ZigBee Device Profile (ZDP) APIs Reference Manual

JN-RM-2017
Revision 1.5
3-Aug-2007

Contents

About this Manual

This manual describes the ZigBee Device Profile (ZDP) APIs of the Jennic ZigBee stack.

Organisation

The manual consists of four chapters, as follows:

- Chapter 1 provides an introductory summary of the ZDP APIs.
- Chapters 2 to 4 detail the functions of the available APIs.

Conventions

Files, folders, functions and parameter types are represented in **bold** type.

Function parameters are represented in *italics* type.

Code fragments are represented in the Courier typeface.

Acronyms and Abbreviations

API	Application Programming Interface
BOS	Basic Operating System
ZDO	ZigBee Device Objects
ZDP	ZigBee Device Profile

Related Documents

[R1]	ZigBee Specification version 1.0, ZigBee Alliance document 053474r06, December 14 2004
[R2]	ZigBee Stack User Guide [JN-UG-3017]
[R3]	ZigBee Application Framework APIs Reference Manual [JN-RM-2018]
[R4]	ZigBee Application Development API Reference Manual [JN-RM-2014]

Feedback Address

If you wish to comment on this manual, or any other Jennic user documentation, please provide your feedback by writing to us (quoting the manual reference number and version) at the following postal address or e-mail address:

Applications
Jennic Ltd
Furnival Street
Sheffield S1 4QT
United Kingdom

doc@jennic.com

1 Introduction

This chapter introduces the ZDP (ZigBee Device Profile) APIs.

1.1 Summary of ZigBee Device Profile (ZDP) APIs

The ZDP APIs interact with the ZigBee Device Objects (ZDO) of a remote node. The ZDO resides in the Application Layer of the ZigBee software architecture. It represents the ZigBee node type of the device (Co-ordinator, Router or End Device) and has a number of communication roles; for more information on the ZDO, refer to the *ZigBee Stack User Guide*.

The following list summarises the ZDP APIs and their status in this release:

- **ZDP Device Discovery API**: Used to obtain the identities of other devices.
- **ZDP Service Discovery API**: Used to obtain the services provided by other devices.
- **ZDP Binding API**: Used for binding and unbinding devices.

These APIs operate on the principle of requests and responses. One node sends a request across the network, then the node that can fulfil this request sends a response. For example, to obtain the network address of the device with a given MAC address, a Network Address Request containing the MAC address is sent out - the device with this MAC address then responds with its network address. Responses are handled in the **JZA_vZdpResponse()** function; see the *ZigBee Application Development API Reference Manual* for more details.

Note: The ZDP functions use 0x0000 for the ZDP profile ID, and 0x00 for the source endpoint and destination endpoint. They also use the AF MSG frame.

1.2 Common Parameter

The following parameter is used in several functions:

Parameter	Type	Description
bTxOptions	**APS_TXOPTIONS**	APS_TXOPTION_NONE SECURITY_ENABLE_TRANSMISSION USE_NWK_KEY ACKNOWLEDGED_TRANSMISSION

Table 1: Common Parameter

1.3 ZDP Status Values

The following status values are used:

Parameter	Value
ZDP_SUCCESS_VALID	0x00
ZDP_INV_REQUESTTYPE	0x80
ZDP_DEVICE_NOT_FOUND	0x81
ZDP_INV_EP	0x82
ZDP_NOT_ACTIVE	0x83
ZDP_NOT_SUPPORTED	0x84
ZDP_TIMEOUT	0x85
ZDP_NO_MATCH	0x86
ZDP_TABLE_FULL	0x87
ZDP_NO_ENTRY	0x88
ZDP_NO_DESCRIPTOR	0x89

Table 2: ZDP Status

1.4 ZDP Cluster IDs

The following cluster IDs are used:

Parameter	Value
ZDP_NwkAddrReq	0x00
ZDP_IeeeAddrReq	0x01
ZDP_NodeDescReq	0x02
ZDP_PowerDescReq	0x03
ZDP_SimpleDescReq	0x04
ZDP_ActiveEpReq	0x05
ZDP_MatchDescReq	0x06
ZDP_ComplexDescReq	0x10
ZDP_UserDescReq	0x11
ZDP_DiscoveryRegisterReq	0x12
ZDP_EndDeviceAnnce	0x13
ZDP_UserDescSet	0x14
ZDP_EndDeviceBindReq	0x20
ZDP_BindReq	0x21
ZDP_UnbindReq	0x22
ZDP_MgmtNwkDiscReq	0x30
ZDP_MgmtLqiReq	0x31
ZDP_MgmtRtgReq	0x32
ZDP_MgmtBindReq	0x33
ZDP_MgmtLeaveReq	0x34
ZDP_MgmtDirectJoinReq	0x35
ZDP_NwkAddrRsp	0x80
ZDP_IeeeAddrRsp	0x81
ZDP_NodeDescRsp	0x82
ZDP_PowerDescRsp	0x83
ZDP_SimpleDescRsp	0x84
ZDP_ActiveEpRsp	0x85

ZDP_MatchDescRsp	0x86
ZDP_ComplexDescRs	0x90
ZDP_UserDescRsp	0x91
ZDP_DiscoveryRegisterRsp	0x92
//ZDP_EndDeviceAnnceRsp	0x93
ZDP_UserDescConf	0x94
ZDP_EndDeviceBindRsp	0xA0
ZDP_BindRsp	0xA1
ZDP_UnbindRsp	0xA2
ZDP_MgmtNwkDiscRsp	0xB0
ZDP_MgmtLqiRsp	0xB1
ZDP_MgmtRtgRsp	0xB2
ZDP_MgmtBindRsp	0xB3
ZDP_MgmtLeaveRsp	0xB4
ZDP_MgmtDirectJoinRsp	0xB5
ZDP_Zdo64bitAddressing	0xFF

Table 3: ZDP Cluster IDs

1.5 Format of Responses

All responses are passed to the user application by a call from the stack to the application through the **JZA_vZdpResponse()** function. The parameters of the response are contained in an array within the payload passed to **JZA_vZdpResponse()**.

As an example, consider the following example for the response **zdpMatchDescRsp**. The parameter list for this response is shown below:

Parameter	Array Pos'n	Type	Valid Range	Description
eZdpStatus	0	**ZDP_Status**	ZDP_SUCCESS_VALID ZDP_INV_REQUESTTYPE ZDP_DEVICE_NOT_FOUND ZDP_NO_DESCRIPTOR	Status of Match Descriptor Request
u16AddrInterest	1 to 2	**UINT16**	16-bit network address	Network address of responding device
nMatchLength	3	**UINT8**	0x00 to 0xFF	Number of matched endpoints
**pMatchList*	4	**UINT8**		Matched Endpoint List

Table 4: Example Parameter List for a Response

In the following code fragment, it can be seen how the response is passed via the **JZA_vZdpResponse()** function, and then parsed to extract the various fields from the payload.

```
PUBLIC void JZA_vZdpResponse(uint8 u8Type,
                             uint8 *pu8Payload,
                             uint8 u8PayloadLen)
{
    uint16 u16MatchAddr;
    uint8  u8MatchEndpoint;
    uint8  u8ListLength;

    switch (u8Type)
    {
    case ZDP_MatchDescRsp:
        /* response to a MatchDescriptor request has been receive. Check it
         * was successful
         */
        if (pu8Payload[0] == ZDP_SUCCESS_VALID)
        {
            /* check at least one matching endpoint was found */
            u8ListLength = pu8Payload[3];
            if (u8ListLength > 0)
            {
                /* extract 16-bit network address. VBOSReverseMemCpy also
                 * converts from Little Endian to Big Endian
                 */
                VBOSReverseMemCpy(&u16MatchAddr, &pu8Payload[1], 2);

                /* extract matching endpoint number */
                u8MatchEndpoint = pu8Payload[4];
            }
        }
        break;


    default:
        break;

    }
}
```

2 Device Discovery API

The Device Discovery API is used to obtain the addresses of network devices; the network address given the MAC (IEEE) address, and vice-versa.

The request functions and corresponding responses are listed below, along with their page references.

zdpNwkAddrReq (Cluster Id: 0x00)

```
void zdpNwkAddrReq(MAC_ExtAddr_s sExtAddr
                   REQUEST_TYPE eReqType,
                   UINT8 nStartIndex,
                   APS_TXOPTIONS bTxOptions);
```

Description

This function submits a Network Address Request - it is used to request the network address of the device with the specified IEEE address (when the IEEE address of the target device is known). The destination addressing used is Broadcast.

Parameters

sExtAddr	64-bit IEEE address of the target device
eReqType	One of the following: ZDP_SINGLE_DEVICE_RESPONSE: Indicates to get the network address of the target device only ZDP_EXTENDED_RESPONSE: Indicates to get the network addresses of all devices that are associated with the network address of the target device
nStartIndex	In the case where the request type is ZDP_EXTENDED_RESPONSE, this is the start index of the device list that is associated with the relative address
bTxOptions	Specifies transmission options for request. The following values can be logical ORed together: APS_TXOPTION_NONE SECURITY_ENABLE_TRANSMISSION USE_NWK_KEY ACKNOWLEDGED_TRANSMISSION

Returns

None

Example

```
void zdpNwkAddrReqExample(void)
{
   MAC_ExtAddr_s sAddrInterest;

   sAddrInterest.u32H = 0;
   sAddrInterest.u32L = 1;

   zdpNwkAddrReq(sAddrInterest,
                 ZDP_SINGLE_DEVICE_RESPONSE,
                 0,
                 APS_TXOPTION_NONE);
}
```

zdpNwkAddrRsp (Cluster Id: 0x80)

Note: This response is passed to the user application by a call from the stack to the application through the function **JZA_vZdpResponse()**. The parameters of the response are contained as an array within the payload passed to the function. For more details, refer to Section 1.5.

Description

This response reports the status of a Network Address Request.

Parameters

These parameters are contained in an array within the payload passed to the function **JZA_vZdpResponse()**.

Parameter	Array Pos'n	Type	Valid Range	Description
eZdpStatus	0	**ZDP_Status**	ZDP_SUCCESS_VALID ZDP_INV_REQUESTTYPE ZDP_DEVICE_NOT_FOUND	Status of Network Address Request
sExtAddr	1 to 8	**MAC_ExtAddr_s**	64-bit IEEE address	IEEE address of responding device
u16AddrRemote	9 to 10	**UINT16**	16-bit network address	Network address of responding device
u8AssocDevCount	11	**UINT8**	0x00 to 0xFF	Number of associated devices
nStartIndex	12	**UINT8**	0x00 to 0xFF	Start Index field value of request
**pu16AddrList*	13 to n	**UINT16**		List of nodes associated with responding device

zdpIeeeAddrReq (Cluster Id: 0x01)

```
void zdpIeeeAddrReq(UINT16 u16AddrInterest,
                    REQUEST_TYPE eReqType,
                    UINT8 nStartIndex,
                    APS_TXOPTIONS bTxOptions);
```

Description

This function submits an Extended (IEEE) Address Request - it is used to request the IEEE address of the device with the specified network address (when the network address of the target device is known). The destination addressing used is Unicast.

Parameters

u16AddrInterest	16-bit network address of the target device.
eReqType	One of the following: ZDP_SINGLE_DEVICE_RESPONSE: Indicates to get the IEEE address of the target device only ZDP_EXTENDED_RESPONSE: Indicates to get the IEEE addresses of all devices that are associated with the network address of the target device
nStartIndex	In the case where the request type is ZDP_EXTENDED_RESPONSE, this is the start index of the device list that is associated with the relative address
bTxOptions	Specifies transmission options for request. The following values can be logical ORed together: APS_TXOPTION_NONE SECURITY_ENABLE_TRANSMISSION USE_NWK_KEY ACKNOWLEDGED_TRANSMISSION

Returns

None

Example

```
void zdpIeeeAddrReqExample(void)
{
    UINT16 u16NwkAddrInterest;

    /* Send request to node with network address of 0x0001 */
    u16NwkAddrInterest = 0x0001;
    zdpIeeeAddrReq(u16NwkAddrInterest,
                   ZDP_SINGLE_DEVICE_RESPONSE,
                   0,
                   APS_TXOPTION_NONE);
}
```

zdpIeeeAddrRsp (Cluster Id: 0x81)

Note: This response is passed to the user application by a call from the stack to the application through the function **JZA_vZdpResponse()**. The parameters of the response are contained in an array within the payload passed to the function. For more details, refer to Section 1.5.

Description

This response reports the status of an Extended (IEEE) Address Request response. The response is created by the device with a network address that matches the network address field of the request.

Parameters

These parameters are contained in an array within the payload passed to the function **JZA_vZdpResponse()**.

Parameter	Array Pos'n	Type	Valid Range	Description
eZdpStatus	0	**ZDP_Status**	ZDP_SUCCESS_VALID ZDP_INV_REQUESTTYPE ZDP_DEVICE_NOT_FOUND	Status of Extended Address Request
sExtAddr	1 to 8	**MAC_ExtAddr_s**	64-bit IEEE address	IEEE address of responding device
u16AddrRemote	9 to 10	**UINT16**	16-bit network address	Network address of responding device
u8AssocDevCount	11	**UINT8**	0x00 to 0xFF	Number of associated devices
nStartIndex	12	**UINT8**	0x00 to 0xFF	Start Index field value of request
**pu16AddrList*	13 to n	**UINT16**		List of nodes associated with responding device

3 Service Discovery API

The Service Discovery API is used to obtain information about network devices; for example, the information contained in the device descriptors.

The request functions and corresponding responses are listed below, along with their page references.

zdpNodeDescReq (Cluster Id: 0x02)

void zdpNodeDescReq(UINT16 *u16AddrInterest*,
APS_TXOPTIONS *bTxOptions*);

Description

This function submits a Node Descriptor Request - it requests the Node Descriptor of the device with the specified network address. The destination addressing is Unicast.

Parameters

u16AddrInterest	16-bit network address of the target device
bTxOptions	Specifies transmission options for request. The following values can be logical ORed together: APS_TXOPTION_NONE SECURITY_ENABLE_TRANSMISSION USE_NWK_KEY ACKNOWLEDGED_TRANSMISSION

Returns

None

Example

```
void zdpNodeDescReqExample(void)
{
    UINT16 u16AddrInterest;
    u16AddrInterest = 0x0001;
    zdpNodeDescReq(u16AddrInterest,
                   APS_TXOPTION_NONE);
}
```

zdpNodeDescRsp (Cluster Id: 0x82)

Note: This response is passed to the user application by a call from the stack to the application through the function **JZA_vZdpResponse()**. The parameters of the response are contained in an array within the payload passed to the function. For more details, refer to Section 1.5.

Description

This response reports the status of a Node Descriptor Request. The response is created by the device with the network address that matches the Network of Interest field in the Node Descriptor Request.

Parameters

These parameters are contained in an array within the payload passed to the function **JZA_vZdpResponse()**.

Parameter	Array Pos'n	Type	Valid Range	Description
eZdpStatus	0	**ZDP_Status**	ZDP_SUCCESS_VALID ZDP_INV_REQUESTTYPE ZDP_DEVICE_NOT_FOUND ZDP_NO_DESCRIPTOR	Status of Node Descriptor Request
u16AddrInterest	1 to 2	**UINT16**	16-bit network address	Network address of responding device
**pDesc*	3 to n	**AF_NODE_DESCRIPTOR**		Refer to AF Node Descriptor in related document [R3]

zdpPowerDescReq (Cluster Id: 0x03)

void zdpPowerDescReq(UINT16 *u16AddrInterest*,
APS_TXOPTIONS *bTxOptions*);

Description

This function submits a Power Descriptor Request - it requests the Power Descriptor of the device with the specified network address. The destination addressing used is Unicast.

Parameters

u16AddrInterest	16-bit network address of the target device
bTxOptions	Specifies transmission options for request. The following values can be logical ORed together: APS_TXOPTION_NONE SECURITY_ENABLE_TRANSMISSION USE_NWK_KEY ACKNOWLEDGED_TRANSMISSION

Returns

None

Example

```
void zdpPowerDescReqExample(void)
{
    UINT16 u16AddrInterest;
    u16AddrInterest = 0x0001;
    zdpPowerDescReq (u16AddrInterest,
                     APS_TXOPTION_NONE);
}
```

zdpPowerDescRsp (Cluster Id: 0x83)

Note: This response is passed to the user application by a call from the stack to the application through the function **JZA_vZdpResponse()**. The parameters of the response are contained in an array within the payload passed to the function. For more details, refer to Section 1.5.

Description

This response reports the status of a Power Descriptor Request. The response is created by the device with the network address that matches the Network of Interest field in the Power Descriptor Request.

Parameters

These parameters are contained in an array within the payload passed to the function **JZA_vZdpResponse()**.

Parameter	Array Pos'n	Type	Valid Range	Description
eZdpStatus	0	**ZDP_Status**	ZDP_SUCCESS_VALID, ZDP_INV_REQUESTTYPE, ZDP_DEVICE_NOT_FOUND, ZDP_NO_DESCRIPTOR	Status of Power Descriptor Request
u16AddrInterest	1 to 2	**UINT16**	16-bit network address	Network address of responding device
**pDesc*	3 to n	**AF_NODE_POWER_DESCRIPTOR**		Refer to AF Power Descriptor in related document [R3]

zdpSimpleDescReq (Cluster Id: 0x04)

void zdpSimpleDescReq(UINT16 *u16AddrInterest*,
UINT8 *u8EndPoint*,
APS_TXOPTIONS *bTxOptions*);

Description

This function submits a Simple Descriptor Request - it requests the Simple Descriptor of the device with the specified network address. The destination addressing used is Unicast.

Parameters

u16AddrInterest	16-bit network address of the target device
u8EndPoint	Endpoint of target device
bTxOptions	Specifies transmission options for request. The following values can be logical ORed together: APS_TXOPTION_NONE SECURITY_ENABLE_TRANSMISSION USE_NWK_KEY ACKNOWLEDGED_TRANSMISSION

Returns

None

Example

```
void zdpSimpleDescReqExample(void)
{
    UINT16 u16AddrInterest;
    UINT8 u8EndPoint;
    u16AddrInterest = 0x0001;
    u8EndPoint = 0xF0;
    zdpSimpleDescReq (u16AddrInterest,
                      u8EndPoint,
                      APS_TXOPTION_NONE);
}
```

zdpSimpleDescRsp (Cluster Id : 0x84)

Note: This response is passed to the user application by a call from the stack to the application through the function **JZA_vZdpResponse()**. The parameters of the response are contained in an array within the payload passed to the function. For more details, refer to Section 1.5.

Description

This response reports the status of a Simple Descriptor Request. The response is created by the device with the network address that matches the Network of Interest field in the Simple Descriptor Request.

Parameters

These parameters are contained in an array within the payload passed to the function **JZA_vZdpResponse()**.

Parameter	Array Pos'n	Type	Valid Range	Description
eZdpStatus	0	**ZDP_Status**	ZDP_SUCCESS_VALID, ZDP_INV_REQUESTTYPE, ZDP_DEVICE_NOT_FOUND, ZDP_INV_EP, ZDP_NOT_ACTIVE, ZDP_NO_DESCRIPTOR	Status of Simple Descriptor Request
u16AddrInterest	1 to 2	**UINT16**	16-bit network address	Network address of responding device
nDescLength	3	**UINT8**	0x00 to 0xFF	Length of Simple Descriptor
**pDesc*	4	**UINT8**		Refer to AF Simple Descriptor in related document [R3]

zdpActiveEpReq (Cluster Id: 0x05)

void zdpActiveEpReq(UINT16 *u16AddrInterest*,
APS_TXOPTIONS *bTxOptions*);

Description

This function submits an Active Endpoint Request - it requests the active endpoint of the device with the specified network address. The destination addressing used is Unicast.

Parameters

u16AddrInterest	16-bit network address of the target device
bTxOptions	Specifies transmission options for request. The following values can be logical ORed together: APS_TXOPTION_NONE SECURITY_ENABLE_TRANSMISSION USE_NWK_KEY ACKNOWLEDGED_TRANSMISSION

Returns

None

Example

```
void zdpActiveEPReqExample(void)
{
    UINT16 u16AddrInterest;
    u16AddrInterest = 0x0001;
    zdpActiveEpReq (u16AddrInterest,
                    APS_TXOPTION_NONE);
}
```

zdpActiveEpRsp (Cluster Id: 0x85)

Note: This response is passed to the user application by a call from the stack to the application through the function **JZA_vZdpResponse()**. The parameters of the response are contained in an array within the payload passed to the function. For more details, refer to Section 1.5.

Description

This response reports the status of an Active Endpoint Request. The response is created by the device with the network address that matches the Network of Interest field in the Active Endpoint Request.

Parameters

These parameters are contained in an array within the payload passed to the function **JZA_vZdpResponse()**.

Parameter	Array Pos'n	Type	Valid Range	Description
eZdpStatus	0	**ZDP_Status**	ZDP_SUCCESS_VALID, ZDP_INV_REQUESTTYPE, ZDP_DEVICE_NOT_FOUND	Status of Active Endpoint Request
u16AddrInterest	1 to 2	**UINT16**	16-bit network address	Network address of responding device
nActiveEP	3	**UINT8**	0x00 to 0xFF	Number of active endpoints
**pActiveEPList*	4	**UINT8**		Active Endpoint list

zdpMatchDescReq (Cluster Id: 0x06)

```
void zdpMatchDescReq (UINT16 u16AddrInterest,
                      UINT16 u16ProfileID,
                      UINT8  u8InClusterCount,
                      UINT8 *pau8InClusterList,
                      UINT8  u8OutClusterCount,
                      UINT8 *pau8OutClusterList,
                      APS_TXOPTIONS  bTxOptions);
```

Description

This function submits a Match Descriptor Request - it requests the endpoint of the target device that matches with the cluster's endpoint. The destination addressing used is either Broadcast or Unicast. In the case of Unicast, the *u16AddrInterest* parameter is the network address of the destination device; for Broadcast, it is equal to 0xFFFF.

Parameters

u16AddrInterest	16-bit network address of target device for Unicast (is set to 0xFFFF for Broadcast)
u16ProfileID	Profile ID which matches with the target device (in range 0 to 0xFFFF)
u8InClusterCount	Number of entries in Input Cluster List (in range 0 to 0xFF)
**pau8InClusterList*	Pointer to Input Cluster List of remote device (length of list is 1 byte x *nInClusters*)
u8OutClusterCount	Number of entries in Output Cluster List (in range 0 to 0xFF)
**pau8OutClusterList*	Pointer to Output Cluster List of remote device (length of list is 1 byte x *nOutClusters*)
bTxOptions	Specifies transmission options for request. The following values can be logical ORed together: APS_TXOPTION_NONE SECURITY_ENABLE_TRANSMISSION USE_NWK_KEY ACKNOWLEDGED_TRANSMISSION

Returns

None

Example

```
void zdpMatchDescReqExample(void)
{
    UINT16 u16AddrInterest;
    UINT8 u8InClusterCnt = 2;
    UINT8 au8InClusterList[2] = {0x01, 0x02};
    UINT8 u8OutClusterCnt = 3;
    UINT8 au8OutClusterList[3] = {0x06, 0x07, 0x08};
    u16AddrInterest = 0x0001;

    zdpMatchDescReq(u16AddrInterest,
                    0xA5A5,
                    u8InClusterCnt,
                    (UINT8 *)&au8InClusterList,
                    u8OutClusterCnt,
                    (UINT8 *)&au8OutClusterList,
                    APS_TXOPTION_NONE);
}
```

zdpMatchDescRsp (Cluster Id: 0x86)

> **Note**: This response is passed to the user application by a call from the stack to the application through the function **JZA_vZdpResponse()**. The parameters of the response are contained in an array within the payload passed to the function. For more details, refer to Section 1.5.

Description

This response reports the status of a Match Descriptor Request. In the case where a profile ID of a device matches the profile ID field of the request, an endpoint list is created containing entries which match those in the *InClusterList* or *OutClusterList* parameter of the request. This list is sent to the device that made the request.

Parameters

These parameters are contained in an array within the payload passed to the function **JZA_vZdpResponse()**.

Parameter	Array Pos'n	Type	Valid Range	Description
eZdpStatus	0	**ZDP_Status**	ZDP_SUCCESS_VALID ZDP_INV_REQUESTTYPE ZDP_DEVICE_NOT_FOUND ZDP_NO_DESCRIPTOR	Status of Match Descriptor Request
u16AddrInterest	1 to 2	**UINT16**	16-bit network address	Network address of responding device
nMatchLength	3	**UINT8**	0x00 to 0xFF	Number of matched endpoints
**pMatchList*	4	**UINT8**		Matched Endpoint List

zdpUserDescReq (Cluster Id: 0x11)

void zdpUserDescReq(UINT16 *u16AddrInterest*,
TX_OPTIONS *bTxOptions*);

Description

This function submits a User Descriptor Request - it requests the User Descriptor of the device with the specified network address. The destination addressing used is Unicast.

Parameters

u16AddrInterest	16-bit network address of the target device
bTxOptions	Specifies transmission options for request. The following values can be logical ORed together: APS_TXOPTION_NONE SECURITY_ENABLE_TRANSMISSION USE_NWK_KEY ACKNOWLEDGED_TRANSMISSION

Returns

None

Example

```
void zdpUserDescReqExample(void)
{
    UINT16 u16AddrInterest;
    u16AddrInterest = 0x0001;
    zdpUserDescReq(u16AddrInterest, APS_TXOPTION_NONE);
}
```

zdpUserDescRsp (Cluster Id: 0x91)

> **Note**: This response is passed to the user application by a call from the stack to the application through the function **JZA_vZdpResponse()**. The parameters of the response are contained in an array within the payload passed to the function. For more details, refer to Section 1.5.

Description

This response reports the status of a User Descriptor Request. The response is created by the device with the network address that matches the Network of Interest field in the User Descriptor Request.

Parameters

These parameters are contained in an array within the payload passed to the function **JZA_vZdpResponse()**.

Parameter	Array Pos'n	Type	Valid Range	Description
eZdpStatus	0	**ZDP_Status**	ZDP_SUCCESS_VALID ZDP_INV_REQUESTTYPE ZDP_DEVICE_NOT_FOUND ZDP_NOT_SUPPORTED ZDP_NO_DESCRIPTOR	Status of User Descriptor Request
u16AddrInterest	1 to 2	**UINT16**	16-bit network address	Network address of responding device
nDescLength	3	**UINT8**	0x00 to 0xFF	Length of User Descriptor
**psUserDesc*	4 to n	**AF_USER_DESCRIPTOR**		Refer to AF User Descriptor in related document [R3]

zdpEndDeviceAnnce (Cluster Id: 0x13)

void zdpEndDeviceAnnce(UINT16 *u16DestAddress*,
APS_TXOPTIONS *bTxOptions*);

Description

This function allows an End Device to send its network address and IEEE address to another device. The destination addressing can be Unicast or Broadcast (0xFFFF).

Parameters

u16DestAddress	16-bit network address of destination device
bTxOptions	Specifies transmission options for request. The following values can be logical ORed together: APS_TXOPTION_NONE SECURITY_ENABLE_TRANSMISSION USE_NWK_KEY ACKNOWLEDGED_TRANSMISSION

Returns

None

Example

```
void zdpEndDeviceAnnceExample(void)
{
    zdpEndDeviceAnnce(0x2F5E, APS_TXOPTION_NONE);
}
```

zdpUserDescSet (Cluster Id: 0x14)

void zdpUserDescSet(UINT16 *u16AddrInterest*,
**AF_USER_DESCRIPTOR ***pUserDesc*,
APS_TXOPTIONS *bTxOptions*);

Description

This function sets the User Descriptor of the device with the specified network address. The destination addressing used is Unicast.

Parameters

u16AddrInterest	16-bit network address of target device
**pUserDesc*	Pointer to User Descriptor - refer to AF User Descriptor in related document [R3]
bTxOptions	Specifies transmission options for request. The following values can be logical ORed together: APS_TXOPTION_NONE SECURITY_ENABLE_TRANSMISSION USE_NWK_KEY ACKNOWLEDGED_TRANSMISSION

Returns

None

Example

```
void zdpUserDescSetExample(void)
{
    UINT16 u16AddrInterest;
    AF_USER_DESCRIPTOR hAfUserDesc;
    UINT8 u8UserDesc[6] = {'J', 'E', 'N', 'N', 'I', 'C'};
    u16AddrInterest = 0x0001;
    hAfUserDesc.u8UserDescLen = 0x06;
    VBOSReverseMemCpy(&hAfUserDesc.u8UserDesc,
              &u8UserDesc,
              hAfUserDesc.u8UserDescLen);

    zdpUserDescSet(u16AddrInterest,
                   &hAfUserDesc,
                   APS_TXOPTION_NONE);
}
```

zdpUserDescConf (Cluster Id: 0x94)

Note: This response is passed to the user application by a call from the stack to the application through the function **JZA_vZdpResponse()**. The parameters of the response are contained in an array within the payload passed to the function. For more details, refer to Section 1.5.

Description

This response reports the status of the User Descriptor.

Parameters

These parameters are contained in an array within the payload passed to the function **JZA_vZdpResponse()**.

Parameter	Array Pos'n	Type	Valid Range	Description
eZdpStatus	0	**ZDP_Status**	ZDP_SUCCESS_VALID ZDP_NOT_SUPPORTED	Status of Discovery Request
u16AddrInterest	1 to 2	**UINT16**	16-bit network address	Network address of responding device

Returns

None

4 Binding API

The Binding API is used for binding and unbinding network devices.

Note: For indirect binding, since the binding table is owned by the Co-ordinator, the destination address of binding-related requests is always the Co-ordinator's address (0x0000). For information on direct and indirect binding, refer to the *ZigBee Stack User Guide*.

The request functions and corresponding responses are listed below, along with their page references.

zdpEndDeviceBindReq (Cluster Id: 0x20)

```
void zdpEndDeviceBindReq(UINT16 u16AddrBindingTarget,
                         UINT8 u8EndPoint,
                         UINT16 u16ProfileID,
                         UINT8 u8InClusterCount,
                         UINT8 *pau8InClusterList,
                         UINT8 u8OutClusterCount,
                         UINT8 *pau8OutClusterList,
                         APS_TXOPTIONS bTxOptions);
```

Description

This function submits an End Device Bind Request to the Co-ordinator. If the Co-ordinator receives two such requests from different devices within 5 seconds of each other, the Co-ordinator compares the parameters *u16ProfileID, pau8InClusterList* and *pau8OutClusterList* of the two requests. A match is achieved when the *u16ProfileID* from the two requests match and there is at least one Cluster ID which is included in the input cluster list of one request, and in the output cluster list of the other. Next, the binding table is searched for an existing entry that includes the same IEEE addresses as the two devices and any one of the matched Cluster IDs. If one is not found, it is created. If one is found, the entry is deleted. The Co-ordinator will then send an End Device Bind Response indicating success or failure.

This request is Unicast to the binding target, and this should be set to the address of the Co-ordinator. A request cannot come from the Co-ordinator itself; it can only come from devices that are End Devices or Routers.

Parameters

u16AddrBindingTarget	16-bit network address of Co-ordinator (0x0000)
u8EndPoint	Endpoint number on local device (in range 0 to 0xF0)
u16ProfileID	Profile identifier of endpoint (in range 0 to 0xFFFF)
u8InClusterCount	Number of input clusters on remote device (in range 0 to 0xFF)
**pauInClusterList*	Pointer to Input Cluster List on remote device (length of list is 1 byte x *u8InClusterCount*)
u8OutClusterCount	Number of output clusters on remote device (in range 0 to 0xFF)
**pau8OutClusterList*	Pointer to Output Cluster List on remote device (length of list is 1 byte x *u8OutClusterCount*)
bTxOptions	Specifies transmission options for request. The following values can be logical ORed together: APS_TXOPTION_NONE SECURITY_ENABLE_TRANSMISSION USE_NWK_KEY ACKNOWLEDGED_TRANSMISSION

Returns

None

Example

The following code is implemented in two different devices. Once each device has connected to the network, the two devices each make their respective End_Device_Bind_Request to the Co-ordinator.

```
/* This code would appear in the application running on Device 1
*/
void zdpEndDeviceBindReqExample_Device1(void)
{
    UINT16 u16BindindTarget = 0x0000;
    UINT8  u8LocalEndpoint = 0x01;
    UINT16 u16ExampleProfileID = 0x0014;
    UINT8  u8RemoteInClusterCnt = 2;
    UINT8  au8RemoteInClusterList[2] = {0x01, 0x02};
    UINT8  u8RemoteOutClusterCnt = 3;
    UINT8  au8RemoteOutClusterList[3] = {0x06, 0x07, 0x08};

    zdpEndDeviceBindReq(u16BindindTarget,
                        u8LocalEndpoint,
                        u16ExampleProfileID,
                        u8RemoteInClusterCnt,
                        (UINT8 *)&au8RemoteInClusterList,
                        u8RemoteOutClusterCnt,
                        (UINT8 *)&au8RemoteOutClusterList,
                        APS_TXOPTION_NONE);
}
/* This code would appear in the application running on Device 2
*/
void zdpEndDeviceBindReqExample_Device2(void)
// Coding in End Device 02
{
    UINT16 u16BindindTarget = 0x0000;
    UINT8 u8LocalEndpoint = 0x30;
    UINT16 u16ExampleProfileID = 0x0014;
    UINT8 u8RemoteInClusterCnt = 3;
    UINT8 au8RemoteInClusterList[3] = {0x06, 0x07, 0x08};
    UINT8 u8RemoteOutClusterCnt = 2;
    UINT8 au8RemoteOutClusterList[2] = {0x01, 0x02};

    zdpEndDeviceBindReq(u16BindindTarget,
                        u8LocalEndpoint,
                        u16ExampleProfileID,
                        u8RemoteInClusterCnt,
                        (UINT8 *)&au8RemoteInClusterList,
                        u8RemoteOutClusterCnt,
                        (UINT8 *)&au8RemoteOutClusterList,
                        APS_TXOPTION_NONE);
}
```

zdpEndDeviceBindRsp (Cluster Id: 0xA0)

Note: This response is passed to the user application by a call from the stack to the application through the function **JZA_vZdpResponse()**. The parameter of the response is contained in an array within the payload passed to the function. For more details, refer to Section 1.5.

Description

This response reports the status of an End Device Bind Request.

Parameters

This parameter is contained in an array within the payload passed to the function **JZA_vZdpResponse()**.

Parameter	Array Pos'n	Type	Valid Range	Description
eZdpStatus	0	**ZDP_Status**	ZDP_SUCCESS_VALID ZDP_NOT_SUPPORTED ZDP_INV_EP ZDP_TIMEOUT ZDP_NO_MATCH	Status of End Device Bind Request

zdpBindReq (Cluster Id: 0x21)

```
void zdpBindReq(MAC_ExtAddr_s sExtAddrSrc,
                UINT8 u8SrcEP,
                UINT8 eClusterID,
                MAC_ExtAddr_s sExtAddrDst,
                UINT8 u8DstEP,
                APS_TXOPTIONS bTxOptions);
```

Description

This function sends a Bind Request to the Co-ordinator. The Co-ordinator will try to create a new entry in the binding table based on the supplied IEEE addresses, endpoints and Cluster ID. It will then send a Bind Response indicating success or failure.

Parameters

sExtAddrSrc	64-bit IEEE (MAC) address of source device for binding
u8SrcEP	Endpoint number of source device for binding (in range 0x01 to 0xF0)
eClusterID	Cluster ID for binding (in range 0 to 0xFF)
sExtAddrDst	64-bit IEEE (MAC) address of destination device for binding
u8DstEP	Endpoint of destination device for binding (in range 0x01 to 0xF0)
bTxOptions	Specifies transmission options for request. The following values can be logical ORed together: APS_TXOPTION_NONE SECURITY_ENABLE_TRANSMISSION USE_NWK_KEY ACKNOWLEDGED_TRANSMISSION

Returns

None

Example

```
void zdpBindReqExample (void)
{
    MAC_ExtAddr_s sAddrInterest01;
    MAC_ExtAddr_s sAddrInterest02;
    UINT8 u8SrcEP = 0x01;
    UINT8 u8DstEP = 0x02;
    UINT8 u8ClusterId = 0x30;

    sAddrInterest01.u32H = 0xffee1234;
    sAddrInterest01.u32L = 0xffee5678;
    sAddrInterest02.u32H = 0xaabb3456;
    sAddrInterest02.u32L = 0xaabb1289;

    zdpBindReq(sAddrInterest01,
               u8SrcEP,
               u8ClusterId,
               sAddrInterest02,
               u8DstEP,
               APS_TXOPTION_NONE);
}
```

zdpBindRsp (Cluster Id: 0xA1)

Note: This response is passed to the user application by a call from the stack to the application through the function **JZA_vZdpResponse()**. The parameter of the response is contained in an array within the payload passed to the function. For more details, refer to Section 1.5.

Description

This function reports the status of a Bind Request.

Parameters

This parameter is contained in an array within the payload passed to the function **JZA_vZdpResponse()**.

Parameter	Array Pos'n	Type	Valid Range	Description
eZdpStatus	0	**ZDP_Status**	ZDP_SUCCESS_VALID ZDP_NOT_SUPPORTED ZDP_TABLE_FULL	Status of Bind Request

zdpUnbindReq (Cluster Id: 0x22)

```
void zdpUnbindReq(MAC_ExtAddr_s sExtAddrSrc,
                  UINT8 u8SrcEP,
                  ZdpPrimitiveClusterID eClusterID,
                  MAC_ExtAddr_s sExtAddrDst,
                  UINT8 u8DstEP,
                  APS_TXOPTIONS bTxOptions);
```

Description

This function sends an Unbind Request to the Co-ordinator. The Co-ordinator will try to delete an existing entry in the binding table based on the supplied IEEE addresses, endpoints and Cluster ID. It will then send an Unbind Response indicating success or failure.

Parameters

sExtAddrSrc	64-bit IEEE (MAC) address of source device for unbinding
u8SrcEP	Endpoint number of source device for unbinding (in range 0 to 0xF0)
eClusterID	Cluster ID for unbinding (in range 0 to 0xFF)
sExtAddrDst	64-bit IEEE (MAC) address of destination device for unbinding
u8DstEP	Endpoint of destination device for unbinding (in range 0x01 to 0xF0)
bTxOptions	Specifies transmission options for request. The following values can be logical ORed together: APS_TXOPTION_NONE SECURITY_ENABLE_TRANSMISSION USE_NWK_KEY ACKNOWLEDGED_TRANSMISSION

Returns

None

Example

```
void zdpUnbindReqExample(void)
{
    MAC_ExtAddr_s sAddrInterest01;
    MAC_ExtAddr_s sAddrInterest02;
    UINT8 u8SrcEP = 0x01;
    UINT8 u8DstEP = 0x02;
    UINT8 u8ClusterId = 0x30;
    sAddrInterest01.u32H = 0xffee1234;
    sAddrInterest01.u32L = 0xffee5678;
    sAddrInterest02.u32H = 0xaabb3456;
    sAddrInterest02.u32L = 0xaabb1289;
    zdpUnbindReq(sAddrInterest01,
                 u8SrcEP,
                 u8ClusterId,
                 sAddrInterest02,
                 u8DstEP,
                 APS_TXOPTION_NONE);
}
```

zdpUnbindRsp (Cluster Id: 0xA2)

Note: This response is passed to the user application by a call from the stack to the application through the function **JZA_vZdpResponse()**. The parameter of the response is contained in an array within the payload passed to the function. For more details, refer to Section 1.5.

Description

This function reports the status of an Unbind Request.

Parameters

This parameter is contained in an array within the payload passed to the function **JZA_vZdpResponse()**.

Parameter	Array Pos'n	Type	Valid Range	Description
eZdpStatus	0	**ZDP_Status**	ZDP_SUCCESS_VALID ZDP_NOT_SUPPORTED ZDP_NO_ENTRY	Status of Unbind Request

Revision History

Version	Date	Description
1.0	15-May-2006	First release
1.1	14-Sep-2006	BOSMemCpy changed to BOSReverseMemCpy in code fragments
1.2	10-Dec-2006	Updated with changes in the Jennic ZigBee Stack v1.5
1.3	30-Mar-2007	Updated with changes in the Jennic ZigBee Stack v1.7
1.4	04-June-2007	Arrays in ZDP responses corrected
1.5	03-Aug-2007	Updated code fragments with MAC_ExtAddr_s and vBOSReverseMemCopy, some common parameters removed, modified enumerations SINGLE_DEVICE_RESPONSE and EXTENDED_RESPONSE

Important Notice

Jennic reserves the right to make corrections, modifications, enhancements, improvements and other changes to its products and services at any time, and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders, and should verify that such information is current and complete. All products are sold subject to Jennic's terms and conditions of sale, supplied at the time of order acknowledgment. Information relating to device applications, and the like, is intended as suggestion only and may be superseded by updates. It is the customer's responsibility to ensure that their application meets their own specifications. Jennic makes no representation and gives no warranty relating to advice, support or customer product design.

Jennic assumes no responsibility or liability for the use of any of its products, conveys no license or title under any patent, copyright or mask work rights to these products, and makes no representations or warranties that these products are free from patent, copyright or mask work infringement, unless otherwise specified.

Jennic products are not intended for use in life support systems/appliances or any systems where product malfunction can reasonably be expected to result in personal injury, death, severe property damage or environmental damage. Jennic customers using or selling Jennic products for use in such applications do so at their own risk and agree to fully indemnify Jennic for any damages resulting from such use.

Jennic
TECHNOLOGY FOR A CHANGING WORLD

Corporate Headquarters
Furnival Street
Sheffield
S1 4QT
United Kingdom

Tel +44 (0)114 281 2655
Fax +44 (0)114 281 2951
E-mail info@jennic.com

Japan Sales Office
Osakaya building 4F
1-11-8 Higashigotanda
Shinagawa-ku, Tokyo
141-0022, Japan

Tel +81 3 5449 7501
Fax +81 3 5449 0741
E-mail info@jp.jennic.com

Taiwan Sales Office
19F-1, 182, Sec.2
Tun Hwa S. Road
Taipei 106
Taiwan

Tel +886 2 2735 7357
Fax +886 2 2739 5687
E-mail info@tw.jennic.com

United States Sales Office
1322 Scott Street, Suite 203
Point Loma
CA 92106
USA

Tel +1 619 223 2215
Fax +1 619 223 2081
E-mail info@us.jennic.com

United States Sales Office
1060 First Avenue, Suite 400
King of Prussia
PA 19406
USA

Tel +1 619 223 2215
Fax +1 619 223 2081
E-mail info@us.jennic.com

Korean Sales Office
701, 7th Floor, Kunam Building
831-37, Yeoksam-Dong
Kangnam-ku
Seoul 135-080
Korea

Tel +82 2 552 5325
Fax +82 2 3453 8802
E-mail info@kr.jennic.com