Z-Stack
应用程序接口

文件编号：F8W-2006-0021

德州仪器有限公司
美国加利福尼亚州圣迭戈
(619)542-1200

版本	描述	日期
1.0	首次发布。	12/11/2006
1.1	增加 ZDO 设备网络启动内容。	03/07/2007

目录

1、引言

1.1 目的

此文件介绍了 ZigBee2006 兼容 Z 协议栈 1.4.0 版本中的 Z 协议栈组件的应用程序编程接口（API）。

1.2 范围

此文件列举了 ZigBee2006 规范兼容 Z 协议栈 1.4.0 版本中所有组件的 API 接口。有关各个接口的详细信息，包括数据结构和函数调用已做出详细说明，以便于程序员理解和在开发过程中使用。API 接口从上层（应用层）至最底层依次介绍。

1.3 缩写词语

AF	应用构架层
AIB	APS 信息库
API	应用程序编程接口
APS	应用程序支持子层
APSDE	APS 数据实体
APSME	APS 管理实体
ASDU	APS 服务数据包单元
MSG	信息
NHLE	上层实体
NWK	网络层
PAN	个人区域网络
STAR	一种含有一个主设备和若干从设备的网络拓扑结构
ZDO	ZigBee 设备对象

2、层概述

本节概述了此文件中所包含的层。

2.1 ZDO 层

ZigBee 设备对象层(ZDO)提供了管理一个 ZigBee 设备的功能。ZDO 层的 API 为应用程序的端点提供了管理 ZigBee 协调器、路由器或终端设备的接口的功能。这包括创建、发现和加入一个 ZigBee 网络；绑定应用程序终端以及安全管理。

2.2 AF 层

应用构架层（AF）接口支持一个终端（包括 ZDO 层）接口到基本协议栈。Z 协议栈的 AF 层在开发人员需要建立一个设备描述时提供了所需的数据结构和辅助功能，是传入信息的终端多路复用器。

2.3 APS 层

应用支持子层（APS）API 提供一般性的支持服务，能同时用于 ZDO 层和制造商定义的应用对象。

2.4 NWK 层

ZigBee 网络层（NWK）为上层（应用层）组件提供管理和数据服务。

3、应用程序接口

本节提供了 Z 协议栈使用的常用的数据结构的概览，以及访问指定层主要功能 API 接口。

3.1 ZigBee 设备对象（ZDO）

本节列举了 ZDO 层提供的所有函数调用，这些对 ZigBee 设备 Profile（ZDP）中规定的所有命令执行和响应来说都是必需的，以及其它能使设备像 ZigBee 设备一样运行的功能。在概述一节中根据功能将 ZDO 的所有 API 函数分类，将在以下的章节中对每一类进行讨论。

3.1.1 概述

ZDP 描述了 ZDO 内部一般性的 ZigBee 设备功能是如何实现的。它定义了设备描述和 Cluster 使用的命令和响应对。通过命令数据结构中信息定义，ZDP 提供了如下功能的 ZDO 和应用：

- 设备网络启动
- 设备和服务发现
- 终端设备绑定、绑定和取消绑定服务
- 网络管理服务

设备发现是为一个 ZigBee 设备找到另一个 ZigBee 设备的过程。设备发现的一个例子是网络寻址请求，它以广播的形式发送，携带已知的 IEEE 地址作为数据的有效载荷。目标设备应该做出响应并告知其网络地址。设备发现使 PAN 网络中的设备能发现其它 ZigBee 设备提供的服务。它使用各种描述符来指定设备的功能。

终端设备绑定，绑定和取消绑定服务为 ZigBee 设备提供了绑定功能。典型的绑定用于网络配置期间，即当用户需要将控制设备绑定到被控制的设备（比如开关和灯光）时。特别地，终端设备绑定支持一个简化的绑定方法，即用户的输入用来识别控制或被控制的设备对。绑定和取消绑定服务提供了创建和删除绑定表条目的功能，其绑定表功能是映射控制信息到其各自的目的地。

网络管理服务提供了检索设备管理信息的功能，包括网络发现结果，路由表内容，到周边节点的链路质量以及绑定表内容。它还提供了通过断开与 PAN 网络设备的联系来控制网络联系的功能。网络管理服务主要是针对用户或调试工具而设计，以便管理网络。以上三种功能的 API 将在下面的分节中讨论。

3.1.2 ZDO 设备网络启动

通过默认的 ZDApp_Init()（在 ZDApp.c 中）开始启动 ZigBee 网络中的设备。但是一个应用程序可以跳过这个默认行为。对于控制网络设备启动的应用程序，必须包含 HOLD_AUTO_START 作为编译选项，还建议包含 NV_RESTORE 编译选项（以便在 NV 中保存 ZigBee 网络状态）。如果设备包含这些编译标志，将需要调用 ZDOInitDevice()来启动网络中的设备。

3.1.2.1 ZDOInitDevice()

启动网络中的设备。这个函数将读取 ZCD_NV_STARTUP_OPTION (NV 项目)的值来决定是否存储设备的网络状态。要调用这个函数，设备必须先编译 HOLD_AUTO_START 编译标志。

如果应用程序想要强制加入一个新设备，在调用此函数之前应用程序应该设置 ZCD_NV_STARTUP_OPTION **NV** 项目下的 ZCD_STARTOPT_DEFAULT_NETWORK_STATE 位，连接新设备意味着不再恢复网络设备的状态。使用 zgWriteStartupOptions() 来设置这些选项：[zgWriteStartupOptions(ZG_STARTUP_SET，ZCD_STARTOPT_DEFAULT_NETWORK_STATE)；]。

函数原型

uint8 ZDOInitDevice(uint16 startDelay)；

参数描述

startDelay—设备启动时延（单位为毫秒）。这一时延增加了一个变化幅度：((NWK_START_DELAY + startDelay) + (osal_rand() & EXTENDED_JOINING_RANDOM_MASK))

返回值

这个函数将返回下面值之一：

名称	描述
ZDO_INITDEV_RESTORED_NETWORK_STATE	设备网络状态已被恢复。
ZDO_INITDEV_NEW_NETWORK_STATE	网络状态被初始化，这可能意味着 ZCD_NV_STARTUP_OPTION 定义不恢复，或者说明没有网络状态可被恢复。
ZDO_INITDEV_LEAVE_NOT_STARTED	在复位之前，发出网络离开命令，且 rejoin 选项设定为 True，因此，设备未在网络中启动（仅一次）。下次，调用本函数将启动。

3.1.3 ZDO 发现 API

ZDO 发现 API 建立、发送 ZDO 设备和服务的发现请求和响应。所有 API 函数及其相应的 ZDP 命令列在下表中。用户可以将命令名称作为关键字在最新的 ZigBee 规范中查阅，以作进一步参考。这些函数将在下面的分节中一一详细讨论。

ZDO API 功能	ZDP 发现命令
ZDP_NwkAddrReq()	NWK_addr_req
ZDP_NWKAddrRsp()	NWK_addr_rsp
ZDP_IEEEAddrReq()	IEEE_addr_req
ZDP_IEEEAddrRsq()	IEEE_addr_rsp
ZDP_NodeDescReq()	Node_Desc_req

ZDP_NodeDescRsp()	Node_Desc_rsp
ZDP_PowerDescReq()	Power_Desc_req
ZDP_PowerDescRsp()	Power_Desc_rsp
ZDP_SimpleDescReq()	Simple_Desc_req
ZDP_SimpleDescRsp()	Simple_Desc_rsp
ZDP_ComplexDescReq()	Complex_Desc_req
ZDP_ActiveEPIFReq()	Active_EP_req
ZDP_ActiveEPIFRsp()	Active_EP_rsp
ZDP_MatchDescReq()	Match_Desc_req
ZDP_MatchDescRsp()	Match_Desc_rsp
ZDP_UserDescSet()	User_Desc_set
ZDP_UserDescConf()	User_Desc_conf
ZDP_UserDescReq()	User_Desc_req
ZDP_UserDescRsp()	User_Desc_rsp
ZDP_EndDeviceAnnce()	Device_annce
ZDP_ServerDiscReq()	System_Server_Discovery_req
ZDP_ServerDiscRsp()	System_Server_Discovery_rsp

3.1.3.1 ZDP_NwkAddrReq()

调用此函数将生成一个根据其已知IEEE地址询问远程设备16位地址的信息。这个信息作为一个广播信息发送给网络中所有设备。

函数原型

```
afStatus_t ZDP_NwkAddrReq( byte *IEEEAddress, byte ReqType,
    byte StartIndex, byte SecuritySuite );
```

参数描述

IEEEAddress—请求设备的IEEE地址。

ReqType—响应信息类型要求。其允许值列在下表中：

ReqType

名称	意义
ZDP_NWKADDR_REQTYPE_SINGLE	只返回设备的短地址和扩展地址。
ZDP_NWKADDR_REQTYPE_EXTENDED	返回设备的短地址和扩展地址，还有所有相关设备的短地址。

StartIndex—响应设备的响应信息可以有更多的响应选项，请求者可以给可能的响应设备指定一个起始索引。索引从0开始。

SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用AF层来发送信息，所以状态值即为定义在ZComDef.h的ZStatus_t中的AF状态值。

3.1.3.2 ZDP_NWKAddrRsp()

这实际上是一个调用ZDP_AddrRsp()的宏。这个调用将建立和发送一个网络地址响应。

函数原型

```
afStatus_t ZDP_NWKAddrRsp( byte TranSeq, zAddrType_t *dstAddr,
    byte Status, byte *IEEEAddrRemoteDev, byte ReqType, uint16 nwkAddr,
    byte NumAssocDev, byte StartIndex,
    uint16 *NWKAddrAssocDevList,
    byte SecuritySuite );
```

参数描述

TranSeq—本次事务的序列号。

DstAddr—目的地址。

Status—状态值如下：

状态	
意义	值
ZDP_SUCCESS	0
ZDP_INVALID_REQTYPE	1
ZDP_DEVICE_NOT_FOUND	2
保留	0x03-0xff

IEEEAddrRemoteDev—远程设备的 64 位地址。

ReqType—该请求的请求类型。

nwkAddr—远程设备的16位地址。

NumAssocDev—计数与远程设备关联的设备个数，以及各自的16位短地址个数。如果远程设备无关联设备，NumAssocDev值应为0，且StartIndex 和NWKAddrAssocDevList的值将为空。

StartIndex—响应设备的响应信息中可以有更多的响应选项。请求者可以给可能的响应设备指定一个起始索引。这是该响应信息的起始索引字段。

NWKAddrAssocDevList—16位地址列表，与远程设备的相关设备一一对应。NWKAddrAssocDevList 中的16位地址个数由NumAssocDev提供。

SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用AF层来发送信息，所以状态值即为定义在ZComDef.h的ZStatus_t 中的AF状态值。

3.1.3.3 ZDP_IEEEAddrReq()

调用此函数将生成一个根据其已知16位网络地址询问远程设备64位地址的信息。

函数原型

```
afStatus_t ZDP_IEEEAddrReq( uint16 shortAddr, byte ReqType,
    byte StartIndex, byte SecuritySuite );
```

参数描述

shortAddr—已知的16位网络地址。

ReqType—响应信息类型要求。

ReqType	
名称	意义
ZDP_IEEEADDR_REQTYPE_SINGLE	返回设备的短地址和扩展地址
ZDP_IEEEADDR_REQTYPE_EXTENDED	返回设备的短地址和扩展地址还有所有相关设备的短地址

StartIndex—响应设备的响应信息可以有更多的响应选项。请求者可以给可能的响应设备指定一个起始索引。

SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用AF层来发送信息，所以状态值即为定义在ZComDef.h的ZStatus_t 中的AF状态值。

3.1.3.4 ZDP_IEEEAddrRsp()

这实际上是一个调用ZDP_AddrRsp()的宏。这个调用将建立和发送一个IEEE地址响应。

函数原型

```
afStatus_t  ZDP_IEEEAddrRsp(  byte  TranSeq,  zAddrType_t  *dstAddr,
    byte  Status,  byte  *IEEEAddrRemoteDev, byte  ReqType,  uint16  nwkAddr,
    byte  NumAssocDev,  byte  StartIndex,
    uint16  *NWKAddrAssocDevList,
    byte  SecuritySuite )；
```

参数描述

TranSeq—本次事务的序列号。

DstAddr—目的地址。

Status—状态值如下：

状态	
意义	值
ZDP_SUCCESS	0
ZDP_INVALID_REQTYPE	1
ZDP_DEVICE_NOT_FOUND	2
保留	0x03-0xff

IEEEAddrRemoteDev—远程设备的64位地址。

ReqType—该请求的请求类型。

nwkAddr—远程设备的16位地址。

NumAssocDev—计数与远程设备关联的设备个数，以及各自的16位短地址个数。如果远程设备无相关设备，NumAssocDev值应为0，且StartIndex 和NWKAddrAssocDevList的值应为空。

StartIndex—本次报告中相关设备列表的起始索引。

NWKAddrAssocDevList—16位地址列表，与远程设备的相关设备一一对应。NWKAddrAssocDevList中的16位地址个数由NumAssocDev提供。

SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.5 ZDP_NodeDescReq()

这实际上是一个调用 ZDP_NWKAddrOfInterestReq()的宏。这个调用将给目的地址字段指定的远程设备建立和发送一个节点描述请求。

函数原型

```
afStatus_t ZDP_NodeDescReq( zAddrType_t *dstAddr, uint16 NWKAddrOfInterest,
    byte SecuritySuite );
```

参数描述

DstAddr—目的地址。

NWKAddrOfInterest—要寻找的 16 位短地址。

SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.6 ZDP_NodeDescMsg()

调用此函数来响应节点描述请求。

函数原型

```
afStatus_t ZDP_NodeDescMsg( byte TransSeq, zAddrType_t *dstAddr, byte Status,
    uint16 nwkAddr, NodeDescriptorFormat_t *pNodeDesc, byte SecuritySuite )；
```

参数描述

TranSeq—本次事务的序列号。

DstAddr—目的地址。

Status—状态值如下：

状态	
意义	值
SUCCESS	0
DEVICE_NOT_FOUND	1

nwkAddr—设备的 16 位地址。

pNodeDesc—节点描述的指针（定义在 AF.h 中）。
SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.7 ZDP_PowerDescReq()

这实际上是一个调用 ZDP_NWKAddrOfInterestReq()的宏。这个调用将建立和发送一个电源描述符请求。使用这个宏来请求远程设备的电源描述。

函数原型

```
afStatus_t ZDP_PowerDescReq( zAddrType_t *dstAddr, int16 NWKAddrOfInterest,
    byte SecuritySuite );
```

参数描述

DstAddr—目的地址。
NWKAddrOfInterest—要寻找的 16 位短地址。
SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.8 ZDP_PowerDescMsg()

调用此函数来响应电源描述符请求。

函数原型

```
afStatus_t ZDP_PowerDescMsg( byte TranSeq, zAddrType_t *dstAddr, byte Status,
    int16 nwkAddr, NodePowerDescriptorFormat_t *pPowerDesc,
    byte SecuritySuite );
```

参数描述

TranSeq—本次事务的序列号。
DstAddr—目的地址。
Status—状态值如下：

状态	
意义	值

SUCCESS	0
DEVICE_NOT_FOUND	1

nwkAddr—设备的 16 位地址
pPowerDesc—电源描述符的指针（定义在 AF.h 中）。
SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.9 ZDP_SimpleDescReq()

调用此函数将建立和发送一个简单描述符请求。

函数原型

```
afStatus_t ZDP_SimpleDescReq( zAddrType_t *dstAddr, uint16 nwkAddr,
    byte epIntf, byte SecuritySuite );
```

参数描述

DstAddr—目的地址。
nwkAddr—已知的 16 位网络地址。
epIntf—想要的应用程序的端点或接口。
SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.10 ZDP_SimpleDescRsp()

调用此函数来响应简单描述符请求。

函数原型

```
afStatus_t ZDP_SimpleDescRsp( byte TranSeq, zAddrType_t *dstAddr,
    byte Status, SimpleDescriptionFormat_t *pSimpleDesc,
    byte SecuritySuite );
```

参数描述

TranSeq—本次事务的序列号。

DstAddr—目的地址。

Status—状态值如下：

状态	
意义	值
SUCCESS	0
INVALID_EP	1
NOT_ACTIVE	2
DEVICE_NOT_FOUND	3

pSimpleDesc—简单描述符的指针（定义在 AF.h 中）。

SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.11 ZDP_ComplexDescReq()

这个调用建立和发送一个复杂描述符请求。它是一个调用 ZDP_NWKAddrOfInterestReq() 的宏。

函数原型

```
afStatus_t  ZDP_ComplexDescReq(     zAddrType_t  *dstAddr,
      uint16  nwkAddr,  byte  SecuritySuite  );
```

参数描述

DstAddr—目的地址。

nwkAddr—已知的 16 位网络地址。

SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.12 ZDP_ActiveEPIFReq()

这实际上是一个调用 ZDP_NWKAddrOfInterestReq()的宏。这个调用将建立和发送一个动态终端或接口请求。使用这个宏来请求远程设备的所有动态终端或接口。

函数原型

afStatus_t ZDP_ActiveEPIFReq(zAddrType_t *dstAddr, uint16 NWKAddrOfInterest,
byte SecuritySuite);

参数描述

DstAddr—目的地址。
NWKAddrOfInterest—要寻找的 16 位短地址。
SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.13 ZDP_ActiveEPIFRsp()

这是一个调用 ZDP_EPIFRsp()的宏。调用此函数来响应动态终端或接口请求。

函数原型

afStatus_t ZDP_ActiveEPIFRsp(byte TranSeq, zAddrType_t *dstAddr,
byte Status, uint16 nwkAddr, byte Count, byte *pEPIntfList, byte SecuritySuite);

参数描述

TranSeq—本次事务的序列号。
DstAddr—目的地址。
Status—状态值如下：

状态	
意义	值
SUCCESS	0
DEVICE_NOT_FOUND	1

nwkAddr—设备的 16 位网络地址。
Count—pEPIntfList 中的动态终端或接口个数。
pEPIntfList—设备的动态终端或接口数组。
SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.14 ZDP_MatchDescReq()

这个调用将建立和发送一个匹配描述请求。使用这个函数查询与应用程序输入或输出 Cluster 列表中相匹配的设备或应用程序。

函数原型

```
afStatus_t ZDP_MatchDescReq( zAddrType_t *dstAddr, uint16 nwkAddr,
    uint16 ProfileID,
    byte NumInClusters, byte *InClusterList,
    byte NumOutClusters, byte *OutClusterList, byte SecuritySuite );
```

参数描述

DstAddr—目的地址。
nwkAddr—已知的 16 位网络地址。
ProfileID—应用程序的 Profile ID，作为 Cluster ID 的参考。
NumInClusters—Cluster 输入列表中的 Cluster ID 个数。
InClusterList—Cluster ID 的输入数组（每个占 1 个字节）。
NumOutClusters—输出列表中的 Cluster ID 个数。
OutClusterList—Cluster ID 的输出数组（每个占 1 个字节）。
SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.15 ZDP_MatchDescRsp()

这是一个调用 ZDP_EPIFRsp()的宏。调用此函数响应匹配描述请求。

函数原型

```
afStatus_t ZDP_MatchDescRsp( byte TranSeq, zAddrType_t *dstAddr, byte Status,
    uint16 nwkAddr, byte Count, byte *pEPIntfList, byte SecuritySuite );
```

参数描述

TranSeq—本次事务的序列号。
DstAddr—目的地址。
Status—状态值如下：

状态

意义	值
SUCCESS	0
DEVICE_NOT_FOUND	1

nwkAddr—设备的 16 位网络地址。
Count—pEPIntfList 中的动态终端或接口个数。
pEPIntfList—设备的动态终端或接口数组。
SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.16 ZDP_UserDescSet()

这个函数建立和发送一个 User_Desc_set 信息来设置远程设备的用户描述符。该请求单播到远程设备本身或含有远程设备发现信息的其它设备。请注意，远程设备应该定义了 NV_RESTORE 以启用此功能。

函数原型

```
afStatus_t  ZDP_UserDescSet(  zAddrType_t  *dstAddr,
                              uint16  nwkAddr,
                              UserDescriptorFormat_t  *UserDescriptor,
                              byte  SecurityEnable  );
```

参数描述

dstAddr—请求信息的目的地址。
nwkAddr—要寻找的远程设备的 16 位网络地址。
UserDescriptor—用户描述符配置。它包含一个长度小于或等于 16 个字符的 ASCII 字符串。它将被字符(0x02)填充空间使得字符总长度为 16。
SecurityEnable—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.17 ZDP_UserDescConf()

这是一个直接调用 ZDP_SendData ()的宏。调用此函数来响应 User_Desc_Conf 命令。

函数原型

afStatus_t ZDP_UserDescConf(byte TranSeq, zAddrType_t *dstAddr,
byte Status, byte SecurityEnable);

参数描述

TranSeq—本次事务的序列号。

DstAddr—目的地址。

Status—状态值如下：

状态	
意义	值
SUCCESS	0x00
INV_REQUESTTYPE	0x80
DEVICE_NOT_FOUND	0x81
NOT_SUPPORTED	0x84

SecurityEnable—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.18 ZDP_UserDescReq()

这个调用将建立和发送一个 User_Desc_Req 命令。它是一个调用 ZDP_NWKAddrOfInterestReq()的宏。

函数原型

afStatus_t ZDP_UserDescReq(zAddrType_t *dstAddr, uint16 nwkAddr,
byte SecurityEnable);

参数描述

DstAddr—目的地址。

nwkAddr—已知的 16 位网络地址。

SecurityEnable—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.19 ZDP_UserDescRsp()

这个调用将建立和发送一个 User_Desc_Rsp 命令。

函数原型

ZStatus_t ZDP_UserDescRsp(byte TransSeq, zAddrType_t *dstAddr,
uint16 nwkAddrOfInterest, UserDescriptorFormat_t *userDesc，
byte SecurityEnable);

参数描述

TranSeq—本次事务的序列号。
DstAddr—目的地址。
nwkAddrOfInterest—已知的 16 位网络地址。
userDesc—本地设备的用户描述符。
SecurityEnable—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.20 ZDP_EndDeviceAnnce()

这个函数为 ZigBee 终端设备建立和发送一个 End_Device_annce 命令，来通知网络中的其它 ZigBee 设备，该终端设备已经加入或重新加入网络。该命令包含终端设备新的 16 位网络地址和 64 位 IEEE 地址，以及 ZigBee 终端设备的容量。它以广播信息的形式向外发送。

收到 End_Device_annce 命令后，所有接收者应在其所有内部参考中检查信息中提供的 IEEE 地址，并更新相应的网络地址。但不会为 End_Device_annce 命令再返回响应信息。

函数原型

afStatus_t ZDP_EndDeviceAnnce(uint16 nwkAddr, byte *IEEEAddr,
byte capabilities, byte SecurityEnable);

参数描述

nwkAddr—本地设备的 16 位网络地址。
IEEEAddr—本地设备的 64 位 IEEE 地址指针。
Capabilities—本地设备的容量。
SecurityEnable—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.21 ZDP_ServerDiscReq()

这个函数建立和发送一个 System_Server_Discovery_req 请求信息，它包含一个 16 位服务器掩码。该请求的目的是寻找某个特定的系统服务器或服务器掩码中指明的服务器。该信息广播发送到所有包含 RxOnWhenIdle 的设备。远程设备将接收到的服务器掩码和存储在本地节点描述符的掩码相比较后，只要有一位相匹配，运用单播形式传输。

函数原型

afStatus_t ZDP_ServerDiscReq(uint16 serverMask, byte SecurityEnable);

参数描述

serverMask—要寻找的系统服务器的 16 位屏蔽位。

SecurityEnable—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.3.22 ZDP_ServerDiscRsp()

这个函数建立和返回一个 System_Server_Discovery_rsp 响应。当收到 System_Server_Discovery_req 命令，且发现服务器掩码有匹配位时此函数将被调用。

函数原型

ZStatus_t ZDP_ServerDiscRsp(byte transID, zAddrType_t *dstAddr, byte status, uint16 aoi, uint16 serverMask, byte SecurityEnable);

参数描述

TransID—本次 ZDO 事务序列号。

dstAddr—响应的目的地。

status—状态始终是 ZSUCCESS。

aoi—目标地址。目前还不使用。

serverMask—指明匹配系统服务器的 16 位屏蔽位。

securityEnable—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.4 ZDO 绑定 API

ZDO 绑定 API 建立、发送 ZDO 绑定请求和响应。所有绑定信息（表）保存在 ZigBee 协调器中。因此只有 ZigBee 协调器可以接收绑定请求。下面的表格列出了 ZigBee 规范中协议栈支持的绑定 API，以及各自的响应命令名称。用户可以将命令名称作为关键字在最新的 ZigBee 规范中查阅，以作进一步参考。这些原语将在下面的分节中一一讨论。

ZDO 绑定 API	ZDP 绑定服务命令
ZDP_EndDeviceBindReq()	End_Device_Bind_req
ZDP_EndDeviceBindRsp()	End_Device_Bind_rsp
ZDP_BindReq()	Bind_req
ZDP_BindRsp()	Bind_rsp
ZDP_UnbindReq()	Unbind_req
ZDP_UnbindRsq()	Unbind_rsp

3.1.4.1 ZDP_EndDeviceBindReq()

这个调用将建立和发送一个终端设备绑定请求（“手动绑定”）。可以发送该信息来尝试一个设备的手动绑定功能。手动绑定之后你可以给协调器发送一个间接（无地址）信息，协调器将把该信息发送至绑定了该信息的设备，或者你将从新的绑定设备接收信息。

函数原型

```
afStatus_t  ZDP_EndDeviceBindReq(  zAddrType_t  *dstAddr,
      uint16  LocalCoordinator,
      byte  epIntf,
      uint16  ProfileID,
      byte  NumInClusters,  byte  *InClusterList,
      byte  NumOutClusters,  byte  *OutClusterList, byte  SecuritySuite  );
```

参数描述

DstAddr—目标地址。
LocalCoordinator—已知 16 位网络地址设备的父节点协调器。
epIntf—应用程序的终端或接口。
ProfileID—应用程序的 Profile ID，作为 Cluster ID 的参考。
NumInClusters—Cluster 输入列表中的 Cluster ID 个数。
InClusterList—Cluster ID 的输入数组（每个占一位字节）。
NumOutClusters—Cluster 输出列表中的 Cluster ID 个数。
OutClusterList—Cluster ID 的输出数组（每个占一位字节）。

SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.4.2 ZDP_EndDeviceBindRsp()

这是一个直接调用 ZDP_SendData ()的宏。调用此函数来响应终端设备绑定请求。

函数原型

```
afStatus_t ZDP_EndDeviceBindRsp( byte TranSeq, zAddrType_t *dstAddr,
    byte Status, byte SecurityEnable );
```

参数描述

TranSeq—本次事务序列号。

DstAddr—目的地址。

Status—状态值如下：

状态	
意义	值
SUCCESS	0
NOT_SUPPORTED	1
TIMEOUT	2
NO_MATCH	3

SecurityEnable—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.4.3 ZDP_BindReq()

这实际上是一个调用 ZDP_BindUnbindReq()的宏。这个调用将建立和发送一个绑定请求。使用这个函数来请求 ZigBee 协调器根据 Cluster ID 去绑定应用程序的。

函数原型

afStatus_t ZDP_BindReq(zAddrType_t *dstAddr, byte *SourceAddr,
byte SrcEPIntf, byte ClusterID, byte *DestinationAddr, byte DstEPIntf,
byte SecuritySuite);

参数描述

DstAddr—目的地址。

SourceAddr—源设备的 64 位 IEEE 地址。

SrcEPIntf—源端点或接口。

ClusterID—绑定信息的 Cluster ID。

DestinationAddr—目标设备的 64 位 IEEE 地址。

DstEPIntf—目标端点或接口。

SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.4.4 ZDP_BindRsp()

这是一个直接调用 ZDP_SendData ()的宏。调用此函数响应绑定请求。

函数原型

afStatus_t ZDP_BindRsp(byte TranSeq, zAddrType_t *dstAddr, byte Status，
byte SecurityEnable)；

参数描述

TranSeq—本次事务的序列号。

DstAddr—目的地址。

Status—状态值如下：

状态	
意义	值
SUCCESS	0
NOT_SUPPORTED	1
TABLE_FULL	2

SecurityEnable—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.4.5 ZDP_UnbindReq ()

这实际上是一个调用 ZDP_BindUnbindReq ()的宏。这个调用将建立和发送一个取消绑定请求。使用这个函数来请求 ZigBee 协调器取消绑定。

函数原型

```
afStatus_t  ZDP_UnbindReq(  zAddrType_t  *dstAddr,
      byte  *SourceAddr,  byte  SrcEPIntf,
      byte  ClusterID,
      byte  *DestinationAddr,  byte  DstEPIntf,
      byte  SecuritySuite  );
```

参数描述

DstAddr—目的地址。
SourceAddr—源设备的 64 位 IEEE 地址。
SrcEPIntf—源端点或接口。
ClusterID—绑定信息的 Cluster ID。
DestinationAddr—目标设备的 64 位 IEEE 地址。
DstEPIntf—目标端点或接口。
SecuritySuite—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.4.6 ZDP_UnbindRsp()

这是一个直接调用 ZDP_SendData ()的宏。调用此函数响应取消绑定请求。

函数原型

```
afStatus_t  ZDP_UnbindRsp( byte  TranSeq，  zAddrType_t  *dstAddr，
      byte  Status, byte  SecurityEnable);
```

参数描述

TranSeq—本次事务的序列号。
DstAddr—目的地址。
Status—状态值如下：

状态	
意义	值
SUCCESS	0
NOT_SUPPORTED	1
NO_ENTRY	2

SecurityEnable—该信息的安全类型要求。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.5 ZDO 管理 API

ZDO 管理 API 建立、发送 ZDO 管理请求和响应。这些信息用来获取设备的状态和更新表格。下面的表格列出了 ZigBee 规范中协议栈支持的管理 API，以及各自的响应命令名称。用户可以将命令名称作为关键字在最新的 ZigBee 规范中查阅，以作进一步参考。这些原语将在下面的分节中一一讨论。

ZDP 管理 API	ZDP 网络管理服务命令
ZDP_MgmtNwkDiscReq()	Mgmt_NWK_Disc_req
ZDP_MgmtNwkDiscRsp()	Mgmt_NWK_Disc_rsp
ZDP_MgmtLqiReq()	Mgmt_Lqi_req
ZDP_MgmtLqiRsp()	Mgmt_Lqi_rsp
ZDP_MgmtRtgReq()	Mgmt_Lqi_req
ZDP_MgmtRtgRsp()	Mgmt_Rtg_rsp
ZDP_MgmtBindReq()	Mgmt_Bind_req
ZDP_MgmtBindRsp()	Mgmt_Bind_rsp
ZDP_MgmtLeaveReq()	Mgmt_Leave_req
ZDP_MgmtLeaveRsp()	Mgmt_Leave_rsp
ZDP_MgmtDirectJoinReq()	Mgmt_Direct_Join_req
ZDP_MgmtDirectJoinRsp()	Mgmt_Direct_Join_rsp
ZDP_MgmtPermitJoinReq()	Mgmt_Permit_Join_req
ZDP_MgmtPermitJoinRsp()	Mgmt_Permit_Join_rsp

3.1.5.1 ZDP_MgmtNwkDiscReq()

如果设备支持该命令（可选），调用此函数将产生一个为目标设备执行网络扫描的请求。只有在 ZDConfig.h 中设置了 ZDO_MGMT_NWKDISC_REQUEST 的编译标志或作为一个正常编译标志，调用程序才可以调用这个函数。

函数原型

```
afStatus_t  ZDP_MgmtNwkDiscReq(  zAddrType_t  *dstAddr,
      uint32  ScanChannels,  byte  StartIndex,  byte  SecurityEnable  );
```

参数描述

DstAddr—目的地址。

ScanChannels—请求扫描包含屏蔽位的信道。信道定义包含在 NLMEDE.h (例如，DEFAULT_CHANLIST)中。

StartIndex—响应设备的响应信息可以有更多的响应选项，请求程序可以给可能的响应设备指定一个起始索引。

SecurityEnable—如果安全功能开启则为 TRUE。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.5.2 ZDP_MgmtNwkDiscRsp()

如果设备支持该命令（可选），调用此函数将产生一个响应信息。如果在 ZDConfig.h 中设置了 ZDO_MGMT_NWKDISC_RESPONSE 的编译标志或作为一个正常编译标志，当 ZDO 层接收到一个“管理网络发现请求”信息时，将自动生成该信息。

函数原型

```
afStatus_t ZDP_MgmtNwkDiscRsp( byte TranSeq, zAddrType_t *dstAddr,
    byte Status,
    byte NetworkCount, byte StartIndex, byte NetworkCountList,
    networkDesc_t *NetworkList, byte SecurityEnable );
```

参数描述

TranSeq—本次事务的序列号。

DstAddr—目的地址。

Status—定义在 ZComDef.h 中，即 ZStatus_t 。

NetworkCount—该信息允许的项目个数。

StartIndex—响应设备的响应信息可以有更多的响应选项，请求程序可以给可能的响应设备指定一个起始索引。这是该响应信息的起始索引字段。

NetworkCountList—该信息响应项目的个数。

NetworkList—网络发现记录的列表。在这个数据结构中，你可以在 NLMEDE.h 中查找 networkDesc_t 获得信息。

SecurityEnable—如果安全功能开启则为 TRUE。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.5.3 ZDP_MgmtLqiReq()

如果设备支持该命令（可选），调用此函数将生成一个返回目标设备相邻设备列表的请求。只有在 ZDConfig.h 中设置了 ZDO_MGMT_LQI_REQUEST 的编译标志或作为一个正常编译标志，调用程序才可以调用这个函数。

函数原型

```
afStatus_t  ZDP_MgmtLqiReq  (  zAddrType_t  *dstAddr,
    byte  StartIndex,  byte  SecurityEnable  );
```

参数描述

DstAddr—目的地址。

StartIndex—响应设备的响应信息可以有更多的响应选项，请求程序可以给可能的响应设备指定一个起始索引。

SecurityEnable—如果安全功能开启则为 TRUE。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.5.4 ZDP_MgmtLqiRsp()

如果设备支持该命令（可选），调用此函数将生成一个响应信息。如果在 ZDConfig.h 中设置了 ZDO_MGMT_LQI_RESPONSE 的编译标志或作为一个正常编译标志，当 ZDO 层接收到一个“管理网络 LQI 请求”信息时，将自动生成该信息。

函数原型

```
ZStatus_t  ZDP_MgmtLqiRsp(  byte  TranSeq,  zAddrType_t  *dstAddr,
    byte  Status,  byte  NeighborLqiEntries,
    byte  StartIndex,  byte  NeighborLqiCount,
    neighborLqiItem_t  *NeighborLqiList,
    byte  SecurityEnable  );
```

参数描述

TranSeq—本次事务的序列号。

DstAddr—目的地址。

Status—定义在 ZComDef.h 中，即 ZStatus_t。

NeighborLqiEntries—该信息允许的项目个数。

StartIndex—响应设备的响应信息可以有更多的响应选项，请求程序可以给可能的响应设备指定一个起始索引。这是该响应信息的起始索引字段。

NeighborLqiCount—该信息响应项目的个数。

NeighborLqiList—相邻设备记录表。在这个数据结构中，你可以在 ZDProfile.h 中查找 neighborLqiItem_t 获得信息。

SecurityEnable—如果安全功能开启则为 TRUE。

返回值

ZStatus_t—状态值定义在 ZComDef.h 中，即 ZStatus_t。

3.1.5.5 ZDP_MgmtRtgReq()

如果设备支持该命令（可选），调用此函数将生成一个返回目标设备路由表的请求。只有在 ZDConfig.h 中设置了 ZDO_MGMT_RTG_REQUEST 的编译标志或作为一个正常编译标志，调用程序才可调用此函数。

函数原型

```
afStatus_t  ZDP_MgmtRtgReq(  zAddrType_t  *dstAddr,
                             byte  StartIndex,  byte  SecurityEnable  );
```

参数描述

DstAddr—目的地址。

StartIndex—响应设备的响应信息可以有更多的响应选项，请求程序可以给可能的响应设备指定一个起始索引。

SecurityEnable—如果安全功能开启则为 TRUE。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.5.6 ZDP_MgmtRtgRsp()

如果设备支持该命令（可选），调用此函数将生成一个响应信息。如果在 ZDConfig.h 中设置了 ZDO_MGMT_RTG_RESPONSE 的编译标志或作为一个正常编译标志，当 ZDO 层接收到一个“管理路由请求”信息时，将自动生成该信息。

函数原型

```
ZStatus_t  ZDP_MgmtRtgRsp(  byte  TranSeq,  zAddrType_t  *dstAddr,
      byte  Status,  byte  RoutingTableEntries,
      byte  StartIndex,  byte  RoutingListCount,
      rtgItem_t  *RoutingTableList,  byte  SecurityEnable  );
```

参数描述

TranSeq—本次事务的序列号。

DstAddr—目的地址。

Status—定义在 ZComDef.h 中，即 ZStatus_t。

RoutingTableEntries—该信息允许的项目个数。

StartIndex—响应设备的响应信息可以有更多的响应选项，请求程序可以给可能的响应设备指定一个起始索引。这是该响应信息的起始索引字段。

RoutingListCount—该信息响应项目的个数。

RoutingTableList—路由记录表。在这个数据结构中，你可以在 ZDProfile.h 中查找 rtgItem_t 获得信息。

SecurityEnable—如果安全功能开启则为 TRUE。

返回值

ZStatus_t—状态值定义在 ZComDef.h 中，即 ZStatus_t 。

3.1.5.7 ZDP_MgmtBindReq()

如果设备支持该命令（可选），调用此函数将生成一个返回目标设备绑定表的请求。只有在 ZDConfig.h 中设置了 ZDO_MGMT_BIND_REQUEST 的编译标志或作为一个正常编译标志， 调用程序才可调用此函数。

函数原型

```
afStatus_t  ZDP_MgmtBindReq(  zAddrType_t  *dstAddr,
      byte  StartIndex,  byte  SecurityEnable  );
```

参数描述

DstAddr—目的地址。

StartIndex—响应设备的响应信息可以有更多的响应选项，请求程序可以给可能的响应设备指定一个起始索引。

SecurityEnable—如果安全功能开启则为 TRUE。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。

3.1.5.8 ZDP_MgmtBindRsp()

如果设备支持该命令（可选），调用此函数将生成一个响应信息。如果在 ZDConfig.h 中设置了 ZDO_MGMT_BIND_RESPONSE 的编译标志或作为一个正常编译标志，当 ZDO 层接收到一个“管理绑定请求”信息时，将自动生成该信息。

函数原型

```
ZStatus_t  ZDP_MgmtBindRsp(  byte  TranSeq,  zAddrType_t  *dstAddr,
    byte  Status,  byte  BindingTableEntries,
    byte  StartIndex,  byte  BindingTableListCount，
    apsBindingItem_t  *BindingTableList,
    byte  SecurityEnable  );
```

参数描述

TranSeq—本次事务的序列号。

DstAddr—目的地址。

Status—定义在 ZComDef.h 中，即 ZStatus_t。

BindingTableEntries—该信息允许的项目个数。

StartIndex—响应设备的响应信息可以有更多的响应选项，请求程序可以给可能的响应设备指定一个起始索引。这是该响应信息的起始索引字段。

BindingTableListCount—该信息响应项目的个数。

BindingTableList—绑定表历史记录清单。你可以在 apsBindingItem_t 中查找 APSMEDE.h 获得关于这个数据结构的信息。

SecurityEnable—如果安全功能开启则为 TRUE。

返回值

ZStatus_t—定义在 ZComDef.h 中的 ZStatus_t 状态值。

3.1.5.9 ZDP_MgmtLeaveReq()

如果设备支持该命令（可选），调用此函数将生成一个让目标设备离开网络或请求另一个设备离开的请求。只有在 ZDConfig.h 中设置了 ZDO_MGMT_LEAVE_REQUEST 的编译标志或作为一个正常编译标志，调用程序才可调用此函数。

函数原型

```
afStatus_t  ZDP_MgmtLeaveReq(  zAddrType_t  *dstAddr,
    byte  *IEEEAddr,  byte  SecurityEnable  );
```

参数描述

DstAddr—目的地址。

IEEEAddr—要离开的设备的 64 位地址（8 个字节）。

SecurityEnable—如果安全功能开启则为 TRUE。

返回值

afStatus_t—这个函数使用 AF 层来发送信息，所以状态值即为定义在 ZComDef.h 的 ZStatus_t 中的 AF 状态值。