ZigBee Input Device Application Profile

Reference Manual

Document Number: ZIDAPRM
Rev. 1.0
2/2012

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Contents

About This Book

This reference manual describes the Freescale ZigBee Input Device (ZID) Application Profile implementation which simplifies application development using the ZID profile.

The Freescale ZID profile layer resides on top of the BeeStack Consumer layer.

Audience

This reference manual is intended for application designers and users of the BeeStack Consumer protocol stack.

Organization

This document contains the following chapters:

Chapter 1 Freescale ZID Application Profile Overview - Provides an introduction to the Freescale ZID Application Profile implementation.

Chapter 2 Freescale ZID Application Profile Software Usage - Provides a description of the Freescale ZID Application Profile interfaces.

Revision History

The following table summarizes revisions to this manual since the previous release (Rev. 0.0).

Revision History

Date	Description / Location of Changes
March 2012	Minor changes throughout to support March software release.

Conventions

This document uses the following notational conventions:

- Courier monospaced type is used to identify commands, explicit command parameters, code examples, expressions, data types, and directives.
- Italic type is used for emphasis, to identify new terms, and for replaceable command parameters.

Definitions, Acronyms, and Abbreviations

The following list defines the abbreviations used in this document.

API	Application Programming Interface
NWK	Network Layer
NLDE	Network Layer Data Entity
NLME	Network Layer Management Entity
SAP	Service Access Point
ZID	ZigBee Input Device
GDP	Generic Device Profile
PBP	Push Button Pair

Chapter 1
Freescale ZID Application Profile Overview

This document describes the Freescale ZigBee Input Device (ZID) software. This chapter details the primary concept of profile implementation and provides an overview of the available libraries, files and the functionality each of them provides.

1.1 Freescale ZID Application Profile Introduction

The Freescale ZigBee Input Device (ZID) Protocol resides between BeeStack Consumer layer (RF4CE layer) and the application layer. It implements the application programming interfaces (API), messages and functionality for building a wireless communication between two types of devices: a ZID Class device and a ZID Adaptor device.

The ZID Class device communicates with a host (a PC or a TV) via the standard HID class protocol by passing over-the-air report descriptors and data containing information that indicates the user interaction. For example the ZID Class device can be a mouse or a keyboard device. The outgoing information from the ZID Class device is interpreted on reception by the host.

The ZID Adaptor acts as an interface between the RF4CE stack and the host (a PC or a TV). The ZID Adaptor is establishing RF4CE and HID connections between the host and remote node (ZID Class device). The Adaptor processes ZID profile commands arrived from RF4CE stack, passing them to the host and vice versa (via API calls).

The ZID Class device has the following role:

- It is a RF4CE controller or target node.
- Supports originator push-button pairing procedure.
- Supports and maintains one or more connections with ZID Adaptor devices.
- Generates and sends reports to the remote nodes.
- Handles the reporting information received from connected ZID Adaptor devices.
- Handles ZID command frames arrived from ZID Adaptor devices.
- Generates the responses for the ZID Adaptor commands.

The ZID Adaptor device has the following role:

- It is a RF4CE target node.
- Supports recipient push-button pairing procedure.
- Establishes and maintains multiple connections with ZID Class devices.
- Handles the reporting information received from ZID Class devices.

- Sends ZID profile commands (requests) to ZID Class devices.
- Handles the responses received from ZID Class devices.
- Generates reports that are sent to ZID Class devices

The following figure shows the software architecture of an application using the Freescale ZID Application Profile Implementation.

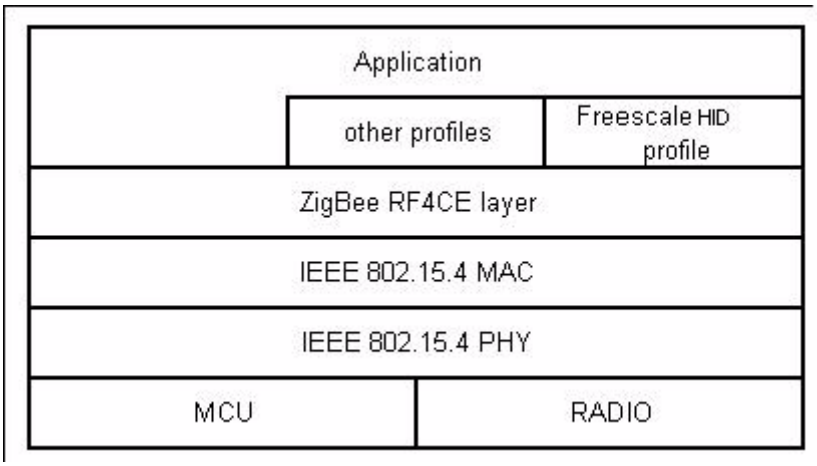

Figure 1-1. Application Software Protocol Stack

1.2 Freescale ZID Application Profile Libraries and Files

This section describes the suite of profile libraries, the files and the functionality they implement. To use the profile, the application must link the library implementing the desired functionality. There are two libraries, one implements ZID Adaptor functionality and other ZID Class device functionality.

NOTE

A node can not act as a ZID Adaptor and ZID Class device simultaneously. The application shall not include both libraries. Use only one library for the desired functionality (ZID Adaptor or ZID Class device).

The application project must include in the compiler's command-line options the following defines which specifies the device type: gZIDProfileClassDevice_d=1 - the node acts as a ZID Class device, gZIDProfileAdaptor_d=1 - the node acts as a ZID Adaptor. These macros have to match the corresponding included library and will enable the API functions of the selected device.

Table 1-1. Profile Libraries and files

Library or file	Description
RF4CE_ZIDProfile_ClassDevice.lib	Contains the following ZID Class device features: - initialize the device - enter device in configuration mode to connect - report data - set/get a local attribute - push/get attributes to/from the remote node - abort a running process - check if the ZID profile is in idle state - remove a connected ZID Adaptor device
RF4CE_ZIDProfile_Adaptor.lib	Contains the following ZID Adaptor features: - initialize the device - enter device in configuration mode to connect - report data - get attributes from a remote ZID Class device - get/set ZID Class device report - handle the data pending for a connected ZID Class device. - abort a running process - check if the profile is in idle state - remove a connected ZID device
ZIDProfileInterface.h	Contains the ZID Profile public macros, type definitions and features API prototypes.
ZIDClassDeviceConfig.h	Contains macros for configuring various local attributes and tables sizes supported by a ZID Class device.
ZIDClassDeviceGlobals.h	Contains the ZID Class device macros, declarations and type definitions for global usage (i.e. local attributes table, reports table, connections information, non-standard descriptor components - format and table).

Table 1-1. Profile Libraries and files

ZIDClassDeviceGlobals.c	Contains the ZID Class device memory declarations for global usage like: local attributes table, local attributes information table, reports table, connections information, the list of non-standard descriptor components etc.
ZIDAdaptorConfig.h	Contains the macros for configuring local attributes and various table sizes (i.e. connections table size, the size of tables to store the non-standard descriptor components of the remote nodes) supported by a ZID Adaptor device.
ZIDAdaptorGlobals.h	Contains the ZID Adaptor macros, declarations and type definitions for global usage (i.e. local attribute information, connections information table, tables for storing the non-standard descriptor components of the remote nodes).
ZIDAdaptorGlobals.c	Contains the ZID Adaptor memory declarations for global usage (i.e. local attributes information, connections information table, tables to store the non-standard descriptor components of the remote nodes, various tables sizes).

Chapter 2
Freescale ZID Application Profile Software Usage

The profile performs ZID Adaptor or ZID Class device tasks depending on what library is included in the application.

2.1 Service Specifications

The profile is implemented on top of the RF4CE network layer, uses RF4CE API calls to pass down data, and handles NLDE indication and confirm messages received from the network layer. The NLDE SAP must be configured to redirect ZID Profile messages to the ZID Profile NLDE SAP, so that these don't erroneously reach the application. The NLME SAP must be configured to redirect the NLME messages, that are used in the push-button pairing procedures, to the Push Button Pair (PBP) NLME SAP. But the application can still access the network layer and receive messages that are not intended for ZID profile or Push Button Pair layers. Refer to the ZID Application Profile User's Guide for a description on how this is accomplished.

Before establishing a connection between a ZID Adaptor device and a ZID Class device, the devices have to be paired using the protocol identifier specified by the ZID Profile Specification. For recipient-side (ZID Adaptor) and originator-side (ZID Class device) push-button pairing procedures refer to the Freescale BeeStack Consumer Reference Manual.

NOTE

The RF4CE network layer handles one request at a time, whether it comes directly from the application or from the profile. Care must be taken to ensure that the application and the profile never make a request to the network layer simultaneously.

For more details about Push Button Pair API calls and messages refer to ZigBee Remote Control (ZRC) Application Profile Reference Manual.

The profile system (including the push-button pairing procedures) is shown in Figure 2-1.

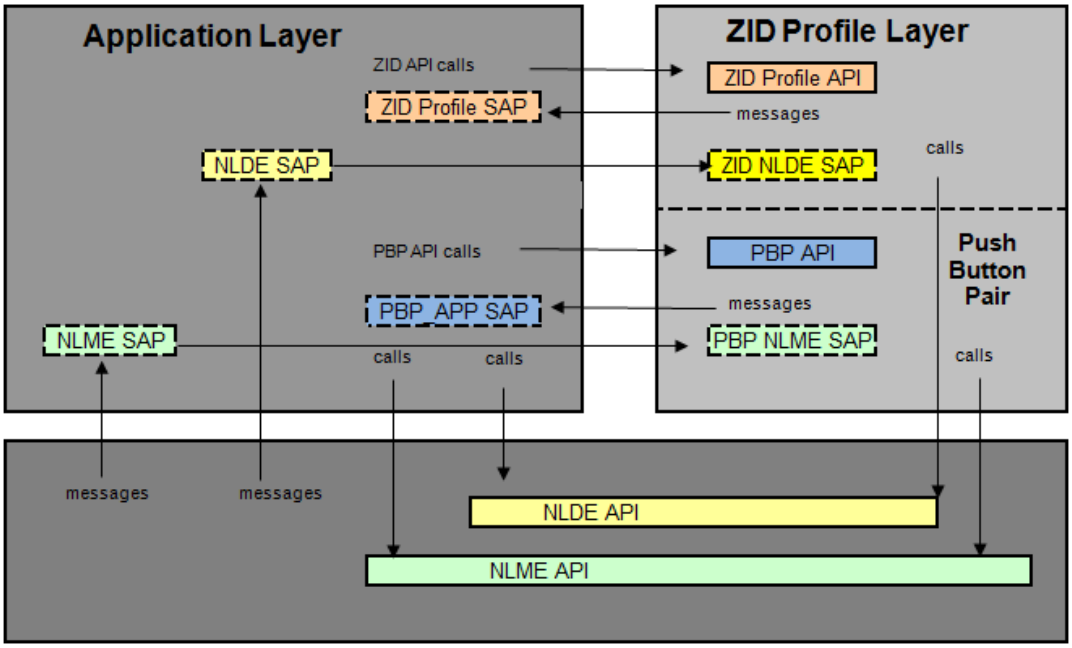

Figure 2-1. Freescale ZID Profile Layer Components and Interfaces

On reception and transmission, depending of what library is included, the profile handles the ZID Profile Specification command frame format. Some of the commands are conformed to the Generic Device Profile (GDP) generic frames format.The ZID profile utilizes the frame commands listed in the following table (refer to ZID and GDP profile specification for more details).

Table 2-1. ZID Profile Specification commands List

Command	ZID Adaptor		ZID Class device	
	Tx	Rx	Tx	Rx
Generic response (GDP command)	X	X	X	X
Configuration complete (GDP command)	-	X	X	-
Heartbeat (GDP command)	-	X	X	-
Get attributes (GDP command)	X	X	X	X
Get attributes response (GDP command)	X	X	X	X
Push attributes (GDP command)	-	X	X	-
Get report	X	-	-	X
Report data	X	X	X	X
Set Report	X	-	-	X

The profile provides specific services (depending of what library is included) that are accessed through the profile layer APIs. The indication and confirm messages are passed to the application layer through the ZID profile SAP.

The ZID profile APIs, messages and returned status (error codes) are described in the following sections of this manual. Some error codes are reused from the network layer and for a detailed description refer to the Freescale BeeStack Consumer Reference Manual.

2.1.1 Freescale ZID Profile APIs

To use ZID Adaptor Profile APIs, the application must link RF4CE_ZIDProfile_Adaptor library and the Adaptor device's files. Similarly, to use ZID Class device APIs, the application must link RF4CE_ZIDProfile_ClassDevice library and the Class device's files.

NOTE

Each ZID API function, that has to send a command over the air, returns the gNWDenied_c status if the ZID Profile is busy with another request or the remote device is not connected. Each feature function also tries to allocate a buffer for sending the command. If the buffer couldn't be allocated, the function exits with gNWNoMemory_c status. Otherwise, the feature function returns gNWSucces_c.

The ZID Adaptor considers a ZID Class device connected if previously the device was successfully paired and the configuration stage, during which the ZID Class device gets and pushes appropriate attribute information to the ZID Adaptor, is successfully completed. For more information refer to the ZID Profile Specification.

The ZID Adaptor device considers a ZID Class device connected when a Configuration Complete command was received during its configuration stage (refer to ZID Profile specification).

In the following sections, the ZID Adaptor profile APIs starts with "ZIDAdp_", the ZID Class device APIs starts with "ZIDClassDev_" and the common function starts with "ZID_". The common functions have the same name for both devices but the functionality may differ, depending on the library included.

2.1.1.1 ZIDAdp_Init

ZIDAdp_Init initializes the ZID Adaptor functionality in ZID Profile layer.

Prototype

```
uint8_t ZIDAdp_Init(void);
```

ZIDAdp_Init has no parameters.

The possible return values for the ZIDAdp_Init API call are shown in the following table.

Table 2-2. ZIDAdp_Init Parameters

Type	Possible Values	Description
bool_t	gNWNoTimers_c gNWSuccess_c	Refer to the below section.

Functionality

ZIDAdp_Init is used to initialize the profile layer, so that the application can execute the ZID Adaptor features.

When ZIDAdp_Init function is called, the ZID Profile configures itself to handle ZID Adaptor features. The function also tries to allocate several timers needed by the features. If the timer could not be allocated, the function exits with gNWNoTimers_c and the initialization is aborted. Otherwise the function returns gNWSuccess_c and the device is now ready to be used

2.1.1.2 ZIDAdp_GetReport

ZIDAdp_GetReport instructs the ZID Profile layer to get a report data from a remote node (from a ZID Class device).

Prototype

```
uint8_t ZIDAdp_GetReport(uint8_t deviceId,
                         uint8_t dataPendingFlag,
                         zidReportType_t reportType,
                         zidReportId_t reportId);
```

The following table specifies the parameters for ZIDAdp_GetReport.

Table 2-3. ZIDAdp_GetReport Parameters

Name	Type	Valid Range	Description
deviceId	uint8_t	0 – (gMaxPairTabelEntries_c -1)	The device identifier.
dataPendingFlag	uint8_t	TRUE or FALSE	It specifies if the device has more data to be sent. This will trigger on the remote node to keep the receiver open for a while.
reportType	zidReportType_t	-	The report type.
reportId	zidReportId_t	0x00 - 0xff	The report identifier.

The possible return values for the ZIDAdp_GetReport API call are shown in the following table.

Table 2-4. ZIDAdp_GetReport API Call Return Values

Type	Possible Values	Description
uint8_t	gNWDenied_c gNWNoMemory_c gNWSuccess_cc	Refer to this chapter note and the below section.

Functionality

ZIDAdp_GetReport is used to configure the ZID Profile for sending the Get Report command and handling the received Report Data indication (refer to ZID Profile specification).

When ZIDAdp_GetReport function is called, the ZID Profile checks if all the conditions (see this chapter note) to begin sending Get Report command are met.

If the function returns gNWSucces_c, the ZID Profile is ready and configured to send the request over the air and receive the report data. The command will be directed to the specified device identifier and the report being requested will be specified by report type and report identifier parameters.

The application is notified of the completion of the Get Report process through a ZID Adaptor Get Report Data Confirm message (zidAdaptorGetReportDataCnf_t).

2.1.1.3 ZIDAdp_SetReportData

The ZIDAdp_SetReportData function instructs the ZID Profile layer to set a report to a remote node (to a ZID Class device).

Prototype

```
uint8_t ZIDAdp_SetReportData(
                            uint8_t         deviceId,
                            uint8_t         dataPendingFlag,
                            zidReportType_t reportType,
                            zidReportId_t   reportId,
                            uint8_t         reportDataLen,
                            uint8_t*        pReportData
                          );
```

The following table specifies the parameters for ZIDAdp_SetReportData.

Table 2-5. ZIDAdp_SetReportData Parameters

Name	Type	Valid Range	Description
deviceId	uint8_t	0 – (gMaxPairTabelEntries_c -1)	The device identifier.
dataPendingFlag	uint8_t	TRUE or FALSE	It specifies if the device has more data to be sent. This will signal the remote node to keep the receiver open for a while.
reportType	zidReportType_t	-	The report type.

Table 2-5. ZIDAdp_SetReportData Parameters (continued)

Name	Type	Valid Range	Description
reportId	zidReportId_t	-	The report identifier.
reportDataLen	uint8_t	-	The report data length.
pReportData	uint8_t*	-	Pointer to report data.

The possible return values for the ZIDAdp_SetReportData API call are shown in the following table.

Table 2-6. ZIDAdp_SetReportData API Call Return Values

Type	Possible Values	Description
uint8_t	gNWDenied_c gNWNoMemory_c gNWSuccess_cc	Refer to this chapter note and the below section.

Functionality

ZIDAdp_SetReportData is used to configure the ZID Profile for sending the Set Report command and handling the response (refer to ZID Profile specification).

When ZIDAdp_SetReportData function is called, the ZID Profile checks if all the conditions (see this chapter note) to begin sending Set Report command are met.

If the function returns gNWSucces_c, the ZID Profile is ready and configured to send the command over the air and receive the Generic response command. The command will be directed to the specified device identifier and the report being sent is specified by report type, report identifier and report data length.

The application is notified of the completion of the Set Report process through a ZID Adaptor Set Report Data Confirm message (zidAdaptorSetReportDataCnf_t).

2.1.1.4 ZIDAdp_SetDataPending

The ZIDAdp_SetDataPending function signals the profile that the application layer has or not data pending for a specific connection (a ZID Class device).

Prototype

```
uint8_t ZIDAdp_SetDataPending(uint8_t connectionIndex, bool_t deviceHasDataPending)
```

The following table specifies the parameters for ZIDAdp_SetDataPending.

Table 2-7. ZIDAdp_SetDataPending Parameters

Name	Type	Valid Range	Description
connectionIndex	uint8_t	0 – gAdpMaxNumOfConnections_c	The connection index.
deviceHasDataPending	uint8_t	TRUE or FALSE	The device has or not data pending.

The possible return values for the ZIDAdp_SetDataPending API call are shown in the following table.

Table 2-8. ZIDAdp_SetDataPending API Call Return Values

Type	Possible Values	Description
uint8_t	gNWInvalidParam_c gNWSuccess_c	Refer to this chapter note and the below section.

Functionality

This function sets the deviceHasDataPending field in the specified connection entry. If the Heartbeat indication message is received from the remote node, the ZID profile will pass the message to the application. If the remote node connection entry has the deviceHasDataPending field set to FALSE, the ZID profile will also respond automatically sending a Generic Response frame.

2.1.1.5 ZIDAdp_EnterConfigMode

ZIDAdp_EnterConfigMode allows the ZID Adaptor device to enter in configuration stage to establish a HID connection with a remote node (with a ZID Class device). The nodes have to be paired before entering in configuration mode. The devices are considered to be connected if the configuration stage completes successfully, which means that the Adaptor device knows the specific HID attributes from the remote node and vice-versa.

Prototype

```
uint8_t ZIDAdp_EnterConfigMode(uint8_t deviceId);
```

The following table specifies the parameters for ZIDAdp_EnterConfigMode.

Table 2-9. ZIDAdp_EnterConfigMode Parameters

Name	Type	Valid Range	Description
deviceId	uint8_t	0 – (gMaxPairTabelEntries_c -1)	The device identifier.

The possible return values for the ZIDAdp_EnterConfigMode API call are shown in the following table.

Table 2-10. ZIDAdp_EnterConfigMode API Call Return Values

Type	Possible Values	Description
uint8_t	gNWDenied_c gNWNoMemory_c gNWSuccess_cc	Refer to this chapter note and the below section.

Functionality

ZIDAdp_EnterConfigMode is used to configure the ZID Profile to receive all specific HID attributes from a remote node and establishing a ZID connection.This function should be called when the push button pair process is successfully ended and an application blackout (aplcConfigBlackoutTime) period expires (refer to GDP profile specification).

When ZIDAdp_EnterConfigMode function is called, the ZID Profile checks if all the conditions (see this chapter note) to begin the configuration stage are met. Besides the returns stated in this chapter note, the function also returns the gNWDenied_c status if the nodes are not paired or the ZID Adaptor device has no room to store a new connection.

If the function returns gNWSucces_c, the ZID Profile is ready and enters in configuration stage. In this stage, the node responses to Get Attributes command and receives all the attributes pushed by the remote node. The remote node is specified by the deviceId parameter.

The application is notified of the completion of the Configuration process through a ZID Enter Configuration Confirm message (zidConfigModeCnf_t).

2.1.1.6 ZIDAdp_RemoveConfiguredDevice

ZIDAdp_RemoveConfiguredDevice removes a connected remote node (a ZID Class device).

Prototype

```
uint8_t ZIDAdp_RemoveConfiguredDevice(uint8_t deviceId);
```

The following table specifies the parameters for ZIDAdp_RemoveConfiguredDevice.

Table 2-11. ZIDAdp_RemoveConfiguredDevice Parameters

Name	Type	Valid Range	Description
deviceId	uint8_t	0 – (gMaxPairTabelEntries_c -1)	The device identifier.

The possible return values for the ZIDAdp_RemoveConfiguredDevice API call are shown in the following table.

Table 2-12. ZIDAdp_RemoveConfiguredDevice API Call Return Values

Type	Possible Values	Description
uint8_t	gNWInvalidParam_c gNWSuccess_c	Refer to the below section.

Functionality

When ZIDAdp_RemoveConfiguredDevice function is called, it searches the connection corresponding to the specified device identifier. If the connection is not found, the function returns the gNWInvalidParam_c status. Otherwise, it removes the connection and returns the gNWSuccess_c status. From now on, the Adaptor device will drop the ZID frames received from the removed node. Note that this function doesn't unpair the remote node (for this, use the unpair network function).

2.1.1.7 ZIDClassDev_Init

The ZIDClassDev_Init function performs the initialization needed for the ZID Class device features.

Prototype

```
uint8_t ZIDClassDev_Init(void);
```

ZIDClassDev_Init has no parameters.

The possible return values for the ZIDClassDev_Init API call are shown in the following table.

Table 2-13. ZIDClassDev_Init API Call Return Values

Type	Possible Values	Description
uint8_t	gNWNoTimers_c gNWSuccess_c	Refer to the below section.

Functionality

ZIDClassDev_Init is used to initialize the profile layer, so that the application can execute the ZID Class device features.

When ZIDClassDev_Init function is called, the ZID Profile configures itself to handle ZID Class device features. The function also tries to allocate several timers needed by the features. If the timers could not be allocated, the function exits with gNWNoTimers_c and the initialization is aborted. Otherwise the function returns gNWSuccess_c and the device is now ready to be used.

2.1.1.8 ZIDClassDev_SendReportIdsList

The ZIDClassDev_SendReportIdsList function forms and sends the report data command frame which contains the report data records specified by a list of report identifiers.

Prototype

```
uint8_t ZIDClassDev_SendReportIdsList(uint8_t  deviceId,
                                      zidSendReportIdsListAction_t action,
                                      uint8_t  txOptions,
                                      uint8_t  reportIdsListSize,
                                      uint8_t* pReportIdsList);
```

The following table specifies the parameters for ZIDClassDev_SendReportIdsList.

Table 2-14. ZIDClassDev_SendReportIdsList Parameters

Name	Type	Valid Range	Description
deviceId	uint8_t	0 – (gMaxPairTabelEntries_c -1)	The device Id.
action	zidSendReportIdsListAction_t	-	Takes the following values: - gSendReportFrameOnlyOnce_c - sends the report frame only once. - gRepeatReportFrame_c - repeats the report frame at the specified ReportRepeatInterval attribute. - gStopRepeatReportFrame_c - stops the report frame repetition.
txOptions	uint8_t	-	The transmission options (which are described the next chapter).
reportIdsListSize	uint8_t	-	The list of report identifiers.
pReportIdsList	uint8_t*	-	Pointer to the list of report identifiers.

The possible return values for the ZIDClassDev_SendReportIdsList API call are shown in the following table.

Table 2-15. ZIDClassDev_SendReportIdsList API Call Return Values

Type	Possible Values	Description
uint8_t	gNWDenied_c gNWNoMemory_c gNWSuccess_c gNWInvalidParam_c	Refer this chapter note and the below section.

Functionality

When ZIDClassDev_SendReportIdsList function is called, it starts building the report data command frame using the specified list of report identifiers. If a report ID is not found or the parameters are not valid, it returns the gNWInvalidParam_c status. Setting the “action” parameter, the built report command frame

can be sent only once or repeated at the specified ReportRepeatInterval local attribute. The ZID Profile can repeat only one report command frame. In order to start repeating another report command frame, which can include one or more report data records, the user must stop the current repetition. To stop the current repetition of a report command frame, the ZIDClassDev_SendReportIdsList function have to be called again with the same deviceId parameter and the “action” parameter set to gStopRepeatReportFrame_c; in this case the rest of the parameters are ignored.

The application is notified of the completion of the transmission process through a ZID Send Report Identifies List Confirm message (zidClassSendReportIdsListCnf_t).

2.1.1.9 ZIDClassDev_EnterConfigMode

ZIDClassDev_EnterConfigMode allows the device to enter in configuration stage for connecting a remote node (a ZID Adaptor device). The nodes have to be paired before entering in configuration mode. In configuration stage, the ZID Class device is getting the remote node’s attributes and is pushing all its HID attributes to the remote node. In this way the remote node knows the ZID Class device attributes information and vice-versa. The ZID Class device considers the remote node connected if the Configuration Complete command frame is sent successfully (refer to ZID Profile specification).

Prototype

```
uint8_t ZID_EnterConfigMode(uint8_t deviceId);
```

The following table specifies the parameters for ZIDClassDev_EnterConfigMode.

Table 2-16. ZIDClassDev_EnterConfigMode Parameters

Name	Type	Valid Range	Description
deviceId	uint8_t	0 – (gMaxPairTabelEntries_c -1)	The device identifier.

The possible return values for the ZIDClassDev_EnterConfigMode API call are shown in the following table.

Table 2-17. ZIDClassDev_EnterConfigMode API Call Return Values

Type	Possible Values	Description
uint8_t	gNWDenied_c gNWSuccess_c	Refer to the below section.

Functionality

When ZID_EnterConfigMode function is called, the ZID Profile checks if the node is paired with the specified device identifier. If it is not paired, returns the gNWDenied_c status.

If the function returns gNWSucces_c, the ZID Profile is ready to enter in configuration stage. In this stage, the node will get the remote node attributes and will push its HID attributes. The node will complete this

stage sending the Configuration Complete frame to the remote node. If the Configuration Complete frame is not send successfully, the process will fail.

The application is notified about the completion of the Configuration process through a ZID Enter Configuration Confirm message (zidConfigModeCnf_t).

2.1.1.10 ZIDClassDev_CompatibilityCheckResp

The ZIDClassDev_CompatibilityCheckResp accepts or refuses a connection (the ZID Adaptor device) during the configuration stage.

Prototype

```
uint8_t ZIDClassDev_CompatibilityCheckResp(bool_t continueConnect);
```

The following table specifies the parameters for ZIDClassDev_CompatibilityCheckResp.

Table 2-18. ZIDClassDev_CompatibilityCheckResp Parameters

Name	Type	Valid Range	Description
continueConnect	bool_t	TRUE or FALSE	Continue to connect or not.

The possible return values for the ZIDClassDev_CompatibilityCheckResp API call are shown in the following table.

Table 2-19. ZIDClassDev_EnterConfigMode API Call Return Values

Type	Possible Values	Description
uint8_t	gNWDenied_c gNWSuccess_c	Refer to the below section.

Functionality

During the configuration stage, the ZID Profile layer sends first the Get Attributes command and waits the response. When the Get Attributes Response command is received, the profile will generate and pass the zidClassDevCompatibilityCheckInd_t message to the application layer. This message contains exactly the received Get Attributes Response command. The application layer precesses the message and decides further if it wants to configure with the remote node, calling the ZIDClassDev_CompatibilityCheckResp function. The application must call this function within 100 milliseconds, starting from when the zidClassDevCompatibilityCheckInd_t message is received. Otherwise, the configuration stage will be ended.

When ZIDClassDev_CompatibilityCheckResp function is called, the ZID profile validates if it is in configuration stage and is waiting the application to check the remote node's compatibility. If the validation fails, the function returns gNWDenied_c status. Otherwise, it will check the `continueConnect parameter and` will continue or not the configuration stage.

2.1.1.11 ZIDClassDev_PushAttr

ZIDClassDev_PushAttr allows the Application layer to push some attributes to the remote node.

NOTE

Not all attributes can be push by the application. The list of attribute identifiers, provided by the application, shall contain the IDs that are not used in configuration stage (as mentioned by the ZID Profile specification, the attributes with an ”aplHID” prefix). Otherwise, the function will return the gNWInvalidParam_c status. For more details refer to ZID profile specification document.

Prototype

```
uint8_t ZIDClassDev_PushAttr(
                              uint8_t deviceId,
                              uint8_t attrIdListLen,
                              zidAttrId_t* pAttrIdList
                              );
```

The following table specifies the parameters for ZIDClassDev_PushAttr.

Table 2-20. ZIDClassDev_PushAttr Parameters

Name	Type	Valid Range	Description
deviceId	uint8_t	0 – (gMaxPairTabelEntries_c -1)	The device identifier.
attrIdListLen	uint8_t	-	The attributes’ identifiers list length.
pAttrIdList	zidAttrId_t*	-	Pointer to the list of attributes’ identifiers.

The possible return values for the ZIDClassDev_PushAttr API call are shown in the following table.

Table 2-21. ZIDClassDev_PushAttr API Call Return Values

Type	Possible Values	Description
uint8_t	gNWInvalidParam_c gNWDenied_c gNWNoMemory_c gNWSuccess_cc	Refer to this chapter note and the following section.

Functionality

ZIDClassDev_PushAttr is used to configure the ZID Profile for sending the Push Attributes command and handling the Generic response (refer to ZID Profile specification).

When ZIDClassDev_PushAttr function is called, the ZID Profile checks if all the conditions (see this chapter note) to begin sending Push Attributes command are met.

If the function returns gNWSucces_c, the ZID Profile is ready and configured to send the command over the air and to receive the response. The attributes being pushed are specified in the list of attributes' identifiers (pAttrIdList) and the command is directed to the specified device identifier.

The application is notified of the completion of the Push Attributes process through a ZID Class Device Push Attributes Confirm message (zidClassDevPushAttrCnf_t).

2.1.1.12 ZIDClassDev_Heartbeat

The ZIDClassDev_Heartbeat sends the heartbeat frame, signaling the remote node that the ZID Class device will have the receiver open for a while (check in with the HID adaptor in case it wishes to send it a command).

Prototype

```
uint8_t ZIDClassDev_Heartbeat(uint8_t deviceId);
```

The following table specifies the parameters for ZIDClassDev_Heartbeat.

Table 2-22. ZIDClassDev_Heartbeat Parameters

Name	Type	Valid Range	Description
deviceId	uint8_t	0 – (gMaxPairTabelEntries_c -1	The device identifier.

The possible return values for the ZIDClassDev_Heartbeat API call are shown in the following table.

Table 2-23. ZIDClassDev_EnterConfigMode API Call Return Values

Type	Possible Values	Description
uint8_t	gNWDenied_c gNWSuccess_c gNWNoMemory_c	Refer to the below section.

Functionality

ZIDClassDev_Heartbeat is used to configure the ZID Profile for sending the Heartbeat command and handling the response (refer to ZID Profile specification).

When ZIDClassDev_Heartbeat function is called, the ZID Profile checks if all the conditions (see this chapter note) to begin sending Heartbeat command are met.

If the function returns gNWSucces_c, the ZID Profile is ready and configured to send the command over the air and to receive the response. The command is directed to the specified device identifier.

The application is notified of the completion of the Heartbeat process through a ZID Class Device Heartbeat Confirm message (zidClassDevHeartbeatCnf_t).

2.1.1.13 ZIDClassDev_RemoveConfiguredDevice

The ZIDClassDev_RemoveConfiguredDevice function removes the connected remote node (the ZID Adaptor device).

Prototype

```
uint8_t ZIDAdp_RemoveConfiguredDevice(uint8_t deviceId);
```

The following table specifies the parameters for ZIDClassDev_RemoveConfiguredDevice.

Table 2-24. ZIDClassDev_RemoveConfiguredDevice Parameters

Name	Type	Valid Range	Description
deviceId	uint8_t	0 – (gMaxPairTabelEntries_c -1)	The device identifier.

The possible return values for the ZIDClassDev_RemoveConfiguredDevice API call are shown in the following table.

Table 2-25. ZIDClassDev_RemoveConfiguredDevice API Call Return Values

Type	Possible Values	Description
uint8_t	gNWInvalidParam_c gNWSuccess_c	Refer to the below section.

Functionality

When ZIDClassDev_RemoveConfiguredDevicefunction is called, it verifies if the specified device identifier is connected. If it is not connected, the function returns gNWInvalidParam_c status. Otherwise, it removes the connection and returns the gNWSuccess_c status. From now on, the node will drop the ZID frames received from the removed node.

Note that this function doesn't unpair the remote node (for this, use the unpair network function).

2.1.1.14 ZID_GetLocalAttr

The ZID_GetLocalAttr function gets a local attribute.This is a common function and has the same role either on ZID Adaptor device or ZID Class device.

Prototype

```
uint8_t ZID_GetLocalAttr(uint8_t attrId, /* IN */
                         uint8_t* pAttrValue,  /* OUT */
                         uint8_t* pAttrLength); /* OUT */
```

The following table specifies the parameters for ZID_GetLocalAttr.

Table 2-26. ZID_GetLocalAttr Parameters

Name	Type	Valid Range	Description
attrId	uint8_t	-	The attribute identifier.
pAttrValue	uint8_t*	-	The read attribute value.
pAttrLength	uint8_t	-	The read attribute length.

The possible return values for the ZID_GetLocalAttr API call are shown in the following table.

Table 2-27. ZID_GetLocalAttr API Call Return Values

Type	Possible Values	Description
uint8_t	gNWUnsupportedAttribute_c gNWSuccess_c	Refer to the below section.

Functionality

When ZID_GetLocalAttr function is called, it starts searching the specified attribute identifier. If the attribute is not found, it returns the gNWUnsupportedAttribute_c status. Otherwise the function returns the requested attribute value and length, and the gNWSuccess_c status.

2.1.1.15 ZID_SetLocalAttr

The ZID_SetLocalAttr function sets a local attribute. This is a common function and has the same role either on ZID Adaptor device or ZID Class device.

Prototype

```
uint8_t ZID_SetLocalAttr(uint8_t attrId,    /* IN */
                         uint8_t*pAttrValue);  /* IN */
```

The following table specifies the parameters for ZID_SetLocalAttr.

Table 2-28. ZID_SetLocalAttr Parameters

Name	Type	Valid Range	Description
attrId	uint8_t	-	The attribute identifier.
pAttrValue	uint8_t*	-	The attribute value.

The possible return values for the ZID_SetLocalAttr API call are shown in the following table.

Table 2-29. ZID_SetLocalAttr API Call Return Values

Type	Possible Values	Description
uint8_t	gNWUnsupportedAttribute_c gNWSuccess_c	Refer to the below section.

Functionality

When ZID_SetLocalAttr function is called, it starts searching the specified attribute identifier. If the attribute is not found, it returns the gNWUnsupportedAttribute_c status. Otherwise the function sets the attribute value and returns the gNWSuccess_c status.

2.1.1.16 ZID_GetZIDAttributes

ZID_GetZIDAttributes instructs the ZID Profile layer to get one or more attributes from a remote node. This is a common function and has the same role either on ZID Adaptor device or ZID Class device. Some additional macros can be used to make the diference between these two devices.

Prototype

```
uint8_t ZID_GetZIDAttributes(uint8_t deviceId,
                             uint8_t dataPendingFlag,
                             uint8_t attrIdListLen,
                             zidAttrId_t* pAttrIdList);

#define ZIDClassDev_GetZIDAttributes(deviceId, attrIdListLen, pAttrIdList) \
          ZID_GetZIDAttributes(deviceId, FALSE, attrIdListLen,pAttrIdList);
#define ZIDAdp_GetZIDAttributes(deviceId, dataPendingFlag, attrIdListLen, pAttrIdList) \
          ZID_GetZIDAttributes(deviceId, dataPendingFlag, attrIdListLen,pAttrIdList);
```

The following table specifies the parameters for ZID_GetZIDAttributes.

Table 2-30. ZID_GetZIDAttributes Parameters

Name	Type	Valid Range	Description
deviceId	uint8_t	0 – (gMaxPairTabelEntries_c -1)	The device identifier.
dataPendingFlag	uint8_t	TRUE or FALSE	DataPendingFlag parameter shall be set (TRUE) only when the Adaptor has more data to be sent. The ZID Class device shall always set this parameter to be FALSE.
attrIdListLen	uint8_t	-	The attributes' identifiers list length.
pAttrIdList	zidAttrId_t*	-	Pointer to the list of attributes' identifiers.

The possible return values for the `ZID_GetZIDAttributes` API call are shown in the following table.

Table 2-31. ZID_GetZIDAttributes API Call Return Values

Type	Possible Values	Description
uint8_t	gNWInvalidParam_c gNWDenied_c gNWNoMemory_c gNWSuccess_cc	Refer to this chapter note and the below section.

Functionality

ZID_GetZIDAttributes is used to configure the ZID Profile for sending the Get Attributes command and handling the response (refer to ZID Profile specification).

When ZID_GetZIDAttributes function is called, the ZID Profile checks if all the conditions (see this chapter note) to begin sending Get Attributes command are met.

If the function returns gNWSucces_c, the ZID Profile is ready and configured to send the over- the-air request and receive the response. The attributes being requested are specified in the attributes' identifiers list (pAttrIdList) and the request is directed to the specified device identifier.

The application is notified of the completion of the Get ZID Attributes process through a ZID Get Attributes Confirm message (zidGetAttrCnf_t).

NOTE

The list of attribute identifiers, provided by the application, shall contain the IDs that are not used in configuration stage (as mentioned by the ZID Profile specification, the attributes with an "aplHID" prefix). Otherwise, the function will return the gNWInvalidParam_c status. For more details refer to ZID profile specification document.

2.1.1.17 ZID_IsIdle

The ZID_IsIdle function checks if the ZID Profile is in idle state. This is a common function and has the same role either on ZID Adaptor device or ZID Class device. Some additional macros can be used to make the diference between these two devices.

Prototype

```
bool_t  ZID_IsIdle(void);
#define ZIDClassDev_IsIdle() ZID_IsIdle()
#define ZIDAdp_IsIdle() ZID_IsIdle()
```

ZID_IsIdle has no prarameters.

The possible return values for the ZID_IsIdle API call are shown in the following table.

Table 2-32. ZIDClassDev_EnterConfigMode API Call Return Values

Type	Possible Values	Description
uint8_t	TRUE FASLE	Refer to the below section.

Functionality

This ZID_IsIdle function checks if the ZID profile is in idle state. If it is idle, the function returns TRUE, otherwise it returns FALSE.

2.1.1.18 ZID_AbortProcess

The ZID_AbortProcess function aborts the ZID profile running process. It is a common function and has the same role either on ZID Adaptor device or ZID Class device. Some additional macros can be used to make the diference between these two devices.

Prototype

```
uint8_t ZID_AbortProcess(void);

#define ZIDClassDev_AbortProcess()              ZID_AbortProcess()
#define ZIDAdp_AbortProcess()                 ZID_AbortProcess()
```

ZID_AbortProcess has no parameters.

The possible return values for the ZID_AbortProcess API call are shown in the following table.

Table 2-33. ZIDClassDev_EnterConfigMode API Call Return Values

Type	Possible Values	Description
uint8_t	gNWDenied_c gNWSuccess_c	Refer to the below section.

Functionality

ZID_AbortProcess checks if a ZID profile process is running. If it is running, the function aborts the process and returns gNWSuccess_c; otherwise it returns the gNWDenied_c status.

2.1.1.19 ZID_ReportData

ZID_ReportData instructs the ZID Profile layer to send an unsolicited report frame command to a remote node. It is a common function and has the same role either on ZID Adaptor device or ZID Class device. Some additional macros can be used to make the difference between these two devices.

Prototype

```
uint8_t ZID_ReportData(
                        uint8_t                         deviceId,
                        uint8_t                         txOptions,
                        uint8_t                         reportRecordsListSize,
                        zidReportDataRecord_t*          pReportRecordsList
                       )
#define ZIDAdp_ReportData(deviceId, txOptions, reportRecordsListSize, pReportRecordsList) \
              ZID_ReportData(deviceId, txOptions, reportRecordsListSize, pReportRecordsList)
#define ZIDClassDev_ReportData(deviceId, txOptions,reportRecordsListSize, pReportRecordsList)\
              ZID_ReportData(deviceId, txOptions, reportRecordsListSize, pReportRecordsList)
```

The following table specifies the parameters for ZID_ReportData.

Table 2-34. ZID_ReportData Parameters

Name	Type	Valid Range	Description
deviceId	uint8_t	0 – (gMaxPairTabelEntries_c -1)	The device identifier.
txOptions	uint8_t		The transmission options (see next chapter).
reportRecordsListSize	uint8_t	-	The size of the list of report data records list.
pReportRecordsList	zidReportDataRecord_t*	-	Pointer to the report data records.

The possible return values for the ZID_ReportData API call are shown in the following table.

Table 2-35. ZID_ReportData API Call Return Values

Type	Possible Values	Description
uint8_t	gNWDenied_c gNWNoMemory_c gNWSuccess_cc	Refer to this chapter note and the below section.

Functionality

ZID_ReportData is used to configure the ZID Profile for sending a Report Data command frame (refer to ZID Profile specification)

When ZID_ReportData function is called, the ZID Profile checks if all the conditions (see this chapter note) to begin sending Report Data command are met.

If the function returns gNWSucces_c, the ZID Profile is ready and configured to send the command over the air. The report will be directed to the specified device identifier. The report data records being sent are specified by `pReportRecordsList` pointer. The transmission options parameter specifies if the Report Data command frame should be sent on Interrupt pipe or Control pipe, with or without data pending (refer to ZID Profile specification).

The application is notified about the completion of the Report Data process through a ZID Report Data Confirm message (zidReportDataCnf_t).

2.1.1.20 ZIDProfile_HandleNwkNIdeMsg

ZIDProfile_HandleNwkNIdeMsg handles all network NLDE messages that are intended for the ZID Profile.

Prototype

```
void ZIDProfile_HandleNwkNIdeMsg(hidProfileToAppMsg_t* hidProfileToAppMsg);
```

ZIDProfile_HandleNwkNIdeMsg has no return status.

The following table specifies the parameters for ZIDProfile_HandleNwkNIdeMsg.

Table 2-36. ZIDProfile_HandleNwkNIdeMsg Parameters

Name	Type	Valid Range	Description
hidProfileToAppMsg	hidProfileToAppMsg_t	-	Pointer to NLDE message

Functionality

ZIDProfile_HandleNwkNIdeMsg is used to handle the incoming NLDE messages that are intended for ZID Profile.

2.1.2 ZID Profile defines, messages structures and data types

The application project must include in the Compiler's command-line options the following defines which specifies the device type:

- gZIDProfileClassDevice_d=1 - the node acts as a ZID Class device.
- gZIDProfileAdaptor_d=1 - the node acts as a ZID Adaptor.

These macros have to match the corresponding included library and will enable the API functions of the selected device.

The ZID profile messages are collected as a union data structure in a single message; it also contains a message type member that enumerates the types of the available messages. These messages are transported from the ZID Profile layer to the Application layer for signaling the completion of a requested feature or the arriving of the data.

The profile messages and enumeration are found in the ZIDProfileInterface.h file.

The following structure/union and enumeration are used:

```
/* General structure of a message received by the application over ZID Profile SAP */
typedef struct zidProfileToAppMsg_tag
{
  zidProfileToAppMsgType_t   msgType;
  union {
    zidReportDataInd_t                zidReportDataInd;
    zidReportDataCnf_t                zidReportDataCnf;
```

```
    zidGetAttrCnf_t                    zidGetAttrCnf;
    zidConfigModeCnf_t                 zidConfigModeCnf;

    zidClassDevCompatibilityCheckInd_t zidClassDevCompatibilityCheckInd;
    zidClassDevSetReportInd_t          zidClassDevSetReportInd;
    zidClassDevPushAttrCnf_t           zidClassDevPushAttrCnf;
    zidClassDevHeartbeatCnf_t          zidClassDevHeartbeatCnf;
    zidClassSendReportIdsListCnf_t     zidClassSendReportIdsListCnf;

    zidAdaptorHeartbeatInd_t           zidAdaptorHeartbeatInd;
    zidAdaptorPushAttrInd_t            zidAdaptorPushAttrInd;
    zidAdaptorGetReportDataCnf_t       zidAdaptorGetReportDataCnf;
    zidAdaptorSetReportDataCnf_t       zidAdaptorSetReportDataCnf;
  } msgData;
}zidProfileToAppMsg_t;


/* Messages types used for informing the application about confirms or indications from
   ZID profile arrived trough the ZID SAP */
typedef enum
{
  /* Common message types */
  gZIDReportDataInd_c,
  gZIDReportDataCnf_c,
  gZIDGetAttrCnf_c,
  gZIDConfigModeCnf_c,

  /* HID class device message types */
  gZIDClassDevCompatibilityCheckInd_c,
  gZIDClassDevSetReportInd_c,
  gZIDClassDevPushAttrCnf_c,
  gZIDClassDevHeartbeatCnf_c,
  gZIDClassSendReportIdsListCnf_c,

  /* Adaptor message types */
  gZIDAdaptorHeartbeatInd_c,
  gZIDAdaptorPushAttrInd_c,
  gZIDAdaptorGetReportDataCnf_c,
  gZIDAdaptorSetReportDataCnf_c,
  gZIDProfileMax_c
}zidProfileToAppMsgType_t;
```

The profile interface includes definitions for:

- Transmit options

```
/*
   Transmission options for Report Data.
   To send a report from a Zid Class device using unicast,
   with ACK, on multiple channels, with security use the combination
   (gTxOpt_CtrlPipeUnicast_c | gTxOpt_SecurePipe_c )
*/
#define gTxOpt_CtrlPipeUnicast_c       maskTxOptions_UseAck_c
#define gTxOpt_CtrlPipeBroadcast_c     maskTxOptions_Broadcast_c
#define gTxOpt_IntPipe_c               maskTxOptions_UseOneChannelOnly_c
#define gTxOpt_SecurePipe_c            maskTxOptions_UseSecurity_c
/* Data Pending bit is set only when Repord Data is sent from Adaptor device */
#define gTxOpt_SetDataPendingBit_c     (1 << 7)
```

- Receive options

```
/* Reception options */
#define gRxOpt_Broadcast_c              maskRxOptions_Broadcast_c
#define gRxOpt_UseSecurity_c            maskRxOptions_UseSecurity_c
#define gRxOpt_SetDataPendingBit_c      (1 << 7)
```

- Generic response code

```
/* Type defining possible values of the Response code field
   inside the ZID Generic Response command */
typedef enum
{
  gZidResp_Success_c = 0x00,
  gZidResp_UnsupportedRequest_c,
  gZidResp_InvalidParameter_c,
  gZidResp_ConfigFailure_c,
  gZidResp_InvalidReportId_c = 0x40,
  gZidResp_MissingFragment_c,
}zidGenericRsp_t;
```

- Attribute identifier

```
/* Type defining a ZID attribute identifier */
typedef enum
{
  gZidAttrId_KeyExTransferCount_c           = gGdpAttrId_KeyExTransferCount_c,
  gZidAttrId_PowerStatus_c,

  gZidAttrId_ZidProfileVersion_c            = 0xA0,
  gZidAttrId_IntPipeUnsafeTxWindowTime_c,
  gZidAttrId_ZidReportRepeatInterval_c,

  gZidAttrId_ZidParserVersion_c             = 0xE0,
  gZidAttrId_ZidDeviceSubClass_c,
  gZidAttrId_ZidProtocolCode_c,
  gZidAttrId_ZidCountryCode_c,
  gZidAttrId_ZidDeviceReleaseNumber_c,
  gZidAttrId_ZidVendorId_c,
  gZidAttrId_ZidProductId_c,
  gZidAttrId_ZidNumEndpoints_c,
  gZidAttrId_ZidPollInterval_c,
  gZidAttrId_ZidNumStdDescComps_c,
  gZidAttrId_ZidStdDescCompsList_c,
  gZidAttrId_ZidNumNullReports_c,

  gZidAttrId_ZidNumNonStdDescComps_c                  = 0xF0,
  gZidAttrId_ZidNonStdDescCompSpec1_c
}zidAttrId_t;
```

- Status

```
/* Status returned when the Configuration fails */
#define gDeviceConfigFailure_d       0x90
#define gDeviceConfigNoCapacity_d    0x91
#define gDeviceMissingFrag_d         0x92

/* Status returned when Report Data is too big */
```

```
#define gReportDataTooBig_d          0x93
```

- Report type

```
/* Type defining a ZID report type */
typedef enum
{
  gZidReportType_Reserved_c = 0x00,
  gZidReportType_Input_c,
  gZidReportType_Output_c,
  gZidReportType_Feature_c,
}zidReportType_t;
```

- Descriptor type

```
/* Type defining ZID Descs type */
typedef enum
{
  gZidDescType_Report_c = 0x22,
  gZidDescType_Physical_c
}zidDescType_t;
```

NOTE

The gTxOpt_SetDataPendingBit_c define is used only when the request is sent from the ZID Adaptor device.

2.1.2.1 Message Data Types

2.1.2.1.1 zidReportDataInd_t message

This message is generated by the ZID Profile to signal the application layer that a report data was received.

The zidReportDataInd_t message has the following structure:

```
/* Define the indication message for Report Data */
typedef struct zidReportDataInd_tag
{
 uint8_t                deviceId;
 uint8_t                profileId;
 uint8_t                vendorId[2];
 uint8_t                LQI;
 uint8_t                rxFlags;
 uint8_t                reportRecordsListSize;
 zidReportDataRecord_t*  pReportRecordsList;  /* points to the list of the report data records */
}zidReportDataInd_t;

/* Define report data payload */
typedef struct  zidReportDataRecord_tag
{
 uint8_t  reportSize;
 uint8_t  reportType;
 uint8_t  reportId;
 uint8_t  reportData[1]; /* where report data starts */
}zidReportDataRecord_t;
```

The structure fields description:

- deviceId – The pairing reference in its pairing table from where the report is being received.
- profileId – The profile identifier of the received message.
- vendorId – The vendor identifier of the received message.
- LQI – The Link Quality of the received message.
- rxFlags – The reception flags (refer to Freescale BeeStack Consumer Reference Manual.)
- `reportRecordsListSize` – The size of the report data records.
- `pReportRecordsList` – Pointer to the list of report data records.

2.1.2.2 zidConfigModeCnf_t message

This message is generated by the ZID Profile to signal the application layer that the Configuration stage was completed.

The ZID Configuration Mode Complete message has the following structure:

```
/* Type definitions for Configuration Mode Confirm */
typedef struct zidConfigModeCnf_tag
{
 uint8_t          status;
 uint8_t          deviceId;
 uint8_t          connectionIndex; /* Index in connection table */
}zidConfigModeCnf_t;
```

The structure fields description:

- status - Indicates either the successful completion of Configuration process (gNWSuccess_c) or a failure (gDeviceConfigFailure_d, gDeviceConfigNoCapacity_d, gDeviceMissingFrag_d).
- deviceId – The pairing reference in its pairing table from where the report is being received.
- connectionIndex – The index of the entry of the connection table.

2.1.2.3 hidGenericCnf_t message

The hidGenericCnf_t message is used to define the following messages:

- zidReportDataCnf_t (ZID Report Data Confirm)
- zidClassSendReportIdsListCnf_t (ZID Class Device Send Report IDs List confirm)
- zidClassDevPushAttrCnf_t (ZID Class Device Push Attributes confirm)
- zidClassDevHeartbeatCnf_t (ZID Class Device HeartBeat Confirm)
- zidAdaptorSetReportDataCnf_t (ZID Adaptor Device Set Report Data Confirm)

These messages are generated by the ZID Profile to signal the completion of the following processes:

- Report Data process (zidReportDataCnf_t)
- ZID Class device - Send Report IDs List process (zidClassSendReportIdsListCnf_t)
- ZID Class device - Push attributes process (zidClassDevPushAttrCnf_t)
- ZID Class device - Heartbeat process (zidClassDevHeartbeatCnf_t)

- ZID Adaptor - Set Report process (zidAdaptorSetReportDataCnf_t)

Each of these processes, except the Report Data and Send Report IDs List, waits for a Generic response command from the remote node. If this response arrives, the ZID Profile passes its status to the application layer using the confirm message.

The hidGenericCnf_t message has the following structure:

```
/* Define a generic confirm message */
typedef struct hidGenericCnf_tag
{
 uint8_t          status;
 uint8_t          deviceId;
}hidGenericCnf_t;
```

The Generic confirm message is also describing the following confirm messages:

```
/* Type definitions for confirm messages */
typedef zidGenericCnf_t         zidReportDataCnf_t;
typedef zidGenericCnf_t         zidClassSendReportIdsListCnf_t;
typedef zidGenericCnf_t         zidClassDevPushAttrCnf_t;
typedef zidGenericCnf_t         zidClassDevHeartbeatCnf_t;
typedef zidGenericCnf_t         zidAdaptorSetReportDataCnf_t;
```

The hidGenericCnf_t structure fields description:

- status – This field has the following meaning for each confirm message:
  - `zidReportDataCnf_t`: Indicates either the successful completion of Report Data process or identifies the network error that has occurred.
  - `zidClassSendReportIdsListCnf_t`: Indicates either the successful completion of ZID Class device - Send Report IDs List process or identifies the network error that has occurred.
  - `zidClassDevPushAttrCnf_t`, `zidClassDevHeartbeatCnf_t` ,`zidAdaptorSetReportDataCnf_t`: Indicates either the response code inside the received ZID Generic Response frame or identifies the network error that has occurred (refer to ZID Profile specification).
- deviceId – The pairing reference in its pairing table for which the confirm message was generated.

2.1.2.4 zidGetAttrCnf_t and zidClassDevCompatibilityCheckInd_t messages

The zidGetAttrCnf_t message is generated by the ZID Profile to signal the completion of the ZID Get Attributes process.

The `zidClassDevCompatibilityCheckInd_t` message is received during the configuration stage to give the application the ability to accept or refuse a connection.

The following structures describe these messages and their payload:

```
/* Define Get Attributes Confirm message */
typedef struct zidGetAttrCnf_tag
{
 uint8_t                 status;
 uint8_t                 deviceId;
 uint8_t                 getAttrRspLength;
 zidAttrStatusRecord_t*  pGetAttrRsp; /* Where the Attribute Status records starts;
                                         The the next fields after zidAttrStatus member are
                                         included in the zidAttrStatusRecord_t record
```

```
                                        if the zidAttrStatus field indicates a successful
read(see Specification) */
}zidGetAttrCnf_t;

/* Type defining a ZID attribute status record */
typedef struct zidAttrStatusRecord_tag
{
  zidAttrId_t        zidAttrId;
  zidAttrStatus_t    zidAttrStatus;
  zidAttrLength_t    zidAttrLength;
  uint8_t            aZidAttrValue[1];   /* where data starts */
}zidAttrStatusRecord_t;

typedef zidGetAttrCnf_t         zidClassDevCompatibilityCheckInd_t;
```

The `zidGetAttrCnf_t` structure fields description:

- status – Indicates either the expected Get Attributes response command was received successfully or identifies the network error that has occurred.
- deviceId – The pairing reference for which the Get Attributes confirm message was generated.
- `getAttrRspLength` – Indicates the Get Attributes response payload length if the status is success.
- `pGetAttrRsp` – Is a pointer that forwards the Get Attributes response command to the application layer. Refer to ZID Profile specification for each response fields.

2.1.2.5 zidClassDevSetReportInd_t message

This message is generated by the ZID Profile to signal the application layer that a Set Report Data command was received and the specified report was set by the ZID profile layer.

```
/* Define the indication message for Set Report request */
typedef struct zidClassDevSetReportInd_tag
{
 uint8_t                 deviceId;
 uint8_t                 profileId;
 uint8_t                 vendorId[2];
 uint8_t                 LQI;
 uint8_t                 rxFlags;
 uint8_t                 setReportLength;
 zidSetReportPayload_t*  pSetReport;
}zidClassDevSetReportInd_t;

/* Define the Set Report payload */
typedef struct  zidSetReportPayload_tag
{
 uint8_t  reportType;
 uint8_t  reportId;
 uint8_t  reportData[1];
}zidSetReportPayload_t;
```

The structure fields description:

- deviceId – The pairing reference in its pairing table from where the report is being received.
- profileId – The profile identifier of the received message.
- vendorId – The vendor identifier of the received message.

- LQI – The Link Quality of the received message.
- rxFlags – The reception flags.
- setReportLength – The size of the set report payload.
- pSetReport – Pointer to the set report payload (`zidSetReportPayload_t`).

2.1.2.6 zidAdaptorHeartbeatInd_t message

This message is generated by the ZID Profile to signal that a Heartbeat command was received from the remote node. If the device has no data pending (deviceHasDataPending field from remote node connection entry is set to FASLE), the ZID profile will respond automatically sending a Generic Response frame.

```
/* Define the indication message for Heartbeat */
typedef struct zidAdaptorHeartbeatInd_tag
{
 uint8_t                deviceId;
 uint8_t                profileId;
 uint8_t                vendorId[2];
 uint8_t                LQI;
 uint8_t                rxFlags;
}zidAdaptorHeartbeatInd_t;

ReportPayload_t;
```

The structure fields description:

- deviceId – The pairing reference in its pairing table from where the report is being received.
- profileId – The profile identifier of the received message.
- vendorId – The vendor identifier of the received message.
- LQI – The Link Quality of the received message.
- rxFlags – The reception flags (refer to Freescale BeeStack Consumer Reference Manual.)

2.1.2.7 zidAdaptorPushAttrInd_t message

This message is generated by the ZID Profile to signal the application layer that a Push Attributes command was received from the remote node (the ZID Class device).

```
//* Define the indication message for Push attribute */
typedef struct zidAdaptorPushAttrInd_tag
{
 uint8_t                deviceId;
 uint8_t                profileId;
 uint8_t                vendorId[2];
 uint8_t                LQI;
 uint8_t                rxFlags;
 uint8_t                attrRecordsLength;
 zidAttrRecord_t*       pAttrRecords;  /* points to the list of the attribute records */
}zidAdaptorPushAttrInd_t;

/* Define report data payload */
typedef struct  zidReportDataRecord_tag
{
 uint8_t  reportSize;
```

```
 uint8_t  reportType;
 uint8_t  reportId;
 uint8_t  reportData[1]; /* where report data starts */
}zidReportDataRecord_t;
```

The structure fields description:

- deviceId – The pairing reference in its pairing table from where the report is being received.
- profileId – The profile identifier of the received message.
- vendorId – The vendor identifier of the received message.
- LQI – The Link Quality of the received message.
- rxFlags – The reception flags (refer to Freescale BeeStack Consumer Reference Manual.)
- `attrRecordsLength` – The length of attributes records.
- `pAttrRecords` – Pointer to `attributes records`.

2.1.2.8 zidAdaptorGetReportDataCnf_t message

This message is generated by the ZID Profile to signal the application layer that the requested (sending Get report command) Report Data command frame was received from the remote node (the ZID Class device).

```
/* Define the Adaptor Get Report Data Confirm message
  (return only one report data record if status is success) */
typedef struct zidAdaptorGetReportDataCnf_tag
{
 uint8_t                 status;
 uint8_t                 deviceId;
 uint8_t                 reportDataLength;
 zidReportDataRecord_t*  pReportData;
}zidAdaptorGetReportDataCnf_t;

/* Define report data payload */
typedef struct  zidReportDataRecord_tag
{
 uint8_t  reportSize;
 uint8_t  reportType;
 uint8_t  reportId;
 uint8_t  reportData[1]; /* where report data starts */
}zidReportDataRecord_t;
```

The structure fields description:

- status - Indicates either the expected Report Data command was received successfully or identifies the network error that has occurred.
- deviceId – The pairing reference in its pairing table from where the report is being received.
- `reportDataLength` – The length of report data.
- `pReportData` – Pointer to `report data`.

2.1.3 ZID Adaptor Configuration and Global Data Types

The application shall configure the ZID Adaptor device setting the macros from ZIDAdaptorConfig.h file.

This file contains the following macros (for their role, refer to their comments):

```
/* Specifies the maximum number of supported connections
   (how many ZID devices can be handled)
   A ZID adaptor device considers a ZID Class device connected if the appropriate
   attributes and descriptors were successful received from it. */
#define gAdpMaxNumOfConnections_c                    3

/* Specifies the maximum number of NON STANDARD Descriptors that can be
   stored for all supported connections. Each remote ZID Class device can have one or more
   non-standard descriptors that have to be stored by the ZID adaptor device.  */
#define gAdpMaxNumOfNonStdDescComps_c                12

/* Specifies the memory pool size of NON STANDARD descriptor components that can be
   stored for all supported connections. Each remote ZID Class device can have one or more
   non-standard descriptor components  that are stored in this static memory pool.
   The static memory pool has the following format:

   byte0+byte1:  number of bytes in use from this memory pool(  includes also this two bytes)
   byte2 - byteX:: - first non-standard descriptor component
   ..........................................................................
   byteX+1 - byteY :  - second non-standard descriptor component, where Y>X+1
   ..........................................................................
*/
#define gAdpNonStdDescCompsMemPoolSize_c             300

/* Specifies the memory pool size of NON STANDARD NULL reports that can be
   stored for all supported connections. Each remote  ZID Class device can have one or more
   non-standard NULL reports that are stored in this static memory pool.
   Each non-standard NULL report is kept in memory in the following format:
   ----------------------------------------
   |Report Type| Report ID| NULL Report Data|
   ----------------------------------------
   The static memory pool has the following format:
   byte0+byte1:  number of bytes in use from this memory pool includes also this two bytes)
   byte2 - byteX: - first non-standard NULL report bytes (includes the report type,
            report ID and report data)
   ..........................................................................
   byteX+1 - byteY: - second non-standard NULL report bytes, where Y>X+1 (includes the report
type, report ID and report data)
   ..........................................................................
*/
#define gAdpNonStdNullRptsMemPoolSize_c              70

/* Default value for  aplZIDProfileVersion attribute;
   Specify the version of the ZID profile specification.
   The format shall be 0xJJMN representing a version JJ.M.N,
   where JJ is the major version number,
   M is the minor version number and N is the sub-minor version number.
   (for ZID v1.0.0, this attribute should take the value {0x00, 0x01})*/
#define gZIDProfileVersion_c                        {0x00,0x01}


/* Default value for aplIntPipeUnsafeTxWindowTime attribute;
   The aplIntPipeUnsafeTxWindowTime attribute is 16 bits in length and
   specifies the maximum time, during which transmissions use the unacknowledged,
   single channel service, permitted before an acknowledged,
   multiple channel transmission is required.
```

```
   This value shall be specified in the range
   (aplcMinIntPipeUnsafeTxWindow (50ms) - 0xffff).
*/
#define gZIDIntPipeUnsafeTxWindowTime_c               50

/* Default value for HIDParserVersion attribute;
   Specify the version of the core HID specification to which the device was designed.
   The format of this value shall be {MN, JJ}  representing a version JJ.MN,
   where JJ is the major version number and M is the minor version number
   and N is the sub-minor version number
   (for HID v1.11, this attribute should take the value {0x11, 0x01})
*/
#define gZIDHIDParserVersion_c                        {0x11,0x01}

/* Default value for ZID CountryCode attribute;
   Specify the country for which the hardware is localized;
   (refer to RF4CE ZID Profile Specification for possible values) */
#define gZIDCountryCode_c                             gZidCountryCode_NotSupported_c

/* Default value for ZID DeviceReleaseNumber attribute;
   Specify the release  number of the device itself; a version JJ.M.N,  where JJ is
   the major version number, M is the minor version number and N is
   the sub-minor version number {0xMN, 0xJJ}  */
#define gZIDDeviceReleaseNumber_c                     {0x00,0x00}

/* Default value for ZID VendorId attribute;
   Specify the USB-IF assigned identifier corresponding to the vendor of the device  */
#define gZIDVendorId_c                                {0x05,0x00}

/* Default value for ZID ProductId attribute;
   Specify a manufacturer assigned product identifier for the device  */
#define gZIDProductId_c                               {0x00,0x00}
```

The ZID Adaptor uses different global data like: connection information table (gAdpConnectionsInfoTbl), tables to store the non-standard descriptor components of the remote nodes (gAdpNonStdDescCompsTbl, gAdpNonStdDescCompsMemPoolSize), tables for storing the non-standard NULL reports of the remote nodes (gAdpNonStdNullRptsMemPool), local attributes information (gZidAttrData, gZidAttrInfoTbl). These global tables and their sizes are declared in ZIDAdaptorGlobals.c file which uses the type definitions from ZIDAdaptorGlobals.h.

To configure the global data use the macros from ZIDAdaptorConfig.h file.

A list of tables are described as follows (refer to the comments for field's description):

```
/* Table of connections */
adpConnectionInfo_t  gAdpConnectionsInfoTbl[gAdpMaxNumOfConnections_c];
const uint8_t  gAdpMaxNumOfConnections = gAdpMaxNumOfConnections_c;

/* NON STANDARD Descriptors table */
adpNonStdDescCompEntry_t gAdpNonStdDescCompsTbl[gAdpMaxNumOfNonStdDescComps_c];
const uint8_t  gAdpMaxNumOfNonStdDescComps  = gAdpMaxNumOfNonStdDescComps_c;

/*  The memory pool of non-standard descriptors */
uint8_t gAdpNonStdDescCompsMemPool[gAdpNonStdDescCompsMemPoolSize_c];
const uint16_t gAdpNonStdDescCompsMemPoolSize  = gAdpNonStdDescCompsMemPoolSize_c;
```

```
/* The memory pool for non-standard Null reports*/
uint8_t gAdpNonStdNullRptsMemPool[gAdpNonStdNullRptsMemPoolSize_c];
const uint16_t gAdpNonStdNullRptsMemPoolSize  = gAdpNonStdNullRptsMemPoolSize_c;

/*This structure define a ZID devices connections */
typedef struct adpConnectionInfo_tag
{
 bool_t                     used;
 uint8_t                    deviceId; /* device Id from Pair table */
 adpProxyTblEntry_t         proxyAttrInfo;
 bool_t                 deviceHasDataPending; /* Specify if the device has data pending or not;
                                                 If device has no data pending (FALSE) the
ZID Profile will respond
                                              automatically sending a Generic Response frame */
}adpConnectionInfo_t;

/* Describe the proxy entry for a connected HID Class device */
typedef struct adpProxyTblEntry_tag
{
  uint8_t             zidParserVersion[2];
  zidDeviceSubclass_t zidDeviceSubclass;
  zidProtocolCode_t   zidProtocolCode;
  zidCountryCode_t    zidCountryCode;
  uint8_t             zidDeviceReleaseNumber[2];
  uint8_t             zidVendorId[2];
  uint8_t             zidProductId[2];
  uint8_t             zidNumEndpoints;
  uint8_t             zidPollInterval;
  uint8_t             zidNumStdDescComps;
  uint8_t             zidStdDescCompsList[gZidStdDescCompsListSize];
  uint8_t             zidNumNullReports;
  uint8_t             zidNumOfNonStdDescComps;

  uint8_t             aplDeviceIdleRate;
  uint8_t             aplCurrentProtocol;
} adpProxyTblEntry_t;

/* Entry for NON STANDARD descriptors */
typedef struct adpNonStdDescCompEntry_tag
{
 bool_t              used;     /* This entry is used(TRUE) or not used(FALSE)*/
 uint8_t             deviceId; /* Specifies connected device Id*/
 uint8_t             nonStdDescCompAttrId; /* Specifies the assigned NON STANDARD descriptor
attribute ID */
 zidDescType_t       descType;
 uint8_t             descSize;
 uint8_t             descId;   /* Report Id or Physical set */
 uint16_t          descPoolOffset;   /* offset in the gAdpNonStdDescCompsMemPool memory pool  */
 uint8_t             nullReportSize;   /* if zero the Null report is ignored */
 uint16_t            nullReportPoolOffset;   /* offset in NULL report memory pool
(gAdpNonStdNullRptsMemPool);
                                                if zero the Null report is ignored */
}adpNonStdDescCompEntry_t;
```

NOTE

The gAdpNonStdDescCompsTbl table contains the non-standard descriptor components of all connected ZID Class devices. The gAdpConnectionsInfoTbl table contains information about the connected ZID Class devices. The relation between these tables is assured by the deviceId field (from `adpNonStdDescCompEntry_t` and `adpConnectionInfo_t` structures).

More tables are presented in the ZIDAdaptorGlobals.c global file which describe:

- The local attributes information (gZidAttrData, gZidAttrInfoTbl)
- The Device Descriptor information (gDeviceDescTblFormat, gDeviceDescAttrInfo) used to build the HID Class Device Descriptor needed to be forwarded to the host (eg. PC) when requested.
- The Configuration Descriptor information (gConfigDescTblFormat, gConfigDescAttrInfo) used to build the HID Class Configuration Descriptor needed to be forwarded to the host (eg. PC) when requested.
- The standard report descriptors (gMouseDesc, gKeyboardDesc, gTouchContactDataDesc, gTouchTapGestureDesc, gTouchScrollGestureDesc, gTouchPinchGestureDesc, gTouchRotationGestureDesc, gTouchSyncDesc, gTouchSyncDesc, gTouchPropertiesDesc, gTouchSupportPropDesc)
- The standard report descriptor components information (gAdpStdDescCompInfoTbl)

2.1.4 ZID Class device - Configuration and Global Data Types

The application shall configure the ZID Class device setting the macros from ZIDClassDeviceConfig.h file:

This file contains the following macros (for their role, refer to the associated comments):

```
/* Specifies the number of supported connections
   (how many connections can be handled)
   A HID Class device considers an Adaptor device connected if the appropriate
   attributes and descriptors were successfully  pushed.  */
#define gNumOfConnections_c         0x01

/* Default value for  aplZIDProfileVersion attribute;
   Specify the version of the ZID profile specification.
   The format shall be {MN, JJ} representing a version JJ.M.N,
   where JJ is the major version number,
   M is the minor version number and N is the sub-minor version number.
   (for ZID v1.0.0, this attribute should take the value {0x00, 0x01})*/
#define gZIDProfileVersion_c        {0x00,0x01}

/* Default value for aplIntPipeUnsafeTxWindowTime attribute;
   The aplIntPipeUnsafeTxWindowTime attribute is 16 bits in length and
   specifies the maximum time, during which transmissions use the unacknowledged,
   single channel service, permitted before an acknowledged,
   multiple channel transmission is required.
   This value shall be specified in the range
   (aplcMinIntPipeUnsafeTxWindow (50ms) - 0xffff).
*/
```

```
#define gZIDIntPipeUnsafeTxWindowTime_c    50

/* Default value for aplReportRepeatInterval attribute;
   The aplReportRepeatInterval attribute is 8 bits in length and
   specifies the interval in milliseconds at which a repeated report
   data command frame is transmitted to the HID adaptor.
   This value shall be specified in the range
   0x00 - aplcMaxReportRepeatInterval(100) milliseconds .
*/
#define gZIDReportRepeatInterval_c    50


/* Default value for HIDParserVersion attribute;
   Specify the version of the core HID specification to which the device was designed.
   The format of this value shall be {MN, JJ}  representing a version JJ.MN,
   where JJ is the major version number and M is the minor version number
   and N is the sub-minor version number
   (for HID v1.11, this attribute should take the value {0x11, 0x01})
*/
#define gZIDHIDParserVersion_c         {0x11,0x01}

/* Default value for ZIDDeviceSubClass  attribute;
   Specify the ZID subclass code of the ZID device
   (refer to RF4CE ZID Profile Specification for possible values)
*/
#define gZIDDeviceSubClass_c           gZidDeviceSubclass_NoSubclass_c

/* Default value for HID ProtocolCode  attribute;
   Specify the HID protocol code of the ZID device supporting the boot protocol
   (refer to RF4CE ZID Profile Specification for possible values) */
#define gZIDHIDProtocolCode_c          gZidProtocolCode_None_c

/* Default value for HID CountryCode attribute;
   Specify the country for which the hardware is localized;
   (refer to RF4CE ZID Profile Specification for possible values) */
#define gZIDCountryCode_c              gZidCountryCode_NotSupported_c

/* Default value for ZID DeviceReleaseNumber attribute;
   Specify the release  number of the device itself; a version JJ.M.N,  where JJ is
   the major version number, M is the minor version number
   and N is the sub-minor version number {0xMN, 0xJJ}  */
#define gZIDDeviceReleaseNumber_c      {0x00,0x00}

/* Default value for ZID VendorId attribute;
   Specify the USB-IF assigned identifier corresponding to the vendor of the device  */
#define gZIDVendorId_c                 {0x05,0x00}

/* Default value for ZID ProductId attribute;
   Specify a manufacturer assigned product identifier for the device  */
#define gZIDProductId_c                {0x00,0x00}

/* Default value for ZID NumEndpoints attribute;
   Specify the number of endpoints on the device in addition to
   the mandatory control endpoint. This value shall be set to 0x01 if only
   the mandatory input endpoint is used by the device or 0x02 if both
   the mandatory input endpoint and the optional output endpoint is used by the device  */
#define gZIDNumEndpoints_c              0x01
```

```
/* Default value for ZID PollInterval attribute;
   Specify the polling interval for endpoint data transfers, in 125us units.
   This value shall be set in the range 1 to 16 and is used as an exponent
   to calculate the value:
   Poll Interval = 2^(gZIDPollInterval_c-1)  */
#define gZIDPollInterval_c               0x0A

/* Specify the maximum number of standard descriptors components that can be supported by the
device*/
#define gZIDClassDevMaxNumOfStdDescComps_c  12

/* Default value for ZID NumStdDescComps attribute;
   Specify the number of RF4CE ZID profile defined standard descriptors
   supported by the device */
#define gZIDNumStdDescComps_c            0x01

/* Default value for ZID StdDescCompsList attribute;
   Specify the list of standard RF4CE ZID profile defined descriptors
   supported by the device.It has the length equal to the value of the
   aplZIDNumStdDescComps attribute.
   For example, a device that supports the mouse operation with a pinch gesture would set
   aplZIDNumStdDescComps to 2 and aplZIDStdDescCompsList to
   {0x01, 0x06, 0x00, 0x00,0x00, 0x00, 0x00,0x00,0x00, 0x00,0x00, 0x00}
   */
#define gZIDStdDescCompsList_c
{0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}


/* Default value for ZID NumDescriptors attribute;
   Specify the number of NON STANDARD ZID descriptors stored on device
   Range 0x00 - 0x0f */
#define gZIDNumNonStdDescComps_c      0x00

/* Default value for ZID number of NULL reports attribute;
   Range 0x00 - gZIDNumNonStdDescComps_c */
#define gZIDNumNullReports_c          0x00
```

Most of the configuration macros are used for setting the default value of local ZID attributes (gZidAttrData, gZidAttrInfoTbl) and the sizes of global tables. These global tables and their sizes are declared in ZIDClassDeviceGlobals.c file which uses the type definitions from ZIDClassDeviceGlobals.h. These files are used to declare the supported reports, the standard and non-standard descriptor components, the local attributes, the report data and ZID Class device's connections.

The ZID Class device handles the folowing tables:

- The connections table - specifies the node's connections.

```
classDevConnectionInfo_t gClassDevConnectionTbl[gNumOfConnections_c];

/* Keep track of connected HOST */
typedef struct classDevConnectionInfo_tag
{
  uint8_t used; /* TRUE or FALSE */
  uint8_t deviceId;
}classDevConnectionInfo_t;
```

- The attributes information table (gZidAttrData, gZidAttrInfoTbl) - includes the supported local attributes.

```
/* ZID attributes table structure */
typedef struct zidAttrData_tag
{
  uint8_t             aZIDProfileVersion[2];
  uint16_t            intPipeUnsafeTxWindowTime;
  uint8_t             reportRepeatInterval;
  uint8_t             zidParserVersion[2];
  zidDeviceSubclass_t zidDeviceSubclass;
  zidProtocolCode_t   zidProtocolCode;
  zidCountryCode_t    zidCountryCode;
  uint8_t             zidDeviceReleaseNumber[2];
  uint8_t             zidVendorId[2];
  uint8_t             zidProductId[2];
  uint8_t             zidNumEndpoints;
  uint8_t             zidPollInterval;
  uint8_t             zidNumStdDescComps;
  uint8_t             zidStdDescCompsList[12];
  uint8_t             zidNumNullReports;
  uint8_t             zidNumOfNonStdDescComps;
} zidAttrData_t;

/* Attribute Information table for handling the ZID attributes */
typedef struct attrInfoTbl_tag
{
 uint8_t  attrId;
 uint8_t  attrSize;
 uint8_t  *pAttrData;
}attrInfoTbl_t;
```

- The ZID Class device's reports table - all the supported reports should be included in this table.

```
const classDevReport_t gaClassDevReportsList[] =
{

 { gZidReportType_Input_c, gKeyboardReportId_c,
   gKeyboardRptDataSize, (uint8_t*)&gKeyboardRptData[0] },
 /* Add the next report HERE
 { Report Type, ReportID, report size, pointer Report Data...}
 */
};

/* Describe the report */
typedef struct classDevReport_tag
{
  zidReportType_t  reportType;  /* Describe the report type */
  uint8_t          reportId;    /* Specifies the report Id */
  uint8_t          dataSize;   /* Size of report DATA */
  uint8_t*         pData;      /* point to data described by report  */
}classDevReport_t;
```

- Non-standard NULL reports table - includes all non-standard NULL reports supported by the device.

```
/* List of non-standard NULL report list. Should contain gZIDNumNullReports_c entries. */
const nonStdNullReportsEntry_t gaNonStdNullReportsList[] =
```

```
{
 /* Add the non-standard NULL report  HERE
 {  NULL ReportID, NULL Report size, pointer to NULL Report ...}
 */
#if (gZIDNumNullReports_c == 0)
{ 0x00, 0x00, NULL}
#endif
};

/* This table will keep track of non-standard NULL reports List */
typedef struct nonStdNullReportsEntry_tag
{
  uint8_t  nullReportId;
  uint8_t  nullReportSize;
  uint8_t* pNullReportData;
}nonStdNullReportsEntry_t /* point to data described by report  */
```

NOTE

To add more reports and descriptor components follow the example code from `ZIDClassDeviceGlobals.c` file (see the ZID Application Profile User's Guide).