STM32WB 系列 ZigBee 配网指南

引言

本文档旨在描述在 STM32WB 系列微控制器上对 Zigbee®功能进行配网的过程。

该指南涵盖以下主题：

- 集中式网络
- 分布式网络
- ZCL 配网群集
- 预配置启动
- Touchlink 配网
- 查找和绑定过程

这将允许 STM32WB 系列微控制器使用 Zigbee®协议加入所需的网络。

1 概述

本文档适用于 STM32WB 系列基于双核 Arm®的微处理器。

注意： *Arm 是 Arm Limited（或其子公司）在美国和/或其他地区的注册商标。*

1.1 术语表

表 1. **术语表**

术语	定义
APS	应用支持子层
PAN	个人局域网
PANID	个人局域网 ID
InterPAN	设备在不同 Zigbee®网络上相互通信的过程
TC	信任中心
TCLK	跟踪时钟，请参阅*《基于 Arm® Cortex®-M4 无线多协议的 32 位 MCU，配有 FPU、低功耗蓝牙和 802.15.4 无线电解决方案》*（RM0434）
Touchlink	参见第 6 节
ZC	Zigbee®协调器
ZCL	Zigbee®群集库
ZDO	Zigbee®设备对象

1.2 参考文档

表 2. **参考文档**

参考	文档
[1]	Zigbee R22 规范
[2]	基本设备行为规范（13-0402）

2 集中式网络

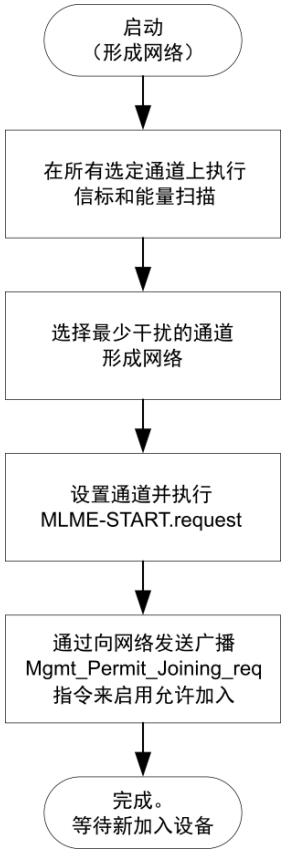
2.1 形成网络

本节必须结合《Zigbee R22 规范》[1]-第 3.6.1.1 节一起阅读。

2.1.1 网络形成流程图

以下流程图在较高层面概述了网络形成过程：

图 1. 描述 Zigbee®网络形成的流程图



2.1.2 网络形成配置

1. Zigbee®启动配置数据结构：

```
struct ZbStartupT config;
```

2. 使用默认配置初始化结构：

```
ZbStartupConfigGetProDefaults(&config);
```

3. 选择以形成新网络：

```
config.startupControl = ZbStartTypeForm;
```

4. 可选地选择要使用的扩展 PANID。如果设置为零，则改用本地设备扩展地址。

```
config.extendedPanId = 0x1234567812345678ULL;
```

5. 配置预配置的链路密钥，或者如果配置信任中心（TC）并使用唯一链路密钥（也称为安装代码），则将该参数设置为 0x00。

```
memcpy(config.security.preconfiguredLinkKey, sec_key_ha, ZB_SEC_KEYSIZE);
```

2.1.3 网络形成启动

调用 Zigbee®协议栈启动函数。提供的回调函数在启动完成后调用，无论启动成功还是失败。

```
ZbStartup(zb, &config, callback, arg);
```

2.1.4 网络允许设备加入

```
struct ZbZdoPermitJoinReqT req;

req.destAddr = ZB_NWK_ADDR_BCAST_ROUTERS;
req.duration = 180;
ZbZdoPermitJoinReq(zb, &req, callback, arg);
```

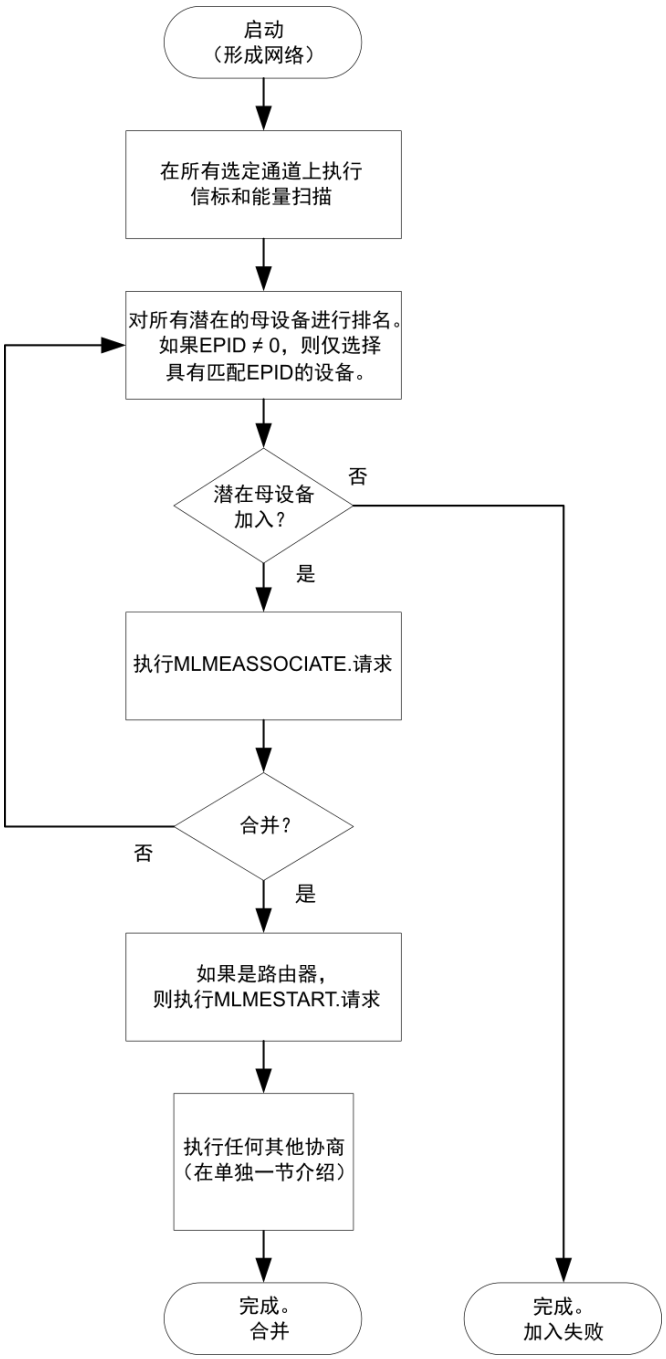
2.2 加入网络

本节必须结合《Zigbee R22 规范》[1]-第 3.6.1.3 节一起阅读。

2.2.1 加入网络流程图

以下流程图在较高层面概述了网络加入过程：

图 2. 描述加入 Zigbee®网络的流程图



2.2.2 加入网络配置

1. Zigbee®启动配置数据结构：

```
struct ZbStartupT config;
```

2. 使用默认配置初始化结构：

```
ZbStartupConfigGetProDefaults(&config);
```

3. 选择加入新网络：

```
config.startupControl = ZbStartTypeJoin;
```

4. 可选地选择要使用的扩展 PANID。如果设置为零，则设备尝试加入任何可用的开放网络：

```
config.extendedPanId = 0ULL;
```

5. 预配置的链路密钥：

```
memcpy(config.security.preconfiguredLinkKey, sec_key_ha, ZB_SEC_KEYSIZE);
```

2.2.3 加入网络启动

调用 Zigbee®协议栈启动函数。提供的回调函数在启动完成后调用，无论启动成功还是失败。

```
ZbStartup(zb, &config, callback, arg);
```

2.3 预配置的链路密钥

本节必须结合《Zigbee R22 规范》[1]-第 4.2.1.2.1 节一起阅读。

在设备加入网络之前，将预配置的链路密钥编程到设备。这是在制造时或应用安装期间完成的。

2.3.1 全局链路密钥

对于所有使用相同链路密钥加入过程的设备，全局链路密钥均有涉及，通常用于解密从包含网络密钥的信任中心（或分布式网络中的母设备）发送的 APS 传输密钥。

例如，在 Zigbee®协议栈测试期间使用的默认链路密钥是：

```
“ZigBeeAlliance09”
```

或：

```
5a:69:67:42:65:65:41:6c:6c:69:61:6e:63:65:30:39
```

使用全局链路密钥的原因在于，所有网络设备都可以在生产时使用相同的密钥进行编程。这意味着安装程序在将设备加入网络时无需执行任何进一步的配置。

2.3.2 安装代码链路密钥

安装代码链路密钥是加入设备的唯一链路密钥。这些密钥仅在具有信任中心的集中式网络中使用。在设备加入网络之前，信任中心会将设备安装代码链路密钥添加到其密钥表中。在加入过程中，当信任中心向加入设备发送 APS 加密的数据包（如包含网络密钥的 APS 传输密钥）时，信任中心使用安装代码链路密钥进行加密。

只有信任中心和加入设备才能解密 APS 数据包有效负载。因此，与使用全局链路密钥相比，安装代码链路密钥可有效提升网络的安全性。如果攻击者能够获取网络全局链路密钥，则可以通过侦听设备加入过程中发送的传输密钥来获取网络密钥。

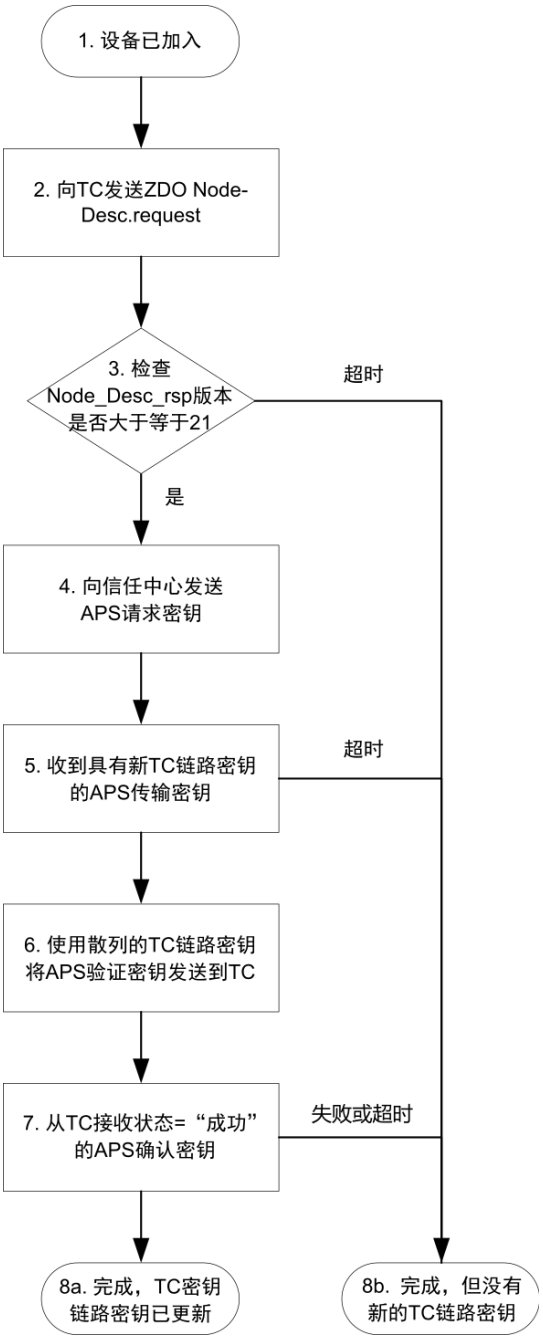
2.4 信任中心链路密钥协商

设备加入集中式网络后，会与信任中心协商唯一链路密钥。该密钥仅在加入设备和信任中心之间共享，因此网络中的其他设备或侦听网络流量的设备无法对使用所协商链路密钥加密的 APS 有效负载进行解密。

2.4.1 信任中心链路密钥协商流程图

下图示出了信任中心链路密钥协商过程。

图 3. 描述信任中心链路密钥协商的流程图



上图所示的步骤详见以下列表：

1. 设备已加入，从 TC 收到网络密钥，并且设备已发送 ZDO Device- Annce 消息。
2. 设备向 TC 发送 ZDO Node-Descriptor.request，以查询 TC 协议栈版本号。

3. 如果收到 ZDO Node-Descriptor.response 并指示协议栈版本至少为 21，则 TC 支持 TCLK 协商。
4. 设备向 TC 发送 APS 请求密钥。收到后，TC 生成新的链路密钥，以便在 TC 和设备之间共享。TC 使用新的链路密钥生成 APS 传输密钥指令，并将其发送到设备。
5. 设备接收新的 TC 链路密钥并更新其密钥表。
6. 设备生成包含新 TC 链路密钥哈希的 APS 验证密钥指令，并将其发送到 TC。收到 APS 验证密钥后，TC 验证散列密钥，并向设备发送 APS 确认密钥指令。
7. 设备接收 APS 确认密钥并检查状态。
8. 该过程存在两种可能的结果：
   a. 调用 ZbStartup 回调，向应用指示加入流程成功。
   b. 调用 ZbStartup 回调，向应用指示加入流程失败。

2.5 集中式网络 API

默认在协议栈中启用信任中心链路密钥协商。然而，以下是在调用 ZbStartup 之前，对其启用/禁用的步骤。下列代码给出了 API 样例。

```
/* Enable or disable the Trust Center Link Key Negotiation */
uint8_t val = 1;
ZbBdbSet(zb, ZB_BDB_TrustCenterRequiresKeyExchange, &val, sizeof(val));

/* Control the Trust Center Link Key Method.
* Set to 0x00 for standard APS Trust Center Link Key negotiation.
* Set to 0x01 for Smart Energy Certificate Based Key Exchange (CBKE) */
uint8_t val = BDB_LINKKEY_EXCHANGE_METHOD_APS;
ZbBdbSet(zb, ZB_BDB_TCLinkKeyExchangeMethod, &val, sizeof(val));
```

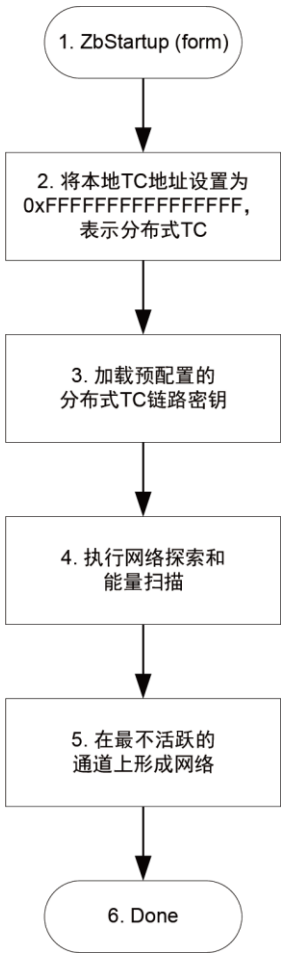
3 分布式网络

分布式网络是没有集中协调器或信任中心的网络。网络中的每个路由器都能够在 MAC 关联后为加入设备提供网络密钥。APS 流量使用已知的全局链路密钥进行加密，该全局链路密钥是所有设备使用芯片制造商设置或在配网期间进行预编程的密钥。

3.1 分布式网络流程图

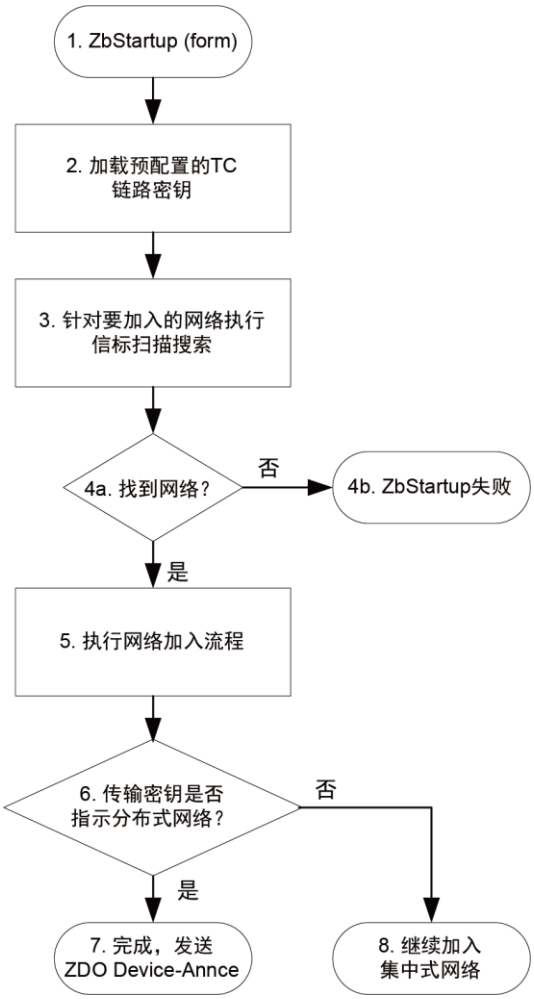
下图示出了 Zigbee®分布式网络的创建。

图 4. 流程图描述了形成 Zigbee®分布式网络的设备



形成分布式网络的流程与形成集中式网络的相同，区别在于信任中心地址在 ZbStartup 配置中被配置为 0xFFFFFFFFFFFFFFFF。此外还设置了预配置的分布式链路密钥，这是所有设备加入网络时使用的全局链路密钥。由于没有集中式协调器（或 TC），所有路由器母设备都充当信任中心，并且在设备加入后发送使用分布式链路密钥加密的传输密钥，该传输密钥包含网络密钥。Zigbee®分布式网络的加入过程如下图所示。

图 5. 描述设备加入 Zigbee®分布式网络的流程图



1. 应用调用 ZclStatusCodeT 以加入新网络。
2. 向协议栈中添加提供给 ZclStatusCodeT 配置的预配置分布式链路密钥。它将用于解密母设备发送的传输密钥。
3. 设备扫描通道，查找要加入的网络。
4. 如果找到网络，则继续正常的 Zigbee®加入流程，否则启动失败（步骤 4b）
5. 设备尝试加入网络。
6. 加入后，传输密钥的扩展源地址必须为 0xFFFFFFFFFFFFFFFF，表示设备已加入分布式 TC 网络。
   – 如果没有，则继续正常的 Zigbee®加入流程（步骤 8）。
   – 如果有，则设备已加入网络，并且具有在网络上通信所需的网络密钥。
7. 加入完成。向网络发送 ZDO Device-Annce。

   否则
8. 继续连接集中式网络。

3.2 分布式网络 API

在形成或加入分布式网络时，配置以下 ZbStartupT 参数。

3.2.1 分布式网络信任中心地址

```
/* Configure the Trust Center to be distributed (0xffffffffffffffff) */
config.security.trustCenterAddress = ZB_DISTRIBUTED_TC_ADDR;
```

3.2.2 分布式网络预配置的分布式链路密钥

```
/*为网络配置“预配置的分布式链路密钥”。
*对于测试，应使用“未经认证的分布式密钥”，即
* d0:d1:d2:d3:d4:d5:d6:d7:d8:d9:da:db:dc:dd:de:df */
memcpy(config.security.distributedGlobalKey, sec_key_distrib_uncert,ZB_SEC_KEYSIZE);
```

3.2.3 分布式网络启动

调用 Zigbee®协议栈启动函数。提供的回调函数将在启动完成后调用，无论启动成功还是失败。

```
ZbStartup(zb, &config, callback, arg);
```

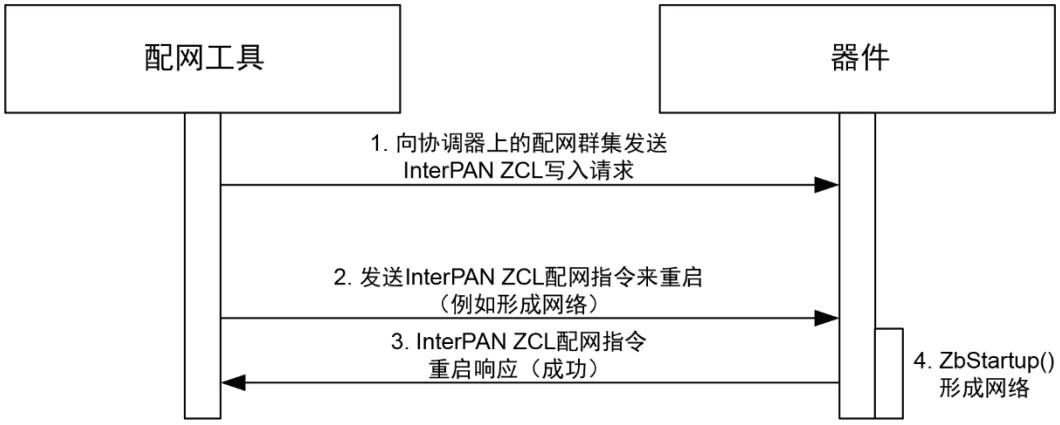
4 ZCL 配网群集

配网群集用于在安装过程中对设备进行网络配置。打开加入设备，然后使用配网工具传输形成或加入网络所需的启动参数。为加入设备配置参数后，配网工具向设备发送指令，以启动形成或加入过程。

4.1 InterPAN 配网顺序图

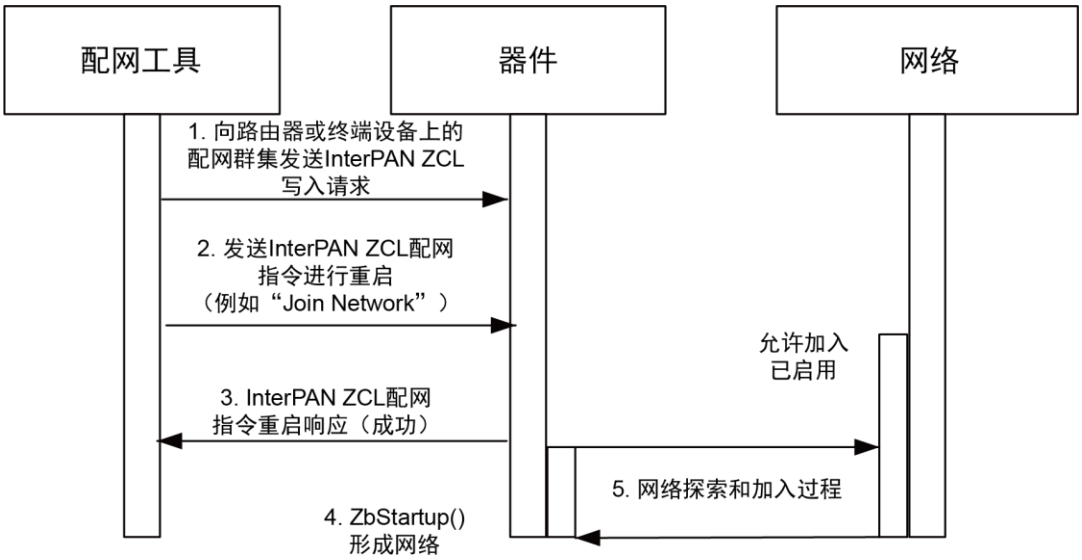
下图概述了 InterPAN 网络创建过程，其中包括必要的消息。

图 6. InterPAN 配网消息



1. 配网工具向要配网的设备发送“ZCL Write Attribute”指令，以便配置“Zigbee®启动属性集”。其中一个属性将 StartupControl 设置为形成网络。
2. 配网工具发送指令以使用“当前启动参数集”重新启动设备。
3. 设备使用“Restart Response”指令和 SUCCESS 状态进行响应。
4. 设备执行“Network Formation”流程并形成新网络。

图 7. InterPAN 配网消息：加入网络

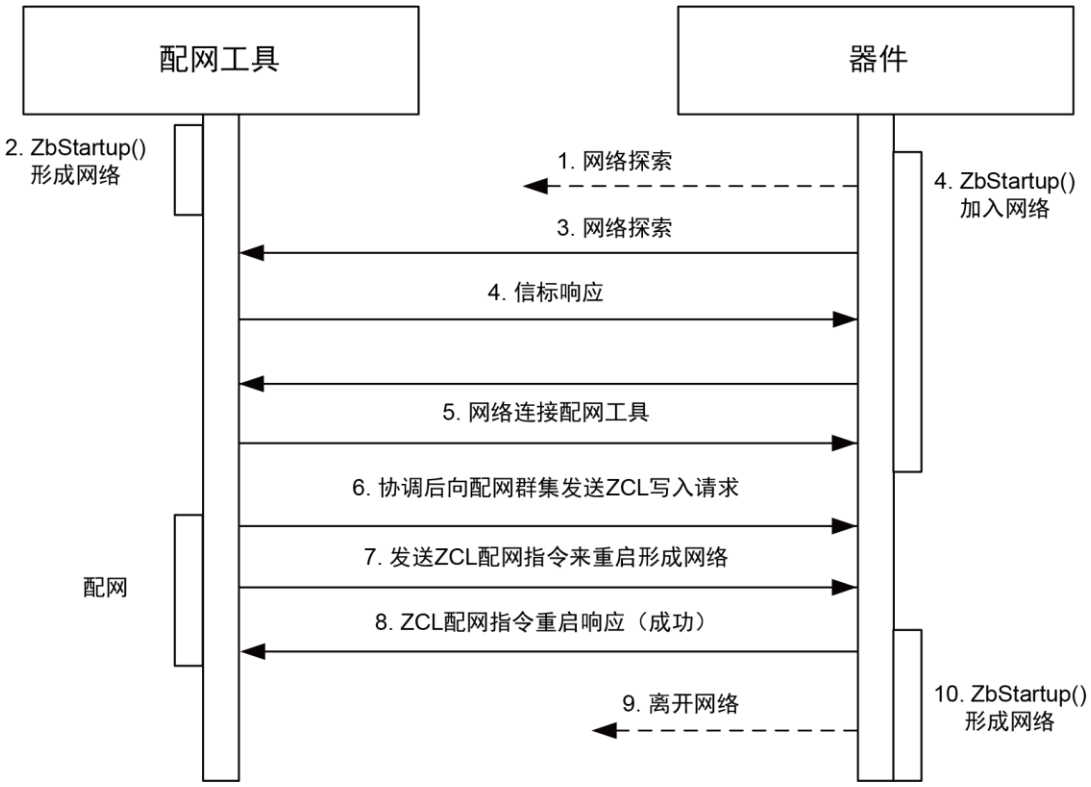


步骤（1-3）与配网工具配置设备以形成网络的步骤相同，但在本例中，将 StartupControl 设置为加入网络。设备收到重新启动请求指令后，将启动“Network Join”流程（步骤 4-5）。

4.2 临时网络配网序列图

设备能够临时加入网络。该过程如下图所示。

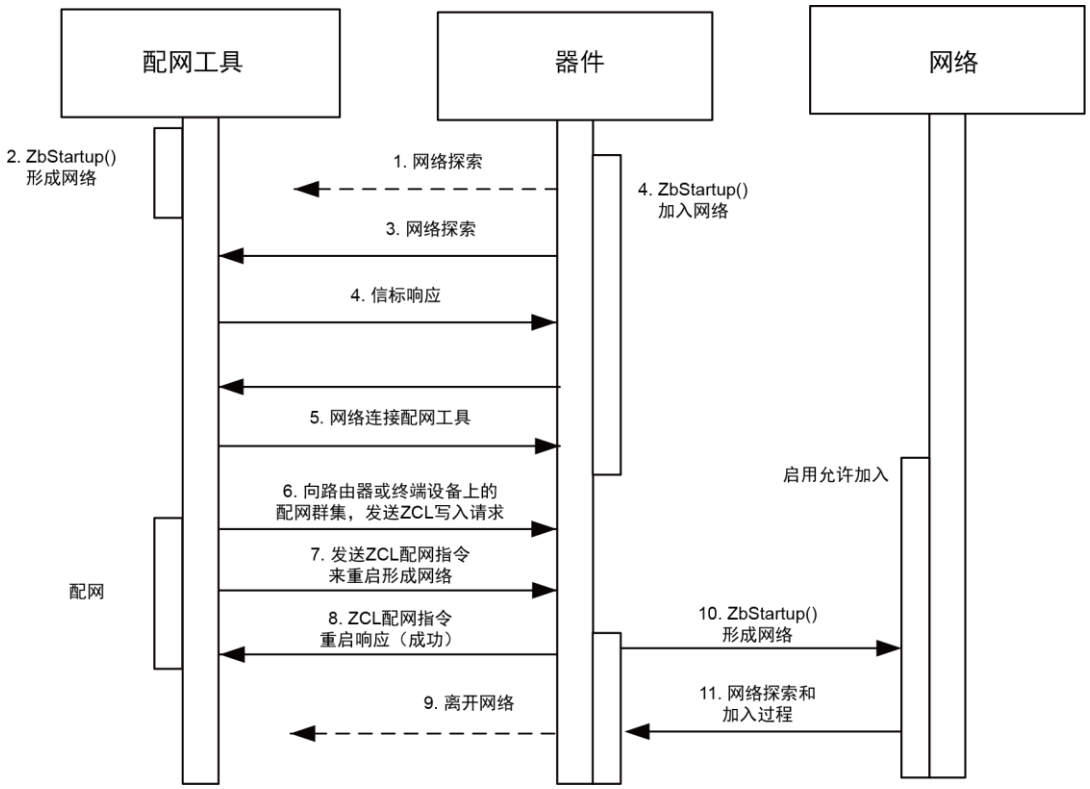
图 8. 配网工具网络形成：告知设备形成新的网络



配网工具的定义详见下文。

1. 设备正在尝试自动或通过用户干预（按钮）加入配网工具网络。此时，尚未形成配网工具网络。
2. 配网工具形成临时网络，供设备加入。要使用的默认 EPID 为 00-50-C2-77-10-00-00-00，因此任何设备都能够从附近的任何其他网络中筛选出配网工具网络。
3. 设备尝试再次发现配网工具网络。
4. 配网工具以信标响应。
5. 设备加入配网工具网络。
6. 配网工具向要配网的设备发送 ZCL 写入属性指令，以便配置 Zigbee®启动属性集。其中一个属性将 StartupControl 设置为形成网络。
7. 配网工具向设备发送使用“当前启动参数集”重新启动的指令。
8. 设备使用重新启动响应指令和“SUCCESS”状态进行响应。
9. 设备离开配网工具网络。
10. 设备执行网络形成流程并形成新网络。

图 9. 配网工具形成网络：告知设备加入所需的网络



步骤（1-7）与配网工具配置设备以形成网络的步骤相同，但在本例中，将 StartupControl 设置为加入网络。设备收到重新启动请求指令后，将启动网络加入流程（步骤 8-10）。

4.3 设备配网服务器

配网服务器由正在配网的设备使用。以下代码提取介绍了如何对 ZCL 配网服务器群集进行分配和实例化。

```
struct ZbZclClusterT *cluster;
struct ZbZclCommissionServerCallbacksT callbacks;

/*为收到的“配网”配置配网群集函数回调处理程序
客户端指令。*/
memset(&callbacks, 0, sizeof(callbacks));
callbacks.restart_device = app_commission_svr_restart_cb;
callbacks.save_startup = app_commission_svr_save_cb;
callbacks.restore_startup = app_commission_svr_restore_cb;
callbacks.reset_startup = app_commission_svr_reset_cb;

cluster = ZbZclCommissionServerAlloc(zb, ZCL_PROFILE_HOME_AUTOMATION,
      true /* APS 是否安全？ */, &callbacks, arg);
```

4.3.1 InterPAN

InterPAN 使配网服务器能够在选定的通道上运行，并从配网工具接收 InterPAN 消息，而无需加入网络。InterPAN 通常不安全。尽管能够发送 APS 安全的 InterPAN 消息，但需要特殊配置。有关发送安全配网工具消息的更常用方法，请参见第 4.3.2 节，其中对加入配网工具网络进行了初步的描述。

```
struct ZbZclCommissionServerEnableInfoT info;

memset(&info, 0, sizeof(info));
info.page = 0;
info.channel = 11;
ZbZclCommissionServerEnable(cluster, &info);
```

设备现在已准备好从配网工具接收配置和启动指令。

4.3.2 加入配网工具网络

配网工具为要配网的设备创建供加入的临时网络，用于直接传输配网消息，而无需使用 InterPAN。网络应使用 Zigbee®定义的扩展 PAN ID：00-50-C2-77-10-00-00-00，以加快设备执行网络探索的速度。通过加入配网工具网络，还简化了 APS 安全性的使用。设备使用已知的预配置的链路密钥加入配网工具，允许使用 APS 安全性传输配网消息，具体取决于应用制造商提出的要求。

4.4 配网客户端

配网工具使用配网客户端来执行配网。以下代码提取介绍了如何对 ZCL 配网客户端群集进行分配和实例化。

```
struct ZbZclClusterT *cluster;
cluster = ZbZclCommissionClientAlloc(zb, ZCL_PROFILE_HOME_AUTOMATION,
    true /* APS 是否安全？ */, &callbacks, arg);
```

4.4.1 InterPAN 通信

使配网客户端能够在所选通道上运行，并将 InterPAN 消息发送到正在配网的设备。该函数在以下代码提取中给出

```
struct ZbZclCommissionClientEnableInfoT info;
memset(&info, 0, sizeof(info));
info.page = 0;
info.channel = 11;
ZbZclCommissionClientEnable(cluster, &info);
```

4.4.2 形成配网工具网络

配网工具形成一个网络，供被配网的设备加入。这可以将配网消息作为单播发送，而不是作为 InterPAN 发送。网络必须使用 Zigbee®定义的扩展 PAN ID 00-50-C2-77-10-00-00-00，从而使设备在执行网络探索时轻松找到网络。有关样例，请参阅以下代码提取。

```
struct ZbStartupT config;

/* Initialize the startup parameters */
ZbStartupConfigGetProDefaults(&config);

/* Form network with EPID = 0x0050c27710000000 */
config.startupControl = ZbStartTypeForm;
config.extendedPanId = ZB_EPID_GLOBAL_COMMISSIONING;

/* Configure the Commissioning Network preconfigured link key.
* In this example, for testing purposes, the ZigBeeAlliance09 key is used. */
memcpy(config.security.preconfiguredLinkKey, sec_key_ha, ZB_SEC_KEYSIZE);

/* Configure the channel list */
config.channelList.count = 1;
config.channelList.list[0].page = 0; /* Page 0 */
config.channelList.list[0].channelMask = 0x02108800; /* Channels: 11, 15, 20, 25 */

/* Call ZbStartup to form the network on one of the chosen channels. */
ZbStartup(zb, &config, callback, arg);
```

配网工具成功形成网络后，将启用“网络允许加入”（例如 ZbNlmePermitJoinReq）。

然后，要配网的设备执行 ZbStartup，将 startupControl 设置为 ZbStartTypeJoin，并将扩展的 PanId 设置为 00-50-C2-77-10-00-00-00，以确保该设备只尝试加入配网工具网络。

4.4.3 配网消息

写入属性

要将配网参数写入正在配网的设备，请使用 ZbZclWriteReq() API。使用 InterPAN 或直接发送 ZCL 写入指令，如前所述。下面给出了伪代码样例。

```
static void
write_rsp_cb(const struct ZbZclWriteRspT *writeResp, void *cb_arg)
{
	/* Handle the response. Check the return status. */
}

main()
{
	enum ZclStatusCodeT status;
	struct ZbZclWriteReqT req;
	uint8_t uint8_val;

	memset(&write_req, 0, sizeof(write_req));

	/* Destination */
	write_req.dst.mode = ZB_APSDE_ADDRMODE_EXT;
	write_req.dst.extAddr = DUT_EXT_ADDR;
	/* If using InterPAN, endpoint is set to ZB_ENDPOINT_INTERPAN.
	* Otherwise, set the endpoint to the remote Commissioning
	* Server endpoint Id. */
	write_req.dst.endpoint = ZB_ENDPOINT_INTERPAN;
	/* Attribute to write */
	uint8_val = ZbStartTypeJoin;
	write_req.count = 1;
	write_req.attr[0].attrId = ZCL_COMMISSION_SVR_ATTR_STARTUPCONTROL;
	write_req.attr[0].type = ZCL_DATATYPE_ENUMERATION_8BIT;
	write_req.attr[0].value = &uint8_val;
	write_req.attr[0].length = 1;

	/* Send command */
	status = ZbZclWriteReq(cluster, req, write_rsp_cb, NULL);
	if (status != ZCL_STATUS_SUCCESS) {
		/* We were not able to send this command (malformed?) */
}
```

发送“配网保存”、“恢复”和“复位”指令

要发送“配网保存”启动参数指令（该指令将告知接收器保存选定的启动配置），应用将调用 ZbZclCommissionClientSendRestoreStartup() API。下面给出了伪代码样例。

```
static void
save_rsp_cb(struct ZbZclCommandRspT *rsp, void *arg)
{
	/* Handle the response. Check the return status. */
}

main()
{
	struct ZbZclClusterT *commiss_client; /* Commissioning Client cluster */
	uint64_t dst_eui; /* EUI of device to commission */
	uint8_t dst_ep; /* Destination endpoint, if not using InterPAN */
	struct ZbZclCommissionClientSaveStartup req; /* command configuration */
	enum ZclStatusCodeT status;

	memset(&req, 0, sizeof(req));
	req.index = 0; /* Index of attribute set to save */

	/* Send the command */
	status = ZbZclCommissionClientSendSaveStartup(commiss_client,
		dst_eui, dst_ep, save_rsp_cb, NULL);
	if (status != ZCL_STATUS_SUCCESS) {
		/* We were not able to send this command (malformed?) */
	}
}
```

以类似的方式配置和发送“恢复”和“复位”指令。

“恢复”指令使用 ZbZclCommissionClientSendRestoreStartup() API 发送。该指令将恢复之前保存的选定的启动配置。

“复位”指令使用 Z*bZclCommissionClientSendResetStartup() API 发送。该指令会将选定的或所有的启动配置复位到出厂默认设置。

发送“配网重新启动”指令

要发送“配网重新启动”启动参数指令（该指令告知接收器执行实际的 Zigbee®启动流程），应用调用 ZbZclCommissionClientSendRestart() API。下面给出了伪代码样例。

```
static void
restart_rsp_cb(struct ZbZclCommandRspT *rsp, void *arg)
{
	/* Handle the response. Check the return status. */
}

main()
{
	struct ZbZclClusterT *commiss_client; /* Commissioning Client cluster */
	uint64_t dst_eui; /* EUI of device to commission */
	uint8_t dst_ep; /* Destination endpoint, if not using InterPAN */
	struct ZbZclCommissionClientRestartDev req; /* command configuration */
	enum ZclStatusCodeT status;

	memset(&req, 0, sizeof(req));
	/* Configure the startup options. In this case, tell the receiver device
	* to startup immediately */
	req.options |= ZCL_COMMISS_RESTART_OPTS_IMMEDIATE;
	/* dst_ext is the Extended Address of the device being commissioned */
	/* rsp_callback is the function to be called upon reception of the
	* Commissioning Restart Response, or possible error. */
	ZbZclCommissionClientSendRestart(commiss_client, dst_ext, &req,
		restart_rsp_cb, NULL);
	if (status != ZCL_STATUS_SUCCESS) {
		/* We were not able to send this command (malformed?) */
	}
}
```

5 预配置启动

预配置的启动用于使设备重新加入网络，就像设备之前已加入一样。

5.1 配置

5.1.1 PANID

```
config.panId = PANID_OF_NETWORK;
```

5.1.2 简短地址

```
config.shortAddress = DESIRED_SHORT_ADDRESS;
```

5.1.3 扩展 PANID

```
config.extendedPanId = EXTENDED_PANID_OF_NETWORK;
```

5.2 ZCL 配网群集 API

5.2.1 启动

调用 Zigbee®协议栈启动函数。提供的回调函数在启动完成后调用，无论启动成功还是失败。

```
ZbStartup(zb, &config, callback, arg);
```

6 Touchlink 配网

Touchlink 是一种配网机制，其中节点仅在设备彼此靠近时才加入。配网消息最初通过未加密的 InterPAN 发送，并形成或加入分布式网络以进行 Zigbee®通信。Touchlink 仅支持分布式网络，不支持集中式网络。

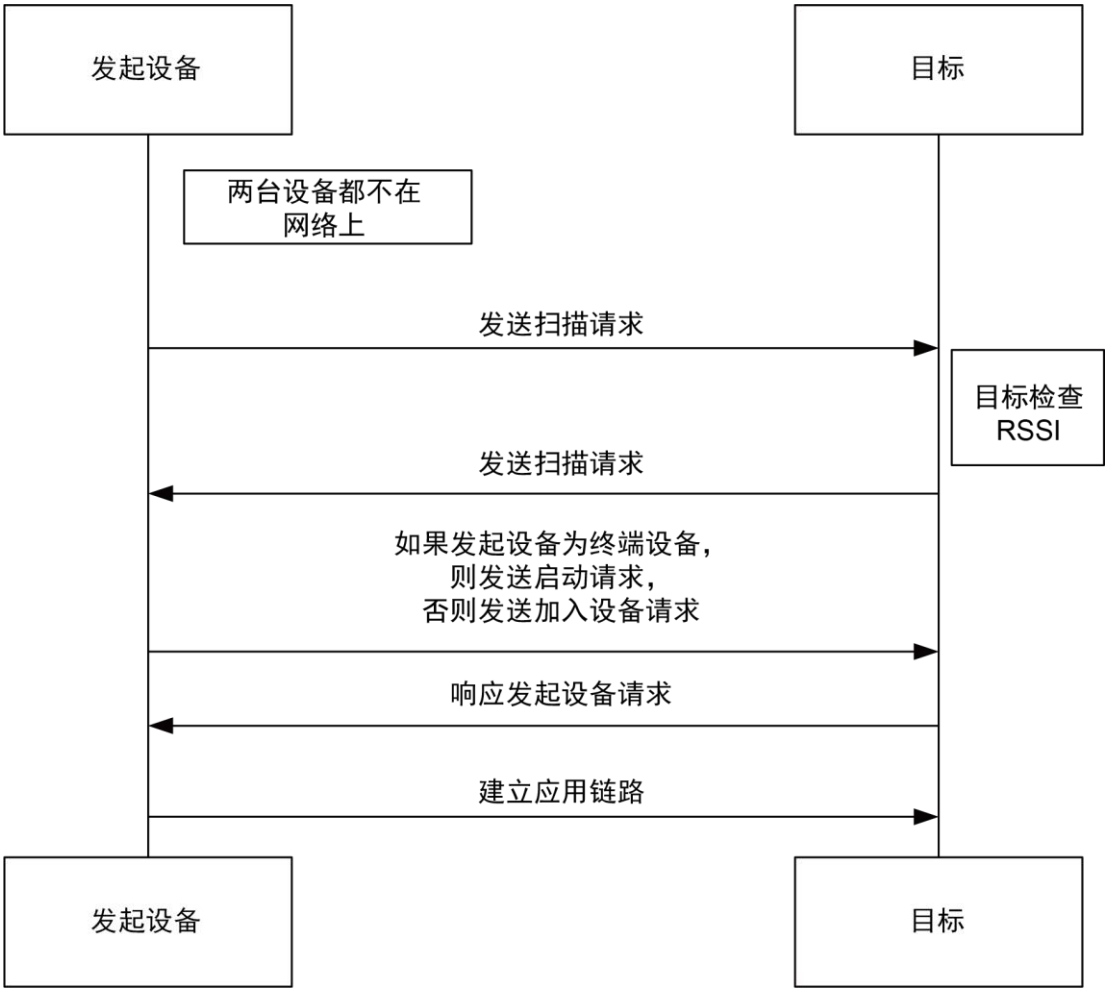
6.1 流程图

本节必须与《基本设备行为规范》（13-0402）[2] - 第 8.7 和 8.8 节一起阅读。

6.1.1 形成网络

要形成新的 Touchlink 网络，设备不得是任何其他网络的组成部分。该过程如下图所示。

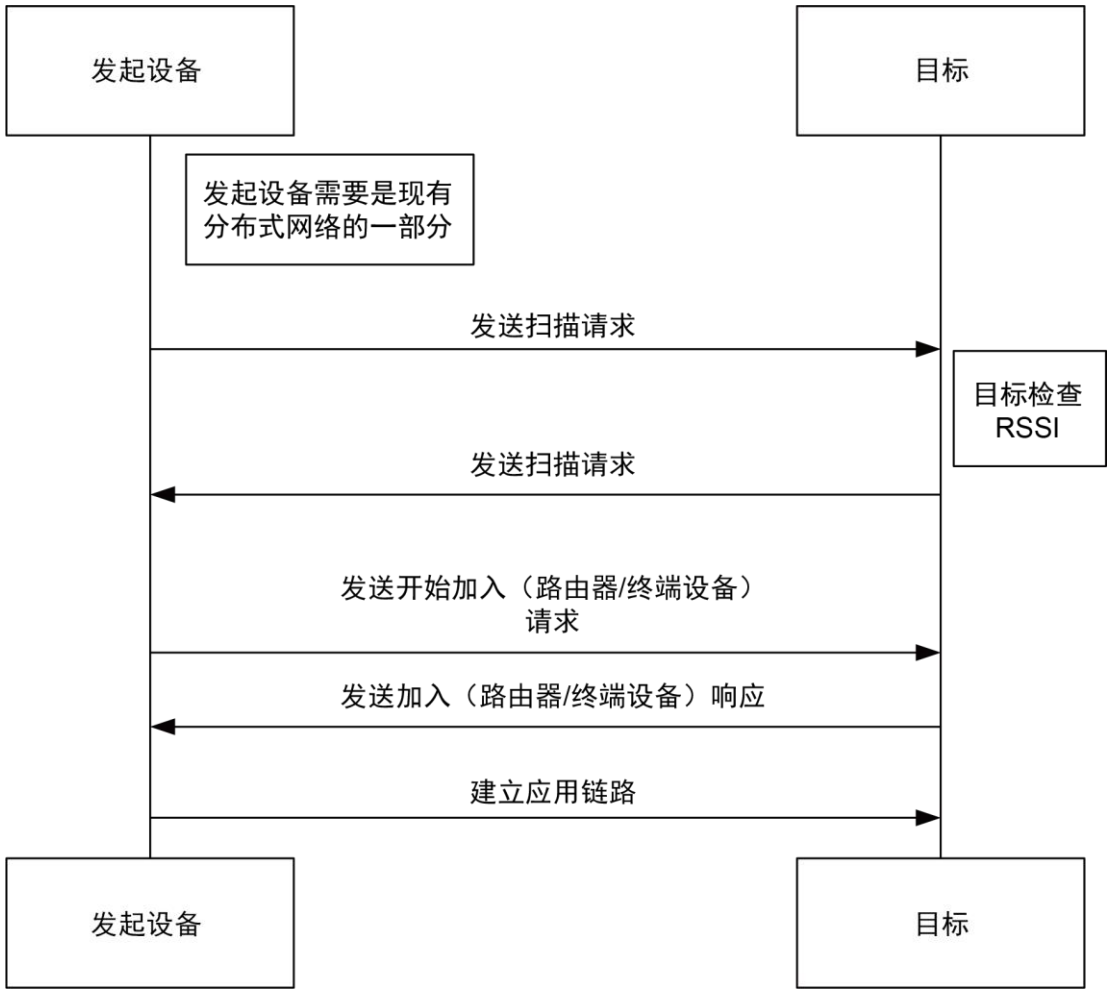
图 10. 形成网络



6.1.2 加入后续设备

下图概述了设备加入现有网络的过程。

图 11. 加入后续设备



注意： *发起设备如果不在网络上并且是路由器，则将查找新的分布式网络。加入的任何其他新设备都必须作为目标加入。*

6.2 配置

在执行 Touchlink 时配置以下 ZbStartupT 参数。

6.2.1 BDB 配网模式

```
/* 为 Touchlink (0x1)配置配网模式 */
config.bdbCommissioningMode |= BDB_COMMISSION_MODE_TOUCHLINK;
```

6.2.2 配置 Touchlink 端点

```
/* 为 Touchlink 群集配置端点 */
config.touchlink.tl_endpoint = TOUCHLINK_ENDPOINT;
```

6.2.3 配置应用端点

```
/* 为应用绑定配置端点 */
config.touchlink.bind_endpoint = APPLICATION_ENDPOINT;
```

6.2.4 配置为 Touchlink 发起设备

```
config.touchlink.flags = 0;
```

6.2.5 配置为 Touchlink 目标

```
config.touchlink.flags = ZCL_TL_FLAGS_IS_TARGET;
```

6.2.6 配置为路由器

```
config.touchlink.zb_info = ZCL_TL_ZBINFO_TYPE_ROUTER;
config.touchlink.zb_info |= ZCL_TL_ZBINFO_RX_ON_IDLE;
```

6.2.7 配置为终端设备

```
config.touchlink.zb_info |= ZCL_TL_ZBINFO_TYPE_END_DEVICE;
```

6.2.8 更改 Touchlink 密钥

```
/*使用“未经认证的分布式全局密钥”
(d0:d1:d2:d3:d4:d5:d6:d7:d8:d9:da:db:dc:dd:de:df).
    *该密钥可在 Touchlink 设备之间作为 APS 链路密钥使用。*/
memcpy(config.security.distributedGlobalKey, sec_key_distrib_uncert,
ZB_SEC_KEYSIZE);

/*使用“Touchlink 认证密钥”
(c0:c1:c2:c3:c4:c5:c6:c7:c8:c9:ca:cb:cc:cd:ce:cf).
    *该密钥与 Touchlink 会话数据“捣碎”以生成网络
密钥 */
ZbBdbSet(zigbee_demo_info.zb, ZB_BDB_TLKey, sec_key_touchlink_cert,
ZB_SEC_KEYSIZE);
{
    /* 将“密钥加密算法”设置为认证 */
    enum ZbBdbTouchlinkKeyIndexT keyIdx = TOUCHLINK_KEY_INDEX_CERTIFICATION;
    ZbBdbSet(zigbee_demo_info.zb, ZB_BDB_TLKeyIndex, &keyIdx, sizeof(keyIdx));
}
```

6.3 Touchlink 配网 API

6.3.1 Touchlink 配网启动

调用 Zigbee®协议栈启动函数。ZbStartup()函数中提供的回调函数在启动完成后调用，无论是启动成功还是失败。

```
ZbStartup(zb, &config, callback, arg);
```

6.3.2 Touchlink 复位

必须将设备设置为发起设备，而且必须设置以下标志：

```
config.touchlink.flags |= ZCL_TL_FLAGS_FACTORY_RESET
```

然后再次调用 ZbStartup。

6.3.3 Touchlink 网络更新

```
zcl_touchlink_nwk_update_req(zb, extaddr, update_id)
```

7 查找和绑定

“查找和绑定”是在匹配的输入和输出群集之间自动建立群集绑定的过程。群集绑定使群集可以相互通信，而无需应用记住目标群集地址。“查找和绑定”依赖于 ZCL 识别群集。ZCL 识别群集必须与需要绑定的群集位于同一端点上。

7.1 自动启动“查找和绑定”过程

如要在加入流程期间启用群集和终结点的自动“查找和绑定”，应用必须在 struct ZbStartupT 启动属性集的 bdbCommissioningMode 参数中启用 BDB_COMMISSION_MODE_FIND_BIND 位。

当任何设备想要加入网络时，服务群集设备必须首先启用“识别服务器”群集，以便进行“查找和绑定”过程。“识别服务器”群集用于启动设备识别过程。

新设备一旦成功加入网络，必须执行以下步骤：

1. 在端点 0xFE（254）上分配“识别客户端”群集（如果尚不存在）。
2. 使用识别客户端，向本地设备上的任何已识别服务器群集发送环回 ZCL 写入请求，以启动识别过程。该过程在接下来的 10 秒内可用。
3. 在本地终结点中搜索任何客户端群集。如果未找到，则终止“查找和绑定”。
4. 从已识别的客户端群集中，向网络广播识别查询指令。记录响应，包括网络地址和源端点。如果未收到响应，则终止“查找和绑定”。
5. 通过遍历识别查询响应，将 ZDO 简单描述符请求发送到与所识别的查询响应的网络地址和端点相匹配的远程设备。如果该远程设备的扩展地址未知，则在发送简单描述符请求之前，首先向远程设备发送 ZDO IEEE 地址请求。需要远程设备的扩展地址，以便在设备之间执行任何绑定。
6. ZDO 简单描述符响应用于将本地设备上的任何群集与要绑定的远程设备进行匹配。
7. 有三个特殊群集：
   - 报警客户端群集
   - 消息群集
   - 轮询控制群集

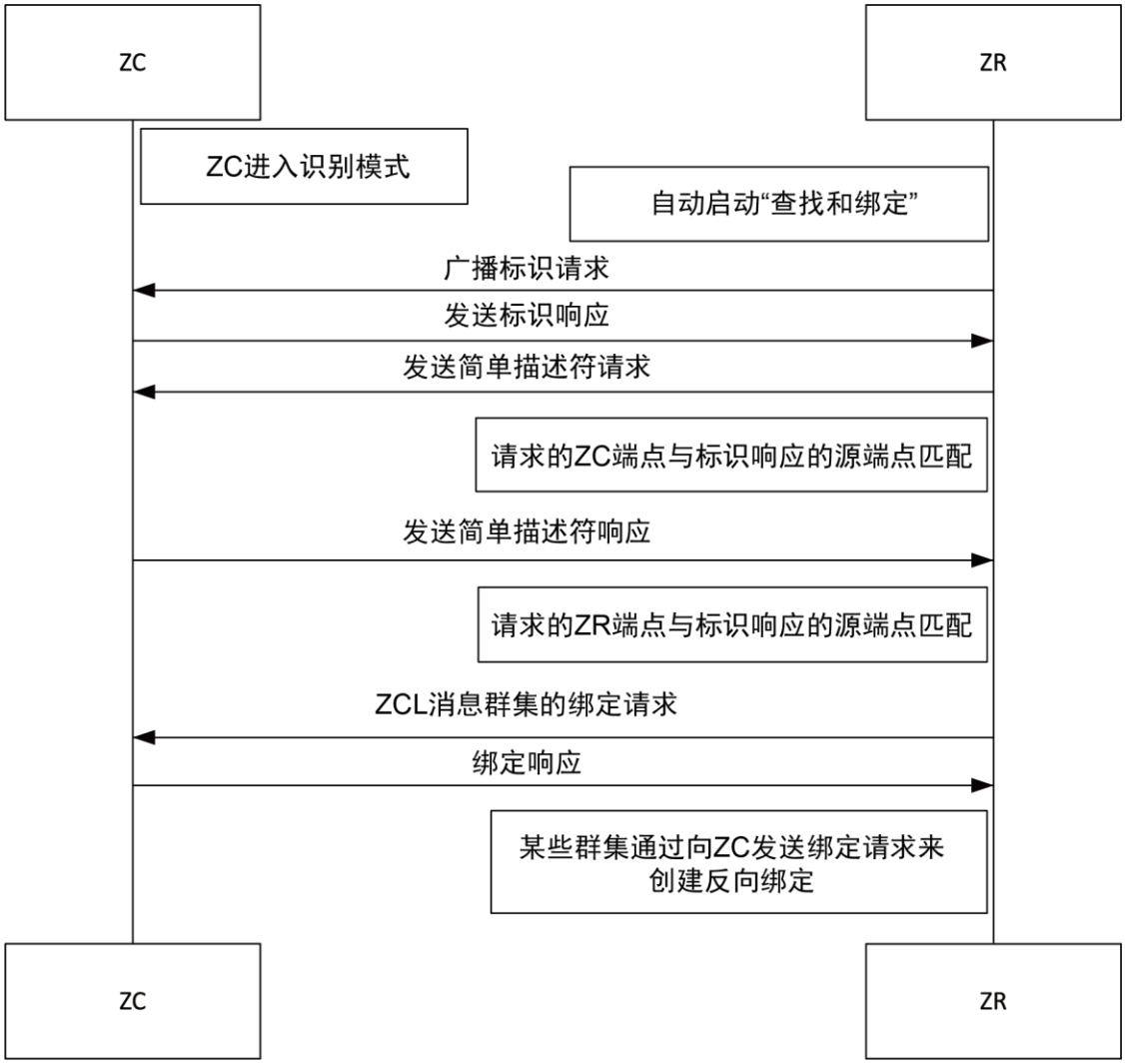
   从服务器到客户端，绑定以相反的方向完成。如果本地端点包括这些群集（作为客户端），则除了正常绑定外，本地端点还会向远程设备发送 ZDO 绑定请求，以便创建从服务器到客户端的绑定。
8. 该过程在包含客户端群集本地设备上的下一个端点重复。

7.2 手动启动查找和绑定

“查找和绑定”过程可以在网络上随时发起。对于具有服务器群集的、安装程序希望进行“查找和绑定”的任何远程设备，必须首先启用“识别服务器”群集，以便在设备上启动识别过程。然后触发本地设备应用来调用 ZbStartupFindBindStart，以便启动“查找和绑定”过程。这些过程步骤与第 7.1 节中列出的步骤相同。

7.3 查找和绑定流程图

图 12. ZR 的查找和绑定图



版本历史

表 3. 文档版本历史

日期	版本	变更
2021 年 7 月 2 日	1	初始版本。

目录

图片目录

表格索引