ZigBee®

Control your world™

September 2010

Understanding ZigBee Gateway

How ZigBee extends an IP network

Foreword

Since its inception, the ZigBee Alliance has worked with a singular focus: create a much needed global wireless language capable of giving "voices" to the myriad of everyday devices that surround us as we go about our daily lives. This focus has been aimed at the little devices often overlooked in an IT centric world, such as light switches, thermostats, electricity meters and remote controls, as well as more complex sensor devices found abundantly in the health care, commercial building and industrial automation sectors. As a result, ZigBee Alliance members have created a set of wireless standards offering extraordinary control, expandability, security, ease-of-use, affordability, and the ability to use ZigBee technology in any country around the world.

Today, organizations use ZigBee to effectively deliver solutions for a variety of areas including consumer electronic device control, energy management and efficiency, health care, telecom services, home and commercial building automation as well as industrial plant management. With this comprehensive set of attributes, the non-profit, open membership and volunteer-driven Alliance has become a thriving ecosystem of approximately 400 member organizations. As an ecosystem, the Alliance offers everything prospective product and service companies need to develop ZigBee products and services and benefit from the Alliance's competitive and stable supply chain.

Intended Audience

The primary purpose of this document is to provide a basic understanding of the capabilities, mechanisms, and recommended practices of the ZigBee Gateway to the reader.

This document is intended to assist several communities of interest in the strategic planning and implementation of gateways between ZigBee networks and IP networks such as the internet. Potential users of this document include:

- Network component/device buyers
- Test houses
- Developers
- System designers and integrators

This profile assumes that the users have some level of familiarity with:

- ZigBee basic concepts: Security (095296r00), Stack (075194r00), Profiles (08046r00)
- IETF basic concepts: network layer, transport layer, http, REST, SOAP
- IEEE Std 802.15.4 MAC basic concepts such as commands

Table of Contents

List of Figures

High-level overview

The ZigBee Gateway supports the following features:

- Address core IP, either IPv4 or IPv6 connectivity
  - IP security domain
  - Configuration
  - IP RPC protocol definitions
  - Network Address and port Translation (NAT)/Firewall traversal
  - Incorporate IP best practices using Internet Engineering Task Force (IETF), W3C and other existing IP-based standards (SOAP, REST)
  - IP terminates at the Gateway
- Provide broad ZigBee/IP application support that can span all profile needs (neutral and generic)
  - Public profiles can use ZigBee Gateway to connect the ZigBee networks to IP networks
  - Private profiles can use standard gateway devices to connect private ZigBee network to remote applications
- Scalable, extensible
  - Layered standard enables both very low cost and very powerful Gateways
  - Framework that can be included within profiles as a basic device type or hybrid devices
  - Profile groups can incorporate and extend from the framework capitalizing on a rich set of base functionality and infrastructure definition
  - Gateway Framework extensions

Usage scenarios

ZigBee Gateways can be used for a multitude of scenarios where the communication between ZigBee networks and IP network infrastructure is required. The use of standard interfaces for the ZigBee and IP network connection creates the opportunity to have universal adapters and operate the connection with IP back-end systems such as Service Platforms or Machine-to-Machine platforms in a standard fashion. Telecom operators and service providers envision such back-end operations that are enabled by ZigBee Gateway.

ZigBee Gateways scenarios are reported below:

- **Energy monitoring in buildings:** For the purpose of energy monitoring in buildings a collection station is typically used in the backend system to integrate ZigBee networks with the IP network infrastructure. The collection station is connected to different ZigBee Gateways located in the buildings that are used to gather power consumption data related to different locations in the building. In this scenario, a ZigBee Gateway based on Gateway Remote Interface Protocol (GRIP) bindings is typically used to meet low cost and easy-to-use requirements. ZigBee Gateways can be used in conjunction with ZigBee Smart Energy networks if the gateway device supports security requirements.
- **Home automation:** The interfaces defined within the ZigBee Gateway specification can be used to connect ZigBee-enabled homes with the service provider's web portals or service platforms in order to enable the remote access to the home. ZigBee Gateway features can also be integrated with home gateway architectures (e.g. ADSL modems or broadband gateways). In this case, the ZigBee Gateway Device might enable advanced web service interfaces such as Simple Object Access Protocol (SOAP). ZigBee Home Automation networks easily work with ZigBee Gateways.

- **Telecom and Retail Services:** ZigBee-based indoor location systems require the use of ZigBee Gateways to establish communication with web-based tracking systems or location platforms. Light, fast and scalable architectures are required within ZigBee Telecom Services where mobile nodes might need to be tracked within a certain area. A ZigBee Gateway supporting an efficient REpresentational State Transfer (REST) interface can be used for this purpose. Since ZigBee Retail Services leverages many of the features of ZigBee Telecom Services, ZigBee Gateways play a crucial role in providing communication with IP systems.
- **Health care monitoring:** A home gateway can be used in these applications to establish a connection to ZigBee Health Care monitoring systems or IP Host applications (IPHA). In this case the interfaces defined within the ZigBee Gateway specification can be used to guarantee a standard communication paradigm towards IP applications such as health monitoring platforms. ZigBee Gateways can be used in conjunction with ZigBee Health Care networks if the gateway device supports optional security requirements

Architecture

Functional

Interworking between ZigBee networks and IP is a key for many applications, and the ZigBee Gateway specification must meet many different needs. Therefore the ZigBee Gateway does not define a single protocol; rather it defines a two-layered API:

- A set of abstract (protocol independent) functions
  - The Gateway specification defines a Remote Procedure Call-based (RPC) API to ZigBee functionality and the management of the IP gateway itself
    - Support for complete Application Support Layer (APS), ZigBee Device Object (ZDO), and security services (SEC) commissioning both into and out of ZigBee networks
    - It allows interactions between IP applications and the profile applications on ZigBee devices
    - By exposing the application interface in a standard way remote IP applications can be interoperable with Gateways from multiple vendors
    - The set of mandatory functions is reduced at a minimum, allowing differentiation of products based on the different levels of implementation
- An extensible set of RPC protocols (i.e. bindings) specifying how to expose the API using a specific protocol
  - The different bindings provide scalability for cost versus feature tradeoffs
  - We do not have to re-invent interfaces for every system since the platform specification is not bound to a specific profile but gives the tools to IPHAs to support the profiles needed
  - Release 1 of the Gateway specification features SOAP, REST and GRIP bindings;
    - SOAP provides higher level web services oriented access to the Gateway API
    - REST provides a lightweight web-based API
    - GRIP is the protocol of choice for simplest ZigBee Gateway Devices, given its tiny footprint

To address diverse applications in an efficient manner, a compliant gateway does not need to

implement all the RPC bindings; it is enough to implement only one of them, or possibly more than one, if desired. Application-specific gateways that build on the general ZigBee Gateway specification can indicate which binding must be implemented, depending on the specific use case and/or the typical scenario. This way, a profile could mandate one of the bindings for specific use cases or allow the vendor to decide which binding to implement (or define a fourth binding not described in the specification).

The two-tiered API is matched by a two-layered functional architecture:

- A northbound "interface" implementing at least one of the three bindings
- A protocol-agnostic layer that implements each sub-segment of the overall API:
  - APS, ZDP, ZigBee Cluster Library (ZCL) and ZigBee Network Layer (NWK) expose the different layers of the ZigBee stack
  - Gateway Management Object (GMO) provides access to low-level ZigBee stack functions as well as high-level "macro" functions. These coarse-grained functions reduce complexity on IPHA and optimized IP network traffic
  - ZigBee Gateway Device (ZGD) specification defines its own information base (GIB) and cluster to advertise the "Gateway service" to ZigBee nodes

Bindings

ZigBee Gateway Devices can expose the standard API through one (or more) of the following RPC protocols ("bindings"):

- SOAP is a standard to perform remote procedure calls through Hypertext Transport Protocol (HTTP)/Extensible Markup Language (XML) requests
  - Syntax of requests is specified by an XML document (Web Services Description Language [WDSL]) provided by Annex D of ZigBee Gateway specification
  - Most popular development environments provide tools that generate stubs by "compiling" WSDL documents, actually turning remote into local calls
  - Applications can concentrate on their business logic without having to deal with the complexities of network communications and data formatting, and achieve interoperability with no effort
- REST, similar to SOAP, encodes remote invocation using HTTP/XML schema, but instead of just tunneling them through HTTP POST, it uses all the HTTP methods to access the API as a "resource repository"
  - XML documents are much shorter and simpler, and in many cases the body does not even exist
  - The footprint of both the Application and the ZigBee Gateway Device stack is very light

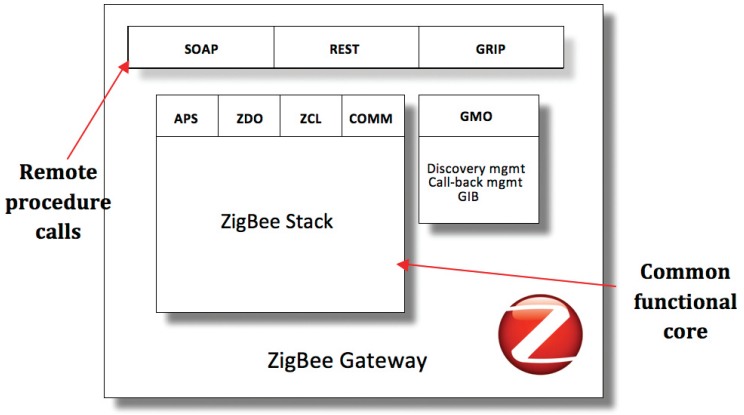

Figure 1 - ZigBee Gateway Architecture

- Many operations can be performed using a Web browser
- In the next section an example regarding the usage of the REST interface is presented.

- GRIP is a binary protocol that exchanges raw ZigBee stack structures on Transmission Control Protocol (TCP) connections:
  - Being a binary protocol, it features minimal bandwidth usage
  - Basic API procedures (e.g. send and receive ZCL/APS/NWK packets) can be implemented just by placing a TCP envelope, so the Gateway implementation could be a tiny layer on top of the ZigBee stack

SOAP Example

Two specific examples are shown below detailing the "On the wire" interaction between a host application and a gateway, employing SOAP over HTTP 1.1

Send a GMO Getversion command to the Gateway

Initial GetVersion request to Gateway Showing POST request headers and specifically the Getversion action	POST /GMO HTTP/1.1 User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol 2.0.50727.3603) Content-Type: text/xml; charset=utf-8 SOAPAction: http://www.zigbee.org/zgd/GetVersion Host: 192.168.118.50:8080 Content-Length: 295 Expect: 100-continue Connection: Keep-Alive
HTTP Response from Gateway	HTTP/1.1 100 Continue
xml encoding of the GetVersion request	<?xml version="1.0" encoding="utf-8"?> <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"> <soap:Body> <GetVersion xmlns="http://www.zigbee.org/zgd/" /> </soap:Body> </soap:Envelope>
HTTP Response from Gateway	HTTP/1.1 200 OK
SOAP Response from Gateway	Server: gSOAP/2.7 Content-Type: text/xml; charset=utf-8 Content-Length: 1301 Connection: close <?xml version="1.0" encoding="UTF-8"?> <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ns2="http://www.zigbee.org/GWGSchema" xmlns:ns10="http://www.zigbee.org/zgd/NWK" xmlns:ns11="http://www.zigbee.org/zgd/GMO" xmlns:ns12="http://www.zigbee.org/zgd/GMOEvent" xmlns:ns13="http://www.zigbee.org/zgd/INTERPAN" xmlns:ns14="http://www.zigbee.org/zgd/INTERPANEvent" xmlns:ns3="http://www.zigbee.org/zgd/ZDP" xmlns:ns4="http://www.zigbee.org/zgd/ZDPEvent" xmlns:ns5="http://www.zigbee.org/zgd/ZCLEvent" xmlns:ns6="http://www.zigbee.org/zgd/APSEvent" xmlns:ns7="http://www.zigbee.org/zgd/NWKEvent" xmlns:ns8="http://www.zigbee.org/zgd/ZCL" xmlns:ns9="http://www.zigbee.org/zgd/APS" xmlns:ns1="http://www.zigbee.org/zgd/"> <SOAP-ENV:Body><ns1:GetVersionResponse><Status>0</Status><Version> <ns2:VersionIdentifier>1</ns2:VersionIdentifier> <ns2:FeatureSetIdentifier>0</ns2:FeatureSetIdentifier> <ns2:RPCProtocol>SOAP</ns2:RPCProtocol> <ns2:RPCProtocol>REST</ns2:RPCProtocol> <ns2:ManufacturerVersion>Exegin Gateway Version 0.0 </ns2:ManufacturerVersion></Version></ns1:GetVersionResponse> </SOAP-ENV:Body></SOAP-ENV:Envelope>

Send a GMO Get command to the Gateway to return the nwkNeighbour table

Initial Get request to Gateway Showing POST request headers and Specifically the Get action	POST /GMO HTTP/1.1 User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol 2.0.50727.1433) Content-Type: text/xml; charset=utf-8 SOAPAction: “http://www.zigbee.org/zgd/Get” Host: 172.16.2.26:8080 Content-Length: 322 Expect: 100-continue
HTTP Response from Gateway	HTTP/1.1 100 Continue
xml encoding of the Get request with the ttribute Id 0x87 *(nwkNeighborTable)*	<?xml version=”1.0” encoding=”utf-8”?> <soap:Envelope xmlns:soap=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance” xmlns:xsd=”http://www.w3.org/2001/XMLSchema”> <soap:Body> <Get xmlns=”http://www.zigbee.org/zgd/”> <attrId xmlns=””>0x87</attrId> </Get> </soap:Body> </soap:Envelope>
HTTP Response from Gateway	HTTP/1.1 200 OK
SOAP Response from Gateway	Server: gSOAP/2.7 Content-Type: text/xml; charset=utf-8 Content-Length: 1060 Connection: close <?xml version=”1.0” encoding=”UTF-8”?> <SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/” xmlns:SOAP-ENC=”http://schemas.xmlsoap.org/soap/encoding/” xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance” xmlns:xsd=”http://www.w3.org/2001/XMLSchema” xmlns:ns2=”http://www.zigbee.org/GWGSchema” xmlns:ns10=”http://www.zigbee.org/zgd/NWK” xmlns:ns11=”http://www.zigbee.org/zgd/GMO” xmlns:ns12=”http://www.zigbee.org/zgd/GMOEvent” xmlns:ns13=”http://www.zigbee.org/zgd/INTERPAN” xmlns:ns14=”http://www.zigbee.org/zgd/INTERPANEvent” xmlns:ns3=”http://www.zigbee.org/zgd/ZDP” xmlns:ns4=”http://www.zigbee.org/zgd/ZDPEvent” xmlns:ns5=”http://www.zigbee.org/zgd/ZCLEvent” xmlns:ns6=”http://www.zigbee.org/zgd/APSEvent” xmlns:ns7=”http://www.zigbee.org/zgd/NWKEvent” xmlns:ns8=”http://www.zigbee.org/zgd/ZCL” xmlns:ns9=”http://www.zigbee.org/zgd/APS” xmlns:ns1=”http://www.zigbee.org/zgd/”> <SOAP-ENV:Body> <ns1:GetResponse> <Status>0</Status> <value>0101ED000301033804001CDAFFFF002035</value> </ns1:GetResponse> </SOAP-ENV:Body> </SOAP-ENV:Envelope>

The results data "0101ED000301033804001CDAFFFF002035" translates to:

```
EXTADDR : 0x001cdafff002035
NWKADDR : 0x3804
TYPE    : 3
RXIDLE  : 1
RELATION: 3
TXFAIL  : 0
LQI     : 237
COST    : 1
AGE     : 1
```

REST Example

In this section, a REST interface example demonstrates how an application can register with a ZigBee Gateway to receive a local end point and then register for some services (i.e. how to receive ZigBee messages by dynamically allocating an end point via the Gateway API). It requires three main steps.

1. Create a new local service descriptor

As a first step, an IPHA needs to register the end point by using the procedure "ConfigureEndpoint" as reported on paragraph 11.4.5 of the specification document. Since the "Endpoint" element (an input of the SimpleDescriptor XML document) may not be present, it is up to the ZigBee Gateway Device (ZGD) to choose an unused end point and associate the SimpleDescriptor to that end point. The shall return the Endpoint identifier associated with this SimpleDescriptor (either the one specified by the IPHA in its SimpleDescriptor, or the Endpoint chosen by the ZGD if the IPHA has not specified one) to the IPHA. The command to be sent will be then:

URI	`http://"ZGD_IP_Addr:ZGD_PORT"/localnode/services`
HTTP method	`POST`

Request XML message	`<?xml version="1.0" encoding="UTF-8"?>`
	`<tns:SimpleDescriptor xmlns:tns="http://www.zigbee.org/GWGRESTSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:gal="http://www.zigbee.org/GWGSchema" xsi:schemaLocation="http://www.zigbee.org/GWGRESTSchema/rest/rest.xsd http://www.zigbee.org/GWGSchema/rest/gal.xsd ">`
	`<gal:ApplicationProfileIdentifier>0x0104</gal:ApplicationProfileIdentifier>`
	`<gal:ApplicationDeviceIdentifier>0x0002</gal:ApplicationDeviceIdentifier>`
	`<gal:ApplicationDeviceVersion>0x00</gal:ApplicationDeviceVersion>`
	`<gal:ApplicationInputCluster>0x0000</gal:ApplicationInputCluster>`
	`<gal:ApplicationInputCluster>0x0003</gal:ApplicationInputCluster>`
	`<gal:ApplicationInputCluster>0x0004</gal:ApplicationInputCluster>`
	`<gal:ApplicationInputCluster>0x0005</gal:ApplicationInputCluster>`
	`<gal:ApplicationInputCluster>0x0006</gal:ApplicationInputCluster>`
	`<gal:ApplicationOutputCluster>0x0003</gal:ApplicationOutputCluster>`
	`</tns:SimpleDescriptor>`

Response XML message	<?xml version="1.0" encoding="UTF-8"?> <tns:SimpleDescriptor xmlns:tns="http://www.zigbee.org/GWGRESTSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:gal="http://www.zigbee.org/GWGSchema" xsi:schemaLocation="http://www.zigbee.org/GWGRESTSchema/rest/rest.xsd http://www.zigbee.org/GWGSchema /home/alberto/workspace/rest/gal.xsd "> <gal:EndPoint>0x01<gal:EndPoint> <gal:ApplicationProfileIdentifier>0x0104</gal:ApplicationProfileIdentifier> <gal:ApplicationDeviceIdentifier>0x0002</gal:ApplicationDeviceIdentifier> <gal:ApplicationDeviceVersion>0x00</gal:ApplicationDeviceVersion> <gal:ApplicationInputCluster>0x0000</gal:ApplicationInputCluster> <gal:ApplicationInputCluster>0x0003</gal:ApplicationInputCluster> <gal:ApplicationInputCluster>0x0004</gal:ApplicationInputCluster> <gal:ApplicationInputCluster>0x0005</gal:ApplicationInputCluster> <gal:ApplicationInputCluster>0x0006</gal:ApplicationInputCluster> <gal:ApplicationOutputCluster>0x0003</gal:ApplicationOutputCluster> </tns:SimpleDescriptor>

Note that four parts comprise a typical gateway command:

1. URI, which represents the specific resource that is being considered.
2. HTTP Method, which can be a POST or a GET, depending if the application is posting an XML data to the resource or requesting data from the resource.
3. Request XML Message, that can be null or it can contain the XML formatted parameters for the request as specified for the specific resource being addressed.
4. Response XML Message, similar to the request message, except that it contains a response message generated by the Gateway.

For this specific command (ConfigureEndpoint), the string "ZGD_IP_Addr:ZGD_PORT" represents the 'NwkRootURI' and the Request XML Message is a SimpleDescriptor element, where its complex type is defined in the ZigBee Gateway specification. For this type it is necessary to provide all of the network parameters such as the profileId, deviceId, etc.

2. Perform a WSN connection

Once the IPHA has assigned an end point, a WSN connection can be set up using the following command:

URI	http://"ZGD_IP_Addr:ZGD_PORT"/localnode/services/"LOCAL_SERVICE_ID"/wsnconnection?urilistener=http://"APPLICATION_IP_Addr:APPLICATION_PORT"/"CALLBACK_DESTINATION_URI"
HTTP method	POST
Request XML message	None
Response XML message	None

In this case, there is no need to provide any request XML message; it is enough to provide a URI as shown above, where the string "ZGD_IP_Addr:ZGD_PORT" represents the 'NwkRootURI', the "LOCAL_SERVICE_ID" is the local end point assigned to the

IPHA in the previous command, and the "APPLICATION_IP_Addr:APPLICATION_PORT" is the IP address where the host application is running. Also, since this specific case uses one of the "shorthand" described on page 186 (in particular the second one), the application needs to provide a URIListener that is the callback URI where any incoming messages are sent ("CALLBACK_DESTINATION_URI" can be any string such as "wsnconnection+LOCAL_SERVICE_ID").

3. Send a ZigBee message

After the previous two commands, the host application is ready to send and receive messages. Any incoming messages will be sent to the specified callback URI. If the host application wants to send a ZigBee message, the following command must be sent:

URI	`http://"ZGD_IP_Addr:ZGD_PORT"/localnode/services/"LOCAL_SERVICE_ID"/wsnconnection/message`
HTTP method	`POST`

Request XML message	`<?xml version="1.0" encoding="UTF-8"?>`
	`<tns:APSMessage xmlns:tns="http://www.zigbee.org/GWGRESTSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:gal="http://www.zigbee.org/GWGSchema" xsi:schemaLocation="http://www.zigbee.org/GWGRESTSchema/rest/rest.xsd http://www.zigbee.org/GWGSchema/rest/gal.xsd ">`
	`<gal:DestinationAddress>`
	`<gal:NetworkAddress>0x0001</gal:NetworkAddress></gal:DestinationAddress>`
	`<gal:DestinationEndpoint>0x02</gal:DestinationEndpoint>`
	`<gal:SourceEndpoint>0x01</gal:SourceEndpoint>`
	`<gal:ProfileID>0x0104</gal:ProfileID>`
	`<gal:ClusterID>0x0000</gal:ClusterID>`
	`<gal:Data>0102030405060708090a0b0c0d0e0f</gal:Data>`
	`<gal:TxOptions>`
	`<gal:SecurityEnabled>true</gal:SecurityEnabled>`
	`<gal:UseNetworkKey>true</gal:UseNetworkKey>`
	`<gal:Acknowledged>true</gal:Acknowledged>`
	`<gal:PermitFragmentation>true</gal:PermitFragmentation>`
	`</gal:TxOptions>`
	`<gal:Radius>3</gal:Radius>`
	`</tns:APSMessage>`

Response XML message	`<?xml version="1.0" encoding="UTF-8"?>`
	`<tns: APSMessageResult xmlns:tns="http://www.zigbee.org/GWGRESTSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:gal="http://www.zigbee.org/GWGSchema" xsi:schemaLocation="http://www.zigbee.org/GWGRESTSchema/rest/rest.xsd http://www.zigbee.org/GWGSchema/rest/gal.xsd ">`
	`<gal:ConfirmStatus>0x00</gal:ConfirmStatus>`
	`<gal:TxTime>0x01234567</gal:TxTime>`
	`</tns:APSMessageResult>`

Notice that to send a message the URI must include the assigned service identifier (i.e., the Endpoint element in the SimpleDescriptor XML document) previously registered, followed by the string "wsnconnection/message". The XML message that needs to be sent is a complex type called "APSMessage" described in the ZigBee Gateway specification.

Conclusion

The ZigBee Gateway Device is a stand-alone device that can meet the IP connectivity requirements of most applications. The specification and hence the devices that stem from the specification were purposely designed to be generic in nature, stopping at a level that was in no way ZigBee public application profile specific. This flexibility allows host applications to use generic gateways to communicate with any public application profile installation without modification. Application-specific details are relegated to the host.

However, it was also envisioned that the ZigBee Gateway specification and the resulting implementation would serve at an Application Programming Interface specification so that developers of Gateways could build application-specific devices. The main idea was that the specification would provide a foundation so that developers could add their application-specific code, whether in the form of an energy service portal or an in-home device, to allow homeowners to interact with any number of applications, including lighting or entertainment control infrastructure. This would allow for standardized gateways to be used in any ZigBee standards, including ZigBee Smart Energy, ZigBee Home Automation, ZigBee Telecom Service.

The ZigBee Gateway specification was focused to meet the common functionality essential to a gateway for ZigBee standards, or public application profiles. More functionality will beaded as market and application requirements emerge.