ICS 35.240.60
V 07

MH

中华人民共和国民用航空行业标准

MH/T 0048.2—2015

民用机场共用旅客处理系统技术规范 第2部分：应用软件数据交换

Specification of common use passenger processing systems in civil aviation airports

Part 2：Interface between applications

2015－09－30 发布　　2015－12－01 实施

中国民用航空局　发布

目　　次

前言 II
1　范围 1
2　术语和定义 1
3　数据交换接口通用标准 1
4　数据交换接口消息处理规则 23
附录A（资料性附录）　数据交换接口及消息处理示例 32

前　言

MH/T 0048分为以下三个部分：

——第 1 部分：系统结构；

——第 2 部分：应用软件数据交换；

——第 3 部分：硬件设备数据交换。

本部分为MH/T 0048的第2部分。

本部分按照GB/T 1.1-2009给出的规则起草。

本部分由中国民用航空局人教司提出。

本部分由中国民用航空局航空器适航审定司批准立项。

本部分由中国民航科学技术研究院归口。

本部分起草单位：中国民航大学、中国民航信息网络股份有限公司。

本部分主要起草人：李建伏、孙皓、贺怀清、张博、惠康华、丁玎、徐涛、武佳、孙启玲、李斌。

民用机场共用旅客处理系统技术规范
第2部分：应用软件数据交换

1 范围

MH/T 0048的本部分规定了共用旅客处理系统的应用软件与管理平台之间数据交换接口技术规范。

本部分适用于中国民用机场共用旅客处理系统的建设。

2 术语和定义

2.1

共用旅客处理系统 CUPPS Common Use Passenger Processing Systems

由国际航协定义，用于航空公司使用机场的终端设备的信息处理规范。

[MH/T 0048.1-2014中的3.1]

2.2

CUPPS 工作站 CUPPS Workstation

运行于CUPPS平台上的硬件和操作系统软件。硬件包括计算机、移动终端设备、智能手机、简易客户终端等。

[MH/T0048.1-2014中的3.2]

2.3

CUPPS 应用 CUPPS Application

运行在CUPPS平台上，使用CUPPS平台接口的应用程序。

[MH/T 0048.1-2014中的3.3]

3 数据交换接口通用标准

3.1 平台架构

平台管理所有连接到平台的请求，并为这些连接提供标准接口来访问机场资源和其他系统。

平台可使用多种体系架构，见图1。

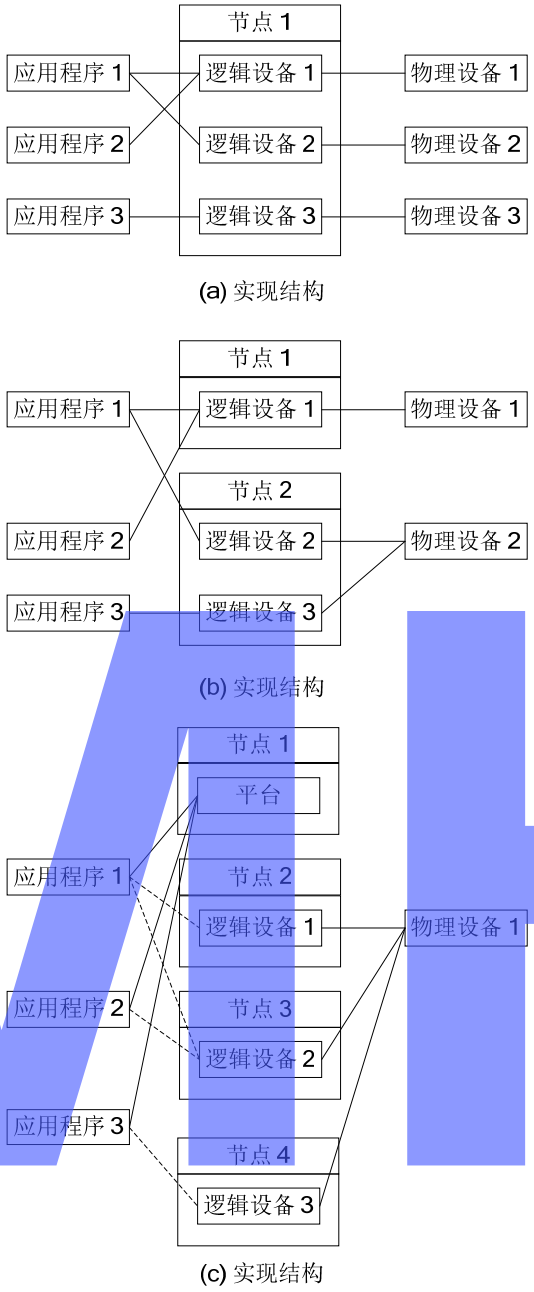


图1　平台架构

平台架构由以下组成，应用程序1使用设备1和2，应用程序2使用设备1，应用程序3使用设备3：

—— (a)实现架构：表示在该平台上应用程序通过一个节点(节点 1)连接到所有设备，每个逻辑设备由一个独立的物理设备实现；

—— (b)实现架构：表示在该平台上应用程序通过两个节点(节点 1 和 2)连接到设备，节点 1 提供了物理设备 1 的逻辑接口。节点 2 为一个多功能物理设备提供逻辑接口；

—— (c)实现架构：表示在该平台上，应用程序首先连接到节点 1，该节点提供了一个逻辑分配机制，使应用程序可重定向到其他节点。当应用程序 1 连接到平台上的节点 1 获取逻辑设备 1 和 2 时，该平台分别将应用程序 1 重定向到节点 2 和 3；当应用 2 连接到平台上的节点 1 获取逻辑设备 2 时，该平台将应用 2 重定向到节点 3；当应用 3 连接到平台上的节点 1 获取逻辑设备 3，该平台将应用程序 3 重定向到节点 4。其中，所有的逻辑设备是通过一个多功能物理设

备来实现。

注：应用程序通过设备的名称和接口访问设备，与设备所在位置无关，名称和接口都由环境变量指定。应用程序不能推理出平台组件的任何特定实现方法，以及设备与平台连接的任何特定物理实现方法。

3.2 接口状态

CUPPS接口的不同状态如表1所示。

在生产环境中，如果应用程序试图使用支持状态之外的接口，平台应记录相应的信息供管理员审查，并触发警报。

表1 CUPPS 接口状态

状态	描述
提出	接口已经被提出，但是还没计划
计划	接口正在计划中
开发	接口处于实验性测试状态，不能在生产环境中使用
支持	接口处于正常的生产状态
不建议	接口仍然被支持，但不推荐使用。该接口即将进入过时状态。在生产过程中对该接口的任何使用，都将产生一个警告，且被平台记录下来
废弃	接口不再被支持。在生产中任何试图使用这种接口的行为都产生一个错误

3.3 接口版本

3.3.1 在每个单独的 TCP(Transmission Control Protocol 传输控制协议)端口上的平台和设备端应同时支持多个版本的 CUPPS 接口。当应用程序连接到 CUPPS 平台后，首先应请求得到该平台支持的接口版本列表，然后从获取的接口版本列表中选择一个版本进行后续的会话。

3.3.2 会话消息需通过已选择的接口版本的校验。如果校验失败，则触发<sessionErrorEvent>消息，并立即关闭该会话。

3.3.3 如果应用程序在版本列表中没有找到支持的接口，应用程序应做以下处理：

a) 记录错误；

b) 发送一个错误事件；

c) 通知终端用户。终端用户根据提示信息，咨询相应的技术支持人员。

所有的 CUPPS 接口都遵守一套通用的基本规则。所有 CUPPS 接口应使用 TCP 套接字，消息交换内容应符合已定义 XSD(XML Schemas Definition，XML 结构定义)的 W3C(World Wide Web Consortium,万维网联盟) XML(Extensible Markup Language，可扩展标记语言)消息格式。

3.4 平台主机名和端口

应用程序通过TCP/IP(Internet Protocol，网络间通信协议)连接服务器，服务器应向应用程序提供对应的服务器主机名和端口。应用程序根据CUPPSPN与CUPPSPP环境变量确定平台的主机名和端口。考虑到生产和测试环境的灵活性，可使用IP地址127.0.0.1。

3.5 会话原则

3.5.1 会话流程

应用程序和平台之间的会话流程图如图2所示，主要包括以下几个步骤：

a) 平台监听一个已知的节点和端口；
b) 应用程序连接到平台的相应节点和端口。在连接时，应用程序向平台发出可用接口版本列表请求，平台收到请求后返回可用接口列表；
c) 应用程序向平台发送期望使用的接口版本。该接口版本经平台确认后，将用于后续会话消息的验证；
d) 平台验证应用程序的合法性，并返回验证结果。如果验证成功，则应在验证结果中包含令牌信息。该令牌信息应用于所有后续通信以及验证设备连接。当平台连接关闭后，令牌失效，基于此令牌的连接也应立即断开；
e) 应用程序使用已定义消息集与平台进行通信。对于每个设备会话，应用程序在断开连接之前，均应发送<deviceReleaseRequest>并获得平台对应的响应；
f) 应用程序断开。当应用程序关闭时，应向平台发送<byeRequest>消息，获取<byeResponse>消息后方可断开连接。

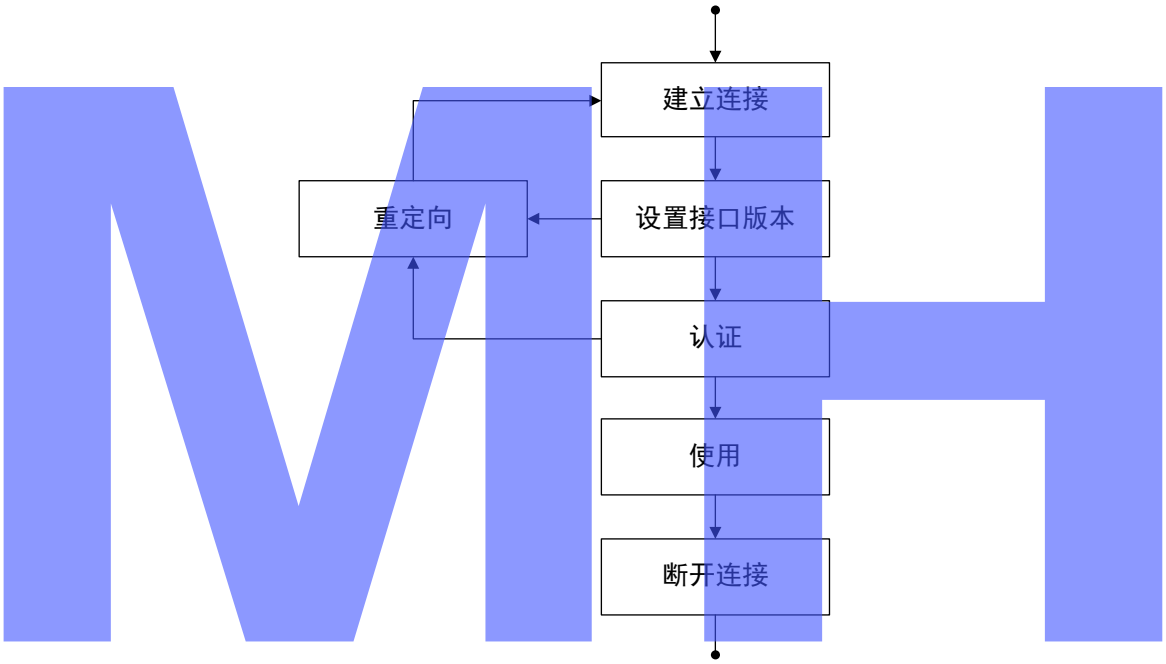


图2 会话流程

基本的会话连接序列图原型参见附录A.1。

3.5.2 消息处理流程

3.5.2.1 建立连接后,当接收端收到消息时，应判断消息流的合法性。

3.5.2.2 消息从接收到处理的流程见图 3。处理流程如下：

a) 接收端逐字节地读取消息头：如果接收端读取到无效字符，则接收端立即停止读取，发送<sessionErrorEvent>的通知，并在该通知中包含信息 eventType = “headerVersion”，随后关闭连接。如果接收端读取的消息长度越界，则接收端发送<sessionErrorEvent>的通知，并包含信息 eventType = “headerLength”，随后关闭连接。如果读取消息头超时，则接收端发送<sessionErrorEvent>的通知，并包含信息 eventType = “headerTimeout”，随后关闭连接；
b) 读取消息体：消息头中包含消息体的长度，如果在读取这个指定长度的消息体时超时，接收端发送<sessionErrorEvent>的通知，其中包含信息 eventType =”bodyTimeout”，随后关闭连接；

c) 解析消息体：如果接收端无法使用选定的接口版本解析消息体，则接收端发送<sessionErrorEvent>的通知，其中包含信息 eventType=“bodyParseFailure”，随后关闭连接；

d) 处理经过解析的信息。

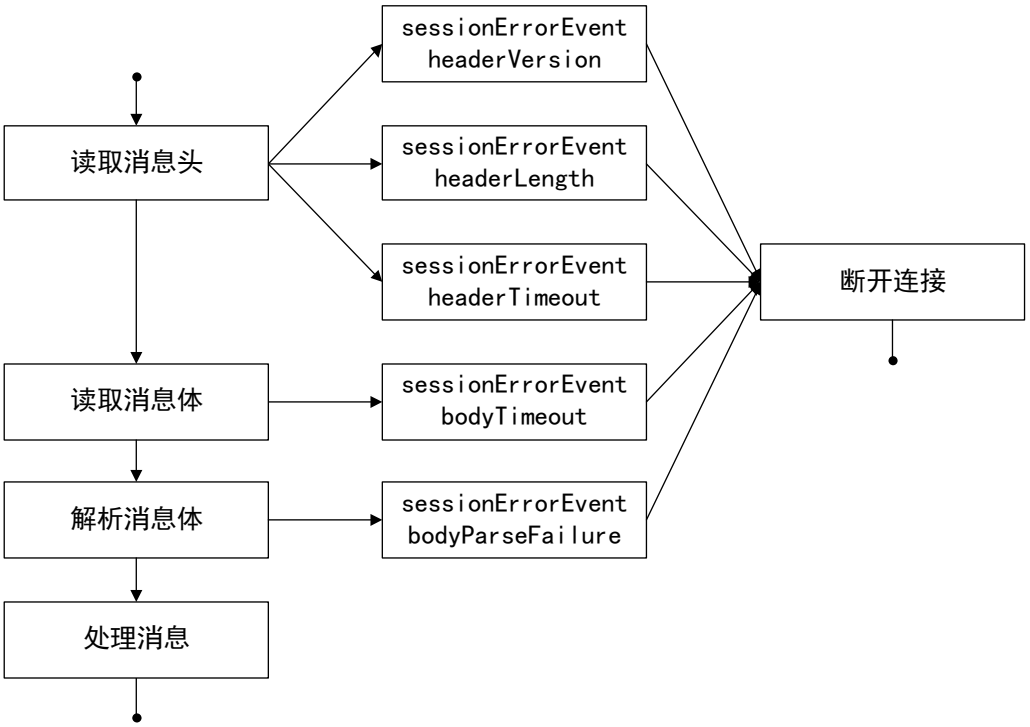


图3 平台消息处理流程

3.5.3 应用程序认证

3.5.3.1 一旦应用程序连接到平台并且选择好接口版本，则该应用程序应向平台请求认证。如果应用程序未能通过认证，则平台应断开与该应用程序的连接。

3.5.3.2 应用程序认证分为两步：

a) 应用程序向平台发送<authenticateRequest>消息请求认证；

b) 平台记录应用程序的认证请求，回复<authenticateResponse>消息。该消息应包含以下信息：

1) 设备令牌：用于后续设备交互操作；
2) 平台参数：平台供应商 ID 和版本信息以用于日志记录；
3) 支持的加密算法列表；
4) 应用程序参数：局部存储、局部临时存储以及永久性全局存储地址；
5) 工作站参数：工作站的名字、厂商信息、平台的配置信息、操作系统默认的打印机名称等；
6) 设备分配列表(样品)；
7) 特殊模式接口，ZI(IATA message software device，IATA 信息设备)和 ZL（logging software device，日志设备）都是被指定的。

应用程序认证请求与回复的消息示例参见附录A.8。

3.5.4 应用程序主动与平台断开连接

3.5.4.1　应用程序与平台端口断开连接之前，应向平台发送<byeRequest>消息以通知平台方应用程序将要断开连接。平台端接收到<byeRequest>消息后，回复<byeResponse>消息，并等待应用程序断开连接。附录 A.6 为与平台断开连接的示例。

3.5.4.2　应用程序发送<byeRequest>消息后，平台端将该应用程序设置为 aSpg 状态，不允许该应用程序继续向平台发送其他消息。如果该应用程序在 aSpg 状态试图发送消息，平台记录<sessionErrorEvent>事件。

3.5.4.3　平台接收<byeRequest>后，等待应用程序断开连接,且不能向应用程序发送其他消息。如果在 PltSockMaxIdleTime[1)]后，该应用程序仍未断开，平台应作相应日志记录，然后断开该连接。

3.5.5　平台端发起的应用程序正常退出机制

3.5.5.1　本部分仅规范了应用程序的正常退出机制，应用程序的异常退出机制见本标准第 1 部分的 9.1.8。

3.5.5.2　平台端可通过发送特定的指令断开应用程序连接。断开连接的形式有两种，即请求方式与控制方式[1)]。

3.5.5.3　请求方式通过平台向应用程序发送指令停止应用程序。在接收到平台的停止指令后，应用程序应关闭与设备的连接、记录日志、通知用户应用程序正在退出，并最终关闭与平台的连接。此过程应在 AppSpgTime[2)]时间内完成，如果需要更多的时间退出应用程序，最多可增加 MaxSpgDegerTimes[3)]次延迟退出的请求。

3.5.5.4　控制方式通过平台端执行指令停止应用程序。在这种情况下，应用程序被置为 aSpg 状态，如果应用程序退出时长超过了 AppSpgTime[4)]，应用程序将会进入 aZom 状态并被强制清除。

图4给出了平台端发起的应用程序正常退出的四种情况：

a)　平台允许应用程序延迟退出，但应用程序不需要延迟的情况，分为：

1)　步骤“1”，平台请求应用程序停止；

2)　步骤“2”，应用程序回复可以退出。平台立即使分配给该应用程序的设备令牌失效；

3)　步骤“3”，应用程序关闭与平台的连接，然后退出程序。

b)　平台允许应用程序延迟退出，且应用程序需要延迟的情况，分为：

1)　步骤“1”，平台请求应用程序停止。应用程序回复平台需要延迟退出。平台重置应用退出计时器 AppSpgTime[5)]；

2)　步骤“2”，应用退出计时器超时，平台继续向应用程序发送停止请求。应用程序回复延迟退出。平台重置应用退出计时器；

1)　见表 3。
2)　见表 3。
3)　见表 3。
4)　见表 3。
5)　见表 3。

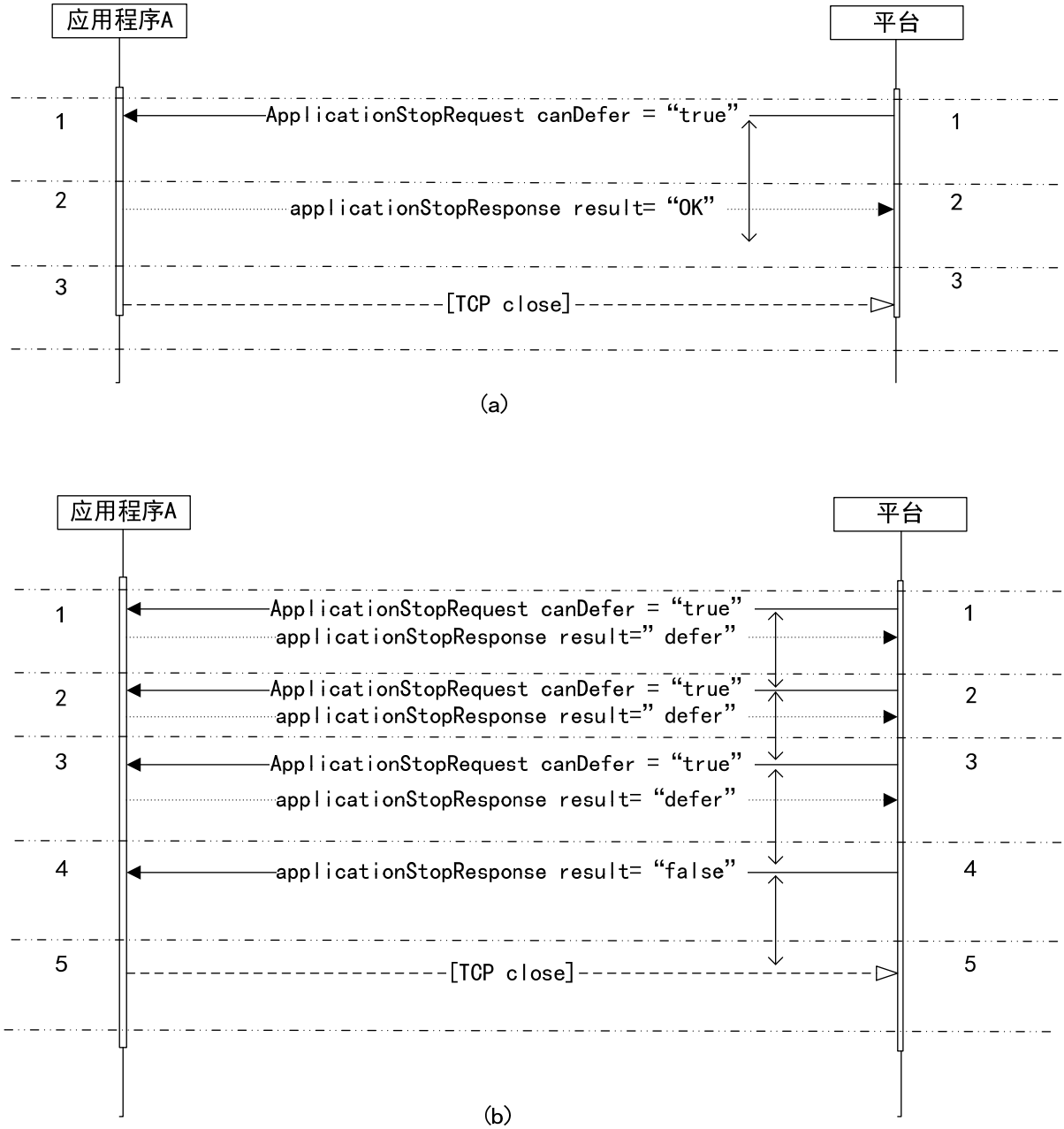


图4　应用程序退出场景

3)　步骤“3”，应用退出计时器超时，平台端将应用程序置为 aSpg 状态。该应用程序回复需要延迟退出时间，但是平台已不允许应用程序延时，所以平台忽略了这一请求；

4)　步骤“4”，应用程序尚未终止与平台的连接，平台最后一次发送停止请求，并忽略所有来自应用程序的消息；

5)　步骤“5”，应用退出计时器超时。平台将应用程序的状态改为 aZom，强行断开连接并释放与该应用程序有关的所有资源。

c)　平台不允许应用程序延迟退出，且应用程序也不需要延迟的情况。分为：

1)　步骤“1”，平台请求应用程序停止，并将应用程序置为 aSpg 状态；

2)　步骤“2”，应用程序回复可以退出。平台立即将分配给应用程序的设备令牌失效；

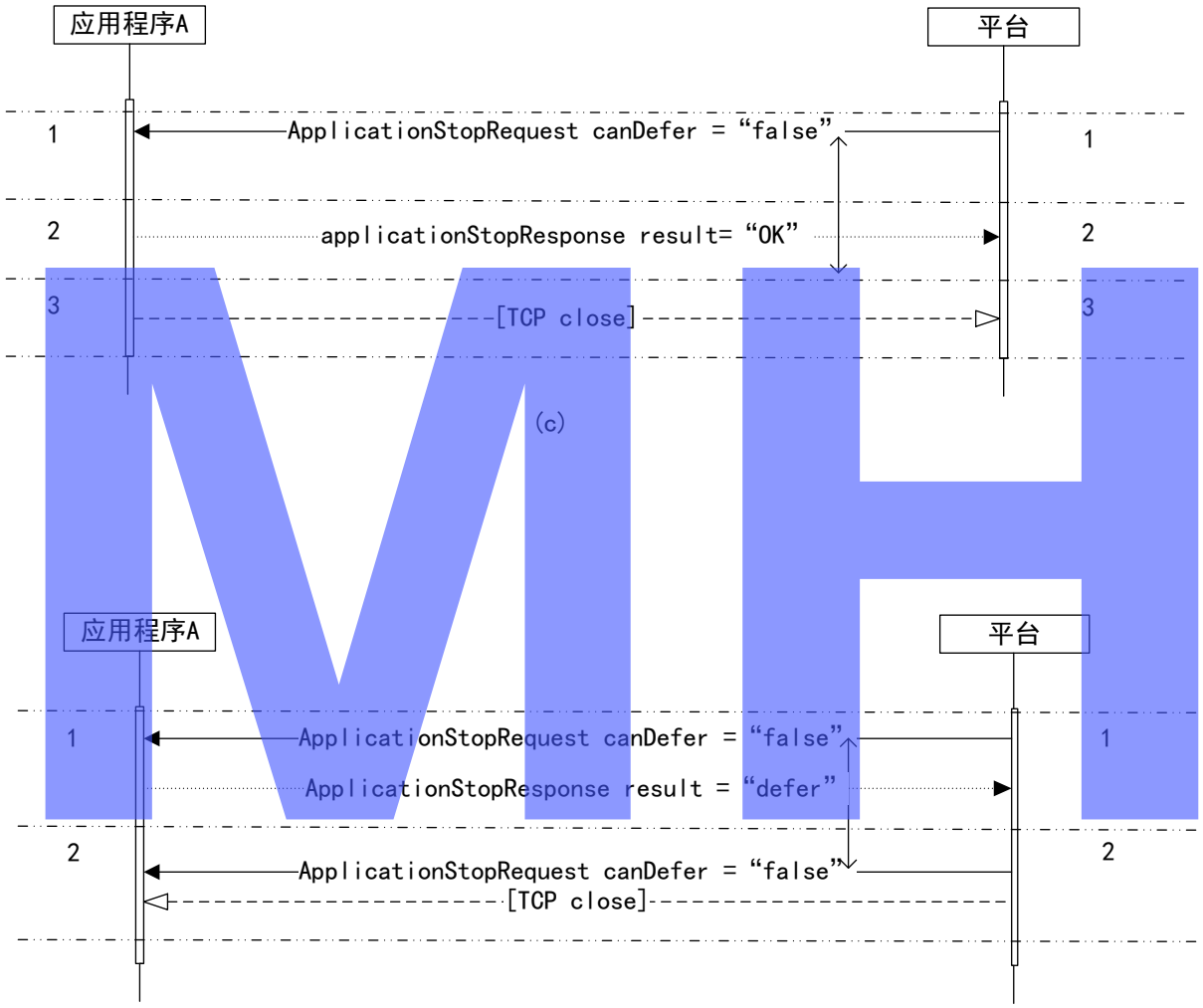
3) 步骤“3”，应用程序关闭平台的连接并且终止。

d) 平台不允许应用程序延迟退出，但应用程序需要延迟的情况。分为：

1) 步骤“1”，平台请求应用程序停止，并将应用程序置为 aSpg 状态，应用程序回复平台需要延迟退出；

2) 步骤“2”，平台关闭连接并且将应用程序置为 aZom 状态，然后断开与应用程序的连接并释放相关的所有资源。

应用停止指令请求的消息示例参见附录A.7。



(d)

图 4(续)

3.6 平台令牌失效和设备会话失效

3.6.1 应用程序在使用平台期间，应和平台保持连接。如果应用程序与平台的连接中断，与该连接相对应的设备令牌应立即失效。如果应用程序使用<byeRequest>要求终止与平台的连接，则其设备令牌也应失效。

3.6.2 使用失效令牌的设备连接是无效的。失效的设备会话在 PltSockMaxIdleTime[6]内依然保持连接状态。在此期间，不允许再往这个连接上发送消息。如果应用程序试图在无效会话上发送信息，则平台认为该消息是非法消息并回复。

3.7 消息头

3.7.1 CUPPS 信息交互协议提供多个版本的消息头。消息头遵循 vvllllllll[xxx]的基本格式，每一位为[A～Z，0～9]对应的 ASCII 字符，其中：

——vv 为用 2 个字节表示的消息头版本；

——llllllll 为一个 8 个字节长的数据表示消息体长度，数据的内容可用大写或小写字母表示，即 0123abcd 和 0123ABCD 是一样的；

——[xxx]为消息头版本对应的扩展数据。

表 2 为消息头格式。

表2 平台消息头格式

版本	状态	备注
01	支持的	2 个字节消息头版本，后跟 8 字节的长度信息。

3.7.2 从会话连接尝试读取信息时，消息头的长度指定了应读取的字节数。当写入会话连接时，长度信息应按照对应版本规定的规则进行计算。

3.7.3 如果接收方接收到不支持的消息头版本，应使用 01 版本消息头向发送方回复错误信息，并终止该会话。

3.7.4 消息头版本 01 定义最大消息长度，即 PltMaxMsgSize[7]字节(大约 16MB)。由于会话使用 8 字节字段的消息长度，前两个字节的长度都被置为 0。

示例：消息头版本 01 错误样例如下：

1. 01FF00003F<消息体长度 FF00003F 越界，最大支持 FFFFFF/>
2. 01 . . . . . [消息头版本]
3. 00 . . . . . [00 是可用的，FF 会导致超过 PltMaxMsgSize 错误]
4. 00003F[消息体的长度信息]

3.7.5 使用版本 01 时，读取或写入连接时应遵循以下规则：

——当往连接中写数据时，长度信息为所有消息体字节数，消息体使用 UTF8 编码方式；

——当从连接读取数据时，应按消息头中定义的消息体长度信息读取指定长度数据。消息体使用 UTF8 编码方式。

示例：消息头版本 01 正确样例如下：

1. 0100000013 <CUPPS>body</CUPPS>
2. 01 . . . . . [消息头版本 01]
3. 00000013 . . . . . [消息体长度为 19 个字节]
4. <CUPPS>body</CUPPS>[消息体]

3.8 流协议

6) 见表 3。

7) 见表 3。

3.8.1　CUPPS 连接使用简单标准流协议传输 UTF8 编码的 XML 信息。每个消息均以消息头开始，按字节从流中读取。

3.8.2　图 5 和图 6 分别为接口发送和接收消息状态机，消息读取端应按字节读取数据。消息接收的最大时长为 PltStreamAccumTime[8)]。

3.8.3　当从连接中读取消息头时，接收方应按位读取，一旦碰上无效字符，则立即停止读取。

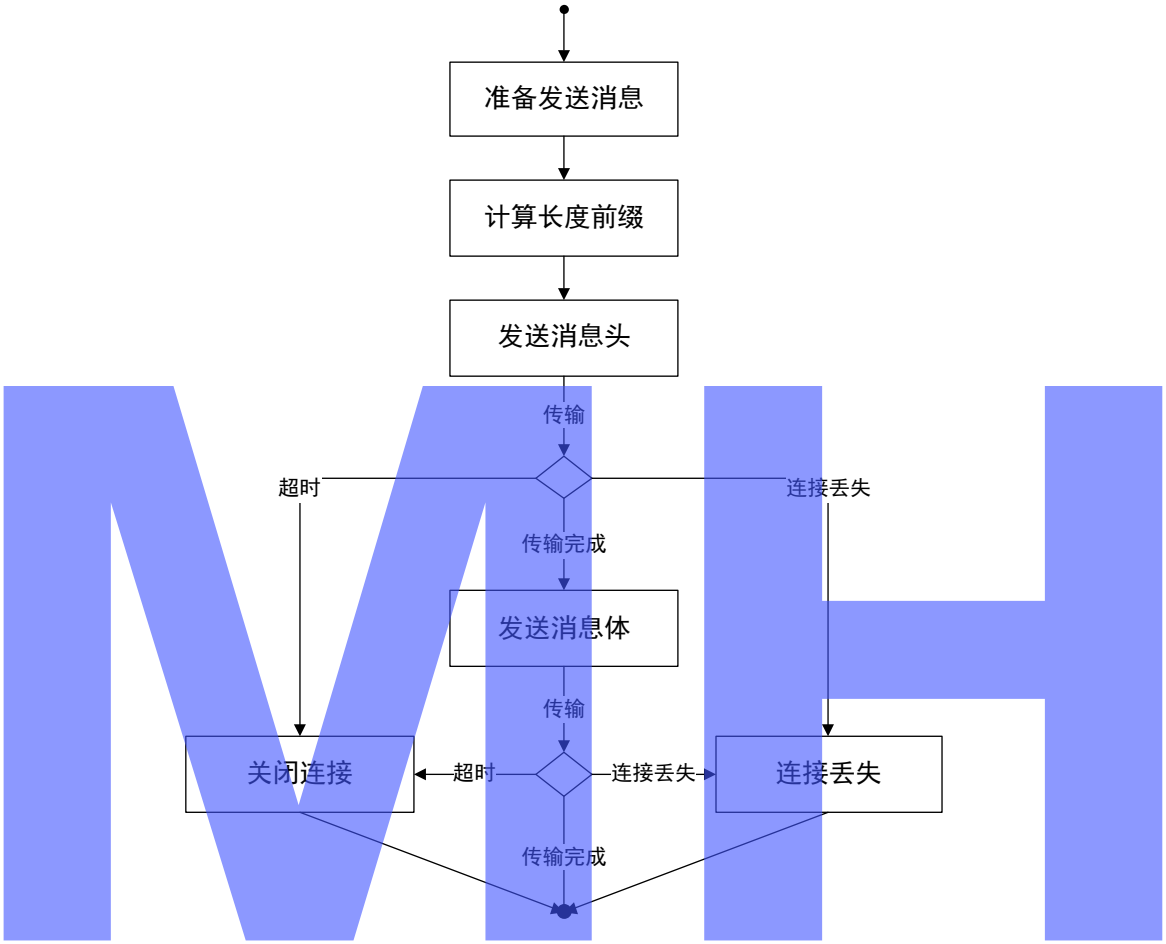


图5　消息发送状态机

8)　见表 3。

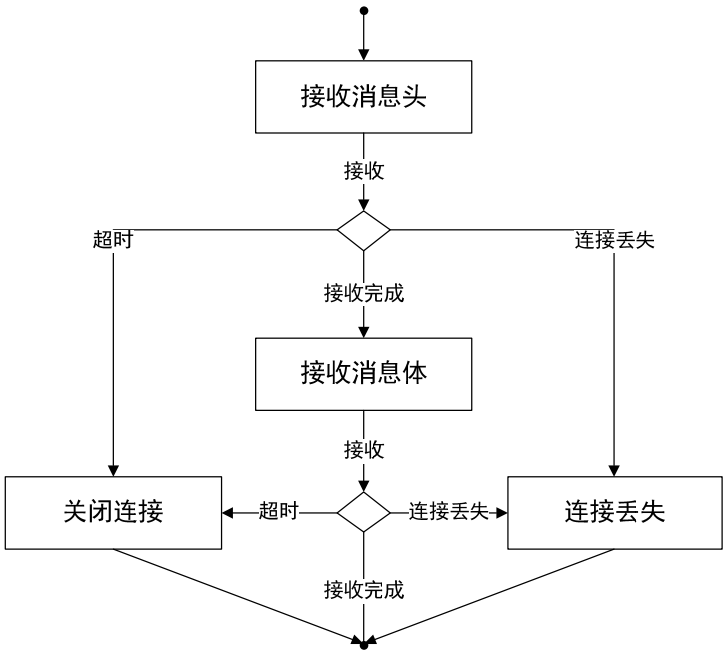


图6　消息接收状态机

3.8.4　应用程序和平台之间的交互消息应使用先进先出的顺序原则处理，包含应用程序断开请求的流消息应按其在流中出现的顺序处理。

3.8.5　平台与应用之间的消息交互方式分为两种：请求/应答与通知。在请求/应答方式下，请求需要应答确认。交互支持同时发送多个请求。请求消息包含 messageID 属性，应答消息的 messageID 与请求信息的 messageID 一致。5.1.5 详细规定了 messageID 的定义与生成规则。在通知方式下，消息从发送端传输到接收端，无需回应。

3.9　接口消息里的二进制数据

接口交互消息中的二进制数据应使用Base64方式进行编码、传输。

3.10　平台参数

应用程序在启动之前应设置的平台参数见表3。

表3　平台参数

参数	定义	功能描述	建议值	时间跨度
AppAthTime	应用程序认证时长	平台认证应用程序的最大时长	5.0s	开始：平台接收到来自应用程序的完整认证请求；结束：平台开始发送对应用程序认证响应
AppSpgTime	应用程序停止所需时长	从收到平台关闭指示后关闭程序所需的最大时长	45.0s	开始：平台命令应用程序关闭的时刻；结束：应用程序关闭完成的时刻

表 3（续）

参数	定义	功能描述	建议值	时间跨度
AppStgTime	应用启动时长	平台启动应用程序所消耗的最大时长，包含运行环境的准备和应用程序本身启动的时长	30.0 s	开始：终端用户点击应用程序对应菜单项的时刻；结束：应用程序启动的时刻
AppStpTime	应用已停止时长	平台在应用程序退出后，进行清理操作的最大时长	30.0 s	开始：应用程序终止的时刻；结束：平台清理完成的时刻
DevAcqMaxTime	请求设备的最大时长	处理<deviceAcquireRequest>消息的最大时长	30.0 s	开始：<deviceAcquireRequest>完全被接收的时刻；结束：<deviceAcquireResponse>完全被发送出去的时刻
DevLkdTime	锁定设备时长	设备被锁定的最大时长，超过这个时间设备将进入 dStd 状态	60.0 s	开始：设备进入 dLkd 的时刻；结束：当前时刻
DevNormTime	读设备读取时长	读设备的一个特定参数。如果不间断读取信息的设备在 DevNormTime 定义的时间内读取到多个数据值可用，则平台只将最后读取到的数据发送给应用	5.0 s	开始：最后一个读设备的消息被发送到应用程序的时刻；结束：当前时刻
DevPollMaxFreq	设备最大查询时长	平台允许应用程序轮询设备状态的最大频率。过多轮询会导致平台回复 pollingTooFast 消息	5.0 s	
DevSpgTime	设备停止时长	设备停止所需的最大时长。如果设备停止超时，则进入 dZom 状态	30.0 s	开始：设备进入 dSpg 状态的时刻；结束：当前时刻
DevStgTime	设备启动时长	正常启动设备所需的最大时长。如果设备启动超时，则从 dStg 状态进入 dZom 状态。	60.0 s	开始：设备进入 dSpg 状态的时刻；结束：当前时刻
MaxDevLockTime	锁定设备的最大时长	平台处理<deviceLockRequest>消息的最大时长	5.0 s	开始：平台接收到<deviceLockRequest>的时刻；结束：平台返回<deviceLockResponse>的时刻
MaxDevStsTime	获取设备状态的最大时长	平台处理<deviceStatusRequest>消息的最大时长	5.0 s	开始：平台接收到<deviceStatusRequest>的时刻；结束：平台返回<deviceStatusResponse>的时刻

表 3（续）

参数	定义	功能描述	建议值	时间跨度
MaxDevUnlTime	解锁设备的最大时长	平台处理<deviceUnlockRequest>消息的最大时长	2.0 s	开始：平台接收到<deviceUnlockRequest>的时刻；结束：平台返回<deviceUnlockResponse>的时刻
MaxMessageID	平台消息 ID 的最大值	平台端发起消息的最大 ID 值	0xffffffff	
MaxSpgDeferTimes	推迟停止的最大次数	应用程序推迟关闭的最大次数	4 次	
MinMessageID	消息 ID 的最小值	应用程序端发起消息的最小 ID 值	0x00000001	
MinPlatformMsgID	平台消息 ID 最小值	平台端发起消息的最小 ID 值	0xffff0000	
MsgIDListLen	消息 ID 清单的长度	最近发送或接收消息 ID 值的最小条数	255 条	
PGSMin	全局永久存储的最小值	平台为每个航空公司提供的全局存储空间的最小值	1 GB	
PLSMin	本地永久存储的最小值	平台为每个航空公司提供的本地存储空间的最小值	1 GB	
PNSMin	网络存储最小值	平台为每个航空公司提供的持久网络存储空间的最小值	1 GB	
PltAltMaxDlvTime	传递警告消息的最大时长	平台向所有工作站发送警报消息的最大时长	600.0 s	
PltAppStrMin	平台应用程序存储最小值	平台为每个航空公司应用程序提供的存储空间的最小值	5 GB	
PltLogMinStr	存储平台日志文件的最小时长	平台为每个航空公司提供日志存储时间的最小值	5 天	
PltMaxMsgSize	最大消息长度	接口消息的最大长度	0x00ffffff 字节	
PltMsgHdrSize	消息头大小	消息头的字节数	可变的	
PltRepGran	报告时间间隔	平台报告基于时间的活动的时间间隔	60.0 s	开始：平台上一次报告的时刻；结束：当前时刻

表 3（续）

参数	定义	功能描述	建议值	时间跨度
PltSockMaxIdleTime	连接最大空闲时间	PltSockMaxIdleTime用于以下两种情况：(a)连接建立之后，未接收到任何字符之前的时间段；(b)客户端发送<byeRequest>或<deviceReleaseRequest>指令准备结束会话之后，关闭连接之前的时间段	600.0 s	开始：a)连接建立后未接收到任何字符前，(b)客户端发送<byeRequest>或<deviceReleaseRequest>之后，关闭连接之前；结束：当前时刻
PltStreamAccumTime	消息流接收时间	平台与应用之间，同一消息的字节发送之间，允许的最长空闲时间。当接收端获取到任一字节时，PltStreamAccumTime 计时器复位。当接收端接收缓冲区被清空时，PltStreamAccumTime 计时器启动。当 PltStreamAccumTime 计时器超时，则给对方发送<sessionErrorEvent>的<notify>消息	60.0 s	开始：从连接缓冲区中读取到第一个字节；结束：当前时刻
PltStreamOutMsgs	最大重复消息数量	同一个连接支持的重叠消息的最大值	5 条	
PltStreamOutMsgsZL	ZL 设备最大重复消息数量	ZL 设备的连接中支持的重叠消息的最大值。对于 ZL 设备，应用程序可在收到 logWriteResponse 消息之前，继续发送 logWriteRequest。当累积到 PltStreamOutMsgsZL 个未收到回复的请求时，应用程序应暂停发送 logWriteRequest	20 条	
PRMaxResponseTime	打印的最大响应时间	PR 设备处理<printRequest>消息的最大时长。如果在这段时间过后还没有开始打印，则平台端取消此次打印任务	120.0 s	
TSMin	最小临时存储空间	应用程序临时存储空间的最小值	1 GB	
UsrAthTime	用户认证时间	认证终端用户的最大时长	10.0 s	开始：接收到认证消息第一个字节的时刻；结束：当前时刻
UsrAthTries	用户认证次数	工作站连续尝试登陆均失败的最大次数	3 次	

表 3（续）

参数	定义	功能描述	建议值	时间跨度
UsrSpgTime	用户停止时间	用户对象退出的最大时长。一旦超时，工作站将进入 uZom 状态，在随后的清理结束后，应为下一个用户登陆做好准备	30.0 s	开始：用户开始退出 结束：用户对象清除的时刻
UsrSsTime	用户屏保时间	用户在工作中运行屏幕保护程序的最大时长.一旦超时，用户从 uSs 状态进入 uSpg 状态	600.0 s	开始：用户在工作站上最后一个活动的时刻 结束：当前时刻
UsrToTime	用户超时时间	用户的最大空闲时间。一旦超时，用户从 uStd 状态进入 uSs 状态	600.0 s	开始：用户在工作站上最后活动的时刻，如键盘输入、鼠标操作、设备使用等；结束：当前时刻
WsAltTime	工作站警告超时时间	工作站解除警告信息的最大时长	600.0 s	开始：工作站进入 wAlt 的时刻；结束：当前时刻
WsSpgTime	工作站停止时间	工作站退出的最大时长。一旦超时，工作站从 wSpg 状态进入 wZom 状态	30.0 s	开始：工作站进入 wSpg 的时刻；结束：当前时刻
对于与时间相关的参数，平台和应用程序应允许运行时间偏差在±1%之间。				

3.11 设备锁定与解锁

3.11.1 令牌锁与连接锁

3.11.1.1 令牌锁的锁定机制基于令牌，连接锁的锁定机制基于连接。对于连接锁，一旦一个连接成功锁定设备，则其他任何连接均不能锁定该设备。而对于令牌锁，使用同样令牌的不同连接可同时锁定设备。

3.11.1.2 采用连接锁时，设备的锁定状态从<deviceLockRequest>开始，到对应的<deviceUnlockRequest>结束。采用令牌锁时，设备的锁定状态从第一个<deviceLockRequest>开始，到成功锁定的最后一个<deviceUnlockRequest>结束。

3.11.1.3 图 7 为两种锁机制的示例，其中有两个应用程序连接 A 与 B，均希望使用设备 BC1。该图给出了应用程序与平台之间的交互，在交互开始之前，应用程序与平台均已成功建立连接。

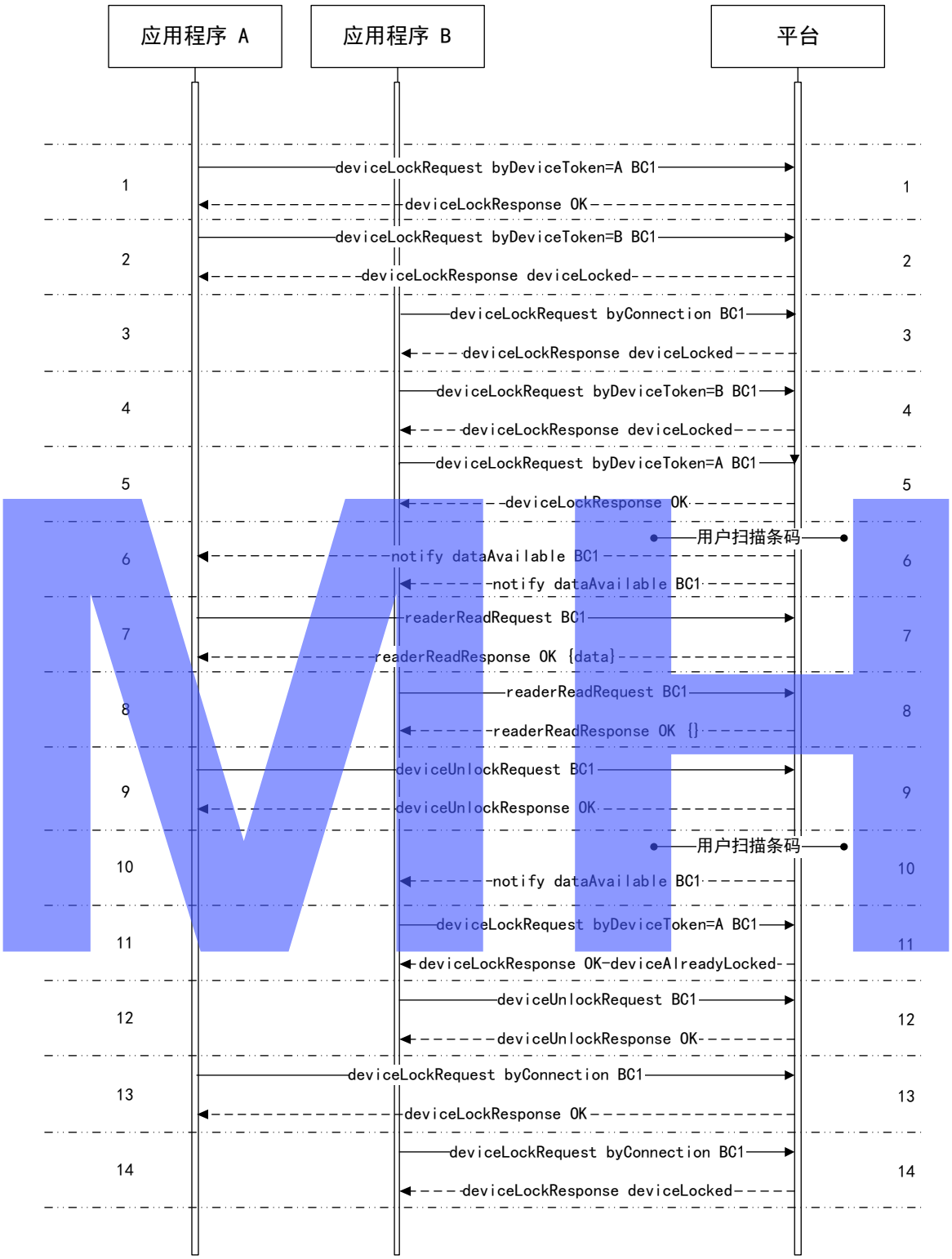


图7　令牌锁与连接锁锁定示例

3.11.1.4　图 7 中应用程序与平台之间的交互步骤如下所示。在每个步骤之后，记录设备持有的锁状态，其中“*”代表连接锁，“t=x”代表令牌为 x 的令牌锁：

a)　步骤“1”，应用程序 A 使用设备令牌 A 获得设备 BC1 的锁定。其中：

1) A = { BC1 t=A };
2) B = {};

b) 步骤“2”，应用程序 A 试图使用令牌 B 锁定设备。因为这个设备已经使用令牌 A 锁定，所以平台拒绝的该锁定请求。其中：
1) A = { BC1 t=A } ;
2) B = {};

c) 步骤“3”，应用程序 B 试图使用连接锁锁定设备。但因为这个设备已经使用令牌 A 锁定，所以平台拒绝了该请求。其中：
1) A = { BC1 t=A };
2) B = {};

d) 步骤“4”，应用程序 B 试图使用令牌 B 锁定设备。因为这个设备已经使用令牌 A 锁定，所以平台拒绝了该请求。其中：
1) A = { BC1 t=A };
2) B = {};

e) 步骤“5”，应用程序 B 试图使用令牌 A 锁定设备。因为设备已经被令牌 A 锁定，而且使用的同样的令牌，平台允许该锁定请求。现在应用程序 A 和 B 共享设备 BC1 的锁。其中：
1) A = { BC1 t=A };
2) B = { BC1 t=A};

f) 步骤“6”，用户使用读设备 BC1 扫描条码。设备已经被令牌锁定，所有与此设备保持令牌锁的连接，都会收到有可用数据的通知。其中：
1) A = { BC1 t=A };
2) B = { BC1 t=A };

g) 步骤“7”，应用程序 A 试图从 BC1 读取数据。当数据返回到应用程序 A 时，平台清除 BC1 的数据。其中：
1) A = { BC1 t=A};
2) B = { BC1 t=A };

h) 步骤“8”，应用程序 B 试图从 BC1 读取数据。因为在步骤“7”中应用程序 A 的读取请求后已经清除了数据，所以返回结果中，并不包含读取到的数据；

i) 步骤“9”，应用程序 A 释放它对 BC1 的锁。其中：
1) A = {};
2) B = { BC1 t=A };

j) 步骤“10”，用户使用读设备 BC1 扫描条码。设备已经被令牌锁锁定。所有与此设备保持令牌锁连接，都会收到有可用数据的通知，而此时只有应用程序 B。其中：
1) A = {};
2) B = { BC1 t=A };

k) 步骤“11”，应用程序 B 试图使用令牌 A 锁定 BC1。因为应用程序 B 已经用令牌 A 锁定 BC1，所以平台回复提示应用程序已经锁定该设备。其中：
1) A = {};
2) B = { BC1 t=A };

l) 步骤“12”，应用程序 B 对 BC1 解锁。其中：
1) A = {};
2) B = {};

m) 步骤“13”，应用程序 A 使用连接锁锁定设备。其中：

1) A = { BC1* };
2) B = {};

n) 步骤“14”，应用程序 B 试图使用连接锁锁定 BC1。因为应用程序 A 连接已经使用连接锁锁定该设备，所以平台拒绝了该请求。其中：
1) A = { BC1* };
2) B = {}。

3.11.2 设备锁定超时

一旦应用程序锁定设备，平台为应用程序启动一个时长为DevLkdTime[9]的计时器。如果定时器超时，设备连接仍没有I/O（Input/Output,输入输出）交互，则该应用程序对设备的锁定超时，平台立即向所有获得该设备连接的应用程序发送<deviceLockExpiredEvent>消息。锁定超时的示例参见附录A.2。

3.11.3 子设备锁定规则

子设备锁定规则只适用于标准模式。当父设备被锁定时，从属于该父设备的子设备也被锁定。

子设备锁定的示例参见图8～图12。其中应用程序A和B与平台通信，每个应用程序都要使用BG1设备或它的子设备BC1和DD1。假定所有图中每个应用程序和平台已成功连接到设备。

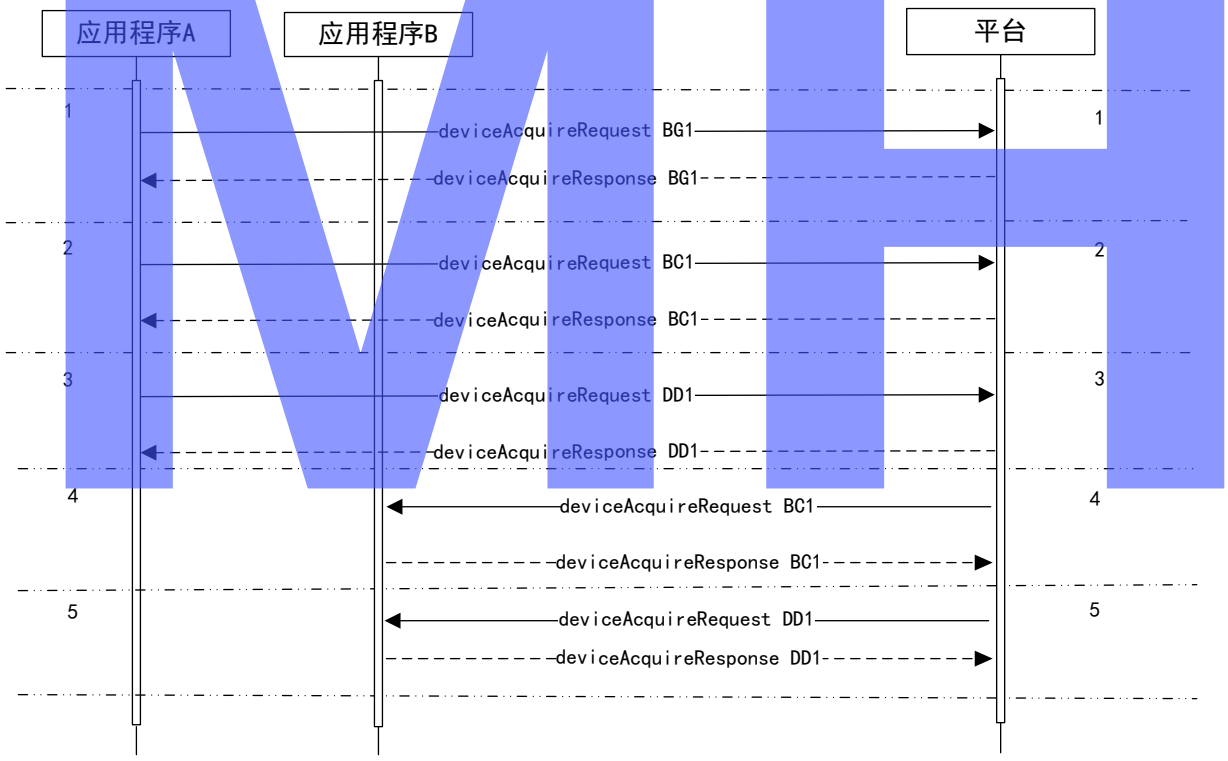


图8 子设备锁定示例(1)

9) 见表 3。

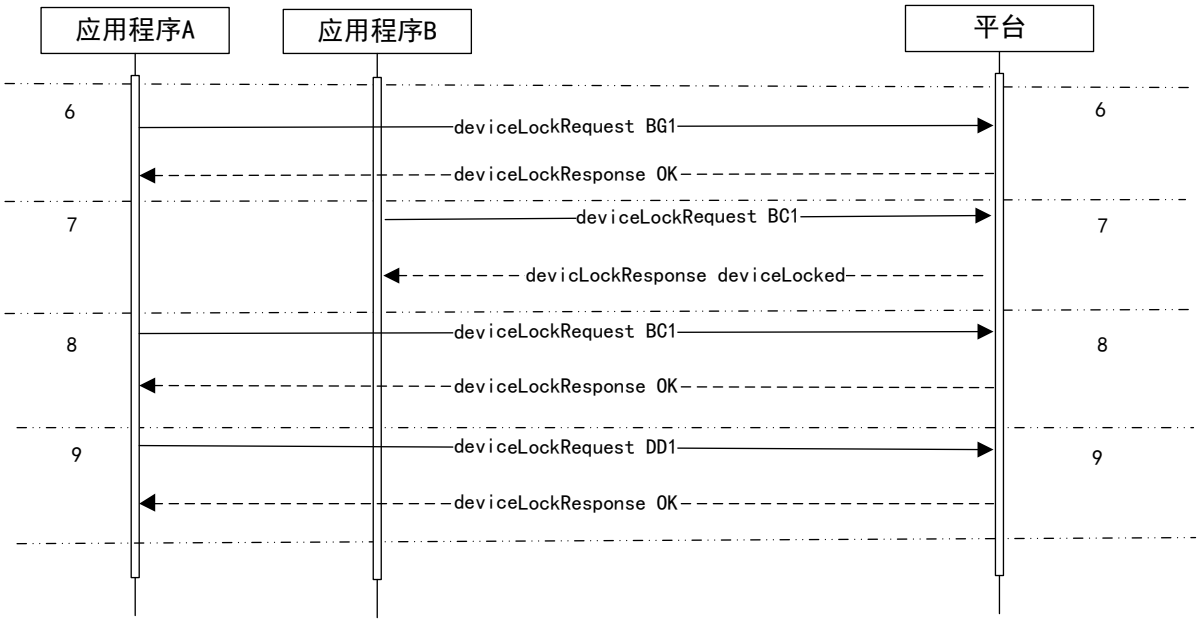


图9　子设备锁定示例(2)

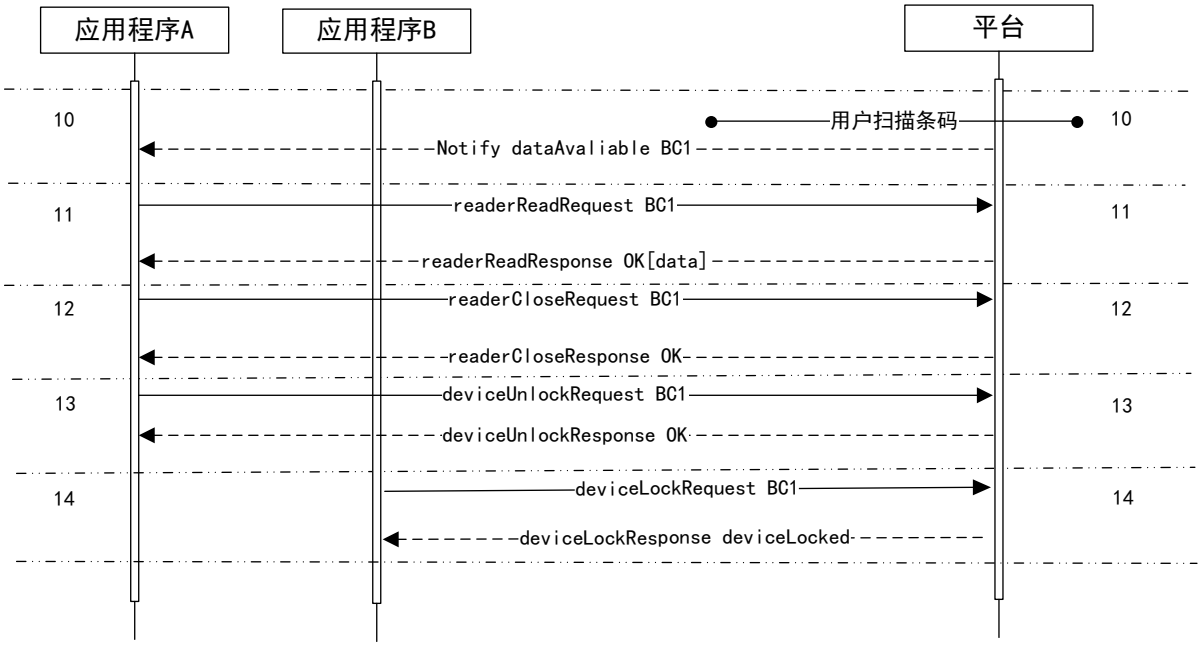


图10　子设备锁定示例(3)

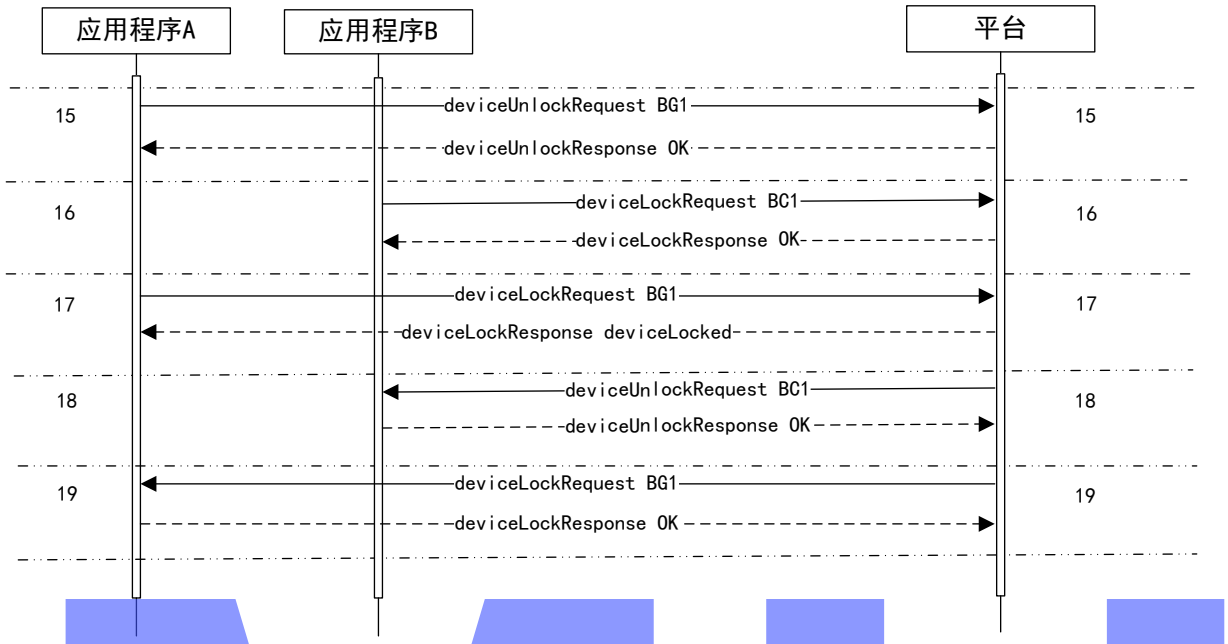


图11　子设备锁定示例(4)

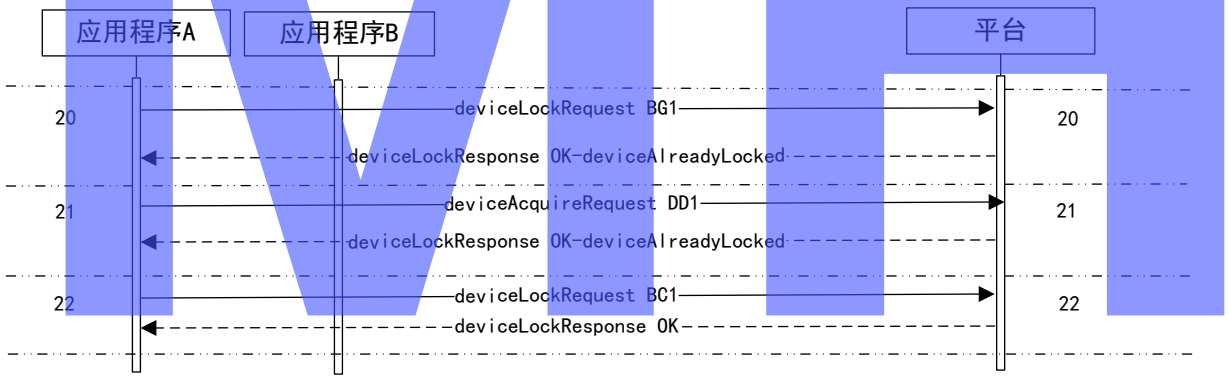


图12　子设备锁定示例(5)

每个步骤如下所示：

注：{}内为应用程序已经获取的锁，其中“*”代表显性锁，即设备被明确锁定；没有“*”代表隐性锁，即设备没有被明确锁定，但其父设备被锁定了。

a) 步骤“1”～“3”，应用程序 A 获得与设备 BG1、BC1 和 DD1 的连接。其中：
   1) A = {}；
   2) B = {}；
b) 步骤“4”～“5”，应用程序 B 获得与设备 BC1、DD1 的连接。其中：
   1) A = {}；
   2) B = {}；
c) 步骤“6”，应用程序 A 锁定设备 BG1。因为 BG1 包含子设备，应用程序 A 的也会隐式地锁定其子设备 DD1 和 BC1。其中：

1) A = { BG1*, BC1, DD1 };
2) B = {}

d) 步骤“7”，应用程序 B 试图锁定 BG1 的子设备 BC1。因为应用程序 A 已锁定了 BG1 及其子设备 DD1 和 BC1，所以该锁定请求被拒绝。其中：
1) A = { BG1*, BC1, DD1 };
2) B = {};

e) 步骤“8”～“9”，应用程序 A 显式地锁定 BG1 的子设备 BC1 和 DD1。因为应用程序 A 已锁定 BG1，因此该锁定请求不会出现错误。其中：
1) A = { BG1*, BC1*, DD1* };
2) B = {};

f) 步骤“10”，用户使用 BG 设备扫描条码。应用程序 A 持有 BC1 的锁，平台通过 BC1 的连接发送数据可用通知。其中：
1) A = { BG1*, BC1*, DD1* };
2) B = {};

g) 步骤“11”，应用程序 A 从平台读取条码数据。其中：
1) A = { BG1*, BC1*, DD1* };
2) B = {};

h) 步骤“12”，应用程序 A 关闭 BC1 的输入。应用程序 A 保持 BG1 及其所有子设备的锁定。其中：
1) A = { BG1*, BC1*, DD1* };
2) B = {};

i) 步骤“13”，应用程序 A 解锁 BC1。因为应用程序 A 仍然持有 BG1 的锁，平台继续锁定 BG1 及其所有子设备。其中：
1) A = { BG1*, BC1, DD1* };
2) B = {};

j) 步骤“14”，应用程序 B 试图锁定 BC1。因为应用程序 A 持有 BG1 和其所有子设备的锁，因此平台拒绝了该请求。其中：
1) A = { BG1*, BC1, DD1* };
2) B = {};

k) 步骤“15”，应用程序 A 解锁 BG1。平台释放了 BG1 的锁，但是仍然保持对 BG1 子设备 DD1 的锁。在步骤(8-9)，应用程序 A 显式锁定了 BC1 和 DD1，但是在步骤“13”显式解锁了 BC1。因此，应用程序 A 只持有一个 DD1 的锁。其中：
1) A = { DD1* };
2) B = { };

l) 步骤“16”，应用程序 B 试图锁定 BC1，子设备 BC1 与父设备 BG1 均没有被锁定，因此平台通过了锁定申请。其中：
1) A = { DD1* };
2) B = { BC1* };

m) 步骤“17”，应用程序 A 试图锁定 BG1。因为应用程序 B 拥有一个 BG1 子设备的锁，因此平台拒绝本次锁定尝试。其中：
1) A = { DD1* };
2) B = { BC1* };

n) 步骤“18”，应用程序 B 解锁 BC1。其中：
1) A = { DD1* };

2) B = {};

o) 步骤“19”，应用程序 A 试图锁定 BG1。因为 BG1 未锁定，而子设备已经被应用程序 A 锁定，与锁定 BG1 不冲突，所以平台准许了该请求。其中：

1) A = { BG1*, BC1, DD1* };

2) B = {};

p) 步骤“20”，应用程序 A 试图锁定 BG1。因为应用程序 A 已经拥有一个 BG1 的显性锁，因此平台向应用程序 A 返回已经拥有锁定的通知。其中：

1) A = { BG1*, BC1, DD1* };

2) B = {};

q) 步骤“21”，应用程序 A 试图锁定 DD1。因为应用程序 A 已经拥有一个 DD1 的显性锁，因此平台向应用程序 A 返回已经拥有锁定的通知。其中：

1) A = { BG1*, BC1, DD1* };

2) B = {};

r) 步骤“22”，应用程序 A 试图锁定 BC1。因为应用程序 A 已经拥有一个 BC1 的隐性锁，因此平台并没有返回一个错误的通知，而返回了一个成功的结果。同步骤“8”～“9”的结果相同。其中：

1) A = { BG1*, BC1*, DD1* };

2) B = {}。

3.12 令牌

3.12.1 总则

令牌由16个可打印的ASCII字符组成。令牌一般用于验证，作为密钥使用，可由平台或应用程序生成。在CUPPS交互消息中，令牌存储于<token>节点。

3.12.2 平台生成令牌

3.12.2.1 本文件对平台生成令牌的方式不予规定，但平台生成的令牌应满足接口中关于<token>节点的定义。

3.12.2.2 平台端定义的令牌应保证对所有应用程序的连接是唯一的且作为<authenticateResponse>消息的一个属性提供给应用程序。

3.12.3 应用生成令牌

3.12.3.1 本文件对应用生成令牌的方式不予规定，但生成的令牌应满足接口中关于<token>接口的定义。

3.12.3.2 应用生成的令牌分别为：

——事件令牌：用于应用程序对事件的订阅。应用程序在发送<authenticateRequest>消息时包含事件令牌属性。如果其他应用程序使用该令牌发送<subscriptionAddRequest>消息来订阅，则当该应用程序发生特定事件时，所有订阅过的应用程序都会收到平台对该事件的通知；如有一个特定的应用程序注册所有应用程序的事件，则这个特定的应用程序就可收到所有应用程序所发生的事件的通知；

——安全令牌：用于获取日志。当应用程序使用 ZL 设备进行写操作时，应先发送包含安全令牌的<logOpenRequest>消息，然后再往 ZL 设备进行写操作。如果应用程序要重新获取以前的日志信息，则应提供相应的令牌。

注：安全令牌是为了实现读写机制，写日志一方提供令牌，而读取日志的一方应提供一致的令牌。

3.12.4 设备查询

应用程序应先通过<authenticateResponse>消息获得设备令牌，才可使用平台进行设备查询服务。应用程序可根据条件查询相应的设备信息，表4给出了可用的查询条件。设备查询的示例见附录A.9。

表4 查询条件

条件	描述
设备类型	仅限于指定的设备类型
附近设备	仅限于附近的特定节点。该信息基于系统配置
设备名称	仅限于指定的设备名称

3.12.5 异常处理

当开发人员捕获到运行时的异常时，应将其通过<exceptionEvent>消息报告给系统管理员或其他人员进行处理，<exceptionEvent>消息包含此异常信息，例如发生异常的用户、工作站、设备名称、应用程序名称等。<exceptionEvent>消息的exceptionSource属性描述了异常是发生在平台还是在应用程序，arg属性提供具体的异常类型，例如空指针异常，主机连接异常等。消息正文提供发生异常时具体的异常信息。

4 数据交换接口消息处理规则

4.1 消息处理规则

4.1.1 平台端发送的请求和通知

平台端可向应用程序发送消息和通知。示例参见附录A.3。示例1为平台向应用程序发送的通知，通知内容为设备有可读数据；示例2为平台向应用程序发送的通知，通知内容为设备的状态已改变。

4.1.2 非法消息处理

4.1.2.1 非法消息的判定可在消息交互的发送方，也可在接收方。消息交互需遵循特定的顺序，违反顺序的消息即为非法消息。当应用程序或平台发送非法消息时，产生<illogicalMessageErrorEvent>事件消息。当接收端检查到非法消息时，使用 Base64 编码该非法消息，发送<illogicalMessageErrorEvent>事件消息给接收端，并立即断开连接。

4.1.2.2 应用程序与平台交互时，如果应用程序与设备连接产生非法消息，则相关的设备令牌将失效，连接断开，应用程序需重新建立设备连接并重新进行设备的初始化操作。

4.1.2.3 在非法消息的回复中，应包含合法消息名称的完整列表。示例参见附录 A.10。

4.1.3 连接 Keep-Alive 设置

应用程序和平台之间的所有连接，应使用TCP层的Keep-Alive机制。示例为一个使用Keep-Alive的C语言示例。其他类型的语言，也应提供相似的调用，可根据实际的编程语言进行设置。

示例：setsockopt() 函数 Keep-Alive 设置

```
//Set the option to on
int optval = 1;
setsockopt(sock, SOL_SOCKET, SO_KEEPALIVE, &optval, sizeof(optval));
```

4.1.4 流错误和解析错误

应用程序应确保传输到平台的消息是合法的XML消息。应用程序和平台通信时错误流的处理流程见图13。

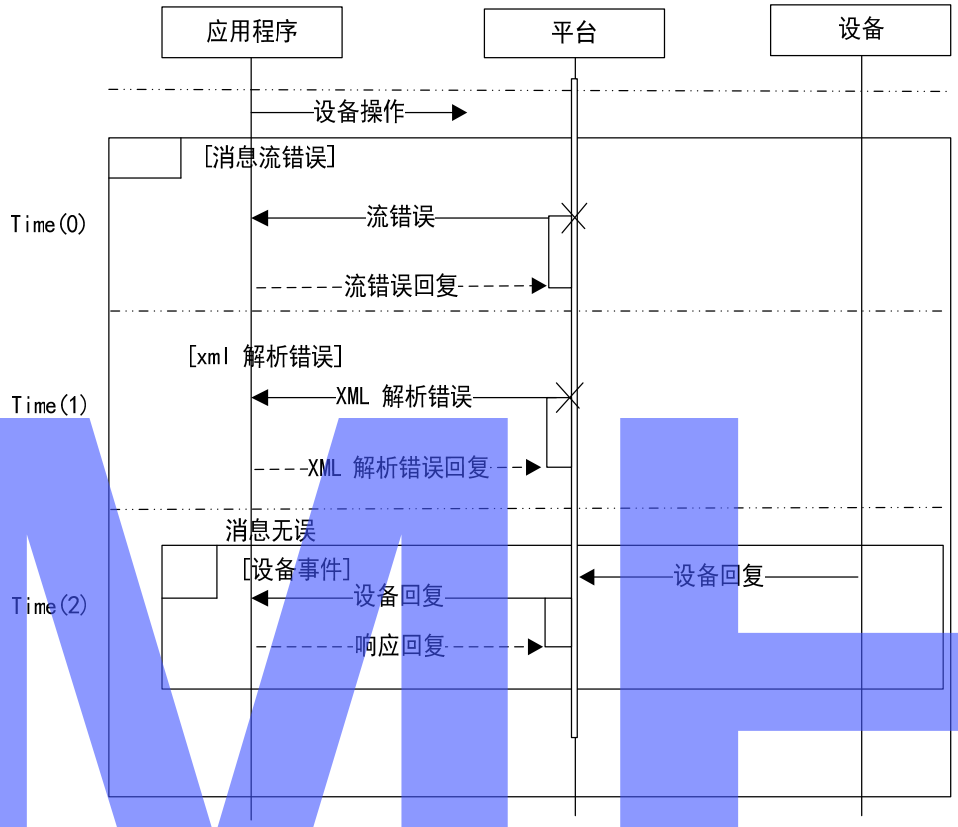


图13 流错误示例

在图13中：

——在 Time(0)之前，应用程序向平台发送一个消息。在这个示例中，该消息是一个设备命令；

——在 Time(0)，平台确认接收到的消息是否完整，即是否产生了流错误。如果发生流错误，则返回应用程序流错误消息，关闭连接。应用程序应根据需要重新建立连接；

——在 Time(1)，平台使用应用程序已经选择的接口版本对接收的消息进行解析。如果发生 XML 解析错误，则给应用程序返回一个 XML 解析错误消息，并关闭连接。应用程序应根据需要重新建立连接；

——在 Time(2)，没有发生流错误或 XML 解析错误。如果检测到一个设备事件，应将该设备事件的通知实时发送给应用程序。

4.1.5 messageID 处理逻辑

在应用程序与平台的交互中，请求与应答消息的messageID应一致。针对每个连接，平台和应用都应保留最近发送或接收的消息清单，这个清单的长度为MsgIDListLen[10]。应用程序发起消息的messageID

10) 见表 3。

范围为MinMessageID[11]至MinPlatformMsgID[12]。平台端发起消息的messageID范围为MinPlatformMsgID[13]至MaxMessageID[14]。

发送消息的messageID取值为上一条发送消息的ID值加1。当应用程序发起的messageID到达MinPlatformMsgID[15]-1时，重置为MinMessageID[16]。当平台发起的messageID到达MaxMessageID[17]时，重置为MinPlatformMsgID[18]。

应用程序和平台之间的每个连接都使用自己的messageID顺序。一个连接上的messageID与其它连接的messageID无关。处理messageID的流程见图14。

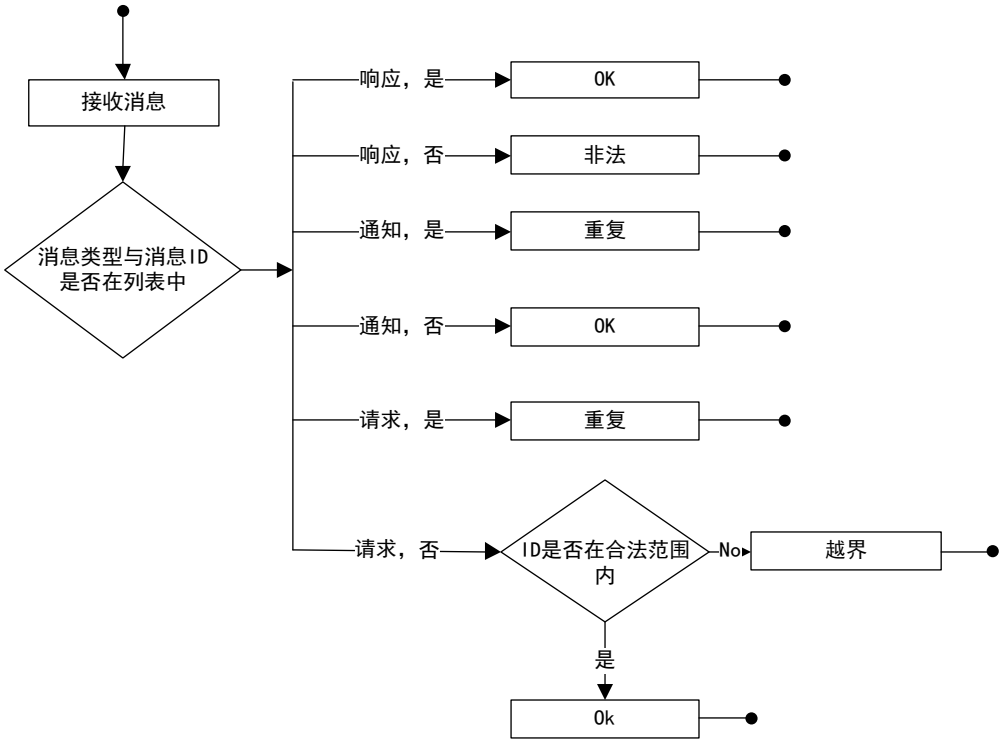


图14　messageID 的处理流程

在图14中：

——如果消息为响应消息，则 messageID 应已在最近处理的 messageID 列表里，其中：

- 如果在列表中找到该 messageID，则该消息可继续做处理；
- 如果在列表中没找到该 messageID，则发送一个<messageIDErrorEvent>，由消息的接收方断开连接；

——如果消息为通知，则 messageID 应不在该列表中，其中：

11）见表 3。
12）见表 3。
13）见表 3。
14）见表 3。
15）见表 3。
16）见表 3。
17）见表 3。
18）见表 3。

- 如果在列表中找到该 messageID，则发送一个<messageIDErrorEvent>，由消息的接收方断开连接；
- 如果在列表中没有找到该 messageID，则该消息可继续做处理；

——如果该消息为请求，则 messageID 应不在该列表中。其中：

- 如果在列表中找到该 messageID，则发送一个<messageIDErrorEvent>，由消息的接收方断开连接；
- 如果 messageID 不在该列表中，则检查 messageID 是否在有效范围内。如果该 messageID 在有效范围之外，则发送一个<messageIDErrorEvent>，由消息的接收方断开连接。发送方的 messageID 值应为上一次 messageID 值加 1。

4.2 接口状态机

4.2.1 状态机的状态

接口状态机的四种状态见图15，分别为：

——启动状态，表示应用程序和平台之间接口实例的启动。在启动阶段，平台可重定向与应用程序的连接。因此，该图给出了绕过设备和平台使用直接进入退出状态的一条连接；

——设备使用状态，表示与设备相关的接口正在被使用；

——平台使用状态，表示与设备无关的接口正在被使用；

——退出状态，表示一个接口实例退出。

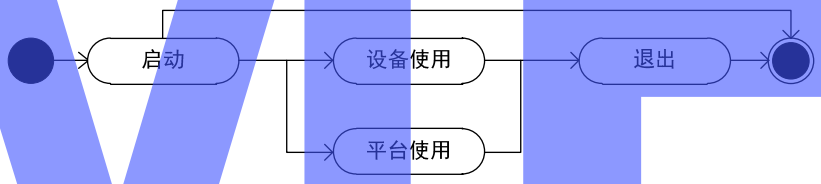


图15 接口状态机

4.2.2 启动状态

4.2.2.1 接口状态机的启动状态见图 16。在接口实例启动时，应用程序设置接口版本并进行平台认证。

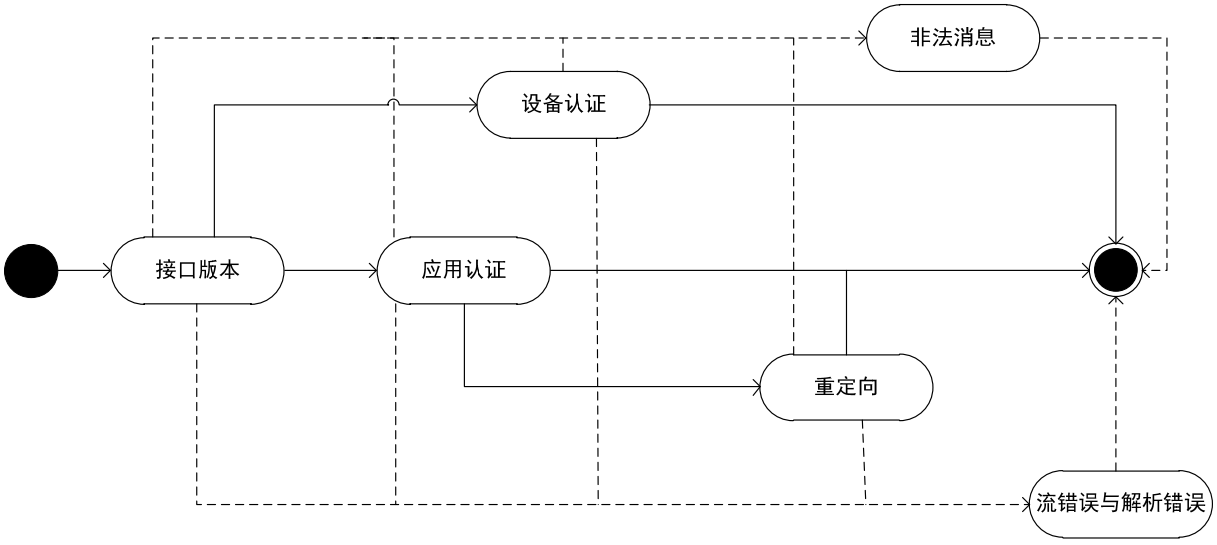


图16　接口状态机的启动状态

4.2.2.2　接口启动时，进入接口版本选择状态。应用程序在该状态内，确认选择适当的接口版本，随后发送<authenticateRequest>或者<deviceAcquireRequest>消息。

4.2.2.3　一旦设置接口版本，该连接应被认证。如果该连接与平台的管理部分(例如:非设备相关的功能)进行通信，则发送<authenticateRequest>消息。如果该连接与设备进行通信，则发送<deviceAcquireRequest>消息。

4.2.2.4　当应用程序接收到<authenticateResponse>消息时，首先解析该消息判断是否包含 redirect 元素。如果包含，则该状态机将立即进入重定向状态。

4.2.3　设备使用状态及模式

4.2.3.1　设备使用状态

接口状态机的设备使用状态见图17。一旦应用程序获得设备连接，应先设置接口模式来管理与设备之间的通信，然后设备进入相应的模式。

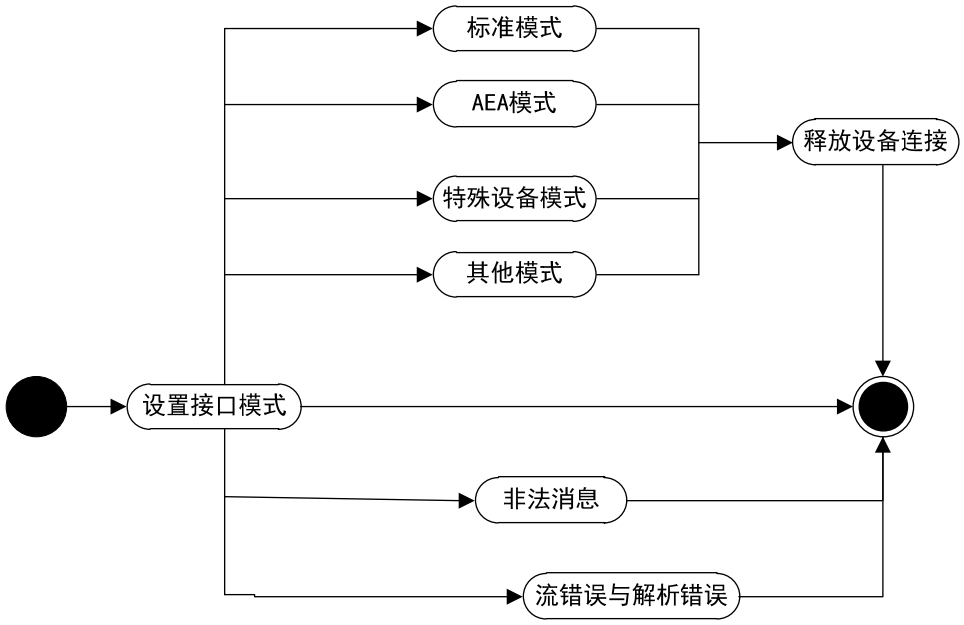


图17　接口状态机的设备使用状态

4.2.3.2　模式

4.2.3.2.1　标准模式

标准模式管理读设备、打印设备。

读设备状态见图18。接口首先进入未锁定状态。应用程序应先锁定该设备，才能进行与该设备的交互。

如果应用程序从设备读取数据，应先锁定该设备。当结束使用设备时，应解锁并释放这个设备。

打印设备状态见图19。接口等待应用程序发送消息来进行各种操作。

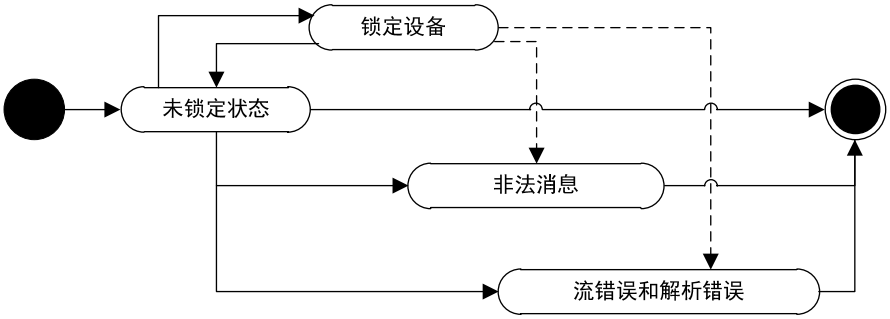


图18　读设备状态

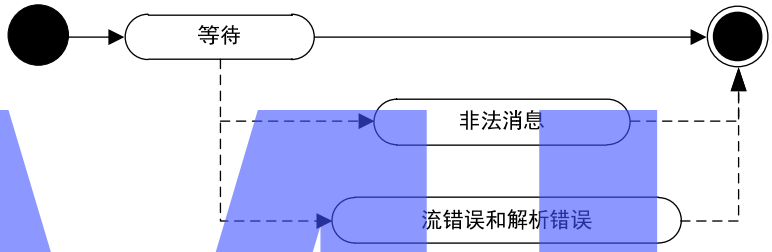


图19　打印设备状态

4.2.3.2.2　AEA 模式

AEA(Association of European Airlines)模式见图20。在AEA模式状态下，平台端等待应用程序发送消息来进行各种操作。当应用程序发送<deviceReleaseRequest>时，退出AEA模式状态。

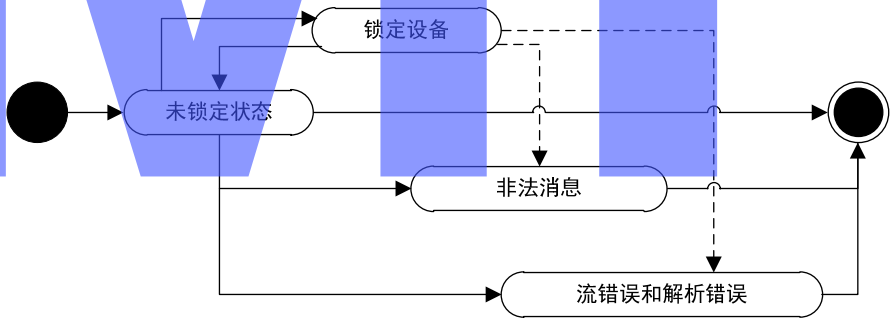


图20　AEA 模式

4.2.3.2.3　特殊设备模式

特殊设备模式见图21，包括：

——日志设备 ZL；

——IATA（International Air Transport Association，国际航空运输协会）消息设备 ZI。

特殊设备模式不支持锁定。试图锁定或解锁特殊设备会产生<illogicalMessageErrorEvent>消息。

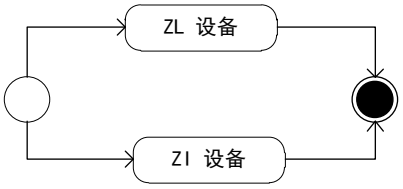


图21　特殊设备模式

ZL设备为应用程序提供日志记录接口。应用程序可使用该设备进行打开、关闭、读写等操作。

图22为ZL设备的接口状态机。应用程序操作设备之前应发送<logOpenRequest>打开ZL设备。当应用程序完成对设备的操作时，使用<logCloseRequest>关闭该设备。

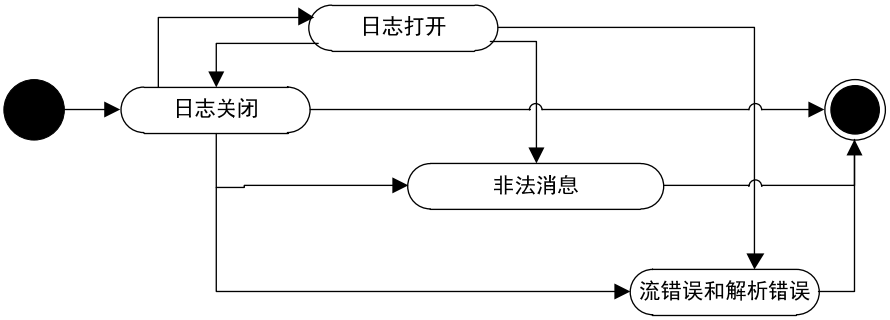


图22　ZL 设备

ZI设备为应用程序和平台之间交换IATA定义的或应用程序需要向平台发送的消息提供了一个标准接口，例如旅客信息报文及安检报文。平台可将应用程序发来的消息转发给机场端的特定系统，例如应用程序可通过ZI设备，将BSM(Baggage Source Message，行李报文信息)报文发送给平台端，平台端将BSM报文转发给行李分拣系统。本标准不规定平台与机场端特定系统的交互标准。

图23给出了用于ZI设备的接口状态机。平台端等待应用程序发送消息进行各种操作。

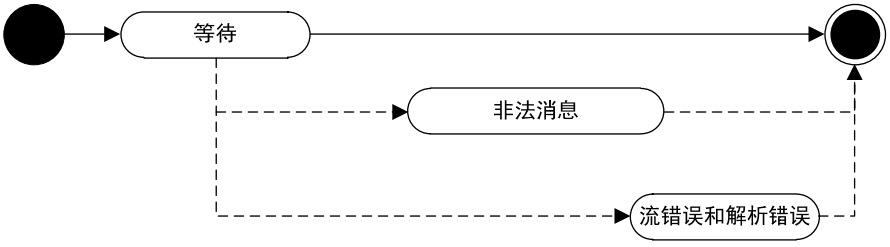


图23　ZI 设备

4.2.3.2.4　其他模式

其他模式与读设备模式类似，见图24。接口首先进入未锁定状态。应用程序应锁定该设备后，才能进行进一步与该设备的交互。

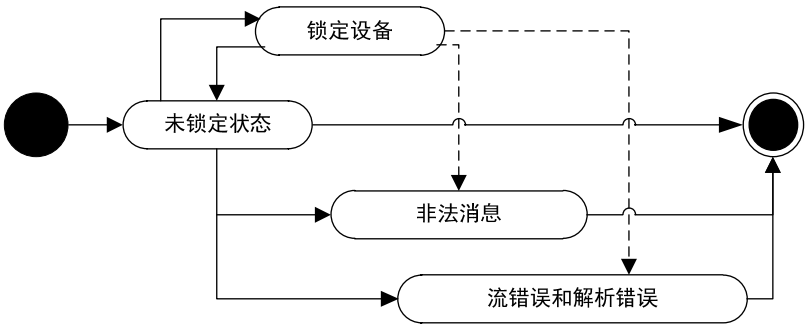


图24　其他模式

4.2.4　平台使用状态

平台启动时等待事件到来，事件触发时，接口状态机进入平台使用状态(见图25)。根据事件内容的不同，平台使用状态包括以下状态：

——警报状态：管理订阅事件的发布，可向工作站发送警报信息；

——通知状态：表示平台向应用程序发送特定的通知；

——平台引导应用程序退出状态：表示平台端发起的应用程序处于退出状态；

——应用程序退出状态：表示应用程序端发起的应用程序处于退出状态。

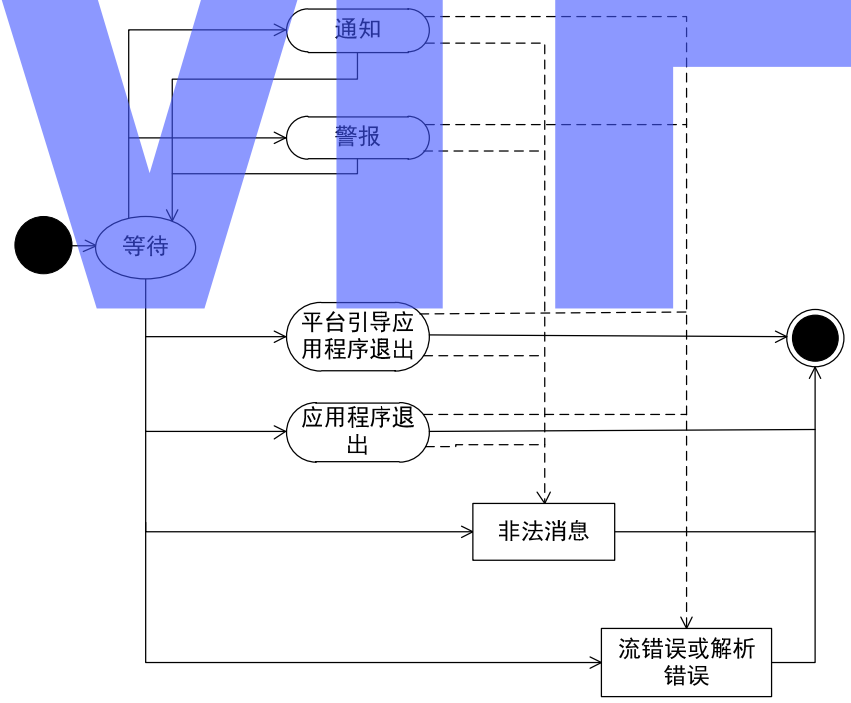


图25　接口状态机-平台使用

4.2.5 退出状态

退出状态表示连接被关闭。如果连接在一段时间AppSpgTime [19](平台连接)或者 DevSpgTime[20](设备连接)内没有被关闭，该连接将被平台关闭。

19) 见表 3。
20) 见表 3。

附 录 A
（资料性附录）
数据交换接口及消息处理示例

A.1 会话连接序列图

图A.1为一个基本的会话连接序列图原型。图中：

a) 在时间“0”时，应用程序连接到平台；

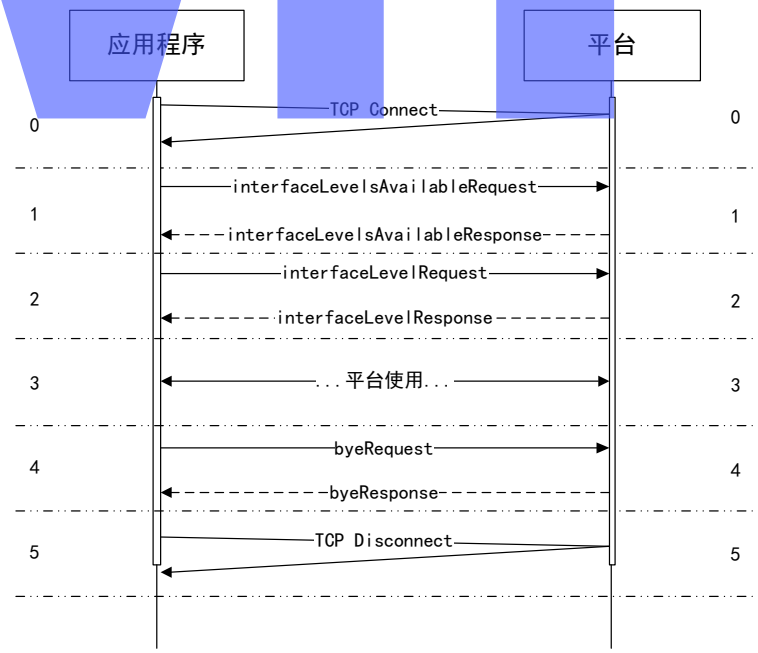
b) 在时间“1”时，应用程序向平台发送<interfaceLevelsAvailableRequest>消息，平台在接收到该消息后返回<interfaceLevelsAvailableResponse>消息。在此过程中，任何流错误都将导致会话中断，任何解析错误都会引起 XML(ExteileMarkuLaguage，扩展性标识语言)解析错误流程处理；

c) 在时间“2”时，应用程序向平台发送<interfaceLevelRequest>消息请求可用接口版本，该消息包含接口版本。该平台通过<interfaceLevelResponse>确认版本的选择；

d) 在时间“3”时，应用程序和平台根据需要进行消息交互。交互的第一条消息应是身份认证。对于平台的连接，应用程序使用< authenticateRequest >进行身份认证，并获取设备令牌。对于设备的连接，应用程序使用包含令牌信息的< deviceAcquireRequest >进行认证，并建立到设备的逻辑连接；

e) 在时间“4”，应用程序通过发送< byeRequest>通知平台即将终止会话。与应用程序相关联的设备令牌立即失效。平台回复<byeResponse >确认。平台会保持会话连接打开，并等待应用程序断开会话连接；

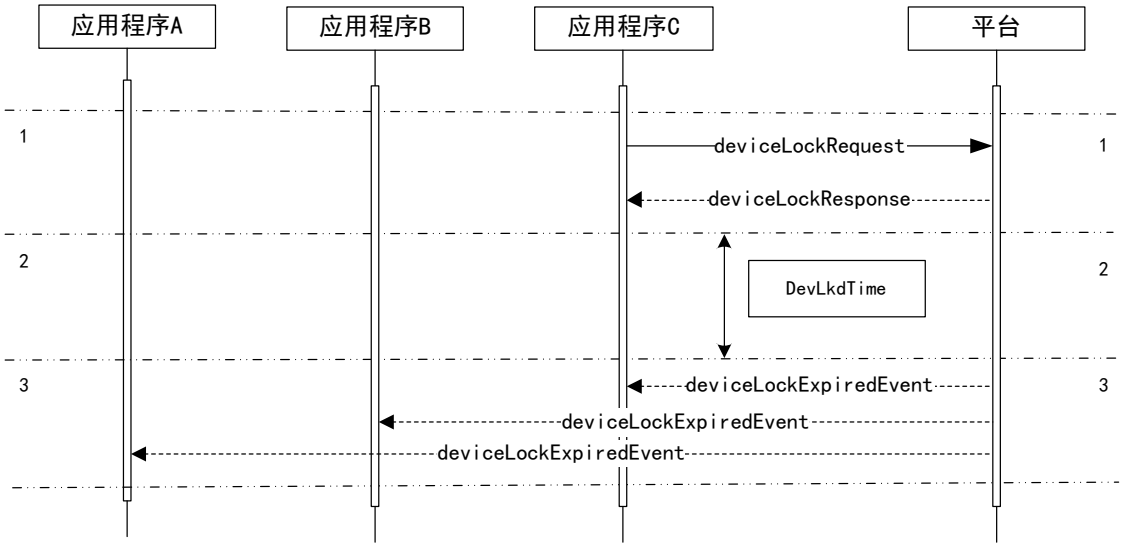
f) 在时间“5”，应用程序关闭 TCP 连接。



图A.1 平台会话序列原型

A.2 锁定超时

图A.2为一个锁定超时的示例。



图A.2 锁定超时

在图A.2中：

——步骤“1”之前，假设三个应用程序A、B和C已成功获取设备连接；

——步骤“1”，应用程序C锁定设备；

——步骤“2”，经过时长DevLkdTime[21)]，应用程序C与该设备的连接没有活动；

——步骤“3”，平台通知所有已获得该设备连接的应用程序，包括应用程序C；

——步骤“3”之后，所有已获取设备连接的应用程序，即应用程序A、B和C，都可尝试锁定设备或经允许后使用该设备。

A.3 平台端向应用程序发送通知

示例1和示例2为平台向应用程序发送通知的例子，其中示例1中通知内容为设备有可读数据，示例2中通知内容为设备的状态已改变。

示例1：

```
<?xml version="1.0" encoding="UTF-8"?>
<cupps messageID="4294901764"
        messageName="notify"
        xmlns="http://www.cupps.aero/cupps/01.03"
        mlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <notify dataAvailable="true" />
</cupps>
```

21) 自表3。

示例2：

```
<?xml version="1.0" encoding="UTF-8"?>
<cupps messageID="4294901760"
        messageName="notify"
        xmlns="http://www.cupps.aero/cupps/01.03"
        mlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <notify>
        <deviceStatusNotification>
            <btStatus   init="false"
                        paperJam="false"
                        paperOut="false"
                        powerOff="true"
                        ready="false"
                        unknown="false" />
        </deviceStatusNotification>
        </notify>
    </cupps>
```

A.4 应用程序向平台请求可用接口版本列表

示例1为应用程序向平台请求可用接口版本列表。示例2为平台响应应用程序请求的回复。

示例1：〈interfaceLevelsAvailableRequest〉

```
<?xml version="1.0" encoding="utf-8"?>
<cupps messageID="1"
        messageName="interfaceLevelsAvailableRequest"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <interfaceLevelsAvailableRequest hsXsdVersion="01.01.0127"/>
    </cupps>
```

示例2：〈interfaceLevelsAvailableResponse〉

```
<?xml version="1.0" encoding="UTF-8"?>
<cupps messageID="1"
        messageName="interfaceLevelsAvailableResponse"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <interfaceLevelsAvailableResponse result="OK">
        <interfaceLevel     level="01.01"
                            wsLocalPath="C:/CUPPS/XSD/01.01/cupps-01.01.0128.xsd"
                            xsdVersion="01.01.0128" />
        <interfaceLevel     level="01.03"
                            wsLocalPath="C:/CUPPS/XSD/01.03/cupps-01.03.0018.xsd"
                            xsdVersion="01.03.0018" />
    </interfaceLevelsAvailableResponse>
```

```
</cupps>
```

A.5 应用程序选择接口版本

示例1为一个应用程序选择接口版本01.03的例子，示例2为平台确认接口版本请求的一个例子。

示例1：

```
<interfaceLevelRequest>
<?xml version="1.0" encoding="utf-8"?>
<cupps messageID="2"
       messageName="interfaceLevelRequest"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <interfaceLevelRequest level="01.03" />
</cupps>
```

示例2：

```
<interfaceLevelResponse>
<?xml version="1.0" encoding="UTF-8"?>
<cupps messageID="2"
       messageName="interfaceLevelResponse"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <interfaceLevelResponse result="OK" />
</cupps>
```

A.6 与平台断开连接

示例1为与平台断开连接的请求消息<byeRequest>，示例2为与平台断开连接的回复消息<byeResponse>消息。

示例1：<byeRequest>

```
<?xml version="1.0" encoding="utf-8"?>
<cupps messageID="4"
       messageName="byeRequest"
       xmlns="http://www.cupps.aero/cupps/01.03"
       mlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <byeRequest />
</cupps>
```

示例2：<byeResponse>

```
 <?xml version="1.0" encoding="UTF-8"?>
<cupps messageID="4"
       messageName="byeResponse"
       xmlns="http://www.cupps.aero/cupps/01.03"
       mlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <byeResponse result="OK" />
```

```
    </cupps>
```

A.7 应用退出策略

示例1为<applicationStopCommandRequest>，示例2为< applicationStopCommandResponse >。

示例1：

```
<?xml version="1.0" encoding="utf-8"?>
<cupps messageID="4294901760"
        messageName="applicationStopCommandRequest"
        xmlns="http://www.cupps.aero/cupps/01.03"
        mlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <applicationStopCommandRequest canDefer="true"
        stopMessage="platform request application to stop!" />
    </cupps>
```

示例2：

```
<?xml version="1.0" encoding="utf-8"?>
<cupps messageID="4294901760"
        messageName="applicationStopCommandResponse"
        xmlns="http://www.cupps.aero/cupps/01.03"
        mlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <applicationStopCommandResponse result="OK" />
</cupps>
```

A.8 应用程序认证

示例1以汉莎航空公司为例，消息中包含多个应用程序的应用程序列表并包含了01.02版本的LHABC以及02.03版本的LHABD应用。示例2为<authenticateResponse>认证回复。

示例1：

```
<?xml version="1.0" encoding="utf-8"?>
<cupps messageID="3"
        messageName="authenticateRequest"
        xmlns="http://www.cupps.aero/cupps/01.03"
        mlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <authenticateRequest airline="LH"
                         eventToken="A3G6CJ8L3J9M1DLH"
                         platformDefinedParameter="ABKD78DF3K:25334;asd=">
        <applicationList>
            <application    applicationName="LHABC"
                            applicationVersion="01.02" />
            <application    applicationData="printer interface"
                            applicationName="LHABD" applicationVersion="02.03" />
        </applicationList>
```

```
    </authenticateRequest>
</cupps>
```

示例2：

```
<?xml version="1.0" encoding="utf-8"?>
<cupps messageID="3"
       messageName="authenticateResponse"
       xmlns="http://www.cupps.aero/cupps/01.03"
       mlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <authenticateResponse deviceToken="H0SMI5GQYNK4RT4K" result="OK">
        <platformParameter  platformVendor="TravelSky"
                            platformVersion="1.0.0.0">
            <cryptAlgorithmInfo>
                <defaultCryptAlgorithm>
                    <cryptAlgorithm name="aes-strong" />
                </defaultCryptAlgorithm>
                <availableCryptAlgorithms>
                    <cryptAlgorithm name="aes-strong" />
                    <cryptAlgorithm name="des-weak" />
                </availableCryptAlgorithms>
            </cryptAlgorithmInfo>
        </platformParameter>
        <applicationParameter>
            <storage driveLetter="C" path="//CUPPSTEMP/LH/"
                    storageT="transient"/>
            <storage driveLetter="C" path="//CUPPSLOCAL/LH/"
                    storageT="persistentLocal" />
            <storage driveLetter="Z" path="//NODE.B.COM/PATH/DATA/"
                    storageT="persistentGlobal" />
        </applicationParameter>
        <workstationParameter defaultSpooledPrinter="//PEKTZLA007/printer1"
            locDesc="Travelsky CUPPS Testbed" name="PEKTZLA011">
            <vendorModelInfo miscInfo="ThinkCentre" model="X" vendor="Lenovo" />
        </workstationParameter>
        <deviceList>
            <device deviceIndex="1" deviceName="PEKTZLA007BC2"
                deviceParameterType="bcDeviceParameter">
                <supportedInterfaceModes>
                    <interfaceMode mode="standard" />
                </supportedInterfaceModes>
                <bcDeviceParameter numberVisualIndicator="0">
                    <vendorModelInfo miscInfo="1900" model="1900"
                        vendor="Honeywell" />
```

```
            <ipAndPort ip="172.27.9.221" port="15001" />
            <bcStatus init="false" powerOff="true" ready="false"
                unknown="false" />
        </bcDeviceParameter>
    </device>
    <device deviceIndex="2" deviceName="PEKTZLA007BG1"
        deviceParameterType="bgDeviceParameter">
        <supportedInterfaceModes>
            <interfaceMode mode="aea" />
            <interfaceMode mode="standard" />
        </supportedInterfaceModes>
        <bgDeviceParameter>
            <vendorModelInfo miscInfo="abc" model="120S"
                vendor="IER" />
            <ipAndPort ip="172.27.9.221" port="15001" />
            <bgStatus init="false" powerOff="true" ready="false"
                unknown="false">
                <bcStatus init="false" powerOff="true" ready="false"
                    unknown="false" />
                <ddStatus init="false" powerOff="true" ready="false"
                    unknown="false" />
            </bgStatus>
        </bgDeviceParameter>
        <device deviceIndex="2.1" deviceName="PEKTZLA007BC1"
            deviceParameterType="bcDeviceParameter">
            <supportedInterfaceModes>
                <interfaceMode mode="standard" />
            </supportedInterfaceModes>
            <bcDeviceParameter numberVisualIndicator="0">
                <vendorModelInfo miscInfo="123" model="120S"
                    vendor="IER" />
                <ipAndPort ip="172.27.9.221" port="15001" />
                <bcStatus init="false" powerOff="true" ready="false"
                    unknown="false" />
            </bcDeviceParameter>
        </device>
        <device deviceIndex="2.2" deviceName="PEKTZLA007DD1"
            deviceParameterType="ddDeviceParameter">
            <supportedInterfaceModes>
                <interfaceMode mode="standard" />
            </supportedInterfaceModes>
            <ddDeviceParameter numCharsPerLine="0" numLines="0"
                pixelsTall="0" pixelsWide="0" supportsAnimation="false">
```

```
            <vendorModelInfo miscInfo="123" model="120S"
                vendor="IER" />
            <ipAndPort ip="172.27.9.221" port="15001" />
            <ddStatus init="false" powerOff="true" ready="false"
                unknown="false" />
        </ddDeviceParameter>
    </device>
</device>
<device deviceIndex="3" deviceName="PEKTZLA007BP1"
    deviceParameterType="bpDeviceParameter">
    <supportedInterfaceModes>
        <interfaceMode mode="aea" />
    </supportedInterfaceModes>
    <bpDeviceParameter supportsColor="none">
        <vendorModelInfo miscInfo="2i" model="PF " vendor="Intermec" />
        <ipAndPort ip="172.27.9.221" port="15001" />
        <bpStatus init="false" paperJam="false" paperOut="false"
            powerOff="true" ready="false" unknown="false" />
    </bpDeviceParameter>
</device>
<device deviceIndex="4" deviceName="PEKTZLA007BT1"
    deviceParameterType="btDeviceParameter">
    <supportedInterfaceModes>
        <interfaceMode mode="aea" />
    </supportedInterfaceModes>
    <btDeviceParameter supportsColor="none">
        <vendorModelInfo miscInfo="2i " model="PF" vendor="Intermec" />
        <ipAndPort ip="172.27.9.221" port="15001" />
        <btStatus init="false" paperJam="false" paperOut="false"
            powerOff="true" ready="false" unknown="false" />
    </btDeviceParameter>
</device>
<device deviceIndex="5" deviceName="PEKTZLA007MS1"
    deviceParameterType="msDeviceParameter">
    <supportedInterfaceModes>
        <interfaceMode mode="standard" />
    </supportedInterfaceModes>
    <msDeviceParameter numberOfTracks="3" style="swipe"
        supportsJIS2="false">
        <vendorModelInfo miscInfo="123" model="111"
            vendor="PrehKeyTec" />
        <ipAndPort ip="172.27.9.221" port="15001" />
        <msStatus init="false" powerOff="true" ready="false"
```

```
                unknown="false" />
        </msDeviceParameter>
    </device>
    <device deviceIndex="6" deviceName="PEKTZLA007OC1"
        deviceParameterType="ocDeviceParameter">
        <supportedInterfaceModes>
            <interfaceMode mode="standard" />
        </supportedInterfaceModes>
        <ocDeviceParameter numCharsPerLine="30" numLines="3"
            style="swipe">
            <vendorModelInfo miscInfo="123" model="111"
                vendor="PrehKeyTec" />
            <ipAndPort ip="172.27.9.221" port="15001" />
            <ocStatus init="false" powerOff="true" ready="false"
                unknown="false" />
        </ocDeviceParameter>
    </device>
    <device deviceIndex="7" deviceName="PEKTZLA007PR1"
        deviceParameterType="prDeviceParameter">
        <supportedInterfaceModes>
            <interfaceMode mode="standard" />
        </supportedInterfaceModes>
        <prDeviceParameter supportsColor="none">
            <vendorModelInfo miscInfo="KH" model="1600" vendor="Epson" />
            <ipAndPort ip="172.27.9.221" port="15001" />
            <prStatus init="false" paperJam="false" paperOut="false"
                powerOff="true" ready="false" unknown="false" />
            <supportedStocks>
                <supportedStock bottomMargin="0" leftMargin="15"
                    perforationLeft="10" rightMargin="15"
                    stockHeight="272"
                    stockName="A4" stockWidth="180" supportsColor="none"
                    topMargin="15" />
                <supportedStock bottomMargin="0" leftMargin="15"
                    perforationLeft="10" rightMargin="15" stockHeight="185"
                    stockName="A5" stockWidth="128" supportsColor="none"
                    topMargin="15" />
            </supportedStocks>
        </prDeviceParameter>
    </device>
    <device deviceIndex="8" deviceName="PEKTZLA007ZI1"
        deviceParameterType="ziDeviceParameter">
        <supportedInterfaceModes>
```

```
                <interfaceMode mode="special" />
            </supportedInterfaceModes>
            <ziDeviceParameter>
                <ipAndPort ip="172.27.9.221" port="15001" />
                <ziStatus init="false" ready="false" unknown="false" />
            </ziDeviceParameter>
        </device>
        <device deviceIndex="9" deviceName="PEKTZLA007ZL1"
            deviceParameterType="zlDeviceParameter">
            <supportedInterfaceModes>
                <interfaceMode mode="special" />
            </supportedInterfaceModes>
            <zlDeviceParameter>
                <ipAndPort ip="172.27.9.221" port="15001" />
                <zlStatus diskError="false" init="false" ready="false"
                    unknown="false" />
            </zlDeviceParameter>
        </device>
    </deviceList>
</authenticateResponse>
</cupps>
```

A.9 设备查询

示例1为<deviceQueryRequest>消息的一个例子,示例2为<deviceQueryResponse>消息的一个例子。

示例1：<deviceQueryRequest>

```
<?xml version="1.0" encoding="utf-8"?>
<cupps messageID="12"
        messageName="deviceQueryRequest"
        xmlns="http://www.cupps.aero/cupps/01.03"
        mlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <deviceQueryRequest deviceName="PEKTZLA007BP1"/>
</cupps>
```

示例2：<deviceQueryResponse>

```
<?xml version="1.0" encoding="utf-8"?>
<cupps messageID="12"
        messageName="deviceQueryResponse"
        xmlns="http://www.cupps.aero/cupps/01.03"
        mlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <deviceQueryResponse result="OK">
        <deviceList>
            <device deviceIndex="3" deviceName="PEKTZLA007BP1"
```

```
            deviceParameterType="bpDeviceParameter">
            <supportedInterfaceModes>
                <interfaceMode mode="aea" />
            </supportedInterfaceModes>
            <bpDeviceParameter supportsColor="none">
                <vendorModelInfo miscInfo="2i" model="PF " vendor="Intermec" />
                <ipAndPort ip="172.27.9.221" port="15001" />
                <bpStatus init="false" paperJam="false" paperOut="false"
                    powerOff="true" ready="false" unknown="false" />
            </bpDeviceParameter>
        </device>
    </deviceList>
</deviceQueryResponse>
  </cupps>
```

A.10 非法消息

示例为一个非法消息的例子，expectedMessageNameList属性包含所有合法的消息名称。

示例：<illogicalMessageErrorEvent>

```
<?xml version="1.0 " encoding="utf-8"?>
<cupps messageID="4294901762" messageName="notify"
 xmlns="http://www.cupps.aero/cupps/01.03"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <notify>
 <illogicalMessageErrorEvent
 applicationName="LHABC"
 applicationVersion="01.02 "
 user="LH"
 workstation="NTERZCK1">
 <illogicalMessage>
 PD94bWwgdmVyc2lvbj0iMS4wIj8+DQo8Y3VwcHMgeG1sbnM6eHNp
 PSJodHRwOi8vd3d3LnczLm9yZy8yMDAxL1hNTFNjaGVtYS1pbnN0
 YW5jZSIgeG1sbnM6eHNkPSJodHRwOi8vd3d3LnczLm9yZy8yMDAx
 L1hNTFNjaGVtYSIgbWVzc2FnZU5hbWU9ImF1dGhlbnRpY2F0ZVJl
 cXVlc3QiIG1lc3NhZ2VJRD0iNCIgeG1sbnM9Imh0dHA6Ly93d3cu
 Y3VwcHMuYWVyby9jdXBwcy8wMS4wMyI+DQogIDxhdXRoZW50aWNh
 dGVSZXF1ZXN0IGFpcmxpbmU9IkxIIiBldmVudFRva2VuPSJBM0c2
 Q0o4TDNKOU0xRExIIiBwbGF0Zm9ybURlZmluZWRQYXJhbWV0ZXI9
 IkFCS0Q3OERGM0s6MjUzMzQ7YXNkPSI+DQogICAgPGFwcGxpY2F0
 aW9uTGlzdD4NCiAgICAgIDxhcHBsaWNhdGlvbiBhcHBsaWNhdGlv
 bk5hbWU9IkxIQUJDIiBhcHBsaWNhdGlvblZlcnNpb249IjAxLjAy
 IiAvPg0KICAgICAgPGFwcGxpY2F0aW9uIGFwcGxpY2F0aW9uTmFt
```

ZT0iTEhBQkQiIGFwcGxpY2F0aW9uVmVyc2lvbj0iMDIuMDMiIGFw
cGxpY2F0aW9uRGF0YT0icHJpbnRlciBpbnRlcmZhY2UiIC8+DQog
ICAgPC9hcHBsaWNhdGlvbkxpc3Q+DQogIDwvYXV0aGVudGljYXRl
UmVxdWVzdD4NCjwvY3VwcHM+</illogicalMessage>
<expectedMessageNameList>
<messageName messageName="deviceQueryRequest"/>
<messageName messageName="notify"/>
<messageName messageName="subscriptionAddRequest"/>
<messageName messageName="subscriptionListRequest"/>
<messageName messageName="subscriptionRemoveRequest"/>
<messageName messageName="byeRequest"/>
</expectedMessageNameList>
</illogicalMessageErrorEvent>
</notify>
</cupps>