery-tag=6, \*\* 添加 到递送中 \*\*

(1)state=Received(2,0), state. \*\* resume=true)

{ \*\* 有效载荷 \*\* }

(1) 状态可以是 Received(1,150)和 Received(2,0)之间的任何一点。接收器必须忽略所有数据，直到它之前没有接收到的第一个位置（即第 2 节偏移 0）。

对于发送标记 7，接收方保留了组成报文的部分数据，但不是全部数据。接收方未收到的电文的第一个位置是在发送方可以继续发送的第一个位置之前。因此，发件人不可能继续发送完信息。发件人唯一的选择是中止传送，然后（如果可能）以新的传送方式重新发送，或者在无法中止传送的情况下以某种方式报告可能的信息丢失。

TRANSFER(delivery-id=5, ----------> \*\* 丢弃与 \*\* delivery-tag=7, \*\* 有关的任何状态。\*\* resume=true、

aborted=true)

对于交货标记 8，接收方表示已将交货处理到它希望得到终端结果（在本例中为[接受](#_bookmark133)）的程度。这与交货标记 3 的情况相同。

TRANSFER(delivery-id=6, ----------> \*\* 处理 \*\* 交货标记=8, \*\* 已被接受的确认，并结算。\*\* 结算=true、

more=false, state=Accepted、

resume=true)

对于交货标记 9，接收方表示它知道交货已经开始，但没有提供任何迹象表明，除了交货存在这一事实外，它保留了有关交货的任何数据。这与交货标记 7 的情况相同。

TRANSFER(delivery-id=7, ----------> \*\* 丢弃任何与 \*\* delivery-tag=9, \*\* 有关的状态。\*\* resume=true、

aborted=true)

对于递送标记 10 至 14（含 14），发件人表示已达到终端结果，即已[接受](#_bookmark133)。发件人一旦到达终端结果，就不能再改变。因此，如果发件人有能力恢复发送（即使发送的唯一可能结果是预定义的终端结果，如已[接受](#_bookmark133)），在确定接收方不需要重新发送信息数据之前，发件人不得将此状态作为其未结算映射中的状态值。

对于递送标记 10，接收方没有递送记录。然而，与递送标记 1 和递送标记 5 的情况不同的是，由于我们知道发件人只有在知道收件人已经收到了整个信息（或者发件人已经自发地达到了不可能恢复的终端结果）的情况下才会达到这种状态，因此我们没有必要重新发送信息。

对于递送标记 11，我们必须假设发件人自发达到了终端结果（无法恢复）。在这种情况下，发件人可以直接中止投递，因为投递无法恢复。

TRANSFER(delivery-id=8, ----------> \*\* 丢弃与 \*\* delivery-tag=11, \*\* 有关的任何状态。\*\* resume=true、

aborted=true)

对于交货标记 12，发件人和收件人对终端结果的看法一致，但都没有结算。在这种情况下，发件人应直接结算交货。

TRANSFER(delivery-id=9 ,----------> \*\* 本地结算交货 \*\* delivery-tag=12、

结算=true, 恢复=true)

对于交付标记 13，发送方和接收方都获得了终端结果，但结果不同。在这种情况下，由于结果在发件人处实际生效，因此发件人的观点才是确定的。因此，发件人必须重申这是最终结果，而收件人也应予以回应，并达成和解。

TRANSFER(delivery-id=10----------> \*\* 更新 受实际结果影响 的任何状态\*\* delivery-tag=13, \*\* 然后 \*\*

结算 =false, \*\* 结算交货 \* \* state=Accepted

resume=true)

<---------- DISPOSITION(first=10, last=10、

state=Accepted, settled=true)

对于发送标签 14，情况与发送标签 11 基本相同，因为接收器的空状态与接收节号=0、节偏移=0 的状态基本相同。

TRANSFER(delivery-id=11, ----------> \*\* 丢弃与 \*\* delivery-tag=14, \*\* 有关的任何状态。\*\* resume=true、

aborted=true)

* 1. **来源和目标**

消息层定义了两种具体类型（[源](#_bookmark141)和[目标](#_bookmark142)），用作链接的*源*和*目标*。在建立或恢复链接时，这些类型将在[附加](#_bookmark77)帧的*源*和*目标*字段中提供。[源](#_bookmark141)由一个地址（传出链接端点的容器将把该地址解析为该容器中的一个节点）和确定以下内容的属性组成：

* 发送节点的报文将在链路上发送、
* 发送信息如何影响发送节点的信息状态、
* 当信息源被销毁时，已在链路上传输但尚未到达接收端终端状态的信息的行为。
  + 1. **过滤信息**

信息源可以通过指定*过滤器*来限制从信息源传输的信息。过滤器可被视为将信息作为输入并返回布尔值的函数：如果信息源接受该信息，则返回 true，否则返回 false。除非节点上信息的状态或注释发生变化（如通过更新传送状态），否则*过滤器*不得更改信息的返回值。

* + 1. **配送模式**

源 "定义了一种可选的 "分发模式"（distribution-mode），用于通知和/或指示分发节点对链接的行为方式。源的分发模式决定了如何在其关联的链接中分发来自分发节点的信息。有两种已定义的分发模式：*移动*和*复制*。指定分发模式后，分发节点相对于与 "源 "相关联的 "链路 "的行为会受到两种相关影响。

*移动*分发模式会使从分发节点传送的报文在链路上传送前过渡到 "获取 "状态，并在传送成功后过渡到 "存档 "状态。*复制*分发模式则使报文在分发节点的状态保持不变。

信源不得重新发送先前已从信源成功传送的报文，即在接收方达到 ACCEPTED（接受）传送状态的报文。对于默认配置的*移动*链路来说，这一点很容易实现，因为这样做的最终结果将导致节点上的报文处于存档状态，因此无论如何都没有资格传输。对于*复制*链接，状态必须保留在

以确保符合要求。在实践中，对于在节点上保持严格的报文顺序的节点来说，状态可能只是最近传输的报文的记录。

* + 1. **资料来源**

<type name="source" class="composite" source="list" provides="source">

<descriptor name="amqp:source:list" code="0x00000000:0x00000028"/>

<field name="address" type="\*" requires="address"/>

<field name="durable" type="terminus-durability" default="none"/>

<field name="expiry-policy" type="terminus-expiry-policy" default="session-end"/>

<field name="timeout" type="seconds" default="0"/>

<field name="dynamic" type="boolean" default="false"/>

<field name="dynamic-node-properties" type="node-properties"/> 动态节点属性

<field name="distribution-mode" type="symbol" requires="distribution-mode"/> 分配模式

<field name="filter" type="filter-set"/>

<field name="default-outcome" type="\*" requires="outcome"/>

<field name="outcomes" type="symbol" multiple="true"/>

<field name="capabilities" type="symbol" multiple="true"/>

</类型

对于不执行地址解析（也不接受来自其伙伴的自发链路连接）而只用作报文生产者的容器，没有必要提供虚假的源细节。为此，可以使用 "最小 "源，其中所有字段都不设置。

**实地详情**

地址 *源地址*

在动态标志设置为 true（即接收方要求发送方创建一个可寻址节点）的情况下，接收链路端点发送的[附加](#_bookmark77)帧上不得设置源地址。

源地址必须在动态标志设置为 true（即发送方应接收方请求创建了一个可寻址节点，并正在通信该创建节点的地址）的发送链接端点发送的[附件](#_bookmark77)帧上发送时设置。生成的地址名称应包括链接名称和远程容器的容器 ID，以便于识别。

耐用 *表示总站的耐用性*

表示将持久保留总站的哪种状态：持久报文状态、仅存在和总站配置，或根本没有状态。

到期政策 *源的过期策略*

参见[第 3.5.6 小节 "终点站失效政策](#_bookmark144)"。

超时 *将过期的源保留的持续时间*

源会按照过期政策的指示开始过期。

动态 *请求动态创建远程节点*

当接收链路端点设置为 true 时，该字段将构成发送对等点在源点动态创建节点的请求。在这种情况下，地址字段不得设置。

当发送链路端点将该字段设置为 true 时，表示创建了一个动态创建的节点。在这种情况下，地址字段将包含所创建节点的地址。生成的地址应包括链路名称、会话名称或客户机标识（Client-ID），以便于跟踪。

dynamic-node-properties *动态创建节点的属性*

如果动态字段未设置为 true，则该字段必须保持未设置状态。

由接收链路端点设置时，该字段包含接收方希望创建的节点的预期属性。由发送链接端点设置时，该字段包含动态创建节点的实际属性。请参阅 [3.5.9](#_bookmark147) 节[点属性小节](#_bookmark147)。

分配模式 *Link 的分配模式*

如果端点支持一种以上的分发模式，该字段必须由链路的发送端设置。当节点支持多种分发模式时，该字段可由链路的接收端设置，以表示偏好。

过滤 *一组谓词，用于过滤进入链接的信息*

请参阅 [3.5.8 过滤器设置小节](#_bookmark146)。接收端点设置其所需的过滤器，发送端点设置实际存在的过滤器（包括节点默认的任何过滤器）。接收端点必须检查现有过滤器是否满足其需要，如果不满足，则负责分离。

*未结算转账的默认结果*

表示转账结算时（包括源被销毁时）接收方尚未达到终端状态的转账所使用的结果。该值必须是有效的结果（如[已释放](#_bookmark135)或已[拒绝](#_bookmark134)）。

成果 *可在此链接上选择的成果描述*

该字段中的值是可在此链接上选择的结果的符号描述符。该字段可以为空，表示所有报文传输都将使用*默认结果*（如果未设置默认结果，也未提供任何结果，则所[接受的](#_bookmark133)结果必须得到源的支持）。

出现时，值必须是有效结果的符号描述符，如 "amqp:accepted:list"。

功能 *发件人支持/需要的扩展功能*

见 http://www.amqp.org/specification/1.0/source-capabilities。

* + 1. **目标**

<type name="target" class="composite" source="list" provides="target">

<descriptor name="amqp:target:list" code="0x00000000:0x00000029"/>

<field name="address" type="\*" requires="address"/>

<field name="durable" type="terminus-durability" default="none"/>

<field name="expiry-policy" type="terminus-expiry-policy" default="session-end"/>

<field name="timeout" type="seconds" default="0"/>

<field name="dynamic" type="boolean" default="false"/>

<field name="dynamic-node-properties" type="node-properties"/> 动态节点属性

<field name="capabilities" type="symbol" multiple="true"/>

</类型

对于不执行地址解析（也不接受来自其伙伴的自发链接附加）而只用作消息消费者的容器，没有必要提供虚假的源细节。为此，可以使用一个 "最小 "目标，其中所有字段都不设置。

**实地详情**

地址 *目标地址。*

在动态标志设置为 "true"（即发送方请求接收方创建可寻址节点）的情况下，在发送链路端点发送的[附件](#_bookmark77)帧上发送时，不得设置目标地址。

当接收链接端点发送的[附件](#_bookmark77)帧中动态标志设置为 true 时（即接收方应发送方请求创建了一个可寻址节点，并正在通信该创建节点的地址），必须设置源地址。生成的地址名称应包括链路名称和远程容器的容器 ID，以便于识别。

耐用 *表示总站的耐用性*

表示总站的哪种状态将被持久保留：持久报文状态、仅存在和总站配置，或完全不保留状态。

到期政策 *目标的到期政策*

参见[第 3.5.6 小节 "终点站失效政策](#_bookmark144)"。

超时 *过期目标的保留时间*

目标根据到期政策开始到期。

动态 *请求动态创建远程节点*

当发送链接端点设置为 true 时，该字段将构成接收对等点在目标处动态创建节点的请求。在这种情况下，不得设置地址字段。

当接收链路端点将该字段设置为 true 时，表示创建了一个动态创建的节点。在这种情况下，地址字段将包含创建节点的地址。生成的地址应包括链路名称、会话名称或客户标识（以某种可识别的形式），以便于跟踪。

dynamic-node-properties *动态创建节点的属性*

如果动态字段未设置为 true，则该字段必须保持未设置状态。

由发送链路端点设置时，该字段包含发送方希望创建的节点的预期属性。由接收链接端点设置时，该字段包含动态创建节点的实际属性。请参见 [3.5.9](#_bookmark147) 节[点属性小节](#_bookmark147)。

功能 *发件人支持/需要的扩展功能*

见 http://www.amqp.org/specification/1.0/target-capabilities。

* + 1. **终点站耐用性**

总站的耐久性政策。

<type name="terminus-durability" class="restricted" source="uint">

<选择项 name="none" value="0"/>

<choice name="configuration" value="1"/>

<选择项 name="unsettled-state" value="2"/>

</类型

确定 Terminus 的哪种状态为持久值。

**有效值**

1. 不持久保留总站状态。
2. 只有总站的存在和配置才得以永久保留。
3. 除了总站的存在和配置外，持久信息的未定状态也会持久保留。
   * 1. **总站到期政策**

总站的到期政策。

<type name="terminus-expiry-policy" class="restricted" source="symbol">

<choice name="link-detach" value="link-detach"/>

<choice name="session-end" value="session-end"/>

<选择 name="connection-close" value="connection-close"/>

<选择项 name="never" value="never"/>

</类型

确定 Terminus 的到期计时器何时开始从超时值开始倒计时。如果链路随后在 Terminus 过期前重新连接，则倒计时终止。如果随后重新满足 terminus-expiry-policy 的条件，到期计时器将从最初配置的超时值重新开始计时。

**有效值**

**链接-脱离** 到期计时器从 Terminus 分离时开始计时。

**会话结束** 过期计时器从最近关联的会话结束时开始计时。

**连接关闭** 过期计时器从最近关联的连接关闭时开始计时。

**永不** 终点站永不失效。

* + 1. **标准配送模式**

链接分发政策。

<type name="std-dist-mode" class="restricted" source="symbol" provides="distribution-mode">

<选择项 name="move" value="move"/>

<choice name="copy" value="copy"/>

</类型

多个链接连接到同一节点时的信息分发策略。

**有效值**

**移动** 一旦在链路上成功传输，同一节点的其他链路将不再可以使用该报文

**复制** 一旦在链路上成功传输，信息仍可用于同一节点的其他链路

* + 1. **过滤器套件**

<type name="filter-set" class="restricted" source="map"/>

一组已命名的过滤器。映射表中的每个键都必须是[符号](#_bookmark29)类型，每个值都必须是[空值](#_bookmark8)或提供原型*过滤器*的描述类型。过滤器作为信息的函数，会返回一个布尔结果，表示信息是否可以通过该过滤器。当且仅当一条信息通过每个指定的过滤器时，它才会通过过滤器集。如果给定密钥的值为空，则表示没有该密钥（即所有信息都通过空过滤器）。

过滤器类型是一个已定义的扩展点。特定[信号源](#_bookmark141)支持的过滤类型将由[信号源](#_bookmark141)的能力来表示。常用的过滤器类型以及与之相关的功能在此注册[：](http://www.amqp.org/specification/1.0/filters)http://www.amqp.org/specification/1.0/filters。

* + 1. **节点属性**

节点的属性

<type name="node-properties" class="restricted" source="fields"/> 类型名称="节点属性" class="restricted" 源代码="字段"/>

包含节点属性的符号键映射--用于请求创建或报告动态节点的创建。

定义了以下常用属性：

**生命周期策略** 动态生成节点的生命周期。

从定义上讲，生命周期绝不会小于创建节点的链路的生命周期，但可以使用生命周期策略来延长动态创建节点的生命周期。 此条目值必须为

定义的*生命周期策略*之一：[关闭时删除](#_bookmark148)、[无链接时](#_bookmark149)删除、[无信息](#_bookmark150)时删除或[无链接或信息时删除](#_bookmark151)。

**支持的分集模式**

节点支持的分发模式。

该条目值必须是一个或多个有效的*分布模式*符号。也就是说，该值必须与用以下属性定义的字段中的有效类型相同：

*type="symbol" multiple="true" requires="distribution-mode"*

* + 1. **关闭时删除**

动态节点的生命周期与导致创建的链路的生命周期一致。

<type name="delete-on-close" class="composite" source="list" provides="lifetime-policy">

<descriptor name="amqp:delete-on-close:list" code="0x00000000:0x0000002b"/>

</类型

使用此生命周期策略动态创建的节点将在导致其创建的链接不复存在时被删除。

* + 1. **无链接时删除**

动态节点的生命周期，与节点链接的存在相关。

<type name="delete-on-no-links" class="composite" source="list" provides="lifetime-policy">

<descriptor name="amqp:delete-on-no-links:list" code="0x00000000:0x0000002c"/>

</类型

使用此生命周期策略动态创建的节点，将在没有该节点作为源节点或目标节点的链接时被删除。

* + 1. **无信息时删除**

动态节点的生命周期，适用于节点上存在的报文。

<type name="delete-on-no-messages" class="composite" source="list" provides="lifetime-policy">

<descriptor name="amqp:delete-on-no-messages:list" code="0x00000000:0x0000002d"/>

</类型

使用此生命周期策略动态创建的节点将在导致其创建的链接不复存在且节点上没有任何信息时被删除。

* + 1. **在无链接或信息时删除**

节点的生命周期，适用于节点上存在的报文或链接。

<type name="delete-on-no-links-or-messages" class="composite" source="list" provides="lifetime-policy">

<descriptor name="amqp:delete-on-no-links-or-messages:list" code="0x00000000:0x0000002e"/>

</类型

使用此生命周期策略动态创建的节点将在没有将该节点作为源节点或目标节点的链接时被删除，节点上也没有任何消息。

**第四册**

**交易**

* 1. **事务性信息传递**

事务性报文传送可使原本独立的传输取得协调的结果。这可以扩展到任意数量的传输，这些传输分布在任一方向的任意数量的不同链接上。

在每次事务交互中，一个容器充当*事务资源*，另一个容器充当*事务控制器*。*事务资源*按照*事务控制器*的要求执行*事务性工作*。

*事务控制器*和*事务资源*通过*事务控制器*建立的*控制链路*进行通信。*事务控制器*通过*控制链路*发送[申报](#_bookmark166)和[放行](#_bookmark167)信息，分别用于分配和完成事务（它们并不代表事务性工作的分界线）。*控制链路*上不允许有事务性工作。请求的每个事务性操作都用所需的事务 ID 明确标识，因此可以在控制会话内的任何链路上进行，如果控制器的能力允许，也可以在控制连接上的任何链路上进行。如果在控制*链接*关闭时，还存在其创建的未解除事务，那么所有这些事务都会立即回滚，试图在这些事务上执行进一步的事务工作将导致失败。

* 1. **申报交易**

作为事务资源的容器会定义一个特殊的目标，作为[协调器](#_bookmark165)发挥作用。*事务控制器*会与该目标建立控制链接。请注意，与[协调器](#_bookmark165)的链接不能恢复。

要开始事务性工作，事务控制器必须从资源中获取事务标识符。为此，它要向[协调器](#_bookmark165)发送一条信息，信息主体由单个 [amqp-value](#_bookmark120) 部分中的[声明](#_bookmark166)类型组成。其他标准信息部分（如标题部分）应忽略。由于发送方需要接收并解释接收方的声明结果，因此该报文不得发送完毕。如果协调器接收到发送方已结算的[传输](#_bookmark79)，则应带着适当的错误[脱离。](#_bookmark81)

如果声明成功，协调器将以[声明的](#_bookmark169)处置结果作为回应。

其中包含为交易分配的标识符。

如果协调器无法执行事务控制器指定的[声明](#_bookmark166)，事务协调器必须将该错误作为[事务错误](#_bookmark173)传达给控制器。如果

如果与[协调器](#_bookmark165)链接的源支持被拒绝的结果，则必须拒绝该信息，并在该结果中带有[事务错误](#_bookmark173)。如果信息源不支持被拒绝的结果，则*事务资源*必须分离与协调器的链接，分离执行符必须带有[事务错误](#_bookmark173)。

事务控制器应建立一个控制链接，允许被拒绝的结果。

事务控制器 事务资源

===============================================================================

ATTACH(name=txn-ctl、 >

...,

target= 协调员(

能力=

"amqp:本地-交易")

)

< ATTACH(name=txn-ctl、

...,

target= 协调员(

能力=

["amqp:local-transactions"、"amqp:multi-txns-per-ssn"]

)

)

<--------- FLOW(...,handle=1, link-credit=1)

TRANSFER(delivery-id=0, ...) >

{ AmqpValue( Declare() )}

<--------- DISPOSITION（first=0，last=0、

state=Declared(txn-id=0) )

图 4.1：声明事务

* 1. **解除交易**

控制器将通过向协调器发送[放电](#_bookmark167)报文（以单个 [amqp-value](#_bookmark120) 部分编码）来结束事务性工作。控制器通过在[放行](#_bookmark167)正文中设置*失败*标记来表明它希望提交或回滚事务性工作。与[声明](#_bookmark166)报文一样，如果发送者发送的是预结算[传输](#_bookmark79)，则属于错误。

如果协调器无法完成[卸载](#_bookmark167)，协调器必须将该错误作为[事务错误](#_bookmark173)传达给控制器。如果与[协调器](#_bookmark165)链接的[源](#_bookmark141)支持被[拒绝的](#_bookmark134)结果，则必须[拒绝](#_bookmark134)该消息，并在该结果中包含[事务错误](#_bookmark173)。如果[源](#_bookmark141)不支持[被拒绝的](#_bookmark134)结果，则*传输资源*必须[分离](#_bookmark81)与协调器的链接，[分离](#_bookmark81)执行符携带[事务](#_bookmark173)错误。请注意，协调器必须始终能够完成失败标志设为 true 的[排放](#_bookmark167)（因为协调器失败会导致回滚，而这正是控制器所要求的）。

事务控制器 事务资源

===============================================================================

TRANSFER(delivery-id=0, ...) >

{ AmqpValue( Declare() )}

<--------- DISPOSITION(first=0, last=0、

state=Declared(txn-id=0) )

:

事务性工作

:

TRANSFER(delivery-id=57, ...) >

{ AmqpValue(

放电（txn-id=0、

fail=false)

)}

<--------- DISPOSITION（first=57，last=57、

state=Accepted() )

图 4.2：解除交易

* 1. **事务性工作**

事务性工作用 [3.3.1 报文](#_bookmark130)状态中定义的报文[状态](#_bookmark130)来描述。事务性工作正式定义为由以下操作组成：

* 向目标发布信息，即*提供*信息
* 从信息源获取信息，即过渡到*获取信息*
* 在源头退信，即应用终端结果

当事务控制器触发时，事务资源会执行这些操作：

* *发布*信息由接收到的[传输](#_bookmark79)帧启动
* *获取*报文由接收到的[流量](#_bookmark78)帧启动
* 退信由接收到的[处置](#_bookmark80)帧启动

在每种情况下，事务控制器都有责任确定请求的工作所关联的事务。要做到这一点，需要使用事务交付状态（transactional-[state](#_bookmark170)），它将 txn-id 与报文传送规范[第 3.4 节](#_bookmark131) "[交付状态](#_bookmark131) "中定义的终端交付状态之一结合在一起。[传输](#_bookmark79)帧和[处置](#_bookmark80)帧都带有[事务](#_bookmark170)状态，使报文的*发布*和*退订*都与事务相关联。

[传输](#_bookmark79)、[处置](#_bookmark80)和[流量](#_bookmark78)帧可以任一方向传输，即从控制器到资源和从资源到控制器。当这些帧从控制器传送到资源时，任何嵌入的 txn-ids 都会请求资源将事务性工作分配给指定的事务。当从另一个方向，即从资源到控制器时，[传输](#_bookmark79)和[处置](#_bookmark80)帧表示已执行的工作，其中包含的 txn-ids 必须正确表示该工作与哪个事务相关（如果有的话）。对于从资源到控制器的[流量](#_bookmark78)帧，txn-id 并不表示已执行的工作，而是表示未来从该链路进行的传输将与哪个事务相关。

## 事务性发布

如果事务控制器希望将发出的传输与事务关联起来，就必须用携带适当事务标识符的事务状态来设置传输[状态](#_bookmark170)。需要注意的是，如果传输被分割成几个传输帧，那么所有传输帧都必须明确地与同一个事务相关联。控制器若试图解除已发布部分传送的事务，则属于错误。如果发生这种情况，控制链路必须以[事务回滚](#_bookmark174)错误终止。

事务发布的效果是，直到事务（成功）解除后，信息才会在事务资源内的目标节点可用。

事务控制器 事务资源

===============================================================================

TRANSFER(handle=0、 >

delivery-id=0、 )

{ AmqpValue( Declare() )}

<--------- DISPOSITION(first=0, last=0、

state=Declared(txn-id=0) )

TRANSFER(handle=1、 >

delivery-id=1, state=

TransactionalState( txn-id=0) )

{ ... 有效载荷 }

<--------- DISPOSITION(first=1, last=1、

state=TransactionalState( txn-id=0, outcome=Accepted())

)

图 4.3：事务发布

在收到与实时交易相关联的非结算交付时，交易资源必须将推定的终端结果通知控 制器，然后才能成功解除交易。也就是说，资源必须发送一个[处置](#_bookmark80)执行符，该执行符涵盖已发布的转账，交付状态为[交易状态](#_bookmark170)，已识别正确的交易和终端结果。这就告知了控制者在交易成功解除时的结果。

## 事务性退休

交易控制器可能希望将交付结果与交易联系起来。交付本身不需要与结果关联到同一交易，或者与任何交易关联。但是，交付不得与结果关联到不同的*非出清*交易。如果出现这种情况，控制链路必须以[事务回滚](#_bookmark174)错误终止。

为了将结果与交易关联起来，控制器发送一个[处置](#_bookmark80)执行器，将传送状态设置为[交易状态](#_bookmark170)，并设置所需的交易标识符和成功卸载后应用的结果。

事务控制器 事务资源

===============================================================================

TRANSFER(handle=0、 >

delivery-id=0、 )

{ AmqpValue( Declare() )}

<--------- DISPOSITION（first=0，last=0、

state=Declared(txn-id=0) )

FLOW(handle=2、 >

link-credit=10)

< TRANSFER(handle=2、

delivery-id=11, state=null、

{ ... 有效载荷 }

:

:

< TRANSFER(handle=2、

delivery-id=20, state=null、

{ ... 有效载荷 }

DISPOSITION(first=11、 >

last=20, state=TransactionalState(

txn-id=0, outcome=Accepted())

)

图 4.4：事务接收

[卸载](#_bookmark167)成功后，资源将应用给定的结果，并可立即结算转账。如果出现由控制器发起的回退（失败标志设置为 true 的[放行](#_bookmark167)）或由资源发起的回退（[放行](#_bookmark167)信息被拒绝，或与[协调器](#_bookmark165)的链接因错误而分离），则不会应用结果，交付仍将是 "实时 "的，并仍由控制器获取，即资源应期待控制器请求对交付进行处置（在新事务中进行事务性处置或非事务性处置）。

## 交易性收购

就流帧而言，事务性工作并不一定在流帧到达资源时直接启动或完全确定，而是可能在稍后时间以控制器不一定预料到的方式发生。为适应这种情况，资源会将额外的状态与出站链接端点关联起来，即一个可选的 *txn-id*，用于识别与*获取的*报文相关联的传输操作。该状态由控制器通过在流帧*属性*映射中指定 *txn-id* 条目来确定。事务结束时，任何链接端点的 *txn-id 都将*被清除。

如果链接端点不支持事务性采集，则必须用一个

[未执行](#_bookmark102)错误。

虽然 *txn-id* 会在交易解除时清空，但这并不影响未清空贷记的水平。为防止发送链接端点获取任何交易以外的信息，*控制器*应确保在解除交易前发送方没有未清偿的贷记。为此，*控制器*可以在解除交易前将发送链路端点的耗尽模式设为 *true，*或将*链路*贷记减为零，并等待发送方看到这一状态变化的反馈。

如果交易在某一点被解除，电文已被交易获取，但

如果未完全发送（即信息的发送需要一个以上的传输帧，且至少发送了一个（但不是全部）传输帧），则资源必须将此解释为获取的命运完全由卸载决定。如果[卸载](#_bookmark167)表明交易失败，资源必须在完成卸载之前中止或完成剩余信息的发送。

事务控制器 事务资源

===============================================================================

TRANSFER(handle=0、 >

delivery-id=0、 )

{ AmqpValue( Declare() )}

<--------- DISPOSITION（first=0，last=0、

state=Declared(txn-id=0) )

FLOW(handle=2、 >

link-credit=10, drain=true, properties={

txn-id=0

})

< TRANSFER(handle=2、

delivery-id=11, state=

事务状态（txn-id=0）、

{ ... 有效载荷 }

:

:

< TRANSFER(handle=2、

delivery-id=20, state=

事务状态（txn-id=0）、

{ ... 有效载荷 }

DISPOSITION(first=11、 >

last=20, state=TransactionalState(

txn-id=0, outcome=Accepted())

)

图 4.5：交易型获取

## 结算与交易的互动

传输层定义了一个*结算的*概念，它指的是一个递送标记与递送尝试的关联被遗忘的时间点。交易本身并不会改变这一概念，但是交易工作可以回滚这一事实确实会对端点已标记为结算的交付产生影响（因此端点无法再使用先前分配的传输层标识符交换状态信息）。

* + - 1. **交易过账 交付 发送 由控制员结算**

在事务成功结束之前，节点不会提供已交付的报文。如果事务回滚，则无法提供传送。如果资源无法处理递送，则不得允许成功解除相关事务。具体做法是立即销毁声明事务的控制链路，或拒绝任何未将失败标记设为 true 的解除事务尝试。

**控制员未结算的发送；资源结算**

资源必须在提交交易之前确定交付结果，并且必须在接受成功卸载之前将此结果通知控 制器。资源通报的结果必须与从控制器到资源的传输所关联的同一事务相关联。

如果交易回滚，则无法在目标处进行交付。如果资源无法再应用它最初指出的成功释放的结果，就不得允许成功释放相关的事务。为此，可以立即销毁声明事务的控制链接，或拒绝任何未将失败标记设为 true 的事务释放尝试。

**控制员发送的交付未结算；资源未结算**

出院前的表现与前一个病例相同。

成功出院后，资源的未结算交付状态必须反映所应用的结果。如果出清不成功，则未结算交付不应与任何结果相关联。交易清空后，控制器应及时结算任何未结算的交付。

### 事务性退休

这里我们考虑的是资源以非事务方式向控制器发送信息的情况。关于资源以事务方式发送消息的情况，请参阅下文的**事务获取。**

**资源发送的交货未结算；控制员结算**

成功放电后，控制器指定的结果将应用于源。如果控制器请求回滚或卸载尝试不成功，则不会应用结果。此时，控制器不能再影响已结算的交付状态，资源必须应用默认结果。

**资源发送的交货未结算；控制器未结算**

在解除交易前，资源可能会也可能不会结算交货。如果资源在解除交易前结清货款，则解除交易后的行为与上述情况相同。

成功解除后，结果将被应用。否则，状态将恢复到控制器发送（事务性）处置之前的状态。控制器可使用后续的事务性或非事务性更新自由地更新状态。

* + - 1. **交易性收购 交付 发送 按资源结算**

如果成功卸载，则结果适用于该资源，否则获取和结果将回滚。

**资源发送的交付未结算；控制员发送结果**

交易解除前控制器发送的结果必须与同一交易相关联。即使成功解除，结果也会在源应用，否则获取和结果都会回滚。

* 1. **协调**

## 协调员

与交易协调员通信的目标。

<type name="coordinator" class="composite" source="list" provides="target">

<descriptor name="amqp:coordinator:list" code="0x00000000:0x00000030"/>

<field name="capabilities" type="symbol" requires="txn-capability" multiple="true"/>

</类型

协调器类型定义了一个特殊目标，用于与事务协调器建立链接。

**实地详情**

能力 *协调员支持的能力*

由事务控制器（发送端点）发送时，表示协调器所需的能力。由资源（接收端点）发送时，定义协调器的实际能力。请注意，验证协调器的能力是否满足其要求是事务控制器的责任。请参阅 [txn-capability](#_bookmark171)。

## 宣布

用于声明事务 ID 的信息体。

<type name="declare" class="composite" source="list">

<descriptor name="amqp:declare:list" code="0x00000000:0x00000031"/>

<field name="global-id" type="\*" requires="global-tx-id"/>

</类型

声明类型定义了发送给协调器以声明事务的信息体。为该事务分配的 txn-id 由事务控制器选择，并在[声明](#_bookmark169)结果中标识。

**实地详情**

global-id *全局交易 ID*

指定该声明分配的 txn-id 必须与指定的全局事务相关联。如果不设置，分配的 txn-id 将与本地事务相关联。如果协调器不具备[分布式](#_bookmark172)事务功能，则不得设置此字段。请注意，AMQP 1.0 中的分布式事务规范将在第 6 册《分布式事务》中单独提供。

## 出院

解除交易的信息体。

<type name="discharge" class="composite" source="list">

<descriptor name="amqp:discharge:list" code="0x00000000:0x00000032"/>

<field name="txn-id" type="\*" requires="txn-id" mandatory="true"/>

<field name="fail" type="boolean"/>

</类型

放行类型定义了发送到协调器的信息体，以表明 txn-id 不再使用。如果事务与全局标识无关，则也表示本地事务的处理。

**实地详情**

txn-id *标识要解除的交易*

失败*表示交易应回滚*

如果设置了该标志，则表明与该事务相关的工作已经失败，控制器希望回滚该事务。如果事务与全局标识相关联，则全局事务只能回滚。如果事务是本地事务，那么这个标志将控制事务在终止时是提交还是中止。(请注意，AMQP 1.0 中的分布式事务规范将在第 6 册《分布式事务》中单独提供）。

## 交易 ID

<type name="transaction-id" class="restricted" source="binary" provides="txn-id"/>

事务标识最多可以是 32 个八进制二进制数据。

## 已宣布

<type name="declared" class="composite" source="list" provides="delivery-state, outcome">

<descriptor name="amqp:declared:list" code="0x00000000:0x00000033"/>

<field name="txn-id" type="\*" requires="txn-id" mandatory="true"/>

</类型

表示响应发送给事务协调器的声明报文，成功分配了一个事务标识符。

**实地详情**

txn-id *已分配的交易标识*

## 交易状态

事务性信息传输的状态。

<type name="transactional-state" class="composite" source="list" provides="delivery-state">

<descriptor name="amqp:transactional-state:list" code="0x00000000:0x00000034"/>

<field name="txn-id" type="\*" mandatory="true" requires="txn-id"/>

<field name="outcome" type="\*" requires="outcome"/>

</类型

事务状态类型定义了一种交付状态，用于将交付与跨操作关联起来，并指出如果事务提交，将应用哪种结果。

**实地详情**

txn-id *标识与状态相关联的交易*

*临时结果*

该字段表示事务提交时要应用的临时结果。

## 交易能力

表示交易协调员（所需/可用）能力的符号。

<type name="txn-capability" class="restricted" source="symbol" provides="txn-capability">

<choice name="local-transactions" value="amqp:local-transactions"/>

<choice name="distributed-transactions" value="amqp:distributed-transactions"/>

<choice name="promotable-transactions" value="amqp:promotable-transactions"/>

<choice name="multi-txns-per-ssn" value="amqp:multi-txns-per-ssn"/>

<choice name="multi-ssns-per-txn" value="amqp:multi-ssns-per-txn"/>

</类型

**有效值**

**amqp:local-transactions**

支持本地交易。

**分布式交易**

支持 AMQP 分布式事务。

**amqp:promotable-transactions**

支持 AMQP Promotable 事务。

**amqp:multi-txns-per-ssn**

支持单个会话中的多个活动事务。

**amqp:multi-ssns-per-txn**

支持在一个连接上跨会话使用 txn-id 的事务。

## 交易错误

用于表示交易错误的符号。

<type name="transaction-error" class="restricted" source="symbol" provides="出错条件">。

<choice name="unknown-id" value="amqp:transaction:unknown-id"/>

<choice name="transaction-rollback" value="amqp:transaction:rollback"/>

<choice name="transaction-timeout" value="amqp:transaction:timeout"/>

</类型

**有效值**

**amqp:transaction:unknown-id**

指定的 txn-id 不存在。

**amqp:事务:回滚**

交易因不明原因被退回。

**amqp:transaction:timeout**

这笔交易所代表的工作耗时过长。

**第五册**

**安全**

* 1. **安全层**

安全层（Security Layers）用于建立一个经过验证和/或加密的传输层，常规 AMQP 流量可通过该传输层进行隧道传输。安全层可以相互隧道（例如，对等方用于进行身份验证的安全层可以在为加密目的而建立的安全层上隧道）。

与 TLS 的情况一样，安全层的框架和协议定义预计将在 AMQP 规范之外定义。SASL 安全层是一个例外，它依赖于其主机协议来提供框架。因此，我们在下文[第 5.3 节 SASL](#_bookmark181) 中定义了 [SASL](#_bookmark181) 运行所需的框架。当安全层终止时（无论是在安全隧道建立之前还是之后），TCP 连接必须关闭，方法是首先关闭传出流，然后读取传入流，直至其终止。

* 1. **TLS**

要建立 TLS 隧道，每个对等方必须先发送一个协议头。协议头由大写 ASCII 字母 "AMQP "和两个协议 ID 组成，之后是三个无符号字节，分别代表规范版本的主版本、次版本和修订版本（目前为 [1 (TLS-MAJOR](#_bookmark180))、[0 (TLS-MINOR](#_bookmark180))、[0 (TLS-REVISION](#_bookmark180))）。这总共是一个 8 八位字节的序列：

4 个 八位字节1 个八位字节 1 个八位字节 1 个八位字节

+----------+---------+---------+---------+ +

| "AMQP" | "AMQP %d2| 主版本 | 次版本 | 修订版本 | 主版本 | 次版本 | 修订版本

+----------+---------+---------+---------+ +

除了使用两个协议 ID 外，TLS 隧道头的交换遵循传输规范版本协商部分规定的相同规则（参见版本协商）。

下图说明了创建 TLS 安全层所涉及的交互过程：

TCP 客户端TCP 服务器

========================================= AMQP%d2.1.0.0 --------->

< AMQP%d2.1.0.0

:

:

<TLS 协商

:

:

AMQP%d0.1.0.0 --------->（通过 TLS 加密连接）

< AMQP%d0.1.0.0

打开 >

< 打开

协商使用 TLS 安全层时，适用以下规则：

* TLS 客户端对等体和 TLS 服务器对等体分别由 TCP 客户端对等体和 TCP 服务器对等体决定。
* TLS 客户端对等方应使用 RFC- 4366 中描述的服务器名称指示扩展。如果是这样，那么在 [sasl-init](#_bookmark186) 和 [open](#_bookmark75) frame 框架中，如果与 hostname 不同，会发生什么情况，这取决于具体的实现。

AMQP 代理可使用此字段确定要将客户端连接到的正确后端服务，并在使用 TLS 客户端证书时确定要验证客户端凭证的域。

* TLS 客户端必须验证 TLS 服务器提供的证书。
* 实施者可以选择使用单向关闭的 TLS，即使用关闭通知（close notify）启动关闭的应用程序不必等待对等程序响应，可以关闭 TCP 套接字的写入半部分。

## 替代机构

在某些情况下，例如通过防火墙连接，可能无法使用隧道技术建立 TLS 安全层。这可能是因为深度数据包检查防火墙将连接的前几个字节视为 "非 TLS"。

作为替代方案，实施方案可以运行纯 TLS 服务器，即不需要隧道协商握手的服务器。其 IANA 服务名称为 amqps，端口为 SECURE- PORT (5671)。如果在操作上有需要，实施者也可以选择在其他端口上运行纯 TLS 服务器（例如，要通过传统防火墙进行隧道传输，该防火墙只允许在 443 端口上传输 TLS 流量）。

## 常量定义

|  |  |  |
| --- | --- | --- |
| TLS-MAJOR | 1 | 主要协议版本。 |
| TLS-MINOR | 0 | 次要协议版本。 |
| TLS-REVISION | 0 | 修订协议。 |

* 1. **SASL**

要建立 SASL 隧道，每个对等方必须先发送一个协议头。协议头由大写 ASCII 字母 "AMQP "组成，后面是三个协议 ID，后面是三个

无符号字节，代表规范版本的主版本、次版本和修订版本（目前为 [1 (SASL-MAJOR)](#_bookmark193)、[0 (SASL-MINOR](#_bookmark193))、[0 (SASL-REVISION](#_bookmark193))）。这总共是一个 8 八位字节的序列：

4 个 八位字节1 个八位字节 1 个八位字节 1 个八位字节

+----------+---------+---------+---------+ +

| "AMQP" | "AMQP %d3| 主版本 | 次版本 | 修订版本 | 主版本 | 次版本 | 修订版本

+----------+---------+---------+---------+ +

除了使用三个协议 ID 外，SASL 隧道头的交换遵循传输规范版本协商部分规定的相同规则（参见版本协商）。

下图说明了创建 SASL 安全层所涉及的交互：

TCP 客户端TCP 服务器

========================================= AMQP%d3.1.0.0 --------->

< AMQP%d3.1.0.0

:

:

<SASL协商

:

:

AMQP%d0.1.0.0 --------->（通过 SASL 安全连接）

< AMQP%d0.1.0.0

打开 >

< 打开

## SASL 框架

SASL 采用成帧方式进行协商。SASL 帧的类型代码为 0x01。报头的第 6 和第 7 字节将被忽略。实施方案应将其设置为 0x00。扩展报头将被忽略。因此，实施机构应将 DOFF 设置为 0x02。

类型：0x01 - SASL 帧

+0 +1 +2 +3

+-----------------------------------+ -.

0 | SIZE| |

+-----------------------------------+ |---> 帧头

4 | doff | type | <IGNORED>\*1| | （8 字节）

+-----------------------------------+ -’

+-----------------------------------+ -.

8 | ... | |

. .|---> 扩展标头

. <IGNORED>\*2 .| (DOFF \* 4 - 8) 字节

| ... | |

+-----------------------------------+ -’

+-----------------------------------+ -.

4\*DOFF || | |

. .|

. .|

. Sasl机制 / Sasl Init .|

. Sasl Challenge / Sasl Response .|---> 帧正文

. Sasl 结果 。| (SIZE - DOFF \* 4) 字节

. .|

. .|

. | |

| ... | |

+--------------------------+ -'

\*1 应设置为 0x0000

\*2 忽略，因此 DOFF 应设置为 0x02

图 5.1：SASL 框架

SASL 帧的最大尺寸由 MIN-MAX-FRAME-SIZE 定义。SASL 协商中没有协商不同大小的机制。SASL 帧的帧体只能包含一个 AMQP 类型，其类型编码必须是 provides="sasl-frame"。收到空帧是不可恢复的错误。

## SASL 谈判

作为 SASL 服务器的对等方必须使用 [sasl-mechanisms](#_bookmark185) 框架宣布支持的身份验证机制。然后，对等方必须从支持的机制中选择一种，并启动 sasl 交换。

SASL 客户端SASL 服务器

================================

<-- sasl-mechanisms

SASL-INIT -->

...

<-- sasl-challenge \* sasl-response -->

...

<-- sasl-outcome

\* 请注意，根据所选 SASL 机制的具体情况，SASL 质疑/响应步骤可能会发生零次或多次。

图 5.2：SASL 交换

扮演 SASL 客户端角色的对等节点和扮演 SASL 服务器角色的对等节点必须分别与 TCP 客户端和服务器相对应。

## 安全框体

### SASL 机制

宣传可用的 sasl 机制。

<type name="sasl-mechanisms" class="composite" source="list" provides="sasl-frame">

<descriptor name="amqp:sasl-mechanisms:list" code="0x00000000:0x00000040"/>

<field name="sasl-server-mechanisms" type="symbol" multiple="true" mandatory="true"/>

</类型

公布可用于身份验证的可用 SASL 机制。

**实地详情**

*支持的 sasl* *机制*

发送对等方支持的 sasl 安全机制列表。该列表为空或无效。如果发送对等方不要求其伙伴对其进行身份验证，则应发送包含一个元素的列表，其值为 SASL 机制 *ANONYMOUS*。服务器机制按优先级递减排列。

### SASL 初始化

启动 sasl 交换。

<type name="sasl-init" class="composite" source="list" provides="sasl-frame">

<descriptor name="amqp:sasl-init:list" code="0x00000000:0x00000041"/>

<field name="mechanism" type="symbol" mandatory="true"/>

<field name="initial-response" type="binary"/>

<field name="hostname" type="string"/>

</类型

选择 sasl 机制，并在需要时提供初始响应。

**实地详情**

*选定的安全机制*

用于 SASL 交换的 SASL 机制名称。如果接收对等节点不支持所选机制，则必须使用身份验证失败关闭代码关闭连接。每个对等节点必须使用合作伙伴提供的列表中它能处理的最高级别安全配置文件进行身份验证。

初始响应*安全响应数据*

传递给安全机制的不透明数据块。数据内容由 SASL 安全机制定义。

hostname *目标主机的名称*

发送对等程序正在连接的主机的 DNS 名称（完全限定名称或相对名称）。并非必须提供主机名。如果没有提供主机名，接收对等端应根据自身配置选择一个默认主机名。

AMQP 代理可使用此字段确定连接客户端的正确后端服务，并确定验证客户端凭据的域。

如果使用 TLS 层，该字段可能已由 RFC-4366 中描述的服务器名称指示扩展指定，在这种情况下，该字段应为空或包含相同的值。至于与已指定的值有何不同，目前尚不明确。

### SASL 挑战赛

安全机制挑战。

<type name="sasl-challenge" class="composite" source="list" provides="sasl-frame">

<descriptor name="amqp:sasl-challenge:list" code="0x00000000:0x00000042"/>

<field name="challenge" type="binary" mandatory="true"/>

</类型

发送 SASL 规范定义的 SASL 挑战数据。

**实地详情**

挑战*安全 挑战数据*

挑战信息，传递给安全机制的不透明二进制数据块。

### SASL 回应

安全机制响应。

<type name="sasl-response" class="composite" source="list" provides="sasl-frame">

<descriptor name="amqp:sasl-response:list" code="0x00000000:0x00000043"/>

<field name="response" type="binary" mandatory="true"/>

</类型

发送 SASL 规范定义的 SASL 响应数据。

**实地详情**

*安全响应数据*

传递给安全机制的不透明数据块。数据内容由 SASL 安全机制定义。

### SASL 成果

表示 sasl 对话框的结果。

<type name="sasl-outcome" class="composite" source="list" provides="sasl-frame">

<descriptor name="amqp:sasl-outcome:list" code="0x00000000:0x00000044"/>

<field name="code" type="sasl-code" mandatory="true"/>

<field name="additional-data" type="binary"/>

</类型

该帧表示 SASL 对话的结果。SASL 对话成功完成后，安全层就已建立，对等方必须交换协议头，以启动嵌套安全层或建立 AMQP 连接。

**实地详情**

代码 *表示 sasl 对话的结果*

表示 SASL 对话结果的回复代码。

additional-data *RFC-4422 中规定的附加数据*

additional-data 字段包含 SASL 规范（RFC-4422）规定的认证成功后的附加数据。如果身份验证不成功，则不设置此字段。

### SASL 编码

表示 sasl 对话结果的代码。

<type name="sasl-code" class="restricted" source="ubyte">

<选择项 name="ok" value="0"/>

<choice name="auth" value="1"/>

<choice name="sys" value="2"/>

<choice name="sys-perm" value="3"/>

<choice name="sys-temp" value="4"/>

</类型

**有效值**

* + - * 1. 连接验证成功。
        2. 由于提供的凭据存在不明问题，连接验证失败。
        3. 由于系统错误，连接验证失败。
        4. 由于系统错误，连接身份验证失败，该错误不可能在不干预的情况下纠正。
        5. 由于瞬时系统错误，连接验证失败。

## 常量定义

|  |  |  |
| --- | --- | --- |
| SASL-MAJOR | 1 | 主要协议版本。 |
| SASL-MINOR | 0 | 次要协议版本。 |
| SASL-REVISION | 0 | 协议修订。 |