

---

Bluetooth 相关协议翻译

## Introducing Bluetooth LE Audio

李坤和

2023 年 3 月

## Introducing Bluetooth LE Audio

- 版本: v1.0.0

- 发布日期: 2022-01-01

- 小组编制人:

**摘要：**最新 Bluetooth 规范及其如何改变我们设计和使用音频和电话产品的方式的指南。

版本历史

---

# 目 录

感谢.....	1
介绍.....	2
第 1 章 背景和传承 .....	4
1.1 THE HEARING AID LEGACY .....	11
1.2 LIMITATIONS AND PROPRIETARY EXTENSIONS .....	12
1.2.1 Apple's Made for iPhone (Mfi) for hearing devices and ASHA .....	12
1.2.2 True Wireless.....	13
1.2.3 Shared listening .....	16
1.3 WHAT'S IN A HEARABLE? .....	16
第 2 章 BLUETOOTH LE AUDIO 架构 .....	21
2.1 THE USE CASES.....	21
2.1.1 The hearing aid use cases .....	21
2.1.2 Core requirements to support the hearing aid use cases .....	25
2.1.3 Doing everything that HFP and A2DP can do .....	27
2.1.4 Evolving beyond HFP and A2DP.....	27
2.2 THE BLUETOOTH LE AUDIO ARCHITECTURE .....	28
2.2.1 Profiles and Services.....	29
2.2.2 The Generic Audio Framework.....	31
2.2.3 Stream configuration and management – BAPS .....	31
2.2.4 Rendering and capture control.....	33
2.2.5 Content control.....	34
2.2.6 Transition and coordination control.....	35
2.2.7 Top level Profiles .....	36
2.2.8 The Low Complexity Communications Codec (LC3) .....	37
2.3 TALKING ABOUT BLUETOOTH LE AUDIO .....	37
2.3.1 缩略词.....	37
2.3.2 首字母 .....	38
2.3.3 不规则发音 .....	38

---

<b>第 3 章 BLUETOOTH LE AUDIO 中的新概念 .....</b>	<b>40</b>
3.1 MULTI-PROFILE BY DESIGN .....	40
3.2 THE AUDIO SINK LED JOURNEY .....	41
3.3 TERMINOLOGY .....	41
3.4 CONTEXT TYPES .....	45
3.5 AVAILABILITY .....	47
3.6 AUDIO LOCATION .....	48
3.7 CHANNEL ALLOCATION (MULTIPLEXING) .....	49
3.8 CALL CONTENT CONTROL ID – CCID .....	51
3.9 COORDINATED SETS .....	52
3.10 PRESENTATION DELAY AND SERIALISATION OF AUDIO DATA .....	53
3.11 ANNOUNCEMENTS .....	57
3.11.1 <i>Broadcast Audio Announcements</i> .....	59
3.11.2 <i>Basic Audio Announcement</i> .....	59
3.12 REMOTE CONTROLS (COMMANDERS) .....	59
<b>第 4 章 ISOCHRONOUS STREAMS .....</b>	<b>61</b>
4.1 BLUETOOTH LE AUDIO TOPOLOGIES .....	61
4.2 ISOCHRONOUS STREAMS AND ROLES .....	63
4.3 CONNECTED ISOCHRONOUS STREAMS .....	65
4.3.1 <i>The CIS structure and timings</i> .....	66
4.3.2 <i>Controlling audio quality and robustness</i> .....	69
4.3.3 <i>Framing</i> .....	74
4.3.4 <i>Multiple CISes</i> .....	74
4.3.5 <i>Bidirectional CISes</i> .....	76
4.3.6 <i>Synchronisation in a CIS</i> .....	78
4.3.7 <i>The CIG state machine</i> .....	79
4.3.8 <i>HCI commands for CISes</i> .....	81
4.4 BROADCAST ISOCHRONOUS STREAMS .....	82
4.4.1 <i>The BIS structures</i> .....	83
4.4.2 <i>Robustness in a BIS</i> .....	86
4.4.3 <i>The Control Subevent</i> .....	91
4.4.4 <i>BIG synchronization</i> .....	91

---

4.4.5	<i>HCI Commands for BISes</i> .....	93
4.4.6	<i>Finding Broadcast Audio Streams</i> .....	93
4.4.7	<i>Synchronising to a Broadcast Audio Stream</i> .....	97
4.4.8	<i>BASE – the Broadcast Audio Source Endpoint structure</i> .....	98
4.4.9	<i>Helping to find broadcasts</i> .....	100
4.4.10	<i>Periodic Advertising Synchronisation Transfer – PAST</i> .....	101
4.4.11	<i>Broadcast_Code</i> .....	101
4.4.12	<i>Broadcast topologies</i> .....	102
4.5	ISOAL–THE ISOCHRONOUS ADAPTATION LAYER.....	104
<b>第 5 章</b>	<b>LC3, LATENCY AND QOS</b> .....	<b>106</b>
5.1	INTRODUCTION .....	106
5.2	CODECS AND LATENCY .....	107
5.3	CLASSIC BLUETOOTH CODECS – THEIR STRENGTHS AND LIMITATIONS .....	108
5.4	THE LC3 CODEC .....	111
5.4.1	<i>The LC3 encoder</i> .....	112
5.4.2	<i>The LC3 decoder</i> .....	113
5.4.3	<i>Choosing LC3 parameters</i> .....	114
5.4.4	<i>Packet Loss Concealment (PLC)</i> .....	115
5.5	LC3 LATENCY.....	116
5.6	QUALITY OF SERVICE (QOS).....	117
5.6.1	<i>Airtime and retransmissions</i> .....	122
5.7	AUDIO QUALITY .....	123
5.8	MULTI-CHANNEL LC3 AUDIO.....	124
5.9	ADDITIONAL CODECS.....	127
<b>第 6 章</b>	<b>CAP AND CSIPS</b> .....	<b>129</b>
6.1	CSIPS–THE COORDINATED SET IDENTIFICATION PROFILE AND SERVICE .....	129
6.2	CAP–THE COMMON AUDIO PROFILE.....	132
6.2.1	<i>CAP procedures for unicast stream management</i> .....	133
6.2.2	<i>CAP procedures for broadcast stream transmission</i> .....	133
6.2.3	<i>CAP procedures for broadcast stream reception</i> .....	133
6.2.4	<i>CAP procedures for stream handover</i> .....	133

---

6.2.5	<i>CAP procedures for capture and rendering control</i> .....	134
6.2.6	<i>Other CAP procedures</i> .....	134
6.2.7	<i>Connection establishment procedures</i> .....	135
6.2.8	<i>Coping with missing set members</i> .....	135
<b>第 7 章</b>	<b>SETTING UP UNICAST AUDIO STREAMS</b> .....	<b>137</b>
7.1	PACS – THE PUBLISHED AUDIO CAPABILITIES SERVICE .....	137
7.1.1	<i>Sink PAC and Source PAC characteristics</i> .....	138
7.1.2	<i>Codec Specific Capabilities</i> .....	138
7.1.3	<i>Minimum PACS capabilities for an Audio Sink</i> .....	140
7.1.4	<i>Audio Locations</i> .....	142
7.1.5	<i>Supported Audio Contexts</i> .....	144
7.1.6	<i>Available Audio Contexts</i> .....	144
7.2	ASCS – THE AUDIO STREAM CONTROL SERVICE.....	147
7.2.1	<i>Audio Stream Endpoints</i> .....	147
7.3	BAP – THE BASIC AUDIO PROFILE .....	151
7.3.1	<i>Moving through the ASE state machine</i> .....	151
7.4	CONFIGURING AN ASE AND A CIG .....	153
7.4.1	<i>The BAP Codec Configuration procedure</i> .....	154
7.4.2	<i>The BAP QoS configuration procedure</i> .....	156
7.4.3	<i>Enabling an ASE and a CIG</i> .....	159
7.4.4	<i>Audio data paths</i> .....	161
7.4.5	<i>Updating unicast metadata</i> .....	163
7.4.6	<i>Ending a unicast stream</i> .....	164
7.4.7	<i>The Source ASE state machine</i> .....	164
7.4.8	<i>Autonomous Operations on an ASE</i> .....	165
7.4.9	<i>ACL link loss</i> .....	166
7.5	HANDLING MISSING ACCEPTORS.....	166
7.6	PRECONFIGURING CISES.....	167
7.7	WHO'S IN CHARGE? .....	167
<b>第 8 章</b>	<b>SETTING UP AND USING BROADCAST AUDIO STREAMS</b> .....	<b>170</b>
8.1	SETTING UP A BROADCAST SOURCE .....	171

---

8.2	STARTING A BROADCAST AUDIO STREAM .....	172
8.2.1	<i>Configuring the BASE</i> .....	173
8.2.2	<i>Creating a BIG</i> .....	175
8.2.3	<i>Updating a broadcast Audio Stream</i> .....	177
8.2.4	<i>Establishing a broadcast Audio Stream</i> .....	177
8.2.5	<i>Stopping a broadcast Audio Stream</i> .....	177
8.2.6	<i>Releasing a broadcast Audio Stream</i> .....	178
8.3	RECEIVING BROADCAST AUDIO STREAMS.....	178
8.3.1	<i>Audio Announcement discovery</i> .....	178
8.3.2	<i>Synchronising to a broadcast Audio Stream</i> .....	179
8.3.3	<i>Stopping synchronization to a broadcast Audio Stream</i> .....	180
8.4	THE BROADCAST RECEPTION USER EXPERIENCE .....	180
8.5	BASS – THE BROADCAST AUDIO SCAN SERVICE .....	180
8.6	COMMANDERS .....	181
8.6.1	<i>Broadcast Assistants</i> .....	183
8.6.2	<i>Scan Delegators</i> .....	183
8.6.3	<i>Adding a Broadcast Source to BASS</i> .....	184
8.6.1	<i>The Broadcast Receive State characteristic</i> .....	187
8.7	BROADCAST_CODES .....	188
8.8	RECEIVING BROADCAST AUDIO STREAMS (WITH A COMMANDER) .....	189
8.8.1	<i>Solicitation Requests</i> .....	189
8.8.2	<i>Remote Broadcast Scanning</i> .....	191
8.8.3	<i>Receiving a Broadcast Stream</i> .....	192
8.8.4	<i>Broadcast_Codes (revisited)</i> .....	192
8.8.5	<i>Ending reception of a broadcast Audio Stream</i> .....	193
8.9	HANOVERS BETWEEN BROADCAST AND UNICAST.....	193
8.10	PRESENTATION DELAY – SETTING VALUES FOR BROADCAST.....	194
<b>第 9 章</b>	<b>TELEPHONY AND MEDIA CONTROL.....</b>	<b>196</b>
9.1	TERMINOLOGY AND GENERIC TBS AND MCS FEATURES .....	197
9.2	CONTROL TOPOLOGIES.....	199
9.3	TBS AND CCP .....	199
9.3.1	<i>The Call State characteristic</i> .....	201

---

9.3.2	<i>The TBS Call Control Point characteristic</i> .....	202
9.3.3	<i>Incoming calls, Inband and Out-of-Band ringtones</i> .....	203
9.3.4	<i>Terminating calls</i> .....	205
9.3.5	<i>Other TBS characteristics</i> .....	205
9.4	MCS AND MCP .....	206
9.4.1	<i>Groups and Tracks</i> .....	207
9.4.2	<i>Object Types and Search</i> .....	208
9.4.3	<i>Playing Tracks</i> .....	208
9.4.4	<i>Modifying playback</i> .....	210
9.4.5	<i>Playing order</i> .....	210
<b>第 10 章</b>	<b>VOLUME, AUDIO INPUT AND MICROPHONE CONTROL</b> .....	<b>212</b>
10.1	VOLUME AND INPUT CONTROL .....	212
10.1.1	<i>Coping with multiple volume controls</i> .....	213
10.2	VOLUME CONTROL SERVICE .....	214
10.2.1	<i>Persisting volume</i> .....	216
10.3	VOLUME OFFSET CONTROL SERVICE .....	216
10.3.1	<i>Audio Location characteristics</i> .....	217
10.4	AUDIO INPUT CONTROL SERVICE .....	217
10.5	PUTTING THE VOLUME CONTROLS TOGETHER.....	220
10.6	MICROPHONE CONTROL .....	221
10.7	A CODICIL ON TERMINOLOGY .....	222
<b>第 11 章</b>	<b>TOP LEVEL BLUETOOTH LE AUDIO PROFILES</b> .....	<b>224</b>
11.1	HAPS THE HEARING ACCESS PROFILE AND SERVICE .....	224
11.2	TMAP – THE TELEPHONY AND MEDIA PROFILE .....	228
11.2.1	<i>Telephony Roles – Call Gateway and Call Terminal</i> .....	228
11.2.2	<i>Media Player Roles – Unicast Media Sender and Unicast Media Receiver</i>	
	229	
11.2.3	<i>Broadcast Media Roles – Broadcast Media Sender and Broadcast Media Receiver</i> .....	229
11.3	PUBLIC BROADCAST PROFILE .....	231
<b>第 12 章</b>	<b>BLUETOOTH LE AUDIO APPLICATIONS</b> .....	<b>232</b>

---

12.1	CHANGING THE WAY, WE ACQUIRE AND CONSUME AUDIO .....	233
12.2	BROADCAST FOR ALL .....	234
12.2.1	<i>Democratizing sound reinforcement</i> .....	234
12.2.2	<i>Audio augmented reality – the “whisper in your ear”</i> .....	237
12.2.3	<i>Bringing silence back to coffee shops</i> .....	237
12.3	TVS AND BROADCAST .....	239
12.3.1	<i>Public TV</i> .....	239
12.3.2	<i>Personal – at home</i> .....	240
12.3.3	<i>Hotels</i> .....	241
12.4	PHONES AND BROADCAST .....	242
12.5	AUDIO SHARING .....	243
12.6	PERSONAL COMMUNICATION .....	244
12.6.1	<i>Tap 2 Hear – making it simple</i> .....	244
12.6.2	<i>Wearables take control</i> .....	244
12.6.3	<i>Battery boxes become more important than phones</i> .....	245
12.7	MARKET DEVELOPMENT AND NOTES FOR DEVELOPERS .....	246
12.7.1	<i>Combining Bluetooth Classic Audio with Bluetooth LE Audio</i> .....	246
12.7.2	<i>Concentrate on understanding broadcast</i> .....	247
12.7.3	<i>Balancing quality and application</i> .....	248
12.7.4	<i>Reinventing audio</i> .....	248
<b>第 13 章</b>	<b>GLOSSARY AND CONCORDANCES</b> .....	<b>250</b>
13.1	ABBREVIATIONS AND INITIALISMS.....	250
13.2	BLUETOOTH LE AUDIO SPECIFICATIONS .....	252
13.2.1	<i>Adopted Specifications</i> .....	252
13.2.2	<i>Draft specifications</i> .....	253
13.3	PROCEDURES IN BLUETOOTH LE AUDIO.....	253
13.4	BLUETOOTH LE AUDIO CHARACTERISTICS .....	255
13.5	BLUETOOTH LE AUDIO TERMS .....	256
<b>第 14 章</b>	<b>INDEX</b> .....	<b>259</b>

---

感谢

---

## 介绍

2013 年春天，我记得在 Trondheim 与助听器行业代表坐在一个会议室里，他们向 Bluetooth 董事会解释为什么他们应该投入时间和精力开发支持音频流的 Bluetooth Low Energy。助听器公司问我是否愿意主持制定新规范的工作组。每个人都认为这是一个好主意，两个团体——Bluetooth Special Interest Group (SIG) 和听力仪器制造商协会(EHIMA)——代表该行业的贸易机构，签署了一份谅解备忘录，开始制定一项新规范，以支持 Bluetooth Low Energy。

当时，我们都认为这将是一个相当快的开发——助听器不需要极高的音频质量——他们在 Bluetooth 技术方面的主要关注点是最大限度地降低功耗。当时我们没有人意识到助听器行业开发的技术和使用案例远远领先于消费音频市场。尽管长期以来建立的电感环路拾音线圈系统(允许广播音频到达多个助听器)仅提供有限质量的音频，但它们支持的连接拓扑比现有的 Bluetooth A2DP 和 HFP 音频 profile 提供的拓扑更复杂。此外，助听器中使用的电源管理技术和优化使电池寿命比类似尺寸的消费产品高出一个数量级。

在接下来的 12 个月里，随着我们开发功能需求文档，越来越多的传统音频和芯片公司开始关注我们的工作，并决定我们为助听器提出的许多功能同样适用于他们的市场。事实上，它们似乎解决了当前 Bluetooth 音频规范中存在的许多限制。因此，需求清单不断增加，小型助听器项目发展成为 Bluetooth SIG 有史以来最大的单一规范开发，最终形成了现在统称为 Bluetooth LE Audio 的产品。

很难相信这段旅程花了八年。最后，我们对核心规范进行了两次重大修订，引入了全新的高效 codec，并发布了 23 个 profile 和服务规范，以及随附文档和分配编号文档，其中包含约 1250 个新页面。

对于任何没有参与这八年旅程的人来说，这是一套非常可怕的文件。本书的目的是尝试将这些规范放在上下文中，添加它们背后的一些历史和基本原理，以帮助读者理解不同部分如何相互连接。我还提供了一些关于市场的背景信息，以及可以听到的内容，以帮助读者将规范与实际产品联系起来。在深入探讨的章节中，我已使用其缩写名称和章节编号引用了规范的特定部分，例如[BAP 3.5.1]用于 Basic Audio Profile 的第 3.5 节。我试着限制参考文献的数量，这样它们就不会妨碍文本。第 13 章中的词汇表和索引也应该

---

帮助开发人员在文档中导航。

我想尽快把这些信息发布出去，所以我第一次求助于自我出版，避免了我过去与出版社之间经历的漫长的延迟。我必须感谢亚马逊，能够如此轻松地做到这一点。如果你觉得这本书有帮助，如果你能写一篇评论并告诉你的朋友，我会非常感激。如果您认为有遗漏或不清楚的地方，[请发送电子邮件至 nick@wifore.com](#)。自我出版的优势在于，我可以比普通书籍更容易地更新书籍。我还将在尝试回答评论，并在该书的网站 [www.bleaudio.com](http://www.bleaudio.com) 上发布更正。

所有参与规范开发的人都认为 Bluetooth LE Audio 为我们提供了开发令人兴奋的新音频产品和应用程序的工具。我希望这本书有助于解释这些新概念，并且激励你去开发新想法。如果是这样的话，我们许多人花在这方面的八年时间将是值得的。

本书的大部分内容将解释这些新规范是如何工作的，它们是如何结合在一起的，以及您可以使用它们做什么，但我们也将在最后一章中对来进行简要介绍。在进入规范之前，了解我们现在的位置以及可听设备的功能，有助于理解所有内容是如何结合在一起的。这就是第一章的目的。如果你想直接进入细节，请跳到第二章。

一些 Bluetooth LE Audio 规范尚未被采用，因此我依赖于 Bluetooth SIG 公开的将要发布的草案。本版本使用的版本在第 13 章中列出。

---

## 第1章 背景和传承

自从 1998 年第一次发布以来，Bluetooth 技术，按理说，已经成长为最成功的双向无线标准。在无线标准业务中，成功与否通常通过每年销售的芯片数量来看的。在此基础上，Bluetooth 就是赢家，2020 年，有 45 亿的芯片被售出。WiFi 紧随其后，42 亿，接着是 GSM 和 3GPP 电话，18.4 亿，DECT 仅仅 1.45 亿。

然而，在其大部分历史上，只有少量的这些芯片得到实际应用。当 Bluetooth 技术第一次发布时，它的开发者明确了四个主要应用场景。其中三个是音频应用，专注于简单的电话功能：

- 简单的无线耳机，它只是你手机的扩展，定义在 Headset profile
- 围绕房子和商业使用的对讲机规范(intercom specification)，以及
- 用于无绳电话的新技术，希望可以取代具有专利的，在美国使用的模拟标准和出现在欧洲的 DECT 标准。其目的是将无绳和蜂窝功能结合在一部手机中。

第四个称作拨号网络，或者叫 DUN，它提供一种将笔记本电脑连接到 GSM 手机的方法，将手机当作 modem，无论你在何处，都可以访问互联网。与通常情况的新技术一样，尽管一些 PC 和手机公司最初有一些热情，但这四个用例都没有真正起飞。无绳电话和对讲机失败了，因为它们可能会从移动电话运营商那里夺走营收。拨号网络是起作用的，但当时手机数据资费是很昂贵的，这会鼓励人们去使用新的 Wi-Fi 标准。耳机开始销售了，但除非你是出租车司机，否则你不太可能去买。很明显，这些特定的用例可能不是那些会在市场上产生规模的用例，因此，Bluetooth SIG 开始在许多其他功能上发力，例如打印和对象传输，这些都没有引起消费者的更多兴趣。

接下来发生的事情是所有标准机构所希望的一政府法规的出现为 Bluetooth 技术提供了更好的存在理由。

在 1990s 年代末，随着手机和手机合同价格的下跌，全球手机使用量激增，手机从商业工具变成了消费者必需品。移动电话运营商开始成为商业街上的名字，并发展成为大型企业。

随着手机使用量的增加，人们对手机使用地点的担忧也随之增加，因为越来越多的交通事故报告称，司机手持手机分心。世界各地的立法者开始提议禁止开车时使用手机。对于手机行业和移动运营商来说，这都是一场潜在

的灾难。据报道，在美国，近三分之一的移动用户费收入(当时基于电话的数量和时长)来自开车时拨打的电话。这是一个金蛋，这个行业输不起。为了保住这笔收入，他们向立法者提出了一个折衷方案，即如果司机不需要拿着电话，可以恢复安全；取而代之的是，拨打电话可以使用 Hands-Free 方案、或者将电话内置在汽车内、或者使用 Bluetooth 无线耳机。

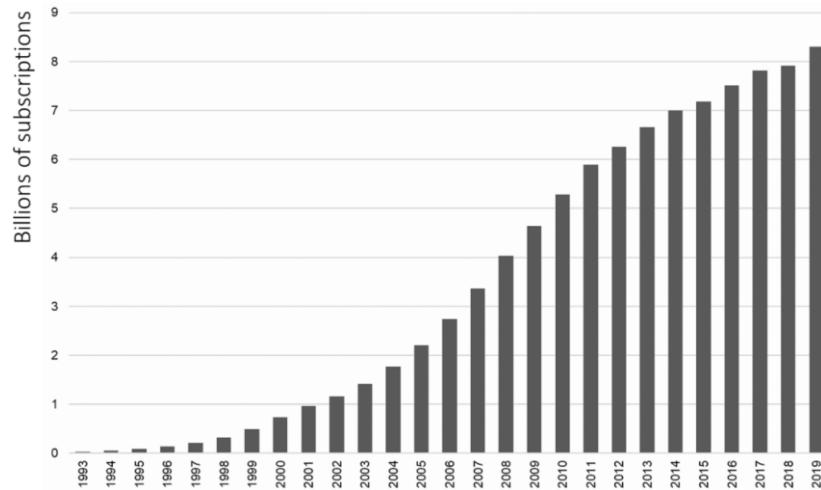


图 1.1 全球移动电话用户的增长

这正是 Bluetooth 技术所需要的激发点。随着新的安全立法生效，手机制造商开始将 Bluetooth 技术应用到越来越多的手机型号中，而不仅仅是高端手机。汽车行业开始将 Bluetooth 技术集成到汽车中，并与 Bluetooth SIG 合作，为该用例开发 Hands-Free profile。这是 Bluetooth 技术的转折点。2003 年，只有大约 10% 的手机销售了包含 Bluetooth 芯片 - 几乎所有的高端手机。第二年就翻倍了。这个数字每年都在增长，如图 1.2 所示。到 2008 年，三分之二的新手机都包含 Bluetooth 芯片。

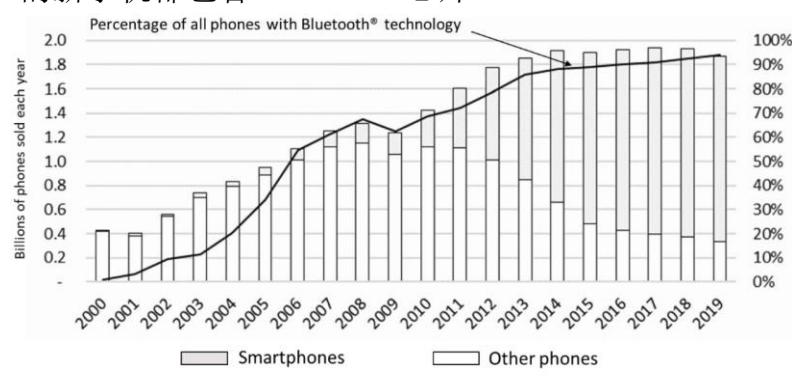


图 1.2 含有 Bluetooth 技术芯片的手机百分比

---

这些芯片中几乎没有真正用于预期目的。显而易见的，因为在同一年只售出了大约 1400 万台耳机，而且这个数字在随后的几年里并没有显着增长。但这是“搭便车”的开始，2005 年有 2.5 亿个 Bluetooth 芯片出货。这些数量带来了竞争和制造效率，推低了价格，并开始了良性循环，增加 Bluetooth 技术的增量成本可以忽略不计，至少在硬件方面是如此。早在 1998 年，也就是 Bluetooth 技术发布的那一年，我们今天所知道的音乐流媒体服务，如 Spotify 和 Apple Music，还有十年的时间。讽刺的是，基于订阅的音乐流并不是一个未知的概念——它已经有一个多世纪的历史了。早在 1881 年，一些电话网络就提供了一项服务，允许你远程收听现场歌剧。但这个概念已经被一个组织得更好的行业模式所取代。物理媒体以蜡盘、唱片的形式出现，你可以随时购买和收听，这扼杀了最初的流媒体概念。在发现他们可以拥有艺术家之后，唱片业在接下来的一百年里尽其所能收紧世界各地的版权法，以保护他们对我们听音乐方式的控制。这将被证明是一个长期的统治。1998 年，我们仍然以实体唱片的形式购买我们的音乐。LP 几乎完全被 CD 取代，但我们当年购买的所有录制音乐中约有 20% 仍然使用盒式磁带。然后情况发生了变化，这在很大程度上要归功于德国弗劳恩霍夫集成电路研究所(IIS)和一家名为 Napster 的小型加州初创公司的单独努力。

Fraunhofer IIS 是一个更广泛的研究组织的一部分，也是开发音频 codec 的卓越中心，该软件压缩音频文件，使其足够小，可以无线传输或存储为数字文件。1993 年，他们为国际 MPEG-1 视频压缩标准开发了一种新的音频 codec。与其他音频编码方案(如用于 CD 的方案)相比，它特别有效，并迅速成为通过互联网传输音频文件的标准。它被称为 MP3。如果没有它，我们将为可以下载的音乐等待更长的时间。

相反，21 世纪初成为了 MP3 播放器和免费可下载音乐的时代。Napster 在 1999 年突然出现，创造了唱片业百年来所面临的第一次真正的颠覆。他们立即将 Napster 告上了侵犯版权的法庭，并试图起诉下载音乐的个人用户。消费者用手指投票，反复点击下载按钮以获得更多音乐。这是第一次，你可以随时听的音乐是免费的。到了 2000 年，也就是它进入市场的一年后，它的大多数用户在个人电脑上下载的歌曲可能比实体专辑更多。然而，唱片公司最终选择了更好的律师，而 Napster 则失败了——最初让音乐免费的努力失败了。尽管杀死 Napster 的战斗一直在法庭上进行，但下一阶段的竞争已经以苹果 iTunes 的形式到来。它不是免费的，但很容易使用。订阅者蜂拥至新产品。六个月后，随着第一台 iPod 的推出，变得更加容易。大多数从

---

事 Bluetooth 技术的工程师可能都是 Napster 和 iTunes 的订阅者，尤其是在长途航班上参加标准会议的时候。在法院对 Napster 的命运做出判决时，已经开始以 Advanced Audio Distribution Profile(更好地称为 A2DP)的形式将音乐流添加到 Bluetooth 中。尽管它的名字很难理解，但它将成为所有 Bluetooth 规范中最成功的，并巩固 Bluetooth 技术作为无线音频标准的地位。

A2DP 及其支持规范于 2006 年被采用。像诺基亚这样的公司，一直积极参与其发展，预计它会立即取得成功，但事实证明它在起飞方面很慢。消费者没有看到购买昂贵的无线耳机的优势，大多数人使用每部手机附带的免费有线耳塞，尽管它们的音频质量通常有限。令业界感到惊讶的是，最初的增长不是来自耳机，而是来自 Bluetooth 扬声器。最初的移动音乐市场是一个你随身携带音乐，但不一定在移动时播放的市场。随着电视开始包含 A2DP，扬声器逐渐成为条形音箱，但耳机市场仍然非常难以增长，直到出现新的服务 Spotify。

Spotify (及其美国前身潘多拉)推出了一种新的商业模式。你可以再次免费听音乐，只要你接受广告。如果您不希望中断，那很好 - 您可以通过支付订阅费用来摆脱它们。它通过产生收入来支付许可费，巧妙地解决了版权问题，无论用户是采用订阅还是广告支持的路线。它有效地复制了 Napster 的所作所为，但找到了合法的方法。它帮助 Spotify 的发布与第一款 iPhone 相吻合，消费者开始意识到智能手机不会主要用于电话。相反，它们的大部分时间都在做其他事情，尤其是在它们的出现与提供无限数据使用的廉价移动数据计划同时出现的时候。移动运营商很快将 Spotify 与他们的手机合同捆绑在一起，用户也签约了。最重要的是，它易于使用。iTunes 仍然是一项服务，用户购买下载后必须做出播放决定。使用 Spotify，你按下一个按钮，音乐就会出现。它可以是你自己的播放列表，也可以是音乐博主、最喜欢的 DJ 或算法的播放列表。它最大的吸引力在于它是平滑的一你按下一个按钮，音乐就会从你的耳塞中传出，直到你停止。您不再需要持有手机；你可以把它放在口袋或包里，这使它非常适合在移动中收听。当你不需要拿着手机的时候，耳机线就成了麻烦。这是用户说服他们自己开始购买 Bluetooth 耳机所需的推动力。品牌 Bluetooth 耳机销量开始上升，因为制造商看到客户不再使用有线耳机。到 2016 年，Bluetooth 耳机的销量超过了有线耳机。

图 1.4 显示了 Spotify 体验在多大程度上推动了这一变化。自 2006 年 Spotify 和 A2DP 规范独立上市以来的十年中，Bluetooth 耳机的增长几乎与 Spotify 用户的增长完全一致。

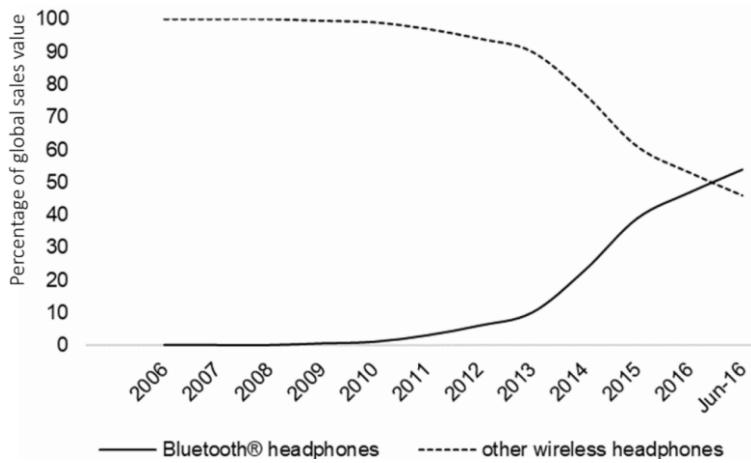


图 1.3 有线耳机到 Bluetooth 耳机的转变

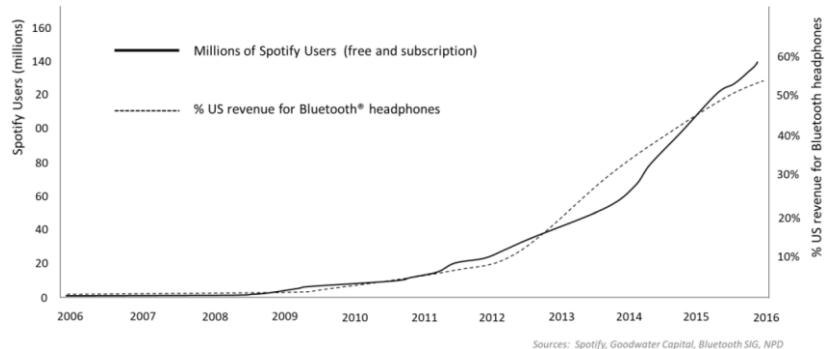


图 1.4 Spotify 订阅和 Bluetooth 耳机的增长

无线耳机芯片销售的增长推动了创新。到 2010 年，Bluetooth 芯片公司每年出货超过 15 亿块芯片，并正在寻找更多的方式来使其产品差异化。新的 Bluetooth Low Energy 标准刚刚推出，但直到 2011 年底才出现在 iPhone 4s 上，而且几年后新一代腕带才开始使用。在此期间，Cambridge Silicon Radio 公司 (CSR，后来被 Qualcomm 公司收购) 已成为 Bluetooth 音频设备芯片的领先者，并正在研究如何扩展 Bluetooth audio profiles 的功能。

他们为什么要这么做？他们看到的问题是，两个主要的 Bluetooth 音频标准都是为非常特定的用例编写的，这是没有预见到未来的。结果是，它们的行为非常不同。HFP 专注于低延迟、双向、单声道语音传输，而 A2DP 支持将高质量的音乐流传输到单个设备，无返回音频路径。这两个 profiles 都不容易扩展，这限制了您可以使用它们做什么。CSR 试图突破无线音频的界限。他们已经成功地开发了一种比 Bluetooth 规范要求的 SBC codec 更高效的 codec，并在芯片中添加了自己的增强型 AptX codec。现在，他们决定

---

更进一步，看看能否找到一种方法，将 A2DP 规范用于将立体声传输到两个独立的耳塞。他们的努力取得了成功，开启了 Bluetooth 音频的新时代，这将带来巨大的增长。消费者对无线耳机的接受意味着 Bluetooth 技术与 Spotify 的增长脱钩，并获得了新的发展势头。CSR 的创新是为单个耳塞开发一种接收 A2DP 流的方法，并将立体声流的一个通道与时序信息一起转发给第二个耳塞。使用时序信息，第一耳塞可以延迟呈现其音频通道，直到它知道第二耳塞已经接收到其音频数据并准备好呈现其流。就用户而言，似乎每个耳塞都在接收自己的左声道或右声道。正如我们稍后将看到的那样，让它工作起来并不是那么简单，但它将成为游戏规则的改变者。

先行者不是大公司。2014 年初，欧洲的两家公司—瑞典的 Earin 和德国的 Bragi，启动了立体声无线耳机的众筹。Earin 的产品是一对无线耳塞，使用新的芯片组来推动它之前可以做的事情的界限。他们仍在制作标志性设计，在 2021 年初推出第三代产品。然而，Bragi's Dash 真正吸引了人们的想象力。2014 年春天，他们打破了 Kickstarter 迄今为止资金最高的活动记录，为他们的 Dash 无线耳机筹集了超过 330 万美元。这是一项惊人的工程，它承诺将你能想到的几乎所有功能都塞进一对耳塞中。它提高了你可以用耳朵里的东西做什么的标准，打开了一个全新的市场领域，我为它创造了“可听”这个名字。这是一个雄心勃勃的概念，值得赞扬的是，他们成功地推出了 Dash。尽管有热情的追随者，但它表明仅仅集成技术是不够的—你需要找到它的用途。尽管为软件开发人员提供了 SDK，但 Dash 未能在消费者中获得足够的吸引力。在接下来的两年里，其他众筹项目的目标更加复杂，并成功吸引了约 5000 万美元的资金。许多人未能实现这一目标，他们开始意识到将这么多技术塞进一个小型耳塞是多么困难。硬件是很难的。这些初创公司大多半途而废，甚至 Bragi 也被迫做出艰难的决定，放弃硬件，销售其产品系列，专注于其他主流音频产品的嵌入式操作系统。

渐渐地，更大的公司开始涉足可听的领域，利用他们更深层的资源来制造这些困难的产品。然后，2016 年 9 月，苹果推出了 AirPods。尽管最初受到许多记者的嘲笑，但消费者还是喜欢它们。在短短的两年内，它们成为有史以来销售最快的消费品，自推出以来已售出超过 2.5 亿双。其他品牌迅速跟进，中国的芯片供应商也紧随其后。早在 2014 年，当 Bragi 和 Earin 开始规划市场的未来时，只有四五家芯片厂商生产 Bluetooth 音频芯片组。今天有三十多家。尽管存在供应链问题，但据估计，2020 年约有 5 亿只耳塞出货，预计到 2022 年底，这一数字将翻一番。这些数字是基于经典 HFP 和

---

A2DP profile 的耳塞。随着新的 Bluetooth LE 音频产品开始出货，其附加功能、广播功能和增强的电池寿命，数量可能会超过预期。

然而，无线音频不仅仅是耳机。Bluetooth 音频的大部分增长都是由音乐流媒体推动的，而音乐流媒体又是由 Spotify, Apple Music 和亚马逊 Prime 音乐等服务的内容的现成可用性推动的。流媒体视频的到来同样受欢迎，无线耳塞成为收听的首选设备。消费者喜欢易用性，在大多数情况下，这转化为消费内容，而不是自己生成内容，尽管 TikTok 等应用程序表明，如果内容生成变得简单有趣，用户将自己制作并分享。在这种演变中，语音，主要是语音呼叫的保留，看起来好像它正在成为一种糟糕的关系。当亚马逊推出 Alexa 时，情况发生了变化。尽管有所保留，消费者开始与互联网交谈。从那时起，语音识别出现了复兴，现在大多数家用电器都想与我们通话，让我们与他们通话。这些应用程序可能会随着 Bluetooth LE 音频支持的新功能的引入而增长，本书将介绍这些功能。通过广播拓扑和优先选择哪些设备可以与您通话的能力，可以为语音到机器通信添加额外功能和新机会。它可能是智能家居行业的救星。

围绕耳塞的引入，特别是苹果的 Airpods，其发展速度惊人。我们看到许多新公司正在开发 Bluetooth 音频芯片、微型音频传感器和 MEMS 1 麦克风的进步，同时提供先进音频算法的公司数量大幅增长，以实现有源噪声消除、回声消除、空间声音和频率平衡等功能。

尽管 Hands-Free Profile 和 A2DP 继续为我们提供良好的服务，但它们都是为简单的端对端拓扑设计的。被称为 Bluetooth 经典音频，他们的设计是假设两个设备之间的单个连接，其中每个单独的音频数据包都被确认。如果有两个单独的设备想要接收音频流，则这不起作用。因此，今天的立体声耳塞依赖于专有解决方案，这些解决方案通常涉及添加第二个无线电来在左右耳塞之间进行通信，以确保它们都在正确的时间播放音频。另一个限制是 Hands-Free Profile 不是为我们今天使用的广泛的蜂窝和 IP 语音电话应用而设计的。只有这两个独立的专用音频 profiles 会导致当用户想要从一个应用程序切换到另一个应用程序时出现 multi-profile 问题。这是在您添加并发语音控制和共享音频的新需求之前。

在我们的日常生活中，我们都更频繁地使用无线音频，无论是相互交谈，还是与外界隔绝。为了支持这种行为变化，我们需要少考虑通常一起使用的单个设备之间的连接，如耳机和电话。相反，我们需要更多地了解更广泛的音频生态系统，在那里，我们有不同的设备，我们可以在一天中戴在耳朵上，

---

不断地收听和改变它们对其他设备的使用。为了实现无缝工作，控制需要变得更加灵活。这就是导致 Bluetooth LE Audio 发展的背景。

Bluetooth SIG 意识到，他们的音频规范需要不断发展和适应，既要满足当前的需求，也要考虑到我们在音频方面所做的越来越多样化的事情，以及我们在未来 20 年可能出现的情况。在许多用例中出现了一些非常相似的需求。设计师想要更低的功率。不仅是为了延长电池寿命，而且为了能够支持更多的降噪处理和正在出现的其他有趣的音频算法，例如检测迎面而来的交通或相关对话。对于其他应用程序，他们希望减少延迟，特别是在游戏或收听现场对话或广播时。还有对更高音质的不懈追求。制定规范的 Bluetooth LE Audio 工作组的任务是提出支持这些要求的新标准，以及设想的所有拓扑结构，而不必依赖 non-interoperable 扩展。其目的是让该行业从今天所处的位置(依赖于专有实现)发展到可以混合和匹配来自不同制造商的设备的位置。

## 1.1 The hearing aid legacy

听到许多创新是由助听器行业发起的，这让许多人感到惊讶。多年来，助听器一直需要解决音频质量、延迟、电池寿命和广播传输等问题。它们平均每天佩戴 9 小时，因此电池寿命至关重要。在此期间，助听器不断放大和处理环境声音，以便佩戴者能够听到周围发生的事情和正在被说的话。它们通常包括多个麦克风，以允许音频处理算法识别本地音频环境并对其做出反应，从而滤除干扰声音。在公共场合，如果设施可用，它们可以连接到称为 telecoil 的系统，基本上是感应回路，用于剧院、公共交通和其他公共区域，以收听音频和提供信息。这些是广播系统，可以在 telecoil 的传输区域内处理数百人，或者允许使用非常小的环路进行私人对话。

助听器用户一直希望能够连接到手机和其他 Bluetooth 设备，但传统 HFP 和 A2DP 解决方案的功耗是一个挑战。2013 年，苹果推出了基于 Bluetooth Low Energy 规范的专有解决方案，增加了一个音频流，可以连接到助听器中的特殊 Bluetooth LE 芯片。它被授权给助听器制造商，并受到消费者的欢迎，但它只适用于 iPhone，而且是单向的。

尽管对这一发展表示欢迎，但助听器行业担心，针对苹果的解决方案并不具有包容性。他们想要一个适用于任何电话或电视的全球标准，也可以取代 20 世纪 50 年代过时的 telecoil 规范。2013 年，所有主要助听器公司的代表与 Bluetooth SIG 的董事会坐下来，达成了一项联合协议，提供资源帮助

---

开发新的低功耗 Bluetooth 音频标准，为助听器生态系统带来互操作性。开发工作开始后不久，许多消费音频公司开始研究助听器的使用案例，并意识到它们同样适用于消费市场。虽然助听器的音频质量要求不太严格(因为其用户有听力损失)，但结合环境音频、Bluetooth 音频和广播基础设施的使用案例远比 HFP 和 A2DP 目前涵盖的情况先进。他们有可能解决当前音频规范的许多已知问题。

随着越来越多的公司参与进来，项目扩大了。经过八年的努力，Bluetooth LE 音频项目已经发展成为 Bluetooth SIG 有史以来最大的规范开发项目。由此产生的规范涵盖了 Bluetooth 标准的每一层，包括 1250 多页新文档和更新文档中的文本，其中大部分已被采纳或正在被采纳。

## 1.2 Limitations and proprietary extensions

### 1.2.1 Apple's Made for iPhone (Mfi) for hearing devices and ASHA

2014 年，苹果推出了自己的助听器专用 Bluetooth Low Energy 解决方案，并授权给助听器制造商。它与一家芯片合作伙伴合作开发，为 Bluetooth LE 协议添加了扩展，允许在手机和一个或两个助听器之间单向传输数据。手机上的应用程序允许用户选择要连接的助听器，以及允许他们设置音量(独立或成对)，并选择各种预设，这些预设应用助听器上的预配置设置以应对不同的声学环境。Mfi 听力设备解决方案适用于 iPhone 5 手机、iPad(第四代)设备和后续产品。

Mfi 支持的流行功能之一是“实时收听”，它允许 iPhone 或 iPad 用作远程麦克风。对于佩戴助听器的人来说，这可以让他们将手机放在桌子上，以便接听和传输对话。远程麦克风是助听器的有用附件，而现场收听功能提供了这一功能，无需购买其他设备。早在 2021，苹果就宣布，他们的听力设备 Mfi 将升级为双向音频，实现免提功能。

苹果的动机并不完全是利他主义的。许多国家的无障碍法规迫使他们在手机中安装一个 telecoil，这增加了成本并限制了物理设计。他们希望监管机构接受 Bluetooth 解决方案作为替代方案。诺基亚也有类似的愿望，并在退出手机生产之前积极支持 Bluetooth LE Audio 规范的开发。

苹果的 Mfi 听力解决方案仅适用于 iPhone 和 iPad，而 Android 用户则没有解决方案。在 Bluetooth LE Audio 开发的同时，Google 和助听器制造商 GN Respond 合作开发了一个名为 ASHA (助听器音频流) 的开放式 Bluetooth

---

LE 规范，这是一个适用于 Android 10 及以上版本的软件解决方案。它为 Bluetooth LE 提供了不同的专有扩展，支持从任何兼容的 Android 设备到 ASHA 助听器的单向流。

在 Bluetooth LE 音频开始出现在手机上之前，ASHA 为 Android 用户提供了一个受欢迎的填补。目前支持 Mfi 助听器或 ASHA 的助听器制造商可能会继续这样做。他们将通过添加 Bluetooth LE Audio 扩展其支持，为消费者提供最广泛的选择。最终，这只是另一个协议，意味着更多的固件。然而，大多数新开发可能会朝着全球可互操作的 Bluetooth LE 音频标准发展。

### 1.2.2 True Wireless

2013 年左右，Cambridge Silicon Radio 公司开发了一种向两个耳塞发送立体声流的实用解决方案。在被 Qualcomm 公司收购后，它被重新命名为 TrueWireless，这是世界上大多数人现在用于立体声耳机的术语。当竞争对手看到苹果 Airpods 的成功时，Qualcomm 公司为所有其他想要开发竞争耳塞的公司提供了一个可行的替代方案。真正的 True Wireless Stereo(TWS)这个名称很快被确立并应用于几乎每一种新产品，无论其芯片是谁的。

使用单独耳塞的 A2DP 的主要障碍在于，它是为单点对点通信设计的。早在 2008 年，Bluetooth SIG 就在一份名为“多头耳机使用白皮书”的白皮书中为如何将流发送到多个 Bluetooth LE 音频接收器提供了指导，但它避免了同步问题，假设每个音频接收器可以决定何时渲染流。对于耳塞来说，一旦左右气流不同步，你就会有一种非常不愉快的声音在你的大脑中移动的感觉。白皮书方法还依赖于对音频源 A2DP 规范的修改。这是一个问题，因为这意味着没有采用它们的耳机或扬声器将不兼容。要在市场上取得成功，需要一种能够与任何音频源配合使用的解决方案，这实际上意味着这些解决方案需要，仅仅在音频接收器(耳塞)中实现。

CSR 提出的解决方案称为重播或转发方法，如图 1.5 所示。

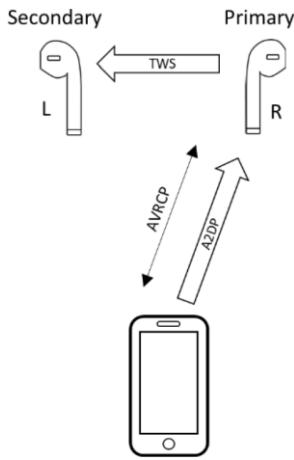


图 1.5 TWS 中继策略

在连接到手机之前，两个耳塞相互配对(可以在制造时完成)。一个设置为主设备，另一个设置为辅助设备。要接收 A2DP 流，主设备将与音频源配对，显示为接收立体声流的单个设备。当音频数据包到达时，它会解码左右声道，在上面的示例中，将左声道直接中继到辅助耳塞。使用 Bluetooth 技术可能是可能的，但如果耳塞有小天线，这可能会有问题，因为耳机可以很好地吸收 2.4GHz 信号。许多公司在首次在户外测试设备时发现了这一点。CSR 提出的解决方案称为重播或转发方法，如图 1.5 所示。在室内，墙壁和天花板的反射通常会确保 Bluetooth 信号通过。在外部，由于没有反射表面，它们可能无法提供帮助。为了补偿这一点，通常会添加第二个无线电，以解决吸收问题。最受欢迎的选择是 Near Field Magnetic Induction(NFMI)，这是一种用于短距离音频传输的高效低功耗解决方案。其他芯片厂商也使用了类似的 Low Band Retransmission (LBRT) 方案。

由于主要设备负责音频中继的定时，因此它确切地知道辅助耳塞将在何时呈现音频中继。它需要延迟其音频流的渲染以匹配。这是中继方法的问题之一，因为中继过程和缓冲增加了音频连接的延迟。对于大多数应用程序，用户不太可能注意到这一点，尤其是因为 A2DP 延迟通常占主导地位。

中继方法也适用于 HFP，尽管在大多数情况下，只有主要设备用于语音返回路径。音量和内容控制，如接听电话或暂停音乐，仍然通过与主耳机的连接使用 Audio/Video Remote Control Profile (AVRCP)。第二无线电设备(如果存在)还可用在主设备和辅助设备之间中继用户接口和 AVRCP 控制，例如暂停、播放和音量。

最近，公司开始转向 sniffing 方法，如图 1.6 所示

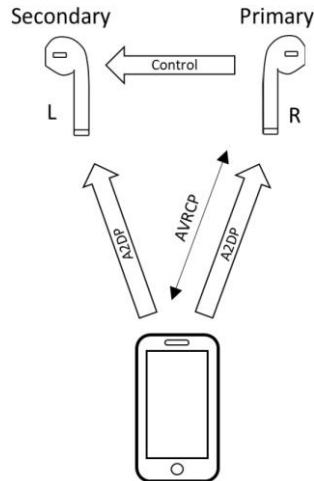


图 1.6 TWS sniffing 方法

在这里，中继方法增加的延迟通过两个耳机监听主 A2DP 流来消除。只有一个耳塞(主要设备)向手机确认收到音频数据。通常，辅助设备不知道在哪里找到音频流，也无法解码音频数据。在这种情况下，主设备通过 Bluetooth 链路或专用 sub-GHz 链路向其提供该信息。该链路由制造商配置，因此其他设备不可能获取 A2DP 流。

由于两个耳塞都直接接收相同的 A2DP 流，因此这可能是一种更为稳健的方案，尤其是作为仅 Bluetooth 解决方案实现时。在没有音频流中继的情况下，延迟明显更好。

这两种方案都需要在耳塞中相当低的 Bluetooth 堆栈水平上进行巧妙的扩展，因此在很大程度上一直是芯片供应商的领域。苹果的成功刺激了竞争对手的芯片供应商进行创新，结果是，现在有十几种不同的真正的无线方案，建立在这两种技术的变体上，其中一些包括主交换等附加功能，因此一个如果失去链接，两个耳机可以改变角色。

这些专有方法的问题在于，它们可以分割市场。苹果、高通和其他公司都为他们的解决方案申请了专利，这导致他们的竞争对手花费时间和精力研究如何绕过这些专利，而不是创新来推动市场向前发展。这也意味着几乎不可能混用两个不同制造商的耳机，因为他们可能使用不同的 TWS 方案。对于消费者 TWS 耳塞来说，这可能不是一个大问题，因为它们总是成对购买，但对于助听器来说，左右耳可能需要不同类型的耳塞。这使得很难将该方案扩展到扬声器，因为环绕声系统通常结合不同制造商的扬声器。尽管专有解决方案可以在其早期阶段刺激市场增长，但它们很少有助于其长期发展。这就是 Bluetooth LE Audio 的作用所在。

### 1.2.3 Shared listening

共享音频不仅仅是向一对耳塞和助听器发送信号，还涉及到扩大能够收听信号的人数。虽然公共设施可以满足大量同时收听的人，但有一个更个性化应用程序，您可以与朋友分享您的音乐。关于多耳机的白皮书(如上所述)阐述了如何通过 A2DP 与其他共享听力，但它所建议的解决方案似乎从未被应用到产品中。相反，这部分市场主要由总部位于巴黎的 Bluetooth 软件公司 Tempow 拥有。他们开发了一种基于电话的音频操作系统，称之为 Dual-A2DP，它可以生成单独的左右同步流，允许两对单独的耳塞同时呈现流。虽然这是有用的，但它是有限的，只有少数制造商采用。Bluetooth LE Audio 超越了这一点，提供了一个可扩展的解决方案，可以从一个听众过渡到多个听众。

## 1.3 What's in a hearable?

无线耳塞相对最近的到来，受到芯片可用性的限制，该芯片可以提供一种支持分离的左右立体声流的方法。但这不是唯一的原因。正如所有进入这一市场的创业公司所发现的，将你需要的所有功能打包到像耳塞一样小的东西中，以再现好的音频是很困难的。事实上，这很难。将 Bluetooth 技术与任何额外的传感器结合起来支持它变得非常困难。只有大约 25% 众筹 hearable 公司曾成功推出产品。即使是像苹果这样的公司，也不得不投入近五年的开发来实现 AirPods。他们甚至设计了自己的 Bluetooth 芯片，以获得使 AirPods 如此成功的性能。这是只有最大的耳机公司——苹果和华为——才有资源做的事情。

这是一个助听器制造商熟知的故事，因为他们多年来一直在完善助听器的小型化。他们还使用定制芯片优化性能，从而使设备在小型锌空气电池上运行数天。Bluetooth 助听器中的元件数量惊人，如图 1.7 所示，它代表了相当基本的设计。它的结构与耳塞非常相似。

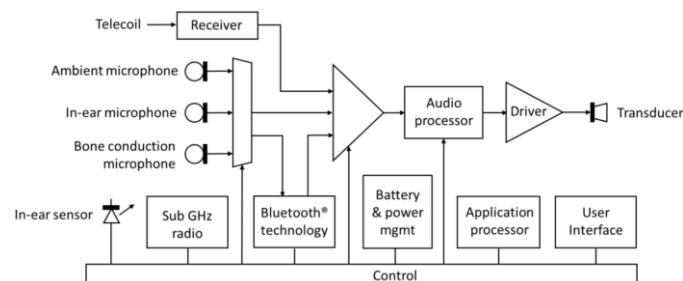


图 1.7 simple Bluetooth 助听体系结构

---

当你将助听器或耳塞拆成碎片时，大多数人遇到的第一个惊喜是其中的麦克风数量。要执行主动噪声消除，您需要一个麦克风监控环境声音，另一个麦克风位于耳道中。如果您想拾取用户的语音，通过 Bluetooth 链接发送到他们的手机，通常会有一个骨传导麦克风，帮助拾取并将语音与周围环境隔离。这是最低限度。然而，通常包括额外的麦克风以生成波束形成阵列以改善方向性。可以包括其他音频算法以检测环境的类型并进一步增强用户的语音。后者很重要，因为耳塞和助听器中的麦克风并不像设计师所希望的那样靠近嘴巴——它们可能经常在耳朵后面。

我们即将看到新的法规生效，以测量耳道中的声级，并警告用户潜在的听力损害，这可能会导致更多的内部麦克风。好消息是，在过去几年中，MEMS 麦克风有了很大的发展，部分原因是语音助理的需求。这些创新增强了方向性和光束控制，因此它们可以跟随您在房间里四处走动。与传统麦克风结构相比，MEMS 的优势在于，您可以将数字信号处理器集成到麦克风本身中，从而减小尺寸和功耗。在最新的可听设备中找到四个或更多的话筒并不罕见。这些输入需要混合并分配到助听器的不同功能。

如果设备同时包含 Bluetooth 技术和 telecoil 接收器，则需要从它们发送适当的音频。对于向第二个耳机发送控制信号或音频流的耳机，这些都需要通过亚 sub-GHz 无线电(通常为 NFMI)进行路由，同时缓冲主信号以同步两个耳机的渲染时间。

在输出端，音频传感器变得更小，传统的开放式线圈结构受到微型平衡电枢传感器和音频阀的挑战。市场也看到了 MEMS 扬声器的出现，提供高声级。虽然在现阶段它们可能更适合耳机，但这项技术可能会转移到耳塞。

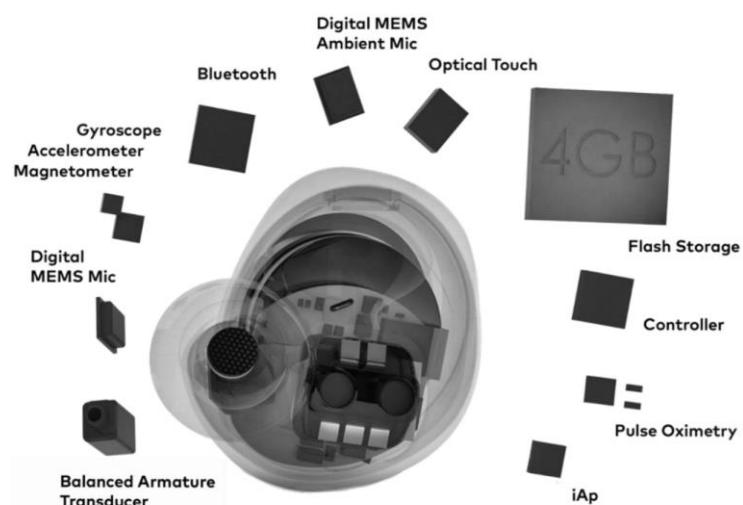
看看 Bluetooth LE 音频可能带来的新用例，这种处理可能会变得非常复杂。接收 Bluetooth 媒体流，并允许同时发送语音命令的噪声消除耳塞在发送之前需要将语音分量与环境声音分离(并应用回声消除)，同时抑制与传入 Bluetooth 信号混合的环境声音。如果传入的 Bluetooth 媒体流是从与环境相同的源广播的，例如当您在剧院、会议室或观看电视时，则需要在保持大约 30 毫秒的总延迟的同时完成所有这些。这是一个挑战。

管理所有这些，需要一个应用处理器，它控制专门的音频处理块。为了尽量减少助听器的功率和延迟，通常在硬件中实现，而不是在通用数字信号处理器(DSP)中实现。应用处理器通常还将控制用户界面，其中命令可以来自助听器上的按钮或电容传感器，通过 Bluetooth 接口，或来自遥控器，遥控器可以使用 Bluetooth 技术或专有的 sub-GHz 无线电链路。大多数设备包

---

括一个光学传感器，用于检测何时将其从耳朵中取出，以便将其置于睡眠状态。最后，还有电池和电源管理功能。许多助听器仍然使用锌-空气电池，其提供比可充电电池更好的功率密度。这提供了两个主要优点——电池寿命更长，重量更小。如果你整天都戴着助听器，重量是很重要的，这就是为什么大多数现代助听器重量不到 2 克——大约是一个 AirPod 重量的一半。

那只是电子设备。将所有这些都安装到耳塞或助听器中是进一步的挑战，挑战了柔性电路和多层封装的极限。设计师需要让它舒适，确保它不会从耳朵里掉出来，同时还要完成所有这些，同时保持良好的听觉路径，这样，无论是合适的，还是内置的电子设备，都不会影响声音的质量。您还需要考虑助听器是否堵塞的问题，即它是否阻塞您的耳朵以停止环境声音，还是打开以允许您听到这两种声音。直到最近，人们还认为，如果想要有效消除环境噪声，遮挡是必要的，但最近的一些创新表明，情况可能不再如此。完全堵塞耳道会导致湿度增加，特别是长时间佩戴时，因此我们可能会看到更多设计采用开放式设计。所有这些都不容易，这也是助听器仍然昂贵的原因之一。然而，越来越多的芯片公司正在提供已经做了大量工作的参考设计。与提供设计专业知识以缩短上市时间的专业顾问合作伙伴计划一起，这导致消费者耳塞市场以惊人的速度增长。但我们仍然处于你可以把什么放进耳朵的可能性的开始。到目前为止，最有野心的是 Bragi's Dash。除了提供真正的无线立体声外，他们还决定添加内置 MP3 播放器和闪存，让您在外出跑步或在健身房时将手机留在家中，但仍然可以直接从耳机收听存储的音乐。



*Image courtesy Bragi GmbH*

图 1.8 来自 Bragi 的 Dash 耳塞——第一款真正可听的耳塞

---

Bragi 认识到耳朵是身体上最适合大多数生理传感器的地方，因此为 Dash 配备了大量传感器和功能：总共九个自由度的运动传感，包括加速度计、磁力计和陀螺仪、温度计、心率监测器和脉搏血氧计。他们制作了一个非常漂亮的图形，显示了所有元素及其相对大小，如图 1.8 所示。这是一个惊人的成就，但可悲的是，这超出了当时市场的期望。正如健身腕带制造商所发现的那样，让顾客关注他们的健康数据令人惊讶地困难。与音乐不同的是，健康数据只会吞噬他人的内容，而健康数据需要大量的分析才能将其转化为一个吸引用户的故。

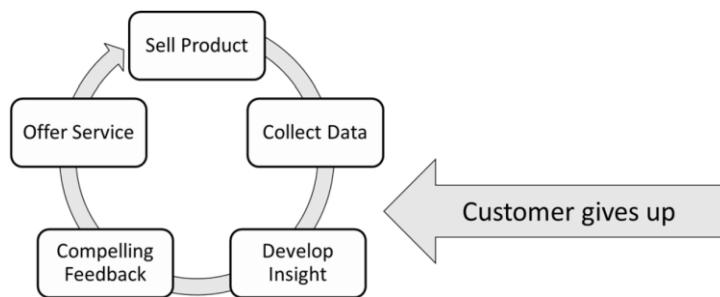


图 1.9 健康和健身数据的第 22 条

如图 1.9 所示，这是一个第 22 条，在你有足够的数据转化为任何有价值的东西之前，你需要捕获大量的数据。在此期间，您需要雇佣一些非常昂贵的数据科学家来尝试和开发一些令人信服的反馈，支付云存储和分析的持续成本，以及每一个新版本的手机操作系统的应用程序更新。不能保证它会提供足够有说服力的反馈，让用户每月支付订阅费以支持持续的开发成本。没有这种洞察力，用户就会放弃，这就是为什么现在有那么多健身带，位于卧室抽屉的后面。制造这些设备的公司很少有数据分析业务背景，而不是纯粹的硬件业务模式，这是没有帮助的。

开发 insight 业务模式的困难在于，为什么 2020 年出货的 5 亿只耳机几乎都集中在一件事上——播放内容，用户可以选择多种成熟的服务。我怀疑，随着时间的推移，我们将看到传感器回归到可听设备，因为耳朵是可穿戴设备测量生物特征的最佳场所。它是稳定的，不会移动太多，靠近血液流动，为测量核心温度提供了一个很好的位置。这是手腕所没有的一切。但该行业已经吸取了教训，数据也很难，这意味着这些传感器可能会作为次要功能出现，让拥有分析资源的公司有时间开发出令人信服的反馈。这就是我们所看到的苹果对手表所做的。这是一个漫长而缓慢的过程。幸运的是，对于耳塞来说，消费者购买耳塞已经有了一个令人信服的理由，这意味着可听设备可

---

以成为一个有用的平台来试验其他东西。所有这些都转化为市场上的巨大兴奋。耳塞是有史以来增长最快的消费产品，而且增长速度没有放缓的迹象。在本书的其余部分，我们将了解 Bluetooth LE Audio 如何增加兴奋感并提高增长率。

---

## 第2章 Bluetooth LE Audio 架构

Bluetooth 规范按照定义良好的过程开发。它从 New Work Proposal 开始，该提案开发用例并评估市场对任何新功能的需求。New Work Proposal 通常由一个小型研究小组生成，该小组由一些想要该功能的公司组成，然后由任何其他对它感兴趣的公司共享和评估。在这一点上，其他 Bluetooth SIG 成员被问到是否有兴趣帮助开发和原型，以便看看是否有足够的临界资源来实现它。

一旦投入被论证了，Bluetooth SIG 董事会将对其进行审查，并将其分配给一个小组，以将初始提案转化为一组需求，并使用例更加充实。对这些需求进行审查，以确保它们适合当前的 Bluetooth 技术架构，而不会破坏它，然后开始开发。一旦规范被认为或多或少是完整的，来自多个成员公司的实施团队就会开发原型，这些原型在互操作性测试事件中相互测试，以检查功能是否正常工作并满足原始要求。这也很好地检查了规范是否可以理解和明确。任何剩余的问题都会在那个阶段得到解决，一旦完成了，并且一切都显示正常，规范就会被采用并发布。只有在这一阶段，公司才被允许制造产品，使其合格并开始销售。虽然我们总是试图避免在此过程中规范蠕变，但我们几乎总是失败，因此新功能往往会添加到原始功能中。在 Bluetooth LE Audio 的发展中尤其如此，因为它已经从一种适度简单的助听器解决方案演变为目前的形式，为未来二十年的 Bluetooth 音频产品提供了工具包。为了帮助理解为什么我们最终获得了二十多个新规范，从原始用例的角度来看一下这个旅程，看看最终架构是如何确定的，这很有用。

### 2.1 The use cases

在 Bluetooth LE 音频开发的最初几年，我们看到四种主要的用例和需求推动了其发展。它从一组来自助听器行业的用例开始。这些重点是拓扑、功耗和延迟。

#### 2.1.1 The hearing aid use cases

助听器的拓扑结构是 Bluetooth Classic Audio profiles 向前迈出的一大步，因此我们将从它们开始。

##### 2.1.1.1 Basic telephony

图 2.1 显示了助听器的两个电话通讯用例，允许助听器连接到电话。这

---

是一个重要的要求，因为把手机放在助听器旁边放在耳朵里经常会造成干扰。



图 2.1 Basic Hearing Aid 拓扑

左边最简单的拓扑结构是从手机到助听器的音频流，它允许返回流，主要针对电话通讯。它可以被配置为使用助听器上的麦克风捕捉回音，或者用户可以对着他们的手机说话。这和 Hands-Free Profile (HFP) 的功能一样。但从一开始，助听器要求就有这样一个概念，即音频流的两个方向是独立的，可以由应用程序配置。换句话说，从电话到助听器的音频流和从助听器到电话的返回音频流可以被单独配置和控制，可以打开或关闭。图 2.1 右侧的拓扑结构超出了 A2DP 或 HFP 可以做的任何事情。在这里，电话向左右助听器发送单独的左右音频流，然后增加了来自每个助听器麦克风的可选返回流的复杂度。这引入了 Bluetooth Classic Audio profiles 无法完成的第二步，需要将单独的同步流传输到两个独立的音频设备。

#### 2.1.1.2 Low latency audio from a TV

这项要求的一个有趣的扩展源于助听器可以继续接收环境声音以及 Bluetooth 音频流。许多助听器不会遮挡耳朵(遮挡是行业术语，用于遮挡耳朵，如耳塞)，这意味着佩戴者总是可以听到环境和放大声音的混合。由于助听器内的处理延迟最小——小于几毫秒，因此不会出现问题。然而，在如图 2.2 所示的情况下，这成为了一个问题，其中一些佩戴者的家人通过电视的扬声器收听声音，而助听器用户则听到来自电视的环境声音的混合以及通过 Bluetooth 连接的相同音频流。

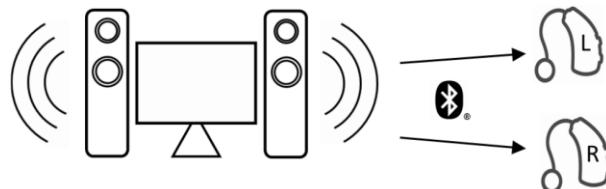


图 2.2 Bluetooth LE Audio 与环境音

如果两个音频信号之间的延迟远远超过 30-40 毫秒，它就会开始增加回声，使声音更难理解，这与助听器应该做的事情相反。30-40 毫秒的延迟比大多数现有 A2DP 解决方案可以提供的延迟要快得多，因此这引入了非常低

---

延迟的新要求。

虽然助听器的带宽要求相对适中，单声道语音的带宽为 7kHz，立体声音乐的带宽为 11kHz，但就现有的 Bluetooth codec，同时实现延迟要求来说，无法轻松满足这些要求。这开启了一项单独的研究，以确定合适的 codec 的性能要求，从而引入了 LC3 codec，我们将在第 5 章中介绍。

#### 2.1.1.3 Adding more users

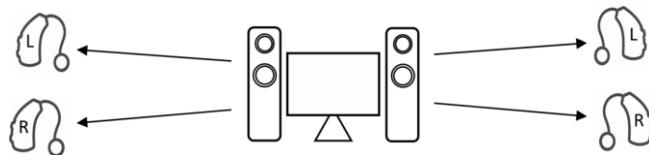


图 2.3 增加更多的听众

听力损失可能会在家庭中发生，而且通常与年龄有关，因此一个家庭中有多人佩戴助听器是很常见的。因此，新拓扑需要支持多个助听器佩戴者。图 2.3 说明了两个人的用例，他们都应该经历相同的延迟。

#### 2.1.1.4 Adding more listeners to support larger areas

拓扑结构也应该是可扩展的，以便多个人可以收听，就像在教室或护理院一样。这一要求取决于一个频谱，该频谱延伸到为当前的拾音线圈感应环路提供广播替代品。这需要一个 Bluetooth 广播发射器，它可以广播单声道或立体声音频流，能够被此范围内的任何数量的助听器接收，如图 2.4 所示。

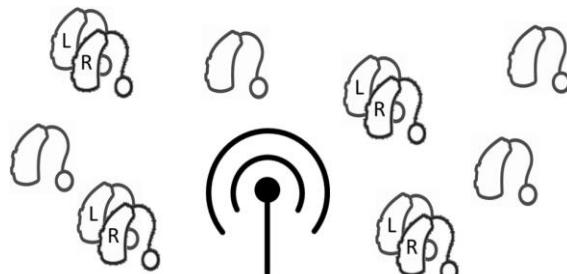


图 2.4 替代拾音线圈的广播拓扑

图 2.4 还承认这样一个事实，即有些人只有一只耳朵有听力损失，而另一些人则两只耳朵都有听力损失(这通常是不同程度的听力损失)。这意味着应该可以在播放单声道信号的同时广播立体声信号。它还强调了这样一个事实，即用户可能会佩戴来自两家不同公司的助听器以应对这些差异，或者一只耳朵里有助听器，另一只耳朵里有消费类耳塞。

#### 2.1.1.5 Coordinating left and right hearing aids

---

无论组合如何，都应该可以将一对助听器视为一组设备，以便两者都连接到相同的音频源，并且以一致的方式在它们上工作，例如音量控制等共同功能。这引入了协同的概念，即来自不同制造商的不同设备将同时接受控制命令，并以相同的方式进行解释。

#### 2.1.1.6 Help with finding broadcasts and encryption

使用拾音线圈，用户只有一个选项来获取信号——打开其拾音线圈接收器，该接收器从围绕它们的感应回路拾取音频，或将其关闭。一个区域中只能存在一个拾音线圈信号，因此您不必选择所需的信号。另一方面，这意味着你不能同时广播多种语言。

借助蓝牙，多个广播发射机可以在同一区域运行。这具有明显的优势，但引入了两个新问题 - 如何拾取正确的音频流，以及如何防止其他人收听私人对话？

为了帮助选择正确的流，重要的是用户可以找到关于他们是什么的信息，这样他们就可以直接跳转到他们的首选。这种体验的丰富性将明显不同，这取决于如何在助听器、电话或遥控器上实现对广播流的搜索，但规范需要涵盖所有这些可能性。许多公共广播不需要保密，因为它们加强了公共音频公告，但其他广播则需要保密。在家中这样的环境中，您不想获取邻居的电视。因此，音频流可以加密非常重要，需要向授权用户分发加密密钥。这一过程必须是安全的，但很容易做到。

除了低延迟，模拟当前助听器的使用增加了一些其他限制。当用户佩戴两个助听器时，无论他们接收的是相同的单声道还是立体声音频流，他们都需要在  $25\mu\text{s}$  内呈现音频，以确保音频图像保持稳定。这对于立体声耳机来说同样如此，但当左右设备可能来自不同的制造商时，这是一个挑战。

#### 2.1.1.7 Practical requirements

助听器非常小，这意味着按钮的空间非常有限。所有年龄段的人都可以佩戴，但一些老年人的手动灵活性有限，因此重要的是，可以在其他设备上实现调节音量和连接的控制，这更容易使用。这可能是音频源，通常是用户的手机，但助听器用户通常也有类似遥控器的小遥控器。它们的优点是可以立即工作。如果您想降低助听器的音量，只需按下音量或静音按钮；您不需要启用手机，找到助听器应用程序并从那里控制它。这可能需要太长时间，并且不是大多数助听器用户所喜欢的用户体验。他们需要一种快速方便的音量和静音控制方法，否则他们会把助听器从耳朵里拿出来，这不是他们想要的行为。

---

围绕音量还有另一个助听器要求，即音量水平(实际上是增益)应在助听器上实现。其基本原理是，如果音频流以线路级别传输，则可以获得最大的动态范围。对于正在处理声音的助听器，重要的是输入信号提供尽可能最佳的信噪比，特别是当它与来自环境麦克风的音频流混合时。如果音频的增益在 source 处降低，则会导致较低的信噪比。

助听器与耳塞或耳机之间的一个重要区别是，助听器大部分时间都佩戴，并持续活动，放大和调整环境声音，以帮助佩戴者更清晰地听到。用户不会经常把它们取下来，然后放回充电箱。每天佩戴一对助听器的典型时间为 9 个小时，尽管有些用户可能会佩戴 15 个小时或更长时间。这与耳塞和耳机非常不同，它们只在用户准备打电话或听音频时才佩戴。耳塞制造商在设计充电盒时非常聪明，鼓励用户在一天中定期给耳塞充电，给人的印象是电池寿命更长。助听器没有这种选择，所以设计师需要尽一切可能降低功耗。

其中一个需要电力的事情是寻找其他设备和维护后台连接。耳塞可以获得明确的信号，知道什么时候该这样做——当它们从充电盒中取出时。大多数还包含光学传感器，可以检测它们在耳朵里的时间，所以如果它们在你的桌子上，它们可以回到休眠状态。助听器无法获得相同的、明确的信号来启动 Bluetooth 连接，因为它们一直处于开启状态，一直作为助听器工作。这意味着他们需要保持与其他设备的持续 Bluetooth 连接，同时等待发生一些事情。这些连接可以是低占空比连接，但不会太低，否则助听器可能会错过来电，或者需要很长时间才能响应正在启动的音乐流应用程序。由于助听器可以连接到多个不同的设备，例如电视，电话甚至门铃，因此这样的连接会消耗太多电量，因此需要一种新的机制，以允许它们与一系列不同的产品进行快速连接，而不会破坏电池寿命。

### 2.1.2 Core requirements to support the hearing aid use cases

在定义了拓扑和连接的要求之后，很明显，需要向 Core specification 中添加大量新功能来支持它们。这导致了第二轮工作，以确定如何最好地满足 Core 中的助听器要求。

该过程的第一部分是分析是否可以通过扩展现有 Bluetooth 音频规范而不是将新的音频流能力引入 Bluetooth Low Energy 来支持新功能。如果这是可能的，它将提供与当前音频 profile 的向后兼容性。结论类似于 Bluetooth LE 首次开发时所进行的分析，它将涉及太多的妥协，最好在 Core 4.1 Low Energy 规范的基础上进行“clean sheet”设计。

---

对 Core 的建议是为了实现一个叫做 Isochronous Channels 新特性，它可以在 Bluetooth LE 中携带音频流，与现有 ACL 通道并行。ACL 通道用于配置、建立以及控制音频流，并且携带更普通的控制信息，比如音量、电话通讯以及媒体控制。Isochronous 通道能支持单向或双向的音频流，并且可以与多个设备建立起多 Isochronous 通道。这就将音频数据和控制面向独立开来，使得 Bluetooth LE 音频更具灵活性。

音频连接的鲁棒性是很重要的，意味着这些连接支持多重传，为了应对某些传输可能遭受干扰的实际情况。对于单播流，有个 ACK/NACK 确认机制，这样，一旦发射器知道数据已经被收到了，就能停止重传。对于广播，是没有反馈的，source 需要无条件地重传音频数据包。在侦测鲁棒性期间，很明显，用于保护 LE 设备免受干扰的调频机制可以改进，因此将其作为另一个需求添加进去。

广播需要一些新的概念，特别是设备如何在没有连接的情况下找到广播，这一个方面。Bluetooth LE 使用广播向其他设备通知它的存在。对于这些广播，想要创建连接扫描的设备，然后，连接到它们发现的设备以获取它支持的功能的细节、怎样连接——包括它发送时，包含的信息，它的跳频序列以及它做的事情。对于 Bluetooth LE 音频来说，这需要比普通 Bluetooth LE 广播多得多的信息。为了克服这个限制，Core 添加了一个新特性，Extended Advertisements (EA) 以及 Periodic Advertising trains (PA)，它允许上述信息携带在通用数据信道上数据包内，这些信道通常不用于广播。与此同时，对于接收设备，也添加了新的程序，使用该信息去确定广播音频流在哪里以及与其同步。

外部设备可以帮助找到广播流，即它可以通知接收器如何连接到广播流——本质上是接收器请求 remote control，并且被告知去哪里的能力。由一个叫做 PAST(Periodic Advertising Synchronization Transfer)的 Core 特性完成，它的关键是让广播获取简单化。PAST 对于助听器已经是有用的特性，因为扫描消耗许多电能。使扫描最少化对于帮助延长助听器电池的寿命时很有用的特性。

助听器需求也让一些另外的特性被添加到核心需求，主要围绕着性能和功耗。第一点就是在主机或控制器中实现新的 codec。其后是让它容易通过硬件实现，通常更加节能。第二点是限制传输或接收的最长时间，这影响了 Isochronous 信道内数据包结构的设计。这背后的原因是，许多助听器使用初级 zinc-air 电池，因为它们的功率密度很高。然而，这种电池化学依赖限

---

制电流尖峰和高功率电流消耗。不遵守这些限制会导致电池寿命显著缩短。满足它们塑造了 Isochronous 信道的整体设计。

Core 要求的最后两个补充，在开发中出现相当晚，是引入了 Isochronous Adaptation Layer (ISOAL) 和 Enhanced Attribute Protocol (EATT)。

ISOAL 允许设备将来自上层的 Service Data Units (SDUs) 转化成链路层中不同大小的 Protocol Data Units (PDUs)，反之亦然。需要这样做的原因是为了解决可能使用了推荐时序设定的 LC3 codec 的设备，codec 针对 10ms 进行了优化，同时连接到以 7.5ms 时序间隔运行的旧的 Bluetooth 设备。

EATT 是对标准 Bluetooth LE 为了同时运行多个 ATT 协议实例的增强。

Extended Advertising 特性是在 Core 5.1 版被采用的，Isochronous 信道、EATT 以及 ISOAL 是最近 Core 5.2 版本采用的，为其他 Bluetooth LE Audio 规范铺平道路。

### 2.1.3 Doing everything that HFP and A2DP can do

随着消费类电子行业开始意识到 Bluetooth LE Audio 功能的潜力，这些功能解决了它们多年来发现的许多问题，他们提出了第三轮需求的实用请求，以确保 Bluetooth LE Audio 能够完成 A2DP 和 HFP 可以做到的一切。他们指出，如果用户体验更差，没有人愿意使用 Bluetooth LE Audio 去替换 Bluetooth Classic Audio。

这些要求提高了对新 codec 的性能要求，并为媒体和电话控制引入了一组更复杂的要求。最初的助听器要求包括相当有限的与手机交互的控制功能，认定了大多数用户将直接控制其手机或电视上更复杂的功能，尤其是因为助听器的用户接口有限。许多消费类音频产品都较大，因此没有这种限制。因此，增加了新的电话和媒体控制要求，以允许更复杂的控制。

### 2.1.4 Evolving beyond HFP and A2DP

第四轮要求反映了音频和电话应用已超过 HFP 和 A2DP。今天的许多电话都是 VoIP，在一台设备上接到多个不同的电话是很常见的——无论是笔记本电脑、平板电脑还是电话。Bluetooth 技术需要一种更好的方式来处理来自多个不同承载者的呼叫。类似地，A2DP 没有预料到流媒体以及随之而来的搜索要求，因为它是在用户拥有本地音乐副本的时候编写的，很少做比选择本地文件更复杂的事情。今天，产品需要更复杂的媒体控制。他们还需要能够在不中断音乐流的情况下支持语音命令。今天的电话和会议应用程序

---

非常复杂，用户可以处理多种类型的呼叫以及音频流，这意味着他们在设备和应用程序之间进行更频繁的转换。HFP 和 A2DP 在架构上的固有差异一直使其难以实现，因此产生了一套最佳实践规则，构成了 HFP 和 A2DP 的 Multi-Profile 规范。新的 Bluetooth LE Audio 架构将不得不超越这一点，并在设计上结合 multi-profile 支持，在设备和应用程序之间以及单播和广播之间实现健壮的和可以互操作的转换。

随着越来越多的消费者开始了解 telecoil 和助听器的广播功能是如何工作的，他们开始意识到广播可能有一些非常有趣的大众消费者应用。最重要的是他们意识到它可以用来分享音乐。这可能是朋友们在手机上分享音乐，在无声的迪斯科舞厅，或者在咖啡馆和公共场所“无声”的背景音乐。公共广播设施，例如为佩戴助听器的人提供旅行信息的设施，现在每个人都可以通过 Bluetooth 耳机访问。音频共享的概念诞生了，我们将在第 12 章中详细讨论。

潜在的新用例开始激增。如果我们可以同步两个耳机的立体声声道，为什么不同步环绕声？公司热衷于确保它支持智能手表和腕带，它们可以作为遥控器，甚至可以作为嵌入 MP3 播放器的音频源。低延迟对游戏社区来说令人兴奋。微波炉可以告诉你晚餐何时煮熟(你可以看出这个想法来自工程师)。随着公司看到它如何使他们的客户，他们的产品策略受益并影响语音和音乐的未来使用，用例的数量继续增长。

## 2.2 The Bluetooth LE Audio architecture

Bluetooth LE Audio 架构是分层构建的，之前的所有其他 Bluetooth 规范也是如此。这如图 2.5 所示，其中显示了与 Bluetooth LE Audio 相关的主要新规范模块(现有关键规范块呈灰色或虚线)。

底部是我们的 Core，其包含了 radio 和链路层，(统称为 Controller)。它负责通过空中发送 Bluetooth 数据包。在它之上的是 Host，其主要任务是告诉 Core 如何处理特定的应用程序。控制器和主机的分开是历史性的，反映了 Bluetooth radio 可以放到 USB 记忆棒里面或者 PCMCIA 卡里面来卖，Host 可以通过 PC 的一个软件应用来实现的历史。如今，Host 和 Controller 都可以放进单个芯片。

在主机端，有了一个新的叫做 Generic Audio Framework 或 GAP 的结构。这是一个音频中间件，其包含了被认为是通用的所有功能，比如，可能被多于一个的音频应用所使用的功能。**Core 和 GAF 是 Bluetooth LE Audio**

---

的核心。它们提供了很大的灵活性。最后，在协议栈的最高处，叫做“top level” profiles，它们将应用的具体信息添加到了 GAP 规范。

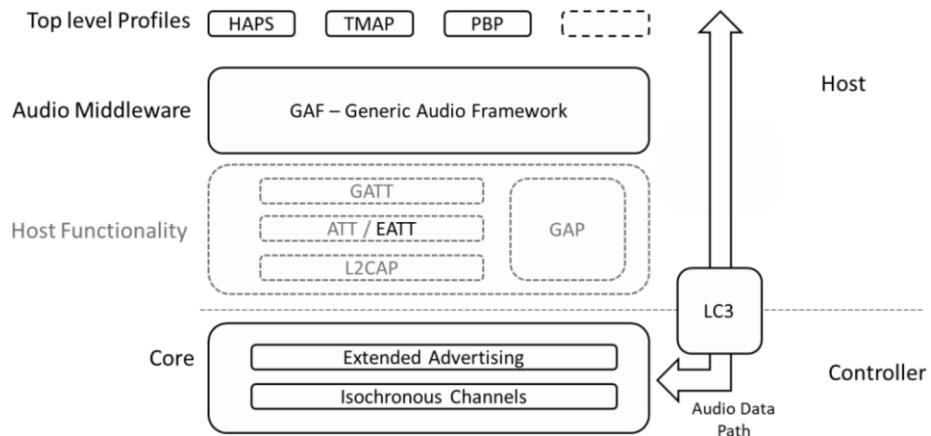


图 2.5 Bluetooth LE Audio 架构

完全可以使用 GAP 规范构建可以互操作的 Bluetooth LE Audio 应用程序。定义了其中的各个规范，以确保基本的互操作性，这将使得可以在任意两个 Bluetooth LE Audio 设备之间传输音频。顶层的 profiles 规范主要添加特定于特定类型的音频应用程序的功能，强制的功能，GAF 仅定义为可选的功能，并添加特定于应用程序的功能。其目的是，构建基于 GAF 中的特性，顶层的 profile 相对简单。

乍一看，Bluetooth LE Audio 架构看上去很复杂，因为我们在 Generic Audio Framework 中使用了 23 中不同的规范，同时扩展了 Core 以及新的 LC3 codec。但这是有逻辑的。每一个规范都试图封装你对音频流的建立的不同方式和控制的不同方面的特定元素。本章的剩余部分，我将主要解释每一个，以及它们怎样相互适配在一起的。然后，在本书的剩余部分，我们将查看每一个独立的规范怎样运行以及它们之间怎样交互。

### 2.2.1 Profiles and Services

GAF 中的所有规范都是使用图 2.6 中的标准 Bluetooth LE GATT 模型，按 Profiles 或者 Services 来分类的。

在 Bluetooth LE 中，Profiles 和 Services 可以被认为是 Clients 和 Servers。Services 是在 state 所处位置实现的，而 profile 规范描述了 state 的行为以及其处理过程。Services 规范定义了一个或多个 characteristics 用来表示状态机的单个特性或状态。它们也可以是 Control Point，导致状态机状态间转换。

---

Profiles 作用于这些 characteristics，读取或写入，以及在值发生变化时得到通知。每个设备可以充当客户端在 server 上进行操作。

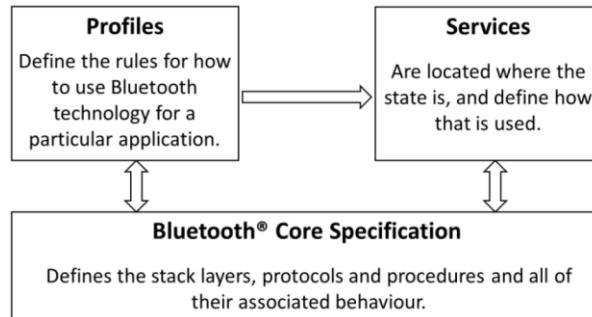


图 2.6 Bluetooth LE Profiles 以及 Services 模型

传统上，在经典的 Bluetooth profiles(没有相应的服务)中，只有一个 client 和一个 server 的一对关系，所有都在一个 profile 规范中描述。在 Bluetooth LE Audio 中，多对一的拓扑结构是很常见的，特别是在比如音量控制和广播源的选择这些功能里面，在这些功能里面，用户可以有多个实现 profile 规范并且扮演 client 角色的设备。在大多数情况下，这些设备按照先到先得的方式进行。

Bluetooth LE Audio 中能够使用的不同控制 profile 的数量，使得 EATT 增强功能被放到了 Core 里面。Profiles 和 Services 使用 Attribute Protocol (ATT)通信，但是 ATT 一次只执行一个命令。如果多个命令出现了，第二个命令就会延迟，因为 ATT 是一个阻塞型协议。为了绕开这个问题，Extended Attribute Protocol (EATT)被添加到了 Core 5.2 版本中，允许多个 ATT 实例同时运行。

图 2.7 提供了 Bluetooth LE Audio 架构的概览，为组成 GAF 的所有 18 个规范以及当前 4 个顶层的 profiles 指定一个名称，或者更准确说是一组字母。虚线框表示协同工作的 profiles 以及 services 集合。大多数情况下，profile 和 service 之间存在一对一的关系，尽管在 Basic Audio Profile (BAP) 以及 Voice Control Profile (VCP)的情况下，一个 profile 可以在三个不同的 service 上运行。Public Broadcast Profile (PBP)就比较反常，它是一个没有 service 的 profile，但这是广播的结果之一，当没有连接时，你无法进行传统的 Client-Server 交互。

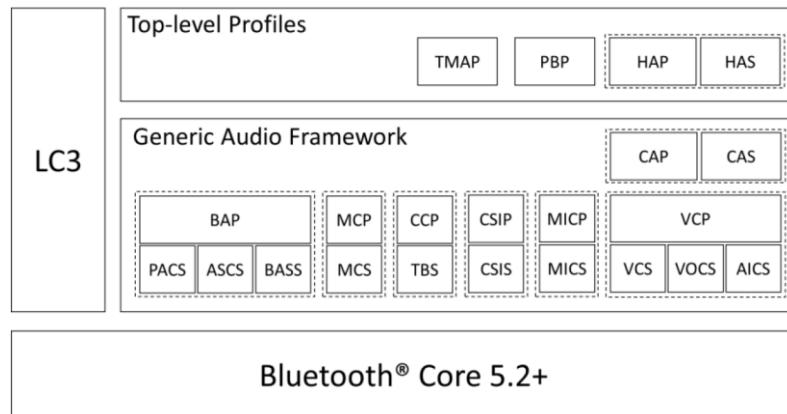


图 2.7 Bluetooth LE Audio 规范概览

### 2.2.2 The Generic Audio Framework

我们现在可以看看 GAF 的组成部分了。各种规范之间有大量的交互，这使得很难在它们之间绘制清晰的层次结构或关系集合，但是它们可以大致分为四个功能组，如图 2.8 所示。

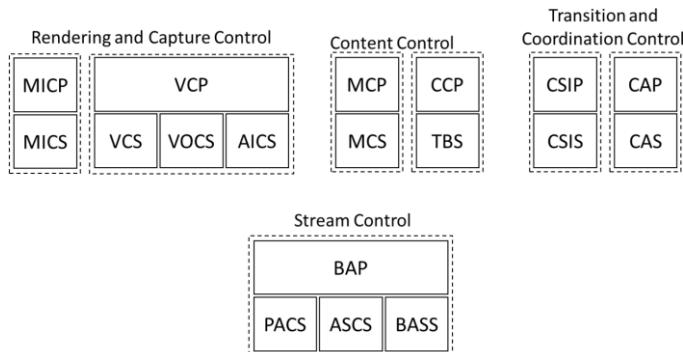


图 2.8 Generic Audio Framework 中规范的功能性分组

这种分组主要是为了解释。在真正的 Bluetooth LE Audio 的实现中，这些规范的大多数之间有着或多或少的交互。完全可以只使用其中的一小部分来完成可以运行的产品，但是要设计功能丰富、可以互操作的产品，则需要它们中的大多数。

### 2.2.3 Stream configuration and management – BAPS

从图 2.8 的底部开始，我们有一个由 4 个规范组成的组，统称 BAPS 规范。这四个规范形成了 Generic Audio Framework 的基础。其核心是 BAP—Basic Audio Profile，用来建立和管理单播和广播音频流。作为 profile，它使用三种服务：

- 
- PACS——Published Audio Capabilities Service，用来暴露设备的能力，
  - ASCS——Audio Stream Control Service，定义了用于建立和维护 unicast 音频流的状态机，以及
  - BASS——Broadcast Audio Scan Service，定义了发现和连接到广播音频流以及分发广播加密密钥的过程。

在它们之间，它们负责我们建立用于承载音频数据的 Isochronous 信道的方式。它们还定义了用于 LC3 codec 配置的标准集合，以及用于广播和单播应用的 Quality of Service (QoS) 设定的相应范围。

每个单独的 Isochronous 信道的状态机都是为单播和广播定义的，这两者都将音频流通过已配置的状态转到流动状态，如图 2.9 的简化状态机所示。

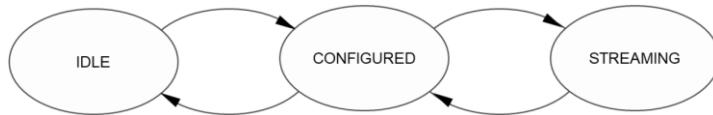


图 2.9 简化的 Isochronous 信道状态机

对于单播，状态机定义在 ASCS 规范中。状态是在 Server 中的各个音频端点中，Client 控制的定义在 BAP 中。对于广播，发送器和接收器之间没有连接，Client-Server 模型的概念有点单薄。因此，状态机仅为发送器定义，并且仅受其本地应用程序的控制。对于广播，接收器需要探测音频流的存在以及接收它，没有方法去影响它的状态。

多个单播或广播 Isochronous 信道按组绑定在一起(我们在第 4 章中探讨)。BAP 定义了这些组以及构成它们的 Isochronous 是怎样组合到一起，用于广播和单播的。

你可以仅使用这三个规范就造出 Bluetooth LE Audio 产品，BAP、ASCS 以及 PACS 用于单播，BAP 单独用于广播(尽管如果你想使用手机或遥控器帮助查找广播，你需要将 PACS 和 BASS 添加进去)。就功能而言，这是一个相当有限的设备——只需要建立音频流，使用它传输音频和停止音频流。但是，通过能够做到这一点，BAPS 规范集合为所有 Bluetooth LE Audio 设备提供了基础水平的互操作性。如果两个 Bluetooth LE Audio 设备具有不同的顶层 profiles，它们仍然应该能够使用 BAP 建立起音频流。它可能具有有限的功能，但是应该提供可接受的性能水平，消除 Bluetooth Classic Audio 中出现的 multi-profile 不兼容问题，在经典 Bluetooth 音频中，没有相同的音频 profile 的设备，不能协同工作。

## 2.2.4 Rendering and capture control

建立完成音频流之后，用户想要控制音量，在他们耳朵中呈现的音频流以及麦克风获取的音频流的音量。

音量是一个出乎意料地困难话题，因为有许多地方可以调节音量——在 source 端、在助听器上、耳塞或者扬声器上、或者另外一个“遥控”设备上，可以是一个智能手表或者单独的 Controller。在 Bluetooth LE Audio 中，最终的音量增益是在助听器、耳塞或者扬声器，而不是在输入音频流中执行的(尽管顶层的 profile 也可能需要)。基于该假设，Volume Control Profile (VCP) 定义了 Client 怎样管理 Audio Sink 上的增益。该增益的状态是在 Volume Control Service (VCS) 中定义的，每个 audio sink 都上都有一个 VCS 实例。音量可以表示为绝对或相对的，也可以是静音的。

如果有多个音频流，如耳塞和助听器，就需要第二种服务。VOCS - the Volume Offset Control Service，有效地用作平衡控制，允许调整多个设备的相对音量。这些可以在不同的设备上呈现，比如独立的左右耳塞或者扬声器，或者单个设备，比如一副耳机或音箱。如图 2.10 所示，Audio Input Control Service (AICS) 认为大多数设备具有支持多种不同音频流的能力。AICS 提供用来控制多种不同输入(可以混合在一起并且在你的耳塞或者扬声器里呈现)的能力。插图说明这三个服务怎样在具有 Bluetooth、HDMI 以及麦克风输入的音箱里使用。

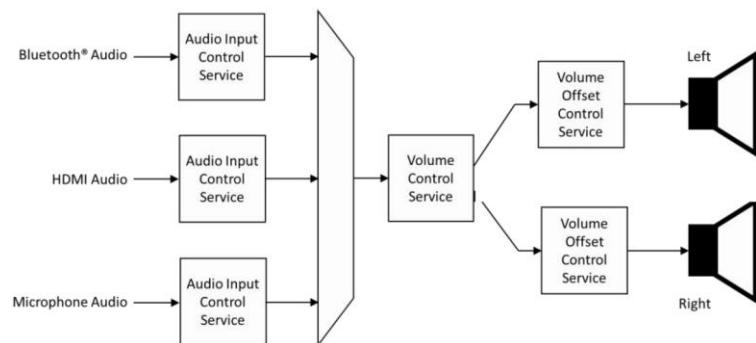


图 2.10 音频输入控制服务(AICS)、音量控制服务(VCS)和音量偏移控制服务(VOCS)

对于助听器，输入可能是 Bluetooth 音频流、听过环境音频流的麦克风，以及接收来自音频环路流的 telecoil 天线。任意时间点，佩戴者可能想听到这些不同输入的混合声音。AICS 就支持这样的灵活性。

音量服务的一个重要特性，就是有任何的改变它们就会通知运行 Voice Control Profile 的 client 设备。这可以确保所有潜在的 Controller 都与它们状

---

态更新保持一致，不论这种改变是出现在 Bluetooth 链路或者来自本地的音量控制。这确保了它们都具有同步过的音量状态，这样，用户可以通过它们中的任意一个来改变音量，而不会因为它们具有过时的当前音量状态而产生的意想不到的效果。

MICP 和 MICS 是一对互补的规范，Microphone Control Profile 和 Service，负责安装在助听器和耳塞内部的麦克风的控制。如今，这些设备通常都包含多个麦克风。助听器监听环境音，同时也监听通过 Bluetooth 收到的音频。随着耳塞越来越复杂，我们越来越多地看到类似的环境音功能被内置到耳塞内，某种程度的透明传输也越来越普及。

MICP 与 AICS 以及 MICS 协同工作，控制麦克风的整体增益及静音。它们通常用于要发送到 Bluetooth 音频流的捕获音频的控制，也有更广泛用途。图 2.11 展示了它们的使用。

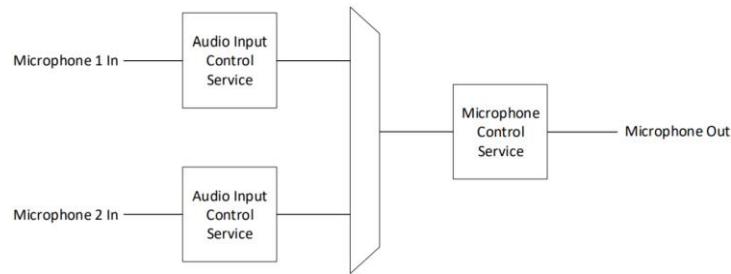


图 2.11 音频输入控制服务(AICS)与麦克风控制服务(MICS)一起使用

### 2.2.5 Content control

明确了音频流是怎样建立和管理的，以及音量和麦克风输入怎样处理，我们开始讨论内容控制。我们收听的内容是在 Bluetooth 规范之外生成的——它可能是流媒体音乐、直播电视、手机通话或者视频会议。内容控制要做的就是允许在启动、停止、应答、暂停和音频流的选择方面进行控制。这些就是嵌入到 HFP 以及伴随 A2DP 的 Audio/Video Remote Control Profile (AVRCP)中的控制类型。在 Bluetooth LE Audio 中，它们被分成了两个规范集合——一种用于各种形式的电话，另一个用于媒体。关键区别在于，电话通讯是关于一个或多个呼叫状态，这通常反应电话通讯服务的状态，而媒体控制作用于音频流的状态——播放时间和播放方式以及选择方式。因为它们与音频流分离，因此现在可以用于帮助控制转换，比如当你接收到电话呼叫时暂停音乐回放、电话完成之后重新播放。对于这对规范，Server 位于 primary audio source——通常是手机、PC、笔记本电脑或 TV，而 Profile 是

---

在接收设备上实现的，比如助听器或耳塞。同呈现和捕获控制一样，多设备可以充当 Client，因此通过智能手机和耳塞，可以控制电话通讯以及媒体状态。

Media Control Service (MCS) 驻留在音频媒体源上，并且反映了音频流的状态。状态机允许使用 Media Control Profile (MCP) 的 Client 通过 Playing、Paused 和 Seeking 状态去转换每个媒体源。简单地说，它允许耳塞控制 Play 和 Stop。然而，MCS 远未止步于此，它提供了当今用户对内容播放器期望的所有功能。它也提供了更高层级功能，用户可以搜索曲目、修改播放顺序、设定分组以及调整播放速度。它定义了元数据结构，可以用于标识音轨并使用现有的 Object Transfer Service (OTS)，允许 client 在 Server 上执行媒体搜索，或更典型地，在 Server 后面的应用上。所有这些意味着，正在运行 Media Control Profile 的适当复杂的设备能够重新创建对音乐播放器的控制。

电话通讯控制通过使用 Telephone Bearer Service (TBS) 类似的方式来处理，它驻留在通话需要的设备上(典型地，如手机、PC 或笔记本电脑)，以及在 TBS 实例中，通过向状态机写入来控制呼叫的补充 Call Control Profile (CCP)。TBS 和 CCP 已经超越了 Hands-Free Profile 的限制，以适应我们现在以多种不同形式使用电话的事实。它不再只是传统电路交换器和蜂窝承载器，而是基于 PC 和 web-based、使用多种不同类型的承载服务的通信和会议应用。TBS 使用通用的状态机将呼叫的状态暴露出来。它支持多个呼叫、呼叫处理以及连接、主叫方 ID、带内以及带外铃声选择，并暴露呼叫信息，比如信号强度。

TBS 和 MCS 都证实服务器设备上可能存在多个媒体源以及多个不同的呼叫应用。为了适应这种情况，两种服务都可以多次实例化——每个应用程序实例一次。这允许具有互补 profile 的 Client 单独控制每个应用程序。或者，可以使用服务的单个实例，媒体或呼叫设备使用特定的实现将 profile 命令引导到正确的应用程序。TBS 和 MCS 的单个实例被叫做 Generic Telephone Bearer Service (GTBS) 以及 Generic Media Control Service (GMCS)，并分别包含在 TBS 和 MC 规范中。我们将在第 9 章中更详细地研究这些。

### 2.2.6 Transition and coordination control

接下来，我们讨论 Transition 和 Coordination 控制规范。它们的目的是将其他规范粘合在一起，为顶层 profile 提供一种调用它们的方法，而不必

---

关心它们自己创立的更多的细节。

Isochronous 信道的一个最主要增强就是能够将音频流传输到多个不同的设备并且在精确的时间将它们呈现出来。它最通常的应用是将立体音频流传到左右耳塞、扬声器或助听器。呈现的拓扑和同步化是由 Core 和 BAP 处理的，但是确保控制操作是同时出现，无论是改变音量还是在连接之间转换则不是由 Core 和 BAP 处理的。这就是 Coordinated Set Identification Profile (CSIP)和 Coordinated Set Identification Service (CSIS)的作用所在。

当两个或多个 Bluetooth LE Audio 设备想要一起使用时，它们就叫做 Coordinated Set，并且可以通过使用 Coordinated Set Identification Service 将它们相互关联起来。这就可以允许另外的 profiles，特别是 CAP，将它们视为一个单个实体。它引入了 Lock 和 Rank，以确保当在音频连接间发生转换时，不论转换到新的单播或广播音频流，该集合的成员总是一起做出反应。这可以防止新连接仅仅在在集合中设备中的一部分起作用，比如 TV 连接到你的左耳塞，而你的手机连接到你的右耳塞。要设计成为 Coordinated Sets 成员的设备通常在制造期间就被配置成了集合成员。

未配置成为 Coordinated Set 成员的多个设备仍然可以在 GAP 中作为 ad-hoc 集合使用。这种情况下，它们需要应用程序对它们单独地进行配置。意味着它们不能受益于 CSIS 的 locking 特性，这可能导致对 ad-hoc 集合成员的连接是不同的。

CAP——Common Audio Profile，引入 Commander 角色，该角色将可用于 Bluetooth LE Audio Stream 控制的功能集合在一起。Commander 相比之前的 Bluetooth 规范来说是一个重大的变化，它允许对于音频远程控制可以无处不在、分布式的。对于加密广播特别有用，因为它提供了一种将广播传输转化为私有收听体验。我们将在第 8 章中更加详细探讨这一点。

CAP 使用 CSIS 和 CSIP 将设备绑在一起，确保程序可以都应用到它们。它也引入了 Context Types 以及 Content Control IDs 概念，允许应用程序基于控制设备、音频数据应用实例以及哪些应用程序是可用的为依据，决定音频流的建立和控制。这用于通知不同音频流的转换，无论是由设备上的不同应用程序触发，还是来自不同设备对于音频连接的请求。许多功能都是基于引入到 Bluetooth LE Audio 的新概念。这些将在第三章详细说明。

### 2.2.7 Top level Profiles

最后在，在 GAF 规范的上面，有顶层 profiles，它们对于特定的音频应

---

用实例提供附加的要求。它们中的第一个就是 Hearing Access Profile 以及 Service (HAP 和 HAS)，其涵盖了助听器生态系统的应用，Telephony and Media Audio Profile (TMAP)，明确规定了更高质量的 codec 的设置和更复杂的媒体以及电话通讯控制的使用，Public Broadcast Profile(PBP)，用于帮助用户选择全局地可互操作的广播音频流。Public Broadcast Profile 因为没有伴随的 service，是不符合常规的，但是，这是由于广播的性质导致的结果，广播是没有任何用于 Client-Server 交互的连接的。

### 2.2.8 The Low Complexity Communications Codec (LC3)

尽管不是 GAF 的一部分，Bluetooth LE Audio 的版本里面包含了新的、高效的 codec，称为 LC3，是 Bluetooth LE Audio Streams 的强制 codec。这为电话语音、宽带和超宽带语音以及高质量音频提供了出色的性能，是 BAP 中的强制 codec。每一个 Bluetooth LE Audio 产品必须支持 LC3 codec 以确保互操作性，但是附加的和特定知识产权的 codec 可用根据制造商的要求添加进去。LC3 将音频编码为单个音频流，因此，立体声被编码为独立的左右音频流。意味着 GAF 能够配置单播流到耳塞，仅仅携带耳塞需要的音频。发送音乐的广播发送器通常在它的广播中包含左音频流和右音频流。单个设备只需要接收并解码与它们想要呈现的 stream 相关的数据。

## 2.3 Talking about Bluetooth LE Audio

编写了 20 多个规范产生了许多新的缩写，包括每个规范的名称。随着时间的推移，工作组提出了不同的方式来称呼这些名称——有时是首字母缩写词(你把它们称为单词)，有时是首字缩写词(只把字母发音)，有时是两者的混合。下表列出了最常用语音的当前发音，以帮助任何人讨论 Bluetooth LE Audio。

### 2.3.1 缩略词

缩写	意义
AICS	Audio Input Control Service
ASE	Audio Stream Endpoint
ATT	Attribute Protocol
BAP	Basic Audio Profile
BAPS	The set of BAP, ASCS, BASS and PACS
BASE	Broadcast Audio Source Endpoint
BASS	Broadcast Audio Scan Service
BIG	Broadcast Isochronous Group
BIS	Broadcast Isochronous Stream
CAP	Common Audio Profile
CAS	Common Audio Service
CIG	Connected Isochronous Group

---

CIS	Connected Isochronous Stream
CSIP	Coordinated Set Identification Profile
CSIS	Coordinated Set Identification Service
CSIPS	The set of CSIP and CSIS
EATT	Enhanced ATT
GAF	Generic Audio Framework
GAP	Generic Access Profile
GATT	Generic Attribute Profile
HAP	Hearing Access Profile
HARC	Hearing Aid Remote Controller
HAS	Hearing Access Service
HAUC	Hearing Aid Unicast Client
INAP	Immediate Need for Audio related Peripheral
L2CAP	Logical Link Control and Adaptation protocol
MICP	Microphone Control Profile
MICS	Microphone Control Service
PAC	Published Audio Capabilities
PACS	Published Audio Capabilities Service
PAST	Periodic Advertising Sync Transfer
PBAS	Public Broadcast Audio Stream Announcement
PHY	physical layer
QoS	Quality of Service
RAP	Ready for Audio related Peripheral
SIRK	Set Identity Resolving Key
TMAP	Telephony and Media Audio Profile
TMAS	Telephony and Media Audio Service
VOCS	Volume Offset Control Service

表 2.1 Bluetooth LE Audio 缩写

### 2.3.2 首字母

其余的组成的缩写就简单按照单个字母的发音，比如 CCP 以及 LC3。

Bluetooth LE Audio 最通用的首字母是: ACL, AD, ASCS, BMR, BMS, BR/EDR, CCID, CCP, CG, CSS, CT, CTKD, EA, FT, GMCS, GSS, GTBS, HA, HCI, IA, IAC, IAS, INAP, IRC, IRK, LC3, MCP, MCS, MTU, NSE, PA, PBA, PBK, PBP, PBS, PDU, PTO, RFU, RTN, SDP, SDU, TBS, UI, UMR, UMS, UUID, VCP and VCS。它们都在术语汇编中解释。

### 2.3.3 不规则发音

OOB 是一个反常的现象，因为它总是全称“out of band”，尽管它在文本上使用简写。

以上对 Generic Audio Framework 主要部分的一次短暂参观。我们将在未来看到更多的规范出现在 GAF 中，但目前，上面描述的规范模拟并扩展了 Bluetooth 目前具有的经典音频 profiles 的音频功能。

在本书剩余部分中，我们将看见 BAPS (BAP, BASS, ASCS 和 PACS) 如何构成 Bluetooth LE Audio 的核心。另外的规范增加了可用性和功能性，

---

而 CAP 将其粘合在一起。作为第一步，我们将研究解决 Bluetooth LE Audio 需求所必须的一些新概念。

---

## 第3章 Bluetooth LE Audio 中的新概念

为了向设计者提供他们需要的灵活性，新的 Bluetooth LE Audio 规范引入了一些重要的新概念。本章，我们将看一下它们的用途以及为什么需要它们。因为这些特性紧密地集成到了规范中，随着我们深入了解 Core 和 GAF 的细节，下面的一些描述将逐渐变清晰。然而，在这个阶段介绍它们是有用的，因为它们贯穿了后面大部分内容。

### 3.1 Multi-profile by design

正如我们所见，Bluetooth Classic Audio 面临的挑战之一就是 multi-profile 问题，multi-profile 使用户可以在流媒体音乐、打电话以及使用语音识别之间切换。随着音频用例复杂度不断增加，这个问题没有得到改善。两个成功的经典 Bluetooth audio profile——Hands-Free Profile(HFP)和音乐流媒体(A2DP)，设计的时候是假设用户使用其中的一个或者另一个，当用户通话转到音乐时，会开启新的、独立的会话。很快地，它们是非独立功能这点就变得明显了，但是手机用户会期望一个用例中断另外一个。多年来，行业制定了解决此问题的一些规则和方法，最终在 2013 年发布了 Multi Profile Specification(MPS)，该规范基本上收集了在这些 profile 之间常见的转换规则集合。

Multi Profile Specification 并不是特别灵活，也没有预见新的用例的出现，如语音识别、语音助理以及来自 GPS 卫星导航的中断。基础规范也不是特别通用。尽管 HFP 已经经历了多个版本，并产生了大约 20 个补充规范，解决了汽车和电话交互方面的问题，但是它仍然难以跟上 non-cellular VoIP 电话通讯应用。不仅仅是它，也包括 A2DP，都没有充分解决音频不断变化的特性，即用户在一天中连接到多个不同的音频源，也可能拥有多个不同的耳机和耳塞。

从一开始，Bluetooth LE Audio 的发展理念就是支持“multi-profile by design”。它认识到用户可能拥有多个耳机和音频源，按照 ad-hoc 方式，不断地改变连接。音频广播的添加使得这一点更加重要，因为预计在场馆、商店、休闲设施以及旅行中对广播的使用，将扩大用户每天可能建立的不同连接的数量。显然，提供工具让用户无缝化使用非常重要。

---

### 3.2 The Audio Sink led journey

伴随这些需求，人们意识到手机不一定会成为音频世界的中心，这在 Bluetooth Classic Audio 规范典中一直是一个默认的假设。随着不同的可用音频源越来越多，考虑用户在一天中如何连接他们的耳塞或助听器就变得重要了。这改变了人们对以手机为中心的音频世界的看法，取而代之的是以 Audio Sink 为主导的旅程。

将电话不再是你所听内容的仲裁者作为一种方法。相反，它假设用户越来越可能整天都佩戴一副助听器或耳塞，不断地改变它们的音频源。它可能从闹钟开始，随后让你的语音助手打开你的收音机，来自公共车站的信息、电脑的语音控制、接听电话、门铃打断以及回家之后再 TV 前的放松，直到微波炉告诉你晚餐准备好了。这种关于谁是控制者的观念转变大多来自助听器佩戴者，他们通常在一天的大部分时间都佩戴助听器。大部分时间里，助听器在没有任何 Bluetooth 连接的情况下工作，只能听见并增强佩戴者周围的声音。但是佩戴者可以将其连接到大量基础设施音频源，超市结账机、公共交通信息、剧院、礼堂和电视，以及支持 Bluetooth Classic Audio 的手机和音乐播放器。一旦它变得广泛使用，更多的消费者可能会使用它。

当然，电话仍然会是音频的主要来源，但是 PC、笔记本电脑、TV、平板 Hi-Fi 以及语音助手也将越来越多。从门铃到烤箱，一系列智能家居设备也加入了他们的行列。与此同时，消费者正在购买更多耳机。拥有一对睡眠耳机、一套耳塞、一对立体声耳机和一个音箱或 Bluetooth 扬声器的人并不少见。当潜在的组合成倍增加时，用户接口必须能够适应这一范围的不同连接和他们之间的转换。

### 3.3 Terminology

Bluetooth LE Audio 规范的不同部分对发送和接收设备及其功能使用不同的名称。每个规范都将这些称为角色，因此当我们从 Core 到顶层 profile 时，对于每个设备，我们总共得到至少四个不同的角色。这样做有个很好的理由，协议栈每往一步，在这些角色里，都会具有更多的独特性，结果是设备可以实现特定的组合以实现功能目标。但是这些角色可能会让人迷惑。Core 里面提到 Central 和 Peripheral 设备，Central 设备发送命令，Peripheral 响应。在 BAP 里面，它们叫做 Client 和 Server。再往上，CAP 定义了 Initiator 和 Acceptor。Initiator 存在于 Central 设备中，负责建立、规划以及管理 Isochronous Streams。Acceptor 是参与到这些数据流中的设备——典型

---

的如助听器、扬声器和耳塞。总是会有一个 Initiator，但是可以有多个 Acceptor。不同的术语如图 3.1 所示。

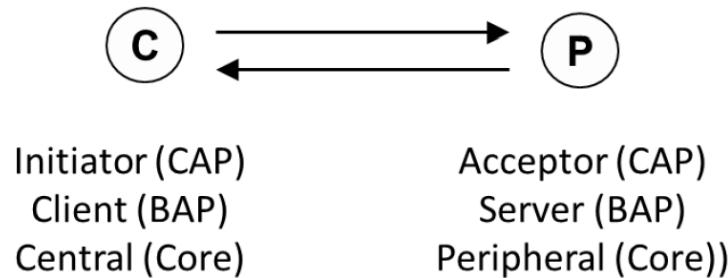


图 3.1 Bluetooth LE Audio 角色

我将在大多数情况下使用 Initiator 和 Acceptor 角色(尽管它们是技术上的角色)，因为我认为它们最好地解释了 Bluetooth LE Audio 工作方式。Initiator 和 Acceptor 都可以同时发送和接收音频流，并且它们都能够包含多个 Bluetooth LE Audio Sink 和 Source。需要记住的重要事情是，Initiator 是执行 Isochronous Streams 配置和规划的设备，Acceptor 是接收这些 streams 的设备。这一概念在单播和广播中都适用。我使用了一个缩写。因为只有 Initiator 可以广播音频流，我将保留几个单词，并将 Initiator 作为 Broadcast Sources，称为 Broadcasters，除非需要明确。

此时，值得稍微转移一下注意力，去看看 Bluetooth LE Audio 中使用的一些术语。将引入一些我们还没有讨论过的特性，我们将在第四章中详细介绍这些特性，在我们讨论 Isochronous Streams 时。

在整个规范中，有许多不同的名称和短语描述通道和数据流。取决于你正在查看的规范，它们有一些非常不同、有时又略有不同的含义，我尝试在图 3.2 中抓住主要的含有。

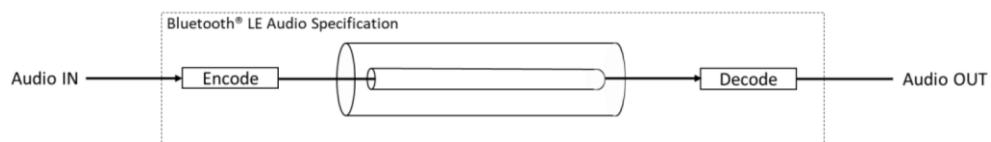


图 3.2 Bluetooth LE Audio terminology

该图显示了音频数据从左到右的概念传输。在最左边，我们有音频输入，通常是模拟信号。在中显示为 Audio IN。在另一端，我们看到 Audio OUT 呈现的音频，它可能在扬声器或耳塞上。这两点上的音频称为 Audio Channel(两端都是相同的 Audio Channel)，相当于 MP3 播放器和扬声器之间有线连接。Audio Channel 在 BAP 中定义为单向流入 Bluetooth 设备输入端，然后从另外一端出来的音频流。(实际上，“out”通常会直接输入到耳机变

---

换器或扬声器)。音频通道的输入输出细节是特定于实现的，并且在 Bluetooth 规范之外。音频是什么以及无线传输和解码后如何处理音频不在本规范的范围内。

音频输入是如何传送到音频输出，是在 Bluetooth 规范中定义的。它们由图 3.2 中的虚线框表示。音频使用新的 LC3 codec 进行编码和解码，除非，实现需要使用特定的附加 codec。可以使用其他的 codec，但所有的设备必须支持 BAP 中定义的一些基本 LC3 配置，以确保互操作性。编码器产生编码的音频数据，该数据进入 Isochronous Streams 的 payload 中。

一旦音频数据被编码，它就被放进 SDU(Service Data Unit)，Core 将其转换为 PDU(Protocol Data Unit)，PDU 就会被发送给接收设备。一旦收到 PDU，就将其重构成 SDU，以将其传送给 codec。Isochronous Stream 用于描述封装在 PDU 中、从编码输出到解码输入的 SDU 的传输。它包括重传和用于同步多个 Isochronous Streams 所需要的任意缓冲。Isochronous Stream 上的编码音频数据流在 BAP 中定义为 Audio Stream，并且与音频通道一样，始终是单向的。不同术语之间的关系如图 3.3 所示。

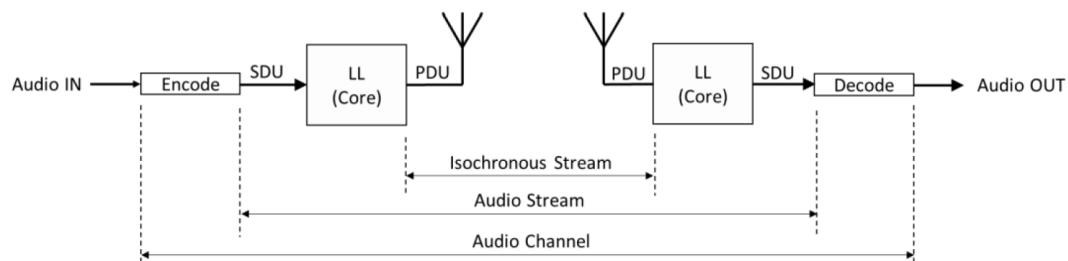


图 3.3 数据流术语的表示

Core 将服务于同一个应用程序的 Isochronous Streams 一起放到 Isochronous Group。对于单播流，它被称为 Connected Isochronous Group (CIG)，它包含了一个或多个 Connected Isochronous Streams (CIS)。对于广播，叫做 Broadcast Isochronous Group (BIG)。当在同一个方向上，承载音频数据的 Isochronous Group 中存在多个 Isochronous Stream 时，这些 Isochronous Streams 在应用层，彼此之间应该有时间上的关系。时间关系指的是预期同时呈现或捕获的音频通道。典型的应用是将左右立体音频流呈现在两个独立的耳塞，并且具有一个或两个来自于它们的麦克风的返回音频流，这突出了一个事实，CIG 或 BIG 可以包含流向多个设备的 Isochronous Streams。

Bluetooth LE Audio 被设计的相当灵活，意味着，有时候有着不同的工

作方式。举个例子，如果我们查看手机到耳机的简单连接，我们需要在每个方向上都有一个音频流。可以通过在每个方向上建立独立的 CISes 来实现。

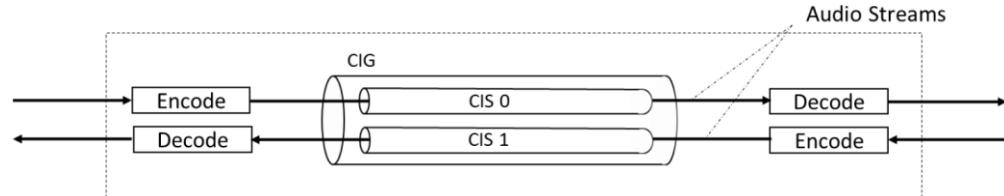


图 3.4 一个 CIG 中的两个独立 CIS

图 3.4 中我们可以看见两个 CIS，一个输出(CIS 0)，一个输入(CIS 1)，两个都包含在同一个 CIG 中。我们可以通过使用单独的 CIS 来优化它，如图 3.5 所示，它展示了单独的双向 CIS，在同一个 CIG 中，承载双向的音频数据。我们将在下一章研究其具体的工作原理。

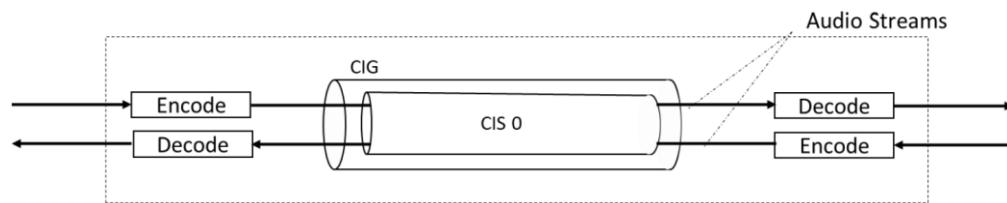


图 3.5 一个 CIG 中的双向 CIS

以上两种选项都是允许的，取决于应用程序决定哪一种最适合它来使用。在大多数情况下，这种双向性优化应该是备选项，因为它节省了通讯时间。

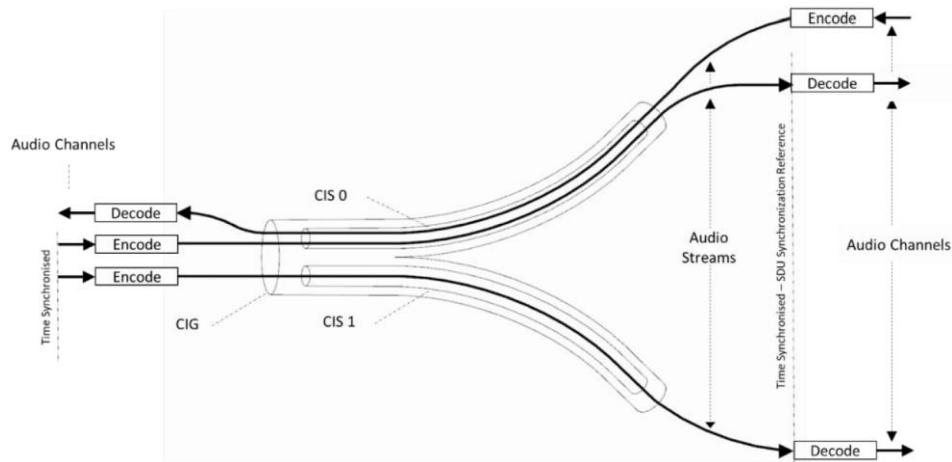


图 3.6 用于两个 Acceptor 的单向和双向 CIS，比如一部支持使用两个助听器进行电话呼叫的手机。

图 3.6 展示一个典型的应用，CIG 里面含有两个 CIS，可能是将手机连

---

接到一对耳塞，但是返回音频的麦克风只在一个耳塞上面实现。图中展示的 CIS 0，是个双向的 CIS，将来自手机通话音频向外传到耳塞，并将耳塞的麦克风音频传回手机。CIS 1 将手机的音频流传到另外一支耳塞。音频流是立体的还是单声道的，取决于应用程序。如果是传的单声道音频，相同的单声道音频数据会通过 CIS0 和 CIS1 独立地发送到两个耳塞。

### 3.4 Context Types

为了决定要连接到哪些音频流，设备需要更多地了解这些流包含什么或它们的用途。就引入了 Context Types，用来描述当前用例或者与用例相关的音频流。它们定义在 Bluetooth Assigned Numbers 中的 Generic Audio。Initiator 和 Acceptor 都可使用 Context Types 来指示它们想要参与的活动或连接类型，并适用于单播和广播。

使用 Bluetooth Classic Audio profiles，Central 和 Peripheral 设备之间的对话基本上是“我想建立音频连接”，没有更多它是什么的信息。由于 HFP 和 A2DP 本质上是单用途的 profile，这不是问题，但是在 Bluetooth LE Audio 中，音频流可用于铃声、语音识别、音乐播放、提供导航指令，或许许多其他应用程序。这些就是 Context Types 的用武之地了。

正如我们稍后将要看到的，Context Type 在单播流配置过程中用作可选元数据。Acceptor 可以在任何时间点，公开它准备接受的 Context Types。比如，如果助听器佩戴者正在进行私人(non-Bluetooth)对话，不想被打扰，他们可以将助听器通过使用“Ringtone” Context Types，将其对于相关的音频流，设置成 unavailable。这意味着助听器将默默地拒绝来电。这比将手机置于静音更为普遍，因为通过使用 Context Types，他们将拒绝来自任何已连接过的手机来电，或者来自任何其他已连接过的 VoIP 来电。他们也可以在通话时设置成 Ringtone Context Type，以防止任何其他来电打断当前通话。这可以在每个设备上完成，意味着你可以将来电呼叫限制到某个特定手机，为 Acceptor 控制哪个设备可以请求音频流，提供了一种强大的方法。

Initiator 在尝试建立音频流时，就会使用 Context Type，通知 Acceptor 相关的用例。如果 Acceptor 上的 Context Type 设置成了 unavailable，配置和音频流的建立过程就会终止。这发生在 Isochronous Stream 建立之前的 ACL 链路上，意味着这些决定可以与现有的音频流并行发生，而不会干扰到它，不论该请求来自于当前提供音频流的设备，或是另外一个想要建立或替换现有音频流的 Initiator。现有的音频流是单播还是广播并不重要，意味

---

着使用广播音频流收听 TV 的助听器用户，可以使用 Context Types 来防止当前的音频流会被任何手机来电打扰。

在当前的 Generic Audio Assigned Numbers 文档中，定义了 12 个 Context Types，它们在 Context Types 的两字节的位域中表示出来，如图 3.1 所示，每一个 bit 代表一种 Context Type。允许同时使用多个值。

Bit	Context Type	描述
0	Unspecified	设备上的另一个 Context Type 没有明确支持的任意类型的音频用例
1	Conversational	人与人之间的对话，通常是语音电话，可以是任何形式，比如，固定电话、蜂窝电话、VoIP、PTT，等等。
2	Media	音频内容。通常，这是一种方式，比如广播、TV 或音乐播放。与 A2DP 处理的内容类型相同。
3	Game	与游戏相关的音频，可能是一种音效、音乐以及对话的混合，通常具有低延迟要求。
4	Instructional	指导信息，比如卫星导航、公告或用户指南，通常比其他用例具有更高优先级。
5	Voice assistants	人机通信和语音识别，不包括指导信息。意味着这是以语言的形式。
6	Live	实况音频，其中 Bluetooth 音频流和环境音都可能同时被感知，意味着有延迟的限制
7	Sound effects	诸如键盘点击、触摸反馈以及其他特定应用程序的声音。
8	Notifications	寻求注意力的声音，比如宣布消息到达。
9	Ringtone	以 in-band 音频流的形式通知来电。Ringtone Context Type 不适用于 out of band 铃音，其使用 CCP 和 TBS 发送信号。
10	Alerts	机器生成的事件通知，可能包括电池严重报警、门铃、秒表以及始终警报，也可能是厨房电器或白色家电发出的周期结束警报。
11	Emergency alarm	高优先级的警报，比如烟雾或火灾报警。
12-15	RFU	保留使用

表 3.1 当前定义的 Context Types

其中大多数是显而易见的，但有两种 Context Types 值得特别注意：“Ringtone” 和 “Unspecified”。

“Ringtone” Context Types 用于通知来电，但仅限于需要建立音频流的 in-band 铃音。如果用户已经在接收来自另一个不同设备的音频流，而不是来电的那个手机，这里可能会有问题。为了在不停止当前音频流的情况下，向用户发送来电信号，需要至少一个耳塞建立来自第二个 Initiator 的独立单播流。实际上，许多 Acceptor 不太可能拥有资源，同时支持来自不同的 Initiator 的并发音频流。Acceptor 可以停止当前音频流以切换到 in-band 铃声，但如果他们拒绝来电，他们需要恢复起初的音频流。大多数情况下，提供 out of band 铃声可能会带来更好的用户体验，该铃声通过使用 CCP 和 TBS 的耳塞产生。用户可以听到混入现有音频流的铃音，并决定是否接受或拒绝来电。如果他们拒绝来电，他们可以继续收听起初的音频流。大多数情况下，最好使用 “Ringtone” 来管理设备是否允许通过铃音来打断当前音频应用程序。我们将在第九章介绍如何处理 out of band 铃声。

“Unspecified” Context Type 是笼统的一类。每个 Acceptor 都必须支持 “Unspecified” Context Type，但是不需要使其可用。当 Acceptor 确实将 “Unspecified” Context Type 设置为可用时，表示它将接受除其明确表示不

---

支持的 Context Type 之外的任何 Context Type。随着实现者习惯于 Context Type，一些人会使用它来支持 Central 设备(通常是手机)，按照与 A2DP 和 HFP 几乎相同的方式管理音频用例。将“Unspecified”设置为 available 的 Acceptor，实际上允许手机完全控制发送的内容。我们将在第七章更详细地讨论 Supported 和 Available 的概念。

所有的 Context Type 都是相互独立的。其中任何一个的使用并不意味着或要求另外一个也被支持，除非此需求是被顶层的 profile 强制的。例如，对于“Ringtone” Context Type 的支持通常并不意味着或要求支持“Conversational” Context Type，因为“Ringtone”本身可用于分机铃声，以提醒听力损失的人固定电话有来电。

Initiator 和 Acceptor 都可以使用 Context Types，以提供关于音频流的用例信息，从而允许它们决定是否接受音频流。为了实现这一点，Context Types 被用在了一系列 characteristics 和 LTV 元数据结构中。你将在一下位置找到它们以这种方式使用：

- Supported\_Audio\_Contexts characteristic [PACS 3.6]
- Available\_Audio\_Contexts characteristic [PACS 3.5] (also used in a Server announcement – [BAP 3.5.3])
- Preferred Audio Contexts LTV structure (metadata in a PAC record) (see also [BAP 4.3.3])
- Streaming Audio Contexts LTV structure (metadata used by an Initiator to label an audio stream) [BAP 5.6.3, 5.6.4, 3.7.2.2]

音频流可以与多个 Context Type 相关联，尽管其目的是 Context Type 的值表示当前用例。Streaming Audio Contexts 元数据拥有在用例改变时，允许设备更新 bit filed value 的程序。这通常用在已建立用于多个用例的音频流的情况。比如，Audio Source 混合来自不同的应用程序的音频，诸如可能中断音乐的卫星导航消息。在此情况下，假设其 QoS 参数合适，继续使用当前音频流是很有效率的，而且只是更新了当前的 Context Type 值。

### 3.5 Availability

Bluetooth LE Audio 支持比 Bluetooth Classic Audio 更多的可能性和组合。为了使设备能够对它们正在做的事情做出明智的选择，它们不仅需要告诉对方它们正在参与的用例(这是 Context Types 的原因)，而且还需要知道它们以后有兴趣参与的用例。这就是 Availability 的所在。

---

正如我们上面所看到的，Acceptor 使用 Context Type 来表示它们是否可以参与用例。通过两种方式完成。Acceptor 使用定义在 PACS 中的 Available\_Audio\_Contexts characteristic 来声明它支持的哪一个 Context Types 当前可用用于建立音频流。无论是 Initiator 还是 Acceptor，音频源都会在它们的 codec 配置的元数据中使用 Streaming\_Audio\_Contexts LTV 结构，以通知 Audio Sink 与此用例有关的音频流。

想要建立 unicast Audio Streams 的 Bluetooth LE Audio 设备，也可以在它们启动音频流之前，使用 Context Type 来表示其可用性，方法是在其广播 PDU 中包含 Streaming\_Audio\_Contexts LTV。类似的方法中，作为广播者，Initiator 将其包含在周期性广播的元数据段中，以便 Broadcast Sinks 和 Broadcast Assistants 可以在接收中，从它们不感兴趣的用例中过滤出它们想要的用例。

对于 Broadcasters，你需要注意，如果 Streaming\_Audio\_Contexts 域未出现在 Codec ID's 元数据中(我们将在第四章里讨论)，它将意味着仅有的 Supported\_Audio\_Contexts 的值是“Unspecified”，因此每个 Broadcast Sink 可以与之同步，除非它们特别将“Unspecified”设置为 non-available。

### 3.6 Audio Location

按照以前的 Bluetooth 音频规范，一个 Bluetooth LE Audio source 对应一个 Bluetooth LE Audio sink。音频流按照单声道或立体声发送，并且由 audio sink 解释它是怎样呈现的。Bluetooth LE Audio 旨在解决具有多个扬声器或耳塞的应用。它能够优化每个 Audio Sink 的播放时间，只需要向其发送所需的音频流。一般说来，对于一对耳塞，左耳塞只接收左音频流，右耳塞只接收右音频流。意味着，每个 Audio Sink 都可以最小化其接收器打开的时间，从而降低其功能。

为了实现这一点，设备需要知道它们打算接收什么空间信息，例如，来自立体声输入的左或右音频流。它们通过指定 Audio Location 来实现。这些定义在 Bluetooth Generic Audio Assigned Numbers，并且遵循用于扬声器放置的 CTA-861-G's Table 34 codes。这些值在四字节宽度的位域中，按 bit 来表示。最通用的 Audio Location 如表 3.1 所示。

---

Audio Location	Value (bitmap)
Front Left	0x00000001 (bit 1)
Front Right	0x00000002 (bit 2)
Front Centre	0x00000004 (bit 3)
Low Frequency Effects 1 (Front Woofer)	0x00000008 (bit 4)
Back Left	0x00000010 (bit 5)
Back Right	0x00000020 (bit 6)
Prohibited	0x00000000

表 3.2 通用 Audio Location 值

每个可以接收音频流的 Acceptor 必须设置至少一个 Audio Location——全部位域留空白是不允许的。Audio Location 通常是在制造时设置的，但在某些情况下，它可以由用户更改——例如，扬声器可以由应用程序或物理开关将其设置为 Front Left 或 Front Right。

注意，单声道不是 location，因为单声道是音频流的属性，不是物理呈现设备。单通道扬声器通常会设置 Front Left 和 Front Right audio location。Initiator 会确定 Acceptor 支持的音频流数量，以及可用的扬声器的数量，并决定是否发送 downmixed stereo 音频流(即单声道)，或发送立体声音频流，假设扬声器可以 downmix 到单声道的话。Broadcasters 将单声道音频流标记为 Front Left 和 Front Right。我们将在第五章进一步了解 Audio Location 的使用方式，但接下来，我们到了 Channel Allocation。

### 3.7 Channel Allocation (multiplexing)

你总是知道 Bluetooth Classic Audio profiles 中传输的内容。HFP 传送单声道音频流，A2DP 用 SBC codec 4 通道模式来传输单声道、双声道、立体声或联合立体声编码音频流。Bluetooth LE Audio 更加灵活，支持 CIS 或 BIS 包含一个或多个通道复合进单个数据包，仅受带宽限制。

使用此方法的原因是 LC3 是单通道 codec，是所有 Bluetooth LE Audio 实现的强制 codec。意味着它将每个音频通道分别编码为固定长度的离散帧。使用 SBC 的联合立体声编码很受欢迎，它将左和右两个音频通道合并为单个编码音频流，这就比分开的左和右通道效率更高。这是由于它能够编码两个输入音频通道之间的差异。LC3 的更高效率的设计意味着，与分别编码每个通道然后串联各个编码帧相比，联合立体声编码几乎没有优势。这种方法允许两个以上的通道被编码并分组在一起。

然而，这意味着需要引入新的机制来将多通道的多个编码帧打包在一起，

这是通过 Channel Allocation [BAP Section 4.2] 来执行的。

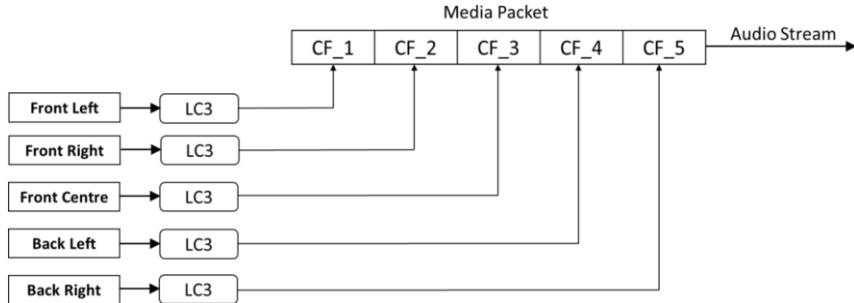


图 3.7 多 Bluetooth LE Audio 通道复合实例

图 3.7 展示了五通道环绕音系统的工作原理。使用 LC3 分别对五个音频输入通道进行编码，然后将编码的帧排列进多媒体数据包，多媒体数据包作为单个 isochronous PDU 传输。LC3 codec 帧始终按照与每个音频通道相关的 Published Audio Capability Audio Location 升序排列，使用 Table 3.2 中总结的 Assigned Numbers。因此，在这种情况下，它们将按照：

Front Left (0x00000000001),  
 Front Right (0x00000000002),  
 Front Centre (0x00000000004),  
 Back Left (0x00000000010),  
 Back Right (0x00000000020) 的顺序排列。

Media Packet 仅包含编码的音频帧。它可用扩展为包含多个编码音频帧 blocks，每个 blocks 包含每个 Audio Location 的一个帧，如图 3.8 所示 CF\_N1 指来自每个输入音频通道的第一个样本，CF\_N2 指这些音频通道的下一个采样。

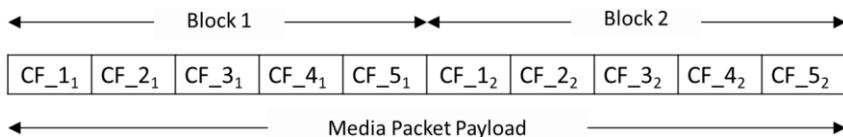


图 3.8 包含五个音频通道的两个 Media Packet

使用 blocks 看上去有效率，但是也有一些重要事项。它会产生更大的数据包，更容易受到干扰。还增加了延迟，因为 Controller 要在开始发送之前等待多个帧。如果将多个块添加到 Burst Number 或 Pre-Transmission Offset 的多 Isochronous Channel 特性中(我们将在下一章中讨论)，它很快会导致长达数百 ms 的延迟。某些情况下，这可能是有用的，但不在我们今天使用的一般音频应用中。

---

多媒体数据包中没有与之相关的 header 信息。相反，可以支持的音频通道数量在 Audio\_Channel\_Location LTV 中指定，其包含在 PAC 记录中的 Codec Specific Capabilities LTV 里，来自于 Initiator 在音频流配置过程中对每一个 Isochronous Stream 设置的值。同时，使用的 blocks 的个数在 Codec\_Frame\_Blocks\_Per\_SDU LTV 结构中配置。

第五章中，在我们讨论 LC3 和 Quality of Service 时，会更详细地讨论这些。

### 3.8 ~~Call~~ Content Control ID – CCID

在 Bluetooth LE Audio 中将控制和数据平面分离的结果是，在诸如手机控制或媒体控制的内容控制与音频流之间，不再有任何直接的关系。这增加了 Bluetooth LE Audio 的灵活性，但导致了更多的复杂性。Initiator 可能有多个并发的应用程序，用户可能希望将控制和这些应用程序中的特定一个相关联。考虑到这种用例，两个单独的电话呼叫，比如蜂窝电话和同时发生的团队会议，此时用户可能想要暂停一个或终止一个，同时保留另外一个。为了应对这种情况，就引入了 Content Control ID，它将 Content Control 实例与特定的单播或广播关联起来。

CCID 可用于广播的说法似乎矛盾，当时强调了一个事实，即尽管广播流不需要 ACL 连接，但有许多应用程序可能存在 ACL 连接。例如，如果你将使用单播收听手机上的音乐转变为广播，以便与朋友分享，你仍然希望能够控制媒体播放器。这种情况下，你将保持 ACL 连接处于活动状态，并且将媒体控制与广播音频流关联在一起。

Content Control ID characteristic 定义在 GATT Specification Supplement 的第 3.45 节，具有一个唯一值用来标识一个服务实例，该服务控制或者提供音频相关特性上状态信息。它是单个字节整数，为在设备上的所有 Content Control services 实例提供唯一标识。

当 Audio Stream 包含由 content control service 控制的内容时，它将 CCID 包括在 Audio Stream 的元数据中的此类服务中列表中，以此告诉 Acceptor 和 Commander 它们可以在哪里找到正确的服务。我们稍后将在 3.12 节中介绍 Commander。

CCID 目前仅用于 Telephone Bearer Service 和 Media Control Service。它们不适用于呈现或者捕获控制。

---

### 3.9 Coordinated Sets

尽管我们只用了几年时间，我们已经非常熟悉 TWS 耳塞的概念，以至于绝大多数人忘记了 Bluetooth Classic Audio 并没有涵盖 TWS 耳塞的工作方式。如第一章所述，它们依赖芯片公司专有技术的扩展。Isochronous Channels 的设计纠正了这一点，允许 Initiator 发送独立的音频流给多个 Acceptor，以及一个公共的参考点，在此参考点处，所有的 Acceptor 都知道它们已经收到了音频数据并可以开始解码了。然而，Bluetooth LE Audio 的灵活性，包括来自广播的新用例，提出了一种将 Acceptor 作为一组设备链接在一起的方法的需求，这些链接在一起的设备可以作为单个实体看待。

Coordinated Set 的概念解决了这一需求。它允许设备公开它们是一组设备的一部分，这组设备支持共用的用例并且应该作为一个实体进行操作。虽然大部分人会立刻想到一对耳塞或助听器，但这一套同样可以是一对扬声器或一套环绕声扬声器。一个耳塞 Coordinated Set 应该被表示为一个设备，这样，发生在一个设备上的任何事情都会在另外一个设备上发生，尽管怎样发生却取决于实现。当你调整一对耳塞的音量时，两个耳塞的音量应该同时变化，如果你决定收听不同的 Audio Source 时，左右耳塞都应该同时变化。你不希望体验你的左耳塞正在收听 TV，而右耳塞正在播放来自手机的音乐。

Coordination 由 Coordinated Set Identification Profile 和 Service(CSIP 和 CSIS)处理，引用自 CAP。CSIS 和 CSIP 的主要特点是，除了将设备识别为 Coordinated Set 中的成员之外，还提供了 Lock 的功能。这确保了当 Initiator 与其中一个成员进行交互时，其他成员可以被锁定，防止任何其他 Initiator 与该集合的其他成员交互。

对这种 Lock 的需求是 ear-worn 设备，如助听器和耳塞，使用 Bluetooth 技术相互之间直接通信存在问题。因为人类的头部很善于衰减 2.4GHz 信号。如果耳塞很小，意味着天线也很小，那么一只耳朵中的设备传输的不太可能被另外一只耳朵中他的伙伴接收到。当前，许多耳塞通过在其中使用用于 ear-to-ear 通讯的不同的、更低频率的 radio 来绕过此限制，此 radio 不会被头部衰减，通常使用 Near Field Magnetic Induction (NFMI)。第二种的这个 radio 增加了成本并占用了空间，但是移除它并依靠耳塞之间的 2.4GHz Bluetooth 链路会增加不同 Initiator 可能对左和右耳塞发送相互冲突的命令的风险。

使用 CSIP 和 CSIS，如果你在左耳塞上接听电话，Initiator 会对右耳塞

---

设置 Lock，并在释放 Lock 前将右耳塞转到相同的音频流。Lock 功能允许在不需要 Coordinated Set 的成员之间彼此通话的情况下，管理这些交互。

如果 Coordinated Set 的成员确实具有另一种 radio 可以穿过头部，Hearing Access Service 具有另一个特征，它会指示此种 radio 可以被用于向另一个助听器传递信息。目前，此特性仅限于有关预设的设置信息，但可能将来会用于其他功能。

通常，Coordinated Set 成员在制造时配置并成对装运，但成员关系可以设置成写入和读取，允许后续的配置，或更换故障或丢失的设备。

### 3.10 Presentation Delay and serialisation of audio data

支持两个耳塞将给为我们带来 Bluetooth LE Audio 必须解决的另一个问题，即确保左耳和右耳的声音完全同时呈现。在过去，音频数据发送到单个设备，其知道怎样提取左和右边信号并且同时呈现它们。对于 Bluetooth LE Audio，CIS 使用不同的传输时隙，将数据串行发送到不同的目的地。尽管输入音频通道在同一时间点将数据交给 Initiator，但编码数据包都是一个接一个到达 Acceptor。它们可能因重传进一步延迟。图 3.9 通过在 3.6 的 CIG 图示上加上到达两个 Acceptor 的左右数据包实例展示了这一点。

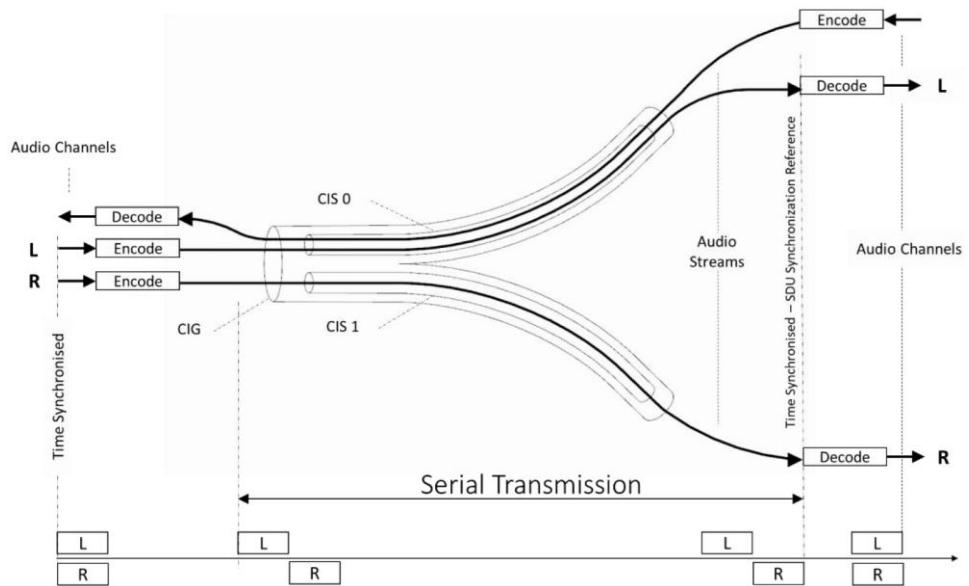


图 3.9 音频数据串行传输

这种串行化会导致一个问题。人类大脑非常善于察觉到达左右耳朵的声音的时间差异。并且利用这个差异估计声音来源的方向。

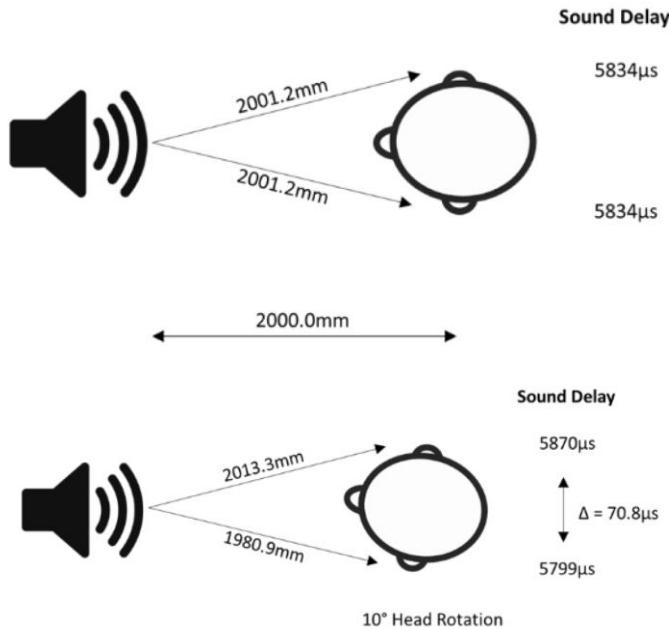


图 3.10 头部旋转对声音到达的影响

图 3.10 展示了如果你离音频源有 2 米远，并且将头旋转 10 度，则相当于声音到达时间差仅仅超过 70us。如果左右耳之间的呈现时间有变化，你的大脑会将其理解为音源在移动。如果差异超过 25us，并且在有规律地变化，你会开始感觉到不愉快的效果，就像是声音在你的脑袋里面移动一样。为了避免这种情况，需要一种确保左右耳塞始终同时呈现各自音频数据的同步技术，这点很重要。

我们不能依赖两个耳塞之间的任何 Bluetooth 通讯。如我们前面所讲，人类头部在衰减 2.4GHz 信号方面是非常有效的，因为它含有大量的水。如果你的耳塞很小，能够合适地插入耳道，那么就不能保证它们能够相互通信。

Bluetooth LE Audio 解决方案分为两个部分。首先，两个耳塞都需要知道一个共同的同步点，即每个 Acceptor 都可以保证每个其他的 Acceptor 有机会接收传输数据包的时间点，不论是单播还是广播。这个时间点必须由 Initiator 提供，因为它是唯一知道它将尝试多少次向所有 Acceptor 发送数据包的设备。(记住，Acceptor 之间通常不能交流，可能都不知道对方的存在)。

大多数情况下，Acceptor 将早于公共同步点接收到它们的音频数据包，因为音频应用数据包会被安排多次重传，以使接收到的机会最大化。这是由于丢失数据包导致的音频流中的 drop-outs 问题。对于听众来说，这是特别恼人的问题，因此使用重传来帮助改进信号的鲁棒性。因此，需要每个

---

Acceptor 都有足够的缓冲来存储从可能的最早到达时间(即数据包首次传输的时间)到公共同步点的数据包。我们将在第四章关于 Synchronisation Point 中了解到更多。

然而，你不能在 Synchronisation Point 呈现音频，因为它仍然是编码过状态。在 Synchronisation Point 和最终呈现点之间，需要对数据进行解码，并且在最终呈现之前，还需要执行一些附加处理，比如数据包丢失隐藏——Packet Loss Concealment(PLC)，有源噪声消除——active noise cancellation (ANC)或助听器音频调整。

不同 Acceptor 所需的时间可能是不同的。虽然你希望由一个制造商提供的一副助听器、扬声器或耳塞被设计成每个具有相同的处理时间，但如果 Acceptor 来自不同的制造商，或一个设备进行了固件升级命令一个没有，这个时间都是可能不同的。以此，Bluetooth LE Audio 规范包含了一个 Presentation Delay 的概念。由 Initiator 定义的值，以 us 为单位，指定了在每个 Acceptor 中，在音频将要被呈现 Synchronisation Point 之后的时间。在 3.11 中展示。SDU Synchronisation Point 的后缀“C>P”指的是 Central 到 Peripheral 方向(Initiator 到 Acceptor)。LL 指的是控制器中的链路层，在这里接收空中发送的数据。

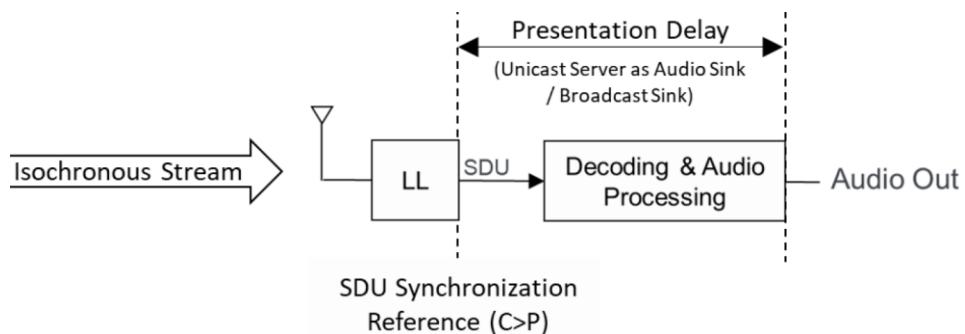


图 3.110 Acceptor 呈现的 Presentation Delay

用于呈现的 Presentation Delay 还可以包含 Host 应用程序相关要素，以故意增加时间直到呈现出来。这是关联到视频的音频流的常用技术，可以用于延迟呈现点以补偿 lip-synch 问题。对于公共广播应用，其中可能有多个广播发射器覆盖大礼堂或体育场，Presentation Delay 也可用于设置特定的延迟，以补偿服务于每个观众群的广播发射器与音频源，之间的不同距离。声音每秒传播 340m，以此声音在大型体育场中传播可能需要半秒钟。因此，场馆通常会对扬声器施加延迟以帮助同步场馆不同区域的声音。使用多个 Bluetooth LE Audio 广播发射器的 Presentation Delay 可以达到相同的效果，

---

使观众更加接近声音的源头。

每个 Acceptor 都包含它支持的最小和最大的值 Presentation Delay。最小值表示其在呈现声音之前，解码收到的 codec 数据包并处理任何音频的最短时间。最大值反映了它可以添加到其中的最长缓冲量。这些值由 Initiator 在配置过程中读取，在它传输每个音频流时，必须遵循所有的 Acceptor 的最大最小 Presentation Delay 值。意味着它不能设置大于任何 Acceptor 的最低 Presentation\_Delay\_Max，或低于任何 Acceptor 的 Presentation\_Delay\_Min 最高值。Acceptor 还可以公开它们对于 Presentation Delay 的首选值，Initiator 应尝试使用该值。除非 Initiator 上面的应用程序设置了一个特定的值，因为它可能用于实况应用程序或对于 lip-synch 的补偿。Presentation Delay 不期望暴露给接受音频的设备用户，总是由 Initiator 上的应用程序来设置。

Presentation Delay 旨在为多个 Acceptor 提供一个通用呈现点。它支持 Acceptor 并不知道彼此存在的情形，比如一只耳朵听了受损的用户，他会只佩戴一个助听器和单个耳塞。相同的 Presentation Delay 会用在两个设备上，两个设备都会同时呈现。在大多数应用程序中，Presentation Delay 的值应该尽可能小。支持更高的 Presentation Delay 值增加了 Acceptor 的资源负担，因为它需要更长时间去缓冲解码的音频流。

对于广播，Initiator 和 Acceptor 之间是没有连接的，Initiator 仅根据其应用程序判断所有潜在的 Broadcast Sinks 都可接受的 Presentation Delay 值。一般来说，应该避免超过 40ms(每个 Acceptor 都必须支持的)，因为如果在环境音也存在的情况下，它们可能引入回声，除非它们专门用在非常大的场馆中以适应这种情况。规范没有定义如果 Presentation Delay 超出其支持的范围，Broadcast Sink 因该做什么。

对于单播和广播，顶层的 profile 可能会指定一个特别的 Presentation Delay 值，特别是它们支持低延迟应用时。例如，当音频流中还存在环境音时，则坑你需要使用“Live” Context Type，以及设置不超过 20ms 的 Presentation Delay。

Presentation Delay 也用于音频捕获，即音频数据从 Acceptor 到 Initiator 的过程(规范中表示为 P>C，或 Peripheral 到 Central)。在这里，它代表了从音频被捕获到，然后进行后续的处理、采样以及编码，到第一个 Isochronous Stream 数据包可以被传输的参考点，如图 3.12 所示。

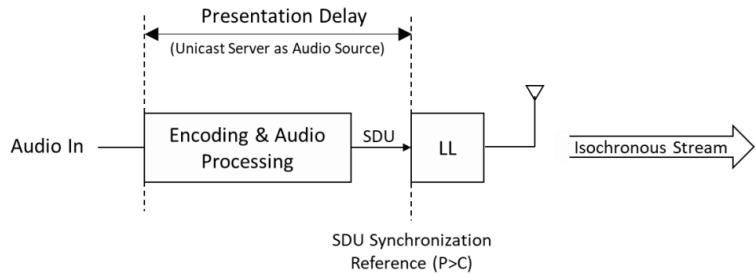


图 3.12 Presentation Delay 用作音频捕获

在音频捕获中使用 Presentation Delay 能够确保每一个麦克风或另外的音频转换器在相同的时间点捕获声音。它顶一个时间间隔，在该间隔期间，每个 Acceptor 都可以准备其音频数据包发送，第一次传输发生在 Presentation Delay 结束时。所有的其他音频源随后将在其分配的传输时隙中发送其编码数据包。当电话想要组合左右耳塞的麦克风数据时，这是符合期望的，因为它可以使用通用捕获时间的信息来组合上述数据。而不是基于捕获时间简单的混合，Initiator 通常会在允许噪声消除算法之前，使用该信息通知数字拼接技术，以对其多个音频流，但是这超出了 Bluetooth 规范。

虽然在 unicast audio sources 捕获中，Presentation Delay 最常见的使用是在每个耳塞或助听器中使用一个麦克风，但它同样适用于具有多个麦克风的单个设备。包括立体声麦克风和立体声耳机，每个盒子都有一个麦克风。在这些设备中，一种实现方式是使用用于多路复用的通道分配，将两个麦克风的信号编码进单个编解码帧(见下文第 3.7 节)，或使用 Presentation Delay 独立地传输它们，以确保捕获是对其的。两种情况下，都需要设置 Presentation Delay 的值，以覆盖音频处理和编码时间。

在麦克风间隔更远的情况下，接收到的数据将不会反映其位置之间的音频路径差异。如果 Initiator 想要恢复该信息，则可能需要调整输入音频流。

理解 Presentation Delay 仅仅应用在 Acceptor 是很重要的，不管 Acceptor 是 Audio Source 或是 Audio Sink。在许多用例中，Acceptor 同时充当两者。通常，每个方向的 Presentation Delay 值是不同的，以应对编码音频比解码音频用时更长的这个事实。在第 4 章中，我们将更详细地研究 Presentation Delay 怎样影响延迟和鲁棒性。

### 3.11 Announcements

作为给与 Acceptor 更多自主权理念的一部分，Bluetooth LE Audio 规范允许它们通过使用 Service Data AD Type，发送 Announcements，通知

---

Initiator 它们可以接受或者传输音频数据了。在 Acceptor 可以连接并且请求连接处，它可以决定是使用 Targeted Announcement，还是 General Announcement(简单说它是可用的，但是不去发起特定的连接)。Announcement 的 AD Type 中使用的 LTV 结构如下表 3.3 所示。

Field	Size (Octets)	Description
Length	1	Length of Type and Value fields
Type	1	Service Data UUID (16 bit)
Value	ASCS UUID <sup>22</sup>	2 0x0184E
	Announcement Type	1 0x00 = General Announcement 0x01 = Targeted Announcement
	Available_Audio_Contexts	4 Available_Audio_Contexts characteristic (from ASCS)
	Metadata Length	1 ≥ 1 if there is additional metadata, otherwise 0
	Metadata	varies Metadata in LTV format

表 3.3 用于 Targeted 和 General Announcements 的 AD 值

Common Audio Profile (CAP) 描述了 Initiator 或 Commander(见 3.12 节)如何通过定义两种特定模式对 Announcements 做出反应，这有助于解释合适使用每种模式。它们是：

- INAP - the Immediate Need for Audio related Peripheral mode (INAP),
- RAP – the Ready for Audio related Peripheral mode (RAP)

INAP 是指 Initiator 或 Commander 想要建立连接的情况，通常是由于用户的操作，需要确定哪些 Acceptor 是可以连接的。如果它发现可用的 Acceptor，它会去连接，或者，如果它发现了多个 Acceptor 话，就会向用户提供一系列可用的 Acceptor，让用户做出选择。在 INAP 模式下操作时，Initiator 或 Commander 通常会以更高的速率开始扫描以找到 Acceptor。

相反，但处于 RAP 模式，Initiator 以较低的速率扫描，但将通过连接来响应来自 Acceptor 的 Targeted Announcement。

当 Acceptor 需要立即连接时，它们应该仅在有限的时间内使用 Targeted Announcement。

Announcement 中还有一个微妙之处，即它们可能包含或不包含 Context Type 值。如果它们用作与音频流有关系的地方，它们应该包含 Context Type，并且叫做 BAP Announcements [BAP 3.5.3]，其中包含了一个 Available Audio Context 域。如果它们用于控制目的或 Scan Delegation，它们就不会包含 Context Type。

Broadcasters 在它们的 Extended Advertisements 中使用 Announcement，

---

以此通知 Broadcast receiver 和 Broadcast Assistant(参见第 3.12 节)它们有广播音频流可用。这有两种形式:

### 3.11.1 Broadcast Audio Announcements

Broadcast Audio Announcement 通知任何扫描设备 Broadcaster 正在传输一组一个或多个音频流。用于 Broadcast Audio Announcement 的 LTV 结构显示如下 3.4 表。

Field	Size (Octets)	Description
Length	1	Length of Type and Value fields
Type	1	Service Data UUID (16 bit)
Value	Broadcast Audio Announcement Service UUID	2
	Broadcast_ID	3
	Supplementary Announcement Service UUIDs	varies

表 3.4 用于 Targeted 和 General Announcement 的 AD 值

### 3.11.2 Basic Audio Announcement

Broadcast Source 在 Periodic Advertising 序列中使用的 Basic Audio Announcement 具有类似易混淆的标题 Basic Audio Announcement，以通过 Broadcast Audio Stream Endpoint 结构(BASE)暴露广播音频流参数。第 8 章中描述了其用途。

## 3.12 Remote controls (Commanders)

Bluetooth 一直是远程控制设备的一个好的选择，但这些都是简单的遥控器，基本上是传统红外遥控器的 Bluetooth 替代品。助听器和耳塞使远程控制成为一个很有吸引力的选项，因为这些设备太小，没有足够的空间容纳许多按钮，并且即使可用容纳，操作一个你看不见的(因为它在你的头部侧边或在你耳朵后面)按钮，也是一种糟糕的用户体验。由于大多数耳塞与手机、笔记本电脑或 PC 一起使用，产生音频的应用程序通常包含了你用来暂停、接听电话或改变音量的这些控制，因此这不是一个问题。然而，当助听器只是用来放大环境声音时，助听器用户也需要控制他们的助听器音量。

为了使其比在助听器上摸索按键更容易，该行业已经开发出了简单的、类似遥控钥匙的遥控器，允许用户轻松调整音量或改变音频处理算法以适应他们的环境(这些被称为预设设置)。即使助听器包含 Bluetooth 技术，大多

---

数情况下，用户不会使能 Bluetooth 链接，并且将手机从口袋或包中取出，接着找到合适的 app，这是一种不方便的音量调节方式。如果你收到噪音的困扰，可用更快更容易地将助听器取出，这不是一个好的用户体验。

这只是问题的开始。Bluetooth LE Audio 的新广播功能带来了更多的公共基础设施，用户需要在多个不同的广播中导航并决定接收哪一个。在助听器和耳塞中都很难做到这一点，还需要相对耗电的扫描。为了解决这些问题，开发了独立设备的概念，它可用提供音量控制、发现并显示广播源，发现加密广播的密钥，允许助听器用于更改他们的预设置。它还可以用来接听电话、或控制媒体播放器。这些设备采用 CAP 中定义的 Commander 角色，但是也可以使用 BAP 中的 Broadcast Assistant 角色来发现广播，以及作为 Content Control Client。大多数顶层的 profile 都为这些设备可用承担的角色引入了更多名称。文档的其余部分，我将使用 Commander 术语，除非它用于 Broadcast Assistant 特定子集。

Commander 是 Bluetooth 拓扑的重要新增角色。可以在手机上实现，作为一个单独的应用程序，或者电话通讯或音频流 app 的与部分。它也可以在具有与 Acceptor 或 Accepter 的 Coordinated Sets 相连接的任何设备里实现。意味着你可以在专用遥控器、智能手表、腕带甚至助听器和耳塞中实现。Commander 可以有一个显示器来展示有关广播的文本信息，以帮助你选择它们，或者只有音量控制按键。Commander 按照先到先得的方式工作，所以你可以有多个 Commander，当你想改变音量或使助听器静音时，你可以使用不论哪个，只要是先到的。由于所有的功能都是明确指定的，意味着多个设备可以互操作地实现它们。在第 12 章，我们将了解它们可能改变我们使用 Bluetooth 音频设备的方式。

介绍完这些新的概念之后，我们现在可以深入研究这些规范，看看它们如何工作，以及它们如何支持 Bluetooth LE Audio 的新用例。

---

## 第4章 Isochronous Streams

如果你过去曾使用过 Bluetooth 应用程序，那么你可能会将精力集中在 profile 上，而很少关注 Core 规范。这可能是因为 Bluetooth classic Audio profiles 使用了与 Core 设定绑定好的 well-defined 传输配置，所以不需要太多了解 profile 或它的相关协议下面发生了什么。使用 Bluetooth LE Audio profile，情况发生了变化，因为与使用 Bluetooth Classic Audio profile 相比，你更有可能影响 Core 的工作方式。

为了尝试并使最灵活的系统成为可能，不仅可以满足今天的音频需求，还可以满足我们尚未考虑到的需求，规范必须允许更多的灵活性。为了实现这一点，我们做出了一个基本的框架决策，以此将音频平面和控制平面切割。这意味着已经定义好 isochronous 物理信道，用来传输音频流。它们在旁边，但与现有的 Bluetooth LE ACL 链路是分开的。Core 中的 Isochronous 物理信道使你可以构建多个 Isochronous Streams，这些音频流可以传输所有类型的音频，从非常低的语音质量到难以置信的高质量音乐。有个例外，稍后我们将看到，Isochronous Streams 是不包含控制信息——它们纯粹用于传输音频。伴随的 ACL 通道用于建立 Isochronous Streams，打开、关闭、增加音量和媒体控制，以及我们需要的所以其他功能，使用标准的 GATT 程序，它是 Bluetooth Low Energy 的一部分。

为了向不同的延迟、不同的音频质量以及不同的鲁棒性级别提供灵活性，开发人员需要控制 Isochronous Streams 的配置方式。那是在 Generic Audio Framework 的 profile stack 的相当高层级里完成的。意味着，当你开始在 Bluetooth LE Audio 应用程序上工作时，即使你仅仅使用顶层的 profile，你仍然需要对底层 Isochronous Streams 的工作方式具有相当的了解。这与你在以前的大多数 Bluetooth 应用程序中所经历的情况不同。为了帮助理解 Bluetooth LE Audio 的总体架构，我们需要了解 Isochronous Streams 是如何开发的，它们做什么，以及如何使用它们。

### 4.1 Bluetooth LE Audio topologies

到目前为止，Bluetooth 规范主要涉及 peer-to-peer 连接：Central 设备连接到 Peripheral 设备，并且交换数据。这是一个非常受限的拓扑结构。正如我们在 True Wireless Stereo 耳塞中看到的，许多公司都开发了专有的扩展以

---

增加灵活性，但 Isochronous Streams 的开发，使用了比这些专有解决方案更为宽泛的拓扑结构以解决此问题。除了将手机连接到一对耳塞或扬声器之外，Bluetooth LE Audio 还需要能够发送独立的左和右信号到左耳塞和右耳塞。还需要能够将相同的信息发送给多个耳塞，并扩大设备和音频流数量。

Isochronous Streams 流有两种类型——单播和广播。单播连接，称为 Connected Isochronous Streams (CIS)，最接近现有的 Bluetooth 音频用例。

“连接”在这个意义上意味着它们正在两个设备之间传输音频数据，使用两个设备之间确认方案提供流量控制。Connected Isochronous Streams 具有一个 ACL 控制通道，该通道在承载音频数据的 CIS 整个生命周期都在运行。

用于广播的 Broadcast Isochronous Streams (BIS) 结构非常相似，但是有个主要的区别。通过广播，传输 Isochronous Streams 的设备不知道可能有多少设备接收音频。设备之间没有连接，也不需要 ACL 链接。最简单的说，广播纯粹滥交。然而，可以向广播添加控制链接。在 Core 级别，Connected 和 Broadcast Isochronous Streams 之间有明确的区别，基于数据是否被确认。然而许多应用程序会在单播和广播之间来回切换以满足不同的用例，而用户却不知道发生了什么事情。但目前，我们将专注于基础知识。广播允许多个设备收听相同的内容，以 FM radio 或广播 TV 的相同方式。Bluetooth LE Audio 广播功能需求最初是由电传(telecoil)助听器应用驱动的，在公共场合，多个佩戴助听器的人能够收听同一个信号。电传线圈的音频质量相对较低，多用于语音。Bluetooth LE Audio 具有更高质量音频，并且安装成本显著降低，行业设想了更广泛的应用和使用。传统的电传(telecoil)地点，如会议中心、剧院和礼拜场所，每个人都可以通过耳机和耳塞获取公共信息，例如航班公告、火车出发时间和公交车时间，不仅仅是佩戴助听器的人。广播也适用于更多个人应用，在这些应用程序中，一群人可以收听相同的电视节目，或通过手机分享音乐。最后一个例子展示了 Bluetooth LE Audio 应用程序如何在不可见的情况下回切换底层协议。如果你正在将音乐从手机传输到耳塞，可能使用了 Connected Isochronous Stream。当你的朋友过来时，你问他们“你们也想听听这个吗？”，你的音乐共享应用程序会将你的手机从私有的，单播连接切换成加密广播连接，这样你们所以人都可以听见相同音乐了，不论是在耳塞、助听器、耳机还是扬声器上，在应用层，收听、并与任何数量的其他人分享音乐应该是无缝的一一用户不需要了解广播或单播。但是在这一转变过程中，会有很多事情发生。

这些不同用例的构建模块基本是相同的。用单播用例的 Connected

---

Isochronous Streams 和用于广播用例的 Broadcast Isochronous Streams 之间存在一些差异，但是基本原理非常相似。我们现在已经准备好好看看它们是如何组合在一起的，这意味着要跳入 Core 了。

## 4.2 Isochronous Streams and Roles

Core 5.2 版本中的 Isochronous Streams 特性是 Bluetooth Low Energy 中的一个新的概念。如果你了解 Hands-Free profile 或者 A2DP，你会知道它们的拓扑结构相当受限。通常在手机和耳机或 Hands-Free 设备之间，HFP 具有双向一对链路。它具有两个角色：Audio Gateway，以及 Hands-Free 设备。A2DP 甚至是更简单的单播链路，规定了产生音频的 Source 设备以及 Sink 设备，它可以是你的耳机，扬声器、放大器或记录设备，用于接收音频。

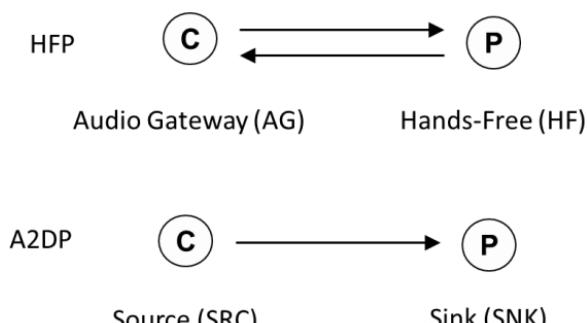


图 4.1Bluetooth Classic Audio 拓扑

Bluetooth LE Audio 基于存在于 Bluetooth LE 规范中的基本不对称，其中一个设备——Central 设备，负责建立和控制 Isochronous Streams。Central 可以连接多个 Peripheral 设备，这些设备使用 Isochronous Streams 发送和接收音频数据。不对称意味着 Peripheral 设备的功耗可能更低。对于 CIS，它们对于 Isochronous Streams 的配置具有发言权，这将影响音频质量、延迟和电池寿命，使它们能够比使用 Bluetooth Classic Audio profile 具有更多的控制权。对于 BIS，Central 做出所有决定，Peripheral 决定它想要接收哪个 Broadcast Isochronous Streams。

重复一下我们在第 3.3 节概述中所说的内容，当我们向 Bluetooth LE Audio 规范协议栈的上方移动，我们可以遇到设备扮演的角色有许多不同的名称。在 Core 中，它们被定义为 Central 和 Peripheral 设备。在 BAPS 规范集合中，它们被称为 Client 和 Server，以及在 CAP 中它们变成了 Initiator 和 Acceptor。Initiator 角色始终存在负责调度 Isochronous Streams 的 Central 设

---

备中。Acceptor 是参与这些音频流的设备。始终有一个 Initiator，但可能有多个 Acceptor。

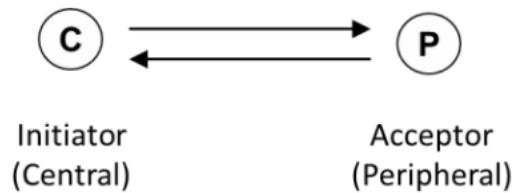


图 4.2 Bluetooth LE Audio 角色

当我们移到顶层 profile 时，会出现大量的新角色名称，包括 Sender、Broadcaster 和 Receiver。我将忽略掉所有这些并且大多数时间使用 Initiator 和 Acceptor(尽管它们在技术上是角色)，因为它们最能够解释 Bluetooth LE Audio 音频流的工作方式。当没有涉及音频流时(控制规范就是这种情况)，我将回到使用 Client 和 Server。

在这个阶段，我想指出的一点是，除了广播之外，任何一个设备都可以作为音频源，生成音频数据，或者作为音频 sink，接收数据。Initiator 和 Acceptor 可以同时是 source 和 sink，并且它们每个都可以包含多个 Bluetooth LE Audio sink 和 source。谁生成音频、谁接收并呈现音频的概念与 Initiator 与 Acceptor 的概念是正交。需要记住的重要一点是，Initiator 是负责制定每个音频数据发送时序的，该任务叫做调度。Acceptor 是接收这些音频流的。此概念适用于单播和广播。Acceptor 可以生成音频数据，比如从你的耳机麦克风捕获你的声音，但是 Initiator 负责告诉它什么时候它需要将数据发送回来。

作为 Initiator 角色，远比 Acceptor 角色复杂，Initiator 通常是手机、TV 以及平板电脑等具有更大电池和更多资源的设备。调度必须考虑 Initiator radio 的其他要求，可能包含另外的 Bluetooth 连接，还常常包括 WiFi。这是一个由芯片设计者处理的复杂操作。但是，正如我们稍后将看到的，Bluetooth LE Audio profile 为应用程序提供了相当大的余地来影响调度，这就是开发人员需要对 Isochronous Streams 的工作原理有清晰的了解的原因。

对于单播 Bluetooth LE Audio，我们在拓扑结构方面有很大的灵活性，如图 4.3 所示。我们可以复制 HFP 或 A2DP 中的相同拓扑结构，其中我们只有一个 Central 设备——通常是你手机，以及一个 Peripheral 设备，比如你的耳机，它们之间有音频链接。在此基础上，Bluetooth 技术可以支持一个 Initiator 和两个或更多的 Acceptor 交互。它的主要应用是允许你的手机和

---

你的左右耳塞或助听器进行交互。它们不在需要来自同一个制造商，因为 Bluetooth LE Audio 是一个可互操作的标准。

我们可以通过添加额外的单播音频流，以此来扩展该功能，以支持多对耳塞或连接多个扬声器以支持环绕音响系统，图 4.3 实例显示了添加的中央低音设备。

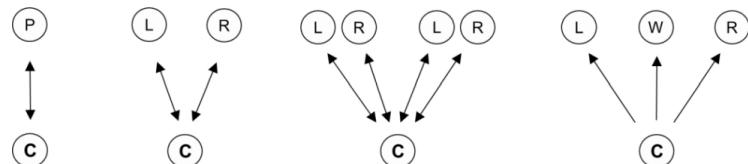


图 4.3 单播拓扑结构

理论上，规范可以支持多达 31 个单播 Isochronous Streams，可以连接 31 个不同的设备。对于音频来说不现实，因为在 3 到 4 个音频流之后，我们将开始耗尽带宽。Core 规范中支持 31 个音频流的原因是，Isochronous Streams 特性旨在支持许多不同的时间关键型应用程序——不仅仅是音频。其中一些需要的带宽要比音频小得多。当我们研究 LC3 codec 时，我们会发现我们必须在延迟、音频质量和鲁棒性之间进行一些权衡，这对我们实际支持的音频数量造成了限制。

使用广播的原因之一就是单播可以支持的音频流有数量的限制。如图 4.4 所示，单个 Broadcaster 可以和过的 Acceptor 交互，它们经常被配置成一对设备，接收单声道或独立的左右音频通道。根据音频质量(通常设置采样率，从而设置播放时间以及最大的音频流的数量)，Broadcaster 可能提供更强功能，比如传播不同语言的同步音频流。这些是设计 Bluetooth LE Audio 应用程序时需要了解的权衡之处，我们将在下一章节中详细介绍。

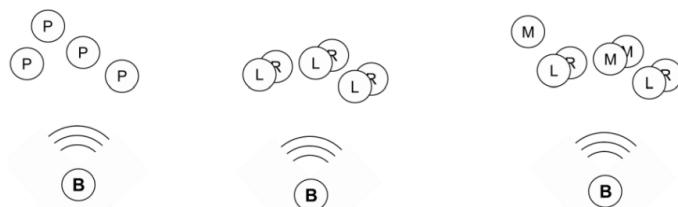


图 4.4 广播音频拓扑结构

### 4.3 Connected Isochronous Streams

为了理解 Core Isochronous 特性，我们将从单播和 Connected Isochronous Streams 开始，它们叫做 CIS。它们的结构十分复杂，但建立在

---

一些简单的原则之上。我将描述 Connected Isochronous Streams 是如何设计的，解释其组件以及工作方式，然后看看广播有何不同。这为你进入 Generic Audio Framework 奠定了基础，我们在 Generic Audio Framework 中使用 Isochronous Streams。

#### 4.3.1 The CIS structure and timings

当你设计数字音频时，你通常会受到限制，即你要以标准的、一致的速率对传入的音频进行采样。一旦输入的音频被采样和编码，它会被发送到 Bluetooth 发射器，发送到接收设备。系统是重复的，音频数据是有时间界限的，并且传输之间有一个恒定的间隔，称为 Isochronous Interval 或者是 ISO\_Interval。CIS 中，每个 Isochronous Interval 的起始点叫做 Anchor Point。如图 4.5 所示，传输从 Initiator 发送包含音频数据(D)的数据包到 Acceptor 开始。当 Acceptor 接收到了，就会发送一个确认，并且定期重复该过程。图 4.5 中的第三个数据包没有确认，以此 Initiator 或认为它没有被收到。

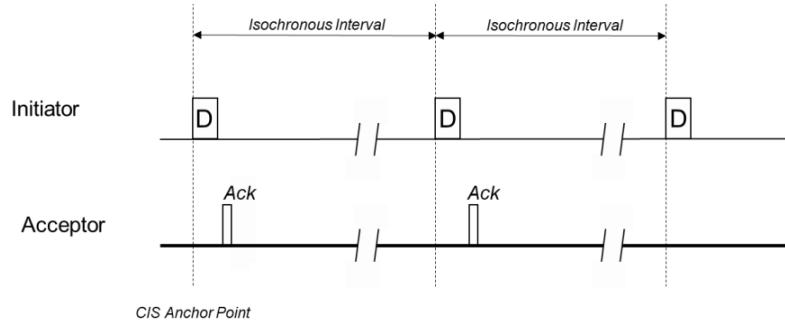


图 4.5 简化的单播音频传输

大多数现代 codec 都优化为以 10ms 的帧速率运行，即一次采样 10ms 的音频，这种方式在音频质量和延迟之间提供了良好的折衷。这是 LC3 的首选设置，LC3 是 Bluetooth LE Audio 的强制 codec。除非我另外有说明，我们将假设在本书中使用 10ms 采样间隔，因此 Isochronous Intervals 始终是 10ms，或 10ms 的倍数。

##### 4.3.1.1 Isochronous Payloads

图 4.5 中所示的，在设备之间发送的空口数据包(D)的 PDU 的数据结构，很简单，如图 4.6 所示。

编码的 ISO PDU 前面是前导码和访问地址，后面是 CRC。当通过 LE 1M PHY 传输时，这些增加了 10 或 14 个字节，(取决于 ISO PDU 负载中是不是包含 Message Integrity Check (MIC))，当使用 LE 2M PHY 时，增加了

---

11 或 15 个字节。

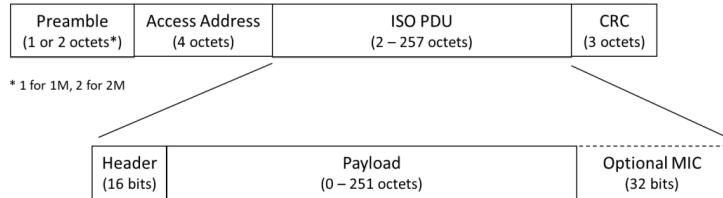


图 4.6 Bluetooth LE 链路层数据包格式

对于 CIS，ISO PDU 叫做 CIS PDU，它的结构如图 4.7 所示。

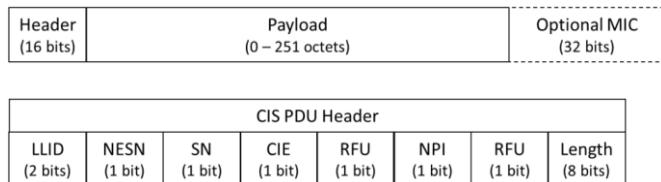


图 4.7 CIS 的 ISO PDU 以及 header

ISO PDU 有个 header，紧接着是最多 251 字节的 payload。如果音频需要加密，在数据包的结尾有个 MIC 的可选项。在 CIS PDU 的 header 中，有 5 种控制元素：

- LLID (Link Layer ID)，指示是否为 framed 或 unframed
- NESN 和 SN，NextExpectedSequenceNumber 和 SequenceNumber
- CIE，Close Isochronous Event bit
- NPI，Null Payload Indicator，指示 payload 是一个空的 PDU，显示没有数据要发送。

#### 4.3.1.2 Subevents and retransmissions

我们将介绍 ISO PDU header 中的控制位，因为它们与 CIS 如何工作的解释相关。在此之前，我们需要研究 Connected Isochronous Stream 的结构。我们已经讨论过 Isochronous Interval，即 CIS 的连续 Anchor Point 之间的时间。Anchor Point 是 CIS 的第一个数据包由 Initiator 传输的时间点，也是每个连续的 Isochronous Interval 的起始点。在 CIS 中，如果需要，我们可以重传 CIS PDU，因为 CIS 结构支持多个 Subevents。每个 Subevents 以来自 Initiator 的传输为开始，以来自 Acceptor 的、期望响应的终点为结束。CIS 中的所有 Subevents 形成了 CIS event，它从 CIS 的 Anchor Point 开始，以从 Acceptor 收到的最后传输 bit 为结束。

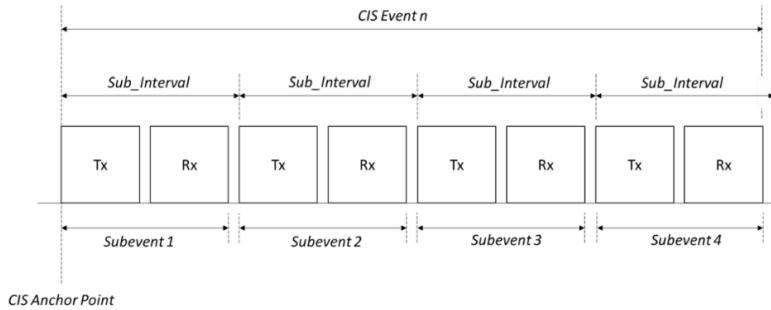


图 4.8 Events, Subevents 和 Sub\_Interval

连续 Subevents 之间的时间叫做 Sub\_Interval 间隔，它是 CIS Subevent 的最大持续时间，加上定义为 150us inter-frame 的间隔。Sub-Interval 间隔在 CIS 配置时就确定了，在 CIS 的生存周期内不会更改。

#### 4.3.1.3 Frequency hopping

定义 Subevents 的一个重要原因是，Bluetooth LE Audio 在每个 Subevents 上都会改变其传输信道，如图 4.9 所示，为了避免干扰。

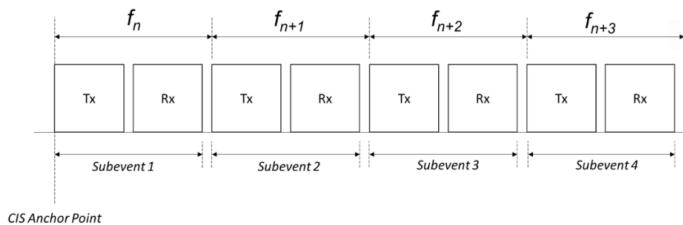


图 4.9 每个 Subevents 都 Frequency hopping

Core 5.2 规范引入了新的信道选择算法，该算法比在 Core4.0 中使用的算法更加高效。它应用在了每个 Subevent。如果未发送 Subevent，信道调频机制假定它已经发生并移到下一个 Subevent 的频率信道。

#### 4.3.1.4 Closing a Subevent

当我们研究一下 isochronous PDU header，就会看见有一个 Close Isochronous Event bit——CIE。Initiator 用它来表示它已经从 Acceptor 收到确认其数据包已经被 Acceptor 收到，以便它将停止该特定的 PDU 的进一步重传。图 4.10 中，Initiator 已经发送了它的第一个 PDU，并且已经收到确认，因此它发送 CIE bit 设置成 1 的数据包，以此告诉 Acceptor 之后没有更多的传输了，因为它正在关闭 event。它通常不会在该传输中包含音频数据，引起疑惑，因此它还将设置 Null Packet Indicator bit，允许它缩短传输数据包。(图 4.10 显示了原始的 CIS Event 过程，以供比较。)

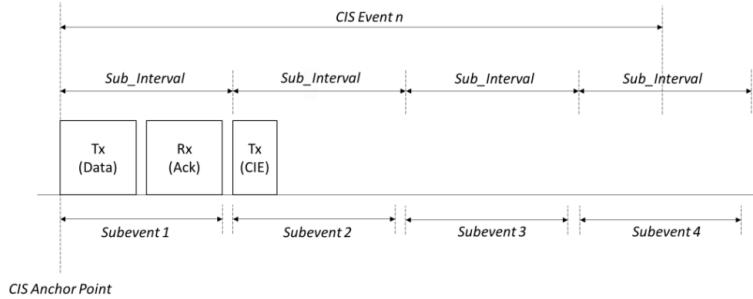


图 4.10 使用 CIE bit 关闭 CIS

这允许 Acceptor 进入睡眠状态，直到下一个 Isochronous Interval 到来。Initiator 可以利用这段时间做其他事情。由于许多 Initiator 也会和另外的 Bluetooth 设备交互，并可能与 wifi 共享 radio 和天线，上述过程很有用。对于 Acceptor，关闭它的接收器直到它准备好再次执行某项操作，可以显著节省电量。

如果 Acceptor 没有收到带 CIE bit 的 header，它会继续监听每一个 scheduled Subevent 中的数据，但不会接收任何来自 Initiator 的数据包，并响应之。

### 4.3.2 Controlling audio quality and robustness

已经看完 CIS 中传输的基本时序之后，我们现在可以来研究一下用于控制音频质量和链路鲁棒性的参数了。对于许多音频应用，latency 是很重要的。对于一些应用，比如当你们在收听实况音频流时，减小 latency 就很重要了，尤其是当你也能听到环境音时。另一方面，如果你是通过手机在播放音乐，并且不能听见或看见播放源的话，latency 就不那么重要了。另外一些应用，比如游戏，具有不同的优先级，并且如果你在看电影时听它的声音，音画同步就变得重要了。

Basic Audio Profile (BAP) 可以通过使用 HCI 命令 LE\_Set\_CIG\_Parameters 设置许多影响音频质量和 latency 的参数。我们会在第 7 章查看一下怎样选择并设置这些参数，但是，目前来说，我们需要理解 Isochronous Channel 结构体怎样配置的。应用程序可以请求的、用于影响 latency 和鲁棒性的关键条目如下：

- Maximum Transport Latency，对于特定的 CIS，Initiator 用于传输 PDUs 可以花费的最长时间。
- 对于 CIS 两个方向的 Maximum SDU size，

- 
- 对于两个方向的 SDU interval。

Maximum Transport Latency 影响全局的 latency，尽管它只有一个元素。虽然许多的应用程序都想将延迟降到最低，但是在无线的世界里，你仍然需要解决无线链路固有的脆弱性，即数据包可能会丢失。为了确保足够的鲁棒性(翻译为：没有断音、咔哒声和静音的音频和音乐流呈现)，我们需要使用各种技术以帮助确保音频数据在可接受的时间内被收到。

#### 4.3.2.1 Flush Timeout and Number of Subevents

上面列出来的是输入给 Controller 的三个参数。Controller 拿到它们并且用它们来计算影响该 CIS 的 Bluetooth LE Audio 链路鲁棒性的三个参数，它们是：

- NSE——Number of Subevents。它明确了在每个 Isochronous Interval 将要预设的 Subevent 的个数。它们用于 CIS 的初始传输。
- PDU 以及 subsequent retransmissions。它们可能不会全部被使用，但是，它们是预设的固定值。
- FT——Flush Timeout。定义了在丢掉 PDU 之前，可以使用多少个连续的 Isochronous Interval 来传输它。不能再继续传输的点就叫做 Flush Point。
- BN——Burst Number，提供给在每个 CIS event 里传输的 payload 数量。

它们可能很难被理解，所有查看一些简单的实例很有用。

Number of Subevents (NSE) 是三个当中最简单的。就是指在每个 Isochronous Interval 之内，可以传输 Isochronous PDU 的机会的数量。最简单的实例，即每个 Isochronous Interval 里，只提供一个 PDU 用于传输，该 PDU 会在第一个 subevent 里传输，并且在同一个 Isochronous Interval 里面，可以被重传最多(NSE-1)的次数。如果它是单向的 CIS，一旦该 PDU 的传输被 Acceptor 应答了，Controller 可以在它下一次传输(可以是 null PDU payload)的 header 里面，将 Close Isochronous Event(CIE)置位，并且在此 CIS Event 中剩余的 Subevent 对于其他 radio 应用来说就变成了 free airtime。

当 Flush Point 与此 Isochronous Interval 的 CIS Event 的结尾相同时，即为 Flush Timeout 为 1 的实例。如图 4.11 所示的简单的实例。为了清晰，如下的实例只有一个 Acceptor。

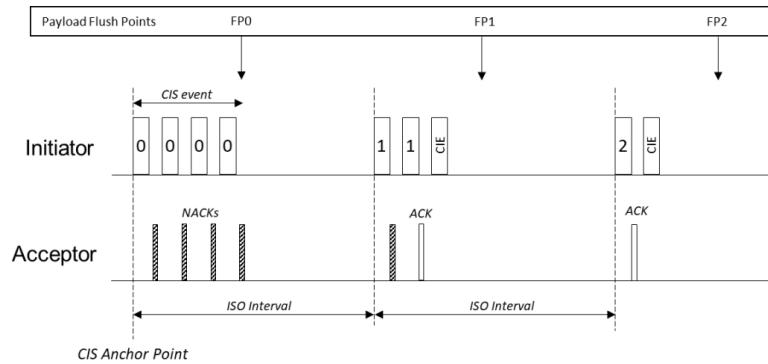


图 4.11 NSE=4 并且 FT=1 的单向 CIS

在此例中，NSE 设置为 4，所以每个数据包有四次的传输机会。在第一个 CIS Event 中，四次尝试都没有成功，所以 P0 被刷了，Acceptor 需要使用某种 Packet Loss Correction 形式来重建它。

第二包(P1)在第二次尝试之后成功了，之后，Initiator 便关闭了 Event。第三包(P2)在第一次就成功了。

如果增加了 Flush Timeout，数据包的传输可以跨越更多的 Isochronous Interval。图 4.12 展示了 NSE 仍然是 4，但是 Flush Timeout 增加到 3 的实例。

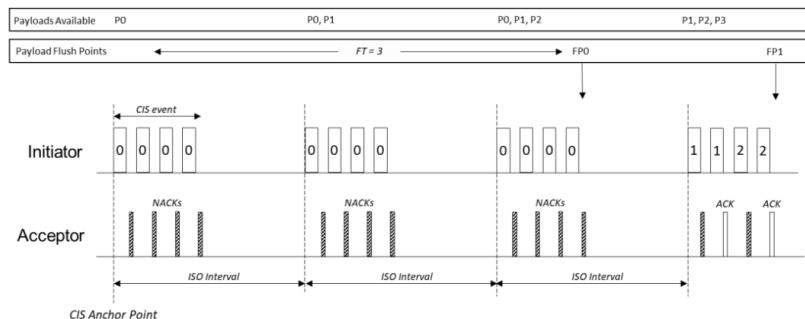


图 4.12 FT=3 并且 NSE=4 的实例

改图展示了如果只使用两个参数——NSE 和 FT 时，会出现的一个问题。它允许数据包影响 slot，直到它到达 flush point。图 4.12 中，第一个 payload，P0，接收有问题，占用了前三个 Isochronous Interval 的所以 Subevent，让 P1 和 P2 等到了 Flush Point FP0 之后。尽管此种情形不常见，但它会让后续的 payload 曝露更久，直到它们足够多的被接收以使系统回到平衡。

#### 4.3.2.2 Burst Number

解决此问题的方法是允许在单个 Isochronous Interval 里面传输一个以上的 payload，这样每个 Isochronous Interval 里面的 Subevent 可以在一个以上

的 PDU 之间分享，并且不是被它们中的某一个独享。这可以通过使用 Burst Number(BN)来完成，它是在一个 CIS event 里面提供的用于传输的 payload 的个数。上面的实例中，在一个 CIS Event 中，只有一个数据包用于传递，这是 SDU 和 PDU 生成的节奏和 Isochronous Interval 相同的情况——每个 10ms 的 Isochronous Interval 有一个编码的 10ms 音频 frame 可用。如果我们想使用 BN，并且我们继续每 10ms 采样输入 Audio Channel，我们需要增加 Isochronous Interval 的数量使它为 10ms 的倍数，这样我们在每个 Isochronous Interval 里面就有更多的可用的数据包。这意味着通过解决一个问题，我们潜在地增加了另一个问题，就是增加了 latency。

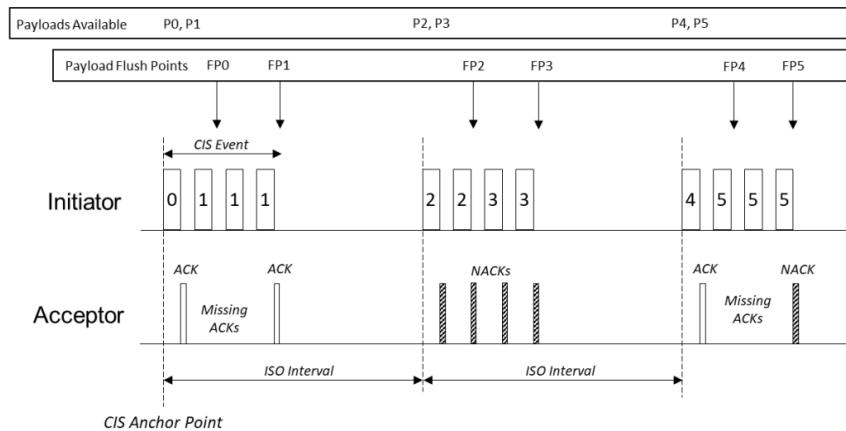


图 4.13 Burst Number=2, NSE=4 并且 FT=1 的效果

在图 4.13 中，Isochronous Interval 加倍了，每个 interval 允许提供两个数据包。10ms codec frame 以及 20ms Isochronous Interval 意味着 Initiator 在每个 Isochronous Interval 中有两个 PDU 可以用于传输。它的一个结果是在每个 Isochronous Interval 中有两个 flush point。在我们的简单实例中，每个 Flush Point 出现在两个 Subevent 之后。FT 和 NSE 参数的更复杂组合、Flush Point 的位置可用 Core [Vol 6, Part B, 4.5.13.5]里的等式来计算。

回到图 4.13，我们可用看见 Initiator 在第一个 Subevent 发送数据包 P0，它很快得到了回应，所以 Controller 立即继续发送数据包 P1，在它得到回应之前发送了 3 次。

在第二个 Isochronous Interval 中，数据包 P2 发送，但是 Acceptor 回了 NACKs，以此表示在接收的数据包中有错误。因为 Flush Timeout 设置为 1，并且 BN 设置为 2，数据包 P2 的 Flush Point 在同一个 Isochronous Interval 中，出现在两次进一步的发送尝试之后，都没被 Acceptor 成功接收。

此刻，Initiator 开始发送 P3，然而再一次，在此例中，Acceptor 用 NACKs 来响应，以此表示接收到的数据包有问题。两次的尝试之后，P3 就被 Initiator 刷掉了。

在图 4.13 的最后的 Isochronous Interval 中，P4 被成功发送了，P5 尝试了 3 次，3 次都没有成功，在每一个实例中，Initiator 会在 PDU 的 Flush Point 之前的每一次可用的 Subevent 中去尝试发送该 PDU。在每次应答接收到之后，它就会转到下一个可用的数据包，或者如果没有数据包的话，关闭 Isochronous Event。

在此例中，P2，P3 和 P5 会被丢弃。在显示生活中，我们期待一个更好的应答速率——这只是一个实例，用来说明这一规则。

作为最后一个实例，图 4.14 展示了使用 Burst Number 为 2，增加 Flush Timeout 到 2 的效果，意味着在两个连续的 Isochronous Interval 中有发送的机会。此例中，没有数据包被刷写掉。

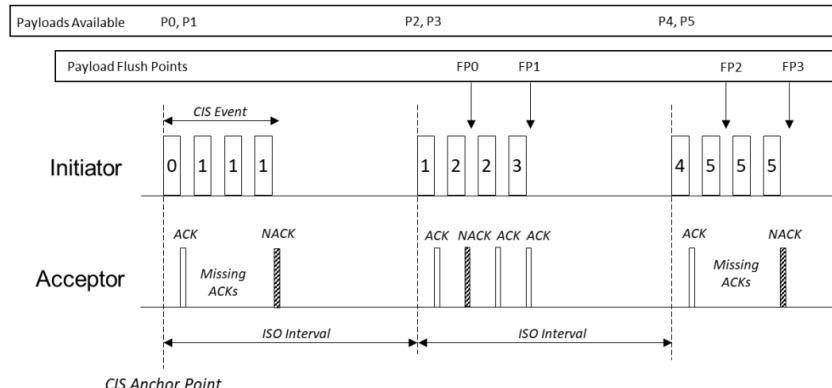


图 4.14 NSE=4，BN=2，并且 FT=2 的实例

使用 Flush Timeout 和 Burst Number 对于跨越多个 Isochronous Interval 提供更多的重传机会非常有用。如果你在吵闹的环境中，它们尤其有用。然而，它们会影响 latency。每增加一个 Flush Timeout 就会增加 latency，因为重传会扩展到多个 Isochronous Interval，同时，Burst Number 增加了 Isochronous Interval 的持续时间。注意，Burst Number 仅限于一个 Isochronous Interval 中到来多个 payload。不能靠它去解决单个 payload 独占传输机会的问题，如图 4.12 所示。

这些参数不能由 HOST 直接设定。受限于 Maximum Transport Latency，Maximum SDU size 以及 SDU interval 的设定值。BN、NSE 和 FT 会在 Controller 中计算得出，会将芯片中的任何其他 radio 需求也考虑进去。然

而，这对于理解这些参数对 Isochronous Channel 的结构的潜在影响很有用。TMAP 的表格 3.19 提供了一个实例，Controller 为一个 CIS 解析 Host 的值，以此适应不同的操作条件，比如优先 airtime for coexistence 或是 minimising latency。参数的准确分配总是由 chip supplier 设置的、scheduler 中的算法决定。

### 4.3.3 Framing

CIS PDU header 中的另一个需要理解的参数是一对 LLID(Link Layer ID)位，它们表示 CIS 是 framed 或是 unframed。Unframed 表示一个 PDU 由一个或更多的完整的 codec frame 组成。用在 Isochronous Interval 是 codec frame 长度整数倍的时候。相反，framed 用在 codec frame 长度和 Isochronous Interval 之间不匹配的时候，导致了 codec frame 会被分段，跨越多个 SDU。这会变得复杂起来，但在 Initiator 可能需要支持具有不同时序的 Bluetooth 连接的时候是很重要的。我们会在之后查阅叫做 ISOAL 的特征的时候再来讨论，ISOAL 就是 Isochronous Adaption Layer，被设计用来解决上述的不匹配。(LLID 位也指示何时没有 ISO PDU，这与 ISO PDU header 中的 NPI 不同。)

### 4.3.4 Multiple CISes

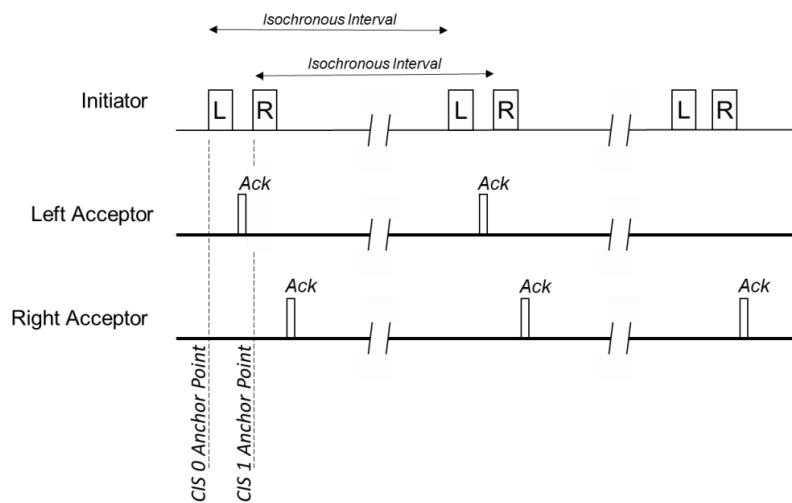


图 4.14 两个 Acceptor 的 CIS 时序

看过了单个、单向的 CIS 的所有特性之后，下一步是增加更多的 CIS。它的最普通的应用就是当 Initiator 向左耳塞和右耳塞发送数据的时候。这种情况下，Initiator 会与两个不同的 Acceptor 建立单独的 Isochronous Stream。

---

在图 4.15 中，我们可以发现它就是我们之前所看到的简单的扩展——Initiator 发送及从第一个 Acceptor 接收应答，然后对第二个 Acceptor 重复此操作。

需要注意的重要一点是，尽管左右 Audio Channel 是在同一时间采样的，但是 ISO PDU 是串行发送的。

注意，**我们具有的一个以上的 CIS 时，它们总是具有相同的 Isochronous Interval**。它们的 Anchor Point 是不同的，因为数据是串行发送的，但是对于每一个 CIS，它们的 Anchor Point 之间都是相同的 Isochronous Interval 间隔。每个 Anchor Point 代表了该 CIS 的 Initiator 到 Acceptor 的第一次传输点。如图 4.16 所示，**每个 CIS 都有一个相关联的 ACL 链接**。该链路总是需要存在，因为它是用来建立 CIS 并且控制它的。如果在 Initiator 和 Acceptor 之间有多个 CIS，它们分享同一个 ACL。如果该 ACL 由于任何原因断开了，任何相关的 CIS 就会被中断。然后应用程序就需要决定要怎样处理在此 CIG 内，其余的、与另外的 Acceptor 建立的 CIS。

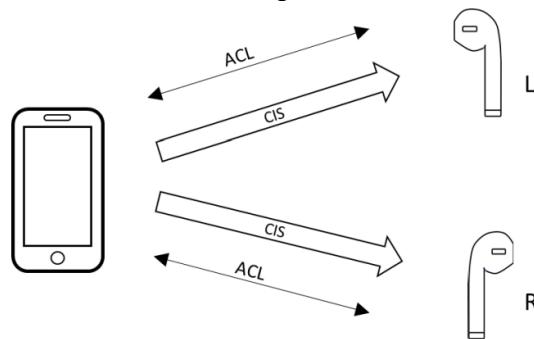


图 4.16 多个 CIS 以及相关联的 ACL

当 Initiator 规划两个或更多的音频通道时，不管它们是连接到一个或是多个 Acceptor，都有两种传输方式选项。

很明显的一种方法是依次传输，这样，你可以发送完所有的 CIS0 的数据之后，再发送 CIS1 的所有数据，继续直到 Initiator 完成两个 CIS 的所有数据包的发送。如图 4.17 中展示的，里面的 NSE 设置为 3。它展示了 CIS0 中所有的 Subevent 在 CIS1 之前传输。

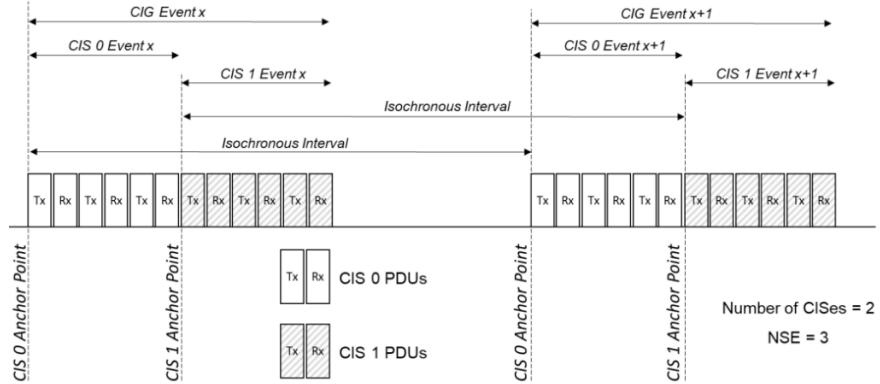


图 4.17 两个 CIS 的 Sequential 传输安排

这种方法的缺点是，如果你具有可靠的连接，在第一个数据包传输得到应答之后，你会在每个 CIS 之间留有空隙。在比如手机这类的设备中，它们经常共享 Bluetooth 和 Wifi radio，上述空隙就是浪费的 airtime，本来可以用于另外的事情。为了解决这个问题，Controller 可以选择 interleave CIS 作为替换方法，如图 4.18 所示。

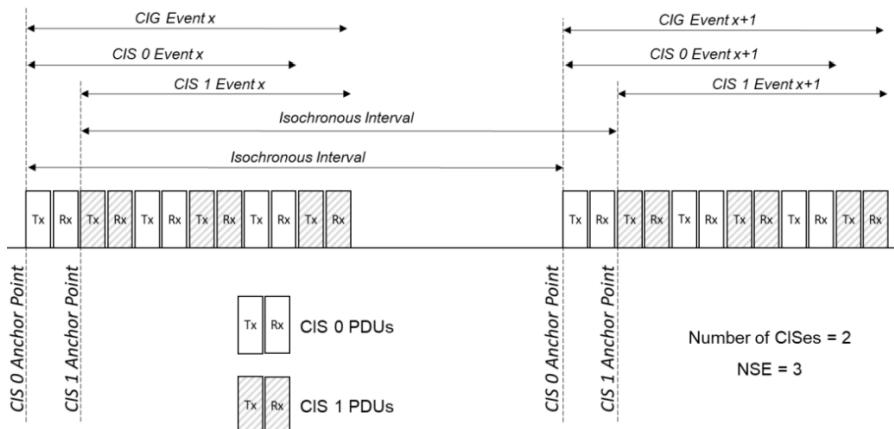


图 4.18 两个 CIS 的 Interleaved 传输安排

在此实例中，每个 CIS0 的 Subevent 之后都跟着 CIS1 的 Subevent，并且剩下的 CIS 都如此。该图与图 4.17 一样，NSE 设置为 3，并且展示了 CIS0 发送和接收它的第一个 Subevent，紧接着是 CIS1。如果两个 Acceptor 都接收到了这些第一个传输并且对它们进行了应答，它们可以关闭这些 CIS Event。这会导致在传输之后的更大的间隙，释放的 airtime 可用于其他目的。

#### 4.3.5 Bidirectional CISes

我们上面所定义的，多个、单向 connected Isochronous Streams 重塑了 A2DP 的用例。对于耳塞和许多其他的音频应用，我们也想要反向的数据，

因此需要双向的 CIS。一种方法是建立另外一个反方向的 CIS，这样我们可以使用一个 CIS 进行从手机到耳塞的传输，使用另外一个进行从耳塞到手机的传输。然而，这样效率很低下。Core 规范提供了优化的方法——将返回数据添加进应答数据包中。仍然使用单个 CIS，但是，现在可以用于两个独立的 Audio Stream。

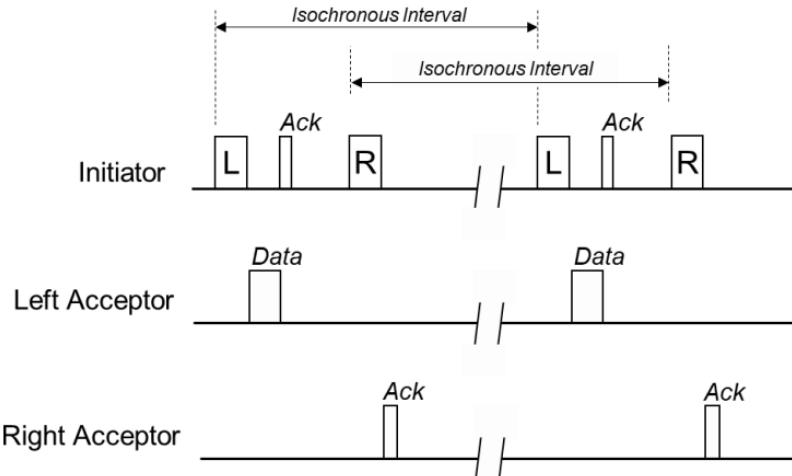


图 4.19 左耳塞双向 CIS 以及有耳塞单向 CIS 实例

图 4.19 展示的实例中，Initiator 与左侧 Acceptor 和右侧 Acceptor 建立了单独 CIS 的。两边都从 Initiator 接收数据，但是左侧的 Acceptor 也会将来来自于它的麦克风的数据发送回手机。之前看到的原理实用了。数据由 Initiator 发送，Acceptor 立即响应已经收到(或者没有)了，如果它有数据，会将数据包进 CIS PDU 中。如果返回的 PDU 的 CRC 没有问题，Initiator 会有应答，关闭该 event，并且，将数据发送到正确的 Acceptor CIS。

两个方向的应答都使用 ISO PDU header 中的 NESN 和 SN 位，在确认数据包时反转。在图 4.19 中，所有的传输的数据包都有相同的 ISO PDU 格式。标记为“Ack”的数据包没有音频数据，因此需要将 NPI bit 设置为 1，以表示它们有 null CIS PDU。来自于左侧 Acceptor 的数据包与 Initiator 的“L”和“R”数据包格式相同，但是包含来自 Acceptor 的麦克风的数据。Initiator 对于左侧 Acceptor 数据的“Ack”会包含 CIE 位，出现在它向右侧 Acceptor 传输数据之前表明它正在以 **sequential(interleaved)** 模式传输 CIS。这些“Acks”也会将 CIE 设置为 1。

NSE 应用在 CIS 的两个方向上，与用于传输到达和来自 Acceptor 的 PDU 的 Subevent 相同。一旦一个方向上的 payload 被确认了，后续发送可以使用 null payload 替换掉 PDU 中的 payload。(这对于 Subevent 的时序没

有影响，但是可以节省一点功耗。）

在双向的 CIS 中，只有 Initiator 可以设置 ISO PDU header 中的 CIE 位，以此表示在当前的 CIE Event 中，它们不会进行更多的 payload 数据传输。除非两个方向的 payload 都被确认了，否则不应该关闭该 CIS Event。如果 Initiator 已经收到它的数据包的确认，但是没有收到 Acceptor 的数据包，Initiator 会在每一个可用的 Subevent 继续发送 null PDU 数据包，给 Acceptor 重传的机会。

双向 CIS 中的两个方向可用有不同的特性，比如 codec 和 PHY 以及甚至可以被不同的应用所使用。实际上，两个方向的一些结构参数可以是不一样的。FT 和 BN 可以不一样，Max\_PDU 和 Max\_SDU 的值也可以不一样，意味着 SDU\_Interval 可以不一样，尽管一个需要是另一个的整数倍。然而，**ISO\_Interval、Packing、Framing 和 NSE 必须是一样的。**

#### 4.3.6 Synchronisation in a CIS

一旦我们添加了第二个 Acceptor，两个 Acceptor 的行为是彼此是独立的，但是，都是在 Initiator 控制之下。如果它们是一个 Coordinated Set，它们会知道另一个的存在，因为它们知道在这个 Coordinated Set 中有多少个成员。然而，你的左耳塞和右耳塞不需要知道另一个的存在、打开或正在接收数据。用户可能将一个耳塞与朋友分享，或者一个中的电池没电了。甚至即使它们有方法进行通讯，也不能保证它们总是能够联系上彼此。为了处理这个问题并且使它们能够精确地同时呈现或者捕获音频流，Core Isochronous Channel 的设计包含了一个同步方法，给予呈现微秒级别精度，而不需要 Acceptor 知晓另一个 Acceptor 的存在与否。图 4.20 展示了它是怎样实现的。

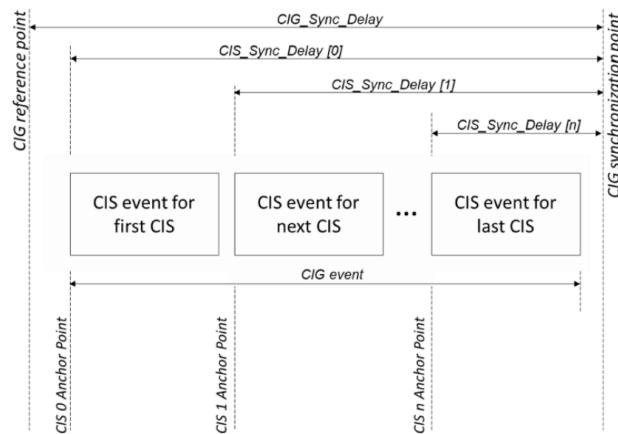


图 4.20 多 CIS 的同步

---

在 CIG 中，CIS Event 是为此 CIG 中的每一个 CIS 规划的。对于一对耳塞来说，会有两个 CIS event——一个用于左耳塞，一个用于右耳塞。也有可能更多，比如多声道音响系统。这些 CIS Event 组成了 CIG Event。在图 4.20 中，为了简化，多个 CIS 按照顺序显示。它们同样可以交错排列。

Core 定义一个 CIG reference point，它可能和第一个 CIS 的 Anchor Point 是同一点，但是不能晚于该 Anchor point。通常来说，它会稍稍早于该 Anchor Point。Core 同样也定义了一个 CIG synchronization point，它出现在该 CIG 中最后一个 CIS 的最后可能的接收事件之后。在此点，Initiator 知道每个 Acceptor 具有将发送给它的每个 PDU 接收到的机会，以及具有将给 Initiator 的任何数据返回的时间。

为了重建此公共点，每个设备都会被告知其支持的每个 CIS 的各自 CIS Sync Delay，它是由与此 CIS 相关联的 ACL link 的 Instant 计算得来。允许计算此 CIG 的 synchronization point，它是每个设备的公共时间戳。有了这个信息，每个设备就能够决定它开始解码音频的时间。BAP 添加了一个特性叫做 Presentation Delay，用来告诉 Acceptor 什么时候开始呈现解码过后的音频流。当我们向上层移动并且开始挖掘 QoS 和 Latency 的基本概念的细节时，我们就会看见怎样使用 Presentation Delay 增加呈现时间的灵活性了。

如果你的设备有麦克风，同样的同步问题会存在，但是在此用例中，你需要调节 Acceptor 捕获的音频的上述点。如果它使用双向 CIS 的上行链路，它会使用相同的 CIG 同步点，但是可能需要一个不同于用来呈现来自 Initiator 的下行链路音频数据的 Presentation Delay 值，因为它需要额外的时间来捕获和编码音频流。

同步时序定义在微秒的精度。意味着在一个 CIG 中的所有 Acceptor 的 Controller 都分配了一个时间戳，时间戳彼此相差几微秒。这需要传递到呈现音频的应用层，尽管这样做的方法取决于实现。然而，此规范意味着左和右耳塞可以将两个音频流以相当近的顺序呈现给耳朵，以至于大脑都察觉不到。

#### 4.3.7 The CIG state machine

看完了构成 Connected Isochronous Streams 的所有特性之后，我们可以查看一下怎样将它们合在一起，并且建立 Connected Isochronous Group。

图 4.21 展示了相当直观的状态机，该状态机用于配置和建立 CIS 和它的组成要素 CIS。它包含了三个状态：

- Configurable CIG 状态，在所有组成 CIG 的 CIS 都定义好的时候。通过叫做 LE Set CIG Parameters 的 HCI 命令集合完成。一旦这些命令发送给了 Controller，Initiator 具有了所有它需要的、用于计算时序安排并优化 airtime 使用的信息。
- Active CIG 状态，此状态下，一个或更多的 CIS 被使能了。CIG 通过使用 LE Create CIS HCI 命令，使能至少一个已配置好的 CIS，从而转变为 active 状态。不需要使能所有的 CIS。一旦 CIG 进入 Active 状态，它就可以断开单个 CIS，并且可以通过 LE Create CIS 命令使能它之前就已经配置好的 CIS。理论上，允许 CIG 在 Active 状态替换 CIS。这对于偶尔需要 CIS 的时候就很有用了，比如需要给语音命令一个返回路径的时候。按照需求，CIS 可以被开启和关闭。CIG 在 active 状态时，不能去配置更多的 CIS。只有通过断开所有的 CIS，停止 CIG 并且重新开启 CIG 才可以完成。
- Inactive CIG 状态，通过断开所有建立好的 CIS 可以使 CIG 进入 Inactive 状态。此状态下，不能改变任何 CIS 的配置，也不能添加新的 CIS，但是可以通过使用 LE Create CIS HCI 命令同重建 CIS，并回到 Active 状态。允许 CIG reactive，而不需要 Controller 重新去开启并为一个或更多配置好的 CIS 规划时序。

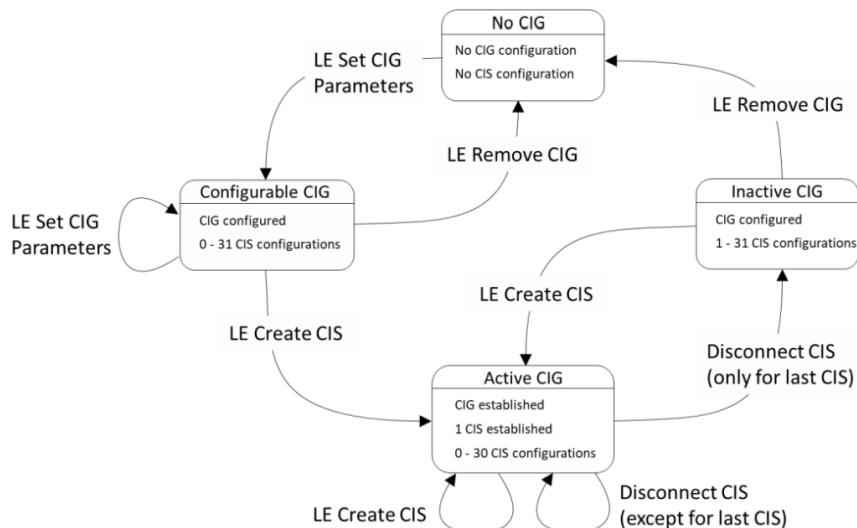


图 4.21 CIG 状态机

当所有的 CIS 被断开之后，可以通过发送 LE Remove CIG HCI 命令移除掉 CIG。

---

#### 4.3.8 HCI commands for CISes

Core 规范定义了 5 个 HCI 命令用来将关于每个 CIG 的信息通知 Controller。第一个命令是 LE Set CIG Parameters 命令[Vol 4, Part E, 7.8.97]，它创建了 CIG 并且在其中配置了一组 CIS。该 CIG 中的每个 CIS 可以是单向的或者双向的。每个都可以在不同的方向使用不同的 PHY。

第一次使用 LE Set CIG Parameters 创建 CIG，让其转变为 Configured CIG 状态，并且具有定义在数组中的数目的 CIS。此命令可以多次使用，以增加更多的 CIS 到 CIG 中，或者改进已经定义好的 CIS。每个 CIS 都分配了 Connection handle。

命令每次发送之后，Controller 就会尝试为 CIG 中所有的 CIS 计算出时序规划，以满足 HCI 命令中明确的需求，如果成功，会使用 HCI Complete Command 来通知。要注意两个 Host 发送的参数——Packing 和 RTN(重传次数)，只是推荐使用。Controller 在计算时序规划时应尝试适应它们，或者忽略它们，或者在它们的引导下创建更好的时序规划。Link Layer 参数 Burst Number、Flush Timeout 和 NSE 是由 Controller 时序规划算法决定的，并且不能通过上层的规范来设置。一些 profile 规范提供推荐参数值，是对于特定的用例的建议优化设定，但是它们是否可被应用所接受，或者是时序规划选择的一部分，取决于具体芯片的实现。

这种从 Host 发送的 HCI 参数与 Controller 设定的参数不匹配可能看起来奇怪，但是这是有很好的理由的，这是为了防止高层的 profile 或者应用尝试优先占用整个 radio 的操作。许多 Bluetooth 芯片包括了 WiFi，它们共用一个天线。因此，Controller 需要计算出怎样满足两者的需求。Bluetooth 技术的实现可能也会运行多个 profile。手机没有理由不能接收 Bluetooth LE Audio broadcast stream，然后将其作为一对 Connected Isochronous Streams 重传给一对助听器(除了可用的 airtime 和 processing 资源的物理限制之外)。然而，在 profile 层级，这些应用没有一个知道另一个的存在。如果每一个应用可以支配 BN、FT 和 NSE 的精确的值，那么最有可能的结果是它们在有另外的应用的约束下无法共存。这就是为什么 HCI 命令避免此层级的控制，运行 Controller 计算出怎样最适合每个应用。时序规划算法对于应用开发者是不可访问的——它们是 Bluetooth 芯片的关键部分。然而，理解为何你不能支配你想要的，并且意识到过度规定你的音频应用的需求的危险性。

当配置好了所有的 CIS，就可以用 LE Create CIS 命令去创建需要的每一个 CIS，该命令会使 CIG 进入 Active CIG 状态。不是所有的 CIS 都需要

---

在这个点创建。在任何点，一个 CIS 可以被断开，另一个可以被创建，当应用程序决定从使用耳塞的麦克风转换到使用手机上的麦克风时，这就很有用了。然而，这是假设 Controller 对所有的 CIS 进行时序规划。

CIG 仍然处于 Active CIG 状态直到最后一个 CIS 被断开了。此刻它转换到 Inactive CIG 状态。可以使用 LE Remove CIG 命令永久性移除 CIG，或者通过在至少一个 CIS 上使用 LE Create 命令使 CIG 重新回到 Active CIG 状态。

每个 CIS 都是由 Initiator 创建，连接参数会通过 Link Layer 命令通知到每个 Acceptor，HCl LE CIS Request event [7.7.65.26] 就会被发送给它的 Host。如果它接受请求，它会用 HCl LE Accept CIS Request 命令来响应[7.8.101]，结果 Initiator 和 Acceptor 双方的 Host 都会收到 HCl LE CIS Established event [7.7.65.25]。当我们到 ASCS 时，我们会理解这些和 Isochronous Stream 是怎样运行。

完成了对 CIG 和 CIS 的组件，并且我们怎样将 unicast Connected Isochronous Streams 合在一起回顾。现在，我们来看看类似的广播，我们需要做相同的事情，但是不需要在两个设备之间建立额外的连接，以此允许它们协商彼此的行为。

#### 4.4 Broadcast Isochronous Streams

广播音频对于 Bluetooth 技术来说是全新的概念。在过去，音频总是从一个设备直接流向另一个设备，如我们在前面的章节所看到的。此概念已经扩展到 Connected Isochronous Streams，这样我们可以让音频流向多个设备。但是这些设备仍然是连接的并且每个正确接收到的数据包都会被确认。使用广播，是没有连接的。任何在广播发射器范围内的设备都可以接收到并且呈现 Audio Stream。与 Bluetooth Classic Audio profiles 最大的不同之处在于它是单向的，没有应答/确认。意味着建立一个没有接收器，只有发送的 Broadcast Source 在技术上是可行的。这些广播可以被在范围内的任意以及每个 Broadcast Sink 接收。

没有应答/确认会有实践上的优势，可以扩展大得多的范围。这是因为连接的链路预算通常是不对称的。Broadcast 发射机通常是电源供电，可以轻松按照允许的最大功耗发送。这就给了耳塞一个很好链路预算。然而，耳塞不太可能以超过 0dBm(1mW)的功率回传应答/确认，既是为了省电，也是因为它的天线太小，可能增益小于 0dB。这意味着，耳塞用于接收广播传输

---

的链路预算可能比返回应答/确认路径要高 10 到 20dB。后者决定了 CIS 的最大范围。对于公共覆盖，Broadcaster 可以覆盖相当大的范围——远超过 connected Isochronous Stream 所能达到的传输范围。

#### 4.4.1 The BIS structures

Broadcast Isochronous Streams 和 Groups 的定义与我们刚刚看完的 Connected Isochronous Streams 的定义非常相似。

图 4.22 展示了 Broadcast Isochronous Streams 具有相同的 PDU 结构，header、payload 以及可选的 MIC(如果你想要加密的话)。然而，BIS 的 header 简单多了，因为它不需要 SN 和 NESN 流量控制位，在 CIS 中的 PDU header 中是有的。它仍然由相同的两个 Link Layer ID 位开始。它们表示 PDU 为 framed 还是 unframed。然后是 CSSN 和 CSTF，用于指示有没有 Control Subevent。CSSN 就是 Control Subevent Sequence Number，CSTF 就是 Control Subevent Transmission Flag，指示 BIS Event 里面有 Control Subevent。这些是新的、在 Connected Isochronous Streams 里面没有的。BIG 中，Control Subevent 用来向所有的 Acceptor 提供控制信息。我们需要它们，因为没有 ACL 连接来通知 Acceptor 诸如调频序列改变的信息。Header 中其余的信息只有 payload 的长度了。

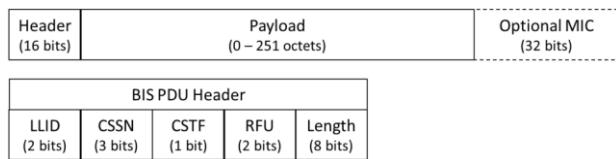


图 4.22 Isochronous PDU 和 Broadcast 的 header

我们看图 4.23 中的 Broadcast Isochronous Stream，它也非常地相似。BIS 和 CIS 之间根本上的不同点是没有应答/确认。BIS 纯粹由包含数据的 Subevent 组成，并由 Initiator 发送。没有双向数据——所有东西都是由 Broadcast Source 发送。如前所述，我们看到每个 Subevent 都是在不同频率的信道上面传输。

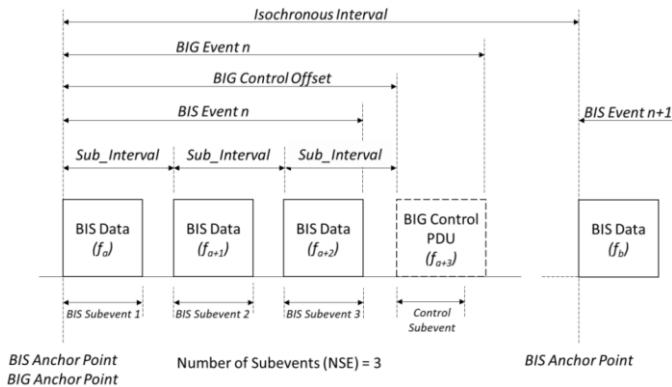


图 4.23 Broadcast Isochronous Stream 结构

**BIG**(Broadcast Isochronous Group, 由一个或多个 CIS 组成)和 CIS 显著的不同点是，潜在的 Control Subevent 的存在，如图 4.23 中虚线部分。当它存在时，(在 BIS Event 中，ISO PDU 的 header 包含 CSTF 标识)，Control Subevent 在最后一个 BIS 的结尾 Subevent 之后发送，并且提供向每个接收 broadcast stream 的设备发送控制信息的机会。我们会在 4.4.3 中讲到。

如图 4.23 中展示的，Broadcast Isochronous Stream 对于 BIS 和 BIG 具有相同的基本元素，Isochronous Interval、Anchor Point。没有应答/确认或者返回数据包，定义了 Sub\_Interval 时间，单个 BIS 内，连续 Subevent 起始点之间的时间。如果有 Control Subevent 的话，也是最后一个 BIS 的结尾 Subevent 和 Control Subevent 的起始点之间的时间。

如果 BIG 包含多个 BIS，每个 BIS 由 BIS\_Spacing 分隔开。通过调整 Sub\_Interval 和 BIS\_Spacing，BIS 可以按照 Sequential 或者 Interleaved 的方式排布，如图 4.24 和 4.25 展示。如果 BIS\_Spacing 大于 Sub\_Interval，它们会按照 Sequential 方式排布。如果小于，它们会按照 Interleaved 的方式排布。

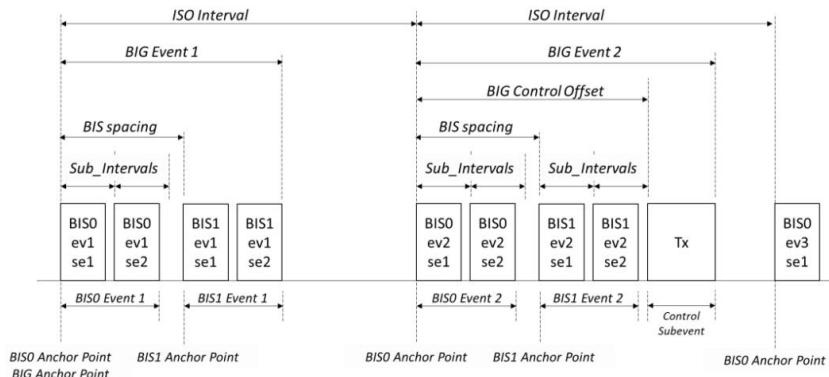


图 4.24 Sequential BIS

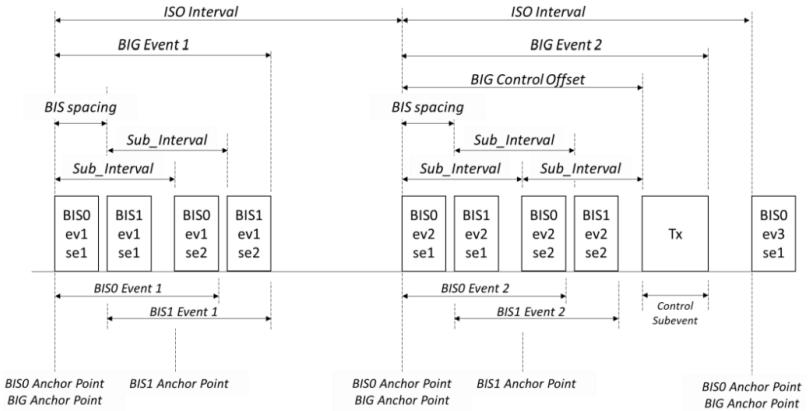


图 4.25 Interleaved BIS

两个图都展示了两个 CIS，每个 CIS 的 NSE 都设置成 2(例如，每个 CIS 有两个 Subevent)。可以从图中获得两个重要的观察点。第一，Control Subevent 从没有计入 NSE——它是一个完全单独的 Subevent。第二，当存在 Control Subevent 时，它毫不影响任何其他数据包或 Anchor Point。它会在最后 BIS 的最后一个 Subevent 之后立即被发送。但是它会增大该包含 Control Subevent 的 Isochronous Interval 的 BIG Event。

当设备已经收到音频数据包时，CIS 可以停止传输，Broadcast Source 就不清楚数据包有没有被收到了，所有它一直重复每一次传输。然而，当 Acceptor 收到数据包之后，它可以关闭它的 radio，并且等待下一次的 BIS。这再一次强调了 Bluetooth LE 中的不对称性。由于 Broadcaster Source 总是在传输，它通常比 Acceptor(比如耳塞)功耗高的多。一般来说，意味着 Broadcaster Source 需要更大的电池，或者永久电源。它们中的典型就是手机或者公共场所的基础设施发送器，通常是个发送者。但需要记住，现在的 broadcast 传输正在被嵌入到小型设备中，比如手表或者腕带。

与 CIS 不同，所有 BIS 具有相同的时序结构。尽管每个单独的 CIS 的结构，通常根据其 codec 配置进行定制，根据 PDU 的大小来优化 Subevent，但每个 BIS Subevent 都有相同的长度，必须符合要传输的最大 PDU。此约束是由于所有的 BIS 时序结构都定义在了 BIGInfo 中，包含在 Periodic Advertisement。BIGInfo 结构受限于它的 size，所有没有粒度去详细指明不同 BIS 的不同时序——它指明“一个大小适用全部”。意味着如果你想广播一个通道的 48kHz 采样率，另一个是 24kHz，小的通道 24kHz 仍然会被放置在大的 48kHz 数据包的 BIS Subevent 中，这会浪费 airtime。如果这种方式导致问题，替换的方式是在单独的 BIG 中传输数据包。这种用例下，意

意味着一个 48kHz 采样数据包 BIG 和另一个 24kHz 采样数据包 BIG。缺点是，这种方式增加了广播的复杂度以及让其数量加倍，因为每个都需要自己的广播集合。实现需要确定哪种方式对于它们的具体应用最好。

#### 4.4.2 Robustness in a BIS

BIS 参数的定义方式有点不同，由于没有应答/确认，我们需要一些不同的策略，以使得发送能够最大限度地被接收。

BIS 仍然具有一个 Burst Number(BN)和一个 Number of Subevent(NSE)，和 CIS 定义的方式一样。为了对于缺乏应答/确认进行补偿，Core 在 Isochronous Interval 中引入了一个 Group Count 的概念，它是一个 NSE 和 Burst Number 的比例值(NSE/BN)。Group Count 用来确定 BIS 中安排的重传方式，将它们安排成 Group，这些组用来将差异化添加进传输策略中。(这些 Group 与 Broadcast Isochronous Group 没有关系——使用同一个词表示两个完全不同的概念。)

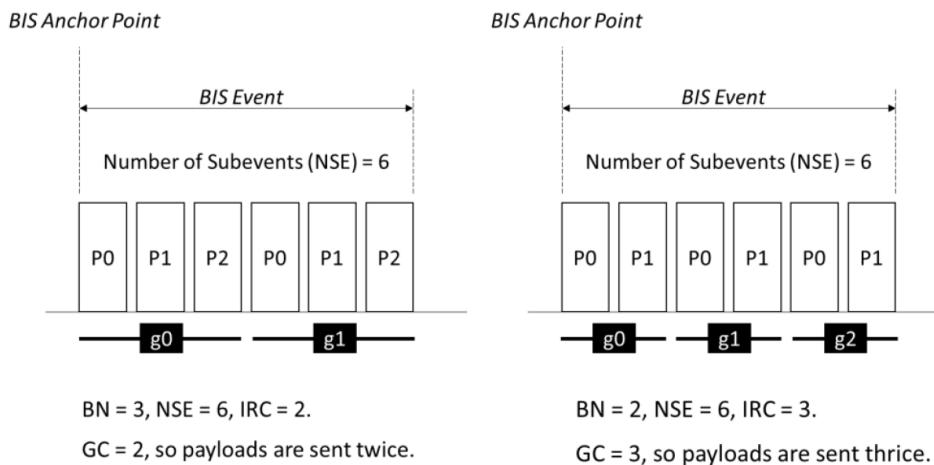


图 4.26 Group Count 作用

图 4.26 展示了两个 BIS 的实例，每个都包含 6 个 Subevent，所以 NSE=6。在第一个用例中，我们将 Burst Number 设置成 3。意味着对于每个 BIS 有 3 个 payload 可用，并且发送两次。在右边的第二个实例中，有相同的 Subevent 个数，但是 Burst Number 是 2，所以我们对于两个 payload 有三次重传。每个单独 Subevent 的 group number 可以范围从 0 到(Group Count - 1)。所以，在左手边的图中，Group Count 为 2，由 group0 和 group1 组成。在右手边的实例中，有三个 group: group0, group1, group2。

一个重要的事情需要注意，尽管两个实例看起来是相同的，但它们有不

---

同的 Isochronous Interval。假设 frame 是 10ms，由于左边的实例包含两个 payload——P0 和 P1，它的 Isochronous Interval 是 20ms。右边实例的 Isochronous Interval 是 30ms，因为它有 3 个 payload，P0，P1，P2。

Group Count 和另外的两个新的参数共同起作用：

- Immediate Repetition Count (IRC)，定义了携带与当前事件相关数据的 Group 的数量，以及
- Pre-Transmission Offset (PTO)。Pre-transmission 的概念是为了允许数据包尽早传输，期望接收设备可以较早收到音频数据包，允许其降低功耗，并且在最后的传输机会出现时将音频数据呈现出来。

这些参数，是由 Controller 中的 scheduler 计算出来的，替代 Flush Timeout。对于 CIS，Flush Timeout 本质上是一个 end-stop，终止一个 PDU 的传输。大多数情况下，它是不需要的，因为只要 Initiator 收到它的数据已经收到的确认之后，它就可以关闭 event 并且停止传输了。Broadcaster 不能这样——它必须一直保持传输。因此，它的优势是最大化数据包传输的差异化，使每个 Acceptor 有最好的找到数据包的机会。它们越快地完成，就可以在下一个数据包来之前越快地进入睡眠。

Pre-Transmission Offset 通过引入与 current event 相关的 Subevent，同时引入与 future event 相关的 Subevent 来起作用。这种对于 PDU 的 Subevent 的分配比起 CIS 的策略来说更加的复杂，所以解释它的最好方式就是通过一系列的实例来展示改变这些值产生的效果。所有的这些都是由 Controller 计算出来的，并且应用程序是接触不到的，但是在你设定 HCI 参数去配置你的 stream 时，可以帮助你理解它。

为了展示 future Subevent 的概念，图 4.27 演示了具有 NSE 为 8 的 Subevent，前五个与 current BIS event 相关，后面四个用于来自于 future BIS event 的数据。

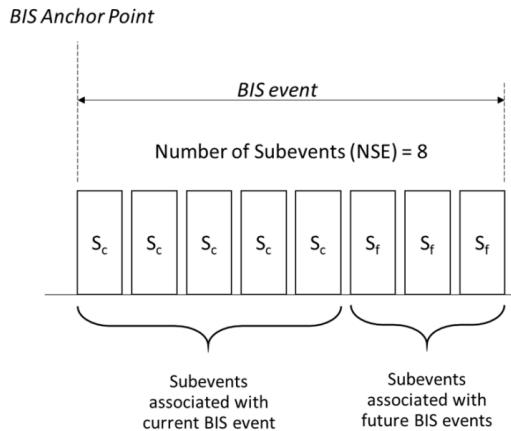


图 4.27 Future BIS event 概念

现在我们可以用更多的实例将所有东西结合起来。

这些我们首次在图 4.26 中看见、与 Immediate Repetition Count (IRC)一起的 group number(g)，决定了按照以下规则，哪些 payload 会在每次的传输时隙中被发送：

- 如果  $g < \text{IRC}$ , group g 应该包含与 current BIS event 相关的数据。
- 如果  $g \geq \text{IRC}$ , group g 应该包含与 future BIS event 相关的数据，此 future BIS event 是在 current BIS event 的  $\text{PTO} \times (g - \text{IRC} + 1)$  BIS event 之后。

我们现在来看一下不同参数导致不同重传机制差异的实例。

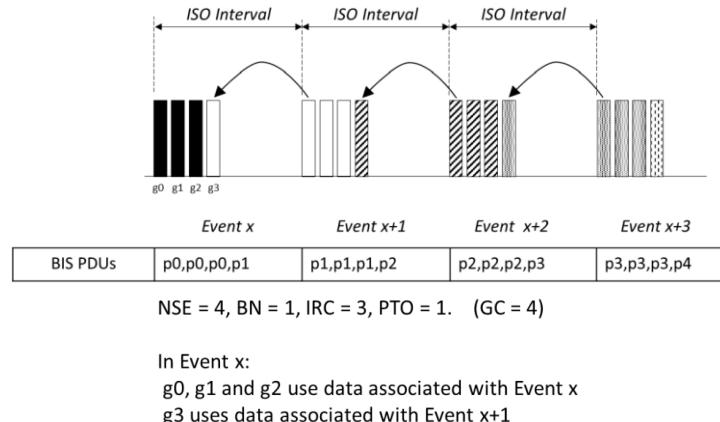


图 4.28 NSE=4, BN=1, IRC=3 以及 PTO=1 的 Pretransmissions

图 4.28 中，我们设置 NSE 为 4，即 4 个 Subevent。Burst Number 为 1，意味着每个 BIS Event 里面提供一个新的 payload(此实例的 Isochronous Interval 为 10ms)。Immediate Repeat Count 是 3，意味着与每个 event 相关的数据会在前三个时隙中传输。上述所示条件表明 BIS event 中最后的传输

来自于 next BIS event。使用此策略，每个 BIS 中，总是有 3 个 current 数据的传输加上一个来自于 next event 的数据传输。

实例中，我们似乎发明了时间旅行，但是我们并没有。我们只是稍微改变了我们的定义。只有当 p0 和 p1 PDU 都可使用的时候，Event x 才能发生。所以，p1 在 Event x 中是真正的 current data。这就显示了一旦 PTO 大于 0，latency 就开始增加，因为 Sync reference point 只有在数据最后的可能传输之后出现，对于 p0 来说，是在 Event x+1 中。

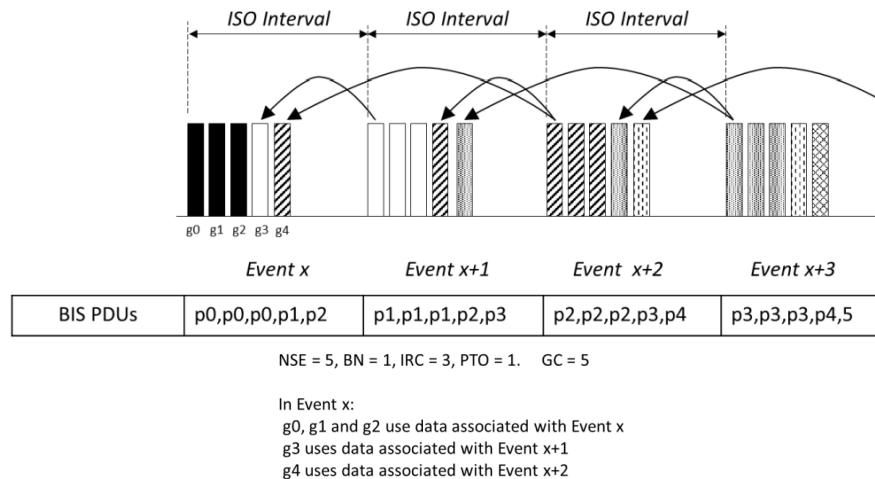
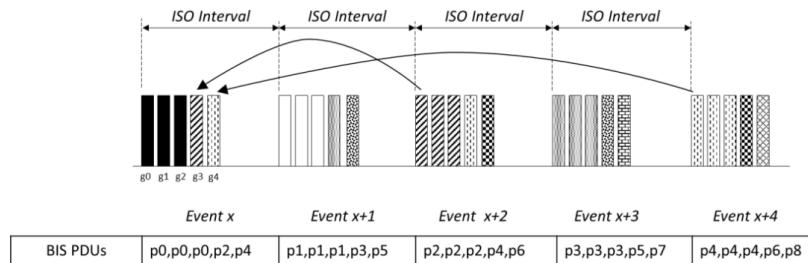


图 4.29 NSE=5, BN=1, IRC=3 以及 PTO=1 的 Pretransmissions

图 4.29 展示了当我们把 Subevent 增加到 5 会发生什么。这里，IRC 仍然保持为 3，但是 NSE 为 5，就会向来自 future BIS event 的数据提供两个 Subevent。它们用于来自 event x+1 和 event x+2 的数据包的 pre-transmit。意味着传输已经扩展到更多的 Isochronous Interval 了，传输的数据来自于 3 个连续的 BIS Event。它提高了用于抵抗 wide-band interference bursts 的健壮性，可持续多于一个的 Isochronous Interval，但是代价是更多的 latency，latency 增加了另一个 10ms。



NSE = 5, BN = 1, IRC = 3, PTO = 2. (GC = 5)

In Event x:  
g0, g1 and g2 use data associated with Event x  
g3 uses data associated with Event x+2  
g4 uses data associated with Event x+4

图 4.30 NSE=5, BN=1, IRC=3 以及 PTO=2 的 Pretransmissions

图 4.30 来看一下我们更进一步的扩展的情况，通过将 Pre-Transmission Offset 增加到 2。产生了来自于  $x+2$  BIS event 和  $x+4$  BIS event 的数据包含，实际上扩展音频数据传输到了 5 个 event 并且总共增加了 40ms 的 latency——比起 PTO=0 时的状况。此图中，我们只是展示了第一个 BIS event x 中的 group3 和 group4 的箭头，不然，这个图会变杂乱，并且不好阅读。

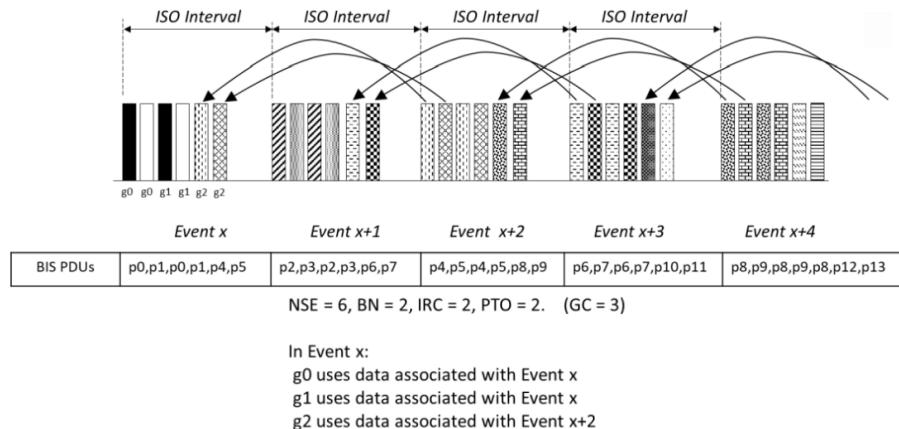


图 4.31 NSE=6, BN=2, IRC=2 以及 PTO=2 的 Pretransmissions

最后，在图 4.31 中，我们将 NSE 设置为 6，Burst Number 和 IRC 设置为 2，并且 Pre-Transmission Offset 也是 2。使用这些设置，每个 BIS event 包含两个数据包用于“current”event，传输两次，一个接一个，如我们在图 4.26 中所见，剩余最后的两个 Subevent 用于两个 future event 中的数据包传输。因为 BN 为 2，Isochronous Interval 就会是 20ms。因为 PTO 是 2，p4 和 p5 来自之后的量两个 Isochronous Interval，所以 latency 就会比图 4.28 中的简单的实例中大 50ms。

为了应付对于 latency 和 robustness 方面的需求，BIS 参数允许一些灵活的传输策略。下一章我们将看到在不同的广播情况下怎样使用它们。

总结，Flush Timeout 和 Pre-Transmission Offset 都会增加 latency。对于 CIS，Flush Timeout 帮助节省 Initiator 和 Acceptor 的电池，因为使用确认机制意味着 event 可以被关闭。对于 BIS，没有确认机制，广播发射器会在每一个 Subevent 发送。PTO 通过提供传输的多样化，给与 Acceptor 最好的接收数据包的机会，以此给与它们有效地节省电量。

---

#### 4.4.3 The Control Subevent

Control Subevent [Core, Vol 6, B, 4.4.6.7]不需要包含在每个 BIG event 中。通常，Control Subevent 用于诸如信道映射更新的项目中，因此，仅仅偶尔出现——当信息需要提供给所有接收广播音频流的设备时。使用 Control Subevent 的优点是设备不需要再去扫描以找到 basic BIG 信息，比如正在使用的跳频信道。这就减少了接收器不得不做、用于确保与 BIG 和 CIS 保持同步的工作量。没有 Control event，Broadcaster Sink 需要持续接收 Periodic Advertising train 并且周期性检查 BIGInfo 以此发现任何改变。

Control Subevent 是在六个连续的 BIG event 中传输的。可能会在随后的任何 BIG event 中传输，但是每次只能传输一个 control event。对于包含 Control Subevent 的 BIG event 中的 BIS Subevent，每个 BIS header 中的 Control Subevent Transmission Flag(CSTF)必须设置成 1，以此表示该 BIG event 中有 Control Subevent 的存在。它的 Control Subevent Sequence Number(CSSN)会告诉接收者，Control Subevents 是不是与已经收到的一样。Control event 使用与另外的 Subevent 一样的跳频机制，采用同一个 BIG event 中的第一个 BIS event 的索引。

#### 4.4.4 BIG synchronization

与 Connected Isochronous Stream 一样，单独的接收设备，通常如一副耳塞或助听器，不需要直到对方的存在。为了保持音频同步，它们需要使用包含在 BIG 中的信息，以此明确什么时候呈现音频。为了它们能够这样做，定义了 BIG Synchronisation Point，该点与音频数据传输的终点重合。因为我们没有 ack 在广播中，BIG Synchronisation Point 会精确的放在 BIG 中的最后的 BIS 的最后 BIS 传输的终点。正常情况下，它会和 BIG event 的终点重合。然而，对于有 Control Subevent 存在的时候，BIG synchronization Point 保持在 BIS Subevent 传输的终点，同时 BIG event 扩展到了 Control Subevent 的终点，如图 4.32 所示。

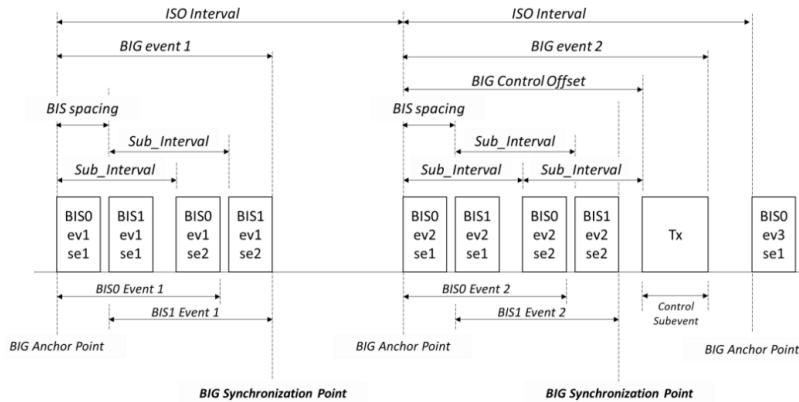


图 4.32 BIG 同步

在 BIG Synchronisation Point, BIG 中的每个监听 Broadcast Isochronous Streams 的设备都知晓每个其他的设备已收到它的数据, 因此, BIG Synchronisation Point 是它们可以应用 Presentation Delay 的固定时间点。这定义在协议栈的更高层并且规定音频需要呈现的点。此处重要的一点是, 音频不是在 BIG Synchronisation Point 呈现的——它是在 Presentation Delay 的终点呈现的, Presentation Delay 是起始于 BIG Synchronisation Point。由于广播只是由来自于 Broadcast Source 的传输组成, 从 Acceptor 返回的数据是没有 Presentation Delay 的概念的。

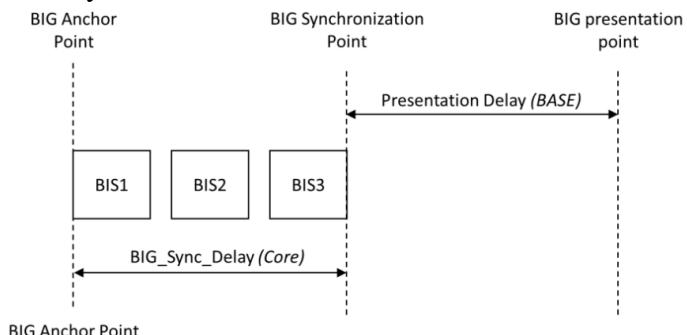


图 4.33 BIG 中 Presentation Delay 的使用

BIG Synchronisation Point 的推到对于广播来说也有点不同。不像 unicast, 每个 BIS 具有相同的时序参数——因为它们需要在 BIGInfo 中定义, 并且没有足够的空间允许单个 BIS 的不同设定。意味着 BIG Synchronisation Point 是从 BIG Anchor Point 开始的固定间隔, 它们都被 Acceptor 知晓, 因为它们在 BIGInfo 中。(相反地, 基于自己的 Anchor Point, 每个 CIS 使用 CIS\_Sync\_Delay, 因为它知道任何其他的 CIS 的相对时间, 并且不能假设和自己拥有的相同)

---

#### 4.4.5 HCI Commands for BISes

与 Connected Isochronous Streams 一样，Host 应用程序能够定义 SDU interval、最大的 SDU size 以及 Maximum Transport Latency，它是允许传输 BIS 数据 PDU 的最大时间。与 CIG 一样，取决于 Controller 中的 scheduler 使用它们去定义实际的 Link Layer 参数，比如，NSE、BN、IRC 以及 PTO。

建立 BIG 比起 CIS 简单的多，注意是因为与任何接收设备没有通信，所以只需要两个 HCI。LE\_Create\_BIG 配置和创建 BIG，里面有 BIS 数量 Num\_BIS，以及 LE\_Terminate\_BIG 命令结束和删除 BIG。没有增加和删除 BIS 的选项——所有事通过简单操作完成。两个命令都仅仅适用于 Initiator，传输一个或多个 BIG。

对于想要在 BIG 里接收一个或多个 BIS 的 Acceptor，有两个类似的命令，一个叫做与 BIS Synchronising 的过程。它们是 LE\_BIG\_Create\_Sync 命令和 LE\_BIG\_Terminate\_Sync 命令。但是在我到了那一步之前，我们需要看一下 Acceptor 怎样找到 Broadcaster 并且连接它的。

#### 4.4.6 Finding Broadcast Audio Streams

当我们回看 connected Isochronous Streams 时，我们讲不出设备怎样初始化连接的。原因是设备使用 Connected Isochronous Streams 按照每个其他的、使用普通的广播和扫描过程的 Bluetooth LE 设备完全一样的方式来连接。配对、绑定、发现彼此的功能并且继续建立 stream。如果它们是一个 Coordinated Set，CAP 程序会确保该 set 中的所有成员使用相同的过程。对于 broadcast Audio Stream，完全没有配对和协商过程，因为没有连接。反而，当有内容向它传输时，接收设备需要找到方法去发现正在传输内容，并且找到同步方式。

完成此事的方法叫做 Extended Advertising，是在 Core 5.1 添加进去的。先回到基础，图 4.34 展示了用于 Bluetooth LE 的信道。对于 Bluetooth LE，2.4GHz 频谱被划分成了 40 个信道，每个有 2MHz 宽。其中三个为保留的用于广播的固定频率——37、38 和 39 信道，是我们熟知的 Primary Advertising Channels。

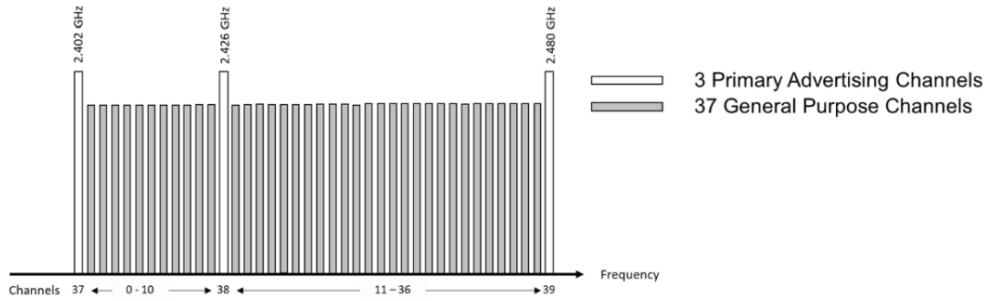


图 4.34 Bluetooth LE 信道

三个信道位置的选择是为了避开 Wi-Fi 频谱中最常用的部分。在三个广播信道之间有 37 个 general purpose channels，标记为 0 到 36。

如果其他设备准备可以接收它们的传输时，Broadcaster 需要提供相当数量的信息。它们不仅仅需要告诉接收者，就调频信道而言，这些传输位于哪里，而且也需要关于 BIG 和它的组成部分 BIS 的结构的所有细节，这两者我们在前面已经说过。许多情况下，在一个 Acceptor 的范围内很可能有多个 Broadcast transmitter，特别是当他们在公共场合应用时，因此多个 broadcast Isochronous Streams 可以从中选择。Acceptor 需要能够从中区分它们，意味着广播包含了关于每个广播流的信息。所有的这些需要 Broadcaster 传送大量的信息。想没有风险地承载它们对于三个 primary advertising channel 来说就太多了，如 37、38 和 39 信道。

为了克服此限制，Core 5.1 中的 Extended Advertising 策略，允许广播信息被移到 general-purpose 信道——0 到 36。为了达到此目的，Broadcaster 将 Auxiliary Pointer (AuxPtr) 包含进它的 primary advertisements 中，用于通知更多的广播信息可以在 general-purpose 信道(现在作为 Secondary Advertising channel 使用)中获得。Auxiliary Pointer 提供了扫描设备需要确定 secondary advertisement 所在的信息。此刻扫描设备不知道 AuxPtr 是否是用于广播。只有当它找到并读取 Extended Advertisement 时才明确。图 4.35 展示了此策略的基本结构。

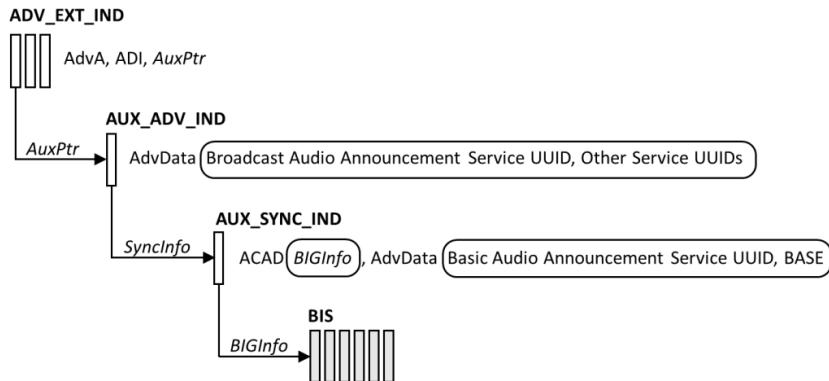


图 4.35 用于 BIG 的 Extended advertising

Extended Advertising 过程是通过使用三个广播 PDU 类型:

- **ADV\_EXT\_IND**: Extended Advertising Indications, 使用 primary advertising channel。每个 BIG 需要自己本身的 advertising set 和 ADV\_EXT\_IND。每个 ADV\_EXT\_IND 的 header 包括 advertising address (AdvA)、ADI (包含用于该 Extended Advertisement 的 Set ID)，以及 Auxiliary Pointer 指向:
- **AUX\_ADV\_IND**: Auxiliary Advertising Indications。它们在 37 个 general purpose 信道上面传输。包含了 BIG 的 Advertising Data 详情。AUX\_ADV\_IND 也包含 SyncInfo 域，用于向扫描设备提供怎样去定位:
- **AUX\_SYNC\_IND**: Auxiliary Synchronisation Information。这些广播包含两条重要的信息。第一条是 Additional Controller Advertising Data 域(ACAD)。接收设备的 Controller 需要此信息来描述 BIG 以及其组成部分 BIS 的结构。另一个是给接收设备的 Host 的信息，包含 AdvData 域。它包含 Basic Audio Announcement Service UUID 和 BASE，包含 BIG 中的 stream 的细节的定义，以用户可读的格式，涵盖了 codec 配置，和描述用例以及音频流的内容的 metadata。

ADV\_EXT\_IND 数据包中的 Auxiliary Pointer (AuxPtr)表明，一些或者全部的广播数据可以在 Extended Advertisement 中找到。然而，需要用来描述和定位广播流的信息量仍然大于通常适合于 Extended Advertisement 数据包的信息量。为了适应此情况，Extended Advertisement 本可以通过包含一个指向 Auxiliary AUX\_CHAIN\_IND PDU 的 AuxPtr 将后续的数据包串联起来，但是它没有这么做。取而代之，它建立 Periodic Advertising 序列，此序列在一个 AUX\_SYNC\_IND PDU 中包含了所有关于 BIG 的信息。Periodic

---

Advertising 序列中的 AUX\_SYNC\_IND 数据包是定期发送的，所有当扫描设备已经发现了它们，就可以跟踪它们，而且不需要回到 Primary 或者 Extended Advertisement。Broadcaster 甚至可以关闭 Extended Advertisement，但是保持 Periodic Advertising 序列运行。

Periodic Advertising 序列的存在是由 Extended Advertisement 的 AUX\_ADV\_IND PDU 中的 Syncinfo 域表示的，其提供了去哪里找到它的信息。如果 Periodic Advertising 序列的信息量太大，无法装入单个 PDU，AUX\_ADV\_IND 也可以使用一个 AuxPtr 指针指向一个 Auxiliary AUX\_CHAIN\_IND PDU。通常 Auxiliary AUX\_CHAIN\_IND PDU 是不需要的。

需要用来定位和接收广播音频流的数据起始于 Extended Advertisements。AUX\_ADV\_IND 包含了 Broadcast Audio Announcement Service UUID，它用来将 Extended Advertisement 标识为属于某特定的广播音频流，使用 3 个字节的 Broadcast\_ID 来标识它。Broadcast\_ID 在 BIG 的生命周期内保持不变。AUX\_EXT\_IND 可能包含来自于顶层 profile 的附加 Service UUID。这些都使扫描设备在很早就有机会过滤掉它发现的不同的广播，这样的话它就不需要与它不感兴趣的广播的 PA 同步，并且不需要将 PA 携带的信息解码。这个特性对于公共广播相当有用，公共广播定义在 Public Broadcast Profile (PBP)，其定义了 Public Broadcast Announcement UUID。

在扫描设备确定完该 BIG 是其想要的之后，它会与该 BIG 的 Periodic Advertising 序列同步。这些广播使用 general-purpose 信道 0 到 36。比如 Extended Advertisements，如果需要，它们可以使用另一个 AuxPtr，包含更多的数据。对于广播音频，它们包含两条至关重要的信息。

第一个是 Additional Controller Advertising Data 域，ACAD。ACAD 包含了广播数据结构。它们是包含有 Advertising Data (AD) type 以及相应的数据的数据结构。所有具有活跃着的广播的设备会使用 ACAD 来发送 BIGInfo 结构，该结构包含了接收设备需要的、用来侦测广播并理解 BIS 时序的所有信息。在 CIG 中此信息会被传递给 Acceptor 的 Link Layer，但是因为广播中没有连接，那是不可能的。因此这是一种替代方法。

图 4.36 展示了 BIGInfo 的结构。它起始于用于通知设备去哪里找到与此 AUX\_SYNC\_IND PDU 相关的 BIG 的“BIG\_Offset”。接下来，BIGInfo 描述了此 BIG 的结构：

- The fundamental spacing——ISO\_Interval、Sub\_Interval 以及 BIS

---

Spacing。

- The number of BISes in the BIG (Num\_BIS) 以及它们的特性——NSE、Burst Number、PTO、IRC 以及 Framing。
- The packet information——Max\_PDU、SDU\_Interval、Max\_SDU 以及 bisPayloadCount。
- The physical channel access information, 以 SeedAccessAddress、BaseCRCInit、Channel Map (ChM) 以及 PHY 的形式, 此外还有
- Group Initialization Vector (GIV) 和 Group Session Key Derivation (GSKD) 允许解码加密广播流。如果最后两个域存在, 扫描设备知道广播是加密的, 意味着它需要获取 Broadcast\_Code(下面讲到)用于解密音频流。

BIGInfo				
BIG_Offset (14 bits)	BIG_Offset_Units (1 bit)	ISO_Interval (12 bits)	Num_BIS (5 bits)	NSE (5 bits)
BIGInfo (continued)				
BN (3 bits)	Sub_Interval (20 bits)	PTO (4 bits)	BIS Spacing (20 bits)	IRC (4 bits)
BIGInfo (continued)				
Max_PDU (8 bits)	RFU (8 bits)	SeedAccessAddress (32 bits)	SDU_Interval (20 bits)	Max_SDU (12 bits)
BIGInfo (continued)				
BaseCRCInit (16 bits)	ChM (37 bits)	PHY (3 bits)	bisPayloadCount (39 bits)	Framing (1 bit)
BIGInfo (continued)		GIV (8 octets)	GSKD (16 octets)	

图 4.36 BIGInfo 结构

GIV 和 GSKD 的存在是了解 BIG 是否包含加密 BIS 的一种简易方式。如果它们不存在, BIGInfo 会更短, 表示 BIS 没有加密。

在我们立刻 BIGInfo 之前, 有件重要的事情需要重复一下, 所有的 BIS 都有完全相同的参数。BIG 中, 一种设置适用于每一个被使用的 BIS, 需要明确规定该设置, 以满足不同 BIS 的最大需求。

#### 4.4.7 Synchronising to a Broadcast Audio Stream

获取到了 BIGInfo 之后, 设备可以使用该信息去找到 BIS。BIG\_Offset, 与 BIG\_Offset\_Unit 一起告知接收者在 BIGInfo 数据包开始之后第一个 BIS

的 Anchor Point 将位于哪里(请记住，对于广播，没有 ACL 立即来提供时序，像 CIS 一样)。Acceptor 可以接收 BIG 中的任意数量的 BIS。在 Bluetooth LE Audio 中，这叫做与 BIG 或 BIS 同步。图 4.37 展示了怎样使用此信息。

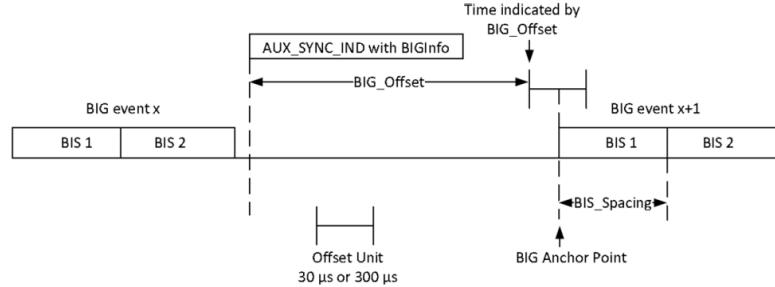


图 4.37 BIG 同步时序

在现实生活中大多数情况下，Broadcast Sink 不会想与所有的 BIS 同步。一个耳塞或者助听器通常只会同步到 BIG 中的左边或者右边 Audio Stream，而一副耳塞会想要同步到左边和右边通道。为了确定去连哪一个 BIS 或 BISes，扫描设备需要再次查看另一条包含在 Periodic Advertising AUX\_SYNC\_IND PDU 中的信息。这是 Broadcast Audio Source Endpoint 结构，它包含在跟随在 Basic Audio Announcement Service UUID 后面的每一个 AUX\_SYNC\_IND PDU 的 AdvData 域中。

#### 4.4.8 BASE – the Broadcast Audio Source Endpoint structure

为了理解 BASE，我们需要移动到上层的 Basic Audio Profile 中，它是我们在 Generic Audio Framework (GAF) 中遇到的第一个 profile。从配置和管理 Audio Stream 角度来看，对于 Connected Isochronous Streams 和 Broadcast Isochronous Streams，它都是关键的 profile。

BIGInfo 描述了 BIS 的结构，告诉设备它包含有多少 BIS 并且去哪里可以找到它们，但是那只是实际的信息。它不提供任何关于 broadcast Audio Stream 的内容方面的信息。BASE 提供细节，告诉设备每个 BIS 包含的音频信息以及它是怎样配置的。在范围内有多于一个的 Broadcaster，接收器能够去选择想听哪一个 broadcaster 时，这就很至关重要了。BASE 由三个等级的信息组成，如图 4.38 展示。我们会在查看怎样建立 Broadcast Stream 时，对其进行更详细的回顾。

最高层级——Level1，提供了对于 BIG 中每一个 BIS 都有效的信息——一个 Presentation Delay 以及这些 BIS 被划分成 Subgroup 的数量。这些 Subgroup 包含具有共同的特性的 BIS，这些特性可能是编码方式，或者

Audio Stream 的语言。

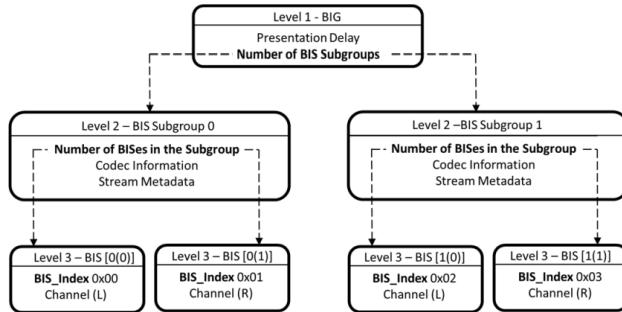


图 4.38 简化的 BASE 结构

每个 Subgroup 都会在 BASE 的 Level2 中进行更详细的描述，Subgroup 中的 BIS codec 信息，以及以 LTV 结构形式的音频流 metadata 会在 Level2 提供。其中大部分是人类可读形式的数据字符串，如节目信息，建议元数据至少包括 ProgramInfo LTV，以便能够显示它的扫描设备可以使用它来帮助用户在不同的音频流之间进行选择。

最后，BASE 的 Level3 提供了每个 BIS 的具体信息。包含了它的 BIS\_Index，用于标识它出现在 BIG 中的顺序，也包含 Channel\_Allocation，用于标识它代表的 Audio Location，比如 Left 或 Right。对于特定的 BIS，Level3 可以包含不同的 codec 信息集合，它会覆盖 Level2 的 Subgroup 的值。这最有可能用于 BIG 中的某个成员使用不同于其他 BIS 设置的场合。

BASE 中包含的信息允许 Broadcast Sink 决定它是否想接收 BIG 以及同步到此 BIG 中包含的哪个 BIS，除此之外，还告诉它在整个 BIG 中，特定的 BIS 所在的位置。一旦解析出了 BIGInfo 和 BASE，Broadcast Sink 就有了它同步到任何 BIS 所需要的所有信息。

大多数用例中，Broadcast Source 只会传输单个 BIG，但是它可以传输多个 BIG。这可能发生在公共信息广播中，不同类型的信息可能包含在不同的 BIG 中。举个例子，机场可能一个 BIG 播放航班通知，另一个用于一般安全通知，再一个，加密过的类型用于工作人员通知。这种情况下，每个 BIG 都有自己的私有广播设置，每个都有不同的 Advertising Set ID (SID)，伴随它拥有的、包含它的特定的 Broadcast\_ID、BIGInfo 和 BASE 的 extended advertising 序列(见图 4.39)。

SID 不应经常改变——通常是不变的，直到设备经过 power cycle，经常是整个生命周期都是不变的。Broadcast\_ID 在 BIG 的生命周期中是不变的。Broadcast Sinks 可以利用这些 ID 去重新连接到它们了解是否识别 SID 和

---

Broadcast\_ID 的 Broadcaster。

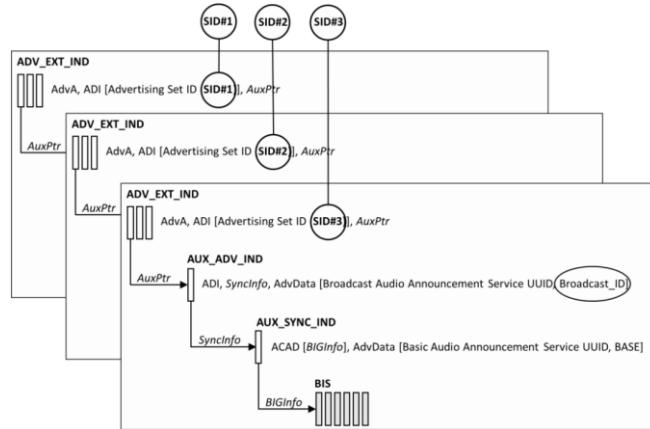


图 4.39 Multiple BIG 广播

#### 4.4.9 Helping to find broadcasts

可以预期，广播音频会以与单播音频非常不同的方式使用。助听器行业热衷于使用 Bluetooth LE Audio 广播来补充和扩展当前的电磁线圈设备，使安装更具成本效益。当前，大多数电磁线圈的使用是为了声音的再加强，特别是为了戴有助听器设备的人。将来，相同的广播传输可以被使用耳塞和耳机拾取，许多公共广播信息服务会被部署。Bluetooth LE Audio 很可能成为 TV 的标准，不仅仅在公共场所，比如健身房和酒吧，也会在家里。Bluetooth SIG's Audio Sharing program 正在推动广播成为咖啡馆和个人音乐分享的解决方案。如果市场照此发展，用户会经常发现自己在多个 Broadcast Sources 的范围内，意味着他们需要去选择要接收哪一个。第一级选择可以通过 AUX\_ADV\_IND 数据包中的 profile UUID 提供，但是设备仍然需要检查并解析每个 BIG 的 BASE 信息，以此来做决定。(早期，可能选择你发现的第一个 BIG，然后按键，移动到下一个，但是，长远来看，这不是一个可衡量的用户体验)。这给产品设计人员提出了一个问题，诸如耳塞之类的设备是没有空间去放置显示屏的，通常也只有很少的空间放置按键。加之，扫描过程相对耗电——持续的扫描会影响耳塞或助听器电池的使用寿命。

为了绕过此限制，Bluetooth LE Audio 规范引入了 Commander 的概念——执行扫描操作的角色，允许用户选择广播并且指导接收设备同步到选择的 BIS 或 BISes。Commander 角色可以作为移动电话、智能手表、耳塞盒或专用遥控器中 APP 的一部分来实现。实际上任何与 Broadcast Sink 有连接的

---

Bluetooth LE 设备都可以作为 Commander。实现 Commander 角色的设备叫做 Broadcast Assistant，并且定义在 Broadcast Audio Scanning Service (BASS)中。他们可以集成到像 TV 这类设备中，以帮助实现耳塞和耳机的自动化连接。

#### 4.4.10 Periodic Advertising Synchronisation Transfer – PAST

Broadcast Assistant 完全按照与其他扫描设备一样的方式扫描暗示 Extended Advertisements 存在的广播。当它发现了这些广播，Broadcast Assistant 可以同步到相应的、包含 Broadcast Audio Announcement Service UUID 的 Periodic Advertising 序列。它可以在读取和解析所找到的 BASE metadata 之前对其扫描应用 filters，之后通常通过显示人类可读的 ProgramInfo，向用户提供可用广播流列表。

一旦用户选择完，Broadcast Assistant 就会使用 PAST 程序，以此向 Broadcaster Receiver 提供它需要的、用于找到相关 Periodic Advertising 序列的信息。这种做法允许 Broadcaster Receiver 直接跳过这些广播包，获取 BIGInfo 以及 BASE，并且同步到恰当的 BIS，而不需要因扫描而消耗掉电量。此过程就叫做 Periodic Advertising Synchronization Transfer 或者 PAST。

Broadcast Assistant 向用户提供的信息的级别完全取决于实现。手机 APP 可以列出范围内的每个 Broadcaster，它可以在用户喜好的基础上，限制显示信息的数量，或使用从一副耳塞中读取的提前配好的设置。Broadcast Assistant 相当于手表或者健身手环上的按键，用来选择其发现的 Broadcast Sources 列表中的最后一个同步音频流。Broadcast Assistants 也可以包括用于获取需要的 Broadcast\_Code 解密私密、加密过的广播的应用，或者与 Broadcaster 创建连接或是 proxy，或通过使用 out of band(OOB)方式。

#### 4.4.11 Broadcast\_Code

Encrypted streams 在 Bluetooth 技术中不是新的技术，安全和保密性一直是此规范的重要部分。然而，在传输和接收设备之间没有连接时(就是 Bluetooth LE Audio 中广播流的用例)实现加密，带来了新的问题，就是怎样处理加密。

许多 broadcast streams 没有被加密，它们会开放式广播，这样任何人都可以收到。然而，这会导致一些问题。首先，范围内的任何人都可以接收。因为 Bluetooth 技术有很好的穿墙效果，在会议室、酒店房间、甚至在家里，这会是一个问题，这些地方的临近房间里的人可以不经意的收到你电视的音

---

频。这可能会让人恼火，也可能让人尴尬，具体取决于内容。商业环境很可能是机密的，因此增加加密功能至关重要。由于音频传输只能在感应线圈的范围内进行，因此在 telecoil 系统中，基本的保密性是固有的。为了在 Bluetooth LE Audio 广播中提供相同的认证等级，音频数据需要加密，这使得接收者需要获得解密密钥，它就叫做 Broadcast\_Code。

搜索广播的设备可以通过检查 periodic advertising 序列中的 BIGInfo 的长度来侦测 broadcast Audio Stream 是否加密过。如果此音频流是加密过的，数据包会包含附加的 24 字节，其包含了 Group Initialization Vector (GIV) 以及 Group Session Key Diversifier (GSKD)。扫描设备会侦测它们的存在，并且，和另外的 metadata 一起，决定是否要同步到该 stream，取决于它们是否可以取回 Broadcast\_Code。

尽管广播不需要 Broadcast Source 和 Sink 配对，在许多用例里，会有 ACL 连接的存在。这听上去挺奇怪的，但是这种方式允许加密连接的远远超出了单播，而不会达到 airtime 的限制。预计这将是大多数家用电视的工作方式，这样邻居就听不到你在听什么了。在此种用例中，用户会使用 BASS 的特性去获取 Broadcast\_Code，与电视配对。Broadcast\_Code 被 Host application 所有。当 Host 建立 BIS 时，将其提供给 Controller，同样地，可以提供给信任的设备。在公共场所，会议室和酒店，最有可能使用 out-of-band 方式，可能类似于用于连接 WiFi 接入点时大约出来的信息。Broadcast\_Codes 也可以通过其他 out of band 方式获取，比如扫描二维码，轻轻敲击一个 NFC 终端，甚至包含在一个可下载的戏票上。我们会在第 12 章更详细地探索这些配置。

注意，广播数据包中用于描述广播的 metadata 是不加密的。因此，需要确保其不包含机密的或可能令人尴尬的信息。

#### 4.4.12 Broadcast topologies

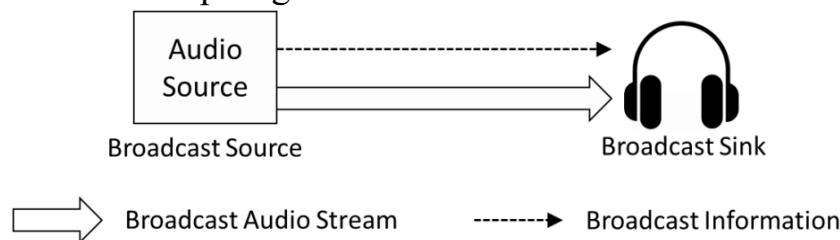


图 4.40 Direct synchronization from headphones

Bluetooth LE Audio 所包含的特性为找到广播提供了许多选择。我们会

---

在后续的章节来检视一下，但是，下面的图展示了一些常见的类型。大多数用例中，他们展示了单个“Broadcast Information” stream，其包含了所有的广播信息。

最简单的用例，耳机自己去扫描、找到并且选择广播音频流，如图 4.40 所示。

图 4.41 基本相同，但是，指出了 Broadcast Sink 也可以是电话。此处，它可以扫描 Broadcaster，向用户显示可供选择的广播音频流，并且将音频推送给有线耳机。尽管不像是有效地使用了 airtime，但这也等同于使用 Bluetooth 传输音频流(Bluetooth Classic Audio 或者 Bluetooth LE Audio)。

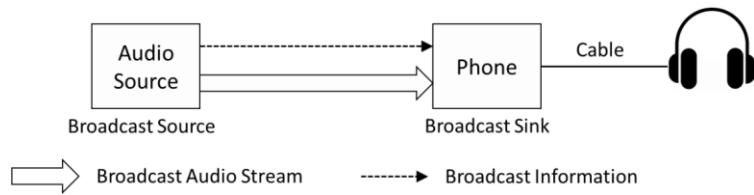


图 4.41 将手机当作 Broadcast Sink 使用

图 4.42 展示了期望的最常用的用例，诸如手机、手表或遥控器作为 Broadcast Assistance 去扫描并向用户显示可选的广播音频流，然后使用 PAST 使用户和耳机或耳塞可以连接到(同步到)选择的广播流。我们也会看到，Broadcast Assistant 也可以用于 volume control 和 mute，允许用户去发现、选择以及控制广播音频流的呈现。如图 4.43 所展示。

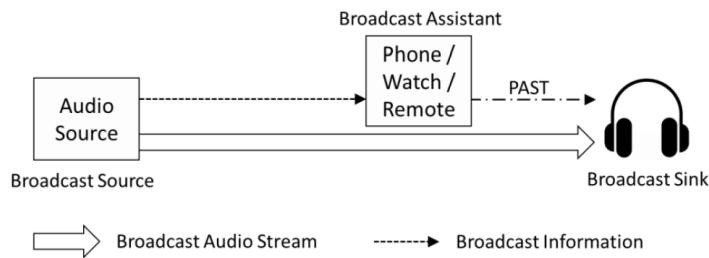


图 4.42 使用 Broadcast Assistance 以及 PAST

尽管在 Broadcast Source 和 Broadcast Sink 之间，对于音频流是没有连接的，设备还是可以使用 ACL 连接去帮助同步到 BIS，这样的话，当你进屋时，电视可以自动地同步到音频流。此种用例中，Broadcast Source 通常也会包含 Broadcast Assistant，可以是已经配对过的用户的耳机。当用户选择连接 Broadcast Assistant，它会使用 PAST 去提供怎样同步到 BIS 以及 BASS 的细节(传输 Broadcast\_Code)。

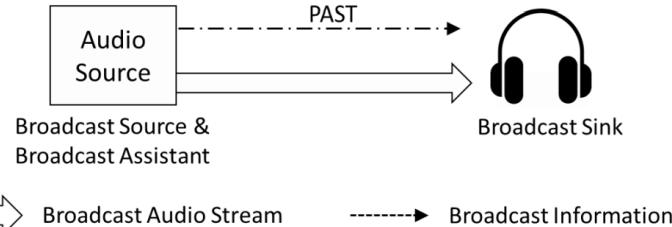


图 4.43 Broadcast Source 和 Broadcast Assistance 组合使用

这是组成 Audio Sharing 的拓扑之一，许多朋友可以分享一个手机的音乐。第 12 章会更加详细地解释这些选项。

## 4.5 ISOAL—The Isochronous Adaptation Layer

ISOAL [Core Vol 6, Part G]是 Core Isochronous Streams feature 的最复杂层面之一。好消息是，它是由 Bluetooth 芯片为你负责管理的，但认识其必要性以及它的功能是很有用的。ISOAL 用于广播和单播音频流。

ISOAL 的存在是为了解决 frame sizes 之间匹配的问题。最常见的原因是当 Acceptor 只支持 10ms frame sizes，但因为 Initiator 正与其他 Bluetooth Classic 设备有连接(比如旧鼠标和键盘，只能运行在 7.5ms 时序间隔上)，需要使用 7.5ms 的 frame。随着新的外围设备变得更加灵活，这种情况会随着时间的推移而改变，但在此之前，ISOAL 提供了一种方法来适应它们，同时整个 Bluetooth 生态系统迁移到 10ms 的定时间隔。

ISOAL layer 位于 codec 和 Link layer 之间的数据通道。如果 codec 是在 Host 中实现的，来自 codec 的 Encoded SDU 可能通过 HCI，或者通过专有接口(不管 codec 位于哪里)传送。如图 4.44 展示，ISOAL 提供了 fragmentation 和 recombination，或者 segmentation 和 reassembly，负责发送 framed 或 unframed PDU 中的编码音频数据。

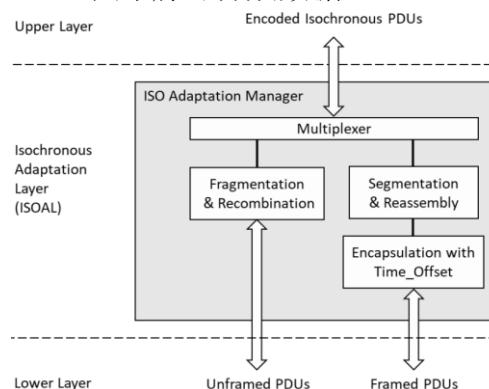


图 4.44 Isochronous Adaptation Layer(ISOAL)架构

---

根据 Host layer 提供的设置，Controller 可以决定是否使用 framed 或者 unframed PDU。对于 unframed PDU，如果 SDU 可以放进单个 PDU，Isochronous Adaptation Manager 可能将它们分段，但是发送时不带 segmentation header。这是最高效的、最低延迟传输 Isochronous 数据的方式。如果 SDU 大于 Maximum PDU size，它会被分片并且在多个 unframed SDU 中发送。Recombination 或 reassembly 程序将它们重新组装成 SDU。Unframed PDU 只能在 ISO\_Interval 等于或者是 SDU\_Interval 的整数倍数时使用，SDU\_Interval 本身等于或是采样 frame 的整数倍数。意味着 SDU 的生成需要与 transport timing 同步，这样它们就不会相互漂移。反之，你需要使用 framed SDU。

对于 framed SDU，Isochronous Adaptation Manager 增加了 segmentation header 和可选的 Time\_Offset。Time\_Offset 允许跨越 PDU 分割多个 SDU，提供了一个参考时间，以保持 SDU 生成和传输时序之间的关联。如果你想要更多的细节，全部的 ISOAL 规范包含在 Core 中的 Vol 6, Part G, Section 6。

设计者需要了解的 ISOAL 的专有方面是它包含一系列定义 Transport\_Delay 的等式。Transport\_Delay 是 SDU 准备发送和在适当的 Synchronisation Reference 处准备解码之间的时间。

对于 framed SDU，CIG 传输时延：

$$\text{Transport_Latency} = \text{CIG_Sync_Delay} + \text{FT} \times \text{ISO_Interval} + \text{SDU_Interval}$$

对于 Central 到 Peripheral 和 Peripheral 到 Central 方向，需要使用 CIG\_Sync\_Delay、FT 和 SDU\_Interval 各个值分别进行计算。

对于使用 framed SDU 的 BIG

$$\text{Transport_Latency} = \text{BIG_Sync_Delay} + (\text{PTO} \times (\text{NSE} \div \text{BN-IRC})) \times \text{ISO_Interval} + \text{ISO_Interval} + \text{SDU_Interval}$$

对于 unframed SDU，计算稍有不同。对于 CIG：

$$\text{Transport_Latency} = \text{CIG_Sync_Delay} + \text{FT} \times \text{ISO_Interval} - \text{SDU_Interval}$$

再一次提醒，对于 Central 到 Peripheral 和 Peripheral 到 Central，你需要使用 CIG\_Sync\_Delay、FT 和 SDU\_Interval 各个值分别进行计算。

对于 unframed BIG，

$$\text{Transport_Latency} = \text{BIG_Sync_Delay} + (\text{PTO} \times (\text{NSE} \div \text{BN-IRC}) + 1) \times \text{ISO_Interval} - \text{SDU_Interval}$$

--oOo--

这就是 Isochronous Streams 的基本原理，现在我们需要来看一下 LC3 和 QoS 选择怎样影响健壮性和时延。

---

## 第5章 LC3, latency and QoS

### 5.1 Introduction

无线音频最具争议的两个方面，尤其是在发烧友中，是音频质量和延迟。在其早期，Bluetooth 技术经常因两者而受到批评，尽管在大多数情况下，音频质量可能更多地与呈现音频的转换器的状态有关，而与 Bluetooth 规范无关。

通过任何无线连接传输音频都需要有折中。干扰在生活中无处不在，这意味着音频数据会有一些丢失。除非你采取措施增加冗余，否则会导致音频中间有间断。通常，需要多次传输音频数据包，这样其中的一个数据包就有多个被接受到的机会。然而，要做到这一点，你必须能够压缩音频，以便有时间传输多个副本。这是使用 **codec** 完成的(这个是 **coder** 和 **decoder** 的组合词)。

**coder** 接收模拟信号，将其数字化并压缩数字数据，以便在比传输原始样本长度更短的时间内传输。这意味着可以在下一个样本采集之前多次传输。如果第一次传输丢失或损坏，则可以在其位置接着重传。接收设备中的 **decoder** 对接收到的数据进行解码，将其扩展以重新生成原始音频信号。由于执行编码和解码需要时间，这导致原始信号和从 **decoder** 出来的重构信号之间有延迟。

音频 **codec** 是相对较新的发明。从 1860 年的留声机到无线电广播、黑胶唱片和磁带，最初的一百年音频传输都是使用原始音频信号。如果有天气的干扰，唱片或蜡筒上的划痕，或磁带上的拉伸，声音就会丢失或失真。随着 CD 的推出，这一点发生了变化，CD 是通过开发将模拟信号转换为数字信号的脉冲编码调制(PCM)而实现的。

PCM 的工作原理是以比我们能听到的频率更高的频率对音频信号进行采样(CD 每秒 44100 次)，将每个采样转换为数字值。解码以相反的方式执行此操作，使用数字到音频 **decoder** 恢复模拟信号。每个样本中的位数越多，输出音频就越接近原始输入。CD 和大多数音频 **codec** 使用 16 位样本。当采样率和每个样本的比特数足够高时，人的耳朵无法检测到差异。然而，纯 PCM 数字文件的文件大小很大，因为不涉及压缩。44.1kHz 和 16 位的采样每秒产生 800kbytes，因此单声道五分钟的歌曲大约为 26MB，立体声为 52MB。这就是为什么标准 CD 只能播放一小时左右的音乐。

---

由 Fraunhofer 研究所开发的 MP3 音频 codec 的问世改变了数字音乐的发行。相较于人类耳朵实际能听到的音频，它使用一种称为感知编码(有时也称为心理声学建模)的技术。这可能是一种高频声音，超出了大多数人能听到的范围，因此可以用较少的数据进行编码，或者是一个保留的音符，其中 encoder 可以指示你只需要重复前一个样本或对其差异进行编码。通过应用这些方法，可以显著减小数字化音频文件的大小。MP3 通常会将数字化音乐文件的大小减少 25% 至 95%，具体取决于内容。很少有听众担心这一过程中任何轻微的质量损失，他们觉得增加的便利性远远超过了对音乐的任何明显影响。

音乐文件大小的减少导致了 Napster 等音乐共享服务的建立和 MP3 播放器的出现。它还为流媒体服务和无线音频传输的发展打响了第一枪，因为文件大小的减少意味着有足够的时间重新传输压缩的音频数据包，从而有助于应对任何传输中断。

## 5.2 Codecs and latency

使用 codec 的一个缺点是它们增加了信号的延迟。这是原始模拟信号到达发射机和重构信号呈现在接收机之间的延迟。

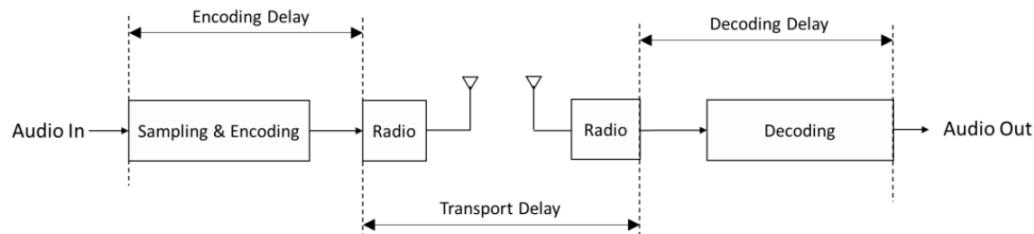


图 5.1 用音频传输中的 latency 元素

图 5.1 显示了构成 latency 的元素。首先，对音频进行采样。感知编码要求 codec 查看多个连续样本，因为压缩的很多机会来自于识别重复声音(或缺少声音)的周期。这意味着大多数 codec 需要捕获足够的连续样本，以获得足够的数据来表征这些变化。此采样周期称为 frame。不同的编码技术使用不同的帧长度，但它几乎总是固定的持续时间。如果它太短，有限的样本数量就会降低 codec 的效率，因为它没有足够的信息来应用感知编码技术，这会影响质量。另一方面，如果 frame 大小增加，质量会提高，但延迟会增加，因为 codec 必须等待更长时间才能收集每帧音频数据。

图 5.2 说明了这种权衡。它会因为 codec 的不同而有所不同，取决于它们如何执行压缩，但对于既可用于语音又可用于音乐的通用 codec，业界发

---

现，帧长度约为 10ms 是一个最佳选择，这在合理的延迟下提供了良好的质量。

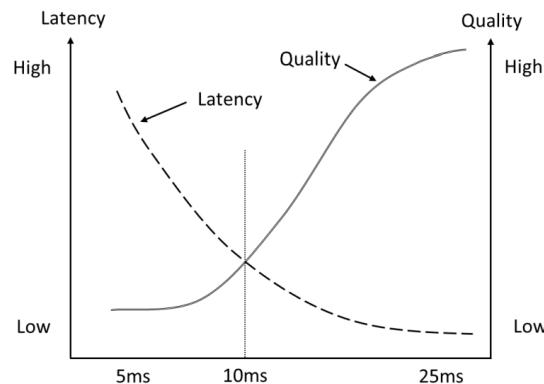


图 5.2 音频 codec frame 大小的最佳选择

还有一个权衡，就是运行 codec 所需的处理能力，这就是所谓的复杂性。当你试图从 codec 中挤出更高的音频质量时，你需要一个更快的处理器，但会缩短电池寿命。对于手机或 PC，这可能不是问题，但如果您正在对助听器或耳塞的麦克风输入进行编码，这就会是一个非常严重的问题。

回到图 5.1 和无线音频传输的一般性原理，一旦音频 frame 被编码，无线电就将其传输到接收设备。与编码相比，传输通常很快，但如果协议包含重传机会，则需要在开始解码之前允许这些重传。从第一次传输开始到最后一次接收传输结束之间的持续时间称为 Transport Delay，其范围可以从几毫秒到几十毫秒。(您可以在收到第一个数据包后立即开始解码，但如果这样做，则需要对其进行缓冲。这是因为需要重建输出音频流，使其没有间隙，因此，必须将其延迟，直到每一次重新传输的机会都已过，以应对数据包需要最大次数的重新传输才能获得解码的情况。否则，提前到达的数据包将提前呈现出来，而其他数据包则不会。)

最后，在接收到编码音频数据之后，需要对其进行解码，然后将其转换回模拟形式以进行呈现(播放)。解码通常比编码更快，并且没有 frame delay，因为 codec 自动扩展输出 frame。它通常比编码使用的功率要小得多，因为大多数 codec 都是为文件在产生时被编码一次，然后被解码多次的用例而设计的(比如当你从 central server 播放音乐时)，所以设计中存在着固有的不对称性。

### 5.3 Classic Bluetooth codecs – their strengths and limitations

如图 5.3 所示，现有的 Bluetooth audio profiles 都是根据各自使用情况

---

的特定要求开发的，并针对每种情况优化了不同的 codec。最初的 HFP 规范设计为使用 CVSD(连续可变斜率增量调制)编码方法，这是一种低延迟 codec，广泛用于电话应用。

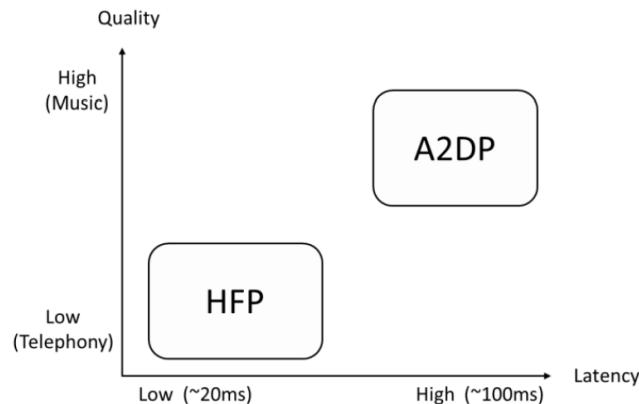


图 5.3 HFP 和 A2DP 的性能

CVSD 是最早的数字化和压缩语音的方法之一。快速采样——通常每秒 64000 个采样，但只捕获当前采样和前一个采样之间的差异。这意味着它是 frameless 的，并且具有相对短的采样和编码延迟。类似地，可以快速执行输出解码。权衡的结果是质量有限，而且由于没有压缩，它实际上是实时的，**没有重传的机会**。

HFP 的后续版本包括 mSBC——A2DP 中规定的 SBC codec 的修改版本，以支持宽带语音。mSBC 实际上是 SBC 的精简版，对单个单声道 stream 具有有限的采样频率。作为 frame-based codec，它增加了延迟，导致特有的总延迟约为 30ms。这就是为什么 HFP 位于图 5.3 的低延迟、中低质量区域。

相反，A2DP 专为高品质音乐而设计。它强制使用 SBC(子频带编码)codec，这是一种 frame-based codec，具有一定的基本的心理声学建模。与原始音频流相比，它可以产生非常好的音频质量，接近有经验的听众可以检测到的极限。A2DP 规范还允许使用由外部公司或标准组开发的可选 codec，包括 AAC(苹果公司在其大多数 Bluetooth 产品中使用)、MP3 和 ATRAC，以及许多公司使用专有 codec 的选项。其中许多产品已经流行起来，其中最著名的是 Qualcomm 公司的 AptX 系列。几乎所有这些 codec 都有更长的延迟。

在图 5.3 中，A2DP 位于图的右上区域，延迟很长。这在一定程度上是由于**希望使用重传来尝试使音频更健壮**。尽管听众过去习惯于接受诸如唱片划痕之类的小故障，但他们似乎对音频流中偶尔出现的爆音或中断容忍度要

---

低得多。最简单的解决方法是增加更多的重传和缓冲，但这意味着无线音乐流通常具有 100 – 200 毫秒的延迟，即使你是从手机或计算机上的文件进行流传输。虽然 codec 没有涉及到这种延迟，但它的发生意味着 codec 设计者通常没有专注于改善延迟，除非它是针对游戏等特定应用程序的。

虽然 100 – 200ms 的延迟听起来有些过分，但对于大多数音乐应用程序来说，这不是问题。当流媒体播放音乐时，无论是来自音乐播放器还是互联网服务，用户都无法指出他们是否在实时收听。只要音乐流在他们按下“播放”按钮的一秒钟内开始，并且音乐流是连续的，没有烦人的干扰，用户就很高兴。然而，当音频是视频的配音时，他们可能会注意到 lip-synch(假唱)问题，因为看到某人说话和听到他们的声音之间的 200ms 延迟看起来是有问题的。电话和电视制造商可以通过延迟视频来补偿任何音频延迟来解决这一问题。A2DP profile 下的 Audio/Video Distribution Transport Protocol (AVDTP)包含 Delay Reporting 功能，允许音频源设备询问接收器音频路径中的延迟。知道了这一点，电视和手机可以延迟视频，从而使声音和图像同步。然而，许多电视和耳机用于音频或视频缓冲的内存有限，因此超过几百毫秒的延迟可能会有问题。

在用户可以听到 Bluetooth 音频和原始的环境声源的情况下，即使是短的音频延迟也会成为一个问题。这一点早已得到助听器行业的认可，用户可以在剧院或电影院通过感应线圈系统收听实况声音，但也可以听到周围的声音。同样的问题也发生在家里，一个看电视的家庭中有一些成员戴着支持无线传输的助听器，还有一些成员没有。目前用于这些应用的感应线圈是模拟的，因此几乎没有延迟。转换到 Bluetooth 就需要一个能够比 SBC 覆盖更多 Quality / Latency 频谱的 codec。

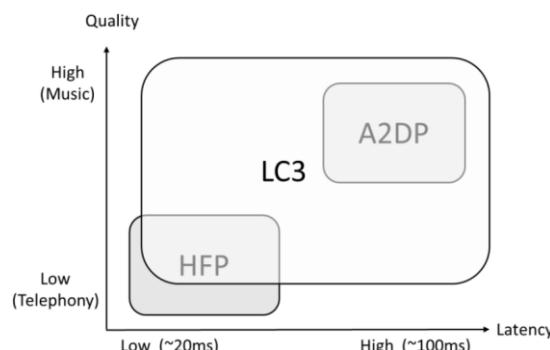


图 5.4 LC3——更高效的 codec

在 Bluetooth LE Audio 开发过程中，很明显，当前的 Bluetooth codec 将

---

难以满足要求。它们不仅在质量和延迟方面受到限制，而且 SBC 的效率也不如耳塞和助听器设计者所希望的那样高。它的复杂度相对较低，但占用了太多的 airtime，这对耳塞的电池寿命有很大影响。对于使用小型 zinc-air 电池的助听器来说，就是一个问题。这些电池对峰值电流和用于接收或发送的电流脉冲的长度都很敏感(Bluetooth 芯片接收通常会比发送消耗更多的电流)。如果超过这些电池的工作极限，它们的寿命可能会大幅缩短。为了解决这些限制，Bluetooth SIG 进行了 codec 寻找，导致了 LC3 的出现。

Bluetooth LE Audio 允许制造商使用其他 codec，但 LC3 对所有设备都是强制性的。这样做的原因是为了确保互操作性，因为每个 Audio Source 和每个 Audio Sink 都必须支持它。codec 的完整规范已发布，属于 Bluetooth RANDZ 许可证，因此任何人都可以编写自己的实现并将其纳入其 Bluetooth 产品，只要这些产品通过 Bluetooth Qualification 过程。考虑到它的质量，这是使用它的强大动力。

## 5.4 The LC3 codec

LC3 是当今最先进的音频 codec 之一，提供了巨大的灵活性，涵盖了从语音到高质量音频的所有内容。任何人都可以为 Bluetooth LE Audio 产品开发自己的 LC3 实现，尽管考虑到编写和优化 codec 的专业性质，很少有人会这样做。因此，我只提供一个关于它的作用和工作原理的概述。LC3 规范中有一个更详细的介绍，以及大约 200 页的规范细节，提供给任何想自己实现的人使用。

LC3 规范是迄今为止在单个 codec 中覆盖无线音频的全部音频质量和延迟要求方面最成功的尝试之一。它针对 10ms 的 frame 大小进行了优化，预计所有新应用程序，特别是公共广播应用程序，都将使用强制的 10ms frame。它还具有 7.5ms 的 frame 大小，以提供与以 7.5ms 间隔运行的 Bluetooth Classic Audio 应用程序的兼容性(对应于 EV-3 SCO 数据包)。它还支持扩展的 10.88ms frame，以提供传统的 44.1kHz 采样。对于需要支持 44.1kHz 的基于 7.5ms 的系统，这一时间缩短为 8.163ms。然而，这些是支持传统的或 Bluetooth Classic Audio / Bluetooth LE Audio 实现组合的独特的变体。

Bluetooth SIG 已委托独立测试实验室进行了广泛的音频质量测试，以量化 LC3 codec 的主观性能。这些结果表明，在所有采样率下，音频质量都超过了相同采样率下的 SBC，并在一半比特率下提供了等效或更好的音频质量。实际的好处是 LC3 编码分组的总大小大约是相同音频流的 SBC 分组

大小的一半。实现者可以利用这一优势，因为它减少了传输的总 airtime，节省了电池寿命，或者他们可以使用它来提高音频质量。这给了他们更多来使用参数的空间，特别是在耳机和助听器等功率受限的设备中。省电还可以为耳塞增加额外功能，如更先进的音频算法或生理传感器，同时保持较长的电池寿命。

Feature	Supported Range
Frame Duration	10ms (10.88@44.1kHz 采样) 7.5ms (8.163@44.1kHz 采样)
Supported Sampling Rate	8kHz, 16kHz, 24kHz, 32kHz, 44.1kHz 和 48kHz
Supported bitrates	每个 audio channel 的每个 frame 20 – 400 bytes。使用的 bitrates 由 Bluetooth LE Audio profiles 指定或推荐
Supported bits per audio sample	16, 24 和 32。(算法允许大多数中间值，但这些是推荐值。)
Number of audio channels	规范没有限制。实践中，受 profile、实现资源以及 airtime 的限制

表 5.1 LC3 特性

表 5.1 再现了 LC3 规范表 3.1 中的关键参数。

#### 5.4.1 The LC3 encoder

图 5.5 提供了 LC3 编码器的高阶视图。

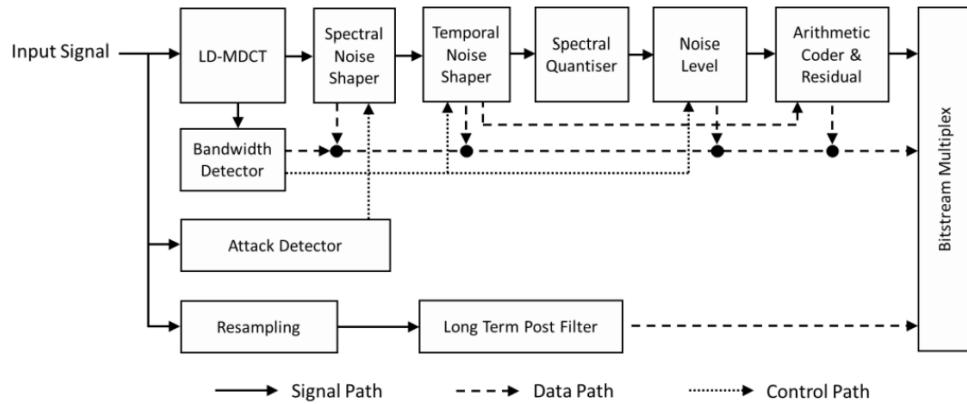


图 5.5 LC3 编码器高阶概览

Encoder 的第一个组成部分就是 Low Delay Modified Discrete Cosine Transform 模块(LD-MDCT)。LD-MDCT 是一种在感知音频编码中执行时间到频率转换的成熟方法。这是低延迟(因此是 LD)，但将音频输入样本转换为频谱系数，并将相应的能量值分组为频带仍然需要时间。这是 codec 延迟的一部分。

LD-MDCT 提供数据的模块之一是 Bandwidth Detector，它检测之前以不同编码率采样的输入音频信号。它可以检测语音通信中常用的语音带宽，即 NB(Narrow Band: 0-4kHz)、WB(Wide Band: 0-8kHz)、SSWB(Semi Super Wide Band: 0-12kHz)、SWB(Super Wide Band: 0-10kHz)和 FB(Full

---

Band: 0-20kHz)。如果检测到不匹配，它会向 Temporal Noise Shaper(TNS) 和 Noise Level 模块发出信号，以防止并避免任何噪声污染空的 upper spectrum。

LD-MDCT 产生的频率分量的主要路径是进入 Spectral Noise Shaper(SNS)，在那里对其进行量化和处理。SNS 的工作是通过对量化噪声进行整形，使最终的解码输出被人耳感知为尽可能接近原始信号，从而最大化感知音频质量。

编码器中的其余模块主要负责控制伪影。codec 最难以处理的声音是具有尖锐的声音，例如打击乐器。这些瞬变对 codec 来说是一件很难处理的事情，响板、钟琴和三角铁是用于评估 codec 性能的关键测试声音。尖锐瞬变的部分问题是，总体而言，这些声音的频谱相当平坦。Attack Detector 向 Spectral Noise Shaper 发送尖锐瞬变的存在信号，以便 SNS 可以将尖锐瞬变的存在通知 Temporal Noise Shaping(TNS)。然后，TNS 减少并潜在地消除具有严重瞬态的信号的伪影。

下一阶段是确定编码量化频谱所需的位数，这是 Spectral Quantizer 的工作。它可以被认为是自动增益控制的一种智能形式。它还计算出哪些系数可以量化为零，解码器可以将其解释为静音。此过程可能会引入一些编码伪影，这些伪影由 Noise Level 模块处理，使用伪随机噪声发生器来填补任何间隙，确保解码器的所有内容都设置为正确的电平。它还使用来自 Bandwidth Detector 的输入来确保编码信号被限制在活动信号区域。一旦完成，频谱系数被熵编码并复用到比特流中。

结果比特流的另一个分量是重采样输入。以 12.8kHz 的固定速率执行，再通过 Long Term Post Filter(LPTF)过滤。对于低比特率，此操作减少了包含音高或音调信息的任何 frame 中的编码噪声。Long Term Postfilter(LTPF) 模块通过在 decoder 一侧，控制 pitch-based postfilter 来感知地改变量化噪声。

编码的 LC3 frame 不包含任何时间信息，例如时间戳或序列号。这取决于系统使用 LC3 来控制数据包的定时，这是我们在查看 Core 时看到的。

#### 5.4.2 The LC3 decoder

解码器如图 5.6 所示，基本上是相反的过程。

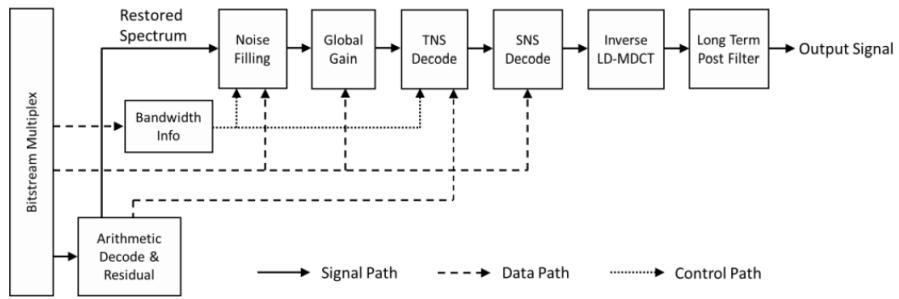


图 5.6 LC3 解码器概览

带宽信息用于确定哪些系数为零，而 Noise Filling 模块插入了带内系数的信息。Temporal Noise Shaper 和 Spectral Shaper 处理这些，然后 Inverse LD-MDCT 模块将它们转换回时域。然后应用 Long Term Post Filter，使用发送的音调信息来定义滤波器特性。

在解码每个 frame 的接收分组之前，如果 Controller 检测到有效载荷中的任何错误，则生成 Bad Frame Indication flag(BFI)，以及每个信道的有效载荷大小参数。如果设置了 BFI 标志，则解码器将跳过该分组，并发出信号，指示应该运行 Packet Loss Concealment(PLC)算法以替换输出音频流中丢失的数据。解码过程中检测到的任何错误也将触发 PLC。

#### 5.4.3 Choosing LC3 parameters

从 Bluetooth LE Audio 设计的角度来看，大多数开发人员最接近 LC3 规范的是他们用于在应用程序中配置 LC3 的参数。这些是表 5.1 特征的子集，如表 5.2 所示。

LC3 Parameter	Values
Sampling Rate	8kHz, 16kHz, 24kHz, 32kHz, 44.1kHz 和 48kHz
Bits per sample	16, 24 或 32
Frame Size	7.5ms 或 10ms(设置 7.5 或 10ms 时，44.1kHz 采样的实际大小自动生成。)
Byte per frame (payloads per channel)	20-40
Number of audio channels	通常是 1 或者 2，但是仅仅受 profile 或实现限制

表 5.2 LC3 配置参数

编码器和解码器的每个采样的比特是本地设置，并且可以不同。在大多数情况下，将它们设置为 16。

对于单播流，Acceptor 可以公开其支持的这些值的组合，以及特定用例中使用的配置的首选项。Initiator 从每个 Acceptor 公开的支持值中进行选择，以配置每个音频流。

为了将这些限制在合理数量的组合之内，BAP 为 LC3 codec 定义了十六种配置，以帮助提高互操作性。这些数据在下表 5.3 中重现，涵盖了 8kHz

至 48kHz 的采样频率。可以在 BAP 表 3.5(Unicast Server)、3.11(Unicast Client)、3.12(Broadcast Source)和 3.17((Broadcast Sink)中找到这些组合。

Codec Configuration Setting	Supported Sampling Frequency	Supported Frame Duration	Supported Octets per Codec Frame	Bitrate (kbps)
8_1	8	7.5ms	26	27.734
8_2	8	10ms	30	24
16_1	16	7.5ms	30	32
16_2	16	10ms	40	32
24_1	24	7.5ms	45	48
24_2	24	10ms	60	48
32_1	32	7.5ms	60	64
32_2	32	10ms	80	64
441_1	44.1	7.5ms	97	95.06
441_2	44.1	10ms	130	95.55
48_1	48	7.5ms	75	80
48_2	48	10ms	100	80
48_3	48	7.5ms	90	96
48_4	48	10ms	120	96
48_5	48	7.5ms	117	124.8
48_6	48	10ms	155	124

16\_2 BAP 在 Acceptors 和 Initiators 作为 unicast Audio Sinks 或 Audio Sources, 以及 Broadcast Sources 时的强制要求

24\_2 BAP 在 Acceptors 作为 unicast Audio Sinks 或 Broadcast Sink 时的强制要求

表 5.3 BAP Codec 配置设定

对于 Broadcast Source 不知道接收设备的 Broadcast Streams 的情况，它必须单方面决定将使用的 LC3 参数。BAP 目前规定，每个广播接收器必须能够解码 10ms LC3 frames，在 16kHz/40 字节 SDU 的编码，以及 24kHz/60 字节 SDU 的编码，因此使用这些值的 Broadcast Source 知道其音频流可以被每个 Bluetooth LE Audio 设备解码。

一些顶层音频 profile 要求支持接收更高质量的编码 LC3 音频流。TMAP 要求支持一系列采用 48kHz 采样、7.5ms 和 10ms frame 以及各种比特率的 codec 配置。然而，没有连接到所有接收设备的 Broadcast Source 不能知道这样的流是否可以被解码。我们将在本章稍后讨论广播设计的含义。

#### 5.4.4 Packet Loss Concealment (PLC)

无线传输的一个令人讨厌的特点是数据包丢失。Bluetooth 与 Wi-Fi、婴儿监视器和其他无线产品共享 2.4GHz 频谱，这通常会产生干扰。Bluetooth 规范是最可靠的无线电标准之一，它采用自适应跳频来避免任何干扰，但有时数据会丢失。如果音频源是同时使用 Wi-Fi 或其他 Bluetooth 外围设备的手机或 PC，也会有需要优先权的情况，这会导致错过 Bluetooth LE Audio 传输。稍后我们将研究减轻这些影响的方法，但事实是，偶尔会有一个数据包无法挽回地丢失。

不幸的是，丢失音频流中的数据包是非常明显的，这会让用户感到烦恼。为了试图隐藏它，已经发展了许多技术。插入静默的方式通常是令人不舒服

的，除非它之前恰好有一个沉默或非常安静的时刻。重复上一帧的方式可能有效，但如果连续丢失帧，则会变得明显。

为了提供更好的收听体验，该行业开发了一系列 Packet Loss Concealment 算法，这些算法试图通过预测丢失的音频最有可能是什么，来隐藏丢失的音频。这些算法在语音和音乐方面都非常有效，但如果丢失了具有或接近尖锐瞬变的片段，则很难隐藏。LC3 规范包括已开发用于匹配 LC3 codec 的 Packet Loss Concealment 算法。在当 Bad Frame Indication flag 发出丢失或损坏帧的信号时，或当解码器检测到内部位错误时使用。建议始终使用该算法或替代 PLC 算法。

## 5.5 LC3 latency

我们在本章开始时了解了 latency 的基础知识，但更详细地理解它很重要。

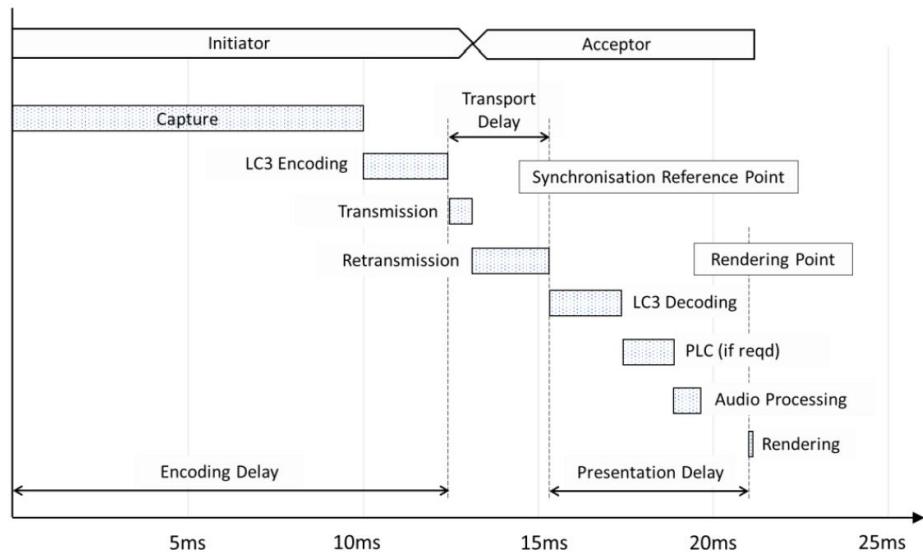


图 5.7 latency 的组成部分

图 5.7 说明了延迟的主要组成部分，显示了延迟是如何在 Initiator 和 Acceptor 之间建立的。任何 frame-based codec 都是从音频采样产生的延迟开始的。对于 10ms 的帧长度(LE Audio 中的标准 frame 长度)，这会导致前 10ms 的延迟。一旦对输入 audio frame 进行了采样，就可以开始编码，对于 LC3 来说，这大约需要 2.5ms，然后它才有一个完全编码的 SDU 向前传递以进行传输。

该图显示的是传输立即启动。如果 Receiver 在其第一次传输时获得有效

---

的 packet，则 Transport Delay 可以小于一毫秒，但如果已经使用了一次或多次重传，则在 Synchronisation Reference Point 接收到最后一次可能的传输之前，还需要几毫秒。和 Bluetooth LE Audio 出现的 Synchronisation Reference Point 最早出现的时间是距离该音频 frame 的第一个采样点大约 14ms，并且是每个 Acceptor 可以开始解码其接收的数据包的固定时间点。Synchronisation Reference Point 是 Presentation Delay 开始的地方。在此期间，Acceptor 需要对 LC3 分组进行解码，这对 LC3 解码器来说需要几毫秒，如果需要，则使用 Packet Loss Concealment，这需要几毫秒。虽然大多数数据包都不需要它，但运行算法的时间必须在需要它的场合进行分配。如果需要进行任何其他音频处理，例如用于噪声消除或语音增强的算法，则需要在 Presentation Delay 结束之前完成这些处理，Presentation Delay 结束点是每个 Acceptor 呈现重构音频流的地方。由于 Basic Audio Profile 要求每个 Acceptor 必须支持 40ms 的 Presentation Delay 值，因此它们需要支持大约 40ms 的缓冲，以在 Rendering Point 之前保存解码的音频数据。实际上，Presentation Delay 的值可能更低或更大——接收设备制造商可能支持低至 5ms，高达数百毫秒的范围。

如果一切都得到了优化，那么 10 毫秒 LC3 数据包的整个过程发生的最快速度是 20 毫秒多一点。使用 7.5ms 的 frame 几乎没有什么不同，因为较短的 frame 需要较长的 look-ahead 延迟，因此节省的时间仅为 1ms 左右。

20ms 的延迟相当于声音传播 7m 所需的时间。我们的听力已经进化，以应对这种程度的延迟。如果我们听到原始声音和 25-30ms 后的回声，大脑就会毫无困难地处理它。这意味着我们可以将 Bluetooth LE Audio Streams 用于耳塞和助听器，它们可以拾取环境声音以及 Bluetooth stream，而佩戴者不会被任何回声影响所分心。然而，上面的例子涉及很多优化。它假设一旦编码完成就可以开始传输，并且所有重传都发生在四分之一个 frame 内，这仅对小 packets 和较低采样频率有效。在大多数实际应用中，其他因素也会有影响，这就让我们来到了 Quality of Service 或 QoS。

## 5.6 Quality of Service (QoS)

Quality of Service 是一个应用于接收音频信号的术语，包括 **延迟、解码音频的感知声音** 以及 **任何音频 artefacts (如爆裂声、裂纹和间隙)** 的发生率。所有这些特性都有相互权衡的地方。上面描述的延迟示例是一个高度理想化的示例，即数据包在预期时间到达。实际上，它们没有。人体非常有效地吸

---

收 Bluetooth 信号，因此，如果某人戴着耳机，但将手机放在牛仔裤的后口袋中，手机和耳机之间的信号可能会衰减 80dB。如果你在房间里，这可能无关紧要，因为耳塞可能会从墙壁或天花板上拾取反射信号。但如果你在外面，没有反射表面，那么会丢失更多的数据包。

在上一章中，我们看到了 Isochronous Channels 的设计通过使用 Retransmissions、Pretransmissions、Burst Numbers、Flush Timeouts 和 frequency hopping 增加了鲁棒性。如果我们应用了足够多的这些特性，我们有非常高的信心，几乎每个数据包都能被接收到，而没有完整接收到的少量数据可以使用 PLC 来填充。然而，当我们应用这些技术时，latency 就开始增加了，功耗也会增加。在多个 Isochronous Interval 上增加重传会为每个 additional Isochronous Interval 的延迟增加额外的 frame 时间。在单个 frame 内进行更多的重传限制了可以容纳的不同流的数量，并提高了发送器和接收器的功率，接收器需要保持活动状态以寻找连续的传输时隙。

这些鲁棒性特征与 codec 设置是分开的。但 codec 设置也会影响鲁棒性。采用 48kHz 采样的更高质量编码将产生更大的数据包，这将更容易受到干扰。类似地，如果您将多个音频流编码为单个数据包，即，信道分配大于 1，则 SDU 将包含多个 codec frame，从而导致更大的数据包，具有相同的问题。

考虑到大量可能的 codec 和鲁棒性配置是允许的，BAP 定义了针对两个主要用例的标准组合集——Low Latency 和 High Reliability，这可以在 BAP 的表 5.2 和 6.4 中找到。它们覆盖 10ms 和 7.5ms frame 间隔。

Low latency 被解释为将允许所有重传在采样率为 8kHz 至 32kHz 的单个 Isochronous Interval 的设置。48kHz 的更大数据包扩展到两个 Isochronous Intervals，44.1kHz 的 Isochronous Intervals 增加到四个。**High reliability QoS 配置将 retransmission 优先于 latency**，允许将 retransmission 扩展到六个或更多广播 Isochronous Intervals，以及十个或更多单播 Isochronous Intervals。

表 5.4 展示了根据这些表得出的每个采样频率允许的最大等 Isochronous Intervals。Low Latency 48kHz 采样设置需要两个 Isochronous Intervals 的原因是允许使用较大的数据包进行足够数量的重传，因为只有有限的数量适合单个 Isochronous Interval。对于低采样频率，较小的数据包意味着每个 Isochronous Interval 适合更多的重传。分配多少重传最终取决于 Controller 中的调度器。

Sampling Frequency	Low Latency		High Reliability			
			Unicast		Broadcast	
	7.5ms	10ms	7.5ms	10ms	7.5ms	10ms
8 kHz	1	1	10	10	6	6
16 kHz	1	1	10	10	6	6
24 kHz	1	1	10	10	6	6
32 kHz	1	1	10	10	6	6
44.1 kHz	4	4	12	9	8	6
48 kHz (80 kbps)	2	2	10	10	7	7
48kHz(96/124kbps)	2	2	10	10	7	7
44.1 kHz 不同，因为它们是为 framed PDU 定义的。所有其他采样率均为 unframed。						

表 5.4 BAP QoS 设定允许的最大 Isochronous Intervals 数量

在图 5.7 的延迟示例中，我们看到使用具有 10ms frame 和显著优化程度的 LC3，总延迟仅仅超过 20ms。这使用了略超过 5ms 的 Presentation Delay。如果低延迟对应用程序很重要，那么实现者需要谨慎选择 QoS 和 codec 配置，因为它可能会导致延迟显著增加。表 5.5 展示了可能实现的实际总延迟。通过计算，LC3 使用 12.5ms 的值对 10ms frame 进行采样和编码。

Sampling Frequency	Low Latency			High Reliability	
	PD=10ms	PD=20ms	PD=40ms	PD=40ms	PD=40ms
8 kHz	32.5ms	42.5ms	62.5ms	147.5ms	112.5 ms
16 kHz	32.5ms	42.5ms	62.5ms	147.5ms	112.5 ms
24 kHz	32.5ms	42.5ms	62.5ms	147.5ms	112.5 ms
32 kHz	32.5ms	42.5ms	62.5ms	147.5ms	112.5 ms
44.1 kHz	53.5ms	63.5ms	83.5ms	137.5ms	112.5 ms
48 kHz (80 kbps)	42.5ms	52.5ms	72.5ms	147.5ms	117.5ms
48kHz(96/124kbps)	42.5ms	52.5ms	72.5ms	152.5ms	117.5ms

表 5.5 BAP QoS 设定典型的端对端延迟

表 5.5 中的值适用于单声道音频流。对于立体声应用，延迟会增加，正如我们将看到的。给实现者的信息是在选择 codec 和 QoS 设置时要小心。

BAP 表 5.2 和 6.4 中的 Quality of Service 建议包括 HCI 命令中用于设置单播或广播流的参数建议。这些建议(请记住，Controller 将其中一些建议用作指导，而非确定值)涵盖：

- SDU Interval (与 Frame Duration 相同，44.1kHz 除外)
- Framing requirements (除 44.1kHz 为框架外，其余均为 unframed)
- Maximum\_SDU\_Size (与 Supported\_Octets\_per\_Codec\_Frame 相同)
- Low Latency 和 High Reliability 选项的推荐 Retransmission Number (RTN)
- Low Latency 和 High Reliability 选项的 Max\_Transport\_Latency
- Presentation Delay，要求 40ms 在支持范围内。

使用这些表中的值可能并不总是产生预期的 latency。其中一个原因是 Presentation Delay 的值。BAP 要求所有 Audio Sinks 在其支持的值范围内包含 40ms 的 Presentation Delay，但不应将其视为默认值——这是表底部注释中所述的值——每个充当 Audio Sink 的 Acceptor 都必须支持该值。这是对

---

互操作性的妥协，以确保每个都有时间解码音频数据、应用 PLC 和任何附加处理。大多数设备都能做得更好。一些更高层的 profiles 要求更高的性能。TMAP 和 HAP 都要求 Acceptors 支持 20ms 的 Presentation Delay，但如果 Initiator 将其设置为 40ms，则会影响延迟。

回顾图 5.7，该示例中的 Presentation Delay 仅约为 5ms，导致总延迟低于 25ms。许多助听器将能够支持这一点，但它是高度优化的。在单播中，Initiator 和 Acceptor 相互交谈，当 Initiator 意识到其正在使用低延迟用例时，它们可以商定更低的 Presentation Delay。然而，基本 Broadcast Source 无法了解其周围 Broadcast Sinks 的功能，因此必须根据使用情况和市场上 Broadcast Sinks 的能力以及其设计者预期哪些 Broadcast Sinks 将访问的它，这些信息来做出决定。

如果一对 Acceptors 可以相互通信，则它们可以单方面行动，可能会根据流的 Context Type 做出更早呈现的决定。然而，这超出了 Bluetooth LE Audio 规范的范围。

回到表 5.5 中的“Low Latency”列，当采样频率高于 32 kHz 时，将其用于环境声音应用时，会产生回声效应。用于增强环境声音时，没有一种 High Reliability 设置是真正适合的，特别是对于广播。

在无法听到环境音频流的情况下(大多数流媒体音乐和电话应用程序的情况)，延迟问题要小得多，除非伴随有视频流，否则可能会导致 lip-synch 问题。然而，在这些情况下，视频应用程序通常管理音频流，因此可以调整相对时序以防止任何问题。

RTN——Retransmission Number，得益于进一步的解释。某些配置中的大值表明存在大量重传，但这并非必要。RTN 在 Core [Vol 4, Part E, Sect 7.8.97]中定义为在 acknowledged 或 flushed CIS Data PDU 之前，其从 Central 重新传输至 Peripheral 或 Peripheral 至 Central 的次数。对于 Broadcast，它只是 PDU 应重传的次数[Vol 4, Part E, Sect 7.8.103]。正如我们已经看到的，Host 不允许指定 FT、PTO、NSE 或 BN 的值，因为它不知道 Controller 可能有哪些其他限制。因此，RTN 只是帮助指导 Controller 调度算法的建议。

大多数时候，音频数据不会被传输 RTN 次。RTN 更好地解释为最大重传次数，而不是平均次数。如果 SDU 的 FT 大于 1，并且 SDU 被发送了最大次数(RTN+1)次，则接下来的 SDU 将只有 1 次发送机会，不可能进行重传。RTN 提供了获得更多传输时隙以适应短突发干扰的机会，但将其设置得过高可能会不利于后续 SDU。平均传输机会数始终为 NSE/BN。实践已经

---

表明，BAP 表 5.3 和 6.4 中给出的值已经过充分测试，通常应遵循。然而，如 BAP 表 6.5 所示，Controller 可以选择其他值。

回到 Isochronous Channel 设置，查看一下 Initiator 的 Host 请求的内容及其实际获得的内容非常有用。BAP 给出了一个示例，说明 Controller 如何根据不同的资源约束来解释它接收的 HCI 参数。为了理解这种影响，我们可以查看 48\_2\_2 广播音频流 QoS 配置的 High Reliability 设置。这在 BAP 的表 6-4 中定义，并在下面的表 5.6 中重现，其中要求最大传输延迟为 65ms。

	Max SDU (octets)	RTN	Max Transport Latency (ms)	Presentation Delay (μs)
48_2_2	100	4	65	40000

表 5.6 48\_2\_2 High Reliability 音频数据广播音频流配置设定

BAP 的表 6.5 为 Controller 提供了在收到包含表 5.6 值的 LE\_Set\_CIG\_parameters HCI 命令时使用的建议 Link Layer 参数。如表 5.7 所示，表 5.7 还显示了所产生的传输延迟和每组参数使用的可用 airtime 百分比。

Option	ISO_Interval (ms)	BN	NSE	IRC	PTO	Num_BIS	RTN	Max_Transport_Latency Mono/Stereo	Airtime Usage Mono/Stereo
1	30	3	9	2	1	1 或 2	2	65/71 ms	18/36%
2	10	1	5	1	1	1 或 2	4	43/46 ms	30/59%
3	20	2	8	2	1	1 或 2	5	65/70 ms	24/48%

表 5.7 48\_2\_2 广播音频流 QoS 配置的推荐 BAP LL 参数

表 5.7 中的三个选项是可能的，因为 Max\_Transport\_Latency 和 RTN 只是指导 Controller 制定调度时的建议。根据它正在做的其他事情，它通常需要考虑其他约束。表 5.7 中的三个选项是以下三种情况下的建议 Link Layer 设置：

- 选项 1，具有最低的 airtime，经过优化，可与 Wi-Fi 和其他以 7.5ms 时序运行的 Bluetooth 设备共存。使用更大的 Isochronous Interval 来承载多个 10ms 帧，为 Wi-Fi 操作提供了最大的间隙。如果 Broadcast Source 是电话或 PC，并且使用 Wi-Fi 获取音乐流、通过 Bluetooth LE Audio 发送，这点非常重要。此选项使用最少的 Bluetooth LE Audio airtime，但具有最高的延迟。
- 选项 2，旨在提供最高的可靠性和最低的延迟，最大化每帧中的重传次数。它使用最多的 airtime，因此仅适用于没有其他 2.4GHz radio 操作的广播设备。
- 选项 3，旨在实现可靠性和共存之间的平衡。对于使用 Wi-Fi 获取音乐流，但没有其他共存问题的 Broadcaster 来说，这将是一个不错的

---

选择。

这只是芯片 scheduler 可以选择的许多可能组合中的三种。随着时间的推移，随着行业获得更多的 Bluetooth LE Audio 经验，其它选项可能会被添加到推荐列表中。

在离开本章主题之前，表 5.8 显示了使用这三个选项中的每一个的立体声广播流的总体延迟，TMAP 和 HAP 规定的 20ms Presentation Delay，以及 BAP 的 baseline 40ms。Controller 将通知 Host 它选择了哪些参数，但 Host 无法控制该选择。这将取决于芯片中的调度算法。设计者应该意识到这种变化。如果应用程序的延迟非常关键，请询问芯片制造商，了解他们在 Controller 调度中所做选择的详细信息。

	Overall Latency	
	PD=20ms	PD=40ms
Option 1	122.3ms	142.2ms
Option 2	78.4ms	98.4ms
Option 3	112.0ms	132.0ms

表 5.8 48\_2\_2 广播音频流 QoS 配置使用 BAP LL 选项的广播立体音频 stream 的整体延迟

### 5.6.1 Airtime and retransmissions

我们在表 5.7 中提到了 airtime。尽管 LC3 codec 比以前的 Bluetooth codecs 更高效，但随着比特率的增加，数据包占用了越来越多的可用广播时间。表 5.7 中选项 2 的数字表明，使用这些参数的立体声 stream 几乎占可用 airtime 的 60%。这还不包括 advertising、其他活动的 Bluetooth 链路或任何其他 2.4GHz radio 活动所需的 airtime。

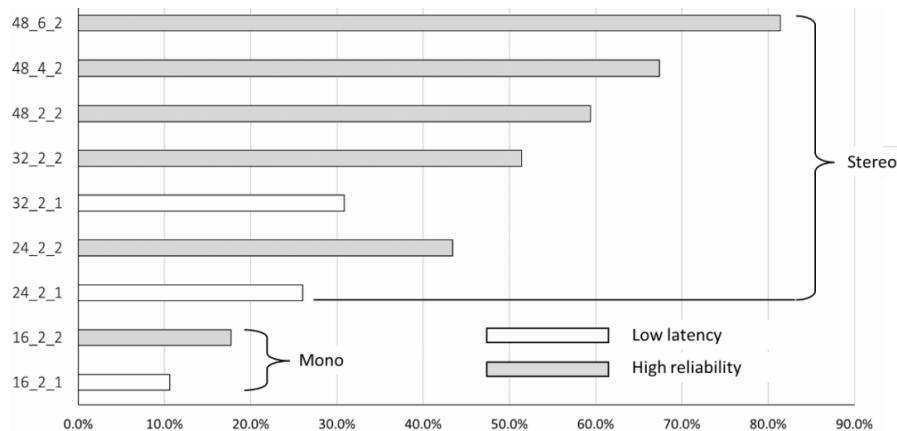


图 5.8 不同单向 QoS 配置的 Airtime 使用情况

图 5.8 展示了 QoS 配置中较高比特率的影响(使用 10ms frame 大小)，以

---

及 High Reliability 设置指定的更多重传的影响。对于 48kHz 采样配置，Low Latency 和 High Reliability 的 airtime 是相同的，因为较大的数据包对干扰的敏感性更高，因此建议每个数据包的最小重传次数为 4。

Broadcast Sources 不知道它们的重传是否已被接收，因此必须发送每个数据包。Unicast Initiators 只需要在接收不到确认时重新发送数据包。意味着在许多情况下，Unicast Initiators 会传输少得多的数据包，从而为设备上需要 airtime 的任何其他资源提供额外的 airtime。但是，如果连接的链路预算很少，例如耳塞和助听器等设备，尤其是在室外使用时，它们可能需要发送最大数量的 retransmissions，因此，必须为这种可能性分配 airtime。

Airtime 是一种有限的资源，提高流的音频质量和可靠性会影响到它。Bluetooth LE Audio 的设计要求之一是支持多个音频流，允许电视或电影院以多种语言传输音轨。参见图 5.8，很明显，任何 48kHz 采样配置都是不可能，除非使用多个无线电来传输每个不同的立体声语言流。相比之下，Low Latency 24kHz 或 32kHz 配置可以轻松应对两种不同的立体声流。如果产品设计师想要利用 Bluetooth LE Audio 传输多个流的能力，他们需要考虑 airtime 的影响。这带我们来到了 Audio Quality。

## 5.7 Audio quality

自从电子音频录制的早期，小部分用户就不断要求更高的质量。这导致了录音和录制技术的进步，以及 Hi-Fi 等术语的营销。除了真正的技术进步，它还看到了伪科学时尚的出现，如无氧铜电缆和镀金连接器，以“确保”模拟信号不被降级。尽管镀金 USB 连接器已经过时，但数字技术并没有降低人们对这些产品的热情。随着数字时代的到来，音频爱好者不断呼吁提高质量，要求更高的采样率、无损 codec 和更高的输出水平。事实上，现在许多 30 岁以上的人都有一定程度的听力损失，意味着他们听不到任何这些“改善”，但这并不能阻止产品营销经理推动更高的音频质量。

在早期，Bluetooth 音频吸引了大量负面报道。有些是确实如此的，因为一些 A2DP 耳机具有资源受限的 codec 实现。用于呈现音频的传感器也相对原始。从那时起，随着耳机、耳塞和扬声器的大量发展，很多事情都发生了变化，几乎没有用户担心 Bluetooth 技术这种音频解决方案，并且它通常被用于成本数千美元的高端音频设备。

在 LC3 codec 开发期间，Bluetooth SIG 委托对其音频质量与其他 codec 进行独立研究。结果证实，在 8kHz 到 48kHz 采样的整个频谱上，它提供了

---

与其他 codec 相同或更好的主观表现。LC3 编码音频流的示例可在 Bluetooth SIG 网站上收听。

测试由一组专家听众进行，他们在录音室使用高质量的有线耳机。他们被要求根据参考录音对每一个声音样本进行评分，并对其与原始录音的接近程度进行评分。测试使用 MUSHRA 协议，混合了来自 EBU 测试录音的声音。

像这样在完美的听力条件下完成的测试结果和真实世界的经验总是很难联系起来，因为它们是主观的。然而，我试图在表 5.9 中描述不同采样频率的一般应用。

Sampling Rate	Description
8kHz	适用于电话通讯
16kHz	高音质语音。适用于语音识别应用。
24kHz	适用于收听条件不完善的音乐，如背景噪音，或听众听力受损的情况。如果听众专注于无噪声环境中的音频流，他们很可能会从较高的采样率中察觉到差异。
32kHz	大多数用户在和任何背景噪声一起收听时都不会察觉到与原始声音的差异。
48kHz	用户无法察觉到与原始音频的差异。

表 5.9 不同采样率下的 LC3 音频质量的主观描述

重要的是，音频应用程序设计者明白，在设计无线音频体验时存在着妥协，并且某些 QoS 选择可能会排除新用例的开发。音频行业的一些部门将使用 LC3 提供的更高质量，并推广最高的采样率，尽管事实上，录制和收听环境的限制可能意味着很少(如果有的话)听众会鉴别它们。此外，所产生的延迟不适用于实时应用程序，某些设备可能无法对其进行解码。其它部门将研究 Bluetooth LE Audio 启用的新功能和用例，并选择 24kHz 或 32kHz 作为允许开发新用例的可接受折衷方案。只有时间才能决定用户最看重什么。可能是他们的应用更灵活，或者是希望在营销文献中看到更大的“质量”数字。

过去，音频行业曾两次决定降低音频质量。首先是 CD 的引入。第二个是 MP3 的使用，它支持音频流。每一次，都在更易于使用和保持当前音频质量之间取得了平衡。在这两种情况下，消费者都表示了对易用性的主要偏好。

## 5.8 Multi-channel LC3 audio

在第 3 章中，我们介绍了 Audio Channels 的概念。这意味着有多种不同的方式在设备之间传输相同的音频数据。要了解它们是如何工作的，最好看几个例子。在每种例子中，以下术语用于描述有多少音频通道被多路复用到一个 CIS 或 BIS 中。每个 CIS 或 BIS 上箭头的数量与它所承载的音频通道的

数量相对应。

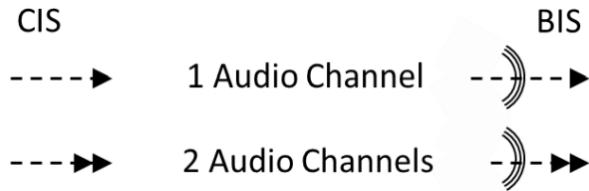


图 5.9 CIS 或 BIS 中音频通道数量的表示

图 5.10 展示了在 Initiator 和单个 Acceptor 之间传输立体声流的两种可能选项。这是将音乐从手机流到耳机或从电视流到条形音箱的简单单播使用情况。

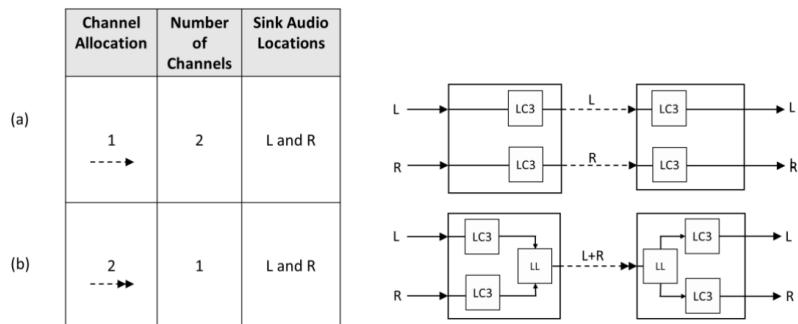


图 5.10 Initiator 到 Acceptor 立体声音频流的复用选项

在图 5.10 的顶部，选项(a)使用两个单独的 CIS，一个用于携带左声道，另一个用于承载右声道。在此之后，选项(b)将 Channel Allocation 设置为 2，以便 LC3 编码器的两个输出由 Controller (Link Layer——LL)串联到单个有效载荷中，并在单个 CIS 中发送。在 Acceptor 处，单独的左和右有效载荷在 Controller 中被分离，在 Presentation Delay 之后被解码并呈现为单独的声音。(Controller 在图中显示为 LL 块(Link Layer)。它不是多路复用流，而是简单地发送或接收单信道 codec frame，则为了简化此图，将它省略了。)

向单声道 Acceptor 发送相同的单播立体声输入有点有趣，因为在输入和输出之间的某个时间点，立体声信号需要被 downmixed 为单个单声道流。有三种可能的方法，如图 5.11 所示。

对于所有三个选项，Acceptor 将其 Audio Sink Locations 设置为 Front Left 加 Front Right，因此其支持的音频位置将为 0x0000000000000011。请注意，没有单声道 Audio Location，因为单声道是流的属性，而不是物理 Audio Location。在选项(a)中，Initiator 可以推断出 Acceptor 将呈现单声道信号，因为它已将 Channel Allocation 设置为 1，这意味着它将仅将流呈现到一个位置。如果它想要左声道和右声道，那么它会将 channel Allocation

设置为 2。设置两个 Channel Allocation 位，但声明它仅支持单个、非多路复用的 CIS，意味着它需要 Initiator 在发送输入音频通道之前将其向下混合为单声道。

在选项(b)和选项(c)中，Initiator 在 Codec Configuration 过程之后接收到的 Channel Allocation 和 Audio Sink Location 信息与 Acceptor 是立体声设备时接收到的信息相同(见图 5.10)。这意味着 Initiator 无法知道它是单声道还是立体声设备。因此，它向其提供立体声信息，或者作为选项(b)中的两个单独的 CIS，或者选项(c)中的单个 CIS 上的多路复用流。在这种情况下，Acceptor 负责 downmix 生成的流。

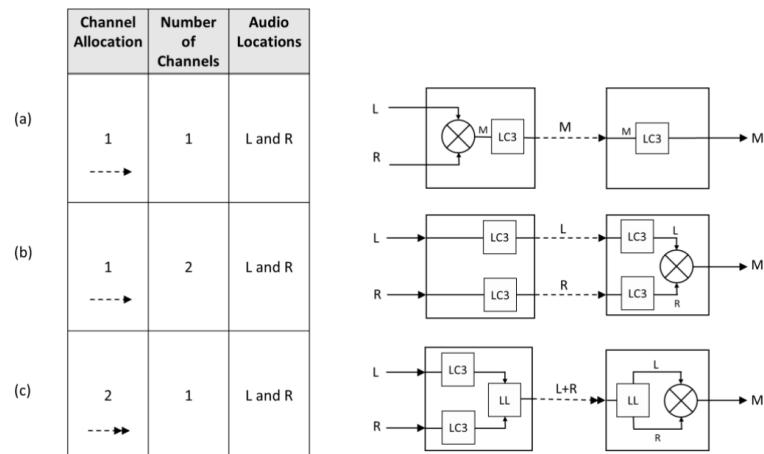


图 5.11 发送单播立体声到单声道 Acceptor

BAP 要求设置多个 Bluetooth LE Audio Locations 的所有设备能够支持至少相同数量的流，因此在选项(a)中，Acceptor 也需要能够支持选项(b)的两个 CIS 配置。如果它要求 Initiator 执行 downmix，它将在首选 PAC 记录中指定单个 Channel Allocation 和 Left plus Right Audio Locations(我们将在第 7 章中介绍)，以表示其对单声道 stream 的需求。

单独耳塞的流行使用情况也有不同的可能配置。图 5.12 展示了向一对 Acceptors 传输立体声音频流的两个选项。

选项(a)说明了通常使用的配置，其中耳塞将其 Audio Location 设置为 **Left 或 Right**(而不是前面示例中的 **Left 和 Right**)。Initiator 将向每个耳塞发送与其 Audio Location 相对应的单个 CIS。

在选项(b)中，两个耳机都将其 Channel Allocation 设置为 2，因此它们将接收多路复用流。然后，每个耳塞必须解码它们所需的流，丢弃另一个。这是一个**效率较低的选项**，但如果耳塞检测到用户已经转向，则允许它们交

换流，3D 声音和虚拟现实耳机中的应用也是如此，特别是如果有额外的空间音频内容可用时。

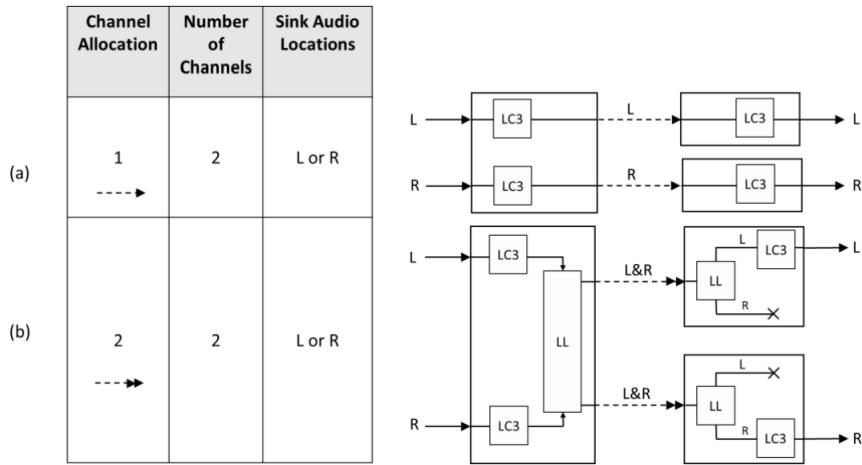


图 5.12 一副立体声耳机 stream 选项

上面所示的配置只是对可能的组合的小部分选择，并非每个 Initiator 和 Acceptor 都支持所有这些配置。BAP 列出了 14 种不同的流配置组合(并非详尽无遗)，强制要求最常见的流配置，并对其他流配置应用条件和可选要求。表 5.10 显示了 Initiator 和单个 Acceptor 之间的连接要求。M 表示它们是强制性的，必须由符合 Bluetooth LE Audio 的设备支持。C 是一种条件支持，通常基于设备是否支持双向流。O 表示支持是可选的。BAP 表 4.2 和 4.24 中提供了这些要求的全部细节，包括条件要求。

Audio Configurations 6 至 9 和 11 涉及两个流。在表 5.10 中，它们带有后缀(i)，表示它们指的是两个流中都包含一个 Acceptor 的用例，因此支持两个 CIS。

表 5.11 显示了 Initiator 与一对 Acceptors 建立流的要求，每个 Acceptors 连接一个 CIS。这些由后缀(ii)表示。

许多其他组合也是可能的，但 Bluetooth LE Audio 设备不能自动期望 Initiator 或 Acceptor 支持这些组合。Initiator 可以从 PAC 记录中以及 Codec\_Specific\_Configuration 的 Additional\_ASE\_Parameters 值确定对其他配置的支持。

## 5.9 Additional codecs

LC3 是一种非常好的 codec，可用于语音和音乐应用的各种采样率。每个 Bluetooth LE Audio 设备都必须支持 16kHz 和 24kHz 的采样率(Initiators

---

只需要支持 16kHz，因为某些公共扩音系统可能仅支持语音)，但大多数实现可能支持更高的采样频率。

尽管如此，在某些特殊应用中，其他 `codec` 的性能可能会更好。为此，Bluetooth LE Audio 规范允许使用额外的 `codec` 或供应商特定的 `codec`。

在 A2DP 中，其他 `codec` 被称为可选 `codec`。这些是由外部规范机构或公司设计的 `codec`，但 Bluetooth 规范包括配置信息，允许合格设备识别它们的存在以及如何设置它们。这允许多个制造商添加使用它们的能力。否则，连接将默认返回到强制的 Bluetooth `codec`。通常，额外的 `codec` 可以从外部标准组织获得许可，因此任何人都可以将其集成到产品中。目前，没有为 Bluetooth LE Audio 定义额外的 `codec`。

---

## 第6章 CAP and CSIPS

在 Bluetooth LE Audio 规范集的 Generic Audio Framework 中，四个 BAPS 规范(Basic Audio Profile、Audio Stream Control Service、Broadcast Audio Scan Service 以及 Published Audio Capabilities Service)完成了大多数艰难的工作。如果你实现了这四个规范，你就可以建立几乎任何单播和广播应用。然而，它们被设计为尽可能通用，这意味着有多种不同的方式将各个部分组合在一起。CAP – Common Audio Profile，定义了一系列程序，用来确立执行在 Initiator 和一个或多个 Acceptors 之间建立 Audio Streams 所需要的所有日常行为的通用方式。对于大多数开发者，他们在 CAP 提供的接口上建立自己的应用。

CAP 与我们在第三章所看到的 content control、use case 以及 rendering control concepts 相关。定义了在 stream configuration 和 establishment 过程中，上述这些何时需要被使用到。

CAP 为流程带来的另一事物就是 coordination。现如今，Bluetooth audio 最大的市场就是耳塞——两个独立的、需要像一个整体一样运行的设备。BAPS 规范处理单独的音频流。当 Initiator 连接到多个 Acceptor，CAP 规定了管理音频流的规则，并且使用 Coordinated Set Identification Profile 和 Service 将它们绑定在一起。

最后，CAP 定义了 Commander 角色，这将广播从简单的磁感线圈的替代品提升到了高灵敏度的音频分发解决方案。

在此章节中，我将提供对于 CAP 过程以及怎样使用它们的概览。本质上，你可以将 CAP 看作一个食谱。BAPS 定义了 Audio Streams 所需的所有原料，CAP 告诉你每个过程要使用哪一类原料，以及使用的顺序。大多数细节会在接下来的建立单播和广播音频流章节展现，在那里，我们会看到 CAP 怎样增强和使用 BAPS 中已经存在的流控制和管理过程。但是在 CAP 开始之前，理解 CSIP 和 CSIS 是很重要的。

### 6.1 CSIPS—the Coordinated Set Identification Profile and Service

因为 A2DP 是设计用来向单个设备发送流的，现如今，市场上每个向多设备发送音频的 Bluetooth 解决方案使用了专有的方法使多个设备一起工作，不管是一副耳塞、助听器或者扩音器。Bluetooth LE Audio 需要定义一套规

---

范的方法来完成此事，这样任意的产品组合都可以一起使用。不仅是确保它们可以同时进行音量控制，而且可以确保当你改变了提供音频流的设备时——比如当你在看电视时，一个电话打进来了，左右耳塞都会同时连接到你的手机。这一要去引出了 Coordinated Sets 的概念。

Coordinated Set Identification Service 会在组成一组(叫做 Coordinated Set)的设备上实例化，该组的成员符合特定的用例，以协调一致的方式运行。典型的例子就是一副耳塞。规范不限于音频设备——它们通用可以用于医疗传感器集合，比如 ECG 贴片，从独立的传感器收集数据。这些设备的所有功能不一定要完全相同，但是必须有一项功能是相同的。比如，一副耳塞，共同的功能是呈现音频流，但是只需要一个具有 mic。

对于一副耳塞， coordination 有四个主要功能：

- 定义 Coordinated Set 成员，比如左和右耳塞
- 用于两个设备的音量和静音控制
- 用于确保两个设备从相同的 Audio Source 接收音频，通常左右耳塞的音频流是不同的，但是这与 CSIPS 不相关，并且
- 运行设备对于耳塞的使用进行锁定，确保其他设备不能够使用它们。

CSIP 定义了两个角色：

- Coordinated Set 中的设备叫做 Set Member
- 发现和管理 Coordinated Set 的设备叫做 Set Coordinator

Set Member 角色本质上是被动的角色——它仅仅陈述自己是 set 的一部分。Set Coordinator 不仅仅要发现成员，还要将此信息传递给其他的过程，以此确保这些过程在 set 里面的所有成员上面起作用。

CSIS 需要 Coordinated Set 的每一个成员具有 Resolvable Set Identity (RSI) AD Type，使得 Client 设备识别到这是一个 Coordinated Set 的成员。意味着两个或者更多的 Acceptor 需要被发现。RSI 是一个随机的 6 字节标识符，随时间变化。减少了有人跟踪你的 Bluetooth 设备的风险。

作为 Initiator 或 Commander 的 Client 设备，一旦与广播 RSI AD Type 的设备建立连接，就会与它已经发现的 set member 配对及绑定，并且读取其 Set Identity Resolving Key (SIRK) 特征值。SIRK 是一个 128bit 的随机数，对于 Coordinated Set 中的成员是通用的，Client 可以它用来解码 Resolvable Set Identity。设备的生命周期都不会改变。SIRK 通常在设备的制造阶段被写入组成 Coordinated Set 的所有设备中。Bluetooth LE Audio 规范没有明确设置或改变它的方法，所以当产品需要在它们的生命周期中加入到某个

---

Coordinated Set, 比如当一个丢失或坏掉的耳塞, 需要被替换时, 制造商需要确定实现此过程的方法。

Client 也需要读取 Coordinated Set Size characteristic, 以此获取在该 Coordinated Set 中又多少设备。然后, 它继续使用定义在 CSIP 中的 Set Members Discovery Procedure 来查找 set 中的其他成员。这涉及 Client 设备寻找暴露 RSI AD Type 的其他 Acceptor, 并于它们连接和配对, 以及读取 SIRK characteristic。如果 SIRK 和第一个 coordinated 设备的相同, 它就是相同的 set 成员。如果不同, Client 设备应该取消配对, 寻找下一个带有 RSI AD Type 的设备, 并且检查它的 SIRK 特征值。当 set 中的所有成员被找到时, Set Members Discovery 过程就终止了, 此过程到达一个特定于实现的超时(通常为 10s), 或者, 由应用程序终止它。如果 Initiator 或 Commander 找不到所有的成员, 它们会以现有成员继续进行, 但是会继续尝试找到丢失的成员。

如果你在运送 Coordinated Set 中的 Acceptors, CAP 要求每个设备都需要实现 CSIS 中定义的四个 characteristics。它们定义在表 6.1 中。

Characteristic Name	Mandatory Properties	Optional Properties
Set Identity Resolving Key (SIRK)	Read	Notify
Coordinated Set Size	Read	Notify
Set Member Lock	Read, Write, Notify	None
Set Member Rank	Read	Notify

图 6.1 用于 CAP 的 Coordinated Set characteristic properties

目前, CAP 过程没有使用到 Set Member Lock 和 Rank characteristics, 尽管它们是被强制在 Acceptor 中实现的。

使用 Set Member Lock 的原因是当 Client 写 characteristic 时, Lock 会使 Client 独占 Acceptor 上的特征。Lock 控制哪些功能是由上层应用定义的。实现方应该小心使用 Lock, 因为它会阻止其他的 Commander 和 Initiator 访问该 Acceptor。当 Client 不需要再使用排他性访问时, 应该释放该 Lock。

Rank 是一个通常在制造时分配的值, 用于向 Coordinated Set 中的成员提供唯一的编号。它不意味着任何优先级, 而是 Coordinated Set 中唯一的正整数, 用于确保所有 Client 以相同的顺序对集合的成员执行它们的操作。Rank 在 CSIP 中的 Ordered Access 过程中使用。这表明, 当出于任何原因应用 Lock 时, Client 应该从其知道的具有最低等级的集合成员开始, 然后通过 Coordinated Set 的其他成员以 Rank 升序递增。防止了两个不同的 Client 同时对不同的 set member 使用 Lock 的竞争情况。Rank 值通常在制造过程中

---

设置。

Ordered Access 过程也被 CAP 用作运行任何 CAP 程序的先导。它检查任何 Acceptor 是否被锁定，只有在确定 set member 中没有一个具有 Lock set 之后，才继续执行 CAP 程序。然后，它将 CAP 程序按照 Rank 增加的顺序，应用到每个 Acceptor。

## 6.2 CAP—the Common Audio Profile

如我上面所说，CAP 可以被认为是所有 Bluetooth LE Audio 的配方书，它将所有其他规范整合为五个主要的程序集，定义了一切的工作方式。顶层的 profiles，比如 HAP、TMAP 和 PBP，然后参考这些 CAP 程序，以及对于特定的用例增加的附加要求。

我在整本书中使用的 Initiator、Acceptor 以及 Commander 角色就是定义在 CAP 中。虽然 Audio Streams 只涉及 Initiator 和 Acceptor，但 CAP 描述了这三个角色怎样包含其他来自于 Generic Audio Framework 规范的一系列组件。这些都列在了表 6.2 的矩阵中，其展示了 Mandatory，Optional，Conditional 以及 Excluded 需求。Conditional 功能通常被强制用于其他功能的特定组合，并且要求角色支持多个可选组件中的一个。Excluded 组合是不被允许的。这些组合的详细细节可以在 CAP 中的表 3.1 中找到。

横杠(-)框框是那些在设备中实现的功能，但是在 CAP 过程范围之外。

Role Component	Acceptor	Initiator	Commander
BAP Broadcast Assistant	X	O	C
BAP Scan Delegator	C	X	C
VCP Volume Controller	X	-	C
VCP Volume Renderer	O	X	X

Role Component	Acceptor	Initiator	Commander
MICP Microphone Controller	X	-	C
MICP Microphone Device	O	X	X
CCP Call Control Server	X	O	X
CCP Call Control Client	O	X	-
MCP Media Control Server	X	O	X
CSIP Set Coordinator	X	C	M
CSIP Set Member	C	X	X

“M” = Mandatory, “O” = Optional, “C” = Conditional, X = Excluded, “-” = Not relevant to this profile.

表 6.2 CAP 角色的组件支持需求。单独的情况见 CAP 的表 3.1

---

### 6.2.1 CAP procedures for unicast stream management

用于管理流的 CAP 程序可以分为 3 类。第一类是对于单播流的三个程序的集合，很大程度上是不言自明的：

- Unicast Audio Start procedure
- Unicast Audio Update procedure
- Unicast Audio Stop procedure

在建立每一个单播流时，使用命令和响应序列，这些程序都涉及 Initiator 和 Acceptor。

### 6.2.2 CAP procedures for broadcast stream transmission

三个程序中的第二个集合用于广播流：

- Broadcast Audio Start procedure
- Broadcast Audio Update procedure
- Broadcast Audio Stop procedure

Broadcast Audio Start、Stop 以及 Update 程序只用在 Initiator，因为它 是单方面建立，没有来自 Acceptor 是否在场的响应。

### 6.2.3 CAP procedures for broadcast stream reception

为了补充 Initiator 的广播程序，流管理程序的第三种集合用于想要获取 广播 Audio Stream 的 Acceptor，给出了两个程序：

- Broadcast Audio Reception Start procedure
- Broadcast Audio Reception Ending procedure

这里没有 Acceptor 的 Broadcast Audio Update 程序，因为 Acceptor 在广 播中是一个被动的、只是消费 Audio Stream 的实体。它不能对于更新广播 流的内容做任何事情。因此对于广播音频接收没有 update 程序。

### 6.2.4 CAP procedures for stream handover

对于 Initiator 想要在传输单播和广播音频流之间进行转换(反之亦然)的 stream handover，CAP 有两个特有的程序。它们是：

- Unicast to Broadcast Audio Handover procedure 以及
- Broadcast to Unicast Audio Handover procedure

这些非常重要，因为它们描述音频分享用例，用户可以从他们的手机或 音源收听私有音频流通过将其广播转到将音乐和朋友分享。尽管传输拓扑在 CIG 和 BIG 之间转换，原本的 Initiator 仍然保持对音频流和用户的 Acceptor

---

的控制。

Handover 程序的重要性的原因是大多数 Initiator 和 Acceptor 没有资源同时处理单播和广播流。对于 High Reliability QoS 设置，根本没有足够的射频资源来完成两个类型音频流。意味着，通常 Initiator 通常需要在开启广播之前停止单播流，即使对于 24Khz 采样率的音频流。为了确保主要用户(比如与朋友共享音乐的用户)的连续性，Initiator 必须将相同的 Context Types 用于新的音频流。大多数情况下，音频流的主要用户想要控制音频流，所有尽管音频流是广播状态，他们仍然保持 ACL 链路，并将新的广播音频流与具有相同的 Content Control ID (CCID) 的单播流相关联。

实际上，在这种切换过程中，传输可能存在短的间隙，会在 Acceptor 初始链接到 Initiator 时较为明显。然而实现中可以尽量减少这个间隙，可以更简单地通过添加一个简单的用户通知“请稍后，我们将转到音频共享”来隐藏它们。

#### 6.2.5 CAP procedures for capture and rendering control

除了针对 Audio Stream 的四个程序集合之外，CAP 也将音量和呈现控制与 5 个 Capture and Rendering Control 程序联系在一起，这些都是不言自明的：

- Change Volume procedure
- Change Volume Offset procedure
- Change Volume Mute State procedure
- Microphone Mute State procedure
- Change Microphone Gain Settings procedure

#### 6.2.6 Other CAP procedures

除了 Audio Stream、Capture 和 Rendering Control 程序之外，CAP 还定义了一组与上述 Audio Stream 程序以前使用的程序。**第一个**就是我们在上一节所看到的，**Coordinated Set Member Discovery** 程序，由 Initiator 在建立 unicast stream 之前，Initiator 或 Commander 在 Coordinated Set 之上，调节音量或捕获设置之前执行。当已经知晓 Coordinated Set 的成员，CAP 总是在另外的 CAP 程序之前运行 CSIP Ordered Access Procedure。**第二个**就是**Distribute Broadcast Code** 程序，定义了用于加密，私有广播流的 Broadcast\_Codes 怎样分发。

---

### 6.2.7 Connection establishment procedures

除了广播用例之外，Bluetooth LE Audio 使用与用在其他 LE 应用之上相同的 peripheral 连接过程，设备发现、LE ACL 连接和绑定都一样。BAP 的第 8 节定义了此过程，参考 Core 规范中的相关章节，并且提供了用于 quick setup 及 reduced power situations 的 Scan Interval、Scan Window 及 Connection Interval。这些都是定义在 Generic Access Profile (GAP) 中的标准的 connection 程序，所有我不会在这里花费更多时间。

CAP 的第 8 节对 BAP 需求进行了扩展，并考虑了多重绑定的情况，即一个 Acceptor 与多个 Initiator(例如，手机和电视)绑定。它还引入了两种新的模式，反映了 Initiator 和 Acceptor 都可以做出连接决策的事实，以支持“Sink led journey”的概念。这些是“Immediate Need for Audio related Peripheral”模式以及“Ready for Audio related Peripheral”模式。

#### 6.2.7.1 Immediate Need for Audio related Peripheral (INAP) mode

INAP 模式用于当 Initiator 需要连接 Acceptor 时，通常是因为用户操作，比如按下按钮或外部操作(如来电)。由于这通常是高优先级响应，Initiator 应该使用 BAP 第 8 节中的 quick setup 参数值来建立连接。Initiator 应该确定响应的 Acceptor 是否为 Coordinated Set 的成员，如果是这样，就会发并连接其他所有的 set member。如果一个以上 Acceptor 响应，Initiator 应该向用户提供可用选项。

#### 6.2.7.2 Ready for Audio related Peripheral (RAP) mode

与 INAP 相反的情况是 RAP 模式，Acceptor 正在寻找 Initiator。通过发送包含描述它想支持的用例的 Context Type 的 Targeted Announcements 来表示。如果 Initiator 支持，应该尝试去连接。在 RAP 模式中，Initiator 应该使用 BAP 第 8 节描述的 reduced power 参数值来连接。当连接到正在发送 Targeted Announcements 的 Acceptor 之后，Initiator 应该决定 Acceptor 是不是 Coordinated Set 的成员，如果是的话，就会 discover 及 connect 到其他所有的 set member。

对于 RAP 或 INAP 模式中不同形式的 announcement，Initiator 响应范围在 CAP 的表 8.4 中描述。

### 6.2.8 Coping with missing set members

有时，Initiator 读取了 Coordinated Set 成员的 Coordinated Set Size characteristic，试图查找其他成员，并发现缺少了一些成员。这可能发生在

---

Audio Stream 建立的开始，或者在会话过程中，当 Initiator 发现与其中一个 Acceptor 的链路丢失时。这可能是因为它们中的一个或多个物理上面丢失、超出范围，或者它们的电池耗尽。发生这种情况时，Initiator 应当尝试查找丢失的成员，但不应该停止连接过程，也不应该终止任何现存的 stream。相反，它应该定期尝试查找丢失的集合成员，并发现它们后通过重新建立连接将它们添加回来。

如果成员丢失，实现可以决定调整单播 Audio Stream 的内容。例如，你的手机正在向 Coordinated Set 的左右耳塞发送立体音乐流，此时其中的一个的连接断开了，它可能决定用 down-mixed 单声道 stream 来替换现有的 stream。然而，此行为没有在 Bluetooth LE Audio 明确规定并且是特定于实现的，尝试查找缺少的 Coordinated Set 成员的 retry cadence 以及超时也是如此。这不应影响 Context Type，因为用例保持不变。

Common Audio Profile 可能最好概括为一个 profile，描述 “this is how to connect” 以及 “do this for all of the streams you’re dealing with”。因此，它汇集了所有其他的 Generic Audio Framework 规范，为顶层 profile 提供了一套通用的程序。随着 GAF 内制定了更多的规范，CAP 将扩展以包含这些规范。

接下来的几章，我们将介绍建立单播和广播 stream 以及 Capture 和 Rendering Control 的所有 CAP 过程。大多数开发者会发现，它们是在使用这些程序，而不是底层 BAP 程序，因为 CAP 程序是被顶层 profile 引用的。然而，理解用于建立和管理 stream 的 BAP 程序和参数是理解 Bluetooth LE Audio 世界的所有内容如何配合的有用实践。这就是我们接下来要做的。

---

## 第7章 Setting up Unicast Audio Streams

这一章，我们会学习怎样配置和建立单播 Audio Stream。本阶段涉及的四个主要的规范是：

- PACS – the Published Audio Capabilities Service
- ASCS – the Audio Stream Configuration Service
- BAP – the Basic Audio Profile, 以及
- CAP – the Common Audio Profile

CAP 位于其他三个之上，定义了使用 BAP、ASCS 以及 PACS 来配置和管理单播 Audio Stream 的步骤，引入了 coordination、Context Types 以及 Content Control ID，以此将用例与 Audio Stream 关联起来。大多数时间，CAP 只是一系列教你怎样执行 BAP 步骤以及怎样使用与之相关的 Context Types 和 Content Control ID 的规则。顶层 Bluetooth LE Audio profile，比如 HAP 和 TMAP，都是建立在 CAP 之上，主要扩展了强制性要求。在本章中，我将专注于底层的 BAP 过程，因为它们负责繁重的任务，但在必要时，也会引用 CAP 以及更高层的 profile。

所有的 Bluetooth LE Audio 规范都是基于 Bluetooth LE 的 GATT，具有 Client-Server 结构。更高层的规范向单播 Audio Stream 提供 context 以及增加 control，但是 BAP，ASCS 和 PACS 是它们建立的基础。

### 7.1 PACS – the Published Audio Capabilities Service

PACS 是这些规范中最简单的一个，本质上是 Acceptor 所能做的事情的一个陈述。Acceptor 使用 PACS 以两个 characteristics 展示这些 capabilities——如果它支持 Audio Sink role，就有个一个 Sink PAC 特征；如果支持 Audio Source role，就有一个 Source PAC 特征。这些包含了 Published Audio Compatibility 记录，叫做 PAC 记录，其包含了它支持的 codec 和配置信息，还有另外其他的可选的功能。在它们之间，它们将设备全部的 capabilities 展示出来。

PACS 可以被认为是一个 Acceptor 音频功能的数据库，是在生产时被写入的，并且在产品整个生命周期内是静态的，尽管它可以在固件升级时被更新。当 PACS 开始配置和建立 audio stream 程序时，它就会明确 Initiator 所包含的信息。PACS 也可以被 Broadcast Assistant 用来过滤其侦测到的 Broadcast Streams，这样 Broadcast Sink 只需要显示兼容的 Audio Streams，

---

但是，会在 broadcast 章涉及到。

有关 PACS 明确的一个重要概念——它是关于 Acceptor 的全部 capability 的实际陈述。PACS 描述了 Acceptor 能够做什么。这是设备相互了解对方的第一步。在 Initiator 读取了这些 capability 之后，在建立 stream 的过程中，Acceptor 通常会公开一组更有限制的首选 capability，并且 Initiator 应该使用这些值。然而，Acceptor 不应该拒绝 Initiator 基于已公开的 PACS 信息的任何配置请求。

这样做的目的是实现高效的配置过程，特别是当涉及多个 Acceptor 时，特别是当它们来自不同的制造商并且具有不同的功能时。这是从 classic Bluetooth 音频 profile 演变而来，它运行 Central 和 Peripheral 设备通过协商循环，在此循环中，两个设备试图确定音频连接的最佳设置是什么。某些实现中，这会导致死锁，无法建立连接，或糟糕的 codec 配置会影响音频质量。这个问题导致包含“S”和“D”编码设置作为 CVSD 的建议选项，以及 mSBC 的“T”参数集。PACS 的目的就是防止这些问题的出现。

### 7.1.1 Sink PAC and Source PAC characteristics

Sink PAC 和 Source PAC 特征都包含了一个“i”PAC 记录数组，其格式如表 7.1 所示。

PAC Record	Description
Number_of_PAC_Record	Number of PAC records [i] for this characteristic
Codec_ID[i] (5 octets)	Octet 0: Codec (defined in the Bluetooth Assigned Numbers) LC3 = 0x06 Vendor Specific = xnn Octets 1&2: Company ID, if vendor specific, otherwise 0x00 Octets 3&4: Vendor specific Codec ID, otherwise 0x00
Codec_Specific_Capabilities_Length[i] (1 octet)	Length of the codec specific capabilities for the codec in the i th PAC Record
Codec_Specific_Capabilities [i] (length varies)	Identification of the capabilities of the codec implementation in the i th PAC Record, normally expressed as a bitfield.
Metadata_length [i] (1 octet)	Length of the metadata associated with the i th PAC Record. 0x00 if there is none.
Metadata [i]	LTV formatted metadata for the i th PAC record

表 7.1 PAC Record 格式

### 7.1.2 Codec Specific Capabilities

每个 Bluetooth LE Audio 规范都包含了至少一个使用 LC3 codec 的 PAC 记录，因为这在 BAP 中是强制性的，BAP 是 Generic Audio Framework 的最低层级。LC3 的 assigned number 是 0x06，所以每个 Bluetooth LE Audio Acceptor 会有至少一个 Codec\_ID 是 0x0000000006 的 PAC 记录。注意，Codec\_ID 是 Host Controller Interface Assigned Number 列表中描述的 Coding Format，并不是 Audio/Video assigned numbers 列表中定义的 Audio Codec

---

ID。

LC3 的 Codec\_Specific\_Capabilities 需求定义在 BAP 的 4.3.1 节，并且由 5 个 LTV 结构组成。每个 LTV 具有定义在 Generic Audio Assigned Numbers 文档中的 Type code，这样设备可以通过读取来识别它。

由于这些 LTV 中的三个是位域，意味着 Codec\_Specific\_Capabilities 结构中通常有多个组合。

#### 7.1.2.1 The Supported\_Sampling\_Frequencies LTV

5 个 LTV 结构中的第一个就是 Supported\_Sampling\_Frequencies，如图 7.2 所示(Type=0x01)，是一个覆盖 8kHz 到 384kHz 范围采样频率的 bitfield。

通过设定相应的 bit 为 1 来表示支持该采样率。(注意，此结构可用于任何 codec。LC3 仅仅支持 8, 16, 24, 32, 44.1, 48kHz 采样率，表格中的阴影部分)。

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
kHz	RFU			384	192	176	96	88.2	48	44.1	32	24	22.05	16	11.025	8

表 7.2 支持的采样率(0x01)

#### 7.1.2.2 The Supported\_Frame\_Durations LTV

第二个 codec capabilities LTV 结构就是 Supported\_Frame\_Durations，表 7.3(Type=0x02)，提供了 LC3 codec 支持的两个 frame durations 信息——7.5ms 和 10ms。当 Bit1 和 Bit0 设置为 1 时，表示对 LC3 的 7.5ms 和 10ms frames 两个选项的支持。如果都支持的话，bit4 和 5 可以用来表示其中一个是否优先选择。Bit4 和 5 只能有一个设置为 1。此 LTV 是强制性的。

Bit	All Other bits	5	4	3	2	1	0
FD	RFU	10ms Preferred	10ms Preferred	RFU	RFU	10ms Supported	7.5ms Supported

表 7.3 支持的 frame duration(0x02)

#### 7.1.2.3 The Supported\_Audio\_Channel\_Counts LTV

第三个支持的 LTV 结构就是 Supported\_Audio\_Channel\_Counts，如表 7.4 所示(Type=0x03)。这是另一个 bitfield，用于表明 CIS 或 BIS 中可以包含的 Audio Channel 数量。Channel Count 就是在一个 Isochronous Stream 上，以相同方向传输的多路复用 Audio Channel 的数量，我们在第 5.8 节介绍过。Bit0 到 Bit7 用于表示支持的 channel count，1 表示一个支持项。至少有一位被设置为 1，否则表示不能建立音频通道。如果不支持多路复用，Channel Count 就是 1，并且此 LTV 结构可以被忽略。

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC	RFU								8	7	6	5	4	3	2	1

---

表 7.4 支持的 Audio Channel Counts (0x03)

#### 7.1.2.4 The Supported\_Octets\_Per\_Codec\_Frame LTV

第 4 个 LTV 结构就是 Supported\_Octets\_Per\_Codec\_Frame，如表 7.5 所示(Type=0x04)。此结构由两对两个字节组成，较低的那一对(字节 0 和 1)指定每个 codec frame 最小字节数，较高两字节指定每个 codec frame 支持的最大字节数。当只支持一个字节数时，两者可以设置成同一个值。它定义了所选的采样率支持的 bitrate 范围(见表 5.1)。此 LTV 也是强制性的。

Octet	15-4	3	2	1	0
Value	RFU	maximum number of octets per codec frame		minimum number of octets per codec frame	

表 7.5 支持的 frame duration(0x02)

#### 7.1.2.5 The Supported\_Max\_Codec\_Frames\_Per\_SDU LTV

Supported\_Codec\_Specific\_Capabilitie 中第 5 个，也是最后一个 LTV 结构是 Supported\_Max\_Codec\_Frames\_Per\_SDU(Type=0x05)，用于描述可以被打包到一个 SDU 的最大 codec frame 的数量。如果每个 SDU 只有一个 frame 时，它可以被忽略。

#### 7.1.2.6 The Preferred\_Audio\_Contexts LTV

Codec Specific Capabilities 之后可以跟着 metadata，它以 LTV 结构进行格式化，并且 Generic Audio Assigned Numbers 文档的 Codec Specific Capabilities LTV structures 章节中进行描述。目前，唯一与 PAC 记录相关的已定义的 metadata 是 Preferred\_Audio\_Contexts LTV。这是一个两字节的 Context Type 值的位域，使用 Generic Audio Assigned Numbers 文档中的 Context Types 标准值。如果 bit 被设置成 0b1，表示在 PAC 记录中，此 Context Type 是一个首选的 codec 配置用例。它通常只在 Acceptor 首先特定的 PAC 记录用于特定的用例时才会出现，**提供比 Supported\_Audio\_Contexts 特征更多粒度。**

### 7.1.3 Minimum PACS capabilities for an Audio Sink

强制的 BAP LC3 需求意味着 Acceptor——作为单播 Audio Sink (比如，接收 Audio Stream)，必须支持接收至少一个 16 和 24kHz 采样频率、10ms frame duration 的音频通道。类似地，每个 **Acceptor(Initiator)**——作为单播 Audio Source(比如，发送 Audio Stream)，必须支持至少编码一个 10ms frame duration、16kHz 采样率的通道。

Audio Sink 和 Source 的强制 16kHz 采样频率需要支持每个 SDU40 字节，而 Audio Sink 的 24kHz 采样率需要 60 字节。对于接下来展示 PAC 记录

的例子，我们只看一下 Sink PAC 特征。相同的原则适用于 Source PAC。

Sink PAC 特征通常会在单个 PAC 记录中公开这两个强制的 codec 设置。或者，可以将它们分离为两个 PAC 记录，每个记录指定一组功能。BAP 只需要对一个音频通道的支持(BAP 中的表 4.2，Audio Configuration 1)，但是为了展示 PAC 记录怎样建立，我们来看一个 Acceptor 支持两个 Audio Channel(对应于 Audio Configuration 4)的例子。如果它将这些作为四个独立的 PAC 记录公开的话，它们将如同表 7.6 所示。

PAC Record	0	1	2	3
Codec ID (LC3)	0x0D	0x0D	0x0D	0x0D
Supported Codec Specific Capabilities Length	0x0B	0x0B	0x0B	0x0B
<b>Supported Codec Specific Capabilities</b>				
<b>Supported_Sampling_Frequencies</b>	(例如: 16kHz=0x04, 24kHz=0x10)			
Length	0x02	0x02	0x02	0x02
Code	0x01	0x01	0x01	0x01
Value	0x04	0x04	0x01	0x01
<b>Supported_Frame_Durations</b>	(例如: 仅支持 10ms)			
Length	0x02	0x02	0x02	0x02
Code	0x02	0x02	0x02	0x02
Value	0x02	0x02	0x02	0x02
<b>Supported_Audio_Channel_Counts</b>	(例如: 1=0x00; 2=0x01)			
Length	0x02	0x02	0x02	0x02
Code	0x03	0x03	0x03	0x03
Value	0x00	0x01	0x00	0x01
<b>Supported_Octets_per_Codec_Frame</b>	(例如: 40 或 60=0x2828 或 0x3C3C)			
Length	0x03	0x03	0x03	0x03
Code	0x04	0x04	0x04	0x04
Value	0x2828	0x2828	0x3C3C	0x3C3C
<b>Supported_Max_Codec_Frames_Per_SDU</b>	(可选的。Default=1)			
Length	0x02	0x02	0x02	0x02
Code	0x05	0x05	0x05	0x05
Value	0x01	0x01	0x01	0x01

表 7.6 支持的 Audio Channel Counts (0x03)

作为它们如何构成的例子，表 7.6 包含了所有支持 Bluetooth LE Audio profile 的强制 codec 需求的 Sink PAC 记录参数，即 BAP 表 5.2 和表 6.4 中的 16\_2\_1、16\_2\_2、24\_2\_1、以及 24\_2\_2QoS 配置。涵盖了：

- 值为 0x04 的 16kHz 以及 0x10 的 24kHz 的 Supported\_Sampling\_Frequencies 特征
- Supported\_Frame\_Durations 为 0x02 表明它仅仅支持 10ms frame
- Supported\_Audio\_Channel\_Counts 的值为 0x00 和 0x01 以支持一个和两个通道
- Supported\_Octets\_per\_Codec\_Frame 值为 0x2828 和 0x3C3C 用于 40 和 60 字节，对应于 16\_2\_n 以及 24\_2\_nQoS 设置，最后
- Supported\_Max\_Codec\_Frames\_Per\_SDU 值为 0x01 时，表示每个 SDU 只有一个 codec frame。当它是默认值时，这个是可以省略的。在此 Sink PAC 特征中，没有 metadata。

---

同样的信息可以在表 7.7 中的单个 PAC 记录更简洁地提供:

PAC Record	Record 0(L-T-V)
Codec_ID (LC3)	0x0D
Codec_Specific_Capabilities_Length	0x0B
Codec_Specific_Capabilities	
Supported_Sampling_Frequencies	0x 02 01 14
Supported_Frame_Durations	0x 02 02 02
Supported_Audio_Channel_Counts	0x 02 03 03
Supported_Octets_per_Frame	0x 03 04 3C28
Supported_Max_Codec_Frames_Per_SDU	0x 02 05 01(default=1 可以省略)

表 7.7 将表 7.6 中的 PAC record 内容作为单个来记录

这两种表示法并不完全等价。PACS 规定, Acceptor 必须支持一个参数值与一个 PAC 记录中声明的参数值的其他所有可能组合。那就意味着, 当一个 PAC 包含多个值时(如表 7.7 的示例), 每个可能的组合都是有效的, 因为 Initiator 可以将 PAC 记录展开为所有可能组合的数组。理论上来说, 意味着如果需要的话, 公布表 7.7 的单个 PAC 记录的 Acceptor 应该支持 24kHz 采样率下的 40 字节以及 16kHz 采样率下的 60 字节, 以及 40 到 60 之间任意字节值, 尽管这些都是非标准的。为了互操作性, 有一个假设, 即 Initiator 应该限制自己使用 BAP 表 5.2 和 6.4 中特定配置, 这些配置是经过尝试和测试过的, 但是 Initiator 应该选择另一种组合。这不是很好的做法, 却是允许的。如果这些组合在一个设备中不都是有效的, 那么 PAC 记录应该表示为单个 PAC 记录。然而, 这可能会导致记录的数量迅速增加, 这不是一件好事。上面的示例是 BAP 所需支持的底线, 它产生 4 个 PAC 记录, 这是可管理的。尝试对 TMAP 做同样的事情将需要 96 个单独的 PAC 记录。正如我们很快会看到的, ASE 配置过程可以指导配置 Isochronous Stream 优先选项, 因此, 单独的 PAC 记录可以保留给应用, 比如供应商独有的

capabilities。详情请参阅 PACS 的第三节

如果一个 Acceptor 支持多个 codec 类型, 每种类型至少需要一个 Sink 或 Source PAC 特征, 因为 PAC 记录的 Codec\_ID 域是唯一的。实现决定 codec 的所有 PAC 记录是在单个或多个 Sink PAC 或 Source PAC 特征中公开。如果有大量的 PAC 记录, 实现可以将它们分割成单独的 PAC 记录特征, 以避免读取它们时超过 ATT MTU 的最大值。

Sink PAC 特征不区分单播和广播 stream, 所有这些值适用于两者。Source PAC 特征在广播时被忽略。

#### 7.1.4 Audio Locations

Sink PAC 和 Source PAC 特征都快分别具有关联的 Sink\_Audio\_Locations 和 Source\_Audio\_Locations 特征, 尽管这些特征是可

---

选的。Audio Locations 的基本概念在第 3 章中。Sink\_Audio\_Locations 和 Source\_Audio\_Locations 特征指定了对于接受和发送的 audio stream，Acceptor 支持那些 audio location。每个特征都有一个 4 字节域，这是一个用来表示它支持的呈现和捕获 location 的位域。

如果它们存在，则至少一位设置成 1。如果它们不存在，设备应该接受 Initiator 建议 Audio Location。许多情况下，这些值是由制造商设置的，并且是只读的，右耳塞将永远是右耳塞，所以有一个 Front Right 的 Sink 和/或 Source Audio Location。扬声器的 Sink\_Audio\_Locations 特征可能同时设置成 Front Left 和 Front Right。当 Initiator 想要建立 stream 时，它会决定发送哪个 Audio Stream 以匹配 Audio Location。如果它找到两个扬声器，它通常会发送一个左声道立体声 stream 到公布 Front Left location 的扬声器，并且发送右声道立体声到公布 Front Right location 的扬声器。如果两个扬声器都声明自己支持均支持 Front Left 和 Front Right(大多是这样)，用户通常会使用应用程序去配置哪个是哪个。Sink 和 Source Audio Location 特征可能是可写的，允许配置实用程序设置扬声器 location 以便其他 Initiator 使用。因为这些都是 Bluetooth LE Audio 规范中的互操作性功能，意味着任何兼容的音频应用程序都应该可以执行这种配置。或者，扬声器制造商可以在扬声器上提供一个物理开关，允许用户指定扬声器接受左或者右声道 stream。

第 5 章中，我们研究了 Initiator 和 Acceptor 阵营出来用于单声道复制的 downmixing 立体声流，表明这可以在编码 Audio Stream 之前完成，也可以在接收和解码之后完成。Acceptor 可以使用它的 PAC 记录向 Initiator 表明他是否能够 downmixing。如果它将 Audio Location 置为 Front Left 和 Front Right，但是只支持单个 Bluetooth LE Audio Channel，那么意味着它不具备 downmixing 能力，而只会呈现向它提供的 stream。如果它显示支持两个通道，意味着它可以将它们解码为单独的 stream 并且呈现出来。然后由 Acceptor 实现来将它们呈现为：

- 分开左右声道，如果它是一个 sound-bar 或者立体扬声器，就会这样做，
- 它解码的两个音频通道的一个，如果它是一个耳塞或助听器，它可能会这样做，或
- 将它们 downmix 到一个单声道音频通道，如果它包含单个扬声器，它就会这样。

后两个选项通常需要扬声器的某种级别的用户配置，但这取决于实现。

---

`Source_Audio_Locations` 特征以相同的方式表现，尽管大多数情况下，它可能仅支持 `Front Left` 和 `Front Right`。如果 `Acceptor` 支持单个音频通道，`Initiator` 会假定它是单声道麦克风。如果它支持两个音频通道，`Initiator` 会假定它是立体声麦克风。

`Sink` 和 `Source Audio location` 特征的使用，以及 `Audio Channel Counts` 为引导 `Audio Stream` 提供了很大的灵活性。其中大部分是隐含的，因此实现者需要仔细考虑它们使用的参数组合。

`Sink` 和 `Source Audio Location` 特征值是可以改变的，包括在连接期间。但是，在连接期间发生这种情况，则不需要终止音频流。新的值将应用于下一个 `stream` 建立过程。应用程序还可以决定改变 `stream`，例如，如果用户因为面对的方向而想要将立体声效果镜像，则可以在左右音频流之间交换左右通道输入。这是特定于实现的，在规范之外。

### 7.1.5 Supported Audio Contexts

每个 `Acceptor` 都需要包含 `Supported_Audio_Contexts` 特征，该特征列出了它支持的 `Context Types`。它通常是一个静态列表，只有当软件更新改变了 `Acceptor` 的功能时，它才会改变。

`Supported Audio Contexts` 这个名称有一点误导性。这个特征所做的是列出 `Context Types`(即用例)，对于这些用例，`Acceptor` 可以使自己 `unavailable`，以防止任何 `Initiator` 试图为 `Acceptor` 不感兴趣的用例建立音频流。对于每个 `Context Type` 的支持是可选的，除了 `Unspecified Context Type`，它必须在每个 `Supported_Audio_Context` 中的到支持[BAP 3.5.2.1]。如果 `Audio Context` 没有被标记为受支持，`Initiator` 仍然可以通过将该 `Context Type` 映射到其音频流的 `Unspecified Context Type` 来尝试建立一个 `stream`。如果 `Acceptor` 目前将 `Unspecified` 设置为 `available`(见下文)，则它没有理由拒绝该请求。作为接收音频流的结果，`Acceptor` 可能因为此 `Context Type`，决定为将来的请求而更改该 `Available_Audio_Contexts`。

鉴于这种行为，`Acceptor` 应该在它认为有理由使该用例 `unavailable` 时，在 `Supported_Audio_Contexts` 特征中设置每个 `Context Type` 位。(`Audio Sink` 不能对所有的东西都不可用，因为 `Assigned Numbers` 文档禁止 `Available_Audio_Contexts` 的值为 `0x0000`)。

### 7.1.6 Available Audio Contexts

`Acceptor` 使用 `Available_Audio_Contexts` 特征来通知 `Initiator` 它不能使

用特定的 Context Type，虽然在之前的 Supported\_Audio\_Contexts 中声明过其是受支持的。为了允许 Initiator 继续音频流的建立过程，Initiator 想要建立的音频流的 Context Type 必须在 Available\_Audio\_Contexts 特征中显示为 available。与静态的 Supported\_Audio\_Contexts 值不同，Acceptor 可以在任何时候更新其 Available\_Audio\_Contexts 值，更新时通知已连接的 Initiator。

为了说明这一点，图 7.1 展示了 Supported\_Audio\_Contexts 和 Available\_Audio\_Contexts 特征的典型设置。

Context Types												
EmergencyAlarm	Alerts	Ringtone	Notifications	Sound effects	Live	Voice assistants	Instructional	Game	Media	Conversational	Unspecified	
Bit	11	10	9	8	7	6	5	4	3	2	1	0
Available Audio Contexts	1	0	1	0	0	0	0	1	0	1	1	1
Supported Audio Contexts	1	0	1	0	0	0	0	1	0	1	1	1

图 7.1 Supported\_Audio\_Contexts 和 Available\_Audio\_Contexts 设置实例

在此实例中，Acceptor 支持 “Unspecified” (强制性的)，也支持 “Emergency Alarms”、“Ringtone”、“Conversational”、“Instructional” 以及 “Media”。它所示的这些都是 “available”。如果 Initiator 想要开始于其中任何 Context Types 建立 stream，Acceptor 应该要接收。

如果 Initiator 想要建立一个 stream 来携带按键音，它可以包含在 “Sound Effects” Context Type 中。因为 Available\_Audio\_Contexts 位图中显示 “Unspecified” 是 “available”的。只要 Acceptor 中的 “Unspecified” 显示为 “available”，Initiator 就允许将它想要使用的 “unavailable” Context Type 映射到它的 Streaming\_Audio\_Contexts metadata 属性中的 “Unspecified”。此种情况下，Acceptor 必需接受该音频流。

Context Types												
EmergencyAlarm	Alerts	Ringtone	Notifications	Sound effects	Live	Voice assistants	Instructional	Game	Media	Conversational	Unspecified	
Bit	11	10	9	8	7	6	5	4	3	2	1	0
Available Audio Contexts	0	0	0	0	0	0	0	1	0	1	1	1
Supported Audio Contexts	1	0	1	0	0	0	0	1	0	1	1	1

图 7.2 较少 bit 置 1 的 Supported\_Audio\_Contexts 和 Available\_Audio\_Contexts

图 7.2 展示了 Supported\_Audio\_Contexts 特征中设定 Context Type 位的值。此例中，“Emergency Alarm” 和 “Ringtone” 是 “supported”，但是当前在 Available\_Audio\_Contexts 特征中没有标记为 “available”。这就意味着，如果 Initiator 尝试建立与这些 Context Types 相关的 stream 时，它会被 Acceptor 拒绝，因为这些都是被设置为 “unavailable”。如果这些位没有被设置为 “supported”，Initiator 是可以将其映射到 “Unspecified”的。然而，此种情况是不被允许的，因为它们被设置成 “supported” 和 “unavailable”。然而 Initiator 仍然可以将与 “Alert”、“Notification” 或者其他 “unsupported” Audio Contexts 相关的 stream 重映射到 “Unspecified”，Acceptor 应该接受。

Context Types												
Emergency Alarm	Alerts	Ringtone	Notifications	Sound effects	Live	Voice assistants	Instructional	Game	Media	Conversational	Unspecified	
Bit	11	10	9	8	7	6	5	4	3	2	1	0
Available Audio Contexts	0	0	0	0	0	0	0	1	0	1	1	0
Supported Audio Contexts	1	0	1	0	0	0	0	1	0	1	1	1

图 7.3 “Unspecified” 设置成 “unavailable”的 Supported\_Audio\_Contexts 和 Available\_Audio\_Contexts 实例

最后，图 7.3 展示了 “Unspecified” 被设置成 “unavailable”的情况(记住，“Unspecified” 始终被设置成 “supported”)。将 “Unspecified” 设置成 “unavailable” 防止 Initiator 将任何其他 “unavailable” Context Type 映射到 “Unspecified”，所有，此种情况下，Acceptor 仅仅对于与 “Instructional”、“Media” 或 “Conversational” 相关的音频流是 “available”的。

图 7.8 从 Initiator 如何解释这些设置说了此行为。实际上，Acceptor 可以设置三个选项：

- 允许 Initiator 继续建立音频流
- 禁止其开启音频流，或者
- 说它不在乎——可以试一试。

这种 “Not Supported” 和 “Available” 组合是不允许的。

Supported Audio Contexts	Available Audio Contexts	Interpretation for Initiator
0b0	0b0	音频流 Context Type 可以被映射到 “Unspecified”
0b0	0b1	此种组合不允许
0b1	0b0	Initiator 不能以此种 Context Type 建立音频流

0b1	0b1	Initiator 可以以此种 Context Type 建立音频流
-----	-----	------------------------------------

表 7.8 Supported 和 Available Bluetooth LE Audio Context Type 设置结果

当 `Supported_Audio_Contexts` 特征对于所有 Client 都是合法时，Acceptor 可能向不同的 Client 公开不同的 `Available_Audio_Contexts` 特性值。这就允许 Acceptor 决定不同的 Initiator 可以做什么，比如控制哪些设备可以将流媒体发送给它或者为每个 Client 建立一个电话音频流。怎样完成取决于实现，而且由 Acceptor 来管理执行此操作所需的不同命名空间。实现时，它为 Acceptor 提供了基于用例来设置连接策略的方法，这在 Bluetooth Classic Audio profile 中是不可能实现的。

## 7.2 ASCS – the Audio Stream Control Service

PACS 明确了 Acceptor 公开其属性的方式，以及它在任何时间点可以做什么。对于大多数设备，对于 codec 设置和 contexts、是否作为 Source 和/或 Sink、以及它支持哪些 Audio Locations，对于这些选项它有很大范围选择性。这个范围需要缩小到单独的参数集来建立音频流，通常是因为当前的资源限制，比如当 Acceptor 支持多个音频流，它们可能需要不同的参数。这就是 Audio Stream Control Service 发挥作用的地方，它为每个音频流定义 Endpoint。Audio Stream Control Service 是在 Acceptor 上实现的，并且定义了 Initiator 如何使用 BAP 中定义的过程在两者之间配置和建立音频流。

### 7.2.1 Audio Stream Endpoints

Audio Stream Control Service 的核心是 Audio Stream Endpoints 的概念。一个音频流 Endpoint 或 ASE 被定义为 Acceptor 中 unicast 音频流的起点或目的地。Initiator 中不包含任何 ASE——它在 Acceptor 上配置 ASE。如果音频数据流入一个 ASE，它就是一个 Sink ASE。如果音频数据流出，它就是一个 Source ASE，两者都是由只读特征描述的。Acceptor 必须为它能够支持的每个音频流，公开至少一个 ASE。

Filed	Size (Octets)	Description
ASE ID	1	A unique ID for each client namespace
ASE_State	1	0x00=Idle 0x01=Codec Configured 0x02=QoS configured 0x03=Enabling 0x04=Streaming 0x05=Disabling 0x06=Releasing
State specific ASE Parameters	varies	Depends on the state (见 ASCS, 表 4-2 到 4-5)

表 7.9 ASE 特征格式

对于每个 ASE，Acceptor 为每个连接的 Initiator 维护一个状态机实例。

ASE 的状态是由 Initiator 来控制的，尽管有些情况下，Acceptor 也有自动地改变 ASE 的状态。Sink 和 Source PAC 为 Initiator 提供广泛的属性，而 Sink 和 Source ASE 代表了每个连接当前的状态和属性。可以通过 Sink 或 Source ASE 特征读取这些信息，会返回表 7.9 中所示的信息。

图 7.4 展示了 Sink ASE 的状态机。Source 的状态机稍微复杂一些，因为它包含一个附加的 Disabling 状态。我们将从 Sink ASE 状态机开始，因为直到 Streaming 状态，启用 ASE 的过程基本上是相同的，然后看看当我们到达 Disabling 状态时的差异。

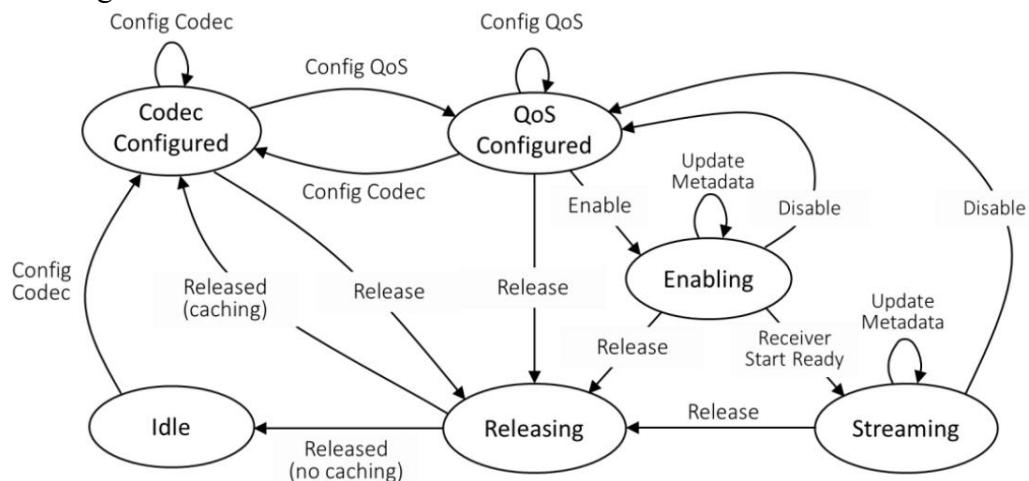


图 7.4 Sink ASE 状态机

ASCS 定义了 Initiator 如何通过这些状态调动 Acceptor 上的每个 ASE 的，方法是使用表 7.10 中列出的操作码，向 ASE Control Point 特征写入一系列命令。

Opcode	Operation	Description
0x01	Config Codec	配置 ASE codec 参数
0x02	Config QoS	配置 ASE 首选 QoS 参数
0x03	Enable	使用 CIS 参数并开始将 ASE 匹配到相应的 CIS
0x04	Receive Start Ready	完成 CIS 建立并发出 ASE 已准备好接收或发送音频数据的信号
0x05	Disable	开始将 ASE 从其 CIS 解除配对
0x06	Receive Stop Ready (Source ASE Only)	表示 Initiator 已经准备好停止接收数据并完成将 Source ASE 与其 CIS 解除配对
0x07	Update Metadata	更新 ASE metadata
0x08	Release	将 ASE 返回到空闲或 codec 已配置状态

表 7.10 ASE Control Point 操作码

操作码 0x01 (Config Codec) 和 0x02 (Config QoS) 可用于转换 ASE 的状态，或更新已经处于该状态的 ASE 配置。操作码 0x07 更新 metadata，但是不能改变状态。所有其他操作码都会导致在 ASE 状态机中转换到另一种状态。

Initiator 可以在 Acceptor 上的单个或多个 ASE 上执行 ASE Control Point

操作，只要这些 ASE 处于相同的状态。如果 CIG 包括一个或多个 Acceptor 上的多个 CIS，则 Initiator 应确保在将每个 Acceptor 上的 ASE 转移至 Enabling 状态之前已经从每个 Acceptor 上的 ASE 收集了相关信息。这是在复述 BAP 的描述，即以正确的顺序对 Coordinated Set 中的所有成员执行 BAP 过程。(大多数情况下，一个 CIG 中的所有 Acceptor 都是一个 Coordinated Set 的成员。然而，CAP 中是允许 ad-hoc set 的，此处 Acceptor 可以不是 Coordinated Set 成员的情况下被分配在一起工作，这种情况下，配置的顺序是实现来确定的)。

ASE 的一个特性就是，Acceptor 对于每一个与 Initiator 的连接都支持一个单独的 ASE 实例。如果有多个 Initiator 使用激活的 ACL 连接到 Acceptor，该 Acceptor 会为每一个 Initiator 维护一组的 ASE 值。为每一个 Initiator 维护不同的 ASE\_ID 命名空间。虽然 ASE\_ID 在每个命名空间中必须是唯一的，但是相同的 ASE\_ID 可以存在于不同的命名空间。意味着每个 ASE\_ID 和其相应的 handle 保持唯一。对其管理取决于实现。对一个命名空间中的 ASE 的更改不会影响不同命名空间中的相同 ASE 的状态(尽管该更改可能导致应用程序将 CIS 从一个 ASE 转移到另一个 ASE，但这是特定于实现的)。

如果 Acceptor 建立了与多个 Initiator 的连接历史记录，他们可以决定使用缓存的配置来维护一组 ASE。此种做法简化了从不同的 Initiator 转换到音频流的过程。尽管规范允许 Acceptor 同时与多个 Initiator 有音频流，实际上，那需要许多资源和复杂的调度，因此可能很少遇到，至少在 Bluetooth LE Audio 设备的初期实现中是这样。大多数情况下，利用缓存配置来实现简单的转换可能是更容易的解决方案。

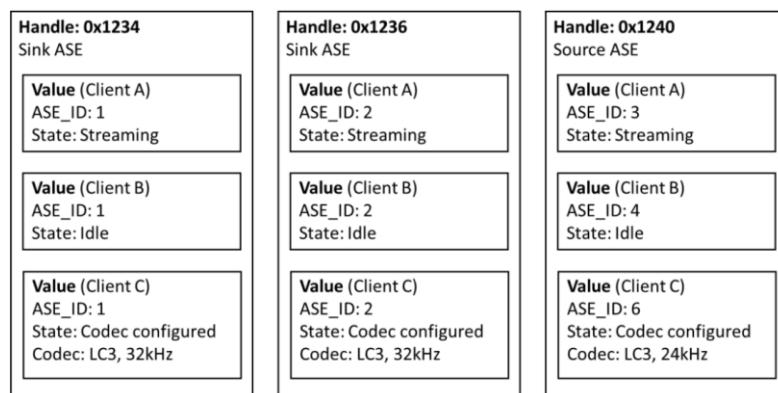


图 7.5 多 Client 公布的 ASE 状态实例

图 7.5 展示了这些原则是怎样运作的。展示了一个具有 3 个 ASE 的

---

Acceptor 如何为三个不同的 Client 公布其实例化状态的。当前，只有 Client A 与这些 ASE 建立了的 stream，它们的 handle 为 0x1234、0x1236、0x1240。如果 Client A 读取这三个 ASE，它会了解到这三个 ASE 都处于 streaming 状态。

如果 Client B 读取相同的 handle，它会得到三个 ASE 都是 Idle 状态。注意，Client B 的 ASE\_ID 可能是不同的，因为 Acceptor 为每个 Client 分配和维护了不同的命名空间。

最后，当 Client C 读取它们时，它会知晓这些 ASE 处于 Codec Configured 状态。这种现象通常是因为 Client C 在此之前已经启用了一组 stream，当它们被释放时，它要求 Acceptor 将 ASE 保持在 Codec Configured 状态，方便下一次 Client C 想要建立 stream 时加快连接。在此种状态下，codec 配置已经公布过，以便 Client C 分辨在将 ASE 转移到 QoS configured 状态之前是否需要执行重新配置。其他状态下，Codec Configuration 信息是不公开的。

Acceptor 必须对其支持的每个 Audio Stream 公布至少一个 ASE。如果它在每个方向上支持多个 Bluetooth LE Audio Channel，它必须为每个音频通道支持一个 ASE，不管它是否使用所有这些 ASE。如果支持多路复用，则每个方向上的 ASE 数量必须大于或等于该方向上设置为 0b1 的 Audio Location bits 的数量，再除以 Audio\_Channel\_Counts LTV[BAP 3.5.3]中设置的最大 Audio Channels 数量。如果一个 Acceptor 可以同时支持来自两个或多个 Client 的同步音频流，它需要为每个 Audio Stream 提供一个 ASE。更简单来说，Acceptor 必须支持它声称可以抛向它的所有东西，每个 Audio Stream 至少一个 ASE。

似乎在 PAC 记录的全局范围和 ASE 的更有限制的配置之间存在冲突，但是它们具有不同的用途和范围。为了理解 PAC 记录和 ASE 之间的关系，使用一个类比似乎有所帮助，我们再次用食物来打比方。想象在一家餐厅，Published Audio Capabilities 就像是厨房里一份完整的材料清单。ASE 是独特的菜肴，它只使用了这些材料的一部分。餐厅的期望是人们从菜单中选择菜肴，因为它们通常是为了适应场景而开发出来的，在餐厅里，可能是早餐、午餐、下午茶或晚餐。然而，有时候客户可能会提出不同的要求。如果一位食客进来要一个汉堡，里面有薯条、土豆泥、烤面包、意大利面和蛋羹(我希望这不是任何菜单上的选项)，BAP 认为顾客总是对的，并允许这么做。为了限制这种情况，公布与顶层 profile 支持的用例相对应的特定的 PAC 记

---

录可能会有所帮助。在这个餐厅的类比中，一天中不同的时间会有不同的菜单。

在状态机中移动涉及 BAP 发出命令和 ASCS 响应的相互作用。与其孤立地讨论 ASCS 的细节，不如看看建立 stream 的实际交互过程。

### 7.3 BAP – the Basic Audio Profile

ASCS 描述了 ASE 的状态，但并没有描述它们的过程，因为它们是驻留在 Server 上的服务。它是 Acceptor 上每个 Audio Stream Endpoint 当前状态的陈述。要执行使我们通过 ASE 状态机来配置 ASE 并启用 CIS 的转换，我们需要转到定义 Client (Initiator) 行为的 Basic Audio Profile 上。BAP 为单播和广播定义了这些过程，CAP 将它们捆绑在一起。在本章中，我们将仅限于单播过程。在下一章，我们将对广播进行同样的讲解过程。

设立 Source ASE 基本上是相同的过程，但有个额外的状态，我们将在最后介绍。

#### 7.3.1 Moving through the ASE state machine

在 ASE 状态机中转换适应标准过程。Initiator 向 Acceptor 的 ASE Control Point 特征发送一个命令，明确要执行的控制操作(比如，它想让 ASE 转换成哪种状态，或者哪种状态需要更新)。该命令指定了本身将应用于哪个 ASE 或 ASEs 以及该特定的状态转换的参数。

该命令采用了如表 7.11 所示的形式，并且包含了一组针对每个状态的操作参数。在我们整理整个状态机配置过程中，我们将查看每一组特定于操作的参数。

Filed	Size (Octets)	Description
Opcode	1	Control Point Operation for the next state or update, shown in Table 7.10
Number of ASEs	1	Total number of ASEs included in this operation
ASE_ID[i]	1	The IDs of those ASEs
Operation Specific parameters	Varies	A list of parameters for each of the [i] ASEs

表 7.11 Initiator 的 ASE Control Point 命令操作的基本结构

在每个阶段，Initiator 都了解 Acceptor 的 capabilities。只有不请求 Acceptor 提供的功能之外的功能，Initiator 就可以期望 Acceptor 遵守其所有的 ASE Control Point 命令值。一旦接收到每个 ASE Control Point 命令，Acceptor 将以 ASE Control Point 特征值通知进行响应，对命令中所包含的 ASE\_IDs 的[i]通知成功与否。如表 7.12 所示。

如果其中任何一个被拒绝或有错误，则会有一组完整的错误响应通知 Initiator，以便它可以重试。注意，所有这些操作都是针对[i] ASEs 数组的。

它们可以作为每个 ASE 的单独命令发送，但是在在一个操作中包含每个 Acceptor 的所有 ASE\_ID 更有效率。

Filed	Size (Octets)	Description
Number of ASEs	1	Total number of ASEs included in this operation
ASE ID[i]	1	The IDs of those ASEs
Response Code[1]	1	0x00 if successful, otherwise an error response code from Table 5.1 of ASCS
Reason	1	An extended reason code from Table 5.1 of ASCS. 0x00 indicates success. Other codes provide reasons where the operation has failed.

表 7.12 通知到 Client 的 ASE Control Point 特征格式

假如没有问题，Acceptor 继续使用 Initiator 在其 ASE Control Point 命令中提供的值来更新其所有 Sink ASE 和 Source ASE 特征状态。如果这导致了任何 ASE 的改变，Acceptor 会对每个已改变的 ASE 的新的值进行通知。

由于 ASE Control Point 命令的目的是通过改变 ASE 的状态或更新参数值来实现对 ASE 的改变，因此 Initiator 会期望收到这些通知。然而，不是所有的 ASE 都需要在每个 ASE Control Point 操作中进行转换或更新。意味着 Acceptor 上与同一 Initiator 关联的不同 ASE 可能处于不同状态。如有疑点，Initiator 应读取它们的 ASE 特征。(如果使用了 CAP 音频流过程，则不应出现这种情况，因为这需要在继续执行下一个命令之前将所有 Acceptor 转换为相同的状态。然而，即使在此处，如果将某些 ASE 配置成为“空闲的” ASE 以供将来使用，则它们可能不会处于“Enabling”或“Streaming”状态，我们将在第 7.6 节中介绍。)

此过程的简化序列图如图 7.6 所示。相同的过程适用于 Sink ASEs 和 Source ASEs。

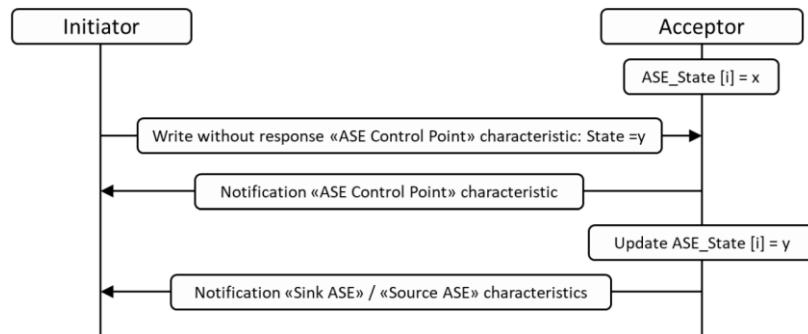


图 7.6 ASE Control Point 操作简化序列图

此过程的巧妙(或复杂，取决于你的观点)之处在于，当你在建立 stream 过程中转换时，ASE Control Point 命令和 Sink ASE 与 Source ASE 特征对于每个状态都是不同的，这为 Initiator 提供了下一步操作所需要的信息。前几个操作中的不同参数允许 Initiator 了解 ASE 的所有功能，并收集建立 CIG 及其组成 CIS 所需的信息。

---

Initiator 的所有操作都具有相同的基本格式，如表 7.9 所示，但 State Specific ASE 参数锁着每一个 ASE\_State 而变化。ASCS 表 4-2 至 4-5 中定义了 ASE 特征 notification，第 5.2 至 5.6 节定义了 ASE Control Point command。

Initiator 可以使用其 ASE Control Point 操作命令配置多个 ASE，但是每个配置的 ASE 都独立地通知其状态。例如，如果 Initiator 要为耳机启用 4 个 ASE，分别为左右立体声启用两个 Sink ASE，为两侧麦克风启用两个 Source ASE，则它将从更新的 ASE Control Point 特征收到一个通知，以及从两对 Sink 和 Source ASE 来的四个独立的通知。

ASCS 状态机中的每一步都定义为 BAP 中的特定过程。在下一节中，我们就通过这些来了解该过程如何工作的。

## 7.4 Configuring an ASE and a CIG

想要建立连接的 Initiator，无论是响应用户操作、外部事件、还是 Acceptor 的请求，都需要首先确定使用场景中涉及的所有 Acceptor 的 capability。假设这是设备的第一次连接，并没有缓存任何内容，Initiator 可以有各种选项来确定 Acceptor 的功能，具体取决于 Initiator 是否只需要 BAP 规定的基本 capability，是否还要另外的顶层 profile 规定的功能，这样意味着它也需要遵循 CAP，或者它是否希望使用可选的或专有的功能。这些选项展示在表 7.13 中。在后续的连接中，可能会有关于 Acceptor capability 的缓存信息，从而允许 Initiator 跳过这些步骤。

Initiator Action	Result
Discover Sink PAC	Acceptor can receive unicast audio with BAP mandatory settings
Discover Sink Audio Location	Acceptor can receive unicast audio with BAP mandatory settings
Discover Sink ASE	Acceptor can receive unicast audio with BAP mandatory settings
Discover Source PAC	Acceptor can transmit unicast audio with BAP mandatory settings
Discover Source Audio Location	Acceptor can transmit unicast audio with BAP mandatory settings
Discover Source ASE	Acceptor can transmit unicast audio with BAP mandatory settings
Discover Profile or Service	Acceptor supports additional mandatory requirements
Read Sink PAC	Discover the Acceptor's capabilities for reception
Read Source PAC	Discover the Acceptor's capabilities for transmission

表 7.13 Initiator 确定 Acceptor capability 的选项

让我们假设我们正在处理 TMAP 或 HAP，并想连接到左右一对 Acceptor。在发现至少一个 Acceptor 支持 HAP 或 TMAP 后，通过发现它们的 UUID，Initiator 知道所支持的强制 codec 配置和 QoS 设置，因此可以直接跳转到 CAP 并为 non-bonded 设备 [CAP 8.1.2] 运行 CAP 连接过程以此建立连接。Initiator 会确定耳塞是否是 Coordinated Set 的成员，如我们的示例所示，然后运行 CAP procedure preamble [CAP 7.4.2]，与该集合中的两个设备绑定。完成后，它可以使用 CAP Unicast Audio Start procedure [CAP

---

7.3.1.2]启动建立连接的主要任务。这会告知 Initiator 逐步完成底层 BAP 过程。

首先，Initiator 需要查明每个 Acceptor 上的 Sink 和 Source ASEs，并且确保每个 Acceptor 至少有一个支持其要建立的每个音频流的恰当方向的 ASE。它同样需要检查 PAC 特征，以确保 Acceptor 能够支持它想要使用的设置。这些过程在 Audio role discovery procedure [BAP 5.1], Audio capability discovery procedure [BAP 5.2]，以及 ASE\_ID discovery procedure[BAP 5.3]的这些 BAP 中有描述。

Audio role discovery 以及 ASE\_ID discovery 都是强制性的，因为除非你知道有合适的 ASE 可连接，否则无法继续。Audio Contexts discovery [BAP 5.4]是可选的，但是如果 Initiator 想要使用可选设置，则是必需的。

BAP 的设计目的是提供一个 fall-back 级别的互操作性，使每个 Initiator 能够以一个合理的音频质量与任何 Acceptor 建立音频流，以确保他们始终能够连接。如果 Initiator 仅仅使用 BAP 或其已经发现 Acceptor 支持的 profile 的强制性设置，它可以有足够的信息跳过此步骤。类似地，如果连接请求来自 Acceptor，直接地，或通过 Announcement，则 Acceptor 可能已经提供了该信息。

下一步就是配置每个 ASE 的 codec。在那之前，Initiator 应该通过使用 Supported\_Audio\_Contexts procedure [BAP 5.4]和 Available\_Audio\_Context procedure [BAP 5.5]，来检查 Acceptor 的 Supported\_Audio\_Contexts 和 Available\_Audio\_Contexts 特征，确认每个 ASE 都支持预期的用例。这些已经在 7.1.5 中描述过。

#### 7.4.1 The BAP Codec Configuration procedure

Parameter	Description
Target Latency [i]	0x01 = low latency <sup>1</sup> 0x02 = balanced latency and reliability 0x03 = high reliability <sup>1</sup>
Target PHY [i]	0x01 = 1M PHY 0x02 = 2M PHY 0x03 = Codec PHY
Codec ID [i]	Codec configuration (ID, length and configuration), as described in Section 7.1.1-Sink PAC
Codec Specification Configuration	and Source PAC characteristics

<sup>1</sup> 这些术语是通用的，不一定与 BAP 中定义为低延迟或高可靠性的配置一致。

表 7.14 配置 codec 操作的状态特定参数

假设必要的 Sink 和/或 Source ASE 需求都满足，Initiator 会通过配置每个 Acceptor 上的 ASE 来建立 Isochronous Streams。它是通过利用 BAP Codec Configuration Procedure [BAP 5.6.1]来实现这一点，在该过程中，它以 0x01

---

的 Cofig\_Codec 操作码写入 ASE Control Point，以便将每个 ASE 转变为 Codec Configured 状态。Initiator 已经确定了支持哪些 codec 配置，因此它现在为当前应用程序选择所需的配置，并将其与目标延迟和 PHY 一起发送。此操作的参数格式已经定义在 ASCS 中的表 5.2，并且总结在了表 7.14 中。

这里有许多点要强调。**第一点**，对于每个音频流，这些参数不需要相同。对于双向 CIS，每个方向的参数，甚至 PHY 和 codec 都可能不同。在大多数情况下，它们不是，但是如果你在一个方向是播放音乐，并使用返回方向进行语音控制，它们很可能是。目前，大多数 profile 要求 Bluetooth LE Audio 运行在 LE 2M PHY 上。

**第二点**，前两个前缀为“Target”的参数是建议值。Acceptor 将使用这些建议值来选择其认为满足需求并适合当前状态的 QoS 参数，然后此命令的响应中返回这些选项。

相反，Codec ID 和 Codec Specific Configuration 参数是事实陈述，以及 ASE 使用特定值的指令。Codec Configuration 通常基于顶层的 profile，该 profile 基于 BAP 中指定的强制选项。我们已在第 5 章(Codec and QoS)中研究了 BAP 中定义的强制的和可选的 QoS 配置，以及另外的 profile。在大多数情况下，Initiator 将选择这些预定义配置中的一种，尽管它也可以使用顶层 profile 中定义的附加 codec 设置及其建议的 QoS 设置，或制造商特定的 codec。它也可以自行配置，尽管这样会有互操作性差的风险。预定义配置已经经过了大量测试，以确保它们性能良好，并且应该始终是首选的选项。

实现者应该注意，有两种听起来非常相似的 Codec Specific LTV 结构。第一种，Codec\_Specific\_Capabilities LTV 结构，用于 PAC 记录，通常表示一系列 capability。第二种，Codec\_Specific\_Configuration LTV 结构，用于上述的 Codec Configuration 操作，并且用于单个特定配置。各个参数的语法略有不同。

在整个过程中，有一定程度的设备相互围绕，提供建议，以便其他设备能够提供最适当的当前资源状态的信息。这不是真正的协商，最好将其描述为信息枚举，允许 Initiator 通过发送给它的 Controller 的命令来配置每一个 CIS。即使如此，其 HCI 命令中的一些参数也是建议值，Controller 对如何配置 CIS 具有最终决定权。

需要注意的**第三点**也是重要的一点，虽然 Acceptor 通过 PAC 记录通报其可以支持的 codec 参数，但 Initiator 写入特定的值，这些值定义了如何配置编码音频流。其中的一些在 Acceptor 发送的、支持的设置中和 Initiator 写

---

入的、实际的设置中定义了不同的。比如，如上所述，Supported\_Sampling\_Frequencies LTV 是一个位域，而 Sampling\_Frequency LTV 是映射到采样频率的特定值。因此，如果 ASE 仅支持 16kHz，则 Acceptor 将通告一个 0x04 的值。作为响应，Initiator 将通过写入 0x03 来配置 ASE codec 以支持 16kHz 采样。但是，回到这个过程...

获得了有关 ASE、其 capability 和 availability 的信息之后，Initiator 可以发出一个操作码为 0x01 的 Write without Response 命令，该命令包括它想在 Acceptor 上配置的所有 ASE\_ID 的数组，该数组可包含两个方向。Acceptor 将通过更新并通知 ASE Control Point 特征，包括适当的响应代码（如果 OK，则为 0x00）。然后，Acceptor 将所有特定的 ASE 转换为 Codec Configured 状态。一旦完成，Acceptor 将通告每一个 ASE 的新状态，包括下列 state-specific 参数（总结在了表 7.15 中），其在 ASCS 的表 4.3 中描述。

Filed	Description
Framing	Support for framed or unframed ISOAL PDUs
Preferred PHY	A bitfield of values. Normally includes 2Mbps
Preferred Retransmission Number (RTN)	How many times packets should be retransmitted
Maximum Transmit Latency	The maximum allowable delay for Bluetooth transport
Presentation Delay Min&Max	Supported range of Presentation Delay
Preferred Presentation Delay Min&Max	Preferred range of Presentation Delay
Codec Configuration	The codec configuration for this ASE

表 7.15 Config\_Codec 操作符后的通知参数(ASCS 的表格 4.3)

BAP codec 配置过程就结束了，我们将转移到 QoS 配置过程[BAP 5.6.2]

#### 7.4.2 The BAP QoS configuration procedure

在 codec 配置过程的最后，Initiator 的 Host 会将每个 Acceptor 在 codec 配置过程结束时通知的配置参数与其当前应用程序的需求进行比较，并决定在所告知的限制范围内使用哪些值。

在发送 Config QoS 命令之前，Initiator 最好使用 LE Set CIG Parameters HCI 命令将这些参数发送给其控制器。这可以确认所选的参数集是被支持的。请记住，这些是与用例相关的建议，用于通知 Controller 中的调度算法。Controller 所选的实际值可能略有不同。此 HCI 命令使用的参数如表格 7.16 所示。此命令的全部细节在 Core, Vol 4, Part E, Section 7.8.98 中。

Parameter	Description
CIG_ID	Assigned by the Initiator's Host
SDU Interval C To P	Time interval between SDUs generated from the Initiator's Host
SDU Interval P To C	Time interval between SDUs generated from the Acceptor's Host
Worst Case SCA	Worst case Sleep Clock Accuracy of all the Acceptors in the CIG
Packing	Preferred packing (sequential vs interleaved)
Framing	Framed if set to 1, otherwise up to the Controller on a per-CIS basis
Codec Configuration	The codec configuration for this ASE
Max Transport Latency C To P	The maximum transport latency for each direction.
Max Transport Latency P To C	
CIS Count	The number of CISes in this instance of this command
CIS ID[i]	Unique ID for each CIS in this CIG

Max SDU C To P[i]	The maximum sizes of SDUs from the Initiator's and Acceptor's Hosts.
Max SDU P To C[i]	
PHY C To P[i]	Bitfield indicating the possible PHYs to use for each direction. If more than one bit is set, the Controller decides.
PHY P To C[i]	
RTN C To P[i]	The Preferred number of retransmissions in each direction. The Controller can ignore this.
RTN P To C[i]	

表 7.16 LE Set CIG Parameters HCI parameters

在表 7.16 中，斜体字突出显示的参数是对控制器的建议，控制器可以选择忽略这些建议。

如我们在第 4 章看到，应用程序不能强制为 Link Layer 设置特定的值。重申这里发生的情况，Initiator 和 Acceptor 交换了信息，这些信息允许 Initiator 决定在 HCI 命令中输入什么，以指示 Core 调度最符合应用程序要求的 CIS。(Bluetooth 实现使用 HCI 命令进行测试，以检查它们是否合规，但无需在生产产品中公开。芯片和协议栈供应商可能提供备选的 API 来提供此功能，但了解 HCI 命令有助于演示该过程的工作原理。)

LE Set CIG Parameters HCI 命令中需要两个值(图 7.16)，这两个值并不是来自 ASE 配置过程。**第一个是 Worst\_Case\_SCA**，就是所有 Acceptor 中最差的睡眠时钟精度。发送该命令之前，Initiator 需要通过索取每个设备的睡眠时钟精度(通常通过使用 LE\_Request\_Peer\_SCA HCI 命令)，来确定该值。

**第二个是 Packing** 参数，该参数决定 CIS event 按照 sequentially 还是 interleaved 来发送。如果设置为 0，它们将按照 sequential 方式发送，如果是 1，它们按照 interleaved 方式。

通常，Framing 参数值设置为 0，以便 Controller 可以决定，但是某些 profile(如 HFP)，可能要求在某些情况下将其设置成 1，以强制它们 framed。BAP 的表 5.2 中建议的 unicast Audio Stream 的 QoS 设置表明，除了那些采样频率为 44.1kHz 的应该要 framed 之外，所有的 Audio Stream 都应该是 unframed。

在此阶段，Controller 不会将其选择的值通告给 Initiator 或 Acceptor，信息将在 CIS 建立时传达。相反，Initiator 的 Host 只能得到其请求的配置可以被调度的确认，以及每个 CIS 的 Connection Handle。

一旦收到 HCI\_Command\_Complete 事件，确认可以调度 CIS，Initiator 应将操作码为 0x02 的 ASE Control Point 操作命令依次发送给它的每个 Acceptor，此操作的特定参数定义在 ASCS 中的表 5.3，如下表 7.17 所示。这会使 ASE 转变到 QoS configured 状态。Initiator 在 Config QoS 命令中使用的值复制了它在 HCI LE\_Set\_CIG\_Parameters 命令中使用的值。

Parameter	Value
Number of ASEs	i (must be at least 1)

CIG_ID	Value used in the HCI LE_Set_CIG_Parameters command (Note: At this stage, these may not be the values which the Controller has actually scheduled.)
CIS_ID [i]	
SDU_Interval [i]	
Framing [i]	
PHY [i]	
MAX_SDU [i]	
Retransmission_Number [i]	
Max_Transport_Latency [i]	
Presentation_Delay [i]	
	Value which the Acceptor will use for rendering or capture

表 7.17 与 0x02 Config QoS 操作符一起使用的 CIS 和 Presentation Delay 值

Presentation Delay 是唯一未包含在 HCI LE Set CIG Parameter 命令中的参数，而是直接发送给 Acceptor。在几乎所有的应用中，Presentation Delay 的值，对于所有的 Sink ASE 都是相同的。每个 Sink ASE 的 Presentation Delay 的值相同的原因是，其决定了音频呈现的时间点。对于耳塞和助听器来说的，呈现的时间点总是一样的。在某些情况下，例如在一个会议室或礼堂中有多个扩音器，就可能需要分配不同的值来解决它们在房间中相对位置不同造成的延迟，但这不常见。

如果有一个或多个 Source ASE，它们可能使用与 Sink ASE 不同的 Presentation Delay 值，不仅仅因为作为 Presentation Delay 一部分的所有 LC3 编码时间明显大于解码时间(大约 13ms，而不是 2ms)。因此，需要更多的时间来编码。然而，所有的 Source ASE 通常使用彼此相同的值。

Initiator 为呈现所选择的 Presentation Delay 值应在 Sink ASE 公开的 preferred Presentation Delay 范围之内，并且 capturing 值应在 Source ASE 公布的范围之内。这些值不应超出 Acceptor 集合在每个方向上公开的 Max 和 Min 值的共同范围，理想情况下应在 preferred Max 和 Min 值的范围内(见表格 7.15)。如果 Context Type 是“Live”，则应使用共同的最小值。

如前所述，Initiator 在 Write without Response 命令中发送这些消息，之后它接收更新过的 ASE Control Point 特征的通知，确认每个 ASE 已经转变成了 QoS Configured 状态，随后是每个 ASE 的更新 ASE 特征的通知。这些通知中的参数显示了先前 ASE Control Point 操作中设置的值：

Parameter	Value
CIG_ID	Values set by the Initiator in its ASE Control Point operation (Note: At this stage, these may not be the values which the Controller has actually scheduled.)
CIS_ID [i]	
SDU_Interval [i]	
Framing [i]	
PHY [i]	
MAX_SDU [i]	
Retransmission_Number [i]	
Max_Transport_Latency [i]	
Presentation_Delay [i]	
	Value which the Acceptor will use for rendering or capture

表 7.18 Acceptor 在收到 Config QoS 操作符(0x02)之后返回的参数

再次重复一下显而易见的一点，Initiator 应在进入下一个状态之前，应遍历其打算在所有 Acceptor 上使用的所有 ASE 的每个状态，即，将它们全

---

部带到 Codec configured 状态，然后再到 QoS configured 状态，等等。这完全是常识，因为 Initiator 需要从它们获取信息，以确保为跨越 Acceptor 的、所有 ASE 设置配置参数。不过，可以肯定的是，CAP 强制要求这样做。

如果多个 Acceptor 或 ASE 没有提供一致的参数集，Initiator 可以为其中的一个或多个重复 Config 和 QoS 配置过程的每个步骤，包括从 QoS configured 状态返回 Codec Configured 状态。然而，预期该过程通常是一个单步过程，仅对每组 ASE 执行依次，因此如果它保持所有状态变化同步，则效率更高。因此，此过程的步骤如下：

- Initiator 发送向多个 ASE 发送带有目标值和显示值的命令
- Acceptor 通过通知其 ASE Control Point 特征来确认所有这些 ASE 的新状态(可能包含故障代码)。
- Acceptor 使用 ASE 特征分别通知每个 ASE 的首选值。(此处说明已配置的内容，以及下一个配置步骤的首选项。)

任何时候，Initiator 都可以检查单个 ASE 特征。参数表示 ASE 的状态，以及与该状态相关的值。如果 Initiator 在 Idle 状态读取 ASE\_ID，则它将接收到的唯一信息是 ASE\_ID 和 ASE\_state=0，表示空闲状态。读取 ASE 特征的最常见原因是在未收到预期通知时确认先前的 ASE Control Point 操作。

当 Initiator 的 Host 向它的 Controller 发送 HCI LE Set CIG Parameters 命令时，它会将 CIG 转到它的 Configured 状态。此时，Initiator 的 Host 仍可以通过使用它的数组结构，对特定的 CIS，重新发送 HCI LE Set CIG Parameters 来修改 CIS 的特性或向 CIG 增加更多的 CIS。如果这样，则需要使用 ASE Control Point 特征的数组能力来写入 ASE，使用 Config QoS 命令更新相关 ASE。

一旦所有的 ASE 都处于 QoS configured 状态，并且 CIG 已配置，我们就可以开始启用音频流。

#### 7.4.3 Enabling an ASE and a CIG

配置完所有的 ASE 之后，Initiator 会拥有指导其 Controller 使能 CIG 和 CIS 所需的所有信息。Acceptor 的 Host 了解了 Isochronous Channel 的主要参数，尽管有些值是临时的，因为实际的 RTN 值取决于调度。如果 Controller 还需要支持其他无线连接，它可能会决定妥协，以此在不同的无线连接之间分配广播时间，并设置一个低于其 Host 请求的值。在建立每一

---

个 CIS 时，将在链路层级别配置这些设置，并通知 Initiator 和每个 Acceptor 的各自的 Host。

Initiator 现在可以使用表 7.19 中所示的参数，通过将操作码设置为 0x03(Enable)写入每一个 Acceptor 的 ASE Control Point 特征，调用 BAP 5.6 中定义的启用 ASE 的过程。从此刻开始，Initiator 就不能更改 CIG、CIS 或 ASE 配置了，除了 Audio Stream metadata。

Parameter	Value
Number of ASE	Total number of ASEs to be enabled (i)
ASE ID [i]	The specific ASEs which are being enabled
Metadata Length [i]	Length of metadata for ASE [i]
Metadata [i]	LTV formatted metadata. This is most commonly the Streaming_Audio_Contexts and CCID_List LTV structures, can include other LTV structures.

表 7.19 Enabling 操作状态特定参数——操作码 0x03(ASCS，表 5.4)

CAP 要求 Enabling 操作中的 metadata 包含 Streaming\_Audio\_Contexts，并为每个 unicast Audio Stream 设置正确的值。如果存在与 Audio Stream 相关的内容控制过程，则 Enabling 命令中的 Metadata 还必需包含 CCID\_List LTV 结构。[CAP 7.3.1.2.6]。

每个 Acceptor 会通知其 ASE Control Point 特征，将其 ASE 转换到 Enabling 状态，然后依次通知 ASE Control Point 命令中指定的每个 ASE 的 ASE 特征，返回其 CIG 和 CIS ID，以及 Initiator 写入的 metadata。如表 7.20 所示。

Parameter	Value
CIG_ID	Value set by the Initiator in its previous Config QoS operation
CIS_ID	
Metadata Length	(0 if There is no metadata for this ASE)
Metadata	LTV structures for Sink or Source ASE

表 7.20 Enabling 和 Disabling 时的 ASE 特征的附加参数

ASE 的 metadata 值只能被 Initiator 设置或更新。Acceptor 不能对这些进行更改，即便它充当 Audio Source。Acceptor 可以随时更新其 Available\_Audio\_Contexts 值，但这些值不会出现在 ASE metadata 中。它们也不会导致当前流的终止。Available\_Audio\_Contexts 中的任何更改仅仅用于后续连接尝试。既然 ASE 处于 Enabling 状态现在是启用所需的 CIS 并将其耦合到 ASE 的时候了。请记住第 4 章中的内容，CIG 中已配置的每个 CIS 并不是都需要建立，因为 Initiator 可以配置 CIS，当前需求发生变化时，可以交换这些 CIS。Initiator 通过调用 Core [Vol 3, Part C, Section 9.3.13] 中的 Connected Isochronous Stream Central Establishment 过程来使能它需要的 CIS。Link Layer 会向 Acceptor 发送 CIS\_Request，该请求使用 Core [Vol 3, Part C, Section 9.3.14] 中的 Isochronous Stream Peripheral Establishment 过程

---

来建立 CIS。此时，Initiator 将开始传输零长度的 PDU。如果是双向的 CIS，则需要建立 Sink 和 Source ASE。

此刻，ASE 的行为存在一个实现者需要注意的条件。如果是在 Enabling 操作之前存在的 CIS，则其状态从 Enabling 到 Streaming 没有通知。相反，Acceptor 会自动将 ASE 转换为其 Streaming 状态，而无需 Initiator 写入“Streaming”命令。如果 Acceptor 正在重用现有的配置，并且原始 CIG 从未被禁用，则最有可能发生这种情况。

正如我们在第 4 章的 CIG 状态机中看到的，一旦启用了 CIG，就不能更改 CIS 的配置，因此，对 codec 的配置，QoS 参数或 Presentation Delay 的任何更改都需要终止 CIG 并重新启动配置过程。可以断开或恢复单个 CIS(使用 LE Create CIS 命令)，但是只有当它们处于“Enabling”或“Streaming”状态时，才能更新它们的 metadata。

#### 7.4.4 Audio data paths

Isochronous Streams 现在已经启动并运行，但我们尚未连接任何音频数据——Isochronous Channels 正在传输空数据包。如果尚未完成，则下一步是为每个 ASE 连接数据路径，使用 Audio Data Path Setup Procedure (BAP 5.6.3.1)，然后依次使用 LE Setup ISO Data Path HCI 命令。

Bluetooth LE Audio 为音频数据路径和 codec 位置提供了很大的灵活性。Codec 可以放在 Host 或者 Controller，音频数据可以通过 HCI，或者更典型地通过 PCM 或厂商专有的路由。

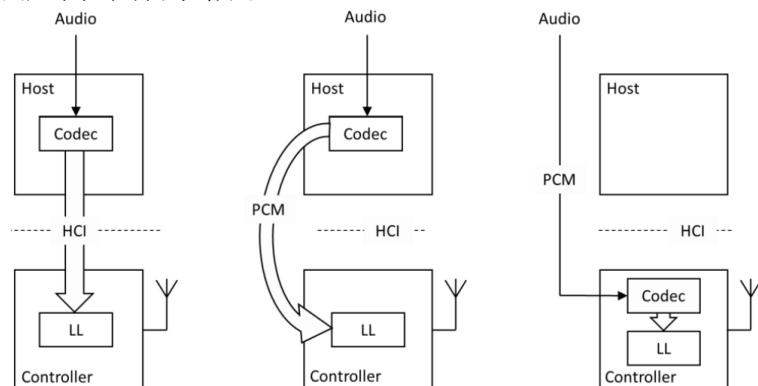


图 7.7 通用 Audio Data Path 配置

图 7.7 展示了 3 个最常用的数据路径配置，可见 codec 可以在 Host 或 Controller 中实现。音频通常通过 PCM 接口从 codec 路由，但是，如果在 Host 中编码，则它也可以通过 HCI 传输。要设置每个数据路径，Initiator 和

Acceptor 都将使用 LE Setup ISO data path HCI 命令将在 Config Codec 状态期间写入的 codec 配置绑定到每个 CIS 的 Connection Handle，根据是 Source ASE 还是 Sink ASE 指定方向。由于 codec 位置和数据路径实现通常取决于所使用的特定芯片组，因此 Initiator 和 Acceptor 中的数据路径配置可能不同。数据路径设置是在每个设备的内部，这是一种实现选择。这种级别的详细信息通常在更通用的 API 下面，对应用开发人员是隐藏的，但对于了解配置过程的每个步骤发生的情况是很有用的。

到目前为止，该过程对于 Sink 和 Source 端的 ASE 都是通用的。此时，这两个过程出现了分歧。它们在状态机中仍然处于相同的路径上，但命令的顺序有所不同。

**对于 Sink ASE**，一旦 Acceptor 成功建立它的数据路径，它可能会自动地将 ASE 转到 Streaming 状态，然后将每个 Sink ASE 特征通知到 Initiator(ASE 状态码被设置成 0x05(Streaming))。一旦 Initiator 收到此通知就可以开始向 ASE Sink 发送音频数据包了。

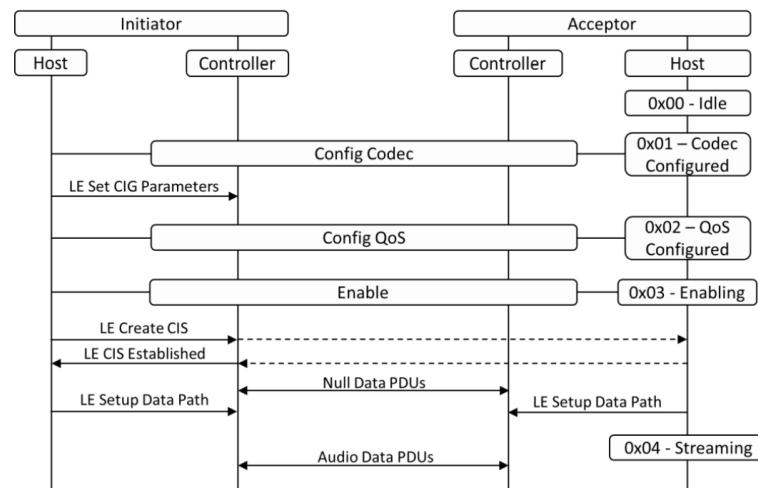


图 7.8 Sink ASE 的建立，展示 ASE 状态

图 7.8 展示了一个非常简化的建立 Sink ASE 的消息序列图，呈现了整个序列以及 Acceptor 自动转换到 Streaming 状态。BAP 的 5.6.3 节提供了更细节的 MSC。

**对于 Source ASE**，Acceptor 需要等待 Initiator 确认其已准备好从该 ASE 接收数据。一旦它从 Acceptor 收到 ASE 特征通知，并且准备好发送音频数据，Initiator 将向 ASE 写入 ASE Control Point 特征，操作码设置为 0x04(Receive Start Ready)。(请注意，Sink ASE 不需要包含在此命令中的

ASE 数组中，因为它们将独立地转换到 Streaming 状态)。此时，Acceptor 将 Source ASE 转换为 Streaming 状态，通知其 ASE 特征并开始音频流。

与 Source ASE 相关的 MSC 在图 7.9 中展示。此处，Initiator 需要向 Acceptor ASE Control Point 特征发送 Receive Start Ready 命令，以此开启音频数据包传输。

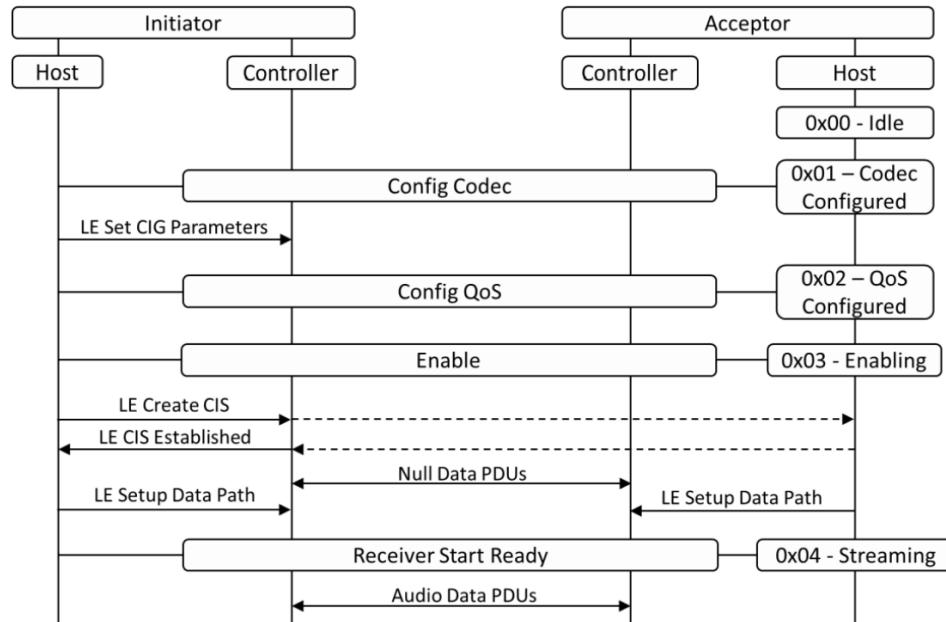


图 7.9 Source ASE 的建立，展示 ASE 状态

#### 7.4.5 Updating unicast metadata

当 Initiator 处于 Enabling 或 Streaming 状态时，它可以通过向 ASE Control Point 写入操作码 0x07(更新 metadata) [BAP 5.6.4] 来更新任何 ASE 的 metadata。这是一种方便的方式，可以将 ASE 用于不同的音频应用程序，而无需拆除和重新建立任何音频流。

这在 CAP Unicast Audio Update procedure [CAP 7.3.1.3] 中进行了描述，允许重用 ASE，保持其当前配置，但是更改 Context Type 和 CCID\_list metadata。一个典型的应用就是使用同一个 stream 从电话转到相同电话上的音乐播放器。存在固有的限制，即无法更改 codec 配置。另一个问题是电话是双向的，而音乐流是单向的。然而，在配置过程中，可以配置足够的 ASE 来应对此问题。在此用例转换的时刻，Initiator 可以禁用和启用 ASE，直到它拥有所需的已配置 ASE 数量，并在过程中更新其 metadata。注意，在双向 CIS 用例中，ASE 可以被禁用，但是 CIS 将保持启用状态以支持另一

---

方向及其 ASE。这是一个很好的例子，说明了如何通过为 Audio Stream 重新调整和重用增加灵活性来处理 Bluetooth Classic Audio multi-profile 问题。

#### 7.4.6 Ending a unicast stream

结束 unicast stream 的过程时 Sink 和 Source ASE 状态机发生分歧的地方。我们将在下一节中介绍 Source ASE。不同之处在于，Source ASE 具有禁用状态，而 Sink ASE 没有。

要停止流向 Sink ASE 的 stream，ASE 需要使用 CAP Unicast Audio Stop procedure [CAP 7.3.1.4]。转换回 QoS Configured 状态。它会调用 BAP Disabling ASE 过程[BAP 5.6.5]，在此处，Initiator 使用 0x05(Disable)命令写入 ASE。此时，Initiator 停止向 ASE 发送音频数据。如果 CIS 是双向的，并且 Source ASE 还没有被禁用，则 Initiator 将继续发送 null PDU，以允许 Acceptor 返回其音频数据。

此时，相关的 CIS 不会自动被禁用。如果 Initiator 想要删除它，它应该使用 HCI\_Disconnect 命令[Core Vol 4, Part E, 7.1.6]。由于禁用 Sink ASE 仅仅是将其移动到 QoS configured 状态，因此可以在稍后通过将 Enable 操作码写入到 ASE Control Point 特征来重新建立 ASE。如果已经使用 HCI Disconnect 禁用 CIS，则仍可以使用 HCI LE Create CIS 命令恢复。

如果没有重用 ASE 的意图，则可以通过执行 BAP Releasing an ASE procedure [BAP 5.6.6] 将其转换到 Releasing 状态。一旦 ASE 处于 Releasing 状态，Acceptor 会自动将其转换为 Idle 状态，或者，如果它想简化下一个 ASE 配置周期，可以通过将 ASE 返回到 Codec configured 状态来缓存 codec 配置。

一旦禁用了与 CIS 相关的所有 ASE，Initiator 就可以禁用任何处于启用状态的 CIS 并拆除相关的数据路径。当 CIG 中所有的 CIS(跨越所有 Acceptor) 都已被禁用，CIG 应转换至 Inactive 状态。

#### 7.4.7 The Source ASE state machine

Source ASE 的行为略有不同，因为我们需要 Initiator 的确认以确保完全的转换。这将引入 Disabling 状态。它只用于 Source ASE，如图 7.10 所示的 Source ASE 状态机。

对于 Source ASE，Acceptor 需要来自于 Initiator 的确认，确认 Initiator 已准备好停止接收音频数据。Acceptor 可以自动停止流传输，但 Receiver Stop Ready 命令会导致流传输完全终止。如果存在重用 ASE 的潜在需求，

这一点很重要。在通知了处于 Disabling 状态的事实后，Acceptor 应等待 Initiator 发出的命令，告知其停止流传输并转到 QoS configured 状态。在那里，Source ASE 可以转移到 Releasing 状态，如果这样做了，则只有在 ASE 再次通过状态机的情况下(整个状态机重新转换一遍)，才能重新使用 CIS。

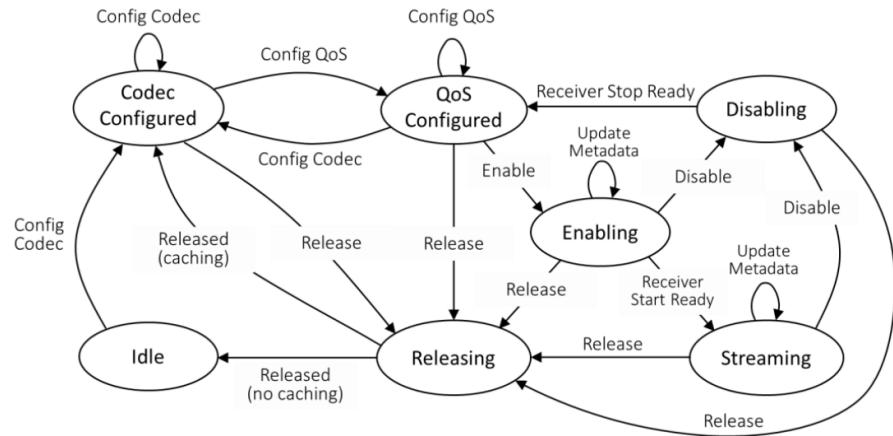


图 7.10 Source ASE 状态机

一旦处于 Releasing 状态，无论是通过 Sink ASE Streaming 状态直接转换、Source ASE Disabling 状态直接转换、或者从任意一方的 QoS configured 状态释放，Initiator 和 Acceptor 会拆除它们的数据路径并且终止 CIS。Acceptor 可以选择转换至 Idle 或 Codec configured 状态，两种操作都是自动执行的。如果 CIS 是双向的，则在 Sink 和 Source ASE 都处于 Releasing 状态之后，才能终止 CIS。当最后 CIS 终止时，CIG 从 Inactive CIG 状态转换到 No CIG 状态。

#### 7.4.8 Autonomous Operations on an ASE

我们已经看到，Acceptor 可以自主将 Sink ASE 从 Enabling 状态转换到 Streaming 状态、从 Releasing 状态转换到 Idle 状态或 Codec configured 状态。这些并不是 Acceptor 能够自主行动的唯一场合。除了表 7.21 所示的转换之外，所有的状态转换都可以由 Acceptor 或 Initiator 执行。然而，在大多数情况下，都是执行上述的状态机之间转换。

ASE Type	Current State	Next State	Initiating Device
Sink and Source	Codec Configured	QoS Configured	Initiator
Sink and Source	QoS Configured	QoS Configured	Initiator
Sink and Source	QoS Configured	Enabling	Initiator
Source	Disabling	QoS Configured	Initiator
Sink and Source	Releasing	Codec Configured	Acceptor
Sink and Source	Releasing	Idle	Acceptor

表 7.21 仅限于 Initiator 或 Acceptor 执行的 ASE 转换

---

#### 7.4.9 ACL link loss

如果 Acceptor 和 Initiator 之间的 ACL 链路丢失，则该 Acceptor 的所有 CIS 都将断开。如果 CIG 中涉及其他 Acceptor，则 Acceptor 应将与丢失的 ACL 链路相关联的所有 ASE 转换到 QoS configured 状态，以便链路恢复时重新启用这些 CIS。Acceptor 会通知状态的更改，尽管 Initiator 是否会收到通知存疑。

由于 Acceptor 不一定知道任何其他 Acceptor 的存在，因此它们可能不知道 CIG 是否处于 active 状态或流传输到另外的 Acceptor。如果它们在链路丢失之后没有收到重连请求，它们可以使用特定于实现的超时来将 ASE 转换到 Idle 状态。在稍后的重连 ACL 链路时，Initiator 应读取该 Acceptor 的 ASE 特征以检查其状态。如果由于超时而从 QoS configured 状态释放，Initiator 通常需要拆除并重新建立整个 CIG。请记住，Initiator 和 Acceptor 之间存在不对称——ASE 状态机位于 Acceptor，而 CIG 状态机位于 Initiator。Acceptor 从不知道 CIG 的状态，它是具有 ACL 连接状态的全局视野的 Initiator 用来驱动 ASE 状态的工具。

### 7.5 Handling missing Acceptors

在开始配置 ASE 之前，CAP 要求 Initiator 连接到目标 Coordinated Set 中的所有 Acceptor。现实世界中，有时不是所有的 Acceptor 都在，因为要么因为其中一个是关闭状态、丢失状态，要么在范围之外。这种情况下，Initiator 应继续设置它可以找到的 Coordinated Set 成员。在此种情况发生之前等待多长时间、是否涉及用户反馈、以及选择发送给可用 Acceptor 的 Audio Channel，都是特定于实现的。Initiator 应继续搜索丢失的 Acceptor，并在发现它们时将它们添加进来。

CIG 的本质是，一旦它处于 Active 状态，就不能配置其他 CIS。因此如果 Initiator 仅为能够找到的 Acceptor 配置 CIS，那么，如果丢失的 Acceptor 出现了，并且没有为其规划 CIS，则必需拆除 CIG，重新配置并建立 CIS，这将中断 Audio Stream。

为了防止音频中断，Initiator 可以针对任何丢失的 ASE，使用缓存值来规划 CIG。如果稍后检测到丢失的 Acceptor，则可以配置它们的 ASE，然后在该 Active CIG 中启用相关的 CIS(因为它们已经被规划过)，并且开始传输音频数据。CAP 没有描述该过程，它是使用 BAP 过程，加之 CAP 过程的一个功能扩展，但是可以带来更好的用户体验。

---

## 7.6 Preconfiguring CISes

存在类似的情况，即 Initiator 知道 Acceptor 可能参与具有不同 ASE 配置的多个不同的用例。这方面的一个常见示例是流式音乐(A2DP 的模拟)，它只需要 Sink ASE 即可使 CIS 将音频从 Initiator 传递到 Acceptor，但可能被呼入的电话中断，需要麦克风返回音频流。为了使这种转换更快速，Initiator 可以配置一组满足两种用例的 ASE，即两个 Sink ASE 和两个 Source ASE，这样当它想要在两个应用程序之间转换时，所需做的就是启用或禁用宏 Source ASE，而不是拆掉所有东西并重新启动。

这样做的缺点是，返回 Isochronous Stream 的 airtime 总是分配好了的，尽管它可能永远不会被使用。用于高可靠性的立体声 48\_2\_2 流占用大约 63% 的 airtime。如果计划包含 32\_2\_2 返回流，则增加到 90%。相比之下，双向 32\_2\_2 stream 只占 airtime 的 41%。这两个应用程序的延迟要求也存在差异。对于使用 High Reliability QoS 设置的音乐流，48\_2\_2 流和 32\_2\_2 返回的总延迟就超过了 140ms。这比双向 Low Latency 32\_2\_1 流的 56ms 长得多。表 7.22 展示了不同 QoS 配置对双向流的 airtime 和 latency 的影响。

QoS Configuration		Airtime (Stereo)	Latency PD = 40ms
Outbound	In-band		
48 2 1	32 2 1	90%	141ms
32 2 1	32 2 1	41%	57ms
24 2 1	16 2 1	31%	56ms

表 7.22 各种双向 QoS 配置的 Airtime 和 Latency

这意味着，在 airtime，QoS 以及 call setup time 之间，存在权衡，设计者需要意识到这一点。做出这一决定通常取决于 Initiator 的其他 airtime 资源的需求。它说明了 Bluetooth LE Audio 的灵活性，但强调一个事实，即与经典 Bluetooth Audio Profile 相比，实现者需要更多了解整个系统。

## 7.7 Who's in charge?

设计者面临的一个问题是决定“谁负责？”。Bluetooth LE Audio 的发展已经有了一些强烈的见，来自手机制造商，它们认为手机对一切负责，但是也有来自扬声器和可听设备制造商，它们觉得自己的产品需要更多的自主性，从而产生了“Sink led journey”的概念。此规范为两者都提供了空间，但它们需要开发人员意识到其中的一些后果。

看一个简单例子很有用，一对耳塞，每个耳塞都有麦克风。手机制造商可能希望同时使用两者，因为这将允许它们处理两种音频，从而产生更好的语音信号。另一方面，耳塞制造商可能更倾向于只使用一个，从而节省另一

---

个耳塞的电池寿命。如果他们看到麦克风上的电池开始下降，他们甚至希望能够在通话过程中更换正在使用的电池。

问题是，如何启用这些选项？假设耳塞之间有通信，其中一个可以考虑不暴露其 Source ASE，尽管这是一种暴力的手段，特别是当手机可以通过读取 PAC 记录来推断其存在时。正如我们上面看到的，Acceptor 应该为它能够支持的每个 Audio Stream 公布 ASE，即使它可能无法在所有时间点支持它。更好的方法是，将 Available\_Audio\_Contexts 设置为 0(这是允许将所有 Context Type bit 设置为 0 的少数情况之一)，使用 PACS 中的 Available\_Audio\_Contexts 特征来指示 server 不能传输音频。

如果 Initiator(电话)可以看见每个耳塞都有一个可用的 Source ASE，那么它可以有效地负责。它可以决定只选择一个(尤其是处理两个时间未对齐的输入 stream 并不简单，而且会增加功耗)。它可以使用其他信息，比如检查每个耳塞的电池状态，当手机只想要使用一个耳塞时，以此指导麦克风的选择。

如果它更智能，它可以查看通话中是否有规律地重复短语，推断这意味着信噪比低，并添加第二个麦克风，试图通过获得几 dBs 的信号来提高质量。同样，手机应用程序可以记录与特定人的通话持续时间，并从该历史中推断出通话的可能长度、耳塞电池寿命，并使用该数据来通知其建立多少音频流以及 QoS 设置的决定，这些设置将使电池在预期通话持续时间内继续使用。如此多的差异化机会！我怀疑这两种方法在短期内都不太可能，因为手机将继续将耳塞视为一种可以随心所欲使用的资源。

另一方面，Sink led journey 希望耳塞具有更多的决策能力。然而，这意味着它们能够相互通信，这目前超出了范围(HAPS 中的 Preset local synchronization 除外)。如果这副耳塞决定只想在一个耳塞上使用麦克风，它们可以在彼此之间决定哪一个将承受电池消耗，另一个可以将其 Available\_Source\_Contexts 设置为 0x0000 (他们可以在电池盒里的时候这样做，就不需要使用单独的 Sub-GHz 无线电链路)。

这种方法的优点是，手机知道这两个耳塞都有 Source ASE，因此可以在 CIS 中配置流。它无法启用该 stream，因为耳塞将 Available\_Source\_Contexts 显示为 0x0000，耳塞会在 config codec 阶段拒绝手机。但是 Initiator 可以对其进行调度，因为它可以在 HCI LE Set CIG Parameters 命令中放入任何想要的内容。虽然我们已经看到了，此过程通常使用 Initiator 从 Acceptor 收到的信息，但它针对丢失的 Acceptor 或 ASE 可

---

使用缓存的或假设的值。意味着在某时刻需要交换麦克风，这是可能发生的。否则，Initiator 需要拆除 CIG 并重建它，从而导致呼叫中断。(这不应该终止呼叫，因为 stream 的丢失不会导致 TBS 状态机发生任何变化，但这不是一个好的用户体验)。

这里的重点是 PACS 和 ASCS 特征在 Audio Stream 如何建立方面允许惊人的权力。Initiator 可以单方面采取行动，但可能会影响设备的工作状态。为了获取最佳性能和用户体验，设计者需要了解提供的选项和如何使用它们。

--oOo--

以上就是设置 unicast Audio Streams 的方法，因此我们现在可以将注意力转向广播了。

---

## 第8章 Setting up and using Broadcast Audio Streams

在本章中，我们将了解到如何配置 broadcast Audio Streams。广播是 Bluetooth LE Audio 主要的新功能，并有可能改变每个人使用音频的方式。他引入的令人兴奋的用例包括共享音频的功能，以及建立 ad-hoc 私有连接。后者由 BASS 中定义的 Broadcast Assistant 功能使能，该功能带来全新的控制和用户体验。

重复一遍，主要规范是 BAP——Basic Audio Profile，但我们现在需要添加另一个 GAF 规范，叫做 BASS——Broadcast Audio Scan Service。BASS 定义了如何使用另外的设备来帮助查找广播音频流，并指示一个或多个 Acceptor 与它们同步。

CAP 再一次发挥作用。除了提供建立和接收广播流的过程之外，它还定义了 Commander 角色。Commander 可以是物理设备，也可以是电话或电视上的应用程序。我们会在完成发送和接收广播音频流的基础知识时，更详细地了解 Commander 角色。CAP 的主要任务是指定有关关联 Context Types 和 Content Control IDs 的规则，并重复完成事情的顺序。

我们将从建立和接收广播音频流的基础知识开始，然后去了解 Commander 角色带来了什么。在第 12 章中，我们将进一步探讨广播音频中的可能性，检查它可以实现的一些新用例。

尽管广播拓扑起源于提高感应式助听器的音频质量，但 Bluetooth LE Audio 提供了比感应线圈式助听器更多的通用性。许多情况下，除了 Broadcast Source 和 Commander 之间的连接之外，Broadcast Source 和 Broadcast Receiver 之间还将存在 ACL 连接。设备甚至可以同时支持广播和单播，充当广播和单播连接之间的中继。但是，在我们讨论这些复杂问题之前，我们将中广播本身的基础知识开始。

广播规范分为三个独立的功能：

- **传输广播音频流**。最简单的说法是，Broadcaster(Broadcast Source 更容易记的名字)独立运行——它通常不知道是否有任何接收者存在，并且在收听它。
- **找到广播音频流**——通常会使用 Broadcast Assistant，通常也叫

---

Commander, (尽管技术上这只是一个角色, Broadcast Assistant 是其中一个 sub-role)。Broadcast Assistant 可以为 Broadcast Receiver 找到广播音频流。Broadcast Assistants 将广播从一个简单的感应线圈提升为一个非常强大的新拓扑, 它允许加密 stream 用于个人和基础设施级别的私密音频。它还使得在多个广播音频流中进行选择更加容易。Broadcast Assistants 可以设计成一个独立的设备, 或可以位于任何 Broadcast Source 中。

- 接收广播音频。broadcast receiver, (本质上同 Broadcast Sink 一样), 可以扫描存在的广播音频流, 并且与之同步。在基础级别上, 它也可以独立运行。Broadcast Receiver 可以同步到加密或未加密的广播音频流, 但是需要获取 Broadcast\_Code 来解密加密过的音频流。可以通过 out-of-band 方式获取, 或通过 Broadcast Assistant 的协助完成。

## 8.1 Setting up a Broadcast Source

在第 4 章中, 我们介绍了 Broadcast Isochronous Streams (BIS) 和 Broadcast Isochronous Groups (BIG) 的基础知识。与单播流不同, Broadcast Source 和 Broadcast Sink 是独立运行的。这使得 Broadcast Source 与其他 Bluetooth Central 设备非常不同, 因为它是单方面运行的。与我们在前面章节探究的单播用例不一样, 两个设备之间没有命令、请求或通知。相反, Broadcast Source 完全由其特定的应用程序驱动。

这样做的结果之一就是 Broadcast Source 有一个非常简单状态机, 如图 8.1 所示。由于没有与任何 Acceptor 进行交互, 因此过程非常简单, 包括 Broadcast Source 中的 Host 到 Controller 的命令。

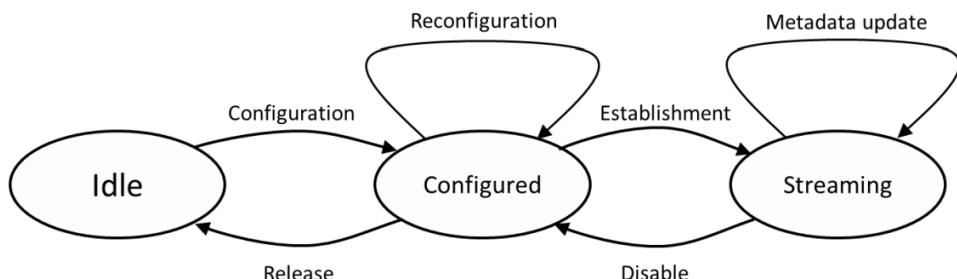


图 8.1 Broadcast Source 状态机和配置过程

为了配置 Broadcast Source, Host 需要向 Controller 提供 BIG 配置的细节。Controller 用它来规划 BIS。还需要提供信息来填充 BASE, 其描述了每

---

个 BIS 的配置和其内容。配置数据由运行在 Broadcast Source 上的应用程序提供。在一些应用中，用于包含在 BASE 中的 metadata 可能是应用的一部分，在其它的应用中，它可能由外部提供，或从控制输入、或从输入音频流中提取，例如 TV 的程序引导数据。

BAP 为 Broadcast stream 的转换和配置更新定义了 6 个过程。

- The Broadcast Audio Stream 配置过程
- The Broadcast Audio Stream 建立过程
- The Broadcast Audio Stream 禁用过程
- The Broadcast Audio Stream Metadata 更新过程
- The Broadcast Audio Stream 示范过程，以及
- The Broadcast Audio Stream 重新配置过程

由于设备之间没有连接，这些过程仅限于 HCI 命令，在 Initiator 准备好开始广播时发送。CAP 将它们组合成 5 个过程，将配置和建立结合到 Broadcast Audio Start 过程：

- Broadcast Audio Start 过程
- Broadcast Audio Update 过程
- Broadcast Audio Stop 过程
- Broadcast Audio Reception Start 过程
- Broadcast Audio Reception Stop 过程

## 8.2 Starting a broadcast Audio Stream

由于 Broadcast Source 不知道谁会接收它的广播音频流，因此与 Coordinated Sets 相关的 CAP 前导步骤都不想关。CAP 所要求的只是 metadata 中包含正确的 Context Types。仅当存在携带来自 Broadcast Source 的 content control 的附带 ACL 连接时，才需要 Content Control IDs。这在个人电视等应用中是需要的，个人电视使用广播来运行多个家庭成员收听，但是它们的耳塞可以用来暂停或改变频道。

BAP 定义了我们需要的所有其他内容，其中程序指导 Controller 建立 Extended Advertising，并提供数据来填充 Extended Advertisements 和 Periodic Advertising 序列中公开的参数。

图 8.2 采用了第 4 章中的 Extended Advertising 图，并强调了建立 Broadcast Source 所需的特定数据元素。建立广播音频流的第一步是组装所有需要的数据，Controller 用来填充 Broadcast Audio Announcement

Service、BIGInfo 和 BASE，在 BAP Audio Stream Establishment procedure [BAP 6.3] 中由描述。

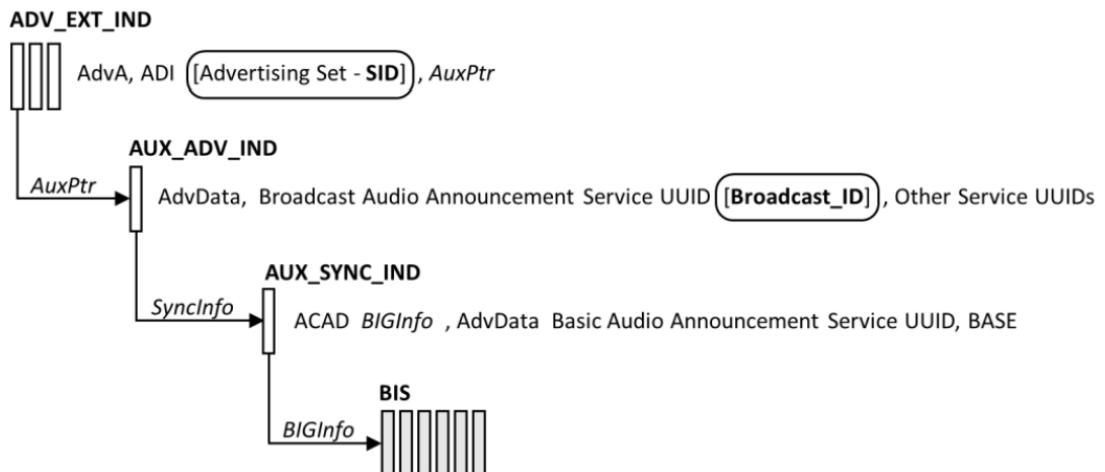


图 8.2 建立 Broadcast Source 所需的数据

### 8.2.1 Configuring the BASE

应用程序会决定多少 BIS 用于传输，以及它们相应的 codec 和 QoS 配置。BAP 建议使用表 3.2 中的 16\_2 或 24\_2 的设置(即 16kHz, 10ms SDU, 或 24kHz, 10ms SDU)对至少一个广播音频流进行编码，以确保每个 Acceptor 都可以对其进行解码。任何其他的 codec 配置将由应用程序或更高级别的 profile 决定。Host 还应获得填充 BASE 所需的任何 metadata 数据内容。当前 metadata 需求如表 8.1 所示。

Profile	Required LTV metadata structures	Comments
BAP	None	
CAP	Streaming Audio Contexts	
CAP	CCID List	Only if content control exists for the Audio Stream
HAP	None	Inherited from PBP
TMAP	None	
PBP	None	ProgramInfo is recommended to describe the Audio Stream

表 8.1 来自不同 Profile 的 BASE 所需 Metadata LTV

配置 stream 所需要的最后一条信息是 Presentation Delay 的值，通常由 OoS 设置来确定。

Controller 现在可以将其 BASE 结构组合在一起，该结构准确描述了它将要传输的 streams 以及它们的配置。

我们在第 4 章中对 BASE 进行了概述，查看了它包含的内容，如图 8.3 所示。

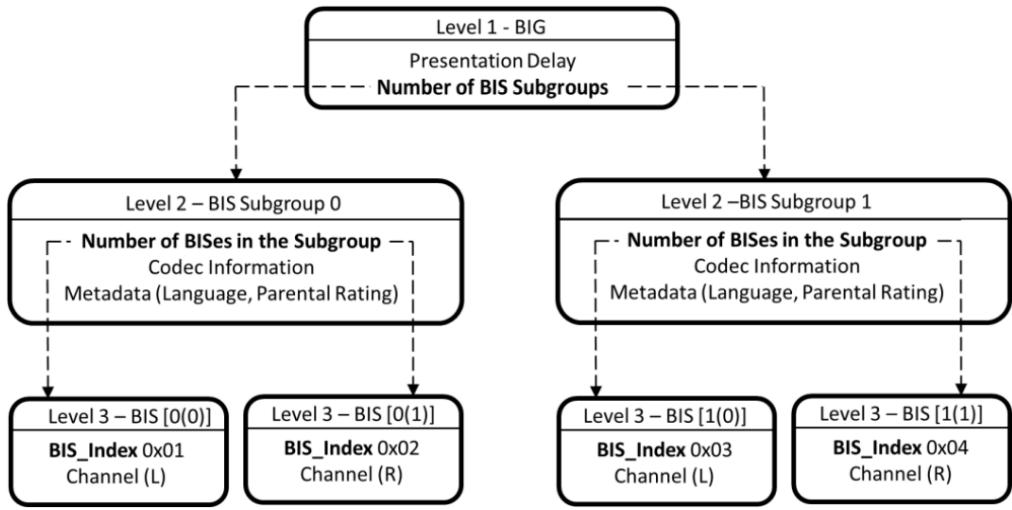


图 8.3 简化的 BASE 结构

整个 BASE 是一个 LTV 结构，以 AD Type 表示，参数如表格 8.2 所示。

Parameter	Size (Octets)	Description
Length	1	LTV 结构的整体长度
Type	1	Service-Data UUID
Level 1——BIG 参数(对所有 BIS 通用)		
Basic Audio Announcement Service UUID	2	0x1852 (来自 Bluetooth Assigned Numbers)
Presentation Delay	3	范围从 0 到 16.7s (us) (0x0000 到 0xFFFF)
Num_Subgroups	1	此 BIG 中，用于对具有共同特征的 BIS 进行分组的 subgroup 数量
Level 2——BIS Subgroup 参数(BIS subgroups 通用参数)		
Num_BIS[i]	1	Subgroup 中的 BIS 数量
Codec_ID[i]	5	Subgroup 的 codec 信息，通常是 LC3 (0x06)
Codec_Specific_Configuration_Length[i]	1	Subgroup 中 Codec_ID 的 codec 配置数据长度
Codec_Specific_Configuration[i]	Varies	Subgroup 中 Codec_ID 的 codec 配置数据
Metadata_Length[i]	1	Subgroup 的 metadata 长度
Metadata[i]	Varies	Subgroup 的 metadata。 CAP 需要每个 subgroup 的 Streaming_Audio_Contexts LTV metadata，如果应用了 content control，则还需要 CCID_List LTV 结构。
Level 3——特定的 BIS 参数(如果需要，针对单个 BIS)		
BIS_Index[i[k]] 需要注意的一点可能是 BIS_index 从 1 开始，而 CIS_ID 从 0 开始。	1	BIG 中每个 BIS 的唯一索引
Codec_Specific_Configuration_Length[i[k]]	1	Subgroup 中，特定 BIS[i[k]] 的 codec 配置数据长度
Codec_Specific_Configuration[i[k]]	Varies	Subgroup 中，特定 BIS[i[k]] 的 codec 配置数据

表 8.2 BASE 参数

BASE 允许很大的灵活性，大多数日常应用程序都不需要其中的大部分。Level 1 参数必需存在并于适用于每个 subgroup。大多数 BIS 可能只有一个 subgroup，该 subgroup 通常携带两个或三个 BIS——一个用于左边，一个用于右边，外加一个可选的单声道或 joint stereo stream。定义多个 subgroup 的原因是某些 BIS 具有不同的 metadata，例如不同的语言。类似这样的差异只能在 Level 2 参数中表达。在 Level 3 中，你可以为 subgroup 内的 BIS 指定不同的 codec 配置，比如具有不同的质量等级，例如一个 stream 的采样率

---

为 48kHz，一个为 24kHz。但这是在 Level 3 中唯一的变化。不同的语言、不同的 parental ratings 等等，都需要在 Level 2 中具有它们自己的 subgroup 定义。

如果由不止一个 BIS，则应该按照 subgroup 和 BIS\_Index 的顺序重复 Level 2 和 Level 3 参数部分，如表格 8.3 所示。

Level	Parameter	Value
2	Num_BIS [0] Codec_ID [0] Codec_Specific_Configuration LV [0] Metadata LV [0]	Num_BIS = 2
3	BIS_index [0[0]] Codec_Specific_Configuration LV [0[0]]	0x01
2	Num_BIS [0] Codec_ID [0] Codec_Specific_Configuration LV [0] Metadata LV [0]	Num_BIS = 2
3	BIS_index [0[1]] Codec_Specific_Configuration LV [0[1]]	0x02
2	Num_BIS [1] Codec_ID [1] Codec_Specific_Configuration LV [1] Metadata LV [1]	Num_BIS = 2
3	BIS_index [1[0]] Codec_Specific_Configuration LV [1[0]]	0x03
2	Num_BIS [1] Codec_ID [1] Codec_Specific_Configuration LV [1] Metadata LV [1]	Num_BIS = 2
3	BIS_index [1[1]] Codec_Specific_Configuration LV [1[1]]	0x04

表 8.3 多个 subgroup 和 BIS 条目顺序

### 8.2.2 Creating a BIG

正如我们在单播中看到的，BIG 是使用 HCI 命令创建——本例中，如表 8.4 中展示的 LE Create BIG 参数。

Parameter	Description
BIG_Handle	Host 设置的 BIG 标识符
Advertising_Handle	periodic advertising train 相关的 handle
Num_BIS	BIG 中的 BIS 数量
SDU_Interval	包含编码音频数据的周期性 SDU 之间的间隔(以 us 为单位)
Max_SDU	每个 SDU 的最大大小，以字节为单位
Max_Transport_Latency	每个 BIS 的最大传输延迟，包含 pre-transmissions
RTN	请求重传次数(建议值)
PHY	可接受 PHY 值的位域。Controller 会从受支持的选项中进行选择。高阶 profile 可能只要求一个值——通常为 2Mbps。 0=1Mbps, 1=2Mbps, 2=LE coded。
Packing	安排 BIS Subevent 的首选方法。Controller 仅将此作为建议。 0=Sequential, 1=Interleaved
Framing	如果设置为 1，BIS 数据 PDU 将 framed。如果设置成 0，Controller 可以决定是使用 framed 或 unframed。
Encryption	如果音频 stream 需要加密，则设置成 1，如果不需要，则设置成 0。
Broadcast_Code	为 BIG 中所有 BIS 生成加密密钥的代码。对于未加密的 stream，此值应该设置成 0。

表 8.4 HCI LE Create BIG 命令参数

LE Create BIG Parameters 命令和等效的 LE Set CIG Parameters 命令之间有一个重要的区别，BIG 中的每个 BIS 都使用相同的参数。这个限制来源于这样一个事实，即 Isochronous Channel 信息包含在 BIGInfo 包中，

---

BIGInfo 包的大小不足以容纳多个不同的 BIS 的细节。结果是，每个 BIS 都是相同大小，必需符合最大的 SDU。BIG 内的不同 BIS 可以使用不同的 codec 配置，甚至是不同的 codec，但是 BIS 结构必需被配置为适合这些不同配置中最大可能数据包。如果 BIG 中不同的 BIS 使用不同的 QoS，则意味着存在冗余空间。

创建了 BASE 结构之后，Host 可以要求 Controller 使用 LE\_Create\_BIG 命令规划 BIG 和 BIS，(定义见 Core Vol 4, Part E, Section 7.8.103)，参数如表 8.4 所示。

一旦命令发出了，Controller 完成了规划，机会向 Host 响应 LE\_Create\_BIG\_Complete 事件 [Core Vol 4, Part E, Section 7.7.65.27]，其中包含其选择使用的实际参数，如表 8.5 所示。其中一些由 Host 应用程序使用，其他应用程序将反映 BIGInfo 中的内容——该数据结构用于通知扫描设备 broadcast Isochronous Streams 的配置。

Parameter	Description	Used in
Subevent Code	0x1B——LE_Create_BIG Complete 事件的 Subevent code	Host
Status	BIG 成功被规划了，设置为 0，否则，当 Error Code 起作用时，设置为 1	Host
BIG_Handle	Host 在其 LE_Create_BIG 命令中发送的相同 BIG handle	Host
BIG_Sync_Delay	单个 BIG 事件中所有 BIS 的所有 PDU 的最大传输时间(以 us 为单位)，使用以下包含的实际参数	Host
Transport_Latency_BIG	BIS 的实际传输延迟(以 us 为单位)。(这包括用于 SDU 的所有 BIG 事件)	Host
PHY	将用于传输的 PHY	BIGInfo
NSE	每个 BIS 的 subevent	BIGInfo
BN	Burst Number。(每个 BIS 事件中的新的 payload)	BIGInfo
PTO	用于 pre-transmissions 的偏移	BIGInfo
IRC	Immediate Repetition Count。(payload 在每个 BIS 事件中传输的次数)	BIGInfo
Max_PDU	Payload 最大字节数	BIGInfo
ISO_Interval	Isochronous Interval 的值。(连续 BIG Anchor Points 之间的时间)。 表示为 1.25ms 的倍数	BIGInfo
Num_BIS	BIG 中的 BIS 总个数	BIGInfo
Connection Handle [i]	BIG 中所有 BIS 的 Connection handle 列表	Host

表 8.5 LE\_Create\_BIG\_Complete HCI 事件返回的参数

此时，Host 已经具备了开启流程所需的一切，流程的第一阶段是建立 Extended Advertising 和 Periodic Advertising 序列。需要将 Broadcast Audio Announcements [BAP 3.7.2.1] 和 Broadcast\_ID [BAP 3.7.2.1.1] 包含在 AUX\_ADV\_IND Extended Announcements 中，以及包含 BASE [BAP 3.7.2.2] 和 BIGInfo 的 Basic Audio Announcement。

Host 使用 HCI\_LE\_Set\_Extended\_Advertising\_Data [Core Vol 4, Part E, Sections 7.8.54]，并进入到 Broadcast 模式[Core Vol 4, Part C, Section 9.1.1]，以启动 Extended Advertising，然后在进入 Periodic Advertising 模式 [Core Vol 4, Part C, Section 9.5.2]之前，使用 HCI\_LE\_Set\_Periodic\_Advertising\_Data 命令[Core Vol 4, Part E, Section

---

7.8.62]来填充 BASE。

一旦建立了 Extended Advertisements 和 Periodic Advertising 序列， broadcast Audio Source 就进入了 Configured State。在此阶段，它还没有传输任何音频数据。

### 8.2.3 Updating a broadcast Audio Stream

在任何时刻，Broadcast Source 都可以更新包含 metadata 的 BASE 中的 LTV 结构。这通常反映了 Audio Channel content 的变化，比如新的 ProgramInfo 或 Context Types。这在 BAP Modifying Broadcast Sources procedure [BAP 6.5.5]中定义。Broadcaster 不需要停止音频流，但可以在 Configured 或 Streaming 状态下动态更改。

不能保证正在接收音频流的 Acceptor 会发现此类更改。一旦 Acceptor 已经使用 BIGInfo 中的信息同步到了 stream，它就不再需要再跟踪 Periodic Advertisements 或读取 BASE，除非其他 profile 需要它这样做。

### 8.2.4 Establishing a broadcast Audio Stream

一旦 Extended 和 Periodic Advertising 开始运行，Broadcast Source 就需要建立它的音频 stream，并开始传输了。它分三个步骤完成，由 BAP Broadcast Audio Stream Establishment procedure [BAP 6.3.2]定义。(不要与 BASE 混淆，虽然有相同的首字母)。

首先，它进入 Broadcast Isochronous Broadcasting Mode[Core Vol 3, Part C, Section 9.6.2]，准备在它的 BIG 的 BIS 中发送 PDU。然后，开始 Broadcast Isochronous Synchronizability Mode [Core Vol 3, Part C, Section 9.6.3]，开始在 Periodic Advertisement 的 ACAD 域中发送 BIGInfo，提醒正在扫描的广播音频流的任何设备其存在。

最后，Broadcast Source 需要按照单播相同的方式，使用 LE Setup ISO Data Path HCI 命令[Core Vol 4, Part E, Section 7.8.109]，建立音频数据路径。

此时，Broadcast Source 完全运行、传输它的 BIG 和连续的 BIS。如果没有连接，它永远不会知道是否有任何设备侦测或同步到这些广播。

### 8.2.5 Stopping a broadcast Audio Stream

为了停止广播，并且回到 Configured 状态，Initiator 需要使用 Broadcast Isochronous Terminate procedure [Core Vol 3, Part C, Sections 9.6.5]。它会

---

停止 BIS PDU 和 BIGInfo 的传输。已经同步过的 Acceptor 会收到 BIG\_TERMINATE\_IND PDU 以提醒它们传输的终止。如果它们没有收到此信息，除了 BIS 和 BIGInfo 消失之外的事实之外，它们不会得到更多其他信息。

Broadcast Isochronous Terminate 过程的最后，Broadcast Source 会返回 Configured 状态，它可以被释放，或重新使能。

### 8.2.6 Releasing a broadcast Audio Stream

为了将 Broadcast Source 返回到它的 Idle 状态，需要推出 Periodic Advertising 模式，这就会将其返回至 Idle 状态。

## 8.3 Receiving broadcast Audio Streams

如果 Broadcast Sink 想要接收广播音频流，它需要通过扫描来找到。不论是 Broadcast Sink 本身还是 Broadcast Assistant，都是用相同的过程去寻找。我们将在本节中了解到 Broadcast Sink 怎样执行此操作，然后看一下当 Broadcast Assistant 可用时，此任务怎样被委派的。

因为 Acceptors 作为独立的实体，CAP 并不能真正用于广播音频流的接收。CAP 定义了 Broadcast Audio Reception Start procedure [CAP 7.3.1.8]，但是，除非 Acceptors 之间有相互之间通信的方法，否则它们不知道彼此的存在。它们知道自己是 Coordinated Set 的成员，但不知道其他成员是不是在周围。Commander 可以完成协调任务，我们稍后会讨论，也可以使用 out-of-band 机制，例如 sub-GHz 无线电，其允许 Acceptor 相互之间进行交流。但 Acceptor 本身的扫描本质上是在 BAP 层级上运行的。

### 8.3.1 Audio Announcement discovery

接收广播音频流的第一步是发现它，通过使用来自 Core [Core Vol3, Part C, Section 9.1.2] 的 Observation Procedure 来执行。这是一个查找广播的标准扫描。这里，感兴趣的类型是包含带有 Broadcast\_ID 的 Broadcast Audio Announcement Service UUID 的 Extended Advertisement。Extended Advertisements 还可以包含其他 Service UUID，比如在 Public Broadcast Profile 中定义的那些，scanner 可以使用这些 UUID 来过滤它想要侦查的 stream。

找到适当的 Extended Advertisement 之后，scanner 将查找 SyncInfo，并且用它来同步到与 Broadcast\_ID 相关联的 Periodic Advertisements。一旦同

---

步了，scanner 就可以找到并解析 BASE 中的数据，该数据提供了 BIS 集合的信息。

此时，Acceptor 的行为与以前的 Bluetooth Peripherals 截然不同。没让 Central 设备告诉自己该做什么，相反，Acceptor 自己去收集范围内的、具有其感兴趣的音频流的、可用的 Broadcast Sources，并且自行决定接收哪一个。这完全是实现驱动的。它可能会查找以前连接过的已知 Broadcast\_ID 或 BASE metadata 中的信息。。它可以连接到它找到的第一个 stream，然后逐步通过任何其他可用的 stream，或者它可以从范围内的每个 Broadcast Source 搜集数据，并且以某种方式呈现给用户，允许他们手动选择。所有这些都是特定于实现的

Acceptor 可用使用附加的信息来通告其选择。如果 stream 的 Context Type 与 Acceptor 的任何可用的 Context Type 都不匹配，通常不会影响到 stream。它也不想接收它不支持的 Audio Location 的 stream。但是决定权在 Acceptor。没有 Commander，或两个 Acceptor 之间专有链路，用户可能必需在左右耳塞上手动选择 stream。这种情况不太可能发生，因为这是一种糟糕的用户体验，制造商提供单独的 Commander (或 Commander 应用程序)，会在左右耳塞之间实现专有链路。然而，设计者应该意识到，设备之间没有 Bluetooth 连接时，允许设备对一起工作的这种需求。但这就是 remote control/Commander 的作用。

### 8.3.2 Synchronising to a broadcast Audio Stream

一旦确定了要接收的 Broadcast Source，Acceptor 就开始了 Broadcast Audio Stream synchronization procedure [BAP 6.6]。这里涉及从 AUX\_SYNC\_IND (以及任意补充的 AUX\_CHAIN\_IND) PDU 中读取 BASE 信息，以此来确定与该 Acceptor 的 Audio Location 相对应 BIS 的 codec 配置和索引号。每个 BIS 的 BIS\_Index 定义了 BIS 在 BIG 内的顺序。知道了这一点，Acceptor 可用确定它想要接收 BIG 中的哪些 BIS。

然后，Acceptor 检索 BIGInfo，其中包含 BIG 结构、hopping sequence、以及 BIS Anchor Point 所在位置的所有信息。一旦完成，它就可以从 Core 中调用 Broadcast Isochronous Synchronization Establishment procedure [Vol 3, Part C, Section 9.6.3]，以获取适当的 BIS。如果没有完成，就需要建立音频数据路径。接收到输入的音频包之后，它将使用 Presentation Delay 值来确定呈现的时间点。

---

### 8.3.3 Stopping synchronization to a broadcast Audio Stream

如果 Acceptor 确定停止接收广播音频流，它会自动断开音频数据路径并停止与 PA 以及 BIS 的同步。

## 8.4 The broadcast reception user experience

虽然 Acceptor 可以自己获取 Broadcast Stream，但这不是一个很好的用户体验。从最基本的角度来说，这是一种对磁感线圈的用户体验的精确模拟，即佩戴助听器的人按下按键以获取磁感线圈 stream。这种用户体验之所以有效，是因为 telecoils 是感应线圈，通常只在线圈范围内，意味着不存在是否连接到正确的线圈问题。如果范围内只有一个 Bluetooth LE Audio 发射器，那么体验将是相同的，但是一旦广播应用程序边的流行起来(这不久就会成真)，许多情况下，你可能都在许多发射器的范围内。这与我们使用 WiFi 的情况完全相同。如果你用手机或笔记本电脑扫描 WiFi 接入点，你经常会发现有十多个或更多。Bluetooth LE Audio 用户体验可能要苦难的多，因为在大多数情况下，你将佩戴一对耳塞或助听器，他们只有最小的用户接口。对于开始和停止 stream 的需求，以及将两个耳塞同时连接到相同的 Broadcast Source，这些也增加了其复杂性。

因此，尽管上面的解释是有效的，并指出了怎样接收广播流，但它们基本上是学术性的。为了在显示世界中实现这一点，我们需要另一种设备，它可以更好地完成找到广播流，并告知一个或多个 Acceptor 怎样处理这些广播流，这些艰难任务。我们还需要一种方法来分发解码加密音频流的密钥。这就引出了 Commander 和 Broadcast Assistants。

## 8.5 BASS – the Broadcast Audio Scan Service

首先，谈谈 BASS——广播音频扫描服务。BASS 是一种在 Acceptor 中实例化的服务，用于公布它知道的广播音频流、它连接到的音频流、以及它是否希望 Broadcast Assistant 协助它管理这些信息，并帮助它建立连接。BASS 与 Broadcast Assistants (在 BAP 中定义，稍后我们将介绍)协同工作，以管理它们选择和管理的 broadcast connections。它是扩展广播生态的关键部分，使用 ACL 连接来辅助广播信息的分发和控制。

BASS 的关键要素有两个特征，无论何时使用，两个特征都是强制的：

Characteristic	Properties	Quantity
Broadcast Audio Scan Control Point	Write, Write w/o response	只有一个
Broadcast Receive State	Read, Notify	一个或多个

表 8.5 BASS 特征

---

Broadcast Audio Scan Control Point 特征允许一个或多个 Broadcast Assistants 通知 Acceptor 它们是否正在代表其进行查找 Broadcast Sources 工作。它们可以告诉 Acceptor 如何找到 BIG、连接到 BIG、断开与 BIG 的连接，并提供 Broadcast\_Code 来解密加密过的音频流。正如我们看到的，Broadcast Audio Scan Control Point 特征中有一个非常重要的参数，即 BIS\_Sync。当 Broadcast Assistant 将其设置成 0b1 时，它会告诉 Acceptor 开始接收 stream。当它设置为 0b0 时，Acceptor 应停止接收 stream。

上一段第一行的“一个或多个 Broadcast Assistants”是一个重要的陈述。Acceptor 可以使用任意数量的 Broadcast Assistants。所需的只有一点——每个都有一个 ACL 连接到该 Acceptor。一旦连接并记录了它们正在帮助该 Acceptor 查找广播流的这个事实，Broadcast Assistants 将以先到先得的方式运行，如果它们在 Coordinated Set of Acceptors 上运行，仅受 CSIP 调用的任何 Lock 的限制。每个 Broadcast Assistants 都需要向 Broadcast Receive States 注册，用于通知，意味着每当 Broadcast State 发生变化时，所有通知都将被更新，无论是通过 Broadcast Assistant 向其写入，还是 Acceptor 的本地操作。

Broadcast Receive State 特征公开了 Acceptor 正在做的事情——它连接到哪一个 BIG、当前真正接收哪一个 BIG 内的 BIS，它是否可以解密这些 BIS，以及它为每个 BIS 所拥有的 metadata。对于其可以同时同步到的每个 BIG，Acceptor 必须具有至少一个 Broadcast Receive State 特征。许多情况下，只有一个。现在轮到 Commander 了。

## 8.6 Commanders

尽管大多数人将广播视为 Bluetooth LE Audio 的一大新功能，但最重要的创新可能是 Commander 的概念，它提供了 remote control 和广播音频流的管理。再多个广播流的世界里，Commander 为用户提供了简单的方法来选择他们想要收听的内容。尽管广播概念的灵感来自于助听器中感应线圈的用户体验，但 Bluetooth LE Audio 应用远远超出了感应线圈可以做到的。

回顾感应线圈范例，它很简单——如果你站在 loop 的边界之内，你就只会收到音频流。(如果你在其上方的空间，就可能会有问题，但是这是边界问题)。通过 Bluetooth 广播，它们可以重叠并穿过墙壁，因此，用户需要一种简单的方式来选择它们想要收听的广播。我们已经看到了 BASE 中的 metadata 信息以及来自于诸如 Public Broadcast Profile (PBP) 的 profile 的服

---

务信息怎样帮助形成选项。然而，这只是公共广播，任何人都可以听到。对于个人广播，广播音频流是加密过的，需要获取加密密钥的方法。再次，这些都是 Commander 来使能的。

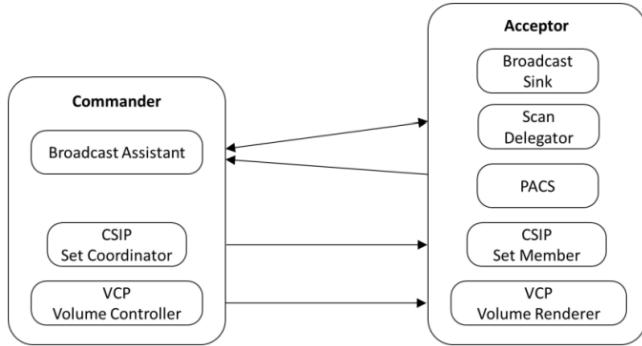


图 8.4 Commander 和广播 Acceptor 的主要元素

图 8.4 展示了 Commander 主要组成部分以及 Acceptor 主要与广播相关的一部分。两者都定义在 CAP 中，但是，我在本书中更广泛的使用它们。将 Commander 视为一个独立的设备是很有用的，比如电视的遥控器，因为对于用户来说，它的功能非常相似。Commander 通常包含四个要素：

- **CSIP Set Coordinator**，确保将任何命令发送到 Coordinated Set 中的所有成员，不论是关于 Broadcast Source 的信息还是音量控制。
- **Broadcast Assistant**，用于扫描 Broadcast Sources，并且让用户选择它们想要收听的那一个。
- **VCP volume controller** (我们会在第 10 章说明)，用来控制 Coordinated Set 中每个成员的音量，以及
- 获得和分发 Broadcast\_Code 以解密广播音频的能力。

如果没有 Commander 所提供的灵活性及其提供的功能，广播加密就会变得麻烦，使得围绕私有广播流的用例显示非常困难。私有广播为 Bluetooth LE Audio 带来了全新的维度。无论是个人电视和音乐、酒店电视接入、会议室共享音乐，还是系的呢 Bluetooth Audio 共享体验，它们都依赖于用户连接到正确的广播源并获取 Broadcast\_Code 以解密音频的简单方式。所有这些都由 BAP 中定义的两个新角色管理。它们是 Scan Delegator 和 Broadcast Assistant。它们与 Broadcast Audio Scanning Service (BASS) 协同工作，分配信息收集以及控制，为广播应用程序提供了丰富的内容。

---

### 8.6.1 Broadcast Assistants

Broadcast Assistants 允许其他设备执行我们在 8.3.1 节中描述的扫描工作。Broadcast Assistant Role 通常是单独 commander 的主要角色，代表 Broadcast Sink 进行扫描。卸下此任务有两个重要原因。**第一个**是扫描相当的耗电的操作，这是你不希望助听器或耳塞做的事情，或者至少不经常做，特别不清楚它在搜寻什么的情况下。这是一项在手机或专用 remote control 上执行的更好的工作，**第二个**原因是用户体验。广播应始终在其 BASE 结构中包含可读取的 metadata，允许带有显示器的设备显示每个信息。显示这些信息在电话或 remote control 上很容易实现，但是在耳塞上基本可能实现。因此，将**扫描和选择**任务移动到其他地方不仅可以节省电池寿命，还可以极大改善用户体验。

Broadcast Assistants 具有与 Acceptor 的 ACL 链接。如果 Acceptor 在运行基于 CAP 的 profile，则它们可能是 Coordinated Set 的成员，这种情况下，包含 Broadcast Assistant 的 Commander 必须允许 CAP 前导过程，以确保其在所有集合成员上运行。有时可能需要多个 Commander，这样用户可以在智能手表和手机中都安装 Commander 应用程序，还可以使用 remote control 作为额外的 Commander。

### 8.6.2 Scan Delegators

Commander 代表具有 Scan Delegator 角色的设备工作。角色定义在 BAP 中，通常在 Broadcast Sink 中实现。它的任务是找到其他担任 Broadcast Assistant 角色的设备，可以接管 Broadcasters 的扫描工作。Scan Delegator 通常可以在 Commander 中实现，因为多个 Commander 可以被串起来，有效地将信息从一个 Commander 传递到另一个。我们会在后面的章节中看到这一点的好处。

Broadcast Sink 中的 Scan Delegator Role 总是包含 BASS 的实例。Scan Delegators 请求 Broadcast Assistants，Broadcast Assistants 可以通过使用 Extended Advertising PDUs 来发送请求来代表它们进行扫描，Extended Advertising PDUs 包括了含有 BASS UUID 的 Service Data AD Type。

任何范围内的 Broadcast Assistant 都可以连接并让 Scan Delegator 知道它已经代表其开始了扫描了，通过 Broadcast Sink 上的 BASS 实例进行交互，写入 Broadcast Audio Scan Control Point 特征[BASS 3.1]，以确保它正在进行扫描(opcode = 0x01)或停止扫描(opcode = 0x00)。由 BASS 实例记录多个

---

Client 的该状态，以确定任何 Client 是否在执行扫描。代表 Scan Delegator 进行扫描的过程称为 Remote Broadcast Scanning。一旦它知道了 Broadcast Assistant 正在代表它进行扫描，Broadcast Sink 可以决定终止自己的扫描过程以节省功耗。Broadcast Assistant 可以读取 Broadcast Sink 的 PAC 记录以发现其 capabilities，并使用这些信息来过滤它发现的 Broadcast Sources，以便向 Scan Delegator 提供对 Broadcast Sink 有效的选项。最常见的是，这将考虑 Broadcast Sink 的 Audio Location、支持的和可用的 Context Types 即其支持的 Codec Configurations。

一旦 Broadcast Assistant 检测到合适的 Broadcast Source，它可以通过向其 Broadcast Receive State 特征之一写入，以此开启 Broadcast Sink 获取广播音频流，来通知 Scan Delegator。这是自动化行为，也可以是用户发起的行为。用户可能希望自动连接到已知的 Broadcast Source，例如当他们走进礼拜场所或办公室时。或者，在选择首选的 Broadcast Source 之前，他们可以要求手机使用扫描应用程序扫描并让他们知道有哪些可用的 Broadcast Source。这种选择是特定于实现的。

### 8.6.3 Adding a Broadcast Source to BASS

一旦在 Commander 上进行了选择，Broadcast Assistant 将其写入到 Broadcast Sink 上的第一个空的 Broadcast Receive State 特征，添加所选的 Broadcast Source 信息[BAP 6.5.4]。在此之前，它已经读取了当前 Broadcast Receive State 特征的状态。如果用户选择了已经存在的 Broadcast Source 其中之一，则应使用 Modify Broadcast Source procedure [BAP 6.5.5] 来修改适当的 Broadcast Source 值。还应检查 Broadcast Sink 是否能够解密任何建议的 Audio Source。

使用 Add Source Operation 操作码 0x02[BASS 3.1.1.4]写入特征的参数分为三大类，如表 8.7 所示。

Parameter	Size (Octets)	Description
Source Information		
Advertiser Address Type	1	Public (0x00) / Random (0x01)
Advertiser_Address	6	Broadcast Source 的广告地址
Advertising_SID	1	广告集的 ID
Broadcast_ID	3	Broadcast Source 的 Broadcast ID
Action		
PA_Sync	1	0x00: 不同步 0x01: 使用 PAST 同步 0x02: 不使用 PAST 同步
PA_Interval	2	The SyncInfo field Interval
Num_Subgroup	1	BIG 中 subgroup 的数量
BIS_Sync [i]	4	BIS_Index values to connect to: 0x00: 不要同步到 BIS_Index[i[k]] 0x01: 同步到 BIS_Index[i[k]]

Configuration		
Metadata Length[i]	1	BIG 中第 i 个 subgroup 的 metadata 长度
Metadata [i]	Varies	BIG 中第 i 个 subgroup 的 metadata

表 8.7 Add Source Operation 的格式

**第一组参数**——Source Information，将 Broadcast Source 和 BIG 的 ID 通知到 Broadcast Sink，以便它知道该信息所指的设备(更准确地说，设备标识)。Advertising Set ID——SID，包含在 primary 广告中，并且在 Extended Advertising 建立时定义[Vol 4, Part E, Section 7.8.53]，通常在设备生命周期中保持不变，并且在 power cycles 之间不能改变。虽然它只是一个字节，但它可用帮助标识设备。Broadcast\_ID 在 Extended Advertising 的 AUX\_ADV\_IND 中，并且在BIG生命周期中是不变的(可能比 SID 短一点)。连同广告地址，Broadcast Sink 用于扫描 Broadcast Source 的信息已经足够。

**下一组参数**告诉 Broadcast Sink 应该做什么。通常由用户在其 Commander 的用户界面上选择 Broadcast Source 触发。PA\_Sync 告知它同步到 periodic advertising train 上，从而获得 BASE 和 BIGInfo。将 PA\_Sync 设置成 0x01 或 0x02 就会 Broadcast Sink 执行此操作。PA\_Interval 是连续 AUX\_ADV\_IND 数据包之间的间隔，如果已知，可用帮助扫描。Num\_Subgroup 提供 BASE 中的 subgroups 的数量，然后是最重要的参数 BIS\_Sync。

BIS\_Sync 是从 Commander 到 Acceptor 的指令，告诉它连接到哪个 Broadcast Stream 或 Streams。它是所有的 BIS 的四字节 bitmap，对于 Acceptor 应该连接到的任何流，其值设置为 0x01。这里有一点微妙之处，就是为什么它们是排列起来的，而不是单个字节，这是因为这样对于每个 subgroups 能有效地屏蔽。因此，每个 BIS\_Sync[i]仅对该 subgroups 有效——所有其他的值都设置成 0b0。这样做的原因是，由于 subgroups 所有目的就是将相关联的 stream 安排在一起，因此，绝不期望 Acceptor 同时从不同的 subgroups 接收 BIS。如果两个 subgroups 使用不同的语音，比如日语和德语，那么你只能选择一种。

图 8.5 展示了 subgroups 中的 BIS\_Index 如何与 BIG 中的排序相关联，其中 BIS 按数字顺序排列。在此情况下，那它们不和 BIS\_Index 值一致，因此需要 BIS\_Index 映射。在写入 BIS\_Sync 值时，只能选择一个 subgroups。因此，如果从图 8.5 中选择了 subgroups 0，BIS\_Sync [0] 的唯一允许值将是 bit 0 或 bit 1 中的一个或两个设置为 0b1。

Broadcast Assistant 可用将 BIS\_Sync 设置成 0xFFFFFFFF，意味着“无

偏好”，这种情况下，Broadcast Sink 可以自行决定使用哪个 BIS。

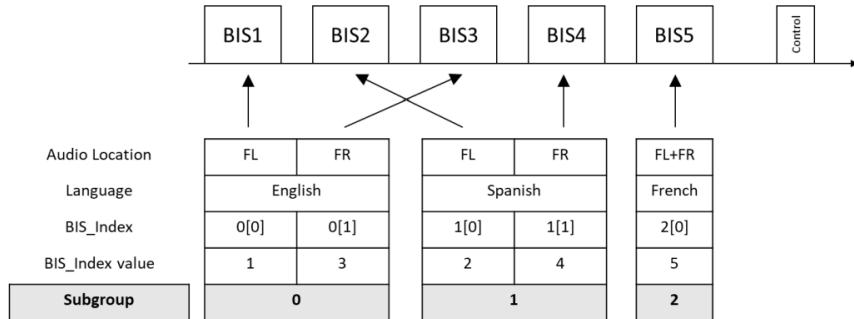


图 8.5 BIS number 到 BIS\_Index 映射

最后，metadata 域为每个 subgroups 提供 Level 2 metadata——通常是 Audio Contexts 和 Audio Locations，允许 Acceptor 决定它们是否适合其想要的，即它们是否匹配当前 Available Audio Context Type 设定。在 Add Source operation 中发送的数据(写入每个 Broadcast receive state)可以随时通过 Modify Source operation 进行修改，Modify Source 操作会修改当前信息。尽管在这两个操作中传递了特定的连接信息，但完全取决于 Acceptor 是否按照写入的与 PA 和 BIS 同步的请求。取决于实现。

如果 Acceptor 决定根据 Commander 的指令行动，则其会使用指令与 Periodic Advertising 序列同步，通过使用 PAST，或使用提供的 Broadcast Source 信息进行扫描，从 PA 中的 ACAD 和 Basic Audio Announcement 中检索出 BASE 和 BIGInfo。这些信息为其提供了与 BIG 同步所需的所有信息，之后它将使用 BIS\_Sync 值来选择其接收的特定 BIS 或 BISes。

PAST——Periodic Advertising Sync Transfer procedure [Core, Vol 6, Part B, Section 9.5.4]是最有效的方法，因为 Broadcast Assistant 向 Scan Delegator 提供了 SyncInfo 数据，允许其直接同步到 PA。这个过程叫做 Scan Offloading。

断开可由 Acceptor 自主执行，或者 Commander 可以使用 Modify Source 操作将适当的 BIS\_Sync bit 设置成 0b0。它还可以告诉 Acceptor 停止与 PA 的同步，但是保留其余的 Source 信息。或者，它可以通过对该 Source\_ID 执行 Remove Source 操作来删除 Source 记录。

#### 8.6.3.1 Set, Source and Broadcast IDs

值得对这些操作中涉及的不同 ID 进行快速精确分享，因为它们可能会令人困惑。

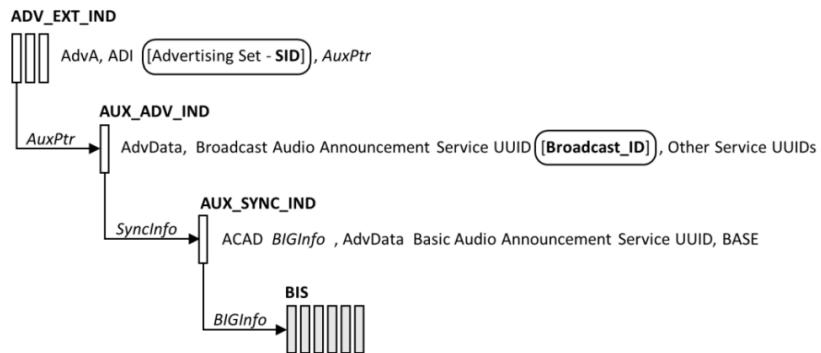


图 8.6 Set 和 Broadcast IDs

如图 8.6 所示，Set 和 Broadcast IDs 是 Broadcast Source 的特性。两者都是随机产生的号码。Advertising Set\_ID 包含在 primary advertisements 中，并标识特定的一组 Extended 和 Periodic Advertising 序列。SID 在设备的生命周期通常是不变的，而且在每个 power cycle 内必须保持不变。当 Acceptor 或 Commander 想要经常地重连到一个已知的 source(无论是个人设备还是固定的、基础设施广播发送器)，SID 可能很有用。

Broadcast\_ID 是特定于 BIG 的，并且携带在 Broadcast Audio Announcement Service UUID 中。在 BIG 的生命周期中是不变的。

Source\_ID 是 Acceptor 产生的号码，用于识别广播设备和 BIG 信息的特定集合。对于 Acceptor 来说，它是本地的，并且用于 Broadcast Assistant 的参考。在 Acceptors 的 Coordinated Set 情况下(如左和右耳塞)，Source\_IDs 不相关，并可能不同，即使两者接收相同的 BIS，因为每个 Acceptor 独立创建自己的 Source ID 值。

### 8.6.1 The Broadcast Receive State characteristic

在 Acceptor 上，Broadcast Receive State 特征用于通知任何 Broadcast Assistant 其关于广播流的当前状态。其结构如 8.8 所示。

Filed	Size (Octets)	Description
Source_ID	1	由 Acceptor 分配
Source Address Type	1	来自 Add / Modify Source 操作或自发行
Source_Address	6	来自 Add / Modify Source 操作或自发行
Source_Adv_SID	1	来自 Add / Modify Source 操作或自发行
Broadcast_ID	3	来自 Broadcast Audio Announcement Service UUID
PA_Sync_State	1	当前与 PA 的同步状态: 0x00——未与 PA 同步 0x01——SyncInfo 请求 0x02——已与 PA 同步 0x03——同步失败 0x04——No PAST 其他值——RFU
BIG_Encryption	1	加密状态: 0x00——未加密

---

		0x01——Broadcast_Code required 0x02——Decrypting 0x03——Bad_Code(密钥错误)
Bad_Code	Varies	如果 BIG_Encryption = 0x03 (错误密钥), 此处包含当前、错误的密钥。否则不存在。
Num_Subgroups	1	Subgroups 个数
BIS_Sync_State[i]	4	第 i 个 subgroups 的同步状态
Metadata_Length[i]	1	第 i 个 subgroups 的 metadata 长度
Metadata[i]	Varies	第 i 个 subgroups 的 metadata

表 8.8 Broadcast Receive State 特征的格式

与可由 Client 操作设置的其他特征以及可由 Server 自主行为更改的其他特征一样, Broadcast Receive State 特征可由 Client 使用以检查操作和状态, 并还可以由 Server 通知, 以指示操作已完成, 例如与 PA 同步, 或请求来自 Client 的动作。当它们收到通知, 其中一些作为错误报告, 而另外一些则向 Broadcast Assistant 发出请求信号。其中, 最重要的是 PA\_Sync State 为 0x01, 以请求 SyncInfo, 以及 BIG\_Encryption State 为 0x02, 以请求 Broadcast\_Code。BIS 同步的自发改变, 或 BIS 丢失, 都可以通过更新 BIS\_Sync\_State 来通知。

## 8.7 Broadcast\_Codes

加密广播音频流的能力使 Bluetooth LE Audio 进入一个全新的音频应用领域。加密有效地提供了一种 pseudo-geofencing 功能, 将广播覆盖范围限制为拥有解密密钥的人。可以用于限制特定区域内重叠广播的访问, 与使用 Wi-Fi 密码的方式相同。例如, 酒店可以分配密码, 以防止人们从相邻的会议室或楼上的电视收听音频。通过提供通过 Bluetooth 链路发送加密密钥, 或允许 Broadcast Assistant 通过 out-of-band 方式获取密钥, 它变得更加强大。在第 12 章中, 我们将研究 Broadcast\_Code 分发给不同应用程序的不同方式。

实现音频共享的个人设备, 如电视、平板电脑和笔记本电脑, 可以将 Broadcast Assistant 和 Broadcast Source 放在一起。然后, Assistant 可以直接访问 Broadcast\_Codes, 并可以将它们直接写入 Broadcast\_Sink。需要 Acceptors 和 Broadcast Source 配对, 但是这是个人设备所期望的。

Broadcast\_Code 的生命周期取决于实现。在某些情况下, 它可能是永久性的——在咖啡店播放音乐的 Broadcast Source 可能永远不会更改其密码。酒店房间内的电视将在客人入住期间保持其 Broadcast\_Code; 个人电视可能会在每一次 power cycle 时更新它, 这样来访的朋友就不会无限期地保留它, 而与朋友共享音频的手机应用程序可能会在每次会话时更新它。这些不同的生存期意味着 Broadcast Sources 和 Commanders 需要支持各种不同的方

---

法来获取 Broadcast\_Codes。

## 8.8 Receiving Broadcast Audio Streams (with a Commander)

了解了 Commanders 之后，我们可以重新审视设备在现实世界中怎样接收广播流，这几乎总是涉及 Commander，无论是独立设备、可穿戴设备、手机应用程序，还是广播电视内置的 Broadcast Assistant。

### 8.8.1 Solicitation Requests

第一项工作是 Scan Delegator 找到自己的一些 Broadcast Assistants，这是通过发送包含 Broadcast Audio Scan Service UUID 的 Service AD Data Type 的 Extended Advertisements 来完成的。这些被称为 Solicitation Requests [BAP 6.5.2]。

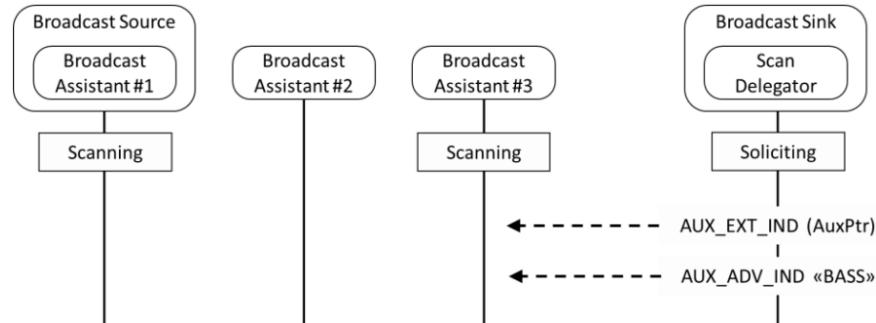


图 8.7 Scan Delegator solicitation process

图 8.7 显示了发送包含 BASS Service UUID 的 Extended Advertisements 的 Scan Delegator。左边有三个 Broadcast Assistants 在范围内。Broadcast Assistant #1 与 Broadcast Source 并置，并且正在进行扫描。Broadcast Assistants #2 和#3 仅具有 Commander 角色，但只有 Broadcast Assistant #3 在进行扫描。这种情况下，Broadcast Assistant #1 和#3 都应该响应 Solicitation 请求。

CAP 一如既往地告诉每个 Broadcast Assistant 从连接开始，建立 Coordinated Set 的成员，然后安排接下来的 BAP 程序。此时，每个 Broadcast Assistant 应读取每个 Broadcast Sink 的 PAC 记录以确定其 capabilities，主要是因为它们没有必要向 Broadcast Sink 告知 Broadcast Sink 无法接受的广播音频流。做出这个决定可能有很多原因。可能是因为广播音频 stream 具有 Broadcast Sink 无法支持的 Context Type；可能具有无法解码的编 codec 配置；具有不兼容的 Channel Allocation 或 Audio Location；需要加密，诸如此类。Coordinated Set 中只有一个 Acceptor 需要执行 Solicitation

过程。一旦 Broadcast Assistant 做出响应并确定它是 Coordinated Set 的成员，CAP 就要求它找到其他成员，并从这一点开始读取其所有 PAC 记录，然后与每个成员上的 BASS 实例交互。

图 8.8 展示了 Broadcast Assistant #3 的简化流程示例，响应 Coordinated Set 的一个成员的 Solicitation 请求。它连接到 Broadcast Sink #1 中的 Scan Delegator，发现它是具有两个 Acceptors 的 Coordinated Set 的成员，然后找到并连接到第二个成员，然后找到并连接到第二个成员。

它发现并读取每个 Acceptor 的 PAC 记录，并将设置其广播过滤策略，以便只报告两个接收方都可以接收的广播音频流。

**接下来**，它将发现并读取 BASS Broadcast Receive States 状态，并为这些特征建立通知，以便了解任何更改。读取状态知晓 Acceptors 当前是否与任何 Periodic Advertising 序列同步，或者是否正在接收任意 BIS。一旦完成了这些过程，它就准备好代表 Acceptors 开始扫描，并写入 Broadcast Audio Scan Control Point 特征以通知接收者。

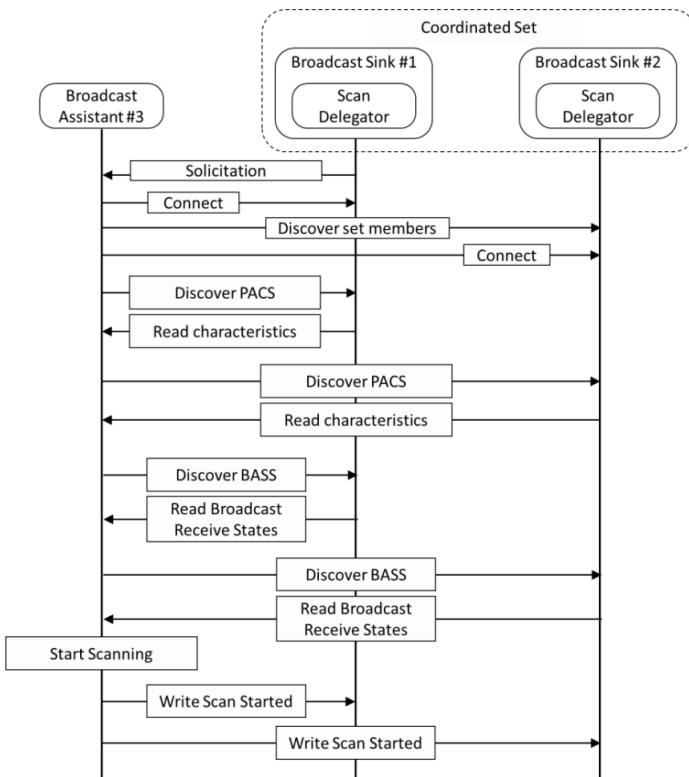


图 8.8 Broadcast Assistant 发现 Broadcast Sink Capabilities 和 State  
一旦完成，并且至少有一个代表 Scan Delegator 的、激活的 Broadcast

---

Assistant 扫描，我们将进入 BAP 的 Broadcast Assistant 过程，从 Remote Broadcast Scanning 开始[BAP 6.5.3]。

### 8.8.2 Remote Broadcast Scanning

通知 Scan Delegator 正在扫描的 Broadcast Assistant 将开始搜索广播发送器。当它找到每个 Advertising Set 时，它将通过 Extended and Periodic Advertisements 广告(AUX\_EXT\_IND 和 AUX\_ADV\_IND)，检查内容并解析结构，以确定广播音频流是否与 Acceptor 的能力匹配。在大多数情况下，它将建立一个相关的、可用的广播音频流列表，并将其呈现给用户。如果 Commander 有合适的用户界面，例如智能手机上的应用程序，则可能会以排序列表的形式显示，用户可以从中选择要连接的广播音频流。

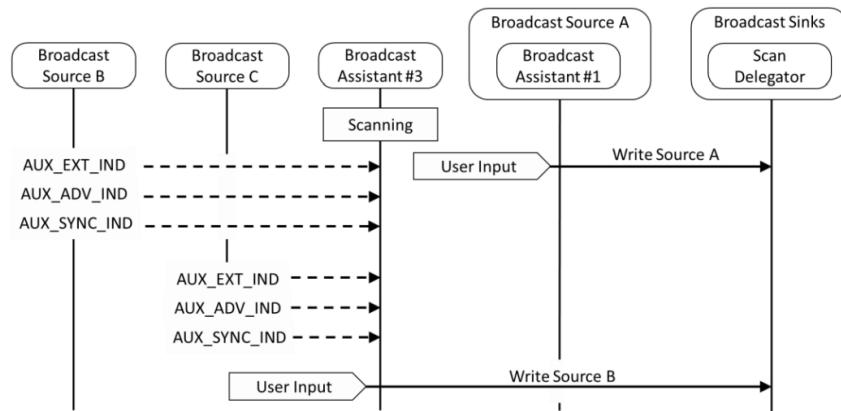


图 8.9 Remote broadcast scanning

图 8.9 继续图 8.8，其中 Broadcast Assistant#3 代表两个 Broadcast Sinks(此处显示为单个实体)进行扫描。它已发现 Broadcast Source B 和 Broadcast Source C。用户在 Commander 上确定收听广播源 C 并选择它，此时 Broadcast Assistant#3 将 Broadcast Source C 的信息写入两个广播接收器的 Scan Delegators 中的 BASS 实例。

图 8.9 还显示了并置的 Broadcast Assistant 的行为。在这种情况下，Broadcast Assistant #1 与 Broadcast Source A 并置。Broadcast Assistant #1 从未告诉 Broadcast Scan Control Point，说它代表 Scan Delegator 进行扫描，因为它的唯一目的是向 Scan Delegator 提供关于其并置的 Broadcast Source 的信息。一旦连接到 Scan Delegator，它就可以将该信息写入 Broadcast Receive State 状态。(并置的 Broadcast Assistant 也可以执行扫描，在这种情况下，它会告诉 Scan Delegator，它正在代表 Scan Delegator 进行扫描，但

---

这是一种实现选择)。

需要记住的一点是，Broadcast Assistants 中的接收器可能比耳塞中的接收器具有更好的灵敏度，因为它们可能具有更大的天线。这意味着他们可能会检测到超出其 Broadcast Sinks 范围的 Broadcast Sources。实现可能希望确定来自 Broadcast Sources 的信号的 RSSI，并使用该信息来安排向用户呈现 Broadcast Sources 的顺序。

### 8.8.3 Receiving a Broadcast Stream

现在，用户已经决定了他们想要听什么，这将我们带回到 Adding a Broadcast Source 的操作，我们在第 8.6.3 节中对此进行了描述。在大多数情况下，当用户在其 Commander 上选择 Broadcast Source 时，Broadcast Assistant 不仅将该 source 的详细信息写入 Broadcast Audio Scan Control Point 特征，还将设置 PA\_Sync 和 BIS\_Sync 参数，以指示每个 Acceptor 开始接收相关流。

由于两个 Acceptors 是独立运行的，他们可能需要不同的时间来获得 PA 序列，然后才能与适当的 BIS 同步，特别是如果他们必须扫描而不是使用 PAST。这可能会导致音频开始在两个耳塞中呈现的时间之间出现明显的延迟。Bluetooth 规范没有包含如何解决这一问题的任何细节，但一种方法是由 Commander 分两个阶段完成，首先设置 PA\_Sync 位，等待两个 Acceptor 同步的通知，然后发送 Modify Source 操作，指示它们同步 BIS 并接收和呈现音频。

### 8.8.4 Broadcast\_Codes (revisited)

在第 12 章中，我们将了解 Broadcast\_Codes 对新音频应用程序的重要性。如果我们回顾图 8.9，并假设来自 Broadcast Source A 和 Broadcast Source C 的广播音频 stream 都是加密的，那么可能会有一种稍微不同的方式来获得两个 Broadcast\_Codes。

Broadcast Source A 的 Broadcast Assistant 知道流的 Broadcast\_Code — 这就是为什么将其并置。因此，一旦用户选择 Broadcast Source A，将在 Add Source Command 中发送 Broadcast\_Code。

在 Broadcast Source C 的情况下，Broadcast Assistant 可能不知道密码，特别是如果它还没有与 Broadcast Source C 配对。在这种情况下，Scan Delegators 之一可以通过通知适当的 Receive State 特征(将 BIG\_Encryption filed 设置为 0x01)来询问其所有 Broadcast Assistants 是否知道

---

`Broadcast_code`, 表示其需要 `Broadcast_Code`。如果 `Broadcast_Sink` 已经具有此流的 `Broadcast_Code`, 并且此流不再有效(通常是因为它是在较早的会话中获得的), 则它应该将 `BIG_Encryption_file` 设置为 `0x03` 以指示错误的 `Broadcast-Code`, 并将该值包括在 `Bad_Code_file` 中, 这样具有相同错误代码的 `Broadcast Assistant` 就不会浪费时间重新发送它。如果任何 `Broadcast Assistant` 知道密码, 就可以通过使用 `Set Broadcast Code` 操作(`0x04`) [BASS 3.1.1.6]和 `Source_ID` 将 `Broadcast_code` 值写入 `Scan Delegator` 的 `Broadcast Audio Scan Control Point` 来响应此通知。

正如我们将在第 12 章中看到的, `Broadcast Assistant` 可以通过一系列 `out-of-band` 方法获得 `Broadcast_Code`, 从而开启一些有趣的新用例。其中的一些可以直接触发广播音频流获取过程。

#### 8.8.5 Ending reception of a broadcast Audio Stream

广播音频流的接收可以由 `Acceptor` 自主地执行, 在这种情况下, 它将在 `Broadcast Receive State` 特征中通知这种变化。如果 `Commander` 收到此通知并知道 `Acceptor` 是 `Coordinated Set` 的成员, 则可决定通过在其 `Broadcast Audio Scan Control Points` 中写入适当值来终止其他 `Acceptor` 的接收。然而, 这是特定于实现的。

或者, 用户可以使用 `Commander` 通过写入每个 `Acceptor` 的 `Broadcast Audio Scan Control Points` 停止接收, 所有 `subgroups` 中的所有 `BIS` 的 `BIS_Sync` 值设置为零, `PA_Sync` 值设为 `0x00`。一旦 `Acceptor` 通过通知其 `Broadcast Receive State` 特征(`PA_Sync_State` 和 `BIS_Sync_State[i]` 均设置为零)指示其不再与 `BIS` 或 `PA` 同步, `Broadcast Assistant` 执行 `Remove Source` 操作(`0x05`)[BASS 3.1.1.7], 以删除相关的 `Broadcast Receive State`。请注意, `Broadcast Sink` 没有状态机。它使用其 `Broadcast Receive State` 特征来同步或释放广播音频流。

### 8.9 Handovers between Broadcast and Unicast

CAP 中定义了两个切换过程, 以涵盖 `Initiator` 希望在单播和广播流之间切换(反之亦然)以传输相同音频内容的用例。这与从单播使用情况更改为广播使用情况的一般情况不同, 但它特定于与朋友共享个人音频之类的应用程序, 用户希望从单个 `personal stream` 转换为广播流, 反之亦然。CAP 中的过程允许 `Acceptor` 从 `Initiator` 接收并发的单播和广播音频流, 目的是切换可以是无缝的, 没有明显的中断。(在大多数情况下, 情况并非如此, 因为

---

Acceptor 将没有资源同时接收两种类型的流，特别是如果它们使用更高采样率 QoS 配置之一）。为了提供流畅的收听体验，实现应尝试隐藏原来的 Acceptor 的传输中断，尽管这可以通过可听的音调或消息来实现。（加入广播音频流的朋友不会经历任何中断，因为他们不会收到原来的音频流。）

过程[CAP 7.3.1.10 和 7.3.1.11]还要求将用于 original stream 的任何 Streaming Context Types 和 CCID\_List 转移到重新配置的流。尽管 original stream 的用户将其从单播转换为广播，但他们仍将保持与其 Audio Source 的 ACL 连接，并期望任何控制功能继续，例如快进或音量控制，因此需要保持 CCID 关联。其他接收广播的人只能访问音频。

## 8.10 Presentation Delay – setting values for broadcast

通过单播，Acceptors 通过公布其 Maximum 和 Minimum Presentation Delay capabilities 以及其首选取值范围，让 Initiator 了解其支持 Presentation Delay 的能力。在广播中，Initiator 和 Acceptor 之间没有信息传输。这意味着 Initiator 设置的值可能不受决定接收它的所有 Acceptors 的支持。

BAP 要求所有 Broadcast Receivers 必须在其范围内支持 40ms 的值，因此，这可能是许多 Initiators 将使用的默认值。然而，它开始引入延迟，这对于实时音频来说可能是过度的。

TMAP 和 HAP 对 Presentation Delay 的值施加了更严格的限制，要求 Acceptor 必须支持 20ms。由于大多数 Acceptors 都有 TMAP 或 HAP (大多数可能同时支持这两种)，因此为 Broadcast Source 设置 20ms 似乎是一个合理的折衷方案。然而，仅仅兼容 BAP 的设备可能不支持此功能。

这就提出了一个问题，即如果 Acceptor 不支持 Broadcast Source 公布的 Presentation Delay 值，该怎么做。规范对此没有提及，但以下指南提供了一种实用的方法。

- 如果 Presentation Delay 低于接受者可以承受的时间，则应使用 40ms 的值(所有设备都必须支持)。

该规范为 Broadcast Source 提供了三个字节来携带 Presentation Delay 值。超过了 65ms 的限制，如果只使用两个字节(单位为 1 $\mu$ sec)，这将是最大值，但这意味着该值可能高达 16.7 秒。大多数 Acceptors 用于缓冲音频流的内存有限，因此可能会出现 Presentation Delay 值过高超出其能力范围的情况。虽然他们可以决定不呈现音频流，但实用的方法是恢复到 40 毫秒的 BAP 值。

---

在这两种情况下，这将确保左侧和右侧设备同时呈现。如果他们能够相互通信，他们可以使用专有的方法来确定最合适的价值，这通常会选择他们支持的最低共同值。

---

## 第9章 Telephony and Media Control

在建立单播和广播流的复杂过程之后，到了涵盖 Telephony 和 Media control 的四个规范，它们虽然很全面，但是非常简单。在 Bluetooth 技术初次被开发出来时，我们的大多数移动产品都相当简单，仅做一件事。移动电话使用蜂窝网络拨打电话，音乐播放器根据其支持的任何物理媒体播放音乐——通常是预先录制的盒式磁带或 CD。从那时起，电话通讯和媒体(我们指的是音乐和其他我们可以开始和停止的音频)变得更加复杂了。

对于 Telephony，我们不再将手机局限于蜂窝网络。向 Voice over IP (VoIP)的转变见证了基于互联网的 Telephony 服务的爆炸式增长，无论是作为手机上的“Over The Top”(OTT)应用程序，还是作为笔记本电脑和 PC 上的程序。疾病的大流行和相关的在家办公的增长加速了它们的使用。虽然我们仍然使用手机通话，但我们也使用 Skype、Zoom、Teams 和越来越多的其他 Telephony 服务，有时同时使用多个 Telephony 服务。

Media 也发生了变化。正如我们在第 1 章中所看到的，Bluetooth 音频的增长与音乐流服务的增长并驾齐驱。我们不再拥有我们所听的大部分内容，而是按需借用。这意味着传统的“Stop”、“Play”、“Fast Forward”和“Fast Reverse”控制需要得到新的补充，这些新的控制可以超越物理设备，到达音频源。

Bluetooth Classic Audio profiles 的控制机制一直难以跟上这一发展，特别是在 Telephony 方面，它们仍然基于旧的“AT”命令集，在 20 世纪 90 年代被集成到早期的 GSM 标准和 2000s 年代早期 Bluetooth 技术的 Headset 和 Hands-Free Profiles 之前，最早到 1981 年，该命令集被设计用于陆线调制解调器。Bluetooth LE Audio 的发展使我们有机会回到 Telephony and Media Control 的第一原则，通用状态机同样适用于蜂窝电话和 VoIP 电话，以及各种本地和远程媒体源。

当前，使用两组 profile 来定义 Content control。就 Telephony 而言，它们是：

- TBS——Telephone Bearer Service，定义处理呼叫的设备的状态，以及
- CCP——Call Control Profile，在 TBS 上运行的 Client。

对于 Media，相应的一对规范为：

- 
- MCS——Media Control Service，定义了媒体源的状态，以及
  - MCP——Media Control Profile，MCP 上运行的 Client。

## 9.1 Terminology and Generic TBS and MCS features

到目前为止，我一直在使用 CAP 中的 Initiator 和 Acceptor 术语作为描述音频流的最简单方法。现在我们看的是与音频流正交的控制，这已经不合适了。由于 Bluetooth LE Audio 基本架构将音频数据平面和控制平面分开，意味着可以在不参与音频流的设备上实现控制功能。因此，我们需要在本章恢复到 Client-Server 术语，其中 Server 就是 Service 的实例——定义媒体播放器或电话的状态。对于 Telephone Bearer 和 Media Control Services，Server 位于 Initiator 上。Client 可以在 Acceptor 上的，但可能同样在 remote control 上，因为这样更容易使用，特别是对于耳塞和助听器这样的小型设备。我们已经习惯了遥控器的易用性——这就是为什么它们被发明了。它不需要看上去像传统的遥控器——可以是智能手表、另一种可穿戴设备、甚至是电池盒上的按钮。此外，Server 上可以同时有多个 Client 操作 Server。你可以在耳塞上接听电话，但用手表终止电话。

正如我们将看到的，control profiles 可以让我们在 Initiator 上的媒体和通话状态之间来回变动，反映用户开始或接听电话，或播放或暂停音乐的愿望。他们不会开启或停止与这些决定相关的任何音频流。**控制命令与传输音频数据的音频流的配置和建立之间的关联**完全取决于实现。Initiator 上的应用程序可以随心所欲地将它们关联在一起。

在 TBS 和 MCS 中，service 规范包括单独的 TBS 和 MCS 实例，可以为设备上的每个应用程序实例化。例如，如果您的手机支持 Skype、WhatsApp 和 Zoom 以及蜂窝电话，则可以为每一个应用程序提供一个单独的 TBS 实例。如果是媒体播放设备，则可以在 Spotify、BBC Sounds 和 Netflix 中包含 MCS 实例。

然而，如果你用一对耳塞控制电话或音乐，而用户界面非常有限，你不太可能想要去使用，甚至无法区分不同的应用。为了解决这一问题，TBS 和 MCS 规范均包含服务的通用版本，分别称为 Generic Telephone Bearer Service (GTBS) 和 Generic Media Control Service (GMCS)。它们的行为方式与 TBS 和 MCS 完全相同，但为 Server 设备提供了单一接口。将来自 Client 的输入命令映射到适当的应用程序取决于实现。

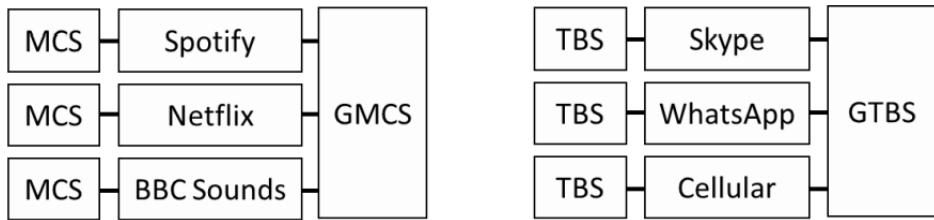


图 9.1 控制服务及其通用对应项的表示

图 9.1 是这一工作原理的简单表示。每个 **content control Server** 都必须包括通用服务的单个实例(GTBS 或 GMCS)，如果要将控制权直接公开给应用程序，则可以具有 TBS 和 MCS 的单独实例。您可以拥有与电话和媒体应用程序一样多的每个服务的实例，而映射则取决于实现。

各自的 **profiles**——CCP 和 MCP，定义了 Client 和 Server 角色，如图 9.2 所示。

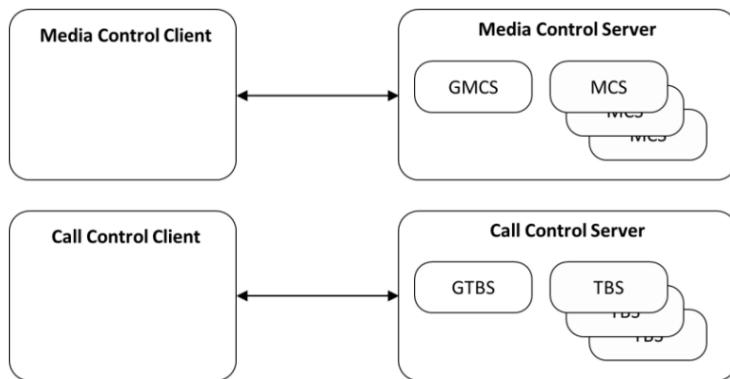


图 9.2 通用和特定的 **control service** 的关系

实现可以使用这些服务的组合来：

- 将每个应用程序视为唯一的媒体播放器或电话服务，
- 将设备视为单个电话或媒体播放器，命令作用于整个设备，或
- 一种组合，命令的子集可以指向特定的应用程序，根据其特定的 Content Control ID (CCID)进行映射。

在所有情况下，映射取决于产品的实现。

TBS 和 MCS 都包含多种功能，我们将在下面介绍。很多时候，Bluetooth LE Audio 都关注耳塞，因为耳塞是目前为止最大的市场。耳塞有限的用户界面(UI)可能会让读者感到奇怪——为什么包含这么多功能，以及为什么只有很少是强制性的。此问题忽略了 Bluetooth Audio 的历史，直到最近，Bluetooth Audio 的主要应用还在车载套件中，此应用中会提供非常丰富的控制和信息接口。TBS 和 MCS 都是为了复制当前车载套件中的许多功

---

能而设计的，这样的话，Bluetooth LE Audio 可以用于构建这些产品的下一代，并扩展它们以添加新功能，特别是用于媒体播放器的控制。

## 9.2 Control topologies

这两对控制规范的核心是定义 Service 规范中的状态概念。该状态保存在音频源设备上——MCS 的媒体播放器或充当 TBS call bearer 的网关的电话设备，但始终是 Initiator。Client 设备可以实现相应的 profile，该 profile 向 Server 上的 control point 写入，从而导致其状态转换。一旦进行了转换（可能在本地发生，用户按下按钮或触摸屏幕），Server 就会通知其新状态。这与我们在 ASCS 状态机中遇到的原理完全相同——有一个设备具有状态，多个 Client 可以读取和操作。在 MCP 和 CCP 的情况下，Client 可以是接收音频流的设备，也可以是包括 Commander 角色的设备，例如 remote control 或智能手表。然而，在这种情况下，如图 9.3 所示，Commander 中的 Client 将在 Initiator 而非 Acceptor 上操作。

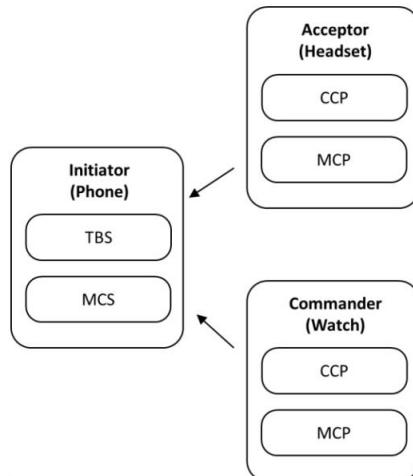


图 9.3 Content Control 拓扑

除了 control point 和 state 特征之外，每个服务还包含许多的特征选择，可以读取或通知这些特征，以确定关于应用程序和向它提供的外部服务的更多信息。

## 9.3 TBS and CCP

Telephony Control 规范的核心就是一个通用的 call state machine，如图 9.4 所示。

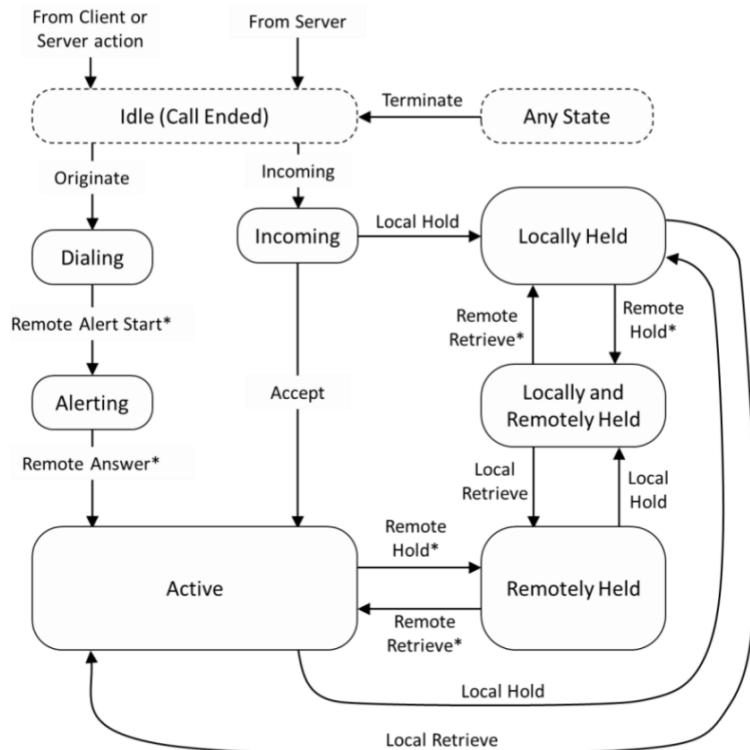


图 9.4 TBS 状态机

该状态机的中心是 Active 状态，表示正在打电话。大多数情况下，当 Initiator 和 Acceptor 之间建立了 Unicast Audio Stream(通常由 bidirectional stream 组成)，就会达到此状态。然而，如果使用有线耳机接听电话，或者将电话用于扬声喇叭(没有激活的音频流)，则状态机一样有效。本文着重于 control plane (在此用例中就是 TBS / CCP 的关系)和 audio data plane (BAP / ASCS 的关系)。根据应用程序的意愿将它们绑定在一起。(虽然它们是 GAF 的一个必要部分，但它们不限于 Bluetooth LE Audio，可以与其他应用程序一起使用)。

围绕状态机的转化可以由外部事件驱动，例如：

- 来电
  - CCP Client 向 TBS Call State 特征写入
  - 本地用户行为，如在电话上接听来电，或
  - remote caller 的操作。

图 9.4 中的远程操作(Remote Alert Start、Remote Answer、Remote Hold 和 Remote Retrieve) 在每个命令的末尾用(\*)标记。这些转换只能由 remote caller 造成。这些状态由 Call State 特征公布，此特征在呼叫改变的

---

任何时候都会通知到 Client。

### 9.3.1 The Call State characteristic

Call State 特征是设备上所有当前呼叫的一个数组。一旦任何呼叫进入 non-Idle 状态，TBS 将为其分配一个唯一的、单字节的顺序索引号，称为 Call\_Index，其值介于 1 和 255 之间，在 Call State 特征中通知。(TBS 的 Call Control 状态机不包含 Idle 状态，但为清楚起见，图 9.4 中显示了一个空闲状态)。Call State 特征的第二个字节标识每个呼叫的当前状态，最后一个字节保存与该呼叫相关联的 Call Flags。

Call State 特征的格式如图 9.5 所示。

Call_Index [i] (1 octet)	State [i] (1 octet)	Call_Flags [i] (1 octet)
-----------------------------	------------------------	-----------------------------

图 9.5 Call state 特征

表 9.1 列出了 Call state 特征中返回的值。

State	Name	Description
0x00	Incoming	来电(通常导致 ringtone)
0x01	Dialing	开始向外拨号，但是远端还没有被通知到(传统的拨号音状态)
0x02	Alerting	远端提醒开始(它们有 ringtone)
0x03	Active	呼叫建立好了，并开始交谈
0x04	Locally Held	呼叫是连接状态，但是本地处于挂起状态(见下面的内容)
0x05	Remotely Held	呼叫是连接状态，但是远端处于挂起状态(见下面的内容)
0x06	Locally and Remotely Held	呼叫是连接状态，但是本地、远端都处于挂起状态(见下面的内容)
0x07-0xFF	FU	留作后用

表 9.1 Call state 特征中定义的状态

#### 9.3.1.1 Local and Remote hold

状态 0x04、0x05 和 0x06 是指已挂起的呼叫。这些状态的区别在于是谁挂起了通话。0x04 表示它已在本地挂起，即由具有此 TBS 实例的电话或 PC 的用户挂起。0x05 表示远端将其挂起，0x06 表示两端都将其挂起。

通过向 Control Point 写入，可以将本地挂起的呼叫重新恢复(回到 Active 状态)。远程挂起呼叫需要由远端来恢复。本地在远端挂起的呼叫上唯一可以的操作是将其终止。

#### 9.3.1.2 Call Flags

Call State 特征的最后一个字节是包含该呼叫信息的位域，其含义和相应的值如表 9.2 所示。

Bit	Description	Value
0	Call Direction	0 = incoming call 1 = outgoing call
1	Information withheld by server	0 = not withheld 1 = withheld
2	Information withheld by network	0 = provided by network 1 = withheld by network

表 9.2 Call state 特征定义的 Call status bits

Bit 1 和 2 表示由于本地或网络策略而可能故意保留信息(例如 URI(通常 是 caller ID)或 Friendly Name)的情况。可以设置其中之一或两者都设置，此 种情况下，相关特征的 fields 可能为空或 null。或者，Server 可以用适当的 文本填充它们，例如“Withheld”或“Unknown Caller”。策略和文本的选 择取决于具体实现。

### 9.3.1.3 UCIs and URIs

Telephony 应用中你会遇到的两个首字母缩写词是 URI 和 UCI，它们代 表 Uniform Resource Identifier 和 Uniform Caller Identifier。两者都是被设计 用于向各种 Telephony 应用以 bearers 提供信息的。

Uniform Caller Identifier 本质上是“bearer”，比如，“skype”或 “wtsap”。UCI 的值列在 Bluetooth Uniform Caller Identifiers Assigned Numbers 文档中，并且通常缩写不超过 5 个字符。因此，WhatsApp 就是 “wtsap”。标准电话号码使用 UCI “E.164”或“tel:”，表示标准拨号器格 式。

Uniform Resource Identifier 是 UCI 和 caller ID 的组合，即呼叫者的号 码或用户名，具体取决于应用程序。它既可以用于来电，此处就是 Caller ID，也 可以用于呼出。应用程序可以使用呼出的 **URI 的 UCI 部分** 来选择适当的 Telephony 应用程序进行呼叫。URI 表示为 UTF-8 字符串。

### 9.3.2 The TBS Call Control Point characteristic

正如我们在 BAP 和 ASCS 中看到的，Client 可以通过向 Control Point 写 入，从而让 Server 在其状态机中来回转换。在此用例中，就是 TBS Call Control Point。需要一个参数，随后的参数由特定的操作码决定，如图 9.6 所示。

Opcode (1 octet)	Parameter (varies)
---------------------	-----------------------

图 9.6 TBS Call Control Point 特征

表 9.3 展示了当前的操作码以及它们的用途。

Opcode	Name	Parameter	Description
0x00	Accept	Call_Index	接听来电
0x01	Terminate	Call_Index	不管其状态，挂断与 Call_Index 相应的呼叫
0x02	Local Hold	Call_Index	将 Call_Index 指向 Active 或 Incoming 呼叫转换为 Local Hold
0x03	Local Retrieve	Call_Index	将 Local Hold 的呼叫恢复到 Active
0x04	Originate	URL	向 URI 拨打电话
0x05	Joint	List of Call_Index 值	将列出的 Call_Index 标识的呼叫合在一起

---

0x06-0xFF	RFU	留作后用
-----------	-----	------

表 9.3 用于 TBS 状态机转换的 Call Control Point 特征操作码

表 9.3 中的操作码是对 Server 的请求，Server 将其传递给相关的 Telephony 应用程序。其中的一些：Accept、Terminate 和 Originate，是指 Telephony 应用程序在本地执行的操作。Local Hold 和 Retrieve 通常需要传递回网络，Join 几乎就是一个网络功能。根据特定的 Telephony bearer 和应用程序，不是所有这些都可以被使用。基于当前的电话资源和网络连接，它们可能会失败。比如，有些电话服务不允许在通话时再去拨打电话。同样，如果没有蜂窝信号，尝试拨打电话就会失败。

TBS 实例可以通过使用 Call Control Optional Opcodes 特征来指示是否支持 Local Hold 和 Join 功能。这是一个位域，Call Control Client 可以读取来确定它是否可以发送这些命令。此域中的值通常是静态的，至少在一个会话期间是静态的，通常在设备的生命周期内也是静态的。如果操作成功了，会通知 Call State 特征，通知当前 State 和 Call\_Index 的 Client。在操作码是 Originate(0x04)的情况下，这是 Client 第一次知道 Call\_Index。

当 Client 写入 TBS Call Control Point 特征失败时，Call Control Point 特征会使用以下格式的操作码 Write 的结果通知出来：

Requested Opcode (1 octet)	Call_Index (1 octet)	Result Code (1 octet)
-------------------------------	-------------------------	--------------------------

图 9.7 TBS Call Control Point 特征通知格式

Result\_Code 表示 Success，或提供此操作失败的原因。这些结果码在 TBS 规范的表 3.11 中列出。

### 9.3.3 Incoming calls, Inband and Out-of-Band ringtones

在传统的 Telephony 设计中，用户知道来电的第一件事就是电话铃声。一旦引入 Bluetooth 连接，这就变得更加复杂。手机仍然可以听到铃声，或者铃声可能出现在耳塞中(如果你戴着的话)。或者两种情形都可能发生。

还有一个更复杂的问题。您在耳塞中听到的铃声可能与手机上的声音相同，也可能是耳塞产生的本地铃声，这会有所不同。如果声音与手机的声音相同，则需要来自手机中音频流，该音频流携带与手机扬声器播放的铃声相同的音频。这叫做 Inband 铃声。Inband 铃声的问题是，您需要建立音频流来传输它，这可能意味着将现有的音频流断开，而转移到另一个设备。想象一下你正在使用耳塞听电视里的声音，你的手机响铃了。对于 Inband 铃声，您的耳塞可能需要终止电视的音频流，并将其替换为来自手机的音频流。如

---

果你接电话，这不是问题，但如果你拒绝了，你需要重新连接到电视，这是一种糟糕的用户体验。

另一种方法是手机将其 Incoming Call 特征通知耳塞。它包含 Call\_Index 和 URI，即来电的 URI 策略和 Caller ID。耳塞中的 Call Control Client 可以生成本地铃声，提醒您有来电。如果您接听了，Call Control Client 将向电话的 Call Control Point 特征写入 Accept 操作码(0x00)。指示你的耳塞终止与电视的音频流，而电话(是不同的 Initiator)将建立双向音频流来传输该通话。

如果你拒绝接听电话，您的电视音频流将保持不变，因为它没有被中断——您的耳塞只会将其本地生成的铃声与电视音频混合。这是一种更干净的用户体验，但是，这确实意味着你的耳机会给你不同于手机的铃声。

如果耳塞可以执行 text-to-voice 转换，允许它播放 Incoming Call 特征中包含的电话号码，作为通话提醒的一部分，则可以增强 Out-of-Band 铃声的使用。如果手机支持可选的 Friendly Name 特征，该特征包含手机联系人列表中呼叫者姓名的文本，则也可以在耳塞内播放。

呼叫提醒的最后一个微妙之处是，可以将手机设置为静音模式，这种模式下，传入的铃声不会发送到手机(或 PC)扬声器，而是由 Acceptor 来通知。Call Control Client 可以通过读取表 9.4 所示的 Status Flags 特征来确定这一点，以及手机是否支持 in-band 铃声。

Bit	Description	Value
0	Inband Ringtone	0 = disabled 1 = enabled
1	Silent Mode	0 = disabled 1 = enabled
2-15	RFU	留作后用

表 9.4 ringtone mode 支持的 Status flags 特征

如果禁用了 Inband 铃声，Call Control Client 通常会在收到 Call State 特征已转变为 Incoming 状态的通知时通报有来电。请注意，这包括不是 Acceptors 的 Call Control Clients。例如，包含 Call Control Client 的智能手表可能会振动或响铃，尽管它不能接受音频流。这突出了一个事实，即 Call Control 状态机没有与 ASE 状态机的直接连接。一个不会直接触发另一个。将 Call Control 状态与音频流管理关联起来，完全取决于 Telephony 应用程序或操作系统。

如果 Status Flags 特征显示已启用 Silent Mode，则表示手机上不会播放铃声。根据 Inband 铃声(第 0 位)的状态，可以使用音频流将铃声发送给 Acceptor，或作为 Out-of-Band 铃声，或都是。如果两者都已启用，则由

---

Acceptor 决定使用哪个。

值得记住的是，可以在 Call Control Server(即电话或 PC)以及 Call Control Client 上执行接受或拒绝呼叫以及所有其他状态转换。所有连接的 Call Control Clients 具有同等访问权限；任何更改状态的操作都将产生通知。

### 9.3.4 Terminating calls

呼叫可以由任何 Call Control Client、电话上的用户操作或 remote caller 终止。它也可能由于各种原因而丢失，例如 Telephone bearer 网络上的故障或信号丢失。每当呼叫被终止时，Call Control Server 将使用 Termination Reason 特征来通知 Call Control Clients 终止的原因。与类似的特征一样，它包括 Call\_Index 来识别呼叫和终止原因，如图 9.8 所示。

Call_Index (1 octet)	Reason_Code (1 octet)
-------------------------	--------------------------

图 9.8 Termination Reason 特征

如果多个呼叫被终止(可能是连接的通话出现了网络问题)，则会为每个 Call\_Index 发送单独的 Termination Reason 通知。终止原因见

Reason Code	Reason
0x00	发起呼叫的 URI 格式不正确
0x01	呼叫失败
0x02	远端终止通话
0x03	Server 终止通话
0x04	线路忙
0x05	网络拥塞
0x06	Client 终止通话
0x07	无服务
0x08	无应答
0x09	未指定
0xA-0xFF	留作后用

表 9.5 Termination Code 特征 Reason\_Code 值

### 9.3.5 Other TBS characteristics

如上所述，对于 Call Control Server 角色，TBS 包含大量其他的特征。这些特征提供了有关 phone call 的更详细信息。它们可以由更复杂的 Client 读取，例如车载套件，通过镜像来自原来的电话应用程序的可用的信息级别，可以使用来它们复制手机的用户体验。

表 9.6 列出了上述未涵盖的其他特征。除了 Bearer Signal Strength 特征及其相关的 Bearer Signal Strength Reporting Interval 特征之外，GTBS 或 TBS 实例必须支持所有这些特性。它们都在 Telephone Bearer Service 规范中进行了描述。

Characteristic Name	Description
Bearer Provider Name	Telephony 服务。比如, Vodafone, T-Mobile
Bearer Uniform Caller Identifier	UCI, 比如, “skype”、“wtsap”, 来自于 Assigned Numbers
Bearer Technology	显示在电话上。比如, 2G、3G、WiFi
Bearer Signal Strength (Optional)	信号强度, 从 0(无信号)到 100
Bearer Signal Strength Reporting Interval (Optional)	报告信号强度的频率
Bearer URI Schemes Supported List	支持的 URI 策略
Bearer List Current Calls	当前呼叫以及它们的状态
Content Control ID (CCID)	CCID 值保持不变指导服务改变
Incoming Call Target Bearer URI	来电 URI。比如, 你的电话号码

表 9.6 TBS 规定的其他特征

有相应的过程来读取 Call Control Profile 中的所有 TBS 特征。虽然对 TBS 中的大多数特征的支持是强制性的, 但唯一强制性的 Call Control 过程是读取 Read Call 状态过程。这反映了一个事实, 即对于具有受限用户界面的设备, 如助听器和耳塞, 大多数控制动作可能发生在手机上。

## 9.4 MCS and MCP

一旦你了解了 Telephony Control, Media Control 规范将非常地相似。Media Control profile 定义了两个角色: Media Control Client 和 Media Control Server。后者驻留在 Initiator 上, 在那里, Media Control Service 定义了用于媒体播放的状态机(如图 9.9 所示), 以及用于通知其正在执行的操作的大量特征。

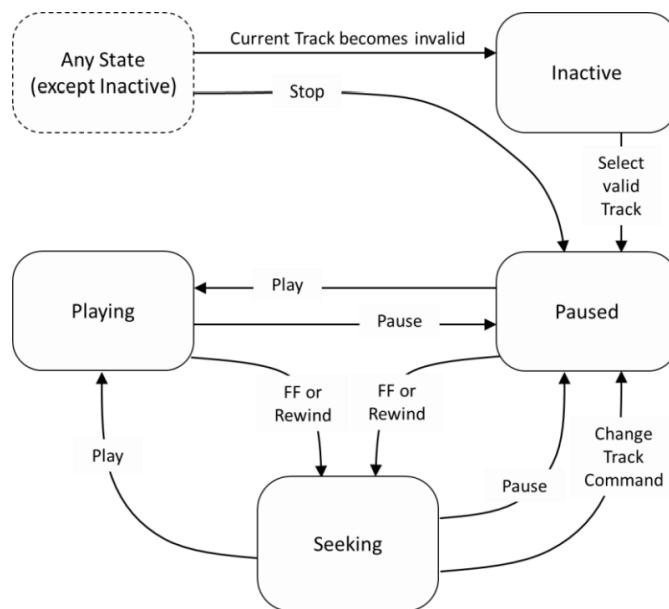


图 9.9 Media Control 的 MCS 状态机

对于 Media, 播放机通常处于 Inactive 状态, 当选择 Track 时, 会转到 Paused State 状态, 从此状态可以转换为 Playing。从 Playing 开始, 它可以

---

通过停止返回到 Paused 状态，或通过发出 Fast Forward 或 Fast Rewind 命令移动到 Seeking 状态。状态机中没有 Stop 状态或命令。对于数字媒体，Stop 和 Pause 命令之间没有真正的区别。这一区别可以追溯到模拟设备，其中 Stop 关闭了录音机或盒式磁带机的电机，而 Pause 则使电机保持运行，并将唱针或播放头从播放介质上抬起。当一切都在数字存储器中时，操作是相同的。

#### 9.4.1 Groups and Tracks

三种主要状态——Paused、Playing 和 Seeking 仅在选择了当前曲目时有效，这让我们去了解 Tracks 和 Groups 的概念，这是 MCS 的关键部分。图 9.10 展现了这个概念，它显示了 Groups 的结构，其包含了 Tracks。定义这些是如何构造完全取决于实现，但可能最好将一个 Group 想象成一张传统的专辑，其中曲目是单独的歌曲。Groups 可以嵌套成为更大 Groups 的一部分，因此，由贝多芬的全部作品组成的 Parent Group 可能包含管弦乐、室内乐、钢琴、声乐作品等 Subgroups，每个 Subgroups 可能包含更多 Subgroups 的合集。MCS 所需要的只是有一个定义的结构来创建从第一组开始到最后一组结束的播放顺序。

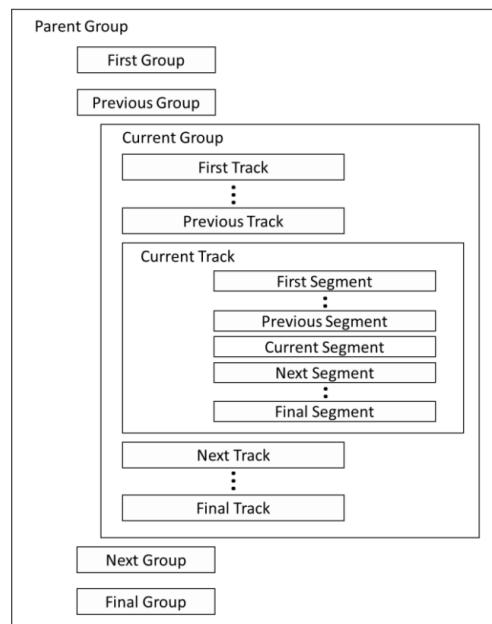


图 9.10 MCS 的 Groups 和 Tracks 安排

Groups 包含 Tracks，也可能包含 Segments。大多数 MCP 控制都在

---

Tracks 上进行，就是大多数人认为的唱片或 CD 上的常规曲目。具体来说，状态机的操作集中在当前曲目上。曲目的关键要素如图 9.11 所示，即 Track Duration、曲目两端的 Offsets 和 Playing Position。

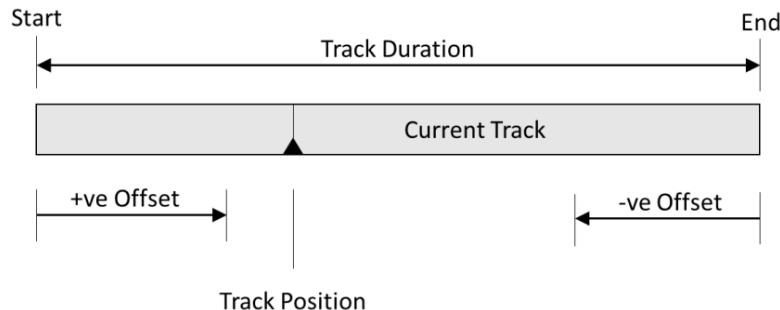


图 9.11 Track 的关键要素

#### 9.4.2 Object Types and Search

MCP 和 MCS 包括许多支持复杂用户界面的功能，例如汽车 AV 系统的屏幕。其中大部分是基于现有的 Bluetooth LE Object Transfer Service(OTS)，该服务允许扩展信息与 Groups 和 Tracks 相关联。这包括扩展名、图标和 URL，可用于获取更复杂的对象，如专辑封面。OTS 还用于返回搜索结果。MCS 允许一些非常全面的搜索功能。您可以使用曲目名称、艺术家名称、专辑名称、组名称、最早年份、最新年份和流派的组合进行搜索。有关这些的更多信息，请参阅 MSC 规范。它们不太可能在短期内用于产品中，因此我将跳过细节，专注于 Media Control 的关键方面。

#### 9.4.3 Playing Tracks

通过解释 Groups 和 Tracks 的结构，我们可以了解 Media Control 是如何工作的。一旦选择了曲目(通常是媒体播放器上的用户操作)，播放器将使用 Track Changed 特征通知其所有连接的 Clients。此特征没有值，即它是空的，但它是媒体播放器现在有一个当前曲目的声明，因此 Client 可以开始控制它。Client 可以读取 Track Title 特征和 Track Duration 特征。它还可以读取 Track Position，在曲目开始时会以 10 毫秒的分辨率返回。(允许的最长曲目持续时间正好超过 16 个月(这可能看起来很荒谬，但“twenty one pilots”小组的最长视频“Level of Concern”长达 177 天 16 小时 10 分 25 秒，很可能有人试图打破这一纪录))。

Client 现在可以使用表 9.7 种的值来写入 Media Control Point 特征了。

Opcode	Name
0x01	Play
0x02	Pause
0x03	Fast Rewind
0x04	Fast Forward
0x05	Stop

表 9.7 Media Control 的 Media Control Point 特征操作码

一旦当前曲目开始播放，Server 应通知 Track Position 特征，以通知 Clients 当前曲目位置。通知的频率取决于实现。

另外的操作码集合可以用于切换当前曲目，如表 9.8 所示。

Opcode	Name	Parameters
0x01	Move Relative	32bit signed integer
0x20	Previous Segment	无
0x21	Next Segment	无
0x22	First Segment	无
0x23	Last Segment	无
0x24	Goto Segment	32bit signed integer

表 9.8 用于曲目内跳转的 Media Control Point 特征操作码

Media Control Service 允许在曲目中定义片段。每个线段都包含一个名称和距离曲目起点的绝对偏移。表 9.8 中的命令允许移动到当前曲目的特定段。这些都不允许在当前曲目之外移动。如果参数导致移动到当前轨迹之外，则结果将限于将位置移动到曲目的起点或终点，但曲目仍然是当前曲目。

发送 Stop 命令会导致当前曲目变无效，用户需要选择另一个曲目作为当前曲目，表 9.7 中的命令才能再被使用。

另外两组操作码允许选择不同的曲目作为当前曲目。

Move within a Group			Move to another Group		
Opcode	Name	Parameters	Opcode	Name	Parameter
0x30	Previous Track	None	0x40	Previous Group	None
0x31	Next Track	None	0x41	Next Group	None
0x32	First Track	None	0x42	First Group	None
0x33	Last Track	None	0x43	Last Group	None
0x34	Goto Track	32bit signed INT	0x44	Goto Group	32bit signed INT

表 9.9 用于 Tracks 和 Groups 跳转的 Media Control Point 操作码

当选择另一个 Group 时，由命令生成的当前组的第一个曲目将变成当前曲目。

所有的 Segments, Tracks 和 Groups 都是从 0 开始计数的。当使用 Goto 命令时，正值 “n” 相当于转到第一曲目，然后再执行 n-1 次 “Next Track” 操作码。负值相当于转到最后一个曲目，然后执行 n-1 次 “Previous Track”

转到新曲目会将其变成当前曲目，并且将放到图 9.9 中状态机的 Paused 状态中。

只强制支持 Media Control Point 特征中至少一个操作码。任何实现都不太可能像这样简单，但 Clients 可以通过读取 Media Control Point Opcodes

---

Supported 特征来检查 Server 支持什么。这是一个位域，表示 Server 支持哪些操作码。

与 Control Point 操作码一样，Client 写入操作码并在 Media Control Point 特征中接收操作通知，Result\_Code 指示 Success、Opcode not supported 或 Media Player Inactive。一旦 Server 完成了请求，在合适的特征的值发生变化时，也会对此操作做出通知。

#### 9.4.4 Modifying playback

Media Control Service 有几个用于修改 Playback 和 Seeking 的功能。第一个功能是使用 Playback Speed 特征请求更改播放速度。这可以由 Client 写入，以请求更快或更慢地播放当前曲目。该特性的参数是一个 8 位有符号整数 “p”，其值范围从 -128 到 127，其中请求的实际播放速度为：

$$speed = 2^{p/64}$$

即速度是 2 的( $p/64$ )次方。这里给出了 0.25 到 3.957 的范围，就是标准速度的 1/4 到 4 倍。

播放速度为盲请求；Client 不知道是否支持 “p” 值。请求后，Server 通知 Playback Speed 特征中设置的值。如果请求超出 Server 支持的范围，则应将其设置为最接近的可用速度。如果媒体源是实时或流式的，则 Playback Speed 特征值可以固定为 1。如何调整速度以及是否保持音频数据的音调完全取决于实现。

另一个可以改变的特征是 Seeking Speed，通过将有符号的整数值写入 Seeking Speed 特征。这是当前 Playback Speed 的倍数，正值表示 Fast Forward，负值表示 Fast Reverse。“Seeking Speed” 值应用于当前的 Playback Speed，而不是默认的 Playback Speed，其中 p 的值将为 0。Seeking Speed 的值为 0 表示 Seeking Speed，该 Seeking Speed 与值为 p=0 的 Playback Speed 相同，与当前 Playback Speed 无关。

在 Seeking 过程中，应通知 Track Position 特征，以提供当前 Track Position 的指示。通知的频率由具体实现来决定。当 Track Position 到达当前曲目的起点或终点时，Seeking 将停止。

在 Seeking 时，MCS 规定不应播放音频数据。然而，描述使用场景的 Context Type(不是音频内容)通常不会改变。

#### 9.4.5 Playing order

MCS 左后一个特点时播放顺序。Playing Order 特征提供了 10 种播放曲

---

目的方式。当你在播放整个 Group 时，它们种的大多数都包含了播放顺序。这些选项在表 9.10 中列出。

Value	Name	Description
0x01	Single once	单曲播放一次
0x02	Single repeat	单曲循环
0x03	In order once	顺序播放 Group 中的所有曲目
0x04	In order repeat	顺序播放 Group 中的所有曲目，然后循环
0x05	Oldest once	按照年代升序播放 Group 中的所有曲目
0x06	Oldest repeat	按照年代升序播放 Group 中的所有曲目，然后循环
0x07	Newest once	按照年代降序播放 Group 中的所有曲目
0x08	Newest repeat	按照年代降序播放 Group 中的所有曲目，然后循环
0x09	Shuffle once	随机播放 Group 中的所有曲目
0x0A	Shuffle repeat	随机播放 Group 中的所有曲目，然后循环

表 9.10 Playing Order 特征值

其中有一些细微差别。播放当前曲目的前两个值（0x01 和 0x02）在完成或 Stopped 时将当前曲目设置为下一曲目。对于 shuffle，随机化留给实现。在大多数情况下，实现不是完全随机的，而是加权的，因为听众不希望在下一个循环开始时将相同的曲目放在一起。这些决定留给实现。Client 只需发送命令。如果 Server 无法支持请求的播放顺序，例如，当它不知道曲目的年份时，它应该忽略该命令。

与 Media Control Point 特征一样，Client 可以通过读取 Playing Orders Supported 特征来确定支持哪些 Playing Order 功能。这是一个 2 字节的位域，其位域的含义如表 9.11 所示。

Value	Name
0x0001	Single once
0x0002	Single repeat
0x0004	In order once
0x0008	In order repeat
0x0010	Oldest once
0x0020	Oldest repeat
0x0040	Newest once
0x0080	Newest repeat
0x0100	Shuffle once
0x0200	Shuffle repeat

表 9.11 Playing Orders Supported 特征中的 bit 的含义

这就是 Telephony 和 Media Control，我们下一步是 Volume、Audio Input 和 Microphone。

---

# 第10章 Volume, Audio Input and Microphone Control

任何从事音频规范工作的人都可能会告诉你，他们的大部分时间都用于讨论 Volume Control；这是一个比音频质量更能引发争论的话题。原因有两个。第一个是关于音量感知以及如何定义最小值和最大值之间的级别的无休止的学术讨论。第二个问题是如何应对多种不同的音量控制方法。第二个问题在电视或功放上只有一个音量旋钮时并不存在。然而，一旦您有了多个遥控器，用户就越来越难确定如何设置音量。

经过必要的数百小时辩论后，Bluetooth LE Audio 工作组决定或多或少地跳过第一个问题，专注于第二个问题——如何协调多个不同的控制点。所以，这一章都是关于 Volume Control 的，只是对音量的解释有一个粗略的介绍，因为这是留给实现的。我们还将讨论 Microphone Control，因为它是类似的，但涉及音频的输入，而不是其输出。所有这些功能都设计用于非编码音频信号。在讨论 volume 的情况下，这意味着在接收和解码之后使用这些功能；而对于 Microphone control service 和 profile，是在传输之前使用。

## 10.1 Volume and input control

完成了前面的几章之后，自然而然到了本章。有三个 service 和一个 profile 涉及音量：

- Volume Control Service (VCS)，可以作为主音量控制旋钮来看
- Volume Offset Control Service (VOCS)，可以看作是“Balance”控制
- Audio Input Control Service (AICS)，允许对任何数量的音频输入进行单独增益控制和静音，这些输入不需要是 Bluetooth 音频流，以及
- Volume Control Profile (VCP)，允许多个 Client 控制这些 Service 尽管它们的大多数名字里面有“volume”，但它们实际上做的是调整增益，即音频信号的放大。

图 10.1 展示了一对耳机或带式助听器的典型实现，它有三种可能的音频输入——Bluetooth Audio Stream、Telecoil audio 输入和 Microphone 输入。这些输入混合在一起，使用 Audio Input Control Service 来设置它们的单独增益，并选择性地静音和取消静音。生成的流的增益由 Volume Control Service 设置控制。如果流是立体声流，那么，一旦它被分成左分量和右分

量, Volume Offset Control Service 的单独实例就可以调整每个扬声器的增益。(该图为硬件描述。在实践中, VCS 和 VOCS 增益设置的总和应用于每个音频通道)。这些不同的组合使用, 模拟了平衡控制的效果。它们还可以单独用于调整每个扬声器的相对音量, 以适应左耳和右耳不同程度的听力损失。

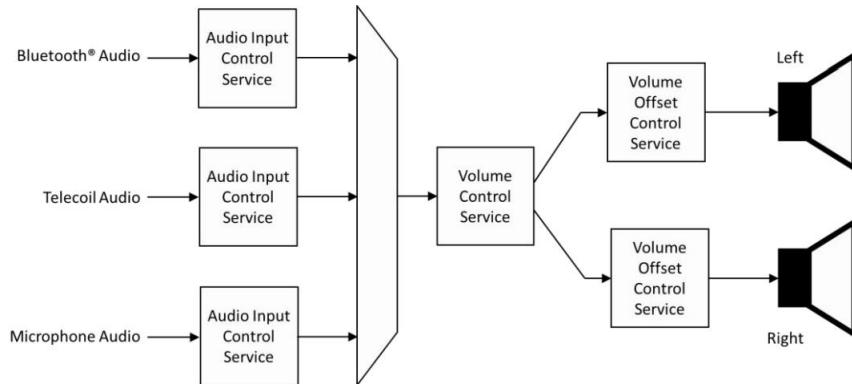


图 10.1 耳机的 Volume 和 Audio Input 服务的典型实现

### 10.1.1 Coping with multiple volume controls

从一开始, 在 Bluetooth LE Audio 开发过程中, 很明显, 用户希望从多个不同的地方控制耳塞和耳机的音量。假设耳塞上有本地控制, 音频源(通常是电话)上有音量控制, 智能手表和其他专用遥控设备中也有单独的音量控制。

要使这种级别的分布式控制运行, 你需要在呈现音频的设备上保持 primary volume control。如果你不这样做, 也让它在音频源上进行控制, 那么您可以得到一个 Controller 在 Source 级别上运行, 另一个在 Sink 级别上运行。这导致了 Source 增益被调低, 而 Sink 增益被调满, 以进行补偿的情况。这意味着用户不能再增加他们耳朵那边的音量, 音频质量也会降低, 因为 codec 正在以非常低的幅度对信号进行采样和编码。降低了非常重要的信噪比, 因为在解码和放大时, 噪声也会被放大。

为了解决这个问题, 增益发生在音频链的末端是很重要的, 但要做到这一点, 所有音量控制 Clients 都需要跟踪其值, 因此它们始终知道自己处于何处。这有点像一个老笑话, 司机停下来问路人是否可以告诉他去都柏林的方向, 路人回答说, “如果我要去都柏林, 我就不会从这里出发”。幸运的是, GATT Client-Server 模型具有通知功能, 因此我们可以利用这些通知来确保每个 Volume Control Client 都知道 Server 上的当前增益级别。但绝对可以肯

---

定的是，Volume Control Service 包括一个名为 Change\_Counter 的附加保护，我们将在下面探讨。

回顾图 10.1，它还说明了该架构，其中 Volume Control 状态尽可能接近呈现的最终点。这就带我们到了单独的 Volume Control Services。这些都是很薄的文档，所以我们可以很快地浏览它们。

## 10.2 Volume Control Service

与我们之前看到的控制服务一样，所有的工作都是由两个特征完成的：

- Volume Control Point, Clients 用于设置音量，以及
- Volume State，在 Volume Control Point 被写入之后，Server 使用它来将当前音量通知出去。

严格地说，我们应该谈论是增益，而不是音量，但由于每个人都习惯了音量的口语用法，我会坚持使用它。第三个特征——Volume Flags，用于让 Client 了解当前音量设置的历史。

Volume State 特征有三个域，都是一个字节长度，如表 10.2 所示。在 Acceptor 上，仅有一个 Volume State 特征实例。

Name	Value
Volume_Setting	0 到 255。0 = 最小值，255 = 最大值
Mute	0 = 未静音 1 = 静音
Change_Counter	0 到 255

图 10.2 VCS 的 Volume State 特征

Volume\_Setting 定义为从 0(最小音量)到 255(最大音量)的值，没有规定这些数字与输出音量的关系。在具体实现中将这些值映射到实际输出量，并期望用户感知近似线性。即，如果 Volume\_Setting 的值设置为 127，则导致的输出听起来应该是最大值的一半。这将我们带回到了关于如何感知音量的争论。我们的听力通常是对数的，这就是为什么声音强度是以分贝作为单位测量的，但这可能会受到我们所听到的声音的影响。将其留给制造商确实意味着，如果不同制造商的扬声器或耳塞一起使用，其音量的变化可能与听众的预期不一致。在某些设置下，一个声音可能会比另一个声音大。解决这一问题属于“太难”的范畴，因此只能留给实现来解决。

Volume\_Setting 域的值因对 Volume Control Point 特征(q.v.)进行操作，也就是更新其当前值进行修改而改变，或因设备用户界面上的本地操作而改变。Volume\_Setting 中存储的值不受任何 Mute 操作的影响。

Mute 指示音频流是否已静音。这与 Volume\_Setting 值无关，因此当

---

设备 unmuted 时，音频音量将恢复到之前的音量。

Change\_Counter 是也上面提到的功能，可确保所有 Volume Control Clients 都同步。实现 VCS 的 Acceptor 开机时，Server 使用 0 到 255 之间的随机值初始化 Change\_Counter。对 Volume Control Point 特征或设备的本地控制的每次后续操作，无论是更改音量还是静音，都会导致 Volume\_Setting 或 Mute 位域的更改，从而导致 Change\_Counter 将其值递增 1。一旦值达到 255，它将滚动到 0 并保持递增。

Change\_Counter 的目的是确保任何试图更改当前音量或静音的 Client 都知道它们当前的值。这是通过要求 Client 的每个写入过程都包含其持有的 Change\_Counter 的当前值来检查的。除非该值与 Volume State 特征中的值匹配，否则假定它们不同步。该命令被忽略，不会发送任何通知。在这种情况下，Client 应认为缺少通知，应读取 Volume State 特征并重试。

要更改音量或使设备静音，Volume Control Client 将使用其中一个 Volume Control Point procedures 写入 Volume Control Service 的 Volume Control Point 特征。该命令包含一个操作码、一个 Change\_Counter 值，如果是 Set Absolute Volume 命令，则包含一个 Volume\_Setting 值。表 10.1 列出了六个操作码。

Opcode	Operation	Operands
0x00	Relative Volume Down	Change_Counter
0x01	Relative Volume Up	Change_Counter
0x02	Unmute / Relative Volume Down	Change_Counter
0x03	Unmute / Relative Volume Up	Change_Counter
0x04	Set Absolute Volume	Change_Counter, Volume_Setting
0x05	Unmute	Change_Counter
0x06	Mute	Change_Counter
0x07-0xFF	Reserved for Future Use	

表 10.1 Volume Control Point 特征操作码

其中大多数都是不言自明的。Relative Volume Down 和 UP 可更改音量设置，而不影响 Mute 状态，因此可用于更改音量，会在设备最终取消静音时生效。操作码 0x05 和 0x06 在不影响音量级别的情况下取消静音和静音，而操作码 0x02 和 0x03 在静音或取消静音的同时改变音量相对大小值。Set Absolute Volume 需要一个附加参数，即所需的绝对音量级别。

尽管所有设备的 Volume\_Setting 范围从 0 到 255，但很少有设备可能包含超过 21 个离散音量级别。这需要 Server 定义 Step Size，每当发出音量设置命令时都会应用该步长。Step Size 实际上是  $256 \div$  步数。Server 使用以下公式计算写入 Volume State 特征的 Volume\_Setting 的新值。

$$\text{Relative Volume Down: } \text{Volume\_Setting} = \text{Max}(\text{Volume\_Setting} - \text{Step Size}, 0)$$

---

Relative Volume Up:  $\text{Volume\_Setting} = \text{Min}(\text{Volume\_Setting} + \text{Step Size}, 255)$

### 10.2.1 Persisting volume

最后一个特征是 Volume Flags，它是一个位域，只定义了一个 bit。Bit 0，就是 Volume\_Setting\_Persisted 域。如果设置为 0，则指示 Client 通过向 Volume Control Point 特征写入 0x04 操作码来重置音量。这将设置一个绝对音量，它可能是由应用程序或 Client 设备确定的默认值。它也可能是上次使用应用程序时记住的值。

如果值为 1，则表示 Volume Control Server 仍具有上次会话的 Volume\_Setting，应继续用于此会话。Volume Client 应通过通知或读取该值来恢复该值，并将其用作此会话的初始值。通过使用以前使用的相同音量设置可以改善用户体验。

## 10.3 Volume Offset Control Service

如果你只有一个单声道扬声器，那么 Volume Control Service 就是你所需要的。一旦处理多个音频流，无论是发送给单个 Acceptor 还是多个 Acceptor，都需要为每个给出的 Audio Location 包含一个 Volume Offset Control Service 的实例。所用特征可以通过使用 Volume Control Profile 中的过程进行访问。

Volume Offset Control Service 包含四个特征，如表 10.2 所示。四个都是强制性的。

Characteristic	Mandatory Properties	Optional Properties
Volume Offset State	Read, Notify	None
Volume Offset Control Point	Write	None
Audio Location	Read, Notify	Write without response
Audio Output Description	Read, Notify	Write without response

表 10.2 Volume Offset Control 特征

Volume Offset State 和 Volume Offset Control 以正常方式工作。Volume Offset State 包括两个域——Volume\_Offset 和 Change\_Counter。

Filed	Size
Volume_Offset	2 字节
Change_Counter	1 字节

表 10.3 Volume Offset State 特征

Volume\_Offset 长度为两个字节，因为它允许的值从 -255 到 255。Volume\_Offset 的值与 Volume\_State 的值相加，以提供呈现的总音量。

Change\_Counter 与我们在 Volume Control Service 中看到的概念相同，

---

用于确保发出 Volume Offset 命令的任何 Client 都知道 Volume Offset 的当前状态。请注意，虽然它是相同的概念和名称，但它是一个单独的实例。VOCS 的每个实例以及 VCS 的单个实例都保持其自己的 Change\_Counter 值，当其 Volume\_Offset 或 Volume\_State 值发生任何更改时，将该值更新。

为了实现更改，Client 使用以下参数写入 Volume Offset Control Point 特征：

Parameter	Size (字节)	Value
Opcode	1	0x01 = Set Volume Offset
Change_Counter	1	0 到 255
Volume_Offset	2	-255 到 255

表 10.4 Set Volume Offset 参数(opcode = 0x01)

### 10.3.1 Audio Location characteristics

Volume Offset Control 服务包含两个有关 Audio Location(offset 会用到的)的特征。

第一个是 Audio Location 特征。这是一个 4 字节的 bitmap，它使用 Generic Audio Assigned Numbers 中的 Audio Locations 格式定义了 Offset 应该应用于哪个 Audio Location。通常，包含单个位置，因为每个 Audio Location 都应用了单独的 VOCS 实例。这通常是 read-only 特征，在制造时设置，但也可以使其 writable。

Audio Output Description 特征允许将文本字符串分配给每个 Audio Location，以便为应用程序提供更多信息，例如 Bedroom Left Speaker。它可以在制造时分配，也可以让用户访问以分配友好名称。

## 10.4 Audio Input Control Service

呈现三个服务的最后一部分是 Audio Input Control Service。这涉及许多产品，如助听器、耳塞和耳机，包含不止一个音频流源，并且可能同时使用多个。最常见的例子是同时使用外部麦克风和 Bluetooth stream。对于助听器，这一直是它们的工作方式。最近，随着透明传输模式的出现，它已经成为耳塞和耳机中越来越流行的功能，这样佩戴者就可以了解周围世界的声音。

Audio Input Control Service 的实例通常被包含在打算在该设备上呈现的每个单独的音频路径中。如果只有一条 Bluetooth LE Audio 路径，它与 VCS 相比没有优势，因此不需要包含在内。不直接反馈到输出的音频输入，例如为提供输入音频噪声消除的麦克风，不具有它们自己的实例，因为它们会被用于处理另一音频流(其可能具有自己的 AICS 实例)。

---

Audio Input Control Service 包含 6 个特征，在表 10.5 列出。它们全部都是强制性的特征。

Characteristic	Mandatory Properties	Optional Properties
Audio Input State	Read, Notify	None
Audio Input Control Point	Write	None
Audio Input Type	Read	None
Audio Input Status	Read, Notify	None
Audio Input Description	Read, Notify	Write without response
Gain Setting Properties	Read	None

表 10.5 Audio Input Control Service 特征

在跳到这些特征之前，我们需要了解 AICS 是如何定义 Gain 的。对于音量，VCS 和 VOCS 只需定义一个从 0 到 255 的线性标尺，就像 Hi-Fi 单元上的旋钮，从 0 到 10，并由制造商将其转换为通常基于分贝的内部映射。在 AICS 中，假设增益将基于分贝，因此 Gain 的数字以分贝作为单位。然而，为了提供更大的灵活性，增益中的实际步长可以定义为 0.1dB 的倍数。这些倍数是制造商特定的选择，在 Gain Setting Properties 特征中显示，以及最小和最大允许值，如表 10.6 所示。

Name	Size(字节)	Description
Gain_Setting_Units	1	增量，以 0.1db 为补偿，应用于所有 Gain_Setting
Gain_Setting_Minimum	1	Gain_Setting 的最小允许值
Gain_Setting_Maximum	1	Gain_Setting 的最大允许值

表 10.6 Gain Setting Properties 特征的域(只读)

在进一步的复杂性中，AICS 允许音频流自动(传统上称为 Automatic Gain Control 或 AGC)或手动控制其 Gain，这可以由 Volume Control Client 或本地用户控制，改变的值暴露在 Audio Input State 特征中。控制为自动时，Server 不支持 Client 所做的任何更改。Server 可以使用表 10.7 中所示的值，通过 Audio Input State 特征的 Gain\_Mode 公布是否允许 Client 将其从 Automatic 更改为 Manual，反之亦然。

Gain_Mode	Value	Description
Manual Only	0	Client 不可以更改这些设置
Automatic Only	1	
Manual	2	手动，但是 Client 可以将 Gain_Mode 更改成 Automatic
Automatic	3	自动，但是 Client 可以将 Gain_Mode 更改成 Manual

表 10.7 AICS Audio Input State 特征中的 Gain\_Mode 域的含义

解决了这些问题后，我们就可以转到 Audio Input State 和 Audio Input Control Point，它们的工作方式与我们预期的一样。Audio Input State 包含四个位域，如表 10.8 所示。

Name	Size (字节)
Gain_Setting	1

---

Mute	
Gain_Mode	
Change_Counter	

表 10.8 Audio Input State 特征的位域

Gain\_Setting 按照 Gain\_Setting\_Units，公布当前的 Gain 值。将 0 写入 Mute 不会影响当前 Gain\_Setting 值，因此通过将 1 写入 Mute 来取消静音将使增益返回到 Gain\_Setting 中的上一个值。如果 Server 处于 Automatic Gain Mode，它将忽略写入 Gain\_Setting 域的任何内容。

Mute 表示音频流是静音(value=0)还是取消静音(value=1)。AICS 为我们提供了一个附加选项，即，值 2 表示禁用 Mute 功能。最终 Audio Stream 仍然可以使用 VCS Mute 进行静音，但通过使用此选项，单个输入流无法单独静音。

Gain\_Mode 和 Change\_Counter 的行为与 VCS 和 VOCS 的行为完全相同。再次强调，每个 AICS 实例都有对其的独立实例化。

通过使用表 10.9 中列出的五个操作码之一写入 Audio Input State Control Point 特征来设置 AICS 参数。

Opcode	Operation	Description
0x01	Set Gain Setting	以 Gain_Setting_Units x 0.1dB 的增量设置增益
0x02	Unmute	取消静音（除非禁用静音）
0x03	Mute	静音（除非禁用静音）
0x04	Set Manual Gain Mode	从自动增益更改为手动增益（如果允许）
0x05	Set Automatic Gain Mode	从手动到自动增益的更改（如果允许）

表 10.9 Audio Input Control Point 特征的操作码

这些都与它在名称中所说的差不多，尽管正如我们已经发现的，Server 可以忽略很多附加说明，特别是如果它不准备放弃对 Gain 和 Mute 功能的控制。

Set Gain Setting 操作码(0x01)的形式如表 10.10 所示。

Parameter	Size (字节)	Value
Opcode	1	0x01
Change_Counter	1	0 到 255
Gain_Setting	2	-128 到 127

表 10.10 Audio Input Control Point 特征 Set Gain Setting 过程的格式

所有其他过程的参数使用相同的较短格式，如表 10.11 所示。

Parameter	Size (字节)	Value
Opcode	1	0x02 = 取消静音 0x03 = 静音 0x04 = Set Manual Gain Mode 0x05 = Set Automatic Gain Mode
Change_Counter	1	0 到 255

表 10.11 Audio Input Control Point 特征 0x02-0x05 操作码格式

---

与 VOCS 一样，还有一些其他特性对于用户使用很有帮助。

Audio Input Type 使用只读文本描述来标识音频输入流，该文本描述可用于 UI。典型值(英语)是 Microphone、HDMI、Bluetooth 等。这些值由制造商设置。

Audio Input Status 将每个 AICS 音频流的当前状态暴露出来。如果值为 0，音频流是 Inactive 的，如果为 1，音频流是 Active 的。

Audio Input Description 允许对于音频流的更详细描述，对于相同类型的多个输入是很有用的，比如，多 Bluetooth 或 HDMI 输入。这是由制造商设置的，或可将其设置为可写入，允许用户去更新。

## 10.5 Putting the volume controls together

开场图显示了这三种服务是如何协同工作的。图 10.3 显示了一对立体声耳机的典型实现，它支持 Bluetooth 连接和本地麦克风。

通过将 VCS 中的单个值与每个耳塞的 VOCS 实例化的 Audio Location 特定值相结合来设置每个耳朵的音量。任何一个输入 Audio Stream 都可以单独静音或控制其增益(如果 Server 允许)，而 VCS 提供全局静音功能。

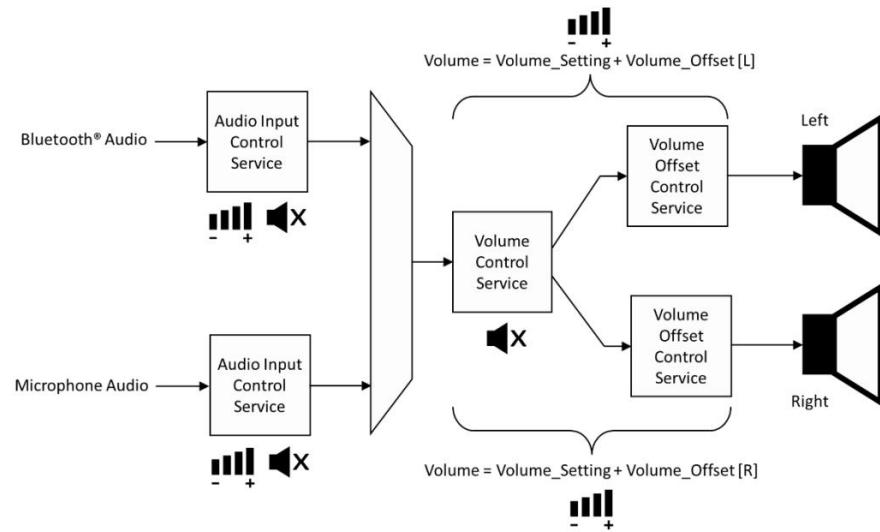


图 10.3 一副耳机的典型实现

图 10.4 展示了立体声对的左助听器的类似方法。因为只有一个通道被呈现，所以 VOCS 只有一个实例，服务于左助听器。右耳的另一个助听器将是相同的，但具有用于 Right Audio Location 的 VOCS 实例。

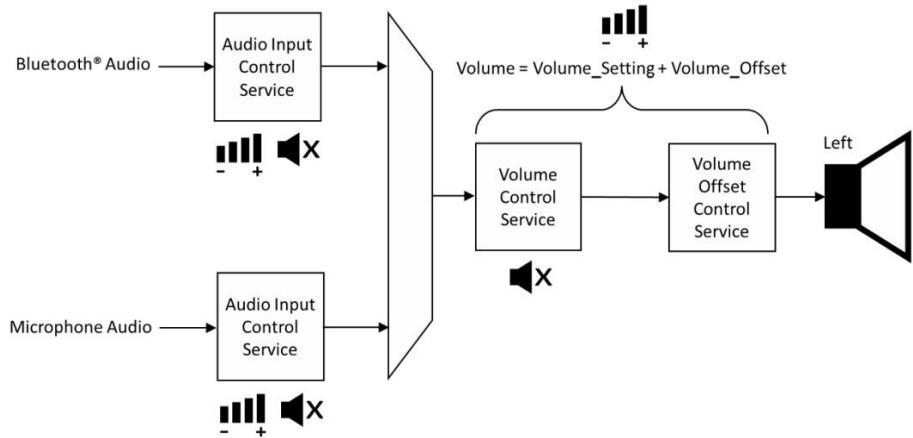


图 10.4 来自于立体声对的左耳助听器典型实现

## 10.6 Microphone control

尽管上面的示例包括麦克风，但它们都将麦克风用作本地输入，本地输入被选择或与要呈现的其他音频输入混合。然而，麦克风也用作音频输入，这些音频输入被捕获并发送回 Initiator。Microphone Control Service 和 Profile (MICS 和 MICP)旨在提供 Microphones 设备范围的控制方式。

Microphone Control Service 可能是所有 Bluetooth LE Audio 规范中最简单的服务，包括单个特征，为麦克风提供设备范围的静音。其最简单的用法如图 10.5 所示。

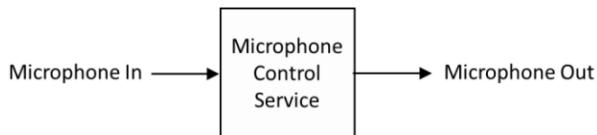


图 10.5 Microphone Control Service 的简单实用

没有相应的 Control Point 特征。相反，Mute 特征可以直接写入，也可以读取和通知。它具有与 AICS Audio State 特征中 Mute field 相同的功能，即：

Description	Value
Mot Muted	0x00
Muted	0x01
Muted Disabled	0x02

图 10.6 MICS Mute 特征值

如果需要控制麦克风增益，MICS 应与 AICS 实例相结合(图 10.7)，尽管在这种情况下，MICS 与仅使用 AICS 相比没有太大优势。然而，大多数

---

Clients 希望使用 MICS 执行麦克风静音，因此这是保留 MICS 的原因。

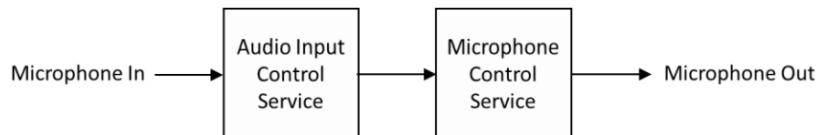


图 10.7 Microphone 服务和 Audio Input Control 服务的结合

当存在多个麦克风时，AICS 和 MICS 的组合更有意义，因为 MICS 为麦克风提供了设备范围的静音功能，这是其真正的用途(图 10.8)。

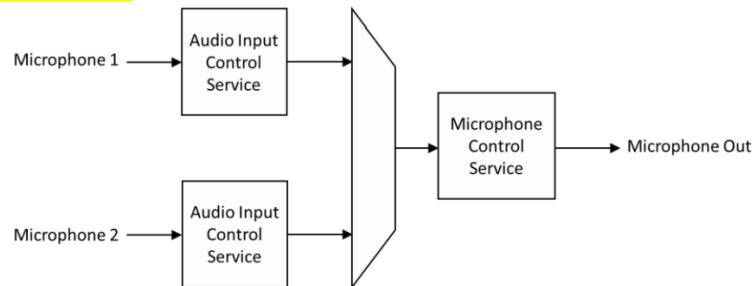


图 10.8 多麦克风的 MICS 和 AICS 使用

最后，MICS 可用于为多个麦克风提供设备范围的静音，作为音量控制策略的一部分，如图 10.9 所示。然而，在大多数情况下，Acceptor 中的多个麦克风将直接输入音频处理模块，因此不太可能需要对单个麦克风进行外部控制。图 10.9 展示了 Bluetooth LE Audio 中音量和麦克风控制服务的灵活性。

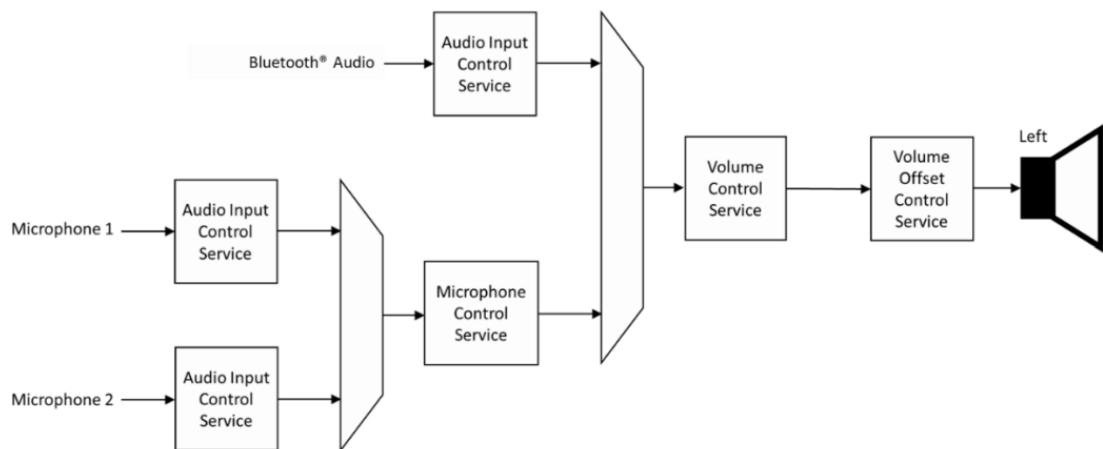


图 10.9 Microphone 和 Volume Control 的结合使用

## 10.7 A codicil on terminology

在本书中，我一直使用术语 Initiator、Acceptor 和 Commander 来描述

---

Bluetooth LE Audio 生态系统中的三种主要设备。正如我在第 3 章中所述，这些术语被定义为 CAP 中的角色，因此纯粹主义者可能会反对我将它们与设备混为一谈。我仍然觉得，合并会使人们更清楚地理解一切是如何结合在一起的。

Commander (作为一种设备)与 Initiators 和 Acceptors 的交互方式可能存在混淆。作为一个角色，Commander 可以与 Initiator 搭配，但作为一个设备，其与 Initiator 及其每个 Acceptors 的交互可能会令人困惑。尽管 CAP 过程涵盖了所有这些互动，但它们是独立的。图 10.10 以 Initiator、Acceptor 和独立 Commander 的简单案例说明了本章中描述的一些 profile 和 service。

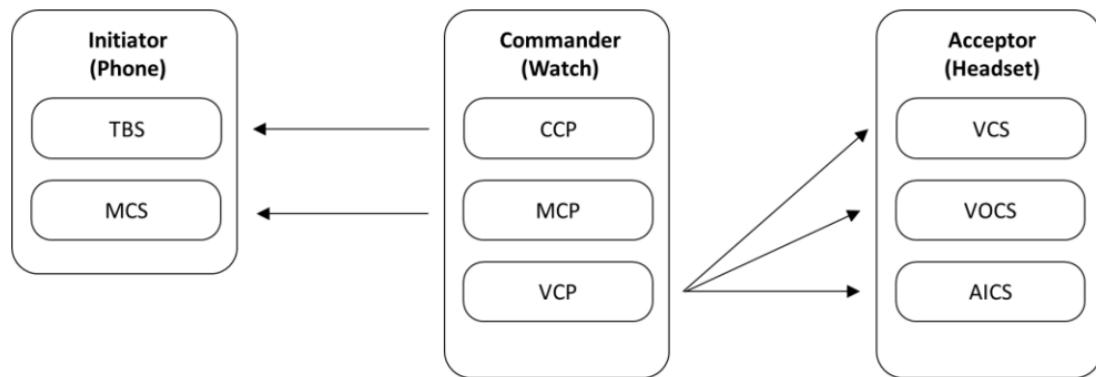


图 10.10 对于 volume 和 control 的 Profile 和 Server 的示例

在这种情况下，在 Commander 中实现了三个 profiles。其中两个——Call Control 和 Media Control 对 Initiator 中的补充服务起作用，而 Volume Control 对 Acceptor 中的 VCS、VOCS 和 AICS 实例起作用。对于语法倾向，Commander 中的 profile 和 Initiator 中的服务被放置在相同位置；Acceptor 中的服务一起运行。

现在我们到了结束 GAF 规范的时候了。Bluetooth LE Audio 中唯一的其他规范是顶级配置 profile，我们即将介绍这些 profile。

---

## 第11章 Top level Bluetooth LE Audio profiles

在介绍了 Generic Audio Framework 中的所有内容之后，我们现在来了解 Bluetooth LE Audio 的顶层 profile。尽管它们仍然被称为 profile，但在大多数情况下，它们比我们讨论的下层的 profile 或 Bluetooth Classic Audio profile 简单得多。与其定义过程，他们通常将自己局限于配置，指定新的角色来强制组合可选功能，并添加超出 BAP 的 QoS 要求。在这样做的过程中，它们通过定义功能组合来满足常见的用例，从而提高了标准。

在本章中，我们将了解这些 profile 包含的内容。由于它们依赖于 GAF 规范中已经定义的特性，因此它们没有补充的服务规范。HAP——Hearing Access Profile 是例外，因为相应的 Hearing Access Service 为 Hearing Aids 引入了新的 Presets 特征。TMAP——Telephony Media 和 Audio Profile 具有标称 TMAS 服务，该服务包含在 TMAP 规范中。Public Broadcast Profile 没有服务。这是广播应用程序固有的一种异常现象，该应用程序不期望 Initiator 和 Acceptor 之间存在连接。缺少连接意味着没有 Client-Server 关系的可能性，因此没有机会实现 Service 规范。

Bluetooth 工作组目前正在开发许多顶级 profile，但计划采用的前三个 profile 是：

- HAP 和 HAS——Hearing Access Profile 和 Service，定义了助听器生态系统中使用的产品的要求。
- TMAP——Telephony 和 Media Audio Profile。旨在涵盖 classic HFP 和 A2DP profile 的主要功能，增加 Bluetooth LE Audio 拓扑带来的新功能。
- PBP——Public Broadcast Profile，这是 Bluetooth LE Audio Sharing 用例的基础(参见第 12 章)。它标识任何 Bluetooth LE Audio Acceptor 都可以接收广播音频流。

这些文件以草稿形式公开提供，但仍可能更改。本章内容反映了编写时的状态，详见第 13.2.2 节。

由于整个 Bluetooth LE Audio 开发是由助听器行业开始的，因此从 HAP 和 HAS 开始似乎是合理的。

### 11.1 HAPS the Hearing Access Profile and Service

Hearing Access Profile 和 Service 旨在满足 Hearing Aid 生态系统中使用

---

的设备的要求，其中包括助听器、为助听器提供音频流的产品以及用于控制助听器的附件。

Hearing Aids 与耳塞和其他 Acceptors 不同，因为它们始终处于开启状态，捕捉和处理环境声音以帮助佩戴者。这意味着驱动 Hearing Access profile 的所有用例都将 Bluetooth LE Audio 为环境音频流的附加音频流。这使得它们与众不同，因为 Bluetooth 连通性并不是购买它们的实际上的原因。它们的不同之处还在于，佩戴它们的每个人都有听力损失，因此它们在音频质量(以及 QoS 设置)和电池寿命之间的要求有不同的平衡。白天把助听器拿出来给它充电是比佩戴耳塞更值得注意的行为，因为在这段时间你可能听不到。提高音频质量会增加功耗，因此 Hearing Access profile 不会在 BAP 规定的设置上增加额外的 QoS 要求，使用 16\_2 LC3 codec 设置用于语音(16kHz 采样率, 7kHz 带宽)和 24\_2 codec 设置(24kHz 采样率、11kHz 带宽)。由于许多助听器不遮挡耳朵，除了 Bluetooth 流之外，还可以听到周围的声音。因此，助听器通常倾向于使用 Low Latency QoS 设置，以避免环境声音和传输声音之间的回声。

HAP 规范描述了公认为助听器的产品的物理设备配置。该 profile 定义了四种不同的 Acceptor 配置，所有这些都包含在文件中的“助听器”一词中。这些是：

- 单个助听器，呈现单个 Bluetooth LE Audio Stream，
- 单独的助听器，接收独立的左右音频流，并将解码的音频合并到一个音频通道，
- 一对助听器(也称为 Binaural Hearing Aid Set)是一个 Coordinated Set 的成员，支持每只耳朵的单独左右音频通道，以及
- 使用单个 Bluetooth 链接的助听器，但为每只耳朵提供单独的左右音频输出。这些被称为 Banded Hearing Aids，在每个耳朵的助听器设备和 Bluetooth 收发器之间都有有线连接，Bluetooth 收发器通常位于颈部或头戴式耳机中。

Hearing Access Profile 定义了助听器生态系统的四个不同角色：

- HA (Hearing Aid)，上述四种类型的助听器中的一种。
- HAUC (Hearing Aid Unicast Client)，它是可以与助听器建立 unicast Audio Streams 的 Initiator。unicast Audio Streams 可以是单向或双向的。
- HABS (Hearing Aid Broadcast Sender)，其是传输广播音频流的发起

---

方，以及

- HARC (Hearing Aid Remote Controller)，这是一种控制音量水平、  
mute 状态和助听器预设的设备(稍后我们将介绍)。

Hearing Access Profile 的大部分列出了不同产品迭代所需的 BAP 和 CAP 功能的强制性组合。虽然许多产品是显而易见的，例如将一对助听器的 CSIS Size 特征设置为 2，但它们确保任何声称支持该 profile 的产品都将包含相同的功能并可互操作。这本质上是 Bluetooth LE Audio 顶级 profile 的要点——通过明确指定必须提供哪些底层 profile、服务和功能，确保特定用例的互操作性。

这是有局限性的。每个助听器必须能够接收符合 BAP 强制要求的 Unicast 或 Broadcast Audio Stream，但不需要接收已使用其他可选 QoS 设置编码的音频流。它必须接受来自任何支持 HARC 角色的设备的音量控制命令，该设备可以是独立的遥控器或手机上的应用程序。这意味着助听器配件将首次与任何品牌的助听器实现全球互操作。

所有助听器都需要支持 Ringtone、Conversational 和 Media Context Types，这意味着它们都支持来电。然而，他们不需要支持 Call Control 或在电话通话时返回语音流。这是一个实践的决定，因为许多助听器将麦克风放在可以获得最佳使用者周围声音拾取效果的位置，而不是拾取佩戴者的声音，所以用户在通话时使用手机麦克风通常会更好。这些是制造商在产品营销中需要传达的限制。

HAP 和 HAS 引入了一个新概念，即支持 Presets。Presets 是专用的音频处理配置，助听器制造商将其用于优化输入耳朵的声音。通常，他们会调整处理以适应不同的环境，例如餐厅、商店、办公室、家庭等。HAP 和 HAS 不会试图标准化这些设置，而是提供一个编号方案，制造商可以将其映射到特定的实施方案。然后，用户可以通过其编号选择特定的预设，或在它们之间循环。它包括添加 Friendly Names 的功能，以便应用程序可以显示当前预设和其他可用预设。它还支持动态预设，其中预设的可用性可能会根据助听器的状态而改变。例如，当没有可用的感应线圈时，为接收感应线圈信号而优化的预设将不可用。预设目前是助听器特有的功能。有关它们的更多信息，请参阅 HAPS 规范。

在 HAS 的预设功能中，有一个有趣的选项，可能会扩展到其他 profile 中。这是一种使用 non-Bluetooth radio 将命令从一个助听器传递到另一个的能力，而无需 Commander 根据其中一个成员的通知向其他成员制定程序。

---

这是 Synchronized Locally 操作，它通知 Commander，发送给一个助听器的任何预设命令都可以本地转发给该组的其他成员，这在 HAS 第 3.2.2.8 - 3.2.2.8 节中有所描述。

HAP 中的另一个增强功能是包括 Immediate Alert Service [HAP3.5.3]。这允许不支持音频流的设备向 Hearing Aid 提供警报，并将其作为警报呈现给用户。没有定义具体的使用情况，但建议制造商可能希望在门铃或微波炉等产品中包含此功能，以帮助助听器佩戴者了解他们需要响应的事项。

HAP 要求的目的是支持助听器设想的所有用例，如图 11.1 所示。

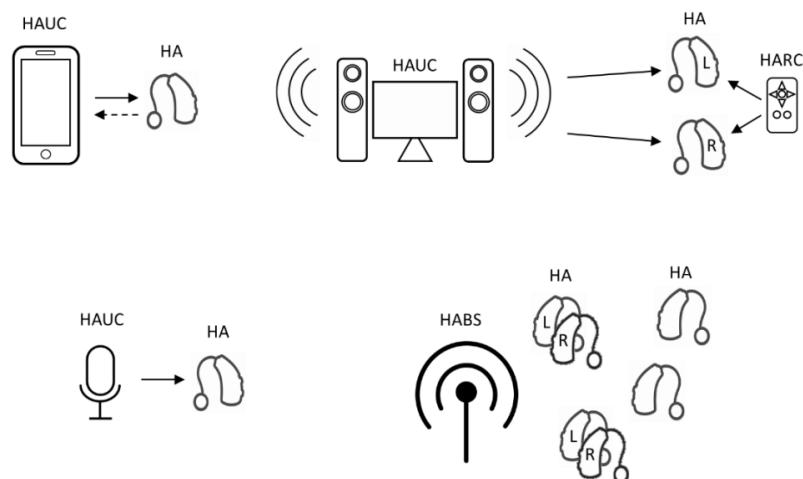


图 11.1 四个 HAP 的主要用例，各自展示了角色

由于许多助听器使用情况将涉及用户听到环境声音以及接收到的 Bluetooth LE Audio Stream，预计设备将倾向于使用 Low Latency QoS 模式。为了帮助确保低延迟，要求 HA 角色中的所有设备都应支持 20ms 的 Presentation Delay 值，用于 Low Latency QoS 模式的接收和发送。

Hearing Access Profile 中的最后一个细微差别是第 4.1 节中的一组要求，当 HAUC 使用 7.5ms Isochronous Interval 且助听器不支持 7.5ms LC3 codec 帧的接收时，该要求规定了 Link Layer 设置。预计在助听器具有最小 LC3 实现(因为不强制支持 7.5ms codec frame)的情况下，以及由于其他 peripheral devices 的定时限制，Initiator 被迫使用 7.5ms 间隔(不能适应优选的 10ms 间隔)，这将是一种边缘情况。在这种情况下，scheduler 应确保为 ISOAL 选择指定的参数，以避免 SDU 的分割和 frame loss rate 的潜在增加。预计随着 Bluetooth LE Audio 的普及，Initiators 将转移到最佳的 10ms 传输间隔，因此这是一个短期解决方案。

所有 HAU、HA 和 HAB 必须支持 2M PHY。虽然未强制使用，但强烈

---

建议您节省 airtime，从而延长电池寿命。

## 11.2 TMAP – The Telephony and Media Audio Profile

与 HAP 一样，TMAP profile 主要是 BAP 和 CAP 中规定的额外需求列表，以模拟 HFP 和 A2DP profile 的用例。TMAP 没有引入任何新的行为或特征，因此在 TMAP 中加入了最小的 TMAS 部分。

由于 TMAP 将电话和媒体应用程序绑定到一个文档中，因此它感觉像是一个组合 profile，实现者可以广泛地选择他们支持的内容。这可能会导致一些奇怪的事情，因为它的许多功能是可选的。理论上，根据您的选择，可以制作支持电话呼叫但不支持音乐的 TMAP 兼容设备，反之亦然。这是将这么多不同的用例捆绑在一起的逻辑结果。音箱或扬声器不支持电话是有道理的，但这意味着耳机也不需要支持电话。相反，由实施者来选择合适的功能和角色，让市场 Darwinism 来处理任何不利的选择。

为了避免任何意外，TMAP 的 TMAS 元素包括 TMAP Role 特征[TMAP 4.7]，它公布设备支持的特定角色。这允许 Acceptors 确定 Initiator 支持的 Roles，这在某些用例中是必需的。这就引出了 Roles。TMAP 没有为 Commander 定义任何新的 Roles，但为 Initiator 和 Acceptor 定义了三对 Roles。

### 11.2.1 Telephony Roles – Call Gateway and Call Terminal

对于电话，TMAP 定义了 Call Gateway (CG) 和 Call Terminal (CT) 角色。Call Gateway 是连接到电话网络(如电话、平板电脑或 PC)的设备，是 Initiator。Call Terminal 通常是耳机，但也可以是扩展扬声器或会议电话的麦克风。一些常见配置如图 11.2 所示

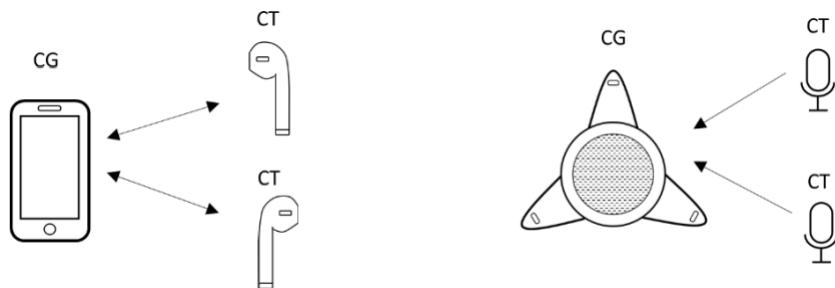


图 11.1 CG 和 CT 角色的典型配置

支持 CG 和 CT 角色的设备必须支持更高的 codec 设置，即在 7.5ms 和 10ms frame rates (BAP 中的 32\_1 和 32\_2) 下进行 32 kHz 采样，并具有 Low Latency 设置。这些对应于 3GPP 为 5G 手机指定的与 Superwideband EVS 语音 codec 相同的音频质量，确保从本地麦克风到远端耳机的质量没有损失，

---

反之亦然。

CG 必须支持 CPP Server 角色，但不要去 CCP 中规定的任何其它功能。

### 11.2.2 Media Player Roles – Unicast Media Sender and Unicast Media Receiver

Media Player 用例使用 Unicast Media Sender (UMS) 和 Unicast Media Receiver (UMR) Roles。Unicast Media Sender 是 Initiator，它是音频源，Unicast Media Receiver 是 Acceptor。典型配置如图 11.3 所示。

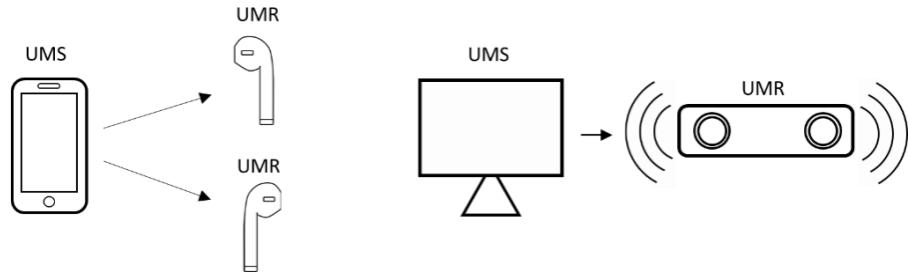


图 11.3 典型的 TMAP Unicast Media 应用

对于 Unicast Media，TMAP 提高了音频质量，要求 Unicast Media Receiver 支持所有六种 48kHz 采样 codec 器配置，从 48\_1 到 48\_6。Unicast Media Sources 必须支持 48\_2 codec 设置和至少一个 48\_4 或 48\_6 设置。所有 TMAP 角色都必须支持 2M PHY，这几乎肯定是为这些配置找到足够的 airtime 所必需的。尽管需要支持这些 QoS 配置，但应由应用程序决定如何配置 ASE。它可以决定使用较低的值。对于 UMS，32 kHz 采样率仍然是可选的，尽管 Acceptor 不太可能不支持它们。用户不太可能听到 32 和 48kHz 之间的差别，因此设置的选择很大程度上取决于营销决策。

TMAP 不要求支持 Unspecified 以外的任何 Context Type。如果 Unicast Media Receiver 想要拒绝从 Call Gateway 建立流的任何尝试，它应该支持 Ringtone，但将其设置为不可用。

TMAP 通过强制支持 Play 和 Pause 操作码，增加了对媒体控制的要求。UMS 还必须支持 Media Control Point 特征。

### 11.2.3 Broadcast Media Roles – Broadcast Media Sender and Broadcast Media Receiver

TMAP 中最后两个角色是 Broadcast Media Sender 和 Broadcast Media Receiver。不出所料，它们是为广播应用设计的。在 TMAP 中，它们通常被设想为个人应用，需要更高的音频质量，但也可以用于公共广播应用，例如

---

电影院。典型用例如图 11.4 所示。

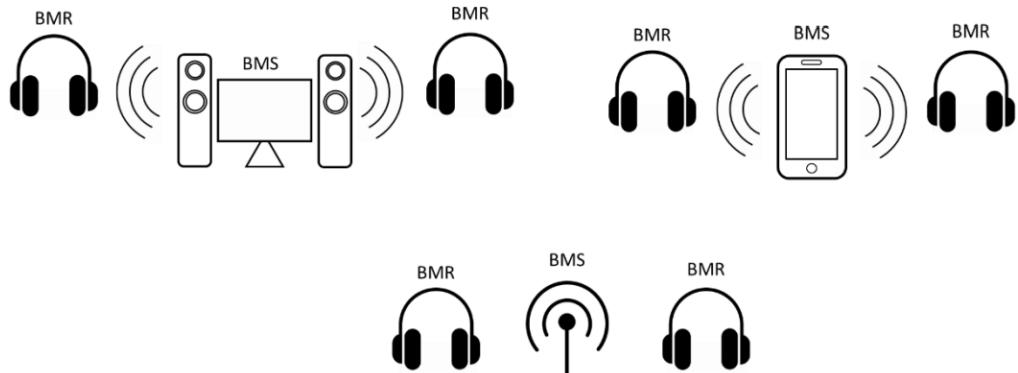


图 11.4 Broadcast Media Sender 和 Receiver 角色的典型的用例

TMAP 为广播角色的 BAP 和 CAP 添加了很少的要求。它要求支持更高质量的 QoS 设置，要求支持 BAP 表 6.1 中定义的 Broadcast Media Receiver 的所有 Low Latency 和 High Reliability 48kHz QoS 模式(48\_1\_1 至 48\_6\_1 和 48\_1\_2 至 48\_6\_2)，以及 Broadcast Media Sender 的 48\_1 和 48\_2 QoS 配置。它还要求 Broadcast Media Sender 必须至少支持 48\_3 或 48\_5(7.5ms 帧) codec 配置之一和 48\_4 或 48\_6(10ms 帧)codec 配置之一。

TMAP 要求 Broadcast Media Receivers 在其 Low Latency 和 High Reliability QoS 模式的 Presentation Delay 范围内支持 20ms 的 Presentation Delay 值。

TMAP 还增加了对广播音频配置的强制支持，要求 Broadcast Receiver 可以接受 BAP 表 4.24 中定义的任何广播音频配置。如表 11.1 所示。BMS 要求与 BAP 保持不变。

Audio Configuration	Stream Direction (Initiator – Acceptor)	BMS	BMR
12	->	M	M
13	->--> ->-->	M	M
14	->-->>	O	M

图 11.4 Broadcast Media Roles 的 Audio Configuration 需求

---

### 11.3 Public Broadcast Profile

Public Broadcast Profile (PBP)是一个简单但非常有趣的 profile。它旨在支持 Audio Sharing 用例，保证 broadcast Audio Stream 被配置为任何 Acceptor 都可以接收。它包含一个新的 UUID——Public Broadcast Service UUID，该 UUID 作为附加 Service Type 与 Basic Audio Announcement 一起添加。这意味着它出现在 Extended Advertisement 中，允许设备读取它，而无需与 Periodic Advertising 序列同步，然后接收并解析 BASE。减少了 scanner 的工作量，降低了其功耗。本质上，它充当一个过滤器，Broadcast Sink 或 Broadcast Assistant 可以使用它来限制其扫描，从而减少识别 Broadcast Sink 可以解码的 Broadcast Sources 所需的功率和时间。

PBP 定义了三个角色——Public Broadcast Source (PBS)、Public Broadcast Sink (PBK) 和 Public Broadcast Assistant (PBA)。PBK 和 PBA 角色的唯一要求是他们能够识别和解释 Extended Advertisement 中的 PBP UUID。

如果相关 BIG 包含的 BIS 中至少一个已使用 BAP 中为 Broadcast Sink 定义的强制性 QoS 设置之一(即 16\_2\_1、16\_2\_2、24\_2\_1 或 24\_2\_2)进行编码，则 Public Broadcast Source 只能在其 Basic Audio Announcement 中包含 PBP UUID。BIG 可能包含以其他 QoS 设置编码的 BIS，但只有当具有这些强制设置的 BIS 激活时，才能发送 PBP UUID。

PBP 的原因是提醒 Bluetooth LE Audio Sinks 注意可以普遍接收的广播音频流。

以上就是我们对 Bluetooth LE Audio 规范的介绍。最后一章将介绍它们支持的一些新应用程序。

---

## 第12章 Bluetooth LE Audio applications

开发 Bluetooth LE Audio 的全部目的是支持新的音频应用程序，而不仅仅是为现有的 Bluetooth Classic Audio profile 提供功率稍低的替代方案。部分原因是需要赶上专有扩展，特别是 True Wireless Stereo。更难能可贵的是允许音频方面的创新，紧跟 TWS 和语音助手所产生的契机，使音频更具普遍性，并使我们能够更灵活地聆听音频。

Broadcast 是感应线圈的产物。感应线圈系统已经存在了很长时间。它运行良好，但非常基础。它是单声道的，音频带宽非常有限，只有当你位于感应环路的范围内时，你才能听到它。虽然 Bluetooth 的目标是提供一个更强大的继任者，但很快就会变得明显，我们可以显著感应线圈的用户体验。

复制感应线圈体验的一个首要问题是，Bluetooth 传输不受安装区域的限制。由于是无线的，它可以穿透墙壁，这意味着相邻房间和空间的人也可以听到任何广播音频。在许多情况下，例如公共大厅和礼拜场所，这不是一个重大问题，因为广播音频的唯一来源将是与特定场所相关的音频。然而，在其他情况下，如酒店房间内的电视，或具有多个会议室的会议中心内的系统，这将成为一个主要问题，因为广播将重叠，结果，人们将很难理解他们想要连接的广播是哪一个，并可能听到他们不应该听到的东西。

这导致在广播流中实现加密，因此只有具有正确 Broadcast\_Code 以解码该流的用户才能收听该流。虽然广播流重叠，但 Broadcast\_Code 提供了一种访问机制，只有授权的听众才能解码特定的广播音频流。这类似于现在的 Wi-Fi 工作方式，用户被赋予一个 SSID 名称来识别特定的 Wi-Fi 网络，然后需要输入一个 Access Code 才能连接到该网络。虽然人们已经习惯了使用 Wi-Fi，但这是一种非常基本且经常令人沮丧的用户体验。每个人都希望 Bluetooth LE Audio 做得更好。

要解决这一问题，用户需要一个更好的机制来访问代码，而不是咖啡厅桌子上的一张纸或墙上的一张通知。它导致了 Broadcast Assistant 和 Commander 的发展，提供了从信任或已知的 Broadcaster 发送信息到听众的方式。随着规范的发展，很明显，这为用户提供了一种非常强大的机制，既可以接收 Broadcast Codes，也可以以比以前的 Bluetooth 连接更灵活的方式访问单个广播。

---

## 12.1 Changing the way we acquire and consume audio

改变我们获取和使用音频的方式是开发人员理解的重要点。在过去二十年的大部分时间里，个人音频的发展都是由手机驱动的。最初，我们将从音乐共享网络获取的文件存储在手机上。最近，随着音频流服务的增长，手机成为了按需向我们的耳朵提供音乐的路由器。然而，这种体验很大程度上取决于与手机的互动，以选择我们想要听到的内容。随着 Bluetooth LE Audio 的功能以及多个 Bluetooth LE Audio Source 的预计可用性，这将发生变化。这些来源可能是个人的，比如我们的手机、笔记本电脑和电视；公共，如安装在餐厅、酒吧、健身房或办公室的广播发送器；或商业，如在电影院或当我们听现场表演时。

Commander 意味着我们将能够在不接触手机的情况下进行更多的选择和控制。看看这对我们的随身携带和佩戴的设备有什么影响会很有趣。如果我们在不接触手机的情况下做更多的事情，那么手机作为我们交流的中心设备的重要性可能会减弱。反过来，随着语音作为一种命令手段变得自然，新的应用程序可能会开发出使用音频作为主要接口的应用程序，而无需与屏幕交互。还有很长的路要走，但智能手机永远不会成为我们个人通信发展的最后一步——其他一些产品也会出现，就像智能手机本身一样。Bluetooth LE Audio 音频带来的新功能使语音和音频更加普及，可能会加速这一变化。

使用多个 Commander 的能力意味着可穿戴设备获得了更大的效用。虽然并非所有这些都与音频有关，但这增加了购买和佩戴它们的原因。这将有助于整个可穿戴设备行业，该行业仍在努力达到他们所希望的销量或客户兴趣。

所有这些变化都需要时间。有些事情会更早发生，有些事情会更晚发生。有些实现可能无法说服用户，但几年后又成功重新使用。大多数新的个人技术都是如此。显而易见的是，它们将改变音频生态系统的动态，为音频制造商提供更多的创新机会，削弱智能手机的实际地位。在本章中，我们将 Bluetooth LE Audio 的一些不同场景，探讨呈现音频共享服务的新方式的机会，确定启用这些服务所需的内容，并评估这对手机设计和其他产品设计可能意味着什么。

Bluetooth LE Audio 规范本身不会改变我们使用音频的方式。为了有效，该行业的许多部门——硬件开发人员、应用程序开发人员、服务提供商和 UX 设计师——需要了解各种可能性，并了解如何使这些新的音频体验引人

---

注目，让用户发现音频如何以新的方式与他们的生活相适应，然后在此基础上提供新的音频使用方式。

## 12.2 Broadcast for all

毫无疑问，目前使用感应线圈的用户和场馆都将欢迎 Bluetooth LE Audio 的出现，作为 hearing reinforcement 的下一步。质量要高得多，安装成本也要低得多。对于助听器使用者来说，它保证了他们可以在更广泛的地方进行听力增强。对于建筑业主和建筑师来说，这应该意味着听力增强成为新建筑的标准特征。(令人遗憾的是，许多新建筑缺乏感应线圈，更多的原因是建筑师缺乏理解，而不是当前技术的成本)。

随着助听器行业对 Bluetooth LE Audio 的持续推广，随着广播产品的推出，这一切都会自然而然地发生。然而，用户将需要新的助听器来接收这些广播，该助听器包含 Bluetooth LE Audio。目前，只有小部分助听器包含任何形式的非专利 Bluetooth 无线技术，需要一段时间才能出现大量用户。与消费市场相比，助听器市场相对较小——新的 Bluetooth LE Audio 助听器可能需要至少五年的时间才能达到 1000 万用户，这主要是因为助听器在销售前需要医疗批准，这会减缓上市时间。这引发了一个有趣的问题，即消费者产品是否会推动音频广播基础设施的初步推出，这将使每个人受益，无论他们是否有听力损失。

### 12.2.1 Democratizing sound reinforcement

如今，公共广播/感应器助听器服务提供商在设计产品时主要考虑的是佩戴助听器的人。这些服务通常以低延迟复制和增强音频公告。很少有人在考虑这是否与没有听力损失的人相关，以及如何扩展到此类人群。很可能许多耳塞和耳机的用户会发现它们的益处。如果发生了，那么大多数人(当然在短期内)很可能会使用智能手机上的应用程序找到并选择它们。因此，第一步将是手机操作系统公布这些广播的信息，允许用户选择并收听这些广播。换言之，我们需要看到类似于图 12.1 的应用程序接口，模仿今天已经完成的发现和配对 Wi-Fi 和 Bluetooth 设备的工作。

尽管这是大多数用户都熟悉的格式，但它仍然比需要的更难操作。图 12.1 的示例说明了一个事实，即第一批应用程序之一可能是公共交通。许多公交车站都有公告牌，但与火车站台不同，它们不会发布公共音频公告，因为它们会干扰街道。许多国家的国家指南已经要求将广播纳入新的公共基础设施，并可能在未来的建议中包括 Bluetooth LE Audio。正如这些看起来的

---

那样，好的交通应用程序应该开始包括对它们的支持，这样旅行应用程序就可以自动检测到是否存在合适的音频流，并询问用户是否愿意收听，从而省去了进入 Bluetooth 设置搜索音频流的不便。

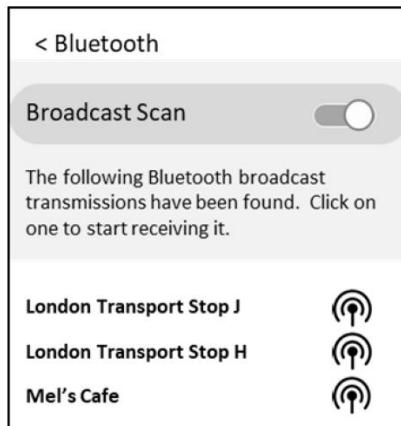


图 12.1 找到 Bluetooth LE Audio Broadcast 的简单用户接口

Bluetooth LE Audio 规范中有几个工具可以帮助开发集成应用程序的良好用户体验。回顾有关广播的章节，广播广告中应包含许多信息，以帮助扫描设备过滤和分类不同的广播。这些是：

- Local Name AD Type，提供设备名称。通常，这是可用广播发送器列表中显示的主要信息项。
- Program\_Info LTV。LTV 结构，其包括关于正在广播的内容的信息。它类似于 TV Electronic Program Guide 中包含的顶级信息，例如电视频道。
- BASE，其可以包含关于内容的进一步信息，包括内容的名称，例如，电视广播的特定节目名称，以及传输语言。

有了这些信息，代表你的耳塞或助听器进行扫描的应用程序可以获得大量信息，以指导他们开始将广播音频嵌入用户体验。同样，广播音频信息的提供者需要仔细考虑如何使用它，以及如何最好地支持应用程序。大多数公共广播的主要用途仍然是声音增强，其中广播音频被设计为低延迟，以便可以与环境音频一起接收和呈现。现在可以为这些公告添加多种语言，但应该规划它们，这样它们不会与其他语言的环境公告冲突，否则它们将更难理解。这是一个很小但很明显的细微差别，但是开发人员需要学习的众多细节之一。

Broadcast stream 提供商还需要考虑用于标记流的信息。当你从一个公共汽车站移动到一辆公共汽车上时，你的应用程序应该检测到作为

---

Broadcast Source 的公共汽车站的丢失，并切换到你所乘坐的公共汽车上的 Broadcast Source。在伦敦这样的地方，公共汽车密度很高，你很有可能在同一路线上的另一辆公共汽车旁边，但行驶方向相反，因此，应用程序应该包含足够的基本智能，以检测它是否已连接到正确的发射器。只要 Local Device 名称和 ProgramInfo 值被合理地赋予，就应该很容易确定。然而，这需要服务提供商、应用程序开发人员和设备制造商共同努力，为用户提供最佳体验。

大多数公共广播流将是单声道语音流，可以使用很少的广播时间，通过 16\_2\_2 QoS 设置对其进行充分编码。这些信息很可能是不常见的——在公共汽车的例子中，要么是公共汽车到站的公告，要么是你上车时，对下一站的公告。如果您的手机有足够的资源，旅行应用程序应该能够监控适当的 BIS，在检测到 BIS 包含音频消息时将其与任何其他音频流混合，并在只有空数据包时忽略它。

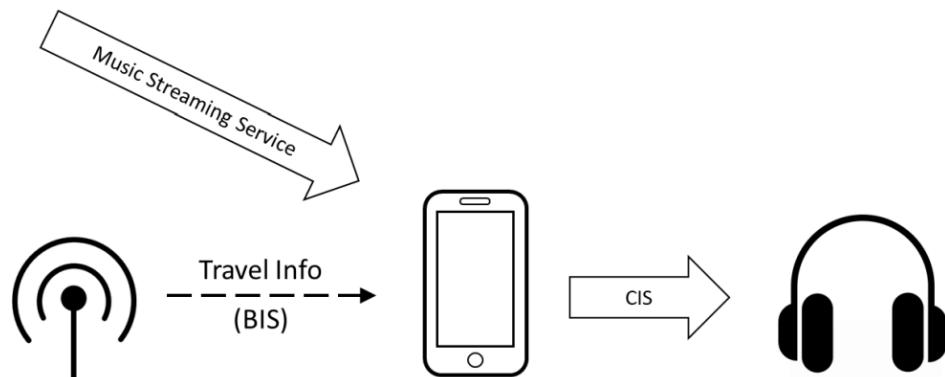


图 12.2 将广播通知混入音频流

图 12.2 显示了基本设置，其中手机同时运行音乐流应用程序和旅行应用程序，后者监控本地广播发送器的相关音频公告。图 12.3 显示了手机如何将来自不同来源的音频组合成单个流，并在单个 CIS 中进行编码和发送。这表明，Initiator 可以自由地混合它想放入流中发送到耳机的任何音频频道。

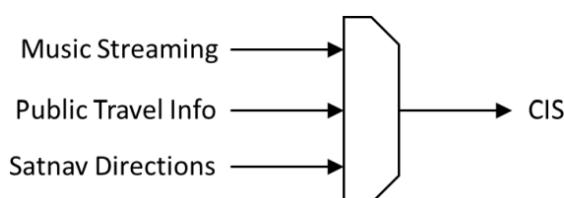


图 12.3 将应用呈现的特定音频流混入 CIS

考虑到人们开始将耳塞放在更长的时间里，使用透明模式来保持会话，

---

这是一种接收相关公告的有用方式，尤其是在他们是 silent 的外语公告时。

### 12.2.2 Audio augmented reality – the “whisper in your ear”

上面的例子是将广播旅行公告与旅行应用程序生成的其他信息混合，这是将广播音频用作增强现实元素的切入点。事实证明，增强现实和虚拟现实最初的承诺相当令人失望，除了专业的商业应用和针对全职玩家的产品之外，几乎没有成功的应用。音频可能提供一个更可接受的入门步骤，特别是对于日常增强现实。

音频带来的好处是它不那么突兀。这是耳朵里的低语，它帮助你做任何你正在做的事情，而无需购买或穿戴那些通常不方便的可穿戴科技产品。我们已经看到多轴传感器集成在耳塞中，可以检测你的头部位置，从而知道你在看什么。结合广播发送器和空间感知应用程序的信息，这些应用程序允许一些创新的新音频应用程序，声音可以开始融入您的日常体验，而无需任何特殊的 AR 或 VR 硬件。方向可以告诉你该走哪条路，该往哪里看。许多用于这些的底层技术已经存在。它只是在等待 Bluetooth LE Audio 的潜力使其成为可能。

### 12.2.3 Bringing silence back to coffee shops

上面的例子显示了如何将现有的感应线圈(助听器)应用程序扩展到更广泛的受众，并集成到当今的智能手机应用程序中。在新的领域，广播基础设施很可能同时发展。其中一个备受关注的是在咖啡馆和餐馆中使用广播提供背景音乐。

在过去的某个时候，有人认为大多数咖啡馆、酒吧和餐馆配备背景音乐是个好主意。过去十年的设计风格在很大程度上是去除任何充当噪音吸收器的家具，这使得谈话变得越来越困难，并提高了背景噪音水平。这产生了一个正反馈，顾客开始大声说话，所以工作人员把音乐调大，导致顾客不得不提高嗓门，把原本应该是愉快的体验变成了不那么愉快的体验。现在，餐馆评论通常包括噪音测量，以显示是否有可能进行对话。尽管有负面的消费者反馈，音乐依然存在。如果 Bluetooth LE Audio 能够实现改变，那就太好了。

酒店业的部分人士认为，Bluetooth LE Audio 显然可以取代扬声器，为想听的人提供音乐，降低说话人的噪音水平。

添加硬件非常简单。它只需要一个安装在现有音频系统输出端的 Bluetooth LE Audio 广播模块。芯片供应商已经在为这样的设备开发参考设

---

计，在规范被采用的 12 个月内，这样的设备应该可以以相对较低的价格轻易获得。产品设计师应确保场馆所有者可以轻松配置产品，以便客户易于发现。

#### 12.2.3.1 Making audio universally available

这就是 Public Broadcast Profile 变得重要的地方。商用 Bluetooth LE Audio 发送器应该允许选择 BAP 规范[BAP 表 6.4]中定义的任何 QoS 设置。然而，并非所有的 Acceptors 都能够解码所有这些设置。资源受限的设备，如助听器，想要实现最大可能的电池寿命，可能不支持采样率高于 24kHz 的解码。因此，如果广播发送器只支持更高的速率，佩戴者将无法听到广播音频。有一些简单的解决方案，但它们需要广播设备制造商来实现。

最简单的方法是将公共空间中的音乐传输限制在 24\_2\_1 QoS 设置。如果有任何背景噪音，即便不是一个完美的收听环境，大多数用户可能会觉得这是令人满意的。

Sampling Rate	Mono airtime	Stereo Airtime	Universally receivable
24 kHz (24_2_1)	13%	26%	Yes
32 kHz (32_2_1)	15%	30%	No
48 kHz (48_2_1)	30%	60%	No

注意：这些数字基于 Low Latency 设置。48kHz 采样数字具有更高的广播时间要求，因为配置要求 RTN 设置为 4，而不是用于 24\_2\_1 和 32\_2\_1 设置的值 2。

表 12.1 不同广播音频流的 QoS 设置 Airtime 要求

表 12.1 提供了本次讨论的数据。考虑到当前音频行业的营销趋势，许多制造商很可能希望能够将其产品宣传为“48kHz audio quality”，这是一个两难的选择。如果这些发送器支持 48kHz 立体声流，但仍希望提供通用支持，则需要至少添加 24kHz 单声道流。如果这包含在同一个 BIG 中，则广播时间将达到 90%，因为尽管它使用更少的广播时间，但 BIG 中的每个 BIS 都分配了相同的时序参数。另一种选择是为 24kHz mono stream 传输第二个单独的 BIG，组合的 BIG 将占广播时间的 72% 左右，但需要更多的广告资源。如果广播发送器使用 Wi-Fi 获取其音频流，那么这可能是行不通的。解决质量和通用接入要求的更实用的解决方案是在一个 BIG 中传输 32kHz 立体声和 24kHz 单声道音频流，这消耗总广播时间的 45% 左右。在同一 BIG 中传输立体声 24kHz 对和 32kHz 对音频流与单个 48kHz stereo stream 花费相同的广播时间。

在上面的示例中，我选择了 Low Latency QoS 设置，尽管事实上，它们包括 24kHz 和 32kHz 采样率的较少重传。在这个特定的应用程序中，延迟没有那么重要，因为通常没有视觉提示，也没有同时刻的环境声音(尽管可

---

能有)。然而，在这种应用中，广播发送器很可能被放置在墙上的高处或连接在天花板上，以获得最佳覆盖，因此没有太多阻挡物吸收传输。这意味着 Low Latency 设置足够了。如果咖啡馆老板把广播发射机放在橱柜里或服务台下面，那就不一样了。这是设备设计人员在其物理设计和安装说明中需要考虑的问题，确保该应用的广播发送器可以很容易地安装在墙上，并且连接到设备的任何导线都足够长。

设备制造商需要了解这些限制条件，并灵活地设计产品，以支持多个 BIG，并允许安装人员和场馆所有者对其进行适当定位和配置。我们不应该期望咖啡馆老板了解播放时间或 QoS 参数——制造商应该为他们提供易于安装和使用的解决方案。这包括使用适用于每个安装的值自定义所有 AD 类型名称和 LTV 字符串的能力。

一个基本的、接受音频输入的广播发送器，无论是从 3.5mm 插孔、RCA 插头还是 USB 输入，都是尽可能简单的，但对于很多场所和应用来说都是足够的。设备供应商几乎肯定会为大型机构提供更复杂的广播设备，以及配置它们、提供音频流，并将其集成到其他音频服务中的管理服务。即使有基本的 Bluetooth LE Audio 广播，也有大量的差异化机会。

## 12.3 TVs and broadcast

从 Bluetooth LE Audio 开发的早期开始，电视行业就对将电视连接到耳塞、耳机和助听器的可能性感到非常兴奋。虽然电视可以使用单播，但它的可扩展性有明显的限制，因为它需要为每个听众单独设置 CIS。因此，共识是大多数电视用户将使用广播，添加加密以确保只有授权用户才能解码音频内容。在本节中，我们将研究三个最常见的应用程序领域的需求。

### 12.3.1 Public TV

今天，大量公共安装的电视都是无声的。这不是因为它们没有音频输出，而是因为它们安装在环境音频会令人讨厌或与其他音频冲突的区域。常见的例子是机场，运营商不希望用户分心，无法听到航班和安全公告；健身房，多台电视显示不同的频道，但都是无声的，因为它们发出的多个相互冲突的声音会很刺耳；俱乐部和酒吧，即使是一台电视发出的声音(通常有多台不同频道的电视)，也会干扰正常的对话，而室外设施，声音不会被听到或干扰。

这些安装都类似于我们上面看到的简单的 Broadcaster。他们不需要加密；他们只需要一个广播发送器。这可能是一个连接到电视音频输出的类似

---

插件设备，或者是一个内置的 Bluetooth LE Audio 广播发送器。它需要允许输入设备位置，以便用户将音频广播与电视相匹配，或者它可以采取 Wi-Fi 路由，将访问信息(类似 WiFi 密码)写在墙面的通知上。

正如我们上面所看到的，许多这些设备，包括集成到电视中的设备，可能都会被管理，尤其是允许特定的消息混合到音频流中。例如，在机场，航班通告可能会被混合到音频流中，这样听电视的人就不会错过它们。这就是我们可能看到 *multiple language streams* 开始的地方。如果电视广播中有多种语言的音轨，则可以在一个 BIG 上以单独的 BIS 发送。使用 24kHz 单声道编码，应该可以从一个广播发送器提供六种不同的语言流。每个输入流将实时混合航班通告与该节目的音频输入，任何未宣布的语言将在随后的时间点混合，在那里它们不会与任何环境通告冲突。如果发送器需要支持更多信道，它可以添加 Bluetooth 发送器。每种语言由 BASE 中的 Language LTV 值标识，表示每种 BIS 的语言。用户可以将其扫描应用程序设置为选择特定语言，以便优先显示这些语言。或者，它可以显示所有可用的语言，让用户选择他们想听的语言。这需要手机的 API 公开这些信息，以及应用程序开发人员了解如何使用这些信息。

### 12.3.2 Personal – at home

个人电视有不同的优先级。许多电视已经支持 Bluetooth Classic Audio 模式，但通常仅限于连接到音箱、一组耳塞或耳机。现实是，在大多数家庭中，家庭或朋友的多个成员都会收听。虽然电视制造商可以通过 Bluetooth LE Audio 单播来实现类似的过渡，但这将很快达到少数用户的 airtime 限制。转向广播更有意义。

与上述公共电视案例不同，大多数用户不希望邻居听到他们正在收听的内容，因此对于个人电视(以及家庭中的其他音频源)，人们期望广播 Audio Streams 始终是加密的。这意味着用户需要一种简单的机制来获得 Broadcast\_Code 以解密音频流。

这就是 Broadcast Assistant 的作用所在。个人 Audio Sources 希望能被多个用户听到，期望每个用户都与之配对并绑定。这提供了长期可靠的连接。当用户进入电视范围内时，他们会要求耳机连接，电视中的 Broadcast Assistant 会使用 PAST 通知他们广播流的位置，随后是当前的 Broadcast\_Code。这一点很重要——一旦用户配对，他们就不再需要使用手机连接广播。当用户进入电视范围内时，基本的 LE 连接可以是自动的。可

---

以通过内部生成的音频消息通知用户“一对耳塞已连接”，提醒用户“存在音频源”。如果他们想连接，他们可以按下按钮或执行耳机所需的任何手势，这样他们就可以从电视接收所需的信息，开始音频流，而无需用户采取更多的动作。对于在不同音频源的房间之间走动的用户来说，这是一种连接到最近音频源的简洁方式。所有 Broadcast Assistants 都会收到通知，通知他们耳机正在停止同步或同步到新的源。这意味着所涉及的每个连接的设备都会跟踪当前状态，从而可以设计出一个非常简洁的解决方案。

如果朋友们来了，想要用耳机或助听器连接，电视主人会将电视设置为 Bluetooth Audio Sharing 模式(最好是电视遥控器上的一个按钮，而不是隐藏在电视的许多层菜单项里)。这将激活与广播流相关联的广告序列，这将为朋友提供获取 Broadcast\_Code 的方法。

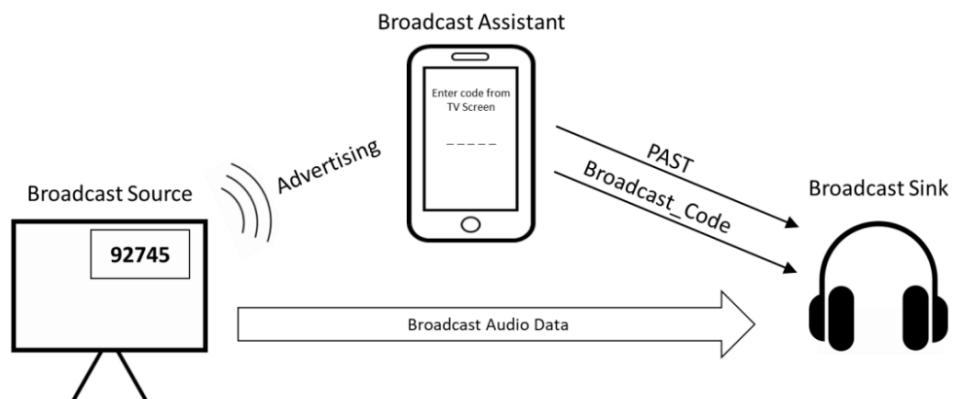


图 12.4 将朋友添加到 TV 广播的示例

Bluetooth LE Audio 规范中没有描述获取 Broadcast\_Code 的方法，因为它超出了范围。通过在电视屏幕上显示一个数字，让用户进入手机应用程序，显示二维码，或使用 NFC 或其他近距离技术，可以找到 Broadcast\_Code。市场可能会在不久的将来就标准选项达成一致。

预计 Broadcast\_Code 值将在每个会话结束时刷新，以确保持续的隐私。实际上，这很可能在每次断电时自动发生。绑定用户将不受影响，因为他们每次连接时都会自动获得新值。

### 12.3.3 Hotels

任何人在入住酒店期间，如果邻居房间的电视太吵了，都会欢迎电视中 Bluetooth LE Audio 的出现，让入住者可以安静地使用。这是一个理想的的应用程序，但存在 Bluetooth 传输穿透墙壁、地板和天花板的问题，将您正在

---

观看的内容暴露给所有邻居。

可以使用与个人电视中一样的方法，用户只要想连接，就会获得 Broadcast\_Code。同样，酒店可以靠用户拥有的一个通用的广播扫描应用程序，让他们输入电视上的 pin-code，或扫描 QR 码，手机充当耳机的 Commander。虽然这些功能都可以运行，但它们很麻烦，不会给用户带来特别流畅的体验。一个更有效的选择是将 Broadcast\_Code 纳入酒店应用程序，这样一来，一旦有人入住，他们的手机就会获得正确的信息，在入住期间保持 Broadcast\_Code 不变，以便从该应用程序访问电视。这是电视管理系统的一个相当简单的扩展，它已经集成在大多数酒店电视中。

## 12.4 Phones and broadcast

在 Bluetooth LE Audio 开发的早期，广播主要被视为一种基础设施应用，向大量人广播。随着人们对这项技术的潜力越来越了解，人们对它为智能手机提供的功能越来越兴奋。人们最兴奋的是能够在手机上与朋友分享音乐。

此种情况很简单。自从我们第一次开始在个人设备中存储音乐以来，它就一直存在。回到索尼最初的随身听时代，你会看到成对的人分享他们的耳塞，一起听音乐。随着我们从便携式音乐播放器发展到智能手机上的流媒体音乐，共享技术还没有进步。相比之下，随着智能手机 3.5mm 音频插孔的移除，共享变得更加困难，因为许多手机不再可能使用有线耳机进行共享了。

Bluetooth LE Audio 广播解决了这个问题。如果你正在听你最喜欢的音乐，你的朋友也来了，你可以让他们加入，从单播流(或 A2DP 流——我们不知道将使用什么底层技术)转换到广播流。这将您的手机设置为好友可以找到的 Broadcast Source。除非您想成为公共 Broadcaster，否则您的应用程序几乎肯定希望加密你的音频，因此您的电话应用程序需要分发 Broadcast\_Code。这可以通过 short-term 配对实现(无需绑定)。这样做的好处是，你朋友的手机一旦收到广播流，就会通知你。然后，你的手机可以停止发送广告，这样其他人就不会知道你的广播了。如果其他朋友想加入，你只需要重复这个过程。在这个过程中的任何时候，你的朋友都可以离开。一旦它们全部离开，您的手机可能会恢复为单播，因为 CIS 中的确认数据包意味着 Initiator 更省电——广播发送器必须发送所有可能的重传 Subevent。

同样，您的任何朋友都可以扮演 Broadcaster 的角色，分享他们喜爱的音乐。此过程持续的时间由他们决定。随着应用程序的发展，我们可能会找到方法，使在一个组内交换 Broadcaster 更加无缝。

---

在某些情况下，流不需要加密。任何想要建立一个特别的无声迪斯科舞厅的人只需要开启广播。

手机甚至可以在不使用手机功能的情况下利用 Bluetooth LE Audio。许多听力损失的人喜欢在会议或餐桌旁使用桌上麦克风来帮助拾音。一旦你的手机有了 Bluetooth LE Audio，一个应用程序就可以将它设置为一个私人 Broadcaster，向桌旁的其他人广播它的麦克风拾音。它不使用蜂窝通讯功能，但充当本地共享麦克风。

## 12.5 Audio Sharing

所有这些用例有个共同点——它们正在将感应线圈体验(仅限于助听器)发展到每个拥有 Bluetooth LE 可听设备(无论是助听器、耳机、一组耳塞，甚至是便携式扬声器)的人。对绝大多数人来说，这是一个新概念。Bluetooth SIG 与听力损失用户团体、助听器和消费者可听设备制造商合作，正在为制造商和场馆制定指南，以促进以 Audio Sharing 为名的广播的普遍性，以帮助鼓励其使用并确保所有用户的互操作性。

互操作性问题对于广播来说是一个重要问题，因为存在两类相互冲突的设备：

- 助听器和资源受限的可听设备，通常支持 Hearing Access profile，仅支持 LC3 的 16kHz 和 24kHz 采样率，以及
- 可能希望使用 48kHz 的最高 LC3 采样率的消费者设备。

为了解决这一差异，并确保所有用户都能获得最佳体验，Audio Sharing 指南将广播发送器分为两类——公共和个人。

- 公共广播发送器是每个人都应该能够访问的发送器。它们必须以 16kHz 或 24kHz 的较低采样率传输流，支持 Public Broadcast Profile，以通知用户这些音频流何时何地可用。检测到 Public Broadcast Service UUID 的用户将知道他们可以同步到该流。如果它们有资源，公共设备也可以同时传输更高质量的流。
- 个人广播发送器是电视、手机、个人电脑和平板电脑等设备。Audio Sharing 指南认识到，用户通常可能更喜欢接收更高质量的流，而这并不是普遍可互操作的。然而，当用户在其设备上选择符合 Audio Sharing 的广播模式时，例如当朋友要求访问您的电视或收听您的音乐流时，必须可以将其配置为通用设置，如果选择该设置，将广播 16kHz 或 24kHz 采样的符合 PBP 的音频流，以便任何

---

设备都可以接收到该模式。用户可以询问其他流接收者需要哪个选项。同样，如果设备有能力，它可以以更高的采样率同时传输流。Bluetooth SIG 计划以类似于 Wi-Fi 联盟对可公开访问的 Wi-Fi 接入点的方式推广这些指南，以帮助鼓励对新 Audio Sharing 生态系统的互操作性和信心。

## 12.6 Personal communication

虽然上面描述的所有广播应用程序都可以由个人访问，但大多数应用程序很可能由一群人使用，因此设计这些应用程序的界面时应该考虑到这一点。然而，有些应用程序将主要针对个人对话而设计。再一次，这些模拟了许多感应线圈应用，其中使用一个小的感应回路为单个人提供音频反馈。这些通常出现在售票处、出租车、银行、超市结账处和酒店接待处。它们使用一个静态麦克风来拾取助听器佩戴者的声音，并使用一个感应线圈将音频流从其他人返回到助听器。在这个应用程序中重要的是，广播音频流是私有的，不会与附近的另一个混合。虽然站在附近的人总能听到对话，但你不想让几米外的人听到。你也不希望错误的信息流被收集。因此，再次需要加密和身份验证。

查找 Broadcast Source 和获取 Broadcast\_Code 的技术与上述技术相同，但这些特定的用例正在引起人们对开发 industry-wide 的连接方法的兴趣。

### 12.6.1 Tap 2 Hear – making it simple

虽然还没有具体说明，但业界的兴趣集中在使用 NFC 提供简单的“Tap2Hear”接口，用户可以在触摸板上轻击手机或任何 Commander 设备。这种接触将传递助听器或耳塞查找正确的 Broadcast Source 和适当的 Broadcast\_Code 所需的信息。一旦完成，对话就可以开始，Broadcast\_Code 和 BIG 详细信息在会话期间保持不变。它提供了一种吸引人的简单用户体验，适用于所有类型的个人连接，无论是在超市结账、接入正确的会议室，还是连接到剧院或电影院的广播流。它同样适用于 Audio Sharing，朋友只需轻击手机即可共享音乐或访问私人电视。

### 12.6.2 Wearables take control

Bluetooth Classic Audio profile 允许在单独的设备上进行远程控制，但这些功能的使用却很少。这些功能主要局限于车载套件功能，偶尔会涉足可穿戴设备。Bluetooth LE Audio，这可能会改变。这有很多原因。

---

Remote Control 的主要驱动因素之一是耳塞的持续增长。正如助听器行业几十年前发现的那样，无论对制造商还是客户来说，将用户控件添加到耳塞或助听器这样的小东西上都不容易。因此，消费者被鼓励使用手机应用程序来控制他们的音频。然而，不断拿出手机并打开合适的应用程序远不是最佳的用户体验。Bluetooth LE Audio 规范中的控制功能使得将 Remote control 结合到其他设备中变得更加容易，并且可以根据用户的需要拥有任意数量的遥控器。这些是低功耗产品，可以很容易地使用硬币电池，因此成本低，并且可以互操作，允许任何 Bluetooth 产品集成这些功能。因此，无论是腕带、智能手表、眼镜还是服装，我们都希望看到可穿戴设备实现这一功能的趋势。这可能是一项功能，它增加了可穿戴设备对许多用户的实用性，使已经相对萎靡不振的产品行业重获生机。

### 12.6.3 Battery boxes become more important than phones

值得在耳机的普通电池充电器盒上加上几句话。这是一个必要的配件，与耳塞一起开发，使其看起来具有可用的电池寿命。第一代耳塞在需要充电之前，大多可以运行一个小时。电池盒是一个很好的主意，既可以安全地保护耳塞，又可以不断地给它们充电，让用户感觉他们可以跑一天。自这些早期产品以来，耳机的电池寿命显著提高，号称可以播放长达 10 小时的连续音乐(尽管自动降噪等处理功能已关闭)。Bluetooth LE Audio 将增加电池寿命，尽管制造商可能会借此机会增加更多耗电功能。不管怎样，电池盒将继续存在，这不仅仅是因为它们除了充电功能外，还提供了一个非常重要的功能，这是一个防止你丢失耳塞的容器。

许多电池盒已经包含 Bluetooth 芯片，以帮助配对，并且在不存在 Bluetooth 芯片的情况下提供音频流，例如在飞机上。在这里，电池盒可以插入飞机个人娱乐系统提供的 3.5mm 插孔，将声音传输到耳塞。Bluetooth LE Audio 的低功耗和低延迟使其成为这些应用中替代 Bluetooth Classic Audio 的理想选择。然而，电池盒也是许多 Remote Control 功能的理想选择。与耳塞和助听器不同，它足够大，可以放置按钮，因此是一个方便使用的音量控制器。它还可以充当 Broadcast Assistant，扫描可用的广播。总的来说，它将提供一个比将手机取出并打开应用程序更快、更简单的界面，因为 Bluetooth LE Audio 控制功能始终作为按钮存在。

在设计 Bluetooth LE Audio Controller 时，思考如何在电池盒上实现它们是一个有用的实践。对用户来说，这是一件很容易交互的事情，但由于其

---

固有的简单性，它常常被忽视。它不一定是实现这些功能的最佳设备，但它的易用性和小巧的尺寸，适合放不下手机的口袋，使它成为一个有吸引力的替代品。

随着设计师们找到方法将这样小型设备变成引人注目的用户接口，我们可能会发现，我们花在手机上的时间更少了。随着耳塞越来越善于将 Bluetooth 音频流与环境声音和对话(音频处理可以增强)混合在一起，我们可能会花更多的时间佩戴耳塞，与周围的各种事物交谈，并与音频交互。手机不会很快消失，尽管我们已经度过了智能手机的峰值，但 Bluetooth LE Audio 可能会推动应用程序，引领我们走向下一步。

## 12.7 Market development and notes for developers

有一种简单的观点认为，建筑物、剧院、酒店和咖啡馆中的广播基础设施将会出现，因为 Bluetooth LE Audio 广播设备的成本将很低，因此人们将从 eBay 和 Amazon 购买这些设备，将其插入现有的音频系统，并张贴 Bluetooth Audio Sharing 共享标志。这种情况肯定会发生。这是 Wi-Fi 早期发生的事情，当时场馆所有者正是这样做的。有了 Bluetooth LE Audio，它应该更容易，因为您不需要安装互联网连接，只需要一根电缆将其连接到现有的音频系统。

然而，有了 Wi-Fi，许多场馆业主发现，与服务提供商合作更容易，后者可以安装、管理并为场馆和客户提供支持。Bluetooth LE Audio 可能会出现相同的模式。我怀疑，我们将看到 Wi-Fi Access Point 提供商销售包括 Bluetooth LE Audio 的接入点，这样一个设备就可以同时管理这两者。目前制造和安装感应线圈的公司也有类似的机会，他们将看到自己的业务向更广泛的客户开放。它需要应用程序开发人员了解广播是如何向用户提供接口的。

在这方面，重要的是要指出，只有当芯片、操作系统和应用程序开发人员了解 Bluetooth LE Audio 的全部潜力并支持这些不同用例的功能时，这些应用程序才会出现。在早期，其中一些可能是不可能的，因为开发人员自然会集中精力发布他们认为最显著的应用程序。随着时间的推移，随着市场的了解，我们获得了大量的产品，将出现更复杂的用例，以及简单直观的用户接口。

### 12.7.1 Combining Bluetooth Classic Audio with Bluetooth LE Audio

Bluetooth Classic 音频实现不会在短期内消失。目前市场上的大多数音

---

频产品使用的是 5.2 之前的芯片组，这些芯片不可升级，从电视到汽车，许多产品的寿命都在十年或更长。至少在未来五年内，手机和大多数耳机将同时支持 Bluetooth Classic Audio 和 Bluetooth LE Audio。如果两者都支持相同的用例，例如 HFP 和 A2DP，实现将决定使用哪个。在手机中，这将在很大程度上取决于各个制造商和芯片的实现。具有更广泛协议栈和开源应用程序的产品可能更加多样化。

对于开发人员来说，重要的一点是确保它适用于用户。虽然 CAP 包含了从单播切换到广播的切换过程，但仅适用于两者都是 Bluetooth LE Audio 的情况。在现实世界中，同样可能的是，从 A2DP 到 LE 广播，然后返回到 LE 单播或 Classic Bluetooth。对于产品开发人员来说，规则必须是预测并允许任何组合。Bluetooth SIG 会定期进行 unplug test，开发者可以用其他制造商的产品测试他们的产品。随着产品开始融入越来越多的 Bluetooth LE Audio 的新功能，这些功能对于确保可互操作的生态系统和用户体验的一致性至关重要。

### 12.7.2 Concentrate on understanding broadcast

从制定规范中得到的一个教训是，对于习惯于 HFP 和 A2DP peer-to-peer 模型的人来说，广播的概念是多么困难。虽然单播包含了许多新概念（请参见第 3 章对其进行回顾），但微微网中确认包模型相对来说是熟悉的。带有发送器的 Broadcast，并不知道是否有人在听广播，接收器完全独立，没有状态机，对许多人来说，这是一个令人惊讶的挑战。添加 BASS，加上 Broadcast Assistant 和扫描代理，似乎是对传统思维的挑战。

本章中的示例试图说明其灵活性。开发人员需要了解广播带来的限制——每个 BIG 对所有 BIS 都有固定的结构，这可能意味着有时多个 BIG 更有效。Scanning 和 Filtering 广播对于良好的用户体验非常重要，因此需要支持相关域，并在适当的情况下由用户进行配置。他们必须了解 Basic Audio Announcement、Initiators 和 Acceptors 使用的 Targeted 和 General Announcement 以及 BASE 结构的含义。Broadcast Assistant 和 Scan Delegator 之间的交互，以及 BASS 中各个特征的使用，这些是上述共享和认证过程的基础。

最后，任何使用 Bluetooth LE Audio 进行设计的人都应该花时间了解和实现控制功能，并了解它们可以分布在多个不同设备之间的事实。这些都是简单的 Client-Server 关系，但每项服务的内容都非常全面，并在丰富的用户

---

体验和基本的用户体验之间产生了差异。

虽然这是一个基本的 LE 功能，但开发人员也应该确保他们了解通知的作用。在 Bluetooth LE Audio 中，通知是 Servers 确保拓扑中的所有内容都是最新的。知道什么时候会出现这些产品，以及如果它们没有按预期到达（通常是去阅读特征）该怎么办，这对于提供一个强大的生态系统至关重要，用户可以从不同的制造商那里挑选和选择多个产品，并相信它们都会相互兼容。

### 12.7.3 Balancing quality and application

开发人员不应忘记 Bluetooth LE Audio 规范中 Optional 和 Mandatory 的区别。在它们之中，很少是强制性的。许多功能都需要强制支持，例如 BAP 和 TMAP 中的强制 codec 设置，但这并不意味着应用程序需要使用它们。强制的意思是，如果应用程序选择使用它们，则必须支持它们。如果它们只是可选的，并且 Initiator 上的应用程序希望使用它们，则 Acceptor 可以拒绝该请求。这意味着产品的功能有很大的灵活性。

话虽如此，但偏离跑道可能会影响互操作性。例如，BAP 中的 QoS 设置已经过彻底测试，开发人员可以确定，每个芯片供应商都会确保其实现是可互操作的，因此应该使用它们。然而，有时会遇到物理问题，典型地是如果你想要传输多个音频流，那么你将开始耗尽 airtime。我们从音频开发的历史中知道，消费者不一定想要最高的质量——他们想要易于使用。CD 和 流媒体服务之所以成功，是因为它们易于使用，而且音频质量足够。发明它们的公司明白，通过勇敢地从音频质量的跑步机上后退一步，它们可以提供一种能够赢得客户芳心的体验。Bluetooth LE Audio 有潜力提供新的、引人注目的服务，在设计时应考虑到这一点。音频质量在需要的时候就在那里，但其他很多东西也是如此。

### 12.7.4 Reinventing audio

在过去的几十年中，我们消费音频的方式已经集中在手机制造商和流媒体服务的手中。尽管苹果的 Airpods 及其竞争对手取得了巨大的成功，但事实上，这很大程度是狗的尾巴。助听器增加了巧妙的功能，但它们在音频链的发展中并没有真正发挥任何重要作用——它们仍然是手机的被动外围设备。Bluetooth LE Audio 的新拓扑结构和分布式控制功能为新一代创新提供了潜力，我们尚未想象的设备将成为我们生活的重要组成部分。这时，可听动物的尾巴可以开始摇晃狗了。参与 Bluetooth LE Audio 开发的每个人的目标都

---

是为新一代创新提供工具。我希望这本书能启发你跳出框框，思考如何实现这一目标。

---

## 第13章 Glossary and concordances

在最后的章节，我列出了组成 Bluetooth LE Audio 的所有规范、本书中使用的缩写词，以及所有 Bluetooth LE Audio 的 procedures 和 sub-procedures，以及它们是在哪里定义的。我只能希望在本书中提供一个对于这些规范的概览。这些 cross-references 可以帮助你浏览这些规范。

### 13.1 Abbreviations and initialisms

本书中使用如下的缩写和首字母缩写，并且贯穿于 Bluetooth LE Audio 的各个规范。

Acronym/Initialism	Meaning
3GPP	Third Generation Partnership Project
A2DP	Advanced Audio Distribution Profile
ACAD	Additional Controller Advertising Data
ACL	Asynchronous Connection-oriented link
ACL	Asynchronous Connectionless
AD	Advertising Data
AdvA	Advertiser Address
AICS	Audio Input Control Service
ASCS	Audio Stream Control Service
ASE	Audio Stream Endpoint
ATT	Attribute Protocol
AVDTP	Audio Video Distribution Transport Protocol
AVRCP	Audio Video Remote Control Profile
BAP	Basic Audio Profile
BAPS	The set of BAP, ASCS, BASS and PACS
BASE	Broadcast Audio Source Endpoint
BASS	Broadcast Audio Scan Service
BFI	Bad Frame Indication
BIG	Broadcast Isochronous Group
BIS	Broadcast Isochronous Stream
BMR	Broadcast Media Receiver
BMS	Broadcast Media Sender
BN	Burst Number
BR/EDR	Basic Rate/Enhanced Data Rate
BW	Bandwidth
CAP	Common Audio Profile
CAS	Common Audio Service
CCID	Content Control Identifier
CCP	Call Control Profile
CG	Call Gateway
CIE	Close Isochronous Event
CIG	Connected Isochronous Group
CIS	Connected Isochronous Stream
CMAC	Cipher-based Message Authentication Code
CRC	Cyclic Redundancy Check
CSIP	Coordinated Set Identification Profile
CSIPS	The set of CSIP and CSIS
CSIS	Coordinated Set Identification Service
CSS	Core Specification Supplement
CT	Call Terminal
CTKD	Cross-Transport Key Derivation
CVSD	Continuous Variable Slope Decode
DCT	Discrete Cosine Transform
DECT	Digital Enhanced Cordless Telecommunications
DSP	Digital Signal Processor
DUN	Dial-up Networking
EA	Extended Advertisement
EATT	Enhanced ATT
EIR	Extended Inquiry Response
FB	Full Band (20 kHz audio bandwidth)

---

FT	Flush Timeout
GAF	Generic Audio Framework
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GC	Group Count
GMCS	Generic Media Control Service
GPS	Global Positioning System
GSS	GATT Specification Supplement
GTBS	Generic Telephone Bearer Service
HA	Hearing Aid (as in the role described in the Hearing Access Profile)
HAP	Hearing Access Profile
HARC	Hearing Aid Remote Controller
HAS	Hearing Access Service
HAUC	Hearing Aid Unicast Client
HCI	Host Controller Interface
HDMI	High-Definition Media Interface
HFP	Hands-Free Profile
HQA	High Quality Audio
IA	Identity Address
IAC	Immediate Alert Client
IAS	Immediate Alert Service
INAP	Immediate Need for Audio related Peripheral
IRC	Immediate Repeat Count
IRK	Identity Resolving Key
ksps	kilo samples per second
L2CAP	Logical Link Control and Adaption Protocol
LC3	Low Complexity Communication Codec
LD-MDCT	Low Delay Modified Discrete Cosine Transform
LE	Low Energy (as in Bluetooth Low Energy)
LL	Link Layer
LSB	Least Significant Bit
LSO	Least Significant Octet
LTK	Long Term Key
LTPF	Long Term Postfilter
LTV	Length   Type   Value
MAC	Message Authentication Code
MCP	Media Control Profile
MCS	Media Control Service
MDCT	Modified Discrete Cosine Transform
MEMS	Microelectromechanical System
MIC	Message Integrity Check
MICP	Microphone Control Profile
MICS	Microphone Control Service
MSB	Most Significant Bit
mSBC	Modified SBC (codec)
MSO	Most Significant Octet
MTU	Maximum Transmission Unit
NB	Narrow Band (4 kHz audio bandwidth)
NESN	Next Expected Sequence Number
NPI	Null Payload Indicator
NSE	Number of Subevents
OOB	Out of Band
OTP	Object Transfer Profile
OTS	Object Transfer Service
PA	Periodic Advertisement
PAC	Published Audio Capabilities
PACS	Published Audio Capabilities Service
PAST	Periodic Advertising Synchronization Transfer
PBA	Public Broadcast Assistant
PBAS	Public Broadcast Audio Stream Announcement
PBK	Public Broadcast Sink
PBP	Public Broadcast Profile
PBS	Public Broadcast Source
PCM	Pulse Code Modulation
PDU	Protocol Data Unit
PHY	Physical Layer
PLC	Packet Loss Concealment
PSM	Protocol Service Multiplexer
PSM	Protocol/Service Multiplexer
PTO	Pre-Transmission Offset
QoS	Quality of Service
RAP	Ready for Audio related Peripheral
RFU	Reserved for Future Use
RSI	Resolvable Set Identifier

---

RTN	Retransmission Number
SBC	low-complexity Sub Band Codec
SDP	Service Discovery Protocol
SDU	Service Data Unit
SIRK	Set Identity Resolving Key
SM	Security Manager
SN	Sequence Number
SNS	Spectral Noise Shaping
SSWB	Semi Super Wide Band (12 kHz audio bandwidth)
SWB	Super Wide Band (16 kHz audio bandwidth)
TBS	Telephone Bearer Service
TMAP	Telephony and Media Audio Profile
TMAS	Telephony and Media Audio Service
TNS	Temporal Noise Shaping
UCI	Uniform Caller Identifier
UI	User Interface
uint48	unsigned 48-bit integer
UMR	Unicast Media Receiver
UMS	Unicast Media Sender
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTF	Unicode Transformation Format
UUID	Universally Unique Identifier
UX	User Experience
VCP	Volume Control Profile
VCS	Volume Control Service
VOCS	Volume Offset Control Service
VoIP	Voice over IP
WB	Wide Band (8 kHz audio bandwidth)

表 13.1 Acronyms and initialisms

## 13.2 Bluetooth LE Audio specifications

下面这些是为 Bluetooth LE Audio 开发的新的规范的一部分。它们是建立在 Core 5.2 版本及以上的新特性的基础之上。

### 13.2.1 Adopted Specifications

这些规范已经被采用，并且具体实现可以符合这些规范。

Specification	Full Name	Family
AICS	Audio Input Control Service	Generic Audio Framework
ASCS	Audio Stream Control Service	Generic Audio Framework
BAP	Basic Audio Profile	Generic Audio Framework
BASS	Broadcast Audio Scan Service	Generic Audio Framework
CCP	Call Control Profile	Generic Audio Framework
CSIP	Coordinated Set Identification Profile	Generic Audio Framework
CSIS	Coordinated Set Identification Service	Generic Audio Framework
GMCS	Generic Media Control Service (part of MCS)	Generic Audio Framework
GTBS	Generic Telephone Bearer Service (part of TBS)	Generic Audio Framework
LC3	Low Complexity Communication Codec	Codec
MCP	Media Control Profile	Generic Audio Framework
MCS	Media Control Service	Generic Audio Framework
MICP	Microphone Control Profile	Generic Audio Framework
MICS	Microphone Control Service	Generic Audio Framework
PACS	Published Audio Capability Service	Generic Audio Framework
TBS	Telephone Bearer Service	Generic Audio Framework
CAP	Common Audio Profile	Generic Audio Framework
CAS	Common Audio Service	Generic Audio Framework
HAP	Hearing Access Profile	Top level profile
HAS	Hearing Access Service	Top level profile
PBP	Public Broadcast Profile	Top level profile
TMAP	Telephony and Media Audio Profile	Top level profile
TMAS	Telephony and Media Audio Service	Generic Audio Framework

表 13.2 已采用的 Bluetooth LE Audio 规范

### 13.2.2 Draft specifications

### 13.3 Procedures in Bluetooth LE Audio

下列是定义在 Bluetooth LE Audio 规范中的 procedures。注意，有些具有相同名字的过程。而它们是具有“上下文敏感性”的不同过程。

Procedure or sub-procedure name	Specification	Section
Answer Incoming Call		
Call Control Point Procedures		
Join Calls		
Move Call To Local Hold		
Move Locally And Remotely Held Call To Remotely Held Call		
Move Locally Held Call To Active Call		
Originate Call		
Read Bearer List Current Calls		
Read Bearer Provider Name		
Read Bearer Signal Strength		
Read Bearer Signal Strength Reporting Interval		
Read Bearer Technology		
Read Bearer UCI		
Read Bearer URI Schemes Supported List		
Read Call Control Point Optional Opcodes		
Read Call Friendly Name		
Read Call State		
Read Content Control ID		
Read Incoming Call		
Read Incoming Call Target Bearer URI		
Read Status Flags		
Set Bearer Signal Strength Reporting Interval		
Terminate Call		
ASE Control operations	BAP	
ASE ID discovery		
Audio capability discovery		
Audio data path removal		
Audio data path setup		
Audio role discovery		
Available Audio Contexts discovery		
Broadcast Audio Stream configuration		
Broadcast Audio Stream disable		
Broadcast Audio Stream establishment		
Broadcast Audio Stream Metadata update		
Broadcast Audio Stream reconfiguration		
Broadcast Audio Stream release		
Broadcast Audio Stream state management		
CIS loss		
Codec configuration		
Disabling an ASE		
Enabling an ASE		
QoS configuration		
Receiver Start Ready		
Receiver Stop Ready		
Released ASEs or LE ACL link loss		
Releasing an ASE		
Supported Audio Contexts discovery		
Updating Metadata		
Configure Audio Input Description Notifications	VCP	
Configure Audio Input State Notifications		
Configure Audio Input Status Notifications		
Configure Audio Location Notifications		
Configure Audio Output Description Notifications		

Configure Volume Flags Notifications		
Configure Volume Offset State Notifications		
Configure Volume State Notifications		
Mute		
Mute		
Read Audio Input Description		
Read Audio Input State		
Read Audio Input Status		
Read Audio Input Type		
Read Audio Location		
Read Audio Output Description		
Read Gain Setting Properties		
Read Volume Flags		
Read Volume Offset State		
Read Volume State		
Relative Volume Down		
Relative Volume Up		
Set Absolute Volume		
Set Audio Input Description		
Set Audio Location		
Set Audio Output Description		
Set Automatic Gain Mode		
Set Gain Setting		
Set Initial Volume		
Set Manual Gain Mode		
Set Volume Offset		
Unmute		
Unmute		
Unmute/Relative Volume Down		
Unmute/Relative Volume Up		
Configure Mute Notifications	MICP	
Read Mute		
Set Mute		
Lock Release procedure	CSIP	
Lock Request procedure		
Ordered Access procedure		
Set Members Discovery procedure		
Fast Forward Fast Rewind	MCP	
Move to First Group		
Move to First Segment		
Move to First Track		
Move to Group Number		
Move to Last Group		
Move to Last Segment		
Move to Last Track		
Move to Next Group		
Move to Next Segment		
Move to Next Track		
Move to Previous Group		
Move to Previous Segment		
Move to Previous Track		
Move to Segment Number		
Move to Track Number		
Pause Current Track		
Play Current Track		
Read Content Control ID		
Read Current Track Object Information		
Read Current Track Segments Object Information		
Read Media Control Point Opcodes Supported		
Read Media Information		
Read Media Player Icon Object Information		
Read Media State		
Read Next Track Object Information		
Read Parent Group Object Information		
Read Playback Speed		
Read Playing Order		

Read Playing Order Supported		
Read Seeking Speed		
Read Track Duration		
Read Track Position		
Read Track Title		
Search		
Set Absolute Track Position		
Set Current Group Object ID		
Set Current Track Object ID		
Set Next Track Object ID		
Set Playback Speed		
Set Playing Order		
Set Relative Track Position		
Stop Current Track		
Track Discovery - Discover by Current Group Object ID		

表 13.4 Bluetooth LE Audio 规范中定义的 Procedures and sub-procedures

### 13.4 Bluetooth LE Audio characteristics

下列是在 Bluetooth LE Audio service 规范中定义的 characteristics。参考 characteristic 定义所在的规范的章节，如 table 中的记录。

Characteristic	Specification	Table
ASE Control Point	ASCS	
Sink ASE		
Source ASE		
Audio Input Control Point	AICS	
Audio Input Description		
Audio Input State		
Audio Input Status		
Audio Input Type		
Gain Setting Properties		
Audio Location	VOCS	
Audio Output Description		
Volume Offset Control Point		
Volume Offset State		
Volume Control Point	VCS	
Volume Flags		
Volume State		
Available Audio Contexts	PACS	
Sink Audio Locations		
Sink PAC		
Source Audio Locations		
Source PAC		
Supported Audio Contexts		
Bearer List Current Calls	TBS	
Bearer Provider Name		
Bearer Signal Strength		
Bearer Signal Strength Reporting Interval		
Bearer Technology		
Bearer Uniform Caller Identifier (UCI)		
Bearer URI Schemes Supported List		
Call Control Point		
Call Control Point Optional Opcodes		
Call Friendly Name		

Call State			
Content Control ID (CCID)			
Incoming Call			
Incoming Call Target Bearer URI			
Status Flags			
Termination Reason			
Broadcast Audio Scan Control Point	BASS		
Broadcast Receive State			
Content Control ID	MCS		
Current Group Object ID			
Current Track Object ID			
Current Track Segments Object ID			
Media Control Point			
Media Control Point Opcodes Supported			
Media Player Icon Object ID			
Media Player Icon URL			
Media Player Name			
Media State			
Next Track Object ID			
Parent Group Object ID			
Playback Speed			
Playing Order			
Playing Orders Supported			
Search Control Point			
Search Results Object ID			
Seeking Speed			
Track Changed			
Track Duration			
Track Position			
Track Title			
Coordinated Set Size		CSIS	
Set Identity Resolving Key			
Set Member Lock			
Set Member Rank			
Mute		MICS	
TMAP Role	TMAP		

表 13.5 Bluetooth LE Audio 规范中定义的 Characteristics

### 13.5 Bluetooth LE Audio terms

以下特定术语在 Bluetooth LE Audio 规范中定义和使用。当使用它们定义的含义时，通常是大写的。如果使用缩写、首字母缩写或首字母缩写，则应在术语后注明。

Phrase	Specification	Section
Additional Controller Advertising Data (ACAD)	Core	Volume 6, Part B, Section 2.3.4.8
Application Profile		Volume 1, Part A, Section 6.3
Broadcast Isochronous Group (BIG)		Volume 6, Part B, Section 4.4.6.2
Broadcast Isochronous Stream (BIS)		Volume 6, Part B, Section 4.4.6.1

BIG Sync Delay		Volume 6, Part B, Section 4.4.6.4
CIG Identifier		Volume 6, Part B, Section 4.5.14
CIG Sync Delay		Volume 6, Part B, Section 4.5.4.1.1
CIS Identifier		Volume 6, Part B, Section 4.5.13.1
Connected Isochronous Group (CIG)		Volume 6, Part B, Section 4.5.14
Connected Isochronous Stream (CIS)		Volume 6, Part B, Section 4.5.13
Enhanced ATT (EATT) bearer		Volume 3, Part F, Section 3.2.1
Extended advertising (EA)		Volume 6, Part B, Section 2.3.1
Generic Access Profile (GAP)		Volume 3, Part C
Link Layer (LL)		Volume 6, Part B
Low Energy asynchronous connection (LE ACL)		Volume 1, Part A, Section 3.5.4.6
PA_Interval		Volume 6, Part B, Section 2.3.4.6
Periodic Advertising Synchronization Transfer (PAST)		Volume 3, Part C, Section 9.5.4
Periodic Advertising Train (PA)		Volume 6, Part B, Section 4.4.5.1
SyncInfo		Volume 6, Part B, Section 2.3.4.6
Unenhanced ATT bearer		Volume 3, Part A, Section 10.2
ASE identifier (ASE ID)	ASCS	Section 4.1
ASE state machine		Section 3
Audio Stream Endpoint (ASE)		Section 4.1
Sink ASE		Section 4
Source ASE		Section 4
Audio Channel	BAP	Section 1.6
Audio Configuration		Section 4.4
Audio Location		Section 1.6 (and GA Assigned Numbers)
Audio Sink		Section 1.6 and 3.3
Audio Source		Section 1.6 and 3.3
Broadcast Audio Source Endpoint (BASE)		Section 3.7.2.2
Broadcast Audio Stream		Section 1.6
Broadcast Sink		Section 1.6
Broadcast Source		Section 1.6
Broadcast ID		Section 3.7.2.1
Presentation Delay		Section 7
Remote Broadcast Scanning		Section 6.5
Scan Offloading		Section 6.5
Unicast Audio Stream		Section 1.6
Call Control Client	CCP	Section 2
Call Control Server		Section 2
Call Gateway (CG)	TMAP	Section 2.2
Call Terminal (CT)		Section 2.2
Caller ID	TBS	Section 1.9
Inband Ringtone		Section 1.9
Local Retrieve		Section 1.9
Remote Hold		Section 1.9
Silent Mode		Section 1.9
Coordinated Set	CSIP	Section 2.1
Set Coordinator		Section 2
Set Members		Section 2
Media Control Client	MCP	Section 2.1
Media Control Server		Section 2.1
Packet Loss Concealment (PLC)	LC3	Appendix B

---

Context Type Published Audio Capabilities (PAC) record	PACS	Section 1.9 (and GA Assigned Numbers) Section 2.2
Service Data AD data type Service UUID	CSS	Section 1.11 Section 1.1

表 13.6 Bluetooth LE Audio 规范中定义的术语

---

## 第14章 Index

Phrase	Meaning	Section
ACAD		
Acceptor		
ACL link loss		
Additional Controller Advertising Data		
ADV EXT IND		
Advertising Set ID		
AICS		
AirPods		
Airtime		
Anchor Point		
Announcements		
ASCS		
ASE Control Point		
ASE state machine		
ASE ID		
ASHA		
Audio Announcement		
Audio Channel		
Audio Configuration		
Audio Data Path		
Audio Input Control Service		
Audio Location		
Audio quality		
Audio Sharing		
Audio Sink		
Audio Stream Configuration Service		
Audio Stream Control Service		
Audio Stream Endpoint		
Audio_Channel_Counts		
Audio_Channel_Location		
Automatic Gain Control		
Autonomous Operation		
AUX ADV IND		
AUX CHAIN IND PDU		
AUX SYNC IND		
Auxiliary Pointer		
Availability		
Available Audio Contexts		
Available Audio Contexts characteristic		
Available Source Contexts		
BAP		
BAP Broadcast Audio Stream Establishment		
BASE		
Basic Audio Announcement		
Basic Audio Announcement Service		
Basic Audio Profile		
BASS		
Bidirectional CIS		
BIG		
BIG_Offset		
BIG_Sync_Delay		
BIGInfo		
BIS		
BIS_Spacing		
Bitrate		
BN		
Bragi		
Broadcast Assistant		
Broadcast Audio Announcement		
Broadcast Audio Announcement Service		
Broadcast Audio Reception Ending procedure		
Broadcast Audio Reception Start procedure		
Broadcast Audio Reception Stop procedure		
Broadcast Audio Scan Control Point		
Broadcast Audio Scan Service		
Broadcast Audio Source Endpoint		
Broadcast Audio Start procedure		

---

Broadcast Audio Stop procedure	
Broadcast Audio Stream configuration procedure	
Broadcast Audio Stream disable procedure	
Broadcast Audio Stream establishment procedure	
Broadcast Audio Stream Metadata update procedure	
Broadcast Audio Stream reconfiguration procedure	
Broadcast Audio Stream release procedure	
Broadcast Audio Streams	
Broadcast Audio Update procedure	
Broadcast Isochronous Group	
Broadcast Isochronous Stream	
Broadcast Isochronous Terminate procedure	
Broadcast Receive State	
Broadcast Receive State characteristic	
Broadcast Receiver	
Broadcast Source	
Broadcast Source State Machine	
Broadcast to Unicast Audio Handover procedure	
Broadcast Code	
Broadcast ID	
Burst Number	
Call Control ID	
Call Control Profile	
Call Gateway	
Call State characteristic	
Call Terminal	
CAP	
CCID	
CCP	
Change Microphone Gain Settings procedure	
Change Volume Mute State procedure	
Change Volume Offset procedure	
Change Volume procedure	
Change Counter	
Channel Allocation	
CIG	
CIG reference point	
CIG state machine	
CIG synchronization point	
CIG_Sync_Delay	
CIS	
Close Isochronous Event	
Codec Configuration procedure	
Codec Configuration Settings	
Codec Configured	
Codec Configured state	
Codec Specific Capabilities	
Codec Specific Configuration	
Codec Frame Blocks Per SDU	
Codec ID	
Codec_Specific_Capabilities	
Codec_Specific_Configuration	
Commander	
Common Audio Profile	
Connected Isochronous Group	
Connected Isochronous Stream	
Content Control ID	
Context Types	
Control Subevent	
Coordinated Set	
Coordinated Set Identification Profile	
Coordinated Set Identification Service	
Coordination Control	
CSIP	
CSIS	
CSSN	
CSTF	
Disabling state	
EHIMA	
Encryption	
Ending a unicast stream	
Extended Advertising	
Extended Attribute Protocol	
Flush Point	

---

Flush Timeout		
Frame Length		
Frame Size		
Framing		
Frequency hopping		
FT		
General Announcement		
Generic Audio Framework		
Generic Media Control Service		
Generic Telephone Bearer Service		
GMCS		
Group Count		
Groups		
GTBS		
Handover		
HAP		
HAS		
Hearing Access Profile		
Hearing Access Service		
Immediate Need for Audio related Peripheral		
Immediate Repetition Count		
INAP		
In-band ringtone		
Incoming call		
Initiator		
IRC		
ISO PDU		
ISO Interval		
ISOAL		
Isochronous Adaptation Layer		
Isochronous Interval		
Isochronous Payloads		
Isochronous Stream		
Latency		
LC3		
LE Create CIS command		
LE Remove CIG command		
LE Set CIG Parameters		
LE Set Extended Advertising Data		
LE Set Periodic Advertising Data		
Link Layer ID		
Local Name AD Type		
Locally Held		
Lock		
Low Complexity Communications Codec		
Low Latency		
Made for iPhone		
Max Transport Latency		
Maximum Transmit Latency		
Maximum SDU Size		
MCP		
MCS		
MCS state machine		
Media Control Client		
Media Control Profile		
Media Control Service		
MEMS microphone		
Metadata		
MICP		
Microphone control		
Microphone Control Profile		
Microphone Mute State procedure		
MICS		
Missing Acceptors		
Missing set members		
Modify Broadcast Source procedure		
MP3		
mSBC		
Multi-channel		
Multi-profile		
Mute		
Near Field Magnetic Induction		
NFMI		
NSE		

---

Number of Subevents	
Object Transfer Service	
Out of band ringtones	
PAC record	
Packet Loss Concealment	
Packing	
PACS	
PAST	
PBP	
Periodic Advertising	
Periodic Advertising Sync Transfer	
Periodic Advertising Synchronisation	
Playback speed	
Playing order	
Playing Tracks	
PLC	
Preferred Audio Contexts	
Preferred Retransmission Number	
Presentation Delay	
Presets	
Pre-Transmission Offset	
Program Info	
PTO	
Public Broadcast Profile	
Published Audio Capabilities Service	
Published Audio CapabilitiesService	
QoS	
QoS configuration procedure	
QoS Configured state	
Quality of Service	
Rank	
RAP	
Ready for Audio related Peripheral	
Receiver Start Ready	
Receiver Stop Ready	
Releasing state	
Remote Broadcast Scanning	
Remote Control	
Remotely Held	
Resolvable Set Identity	
Retransmission	
Retransmission Number	
Retransmissions	
Ringtone	
Robustness	
RTN	
Sampling Frequency	
Sampling Rate	
SBC	
Scan Delegator	
SDU Interval	
Set Coordinator	
Set Identity Resolving Key	
Set Member	
Set Member Lock	
Set Member Rank	
Silent Mode	
Sink ASE	
Sink ASE state machine	
Sink Audio Location	
Sink led journey	
Solicitation Requests	
Source ASE	
Source ASE state machine	
Source Audio Locations	
Streaming Audio Contexts	
Sub_Interval	
Sub_Interval spacing	
Subevent	
Supported Audio Context Types	
Supported Audio Contexts	
Supported Audio Channel Counts	
Supported Codec Specific Capabilities	
Supported Frame Durations	

---

Supported Max Codec Frames Per SDU	
Supported Octets Per Codec Frame	
Supported Sampling Frequencies	
Synchronisation Point	
Synchronisation Reference	
SyncInfo	
Target Latency	
Target PHY	
Targeted Announcement	
TBS	
TBS Call Control Point characteristic	
TBS state machine	
Telephone Bearer Service	
Telephony and Media Audio Profile	
Terminating calls	
TMAP	
Top level profiles	
Track Position	
Tracks	
Transport Delay	
True Wireless	
Unicast Audio Start procedure	
Unicast Audio Stop procedure	
Unicast Audio Update procedure	
Unicast Media Receiver	
Unicast Media Sender	
Unicast to Broadcast Audio Handover procedure	
Updating unicast metadata	
URI	
VCP	
VCS	
VOCS	
Volume Control Profile	
Volume Control Service	
Volume Controller	
Volume Offset Control Service	
Volume State characteristic	