GAPRole 任务

GAPRole 任务是一个单独的任务，它从应用程序中卸载处理大部分 GAP 层功能。此任务在初始化期间由应用程序启用和配置。基于此配置，许多低功耗蓝牙协议栈事件由 GAPRole 任务直接处理，从不传递给应用程序。存在回调，应用程序可以向 GAPRole 任务注册，以便应用程序任务可以收到某些事件的通知并相应地进行。

根据设备的配置，GAP 层始终以以下四种角色之一运行：

- 广播 - 该设备是不可连接的广告商。
- 观察者 - 设备扫描广告但无法发起连接。
- 外围设备 - 该设备是一个可连接的广告器，并在单个链路层连接中作为从设备运行。
- Central - 设备扫描广告并启动连接，并在单个或多个链路层连接中作为主设备运行。低功耗蓝牙中央协议栈最多支持三个同时连接。

的蓝牙核心规范版本5.1允许多角色，其由蓝牙低功耗协议栈所支持的某些组合。有关蓝牙低功耗堆栈功能的配置，请参阅开发自定义应用程序

有关支持的 GAPRole API，请参阅BLE Stack API 参考。

外围角色

外设 GAPRole 任务在peripheral.c 和peripheral.h 中定义。有关完整的 API 外设角色 API，包括命令、可配置参数、事件和回调，请参阅BLE 堆栈 API 参考。该模块的使用步骤如下：

1. 初始化 GAPRole 参数。这种初始化应该发生在应用程序初始化函数中。（例如清单 47 `simple_peripheral_init` 中 所示。）。

清单 47. GAP 外设角色的设置¶

```
// 设置 GAP 外设角色配置文件

{

   uint8_t  initialAdvertEnable  =  TRUE ;

   uint16_t  advertOffTime  =  0 ;

   uint8_t  enableUpdateRequest  =  DEFAULT_ENABLE_UPDATE_REQUEST ;
   uint16_t  desiredMinInterval  =  DEFAULT_DESIRED_MIN_CONN_INTERVAL ;
   uint16_t  desiredMaxInterval  =  DEFAULT_DESIRED_MAX_CONN_INTERVAL ;
   uint16_t  desiredSlaveLatency  =  DEFAULT_DESIRED_SLAVE_LATENCY ;
   uint16_t  desiredConnTimeout  =  DEFAULT_DESIRED_CONN_TIMEOUT ;

   // 设置 GAP 角色参数

   GAPRole_setParameter ( GAPROLE_ADVERT_ENABLED ,  sizeof ( uint8_t ),
   & initialAdvertEnable );
   GAPRole_setParameter ( GAPROLE_ADVERT_OFF_TIME ,  sizeof ( uint16_t ),
   & advertOffTime );  GAPRole_setParameter ( GAPROLE_SCAN_RSP_DATA ,
   sizeof ( scanRspData ),  scanRspData );
   GAPRole_setParameter ( GAPROLE_ADVERT_DATA ,  sizeof ( advertData ),
   advertData ); GAPRole_setParameter ( GAPROLE_PARAM_UPDATE_ENABLE ,
   sizeof ( uint8_t ),  & enableUpdateRequest );
   GAPRole_setParameter ( GAPROLE_MIN_CONN_INTERVAL ,
   sizeof ( uint16_t ),  & desiredMinInterval );
   GAPRole_setParameter ( GAPROLE_MAX_CONN_INTERVAL ,
   sizeof ( uint16_t ),  & desiredMaxInterval );
   GAPRole_setParameter ( GAPROLE_SLAVE_LATENCY ,  sizeof ( uint16_t ),
   &所需的SlaveLatency );
   GAPRole_setParameter ( GAPROLE_TIMEOUT_MULTIPLIER ,
   sizeof ( uint16_t ),  & desiredConnTimeout );
}
```

2. 初始化 GAPRole 任务并将应用回调函数传递给 GAPRole。这也应该发生在应用程序初始化函数中。

清单 48.注册回调和初始化。¶

```
// 启动设备
VOID  GAPRole_StartDevice ( & SimpleBLEPeripheral_gapRoleCBs );
```

3. 从应用程序发送 GAPRole 命令。图 42.是使用GAPRole_TerminateConnection()的应用程序示例。

清单 49.终止连接¶

```
GAPRole_TerminateConnection ();
```

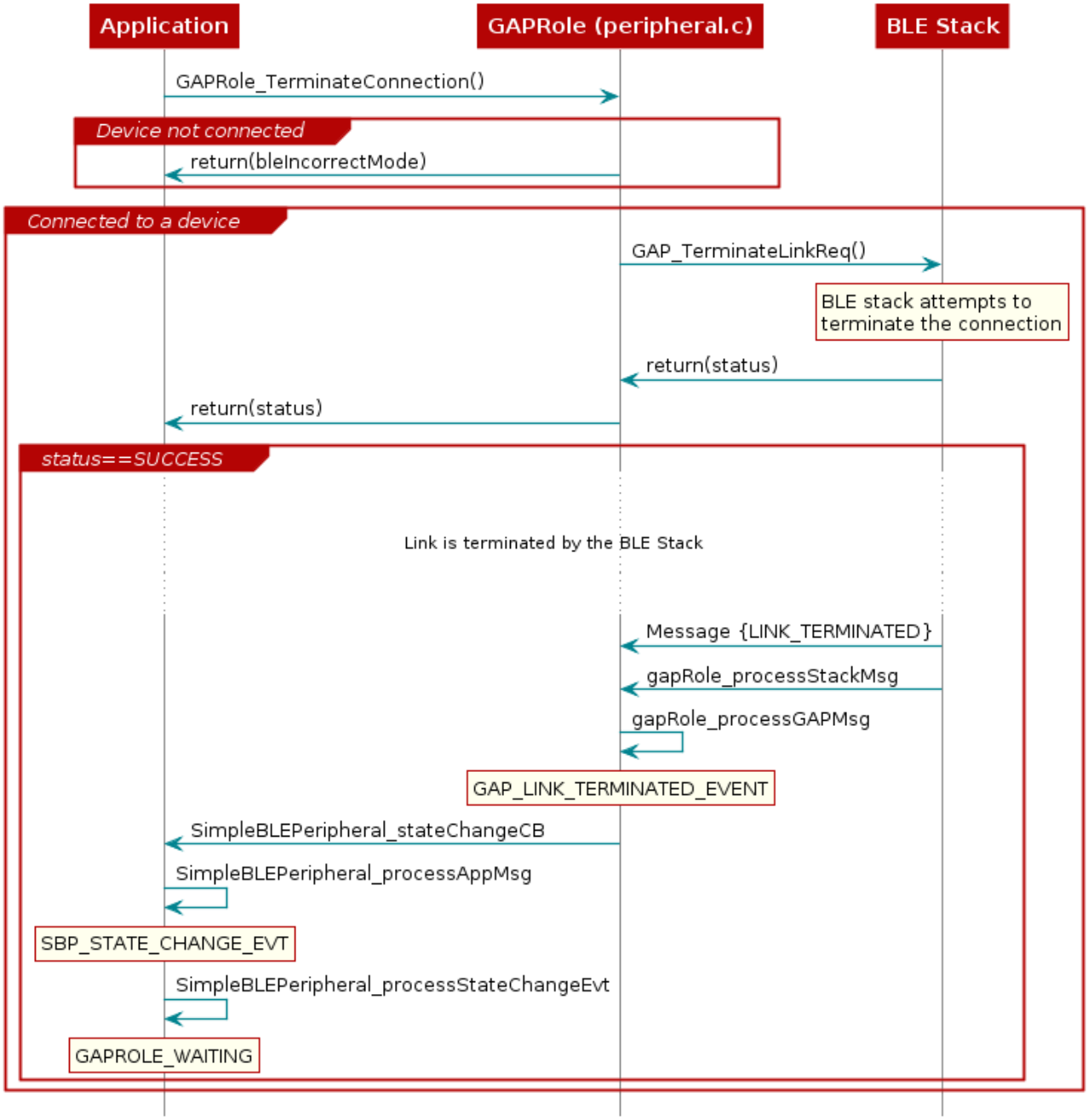

图 42.使用 GAPRole_TerminateConnection() 的应用程序的上下文图。

笔记

返回值仅指示终止连接的尝试是否成功发起。连接事件的实际终止是异步返回的，并通过回调传递给应用程序。

4. GAPRole 任务处理从蓝牙低功耗协议栈传递给它的大部分 GAP 相关事件。GAPRole 任务还将一些事件转发到应用程序。当链接终止时，GAPRole 会自动重新启动广告。可以在 peripheral.c 中找到以下代码片段

清单 50.断开连接时广告重新启动¶

```
    case  GAP_LINK_TERMINATED_EVENT :
      {
        //.......
        //......
        //.......

        // 如果连接断开时设备正在广播
        if  ( gapRole_AdvNonConnEnabled )
        {
          // 继续广播。
          gapRole_state  =  GAPROLE_ADVERTISING_NONCONN ;
        }
        // 否则进入 WAITING 状态。
        else
        {
          if ( pPkt -> reason  ==  LL_SUPERVISION_TIMEOUT_TERM )
          {
            gapRole_state  =  GAPROLE_WAITING_AFTER_TIMEOUT ;
          }
          else
          {
            gapRole_state  =  GAPROLE_WAITING ;
          }

          // 如果启用,则开始广告。
          gapRole_setEvent ( START_ADVERTISING_EVT );
        }
      }
      休息;
```

核心角色

中央 GAPRole 任务在 central.c 和 central.h 中定义。有关完整的中央角色 API，包括命令、可配置参数、事件和回调，请参阅BLE 堆栈 API 参考。有关实现中央 GAPRole 的示例，请参阅 simple_central 示例项目。使用该模块的步骤如下。

1. 初始化 GAPRole 参数。这种初始化应该发生在应用程序初始化函数中（例如在中`SimpleBLECentral_init`）。GAP 参数也可以在这个初始化函数中设置。

```
// 设置中央配置文件
{
   uint8_t  scanRes  =  DEFAULT_MAX_SCAN_RES ;

   GAPCentralRole_SetParameter ( GAPCENTRALROLE_MAX_SCAN_RES ,
   sizeof ( uint8_t ),  & scanRes );
}
```

2. 启动 GAPRole 任务。这涉及将指向应用程序回调函数的函数指针传递给中央 GAPRole。这也应该发生在应用程序初始化函数中。

```
无效 GAPCentralRole_StartDevice ( & SimpleBLECentral_roleCB );
```

3. 从应用程序发送 GAPRole 命令。图 43.是使用GAPCentralRole_StartDiscovery()的应用程序示例。

协议栈的返回状态仅指示是否发起了执行设备发现的尝试。发现的实际设备作为GAP_DEVICE_INFO_EVENT异步返回， 通过中央 GAPRole 回调转发，如下所述。

4. GAPRole 任务对其从协议栈接收的 GAP 事件执行一些处理。该任务还将一些事件转发给应用程序。图 43.还显示了GAP_DEVICE_INFO_EVENT如何 从协议栈处理到应用程序。

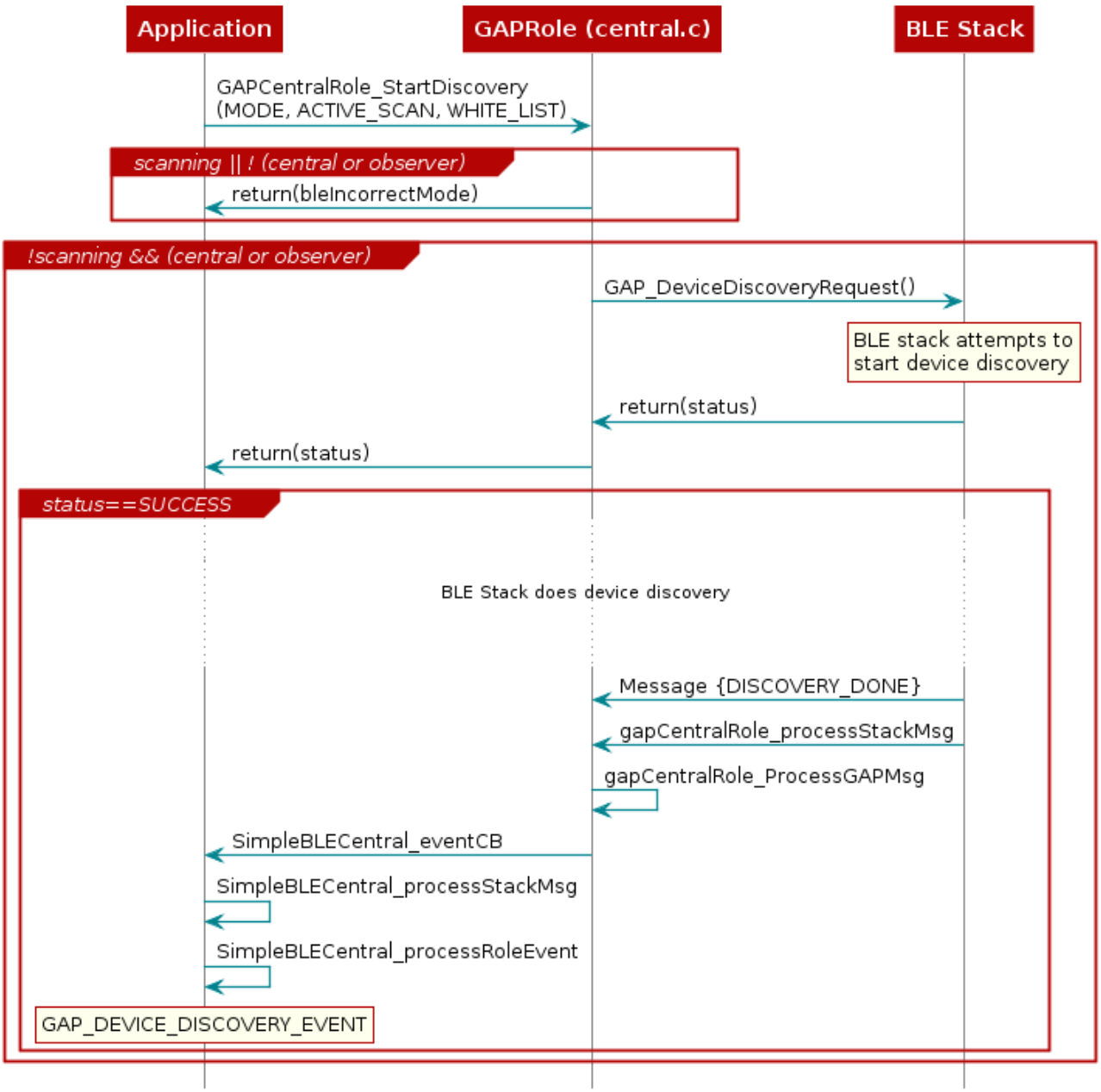

图 43.使用 GAPCentralRole_StartDiscovery() 的应用程序的上下文图。

请注意，在扫描期间，根据蓝牙核心规范版本 5.1 的定义，单个广告/扫描响应将作为GAP_DEVICE_INFO_EVENT返回。默认情况下，重复报告会被过滤，这样每个唯一地址和数据对只会向应用程序返回一个事件。这可以通过TGAP_FILTER_ADV_REPORTS GAP 参数进行配置。扫描完成后，发现的报告摘要将作为GAP_DEVICE_DISCOVERY_EVENT返回给应用程序。

可以使用GAPCENTRALROLE_MAX_SCAN_RES参数设置在一次扫描期间可以发现的最大扫描响应数量。在充满广告/扫描响应的环境中，这可能会对堆使用产生巨大影响，甚至可能破坏堆栈。因此，必须针对最坏情况（在扫描期间发现最大数量的扫描响应）来分析您的应用程序。

在最坏的情况下，在扫描期间发现最大数量的广告/扫描响应 (n)，所有数据大小都最大，应用程序始终如一地处理，因此它不处理任何堆栈的消息，堆可以增长：（8 + 87 * n 个字节）。例如，如果 GAPCENTRALROLE_MAX_SCAN_RES设置为 10，则必须至少有 878 字节可用于从堆分配。这包括完全填充的GAP_DEVICE_DISCOVERY_EVENT。如果此分配失败，则GAP_DEVICE_DISCOVERY_EVENT错误状态将尝试分配，而只有 8 个字节。因此，为了让系统在上述场景中继续运行，堆必须至少有空间分配（8 + 79 * n 个字节）。有关 分析堆的步骤，请参阅调试一章。

发起连接

为了进入与设备的连接，必须发送连接请求。如果找到所需的设备，通常会在扫描完成后执行此操作，但如果设备的地址和地址类型已知，则可以直接连接到设备。用于建立连接的 API 是 GAPCentralRole_EstablishLink()用于中心角色或 GAPRole_EstablishLink()用于多角色。这两个 API 都只是填写正确的结构并调用GAP_EstablishLinkReq()。

发起连接是一个组合操作，它包括

1. 扫描设备
2. 发送连接请求（如果找到设备）

如果没有找到设备，那么链路层将无限期地继续扫描。在启动过程中尝试建立附加连接将返回错误`bleAlreadyInRequestedMode`。

存在这样一种情况，即在一般扫描期间可能检测到广告设备（即通过 找到`GAP_DEVICE_INFO_EVENT`），但在连接启动扫描期间可能未检测到。（例如用户走开，或电池没电了）。出于这个原因，可能需要为在应用程序中启动创建超时。可以使用以下命令停止链路层中的启动连接任务：GAPCentralRole_TerminateLink()或GAPRole_TerminateConnection()连接句柄设置为`GAP_CONNHANDLE_INIT`。