The OSGi Alliance

OSGi Residential

Release 6 Supplement
April 2018

Table of Contents

1 Introduction

Subsequent to the release of OSGi Residential Specification Release 6, an additional specification has been completed and is presented in this supplement.

1.1 Overview of the Residential Supplement Specification

1.1.1 Device Interoperability

A residential gateway can directly attach devices, for example via a USB adapter, through a home network. There is therefore a need to have a unified device abstraction, discovery and control model. For this purpose, this Specification contains the following services:

- *Device Service Specification for ZigBee™ Technology* - This specification defines how OSGi bundles can be developed to discover and control ZigBee devices on the one hand, and act as ZigBee devices and interoperate with ZigBee clients on the other hand. In particular, a Java mapping is provided for the standard representation of ZigBee devices and ZigBee clusters. See *Device Service Specification for ZigBee™ Technology* on page 7.

1.2 Version Information

This document is the OSGi Residential Release 6 Supplement.

1.2.1 OSGi Core Release 6

This specification is based on the OSGi Core Release 6. This specification can be downloaded from:

```
https://www.osgi.org/developer/specifications/
```

1.2.2 Component Versions

Components in this specification have their own specification version, independent of this specification. The following table summarizes the packages and specification versions for the different subjects.

Table 1.1 *Packages and versions*

Item	Package(s)	Version
149 *Device Service Specification for ZigBee™ Technology*	org.osgi.service.zigbee	Version 1.0
	org.osgi.service.zigbee.descriptions	
	org.osgi.service.zigbee.descriptors	
	org.osgi.service.zigbee.types	

When a component is represented in a bundle, a version attribute is needed in the declaration of the Import-Package or Export-Package manifest headers.

1.3 References

[1] *Bradner, S., Key words for use in RFCs to Indicate Requirement Levels*
http://www.ietf.org/rfc/rfc2119.txt, March 1997.

149 Device Service Specification for ZigBee™ Technology

Version 1.0

149.1 Introduction

The [1] *ZigBee Specification* is a standard wireless communication protocol designed for low-cost and low-power devices from the ZigBee Alliance. ZigBee is widely supported by various types of devices such as smart meters, lights and many kinds of sensors in the residential area. OSGi applications need to communicate with those ZigBee devices.

This specification defines how OSGi bundles can be developed to discover and control ZigBee devices on the one hand, and act as ZigBee devices and interoperate with ZigBee clients on the other hand. In particular, a Java mapping is provided for the standard hierarchical representation of ZigBee devices called the ZigBee Cluster Library. The [2] *ZigBee Cluster Library Specification* also describes the external API of a ZigBee Base Driver based upon the OSGi Device Access Specification.

149.2 Essentials

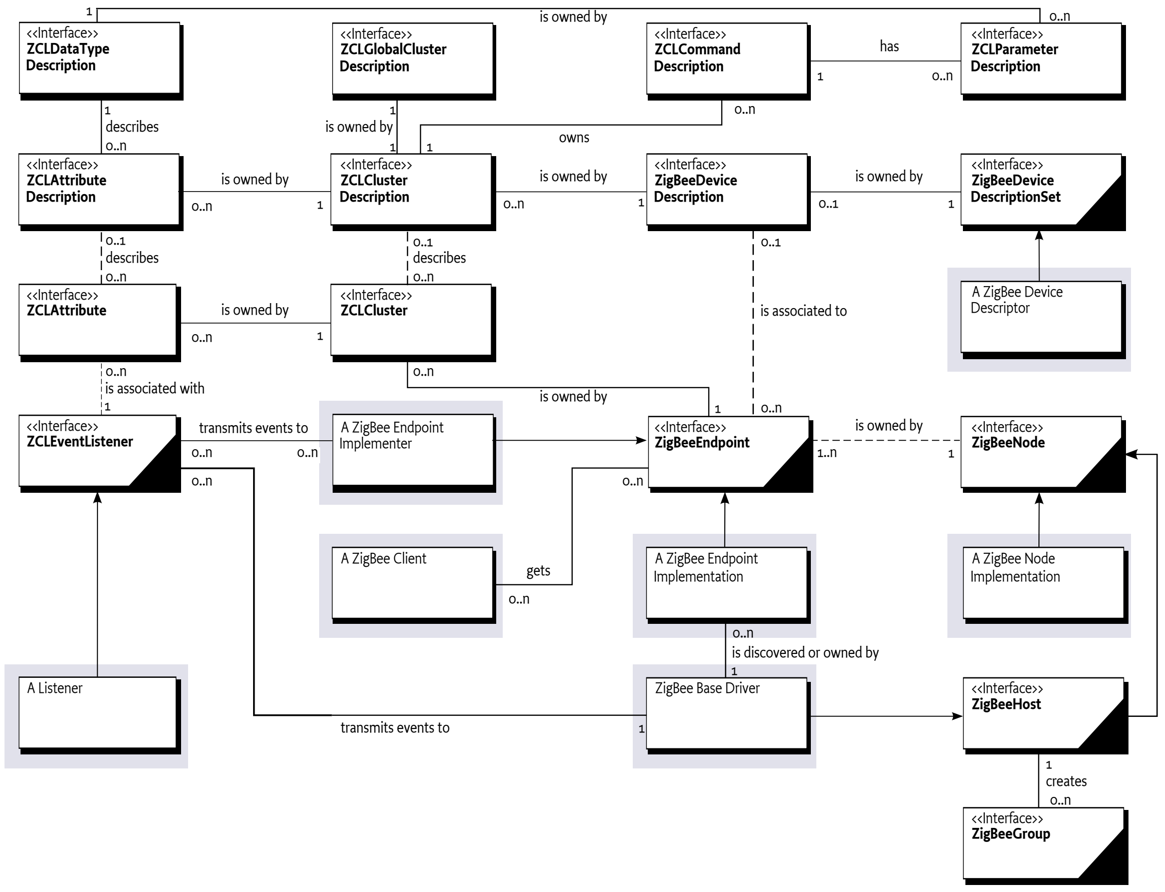
- *Scope* – This specification is limited to general device discovery and control aspects of the ZigBee and the ZigBee Cluster Library specifications. Aspects concerning the representation of specific ZigBee profiles are not addressed.
- *Transparency* – ZigBee devices discovered on the network and devices locally implemented on the platform are represented in the OSGi service registry with the same API.
- *Lightweight implementation option* – The full description of ZigBee device services on the OSGi platform is optional. Some base driver implementations may implement all the classes including ZigBee device description classes while implementations targeting constrained devices are able to implement only the part that is necessary for ZigBee device discovery and control.
- *Network Selection* – It must be possible to restrict the use of the ZigBee protocols to a selection of the connected networks.
- *Logical node type selection* – It is possible to make an OSGi-based device appearing as a ZigBee end device, a ZigBee router or a ZigBee coordinator.
- *Event handling* – Bundles are able to listen to ZigBee events.
- *Discover and Control ZigBee Endpoints as OSGi services* – available ZigBee endpoints are dynamically reified as OSGi services in the service registry.
- *Export OSGi services as ZigBee Endpoints* – available ZigBee endpoints are dynamically reified as OSGi services in the service registry.

149.3 Entities

- *ZigBee Base Driver* – The bundle that implements the bridge between OSGi and ZigBee networks.

- *ZigBee Node* – A physical ZigBee node. This entity is represented by a ZigBeeNode object. It is registered as an OSGi service by the Base Driver.
- *ZigBee Endpoint* – A logical device that defines a communication entity within a ZigBee node through which a specific application profile is carried. This concept is represented by a ZigBeeEndpoint object. Registered as an OSGi service, an endpoint can be local (implemented on the Framework) or external (implemented by another device on the network).
- *ZigBee Device Description* – Statically describes a ZigBee endpoint by providing its input/output clusters and specifies which of described commands and attributes are mandatory or not. This entity is represented by a ZigBeeDeviceDescription object.
- *ZigBee Device Description Set* – A service representing a set of ZigBeeDeviceDescription objects.
- *ZigBee Cluster* – Represents a ZigBee cluster entity, that is, a set of attributes and commands. It allows the read and write of attribute values, and allows command invocation. This concept is represented by a ZCLCluster object.
- *ZigBee Cluster Description* – Cluster description provides details about available commands and attributes for a specific Cluster. A cluster description should be constant. A cluster description holds either a Client or a Server Cluster description and refers to a global cluster description.
- *ZigBee Global Cluster Description* – Global cluster description holds the server and client cluster description as well as common information such as cluster id, description and name. This concept is represented by a ZCLGlobalClusterDescription object.
- *ZigBee Command Description* – Statically describes a specific cluster command by giving its name, id, parameters. This entity is represented by a ZCLCommandDescription object.
- *ZigBee Parameter Description* – A ZigBee parameter description has a name, a range and a data type. This entity description is represented by a ZCLParameterDescription object.
- *ZigBee Attribute* – Holds the current value of an existing cluster attribute, it allows easy (de)encoding. This concept is represented by a ZCLAttribute object.
- *ZigBee Attribute Description* – Statically describes a ZigBee Attributes (data type, name, default value). It does not hold any current value. This concept is represented by a ZCLAttributeDescription object.
- *ZigBee Event Listener Service* – A service that listens to events coming from ZigBee devices.
- *ZigBee Event* – An event generated by a ZigBee node. It contains a modified attribute value of a specific cluster. This concept is represented by a ZigBeeEvent object.
- *ZigBee Command Response Stream* – A stream is a helper that manages asynchronous responses from several endpoints that received a same request message. This entity is represented by a ZigBeeCommandResponseStream. For methods that generates a message to a unique endpoint, a Promise is used instead.
- *ZigBee Host* – The machine that hosts the code to run a ZigBee device or client. It contains information related to the Host. If the host is in the coordinator logical node type, it enables networking configuration. It is registered as an OSGi service. This concept is represented by ZigBeeHost.
- *ZigBee Client* – An application that is intended to control ZigBee devices services.
- *ZigBee Group* – Enables group management. It is registered as an OSGi service.

Figure 149.1 ZigBee Service Specification class Diagram org.osgi.service.zigbee package



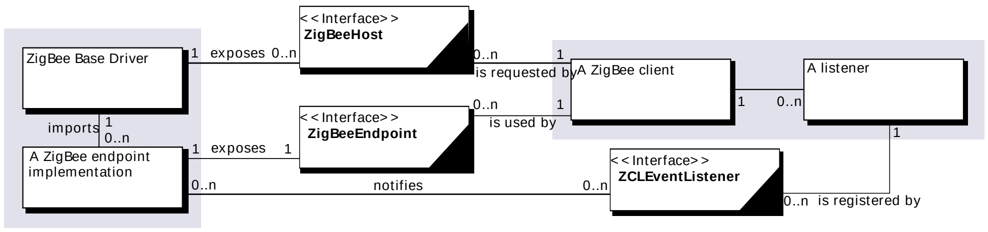
<<Interface>> ZCLDataType Description
<<Interface>> ZCLGlobalCluster Description
<<Interface>> ZCLCommand Description
<<Interface>> ZCLParameter Description
<<Interface>> ZCLAttribute Description
<<Interface>> ZCLCluster Description
<<Interface>> ZigBeeDevice Description
<<Interface>> ZigBeeDevice DescriptionSet
<<Interface>> ZCLAttribute
<<Interface>> ZCLCluster
A ZigBee Device Descriptor
<<Interface>> ZCLEventListener
A ZigBee Endpoint Implementer
<<Interface>> ZigBeeEndpoint
<<Interface>> ZigBeeNode
A ZigBee Client
A ZigBee Endpoint Implementation
A ZigBee Node Implementation
A Listener
ZigBee Base Driver
<<Interface>> ZigBeeHost
<<Interface>> ZigBeeGroup
is owned by
has
describes
owns
is associated to
is associated with
transmits events to
gets
is discovered or owned by
creates

149.4 Operation Summary

OSGi applications interact with ZigBee devices through their object representation (proxies) registered in OSGi service registry. To make a ZigBee device available as an OSGi service to ZigBee clients on the framework, an OSGi service object must be registered under the ZigBeeNode interface with the OSGi framework and an OSGi service must be registered under the ZigBeeEndpoint interface with the OSGi framework for every endpoint that is contained by the ZigBee node.

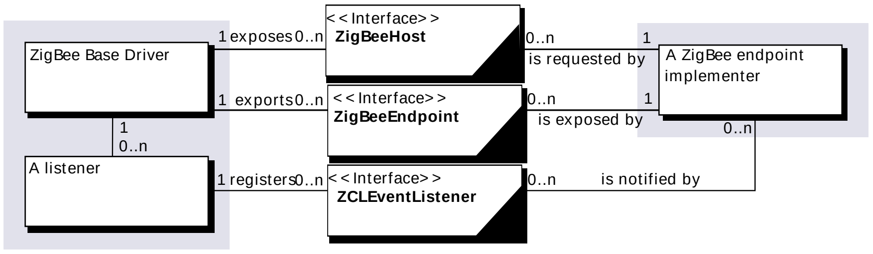
The ZigBee Base Driver is responsible for mapping networked devices into ZigBeeNode and ZigBeeEndpoint objects, through the use of a ZigBee radio chip. The latter is represented on the OSGi framework as an object implementing ZigBeeHost interface. This is called a *device import* situation (see Figure 149.2 on page 10).

Figure 149.2 *ZigBee device import*



OSGi bundles may also expose framework-internal (local) ZigBeeEndpoint instances, registered within the framework (see Figure 149.3 on page 10). The Base Driver then should emulate those objects as ZigBee endpoints associated to the ZigBee node represented by the underlying ZigBee host (ZigBee chip) on the ZigBee network. This is a *device export* situation. For more information about this process, please report to section *Implementing a ZigBee Endpoint* on page 20.

Figure 149.3 *ZigBee device export*



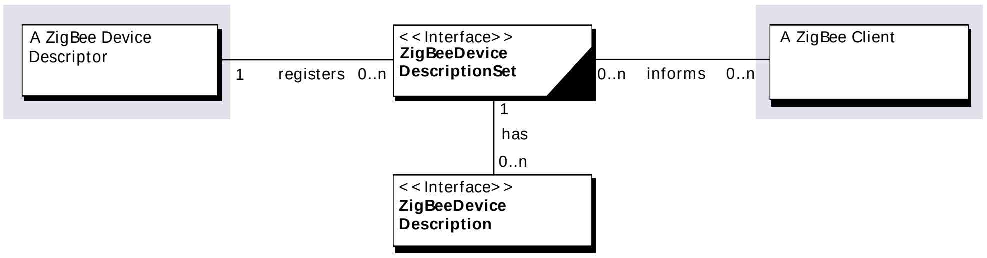
To control ZigBee devices, a bundle should track ZigBeeEndpoint services in the OSGi service registry and control them appropriately. OSGi applications can browse the clusters (ZCLCluster objects) that are discovered on every registered ZigBeeEndpoint and attributes (ZCLAttribute objects) that are discovered on every ZCLCluster. They can invoke commands on these clusters and get the current value of attributes.

Several methods obey an asynchronous mechanism. For instance, ZigBee command invocation is made through the call to ZCLCluster invoke method that returns a Promise. When the command response is received, the Promise is resolved and Promise.getValue() returns the expected response value. The Promise is resolved by the base driver in the device import situation and by the invoked local ZCLCluster in the device export situation. A ZCLCommandResponseStream is used instead of a Promise in case of a method that generates a message broadcast (or groupcast) to potentially several endpoints.

OSGi bundles – called listeners in Figure 149.1 – subscribe to attribute value changes through the Whiteboard Pattern ([6] *Listeners considered harmful: The whiteboard pattern*). They register an object under the ZCLEventListener interface with properties identifying a ZigBee attribute and a special event filter. This registration is conveyed as a ZigBee configure report command on the ZigBee network in the device import situation. Reports are received by the base driver and transmitted as notifyEvent(ZigBeeEvent) method calls on relevant ZCLEventListener services in this situation. Local ZigBeeEndpoint objects directly call these methods to notify listeners with reports in the export situation. The Base Driver conveys events received through a ZCLEventListener to networked the ZigBee endpoints that have subscribed to relevant reports.

Endpoints, clusters, commands and attributes are specified by ZigBee Alliance or vendor-specific descriptions. Those descriptions may be provided on the OSGi platform by any bundle through the registration of ZigBeeDeviceDescriptionSet services (see Figure 149.4 on page 11). Every service is a set of descriptions that enables applications to retrieve information about the clusters, commands, attributes supported by the described type of endpoint.

Figure 149.4 *Using a set of device descriptions*

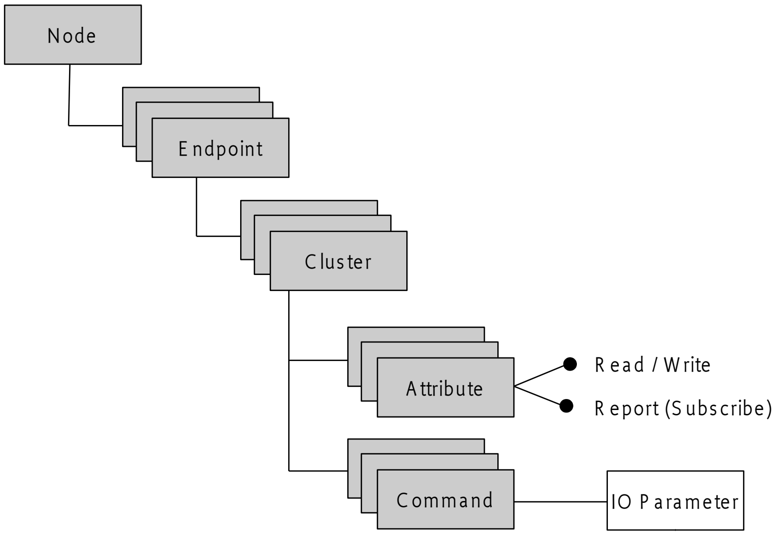

A ZigBee Device Descriptor
1
registers
0..n
<<Interface>> ZigBeeDevice DescriptionSet
0..n
informs
0..n
A ZigBee Client
1
has
0..n
<<Interface>> ZigBeeDevice Description

149.5 ZigBee Base Driver

Most of the functionality described in the operation summary is implemented in a ZigBee base driver. A ZigBee base driver is a bundle that implements the ZigBee protocols and handles the interaction with bundles that use the ZigBee devices. It must discover ZigBee devices on the ZigBee network and map each discovered device into an OSGi registered `ZigBeeNode` service. It must also export, on the ZigBee Network, `ZigBeeEndpoint` services (programmatically registered as OSGi services).

`ZigBeeNode` object also provides simple methods to handle standard ZigBee Device Object networking features: `getLinksQuality()`, `getRoutingTable()`, and `leave()`.

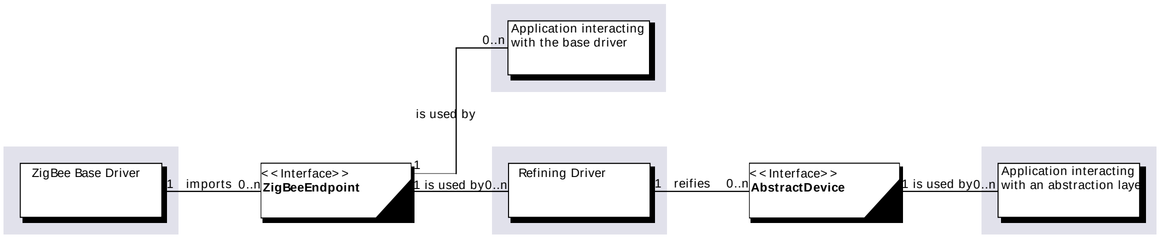
Figure 149.5 *ZigBee Cluster Library model*



All interfaces corresponding to the ZigBee Cluster Library model (see Figure 149.5 on page 11) must be implemented in order to discover and control asynchronously ZigBee devices. Classes related to the description of these entities named with suffix `Description` may optionally be implemented. This rule follows the fact that ZigBee device descriptions are not downloadable on the device itself and are often given to developers in an out-of-band manner.

Several base drivers may be deployed on a residential OSGi device, one for every supported network technology. An OSGi device abstraction layer may then be implemented as a layer of refining drivers above a layer of base drivers. The refining driver is responsible for adapting technology-specific device services registered by the base driver into device services of another model (see AbstractDevice interface in Figure 149.6 on page 12). In the case of a generic device abstraction layer, the model is agnostic to technologies.

Figure 149.6 *The ZigBee Base Driver and a refining driver representing devices in an abstract model*


Application interacting with the base driver
0..n
is used by
1
ZigBee Base Driver
1
imports
0..n
<<Interface>> ZigBeeEndpoint
1 is used by
0..n
Refining Driver
1
reifies
0..n
<<Interface>> AbstractDevice
1 is used by
0..n
Application interacting with an abstraction laye

The ZigBee Alliance defines their own abstract model with ZigBee Profiles, for example, Home Automation, Lighting, and refining drivers may provide the implementation of all ZigBee standard devices with ZigBee-specific Java interfaces. The AbstractDevice interface of Figure 149.6 on page 12 is then replaced by a ZigBee-specific Java interface in that case. The need and the choice of the abstraction depends on the targeted application domain.

149.6 ZigBee Node

A ZigBee node represents a physical ZigBee device and should adhere to a specific application profile that can be either public or private. Profiles define the environment of the application, the type of devices and the clusters used for them to communicate.

A physical device is reified and registered as a ZigBeeNode service in the Framework. A ZigBee node holds several ZigBee endpoints that are registered as ZigBeeEndpoint objects.

ZigBee nodes properties are defined in the ZigBee Specification. These properties must be registered in the OSGi Framework services registry so they are searchable. ZigBeeNode must be registered with the following properties:

- IEEE_ADDRESS – (zigbee.node.ieee.address/BigInteger) specifies the IEEE Address of a ZigBee node.
- LOGICAL_TYPE – (zigbee.node.description.node.type/Short) specifies a device logical type.
- MANUFACTURER_CODE – (zigbee.node.description.manufacturer.code/Integer) specifies a manufacturer code that is allocated by the ZigBee Alliance, relating to the device manufacturer.
- POWER_SOURCE – (zigbee.node.power.source/Boolean) is the ZigBee power source, that is, 3rd bit of "MAC Capabilities" in Node Descriptor, which is set to 1 if the current power source is mains power, set to 0 otherwise.
- RECEIVER_ON_WHEN_IDLE – (zigbee.node.receiver.on.when.idle/Boolean) represents the ZigBee receiver on when idle, that is, 4th bit of "MAC Capabilities" in Node Descriptor, which is set to 1 if the device does not disable its receiver to conserve power during idle periods, set to 0 otherwise.
- PAN_ID – (zigbee.node.pan.id/Integer) (Personal Area Network Identifier) is a 16-bit value that identifies a ZigBee network. Every ZigBeeNode object is associated to a PAN ID, which can be retrieved through the getPanId() method.
- EXTENDED_PAN_ID – (zigbee.node.pan.extended.id/BigInteger) Extended PAN ID is a 64-bit numbers that uniquely identify a PAN. It is intended to enhance selection of a PAN and enable recognition of network after PAN ID change (due to a previous conflict). getExtendedPanId() returns the network extended PAN ID if specified.

  Note: PAN_ID and EXTENDED_PAN_ID are optional, but at least one of these properties MUST be specified.
- DEVICE_CATEGORY (see the OSGi Device Access Specification) – (DEVICE_CATEGORY) describes a table of the categories to which the device belongs. One of the values MUST be “ZigBee” (DEVICE_CATEGORY).

Additional properties (defined in the OSGi Device Access Specification) may be set:

- DEVICE_DESCRIPTION – if the complex descriptor of the device is available, the value MUST be set and MUST be the value returned by getModelName().
- DEVICE_SERIAL – if the complex descriptor of the device is available, the value MUST be set and MUST be the value returned by getSerialNumber().

Finally, service.pid property MUST be set.

ZigBee nodes describes themselves using descriptor data structures:

- getNodeDescriptor() – Returns a Promise object that is asynchronously resolved with a ZigBeeNodeDescriptor object representing the Node Descriptor which contains information about the node capabilities. On failure, the promise is resolved with an exception instead.
- getComplexDescriptor() – Returns a Promise object that is asynchronously resolved with a ZigBeeComplexDescriptor object representing the Complex Descriptor which contains extended information for each device description contained in this node. On failure, the promise is resolved with an exception instead, especially an exception with NO_DESCRIPTOR error code if no Complex Descriptor is provided.
- getPowerDescriptor() – Returns a Promise object that is asynchronously resolved with a ZigBeePowerDescriptor object representing the Power Descriptor which contains power-related information of this node. On failure, the promise is resolved with an exception instead, especially an exception with NO_DESCRIPTOR error code if no Power Descriptor is provided.
- getUserDescription() – Returns a Promise object that is asynchronously resolved with the unique field named "User description" of the User Descriptor, which contains information that allows the user to identify the device using user-friendly character string. On failure, the promise is resolved with an exception instead, especially an exception with NO_DESCRIPTOR error code if no User Descriptor is provided.

ZigBeeNode objects provide invoke methods to send network frames within ZDP layer, while invoking ZigBee Cluster Library (ZCL) commands is enabled on ZCLCluster objects. ZCL commands can be however broadcast on a ZigBee node thanks to broadcast methods. Broadcasting enables the sending of a ZCL command to all clusters identified with an identifier of all endpoints available on the targeted ZigBee node.

All discovered ZigBee nodes in the local networks are registered under the ZigBeeNode interface within the OSGi Framework. Every time a ZigBee node appears or quits the network, the associated OSGi service is registered or unregistered in the OSGi service registry. Thanks to the ZigBee Base Driver, the OSGi service availability in the registry mirrors ZigBee device availability on ZigBee networks. Using a remote ZigBee node thus involves tracking ZigBeeNode services in the OSGi service registry. The following code illustrates how this can be done. The sample Controller class extends the ServiceTracker class so that it can track all ZigBeeNode services and add them to a user interface, such as a remote controller application. The friendly name of this node is retrieved in order to be printed on the user interface.

```
class Controller extends ServiceTracker {
    UI ui;
    Controller( BundleContext context ) {
        super( context, ZigBeeNode.class, null );
    }
    public Object addingService( ServiceReference ref ) {
        ZigBeeNode node = (ZigBeeNode)super.addingService(ref);
        ui.addNode( node );
        return node;
    }
    public void removedService( ServiceReference ref, Object endpoint ) {
        ui.removeNode( (ZigBeeNode) node );
        super.removedService(ref);
    }
    ...
}

public class UI {
    public void addNode(ZigBeeNode node) {
        final Promise p = node.getUserDescription();
        p.onResolve(new Runnable() {
            public void run() {
                try {
                    String friendlyName = (String) p.getValue();
                    createUINode(node, friendlyName);
                } catch (InvocationTargetException e) {
                     log.info("Get User Description command returned "
                               + "a failure: " + e.getCause() + ".");
                     createUINode(node, "No friendly name");
                } catch (InterruptedException e) {
                     e.printStackTrace();
                }
            }
        });
    }
...
}
```

149.7 ZigBee Endpoint

Communication between devices is done through an addressable component called ZigBee endpoint which holds a number of ZigBee clusters. A ZigBee cluster represents a functional unit in a device.

An endpoint defines a communication entity within a device through which a specific application is carried. So, it represents a logical device object used for communication.

For example, a remote control light might allocate Endpoint 7 for the control of lights in the master bedroom, Endpoint 9 to manage the heating and air conditioning system, and Endpoint 14 for controlling the security system.

The ZigBee specification defines that a maximum of 240 Endpoints is allowed per ZigBeeNode. Endpoint 0, also called the ZigBee Device Object (ZDO), is reserved for the management operations on both ZigBee node and ZigBee endpoints, endpoint 255 is reserved for broadcasting to all endpoints, endpoints 241-254 are reserved for future use.

Endpoint 0 and endpoint 255 capabilities are not exposed, only endpoints 1-240 should be registered as services. Endpoints are registered under the ZigBeeEndpoint interface with the following properties:

- IEEE_ADDRESS – (zigbee.node.ieee.address/BigInteger) specifies the IEEE Address of the parent node.
- ENDPOINT_ID – (zigbee.endpoint.id/Short) specifies the endpoint address within the node. Applications shall only use endpoints 1-240.
- PROFILE_ID – (zigbee.device.profile.id/Integer) identifies the profile that the endpoint belongs to. The profile can be either a ZigBee Alliance standard profile or a vendor-specific profile. The ZigBee specification defines several profile identifiers, and some others are vendor specific.

- HOST_PID – (zigbee.endpoint.host.pid/String) – The ZigBee local host identifier is intended to uniquely identify the ZigBee local host, since there could be many hosts on the same platform. All the endpoints that belong to a specific network MUST specify the value of the associated host number.
- DEVICE_ID – (zigbee.device.id/Integer) identifies the device description supported by this endpoint. Like for profile identifiers, the ZigBee specification defines several device identifiers. Vendors are also able to define specific device identifiers.
- DEVICE_VERSION – (zigbee.device.version/Integer) specifies the device description version supported by this endpoint.
- INPUT_CLUSTERS – (zigbee.endpoint.clusters.input/Integer[]) specifies the list of input cluster ids supported by this endpoint. Input cluster are called Server cluster.
- OUTPUT_CLUSTERS – (zigbee.endpoint.clusters.output/Integer[]) specifies the list of output cluster ids supported by this endpoint. Output cluster are called Client cluster.
- DEVICE_CATEGORY (see the OSGi Device Access Specification) – (DEVICE_CATEGORY) describes a table of the categories to which the device belongs. One of the values MUST be "ZigBee" (DEVICE_CATEGORY).

Finally, service.pid property MUST be set. In device import case, it is a free unique identifier that enables OSGi ZigBee clients to identify any imported endpoint across node reboots. It may concatenate the endpoint IEEE address, a separator, for example, '_', and the endpoint ID. In endpoint export case, it is a free unique identifier that enables the base driver to identify any exported endpoint across local bundle restarts. In this case, service.pid property may concatenate bundle identifier, a separator, for example, '_', and a number.

A ZigBeeEndpoint may contain a number of input or output clusters. ZigBeeEndpoint provides getServerCluster(int) and getClientCluster(int) to return a specific server input or client output cluster.

Every endpoint must provide a simple descriptor. getSimpleDescriptor() returns a Promise object that is asynchronously resolved with a ZigBeeSimpleDescriptor object which contains general information about the endpoint or with an exception in case of a failure.

ZigBeeEndpoint interface provides two methods to bind and unbind ZigBee clusters: bind(String,int) and unbind(String,int). The entity that wants to bind clusters is responsible for initializing, maintaining and removing the bindings across ZigBeeEndpoint service events. This entity is the local OSGi Application that asked this binding or the ZigBee Base Driver if the binding has been requested by a remote ZigBee node.

ZigBeeEndpoint interface provides a getBoundEndPoints(int) method that provides the table of bound ZigBeeEndpoint objects identified by their service PIDs.

149.8 ZigBee Device Description

A ZigBee endpoint may have a description used to describe his input and output clusters, and which of these clusters are mandatory or optional. A ZigBeeDeviceDescription object provides associated information about an endpoint.

149.9 ZigBee Device Description Set

ZigBeeDeviceDescriptionSet objects may be registered as OSGi services by any bundle. A ZigBeeDeviceDescriptionSet provides getDeviceSpecification(int,short) which returns the device description, if provided, for the identified endpoint, or null otherwise. A ZigBeeDeviceDescriptionSet service should be registered with the following properties:

- VERSION – (zigbee.profile.version/Short) The application profile version.
- PROFILE_ID – see ZigBeeEndpoint.PROFILE_ID property.
- PROFILE_NAME – (zigbee.profile.name/String) The profile name.
- MANUFACTURER_CODE – see ZigBeeNode.MANUFACTURER_CODE property.
- DEVICES – (zigbee.profile.devices/Integer[]) comma separated list of devices identifiers supported by the set.

149.10 ZCL Cluster

Devices communicate with each other by means of clusters, which may be inputs to or outputs of the device. For example, ZigBee Home Automation profile provides a cluster dedicated to the control of lighting subsystems. Clusters are represented under ZCLCluster interface.

ZCLCluster objects combine one or more ZigBee commands (or frames) and ZCLAttribute objects.

ZCLCluster provides some methods for reading and writing attributes values:

- readAttributes(ZCLAttributeInfo[]) – The ZigBee Base driver MAY generate the read attributes command on behalf of the OSGi application that is invoking this method. The latter returns a Promise object that is asynchronously resolved with a Map of ZCLReadStatusRecord identified by their attribute identifiers. On failure, the promise is resolved with an exception instead.
- writeAttributes(boolean,Map) – The ZigBee Base driver generates the write attributes command on behalf of the OSGi application that is invoking this method. The boolean undivided parameter specifies that if any attribute cannot be written, for example, if an attribute is not implemented on the device, or a value to be written is outside the valid range, no attribute values are changed.

ZCLCluster objects use ZCLFrame to invoke ZigBee commands :

- invoke(ZCLFrame) – a sequence of bytes represents the command frame. The source endpoint is not specified in this method call. To send the appropriate message on the network, the base driver must generate a source endpoint. The latter must not correspond to any exported endpoint.
- invoke(ZCLFrame,String) – a sequence of bytes represents the command frame, and exportedServicePID is the source endpoint of the command request. In targeted situations, the source endpoint is the valid service PID of an exported endpoint.

  A Promise is returned and manages the command response asynchronously.

149.11 ZCL Cluster Description

A ZCLClusterDescription describes the server or client part of a ZCLCluster. It lists the available commands and attributes for this client or server cluster.

Every cluster client and server may have attributes (see [2] *ZigBee Cluster Library Specification*, Chapter 2.2.1), received and generated commands. ZCLClusterDescription provides methods to describe commands, attributes and retrieve general cluster information.

149.12 ZCL Global Cluster Description

ZCLGlobalClusterDescription describes a cluster general information: id, name, description. It provides the ZCLClusterDescription for both client and server part of this cluster.

149.13 ZigBee Command Description

ZCLCommandDescription describes a ZigBee command.

ZCLCommandDescription contains ZCLParameterDescription objects which describe the command parameters.

All clusters (server and client) shall support generation, reception and execution of the default response command.

Every cluster (server or client) that implements attributes shall support reception of, execution of, and response to all commands to discover, read, write, report, configure reporting of, and read reporting configuration of these attributes. Generation of these commands is application dependent.

149.14 ZigBee Attribute

A ZigBee cluster is associated with a set of attributes. Every attribute is represented by an object implementing ZCLAttribute interface extending ZCLAttributeInfo. ZCLAttribute provides getValue() and setValue(Object) to retrieve and set the current attribute value with the use of a Promise, which returns the response asynchronously.

149.15 ZigBee Attribute Description

A ZCLAttributeDescription also extends ZCLAttributeInfo and describes information about a specific ZCLAttribute.

149.16 ZCL Data Type Description

ZCLAttributeInfo and ZCLParameterDescription provide getDataType() and getDataTypeDescription() methods, respectively, which return ZCLDataTypeDescription objects. One object is associated to every ZigBee data type, see ZigBeeDataTypes constants in ZigBee Data Types section below.

149.17 ZCL Simple Type Description

ZCLSimpleTypeDescription extends ZCLDataTypeDescription interface to provide the following methods:

- serialize(ZigBeeDataOutput,Object) – Serializes a Java object corresponding to the Java data type given by getJavaDataType(), and adds the result to the given ZigBeeDataOutput according to ZigBee Cluster Library.
- deserialize(ZigBeeDataInput) – Deserializes the given data into a Java object of the Java data type given by getJavaDataType().

Every ZigBee data type is associated to a ZCLSimpleTypeDescription implementation, except ZigBee Array, Bag, Set, and Structure types.

149.18 Promise and Response Stream objects

Promise and ZCLCommandResponseStream objects handle ZigBee network communication latency and errors. An org.osgi.util.promise.Promise is immediately returned by every method that generates a message exchange with one ZigBee endpoint, that is, the sending of a message to this endpoint and the handling of a unique response from this endpoint. No exception is thrown by this method. The Promise handles the expected result and any occurring error asynchronously. The caller can either get a callback when the Promise is resolved with a value or an error, or the Promise can be used in chaining. Both onResolve(Runnable) callbacks and then(Success, Failure) chaining can be repeated any number of times, even after the Promise has been resolved. When the Promise is resolved, callbacks and chaining are called, Promise.isDone() returns true, and either Promise.getValue() returns a value or Promise.getFailure() returns a relevant Throwable. The type of the value and the type of Throwable are specific to the method returning the Promise. In import situations, the base driver fails the Promise when a timeout is reached before any response is received on the network. The returned failure is then a ZigBeeException with TIMEOUT error code. The associated timeout is given by getCommunicationTimeout(). It can be set by calling setCommunicationTimeout(long) on the appropriate ZigBeeHost object.

A ZCLCommandResponseStream is immediately returned by every method that generates the sending of a message to potentially several endpoints with the expectation of a response from several of them. No exception is thrown by this method. The caller can register a handler with forEach(Predicate). The unique method of the handler is called with a ZCLCommandResponse every time a response is received from one of the targeted endpoints until the ZCLCommandResponseStream is closed. The latter is closed either when the handler returns false to the test method or close() is called. ZCLCommandResponseStream is used for the following message invocation types:

- Broadcasting: Sending a message to all available endpoints of a specific type and receiving responses from each of them, see broadcast(int,ZCLFrame,String).
- Groupcasting: Sending a message to the endpoints of a specific type in a group of endpoints and receiving responses from each of them, see groupcast(int,ZCLFrame,String).
- Nodecasting: Sending a message to the endpoints of a specific type on a node and receiving responses from each of them, see broadcast(int,ZCLFrame,String).

149.19 ZigBee Data Types

The ZigBeeDataTypes class provides all standard ZigBee data type identifiers as constants. It should be noted that actual value of these constants do not necessarily match the values assigned in the ZCL specification for these data types.

The org.osgi.service.zigbee.types package contains an implementation class for each of the ZCL scalar data types, with the exception of NO_DATA and UNKNOWN . Each of these classes declares a static getInstance() method, that returns a singleton of the class itself. Moreover, because they implement the ZCLSimpleTypeDescription interface, they provide methods for getting some metadata information about the ZCL data type they represent, like the relative ZigBeeDataTypes constant (getId() method) and the Java class the ZCL data type is mapped to. Methods to marshal and unmarshal the data type into a ZigBeeDataInput stream and from a ZigBeeDataOutput stream according to the ZigBee specification, are provided as well.

Here is the table of encoding relations between ZigBee types and Java types, used in this specification:

Table 149.1 Mapping of ZCL Data Types to Java

ZigBeeDataType constant	ZigBee type	Java Type
NO_DATA	No data	
GENERAL_DATA_8	8-bit data	Byte
GENERAL_DATA_16	16-bit data	Short
GENERAL_DATA_24	24-bit data	Integer
GENERAL_DATA_32	32-bit data	Integer
GENERAL_DATA_40	40-bit data	Long
GENERAL_DATA_48	48-bit data	Long
GENERAL_DATA_56	56-bit data	Long
GENERAL_DATA_64	64-bit data	Long
BOOLEAN	Boolean	Boolean
BITMAP_8	8-bit bitmap	Byte
BITMAP_16	16-bit bitmap	Short
BITMAP_24	24-bit bitmap	Integer
BITMAP_32	32-bit bitmap	Integer
BITMAP_40	40-bit bitmap	Long
BITMAP_48	48-bit bitmap	Long
BITMAP_56	56-bit bitmap	Long
BITMAP_64	64-bit bitmap	Long
UNSIGNED_INTEGER_8	Unsigned 8-bit integer	Short
UNSIGNED_INTEGER_16	Unsigned 16-bit integer	Integer
UNSIGNED_INTEGER_24	Unsigned 24-bit integer	Integer
UNSIGNED_INTEGER_32	Unsigned 32-bit integer	Long
UNSIGNED_INTEGER_40	Unsigned 40-bit integer	Long
UNSIGNED_INTEGER_48	Unsigned 48-bit integer	Long
UNSIGNED_INTEGER_56	Unsigned 56-bit integer	Long
UNSIGNED_INTEGER_64	Unsigned 64-bit integer	BigInteger
SIGNED_INTEGER_8	Signed 8-bit integer	Byte
SIGNED_INTEGER_16	Signed 16-bit integer	Short
SIGNED_INTEGER_24	Signed 24-bit integer	Integer
SIGNED_INTEGER_32	Signed 32-bit integer	Integer
SIGNED_INTEGER_40	Signed 40-bit integer	Long
SIGNED_INTEGER_48	Signed 48-bit integer	Long
SIGNED_INTEGER_56	Signed 56-bit integer	Long
SIGNED_INTEGER_64	Signed 64-bit integer	Long
ENUMERATION_8	8-bit enumeration	Short
ENUMERATION_16	16-bit enumeration	Integer
FLOATING_SEMI	Semi-precision float	Float
FLOATING_SINGLE	Single precision float	Float
FLOATING_DOUBLE	Double	Double
CHARACTER_STRING	Character string	String
OCTET_STRING	Octet string	byte[]
LONG_CHARACTER_STRING	Character string	String
LONG_OCTET_STRING	Octet string	byte[]
ARRAY	Array	

ZigBeeDataType constant	ZigBee type	Java Type
STRUCTURE	Structure	
SET	Set	
BAG	Bag	
CLUSTER_ID	Cluster ID	Integer
ATTRIBUTE_ID	Attribute ID	Integer
BACNET_OID	BACnet OID[1] (Unsigned 32-bit integer)	Long
TIME_OF_DAY	Time of day	byte[4]
DATE	Date	byte[4]
UTC_TIME	UTC Time	Long
IEEE_ADDRESS	IEEE address (MAC-48,EUI-48/64)	BigInteger
SECURITY_KEY_128	128-bit Security Key	byte[8]
UNKNOWN	Unknown	

[1] BACnet OID (Object identifier) data type is included to allow interworking with BACnet (see [5] *ASHRAE 135-2004 Standard*). The format is described in the referenced standard.

149.20 Implementing a ZigBee Endpoint

OSGi services can also be exported as ZigBee endpoints to the local networks, in a way that is transparent to typical ZigBee devices endpoints. This allows developers to bridge legacy devices to ZigBee networks. A ZigBeeEndpoint MUST be registered with the following properties to export an OSGi service as a ZigBee endpoint:

- ZIGBEE_EXPORT – To indicate that the endpoint is an exportable endpoint.

An OSGi platform can be connected to multiple ZigBee networks. HOST_PID, PAN_ID and EXTENDED_PAN_ID are used to select the appropriate network. At least one of these properties MUST be specified. If provided, HOST_PID has priority over PAN_ID and EXTENDED_PAN_ID to identify the host that is targeted for export.

In addition, the ZigBeeEndpoint service MUST declare the same properties as an imported endpoint. The bundle registering endpoint services must make sure these properties are set accordingly or that none of these properties are set. In case a ZigBee host is not initialized yet or the base driver is not active on the OSGi framework, an endpoint implementation MAY not have any of the above identifiers.

If the Base Driver is active and at a ZigBee host is started, then the Base Driver makes an attempt to export the endpoint on the ZigBee network associated to the ZigBee HOST_PID, PAN_ID or EXTENDED_PAN_ID. The associated ZigBeeNode object MUST be one of the available ZigBeeHost objects. Every time an endpoint is registered or unregistered with both ZIGBEE_EXPORT and PAN_ID and/or EXTENDED_PAN_ID properties set, the associated ZigBeeHost service is modified accordingly (getEndpoints() returns a different array of ZigBeeEndpoint objects).

If - and only if - an error is detected on the properties of the ZigBee endpoint to be exported, then the Base Driver calls the notExported(ZigBeeException) method with a relevant ZigBeeException object as the input argument. The method SHOULD be called even if a ZigBee Host is not started.

The endpoint has to be registered with an ID that is unique. If the chosen ID already exists as a property of a local endpoint with the same host or if it already exists in an optional cache of the base driver, the base driver calls the notExported(ZigBeeException) method with the ZigBeeException object as the input argument with OSGI_EXISTING_ID error code. The base driver may keep IDs in a cache for endpoints that might come back in the registry. The range of potential IDs is 1-240 according to [1] *ZigBee Specification*.

The reader must note that a same ZigBeeEndpoint object cannot be registered several times with distinct PAN IDs since thegetNodeAddress() method can only return one ZigBee node address.

If the PAN ID corresponds to more than one ZigBeeHost service, the ZigBeeEndpoint MUST define the Extended PAN ID property which uniquely identifies a ZigBee network. The base driver will call notExported(ZigBeeException) with the error code OSGI_MULTIPLE_HOSTS if the Extended PAN ID property is not properly defined in this specific situation.

Moreover, if the HOST PID corresponds to more than one ZigBeeHost, the base driver will also call notExported(ZigBeeException) with the error code OSGI_MULTIPLE_HOSTS.

149.21 Event API

Eventing is available in import and export situations:

- External events from the network must be dispatched to listeners inside the OSGi Service Platform. The ZigBee Base driver is responsible for mapping the network events to internal listener events.
- Implementations of ZigBee endpoints must send out events to local listeners. The ZigBee Base driver dispatches events to the network from its own listeners.

ZigBee events are sent using the whiteboard pattern, [6] *Listeners considered harmful: The whiteboard pattern*, in which a bundle interested in receiving the ZigBee events registers an object implementing the ZCLEventListener interface. The service MUST be registered with PAN_ID and/or EXTENDED_PAN_ID properties. These properties indicate the network targeted by the listener since an OSGi platform can host multiple ZigBee networks.

A filter can be set to limit the events for which a bundle is notified. The ZigBee Base driver must register a ZCLEventListener service for every attribute report configured in the configure reporting commands it receives from the network.

The filter refers to the combination of the properties registered with the ZCLEventListener service. Each ZCLEventListener MUST be registered with all the following mandatory properties:

- ID – (zigbee.cluster.id/Integer) Only events generated by endpoints matching a specific cluster are delivered.
- ID – (zigbee.attribute.id/Integer) Only events generated by endpoints matching a specific attribute are delivered.
- ATTRIBUTE_DATA_TYPE – (zigbee.attribute.datatype/Short) The Attribute data type field contains the data type of the attribute that is to be reported (see [2] *ZigBee Cluster Library Specification* 2.4.7.1.4 Attribute Data Type Field).

The optional properties are:

- IEEE_ADDRESS – (zigbee.node.ieee.address/BigInteger) Only events generated by endpoints matching the specific node are delivered.
- ENDPOINT_ID – (zigbee.endpoint.id/Short) Only events matching a specific endpoint are delivered.
- MIN_REPORT_INTERVAL – (zigbee.attribute.min.report.interval/Integer) The minimum interval, in seconds, between issuing reports of the specified attribute (see [2] *ZigBee Cluster Library Specification.* – 2.4.7.1.5).
- MAX_REPORT_INTERVAL – (zigbee.attribute.max.report.interval/Integer) The maximum interval, in seconds, between issuing reports of the specified attribute (see [2] *ZigBee Cluster Library Specification.* 2.4.7.1.6).

- REPORTABLE_CHANGE – (zigbee.attribute.reportable.change/Double) The minimum change to the attribute that will result in a report being issued. This property is mandatory if the data type is analog. If the data type is digital, the base driver will ignore it.

If the endpoint sets a timeout between two attribute reports, the notifyTimeOut(int) method is then called with the timeout argument. In the import situation, the base driver calls this method on the relevant listeners when it receives a configure reporting command with a set TIMEOUT_PERIOD field (see [2] *ZigBee Cluster Library Specification* 2.4.7 Configure Reporting Command). In the export situation, the local endpoint calls this method on relevant listeners and, in case the base driver is one of the notified listeners, it sends a configure reporting request with the appropriate TIMEOUT_PERIOD field to interested endpoints on the network.

A ZigBee event is represented by a ZigBeeEvent object.

If an event is generated by either the local endpoint or via the base driver for an external device, the notifyEvent(ZigBeeEvent) method is called on all registered ZCLEventListener services for which the source event matches the service properties. The way events must be delivered is the same as described in Delivering Events in the Life Cycle Layer chapter of the *OSGi Core Release 6* specification.

The ZigBee base driver SHOULD group subscriptions into one configure reporting request to the targeted ZigBee device. It SHOULD also notify every listener with respect to their specific expectations.

149.22 Monitoring Events and Sending Commands

In the example below, a button of the user interface monitors the state (on or off) of a smart plug and enables the user to switch the plug on and off. To monitor the plug state, a ZCLEventListener is registered with the properties related to the node, endpoint, cluster and attribute representing the plug and its state. When an appropriate event is sent on the network, the base driver (or a local endpoint implementer) notifies the listener. The listener then changes the state value shown by the button. When the user clicks on the button, a command is invoked on the plug.

```
public class UIOnOffButton implements ZCLEventListener {
    public UIOnOffButton(BigInteger ieeeAddress, Short endpointId, Integer
            clusterId, Integer attributeId, Short dataType,
            BundleContext bc) {
        Dictionary properties = new Hashtable();
        properties.put(ZigBeeNode.IEEE_ADDRESS, ieeeAddress);
        properties.put(ZigBeeEndpoint.ENDPOINT_ID, endpointId);
        properties.put(ZCLCluster.ID, clusterId);
        properties.put(ZCLAttribute.ID, attributeId);
        properties.put(ZCLEventListener.ATTRIBUTE_DATA_TYPE, dataType);
        // events will be filtered by the basedriver call notifyEvent() method
        // only when the event comes from a node, endpoint, cluster, attribute
        // matching these properties
        bc.registerService(ZCLEventListener.class.getName(), this, properties);
    }

    public void notifyEvent(ZigBeeEvent event) {
        // change the attribute value of the UICluster
        Object value = event.getValue();
        changeUIValue(value);
    }

    public void notifyTimeOut(int timeout) {
          log.info("Timeout notified");
    }

    public void onFailure(ZCLException e) {
          log.info("Failure registering the listener: " + e);
    }

    public void changeUIValue(Object value) {
    ....
    }

    public void onClick() {
        // the button has been clicked
        // get the ZCLCluster
        ServiceReference[] srs = bundleContext.getServiceReferences(
                          ZigBeeEndpoint.class.getName(),
                          "(&(" + ZigBeeNode.IEEE_ADDRESS + "=" + ieeeAddress
                          + ")(" + ZigBeeEndpoint.ENDPOINT_ID + "=" + endpointID
                          + "))");
        if (srs.length>0){
            ZCLCluster onOffCluster =
                ((ZigBeeEndpoint) bundleContext.getService(srs[0]))
                          .getServerCluster(ZCL_ONOFF_CLUSTER_ID);
            if (onOffCluster != null) {
                final Promise p = onOffCluster.invoke(new ToggleCommand());
                p.onResolve(new Runnable() {
                    public void run(){
                        try {
                            ZCLFrame frame = (ZCLFrame) p.getValue();
                            log.info("toggle command returned success.");
                        } catch (InvocationTargetException e) {
                            log.info("toggle command returned a failure: "
                                      + e + ”.”);
                        } catch (InterruptedException e) {
                             e.printStackTrace();
                        }
                    }
                });
            }
        }
    }

    class ToggleCommand implements ZCLFrame {
        ...
    }
...
}
```

149.23 ZCL Exception

The ZCLException extends the ZigBeeException. It holds information about the different ZigBee ZCL layers. Error codes specified by ZigBee Alliance are conveyed by the errorCode field of ZCLException objects.

149.24 ZDP Exception

The ZDPException extends the ZigBeeException. It holds information about the ZigBee ZDP layer. Error codes specified by ZigBee Alliance are conveyed by the errorCode field of ZDPException objects.

149.25 APS Exception

The APSException extends the ZigBeeException. It holds information about the ZigBee APS layer. Error codes specified by ZigBee Alliance are conveyed by the errorCode field of APSException objects.

149.26 ZigBee Exception

Some error codes are specified by the OSGi Alliance:

- OSGI_EXISTING_ID– another endpoint exists with the same ID.
- OSGI_MULTIPLE_HOSTS– several hosts exist for this PAN ID target or HOST_PID target.

149.27 ZCL Frame

The ZCLFrame contains a ZCLHeader, and a payload. It must used when invoking a command.

The ZCLHeader describes the header of a ZCLFrame.

The transaction id of each ZCLHeader must be managed by the base driver.

Only getters (not setters) are shared by client applications, the base driver and endpoint implementations. Therefore only getters are specified.

149.28 ZigBee Group

ZigBeeGroup enables group management (that is, it provides joinGroup(String) and leaveGroup(String) methods).

The creation of groups is made through the createGroupService(int) method.

A ZigBeeGroup service should be registered with the following property:

- ID – (zigbee.group.id/Integer) The 16-bit group address of the device.

And, the following ZigBeeEndpoint properties:

- DEVICE_CATEGORY

- INPUT_CLUSTERS
- HOST_PID

A ZigBeeGroup service enables the ZigBee groupcasting of command invocation thanks to the groupcast(int,ZCLFrame) and groupcast(int,ZCLFrame,String) methods. A groupcast message is received by the endpoints that are members of the targeted group.

149.29 ZigBee Networking

149.29.1 Logical node type

The ZigBee specification defines three types of ZigBee nodes on the network:

- ZigBee Coordinator (ZC) – The most capable device, the coordinator forms the root of the network. There is exactly one ZigBee coordinator in every network. It is able to store information about the network, to act as the Trust Center and repository for security keys. COORDINATOR represents the ZigBee coordinator.
- ZigBee Router (ZR) – A router is capable of extending a ZigBee network by routing data from other ZigBee devices. ROUTER represents a ZigBee router.
- ZigBee End Device (ZED) – An end device contains just enough functionality to talk to the parent node (either the coordinator or a router); it cannot relay data from other devices. ZED represents a ZigBee end device.

Every discovered ZigBeeNode on the network has a logical node type returned by calling the getLogicalType() method on the node's ZigBeeNodeDescriptor.

149.29.2 Network selection

The base driver provides a ZigBeeHost object for every available ZigBee local host. A ZigBee local host can represent a ZigBee chip on a USB dongle, a ZigBee built-in chip or a ZigBee Gateway Device (see [7] *ZigBee Gateway*). This object must be registered as a ZigBeeHost service. The ZigBeeHost interface has methods to start and stop the host, to store the networking configuration information (channel, channel mask, logical type, PAN ID, Extended PAN ID, security level, network key), and to open the network for devices to join it (permitJoin(short)).

ZigBeeHost also enables the broadcast of ZCL commands on a ZigBee network thanks to the broadcast(int,ZCLFrame) and broadcast(int,ZCLFrame,String) methods. Broadcasting enables the sending of a ZCL command to all clusters identified with an identifier of all endpoints available on the nodes of a ZigBee network within a number of hops defined by the broadcast radius of the coordinator (see the getBroadcastRadius() and setBroadcastRadius(short) methods).

In ZigBee networks, the coordinator must select a PAN identifier and a channel to start a network. After that, it behaves essentially like a router. The coordinator and routers can allow other devices to join the network and route data.

After an end device joins a router or coordinator, it is able to transmit or receive data through that router or coordinator. The router or coordinator that allowed an end device to join becomes the parent of the end device. Since the end device can sleep, the parent must be able to buffer or retain incoming data packets targeting the end device until the end device is able to wake up and receive the data.

149.29.3 Network coordination

When the ZigBeeHost is configured as the network coordinator, the getLogicalType() method on the node's ZigBeeNodeDescriptor MUST return COORDINATOR. The ZigBeeHost object will then be able to use the following operations for managing the network:

- updateNetworkChannel(byte) - Updates the network channel.
- setChannelMask(int) - Sets a new configured channel mask.
- refreshNetwork() – Requests the base driver to launch new discovery requests and refresh devices service registration according to current devices availability. This method is made mandatory since ZigBee specification allows devices not to notify the network when they leave it.

149.29.4 Networking considerations

The Network Address is a 16-bit address that is assigned by the coordinator when a node has joined a network and that must be unique for a given network in order for the node to be identified uniquely. ZigBeeNode provides getNetworkAddress() and getIEEEAddress() which returns device network address and device IEEE MAC address.

149.30 Security

149.30.1 Security management

ZigBee security is based on a 128-bit algorithm built on the security model provided by IEEE 802.15.4. ZigBee specification which defines the Trust Center.

The Trust Center is the device trusted by devices within a network to distribute keys for the purpose of network and end-to-end application configuration management. All members of the network shall recognize exactly one Trust Center, and there shall be exactly one Trust Center in each secure network.

The security of a network of ZigBee devices is based on link keys and a network key. Unicast communication between entities is secured by means of a 128-bit link key shared by two devices, one of those is normally the Trust Center. Broadcast communications are secured by means of a 128-bit network key shared among all devices in the network. The master key is only used as an initial shared secret between two devices when they perform the Key Establishment to generate Link Keys.

Security configuration is provided by the getSecurityLevel() method returning whether the security mode is activated or not on the ZigBee network.

A ZigBeeHost with a COORDINATOR logical node type will acts as a the Trust Center according to the ZigBee specification. It can also be any other device on the network. The Trust Center stores all the shared network keys. The getPreconfiguredLinkKey() method returns the network master key.

149.30.2 Conditional permission

When a bundle registers a ZigBeeEndpoint OSGi service, then the base driver exposes this endpoint on the outside ZigBee network and this endpoint has the ability to communicate with the other network devices. The base driver also provides an equivalent behavior when discovering a ZigBee endpoint from the outside network and exposing it as an OSGi Service in the OSGi Framework service registry. It is therefore recommended that ServicePermission[ZigBeeHost|ZigBeeEndpoint|ZCLEventListener, REGISTER|GET] be used sparingly and only for trusted bundles.

149.31 org.osgi.service.zigbee

Device Service Specification for ZigBee Technology.

This is the main package of this specification. It defines the interfaces that models the ZigBee concepts, like the ZigBee node and the ZigBee endpoint.

Each time a new ZigBee node is discovered, the driver will register a org.osgi.service.zigbee.ZigBeeNode service and then one org.osgi.service.zigbee.ZigBeeEndpoint service for each ZigBee endpoint discovered on the node.

org.osgi.service.zigbee.ZigBeeEndpoint interface provides the org.osgi.service.zigbee.ZigBeeEndpoint.getServerCluster(int) method to get an interface reference to a ZCLCluster object.

org.osgi.service.zigbee.ZCLCluster interface contains methods that directly maps to the ZCL profile-wide commands, like Read Attributes and Write Attributes, and allow the developer to forge its own commands and send them through the invoke() methods.

ZCL Attribute reportings are configured, registering a org.osgi.service.zigbee.ZCLEventListener, provided that this service is registered with the right service properties.

In addition to ZCL frames, the current specification allows also to send ZDP frames. Broadcasting and endpoint broadcasting is also supported for ZCL frames.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. This package has two types of users: the consumers that use the API in this package and the providers that implement the API in this package.

Example import for consumers using the API in this package:

```
Import-Package: org.osgi.service.zigbee; version="[1.0,2.0)"
```

Example import for providers implementing the API in this package:

```
Import-Package: org.osgi.service.zigbee; version="[1.0,1.1)"
```

149.31.1 Summary

- `APSException` - This exception class is specialized for the APS errors.
- `ZCLAttribute` - This interface represents a ZCLAttribute.
- `ZCLAttributeInfo` - This interface provides information about the attribute, like its ZCL attribute ID, if it manufacturer specific and about its data type (see getDataType).
- `ZCLCluster` - This interface represents a ZCL Cluster.
- `ZCLCommandResponse` - A response event for a ZCLCommandResponseStream.
- `ZCLCommandResponseStream` - This type represents a stream of responses to a broadcast operation.
- `ZCLEventListener` - This interface represents a listener to events from ZigBee Device nodes.
- `ZCLException` - This class represents root exception for all the code related to ZigBee/ZCL.
- `ZCLFrame` - This interface models the ZigBee Cluster Library Frame.
- `ZCLHeader` - This interface represents the ZCL Frame Header.
- `ZCLReadStatusRecord` - This interface the reading result of ZCLCluster.readAttributes(ZCLAttributeInfo[]).
- `ZDPException` - This class represents root exception for all the code related to ZDP.
- `ZDPFrame` - This interface represents a ZDP frame.
- `ZDPResponse` - This type represents a successful ZDP invocation.
- `ZigBeeDataInput` - The ZigBeeDataInput interface is designed for converting a series of bytes in Java data types.
- `ZigBeeDataOutput` - The ZigBeeDataOutput interface is designed for converting Java data types into a series of bytes.
- `ZigBeeDataTypes` - This class contains the constants that are used internally by these API to represent the ZCL data types.
- `ZigBeeEndpoint` - This interface represents a ZigBee EndPoint.
- `ZigBeeEvent` - This interface represents events generated by a ZigBee Device node.

- `ZigBeeException` - This class represents root exception for all the code related to ZigBee.
- `ZigBeeGroup` - This interface represents a ZigBee Group.
- `ZigBeeHost` - This interface represents the machine that hosts the code to run a ZigBee device or client.
- `ZigBeeLinkQuality` - This interface represents an entry of the NeighborTableList.
- `ZigBeeNode` - This interface represents a ZigBee node, means a physical device that can communicate using the ZigBee protocol.
- `ZigBeeRoute` - This interface represents an entry of the RoutingTableList

149.31.2 public class APSException extends ZigBeeException

This exception class is specialized for the APS errors. See "Table 2.26 APS Sub-layer Status Values" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf.

149.31.2.1 public static final int ASDU_TOO_LONG = 65

A transmit request failed since the ASDU is too large and fragmentation is not supported.

149.31.2.2 public static final int DEFRAG_DEFERRED = 66

A received fragmented frame could not be defragmented at the current time.

149.31.2.3 public static final int DEFRAG_UNSUPPORTED = 67

A received fragmented frame could not be defragmented since the device does not support fragmentation.

149.31.2.4 public static final int ILLEGAL_REQUEST = 68

A parameter value was out of range.

149.31.2.5 public static final int INVALID_BINDING = 69

An APSME-UNBIND.request failed due to the requested binding link not existing in the binding table.

149.31.2.6 public static final int INVALID_GROUP = 70

An APSME-REMOVE-GROUP.request has been issued with a group identifier that does not appear in the group table.

149.31.2.7 public static final int INVALID_PARAMETER = 71

A parameter value was invalid or out of range.

149.31.2.8 public static final int NO_ACK = 72

An APSDE-DATA.request requesting acknowledged transmission failed due to no acknowledgment being received.

149.31.2.9 public static final int NO_BOUND_DEVICE = 73

An APSDE-DATA.request with a destination addressing mode set to 0x00 failed due to there being no devices bound to this device.

149.31.2.10 public static final int NO_SHORT_ADDRESS = 74

An APSDE-DATA.request with a destination addressing mode set to 0x03 failed due to no corresponding short address found in the address map table.

149.31.2.11 **public static final int NOT_SUPPORTED = 75**

An APSDE-DATA.request with a destination addressing mode set to 0x00 failed due to a binding table not being supported on the device.

149.31.2.12 **public static final int SECURED_LINK_KEY = 76**

An ASDU was received that was secured using a link key.

149.31.2.13 **public static final int SECURED_NWK_KEY = 77**

An ASDU was received that was secured using a network key.

149.31.2.14 **public static final int SECURITY_FAIL = 78**

An APSDE-DATA.request requesting security has resulted in an error during the corresponding security processing.

149.31.2.15 **public static final int SUCCESS = 0**

A request has been executed successfully.

149.31.2.16 **public static final int TABLE_FULL = 79**

An APSME-BIND.request or APSME.ADDGROUP. request issued when the binding or group tables, respectively, were full.

149.31.2.17 **public static final int UNSECURED = 80**

An ASDU was received without any security.

149.31.2.18 **public static final int UNSUPPORTED_ATTRIBUTE = 81**

An APSME-GET.request or APSMESET. request has been issued with an unknown attribute identifier.

149.31.2.19 **public APSException(String errorDesc)**

errorDesc exception an error description.

□ Creates a APSException containing only a description, but no error codes. If issued on this exception the getErrorCode() and getZigBeeErrorCode() methods return the UNKNOWN_ERROR constant.

149.31.2.20 **public APSException(int errorCode,String errorDesc)**

errorCode One of the error codes defined in this interface or UNKNOWN_ERROR if the actual error is not listed in this interface. In this case if the native ZigBee error code is known, it is preferred to use the APSException(int, int, String) constructor, passing UNKNOWN_ERROR as first parameter and the native ZigBee error as the second.

errorDesc An error description which explain the type of problem.

□ Creates a APSException containing a specific `errorCode`. Using this constructor with `errorCode` set to UNKNOWN_ERROR is equivalent to call APSException(String).

149.31.2.21 **public APSException(int errorCode,int zigBeeErrorCode,String errorDesc)**

errorCode One of the error codes defined in this interface or UNKNOWN_ERROR the actual error is not covered in this interface. In this case the `zigBeeErrorCode` parameter must be the actual status code returned by the ZigBee stack.

zigBeeErrorCode The actual APS status code or UNKNOWN_ERROR if this status is unknown.

errorDesc An error description which explain the type of problem.

□ Creates a APSException containing a specific `errorCode` or `zigBeeErrorCode`. Using this constructor with both the `errorCode` and `zigBeeErrorCode` set to UNKNOWN_ERROR is equivalent to call APSException(String).

149.31.3 public interface ZCLAttribute extends ZCLAttributeInfo

This interface represents a ZCLAttribute.

Its extends ZCLAttributeInfo to add methods to read and write the ZCL attribute from and to the ZigBee node with respectively the getValue() and setValue(Object) methods.

149.31.3.1 public static final String ID = "zigbee.attribute.id"

Property key for the optional attribute id of a ZigBee Event Listener.

149.31.3.2 public Promise getValue()

□ Gets the current value of the attribute.

As described in section *2.4.1.3 Effect on Receipt* of the ZCL specification, a *Read attributes* command can have the following status: ZCLException.SUCCESS, ZCLException.UNSUPPORTED_ATTRIBUTE, or ZCLException.INVALID_VALUE.

Returns A promise representing the completion of this asynchronous call. The response object returned by Promise.getValue() is the requested attribute value in the relevant Java data type (see getDataType() method and ZCLDataTypeDescription.getJavaDataType()) or in `byte[]` if getDataType() returns null. The response object is null if an ZCLException.UNSUPPORTED_ATTRIBUTE or ZCLException.INVALID_VALUE error occurs and the adequate ZCLException is returned by Promise.getFailure() .

149.31.3.3 public Promise setValue(Object value)

value the Java value to set.

□ Sets the current value of the attribute.

As described in section *2.4.3.3 Effect on Receipt* of the ZCL specification, a *Write attributes* command may return the following status: ZCLException.SUCCESS, ZCLException.UNSUPPORTED_ATTRIBUTE, ZCLException.INVALID_DATA_TYPE, ZCLException.READ_ONLY, ZCLException.INVALID_VALUE, or ZDPException.NOT_AUTHORIZED.

Returns A promise representing the completion of this asynchronous call. Promise.getFailure() returns null if the attribute value has been successfully written. The adequate ZigBeeException is returned otherwise.

149.31.4 public interface ZCLAttributeInfo

This interface provides information about the attribute, like its ZCL attribute ID, if it manufacturer specific and about its data type (see getDataType).

149.31.4.1 public static final String ID = "zigbee.attribute.id"

Property key for the optional attribute id of a ZigBee Event Listener.

149.31.4.2 public ZCLDataTypeDescription getDataType()

□ Returns the data type of this attribute.

Returns The attribute data type. It may be null if the data type is not retrievable (issue with read attribute and discover attributes commands).

149.31.4.3 public int getId()

- Returns the ID of this attribute.

Returns the attribute identifier (that is, the attribute's ID).

149.31.4.4 public int getManufacturerCode()

- Returns the manufacturer code of this attribute.

Returns The manufacturer code that defined this attribute, if the attribute does not belong to any manufacture extension then it returns -1.

149.31.4.5 public boolean isManufacturerSpecific()

- Checks if the attribute is manufacturer specific.

Returns `true` if and only if this attribute is related to a manufacturer extension.

149.31.5 public interface ZCLCluster

This interface represents a ZCL Cluster. Along with methods to retrieve the cluster information, like its ID, it provides methods to asynchronously send commands to the cluster and other methods that wrap most of the ZCL general commands.

Every asynchronous method defined in this interface returns back its result through the use of a Promise.

149.31.5.1 public static final String DOMAIN = "zigbee.cluster.domain"

Property key for the optional cluster domain. A ZigBee Event Listener service can announce for what ZigBee clusters domains it wants notifications.

149.31.5.2 public static final String ID = "zigbee.cluster.id"

Property key for the optional cluster id. A ZigBee Event Listener service can announce for what ZigBee clusters it wants notifications.

149.31.5.3 public static final String NAME = "zigbee.cluster.name"

Property key for the optional cluster name. A ZigBee Event Listener service can announce for what ZigBee clusters it wants notifications.

149.31.5.4 public Promise getAttribute(int attributeId)

attributeId the ZCL attribute identifier.

- Returns the cluster ZCLAttribute identifying that matches the given attributeId. ZCLCluster.getAttribute(int, int) method retrieves manufacturer-specific attributes.

Returns A promise representing the completion of this asynchronous call. In case of success in getting the attribute, the promise will be resolved with a ZCLAttribute instance. If attributeId do not exist in the cluster, then the promise fails with a ZCLException with status code ZCLException.UNSUPPORTED_ATTRIBUTE.

149.31.5.5 public Promise getAttribute(int attributeId,int code)

attributeId the ZCL attribute identifier

code the manufacturer code of the attribute to be retrieved. If -1 is used, the method behaves exactly like ZCLCluster.getAttribute(int)

- Retrieves a ZCLAttribute object for a manufacturer specific attribute. If the `code` parameter is -1 it behaves like the ZCLCluster.getAttribute(int) and retrieves the non-manufacturer specific attribute `attributeId`.

Returns A Promise representing the completion of this asynchronous call. The promise will be resolved with the requested ZCLAttribute. If a command such as ZCL Read Attributes or Discover Attributes has already been called once by the ZigBee host, the Promise can be quickly resolved. The resolution may be longer the first time one of the ZCLCluster methods to get one or all attributes is successfully called. If attributeId do not exist in the cluster, then the promise fails with a ZCLException with status code ZCLException.UNSUPPORTED_ATTRIBUTE

149.31.5.6 public Promise getAttributes()

□ Returns an array of ZCLAttribute objects representing all this cluster's attributes.

This method returns only standard attributes. To retrieve manufacturer specific attributes use method ZCLCluster.getAttributes(int)

Returns A Promise representing the completion of this asynchronous call. The promise will be resolved with an array of ZCLAttribute objects.

149.31.5.7 public Promise getAttributes(int code)

code The the manufacturer code. Pass -1 to retrieve standard (that is, non-manufacturer specific) attributes.

□ Returns an array of ZCLAttribute objects representing all the specific manufacturer attributes available on the cluster.

This method behaves like the ZCLCluster.getAttributes() method if the passed value is -1.

Returns A Promise representing the completion of this asynchronous call. The promise will be resolved with an array of ZCLAttribute objects. If a command such as ZCL Read Attributes or Discover Attributes has already been called once by the ZigBee host, the Promise can be quickly resolved. The resolution may be longer the first time one of the ZCLCluster methods to get one or all attributes is successfully called.

149.31.5.8 public Promise getCommandIds()

□ Returns an array of all the commandIds of the ZCLCluster.

This method is implemented for ZCL devices compliant version equal or later than 1.2 of the Home Automation Profile or other profiles that adds a general command that enables discovery of command identifiers. When the device implements a profile that does not support this feature, the promise fails with a ZCLException with code ZCLException.GENERAL_COMMAND_NOT_SUPPORTED.

Returns A Promise representing the completion of this asynchronous call. The promise will be resolved with `short[]` containing the command identifiers supported by the cluster.

149.31.5.9 public int getId()

□ Returns the identifier of this cluster.

Returns the cluster identifier.

149.31.5.10 public Promise invoke(ZCLFrame frame)

frame The frame containing the command to issue.

□ Invokes a command on this cluster with a ZCLFrame. The returned promise provides the invocation response in an asynchronous way. The source endpoint is not specified in this method call. To send the appropriate message on the network, the base driver must generate a source endpoint. The latter must not correspond to any exported endpoint.

Returns A promise representing the completion of this asynchronous call. Promise.getValue() returns the response ZCLFrame.

149.31.5.11 public Promise invoke(ZCLFrame frame,String exportedServicePID)

frame The frame containing the command to issue.

exportedServicePID : the source endpoint of the command request. In targeted situations, the source endpoint is the valid service PID of an exported endpoint.

□ Invokes a command on this cluster. This method is to be used by applications when the targeted device has to distinguish between source endpoints of the message. For instance, alarms cluster (see 3.11 Alarms Cluster in [ZCL]) generated events are differently interpreted if they come from the oven or from the intrusion alert system.

Returns A promise representing the completion of this asynchronous call. Promise.getValue() returns the response ZCLFrame.

149.31.5.12 public Promise readAttributes(ZCLAttributeInfo[] attributes)

attributes An array of ZCLAttributeInfo.

□ Reads a list of attributes by issuing a ZCL Read Attributes command. The attribute list is provided in terms of an array of ZCLAttributeInfo objects.

As described in section *2.4.1.3 Effect on Receipt* of the ZCL specification, a *Read Attributes* command results in a list of attribute status records comprising a mix of successful and unsuccessful attribute reads.

The method returns a promise. The object used to resolve the Promise is a `Map<Integer, ZCLReadStatusRecord>`. For each Map entry, the key contains the attribute identifier and the value, a ZigBee Read Attributes Status Record, which is made of the status of the read of this attribute, the ZigBee data type of the attribute and the attribute value in the corresponding Java wrapper type (or null in case of an unsupported attribute or in case of an invalid value). For attributes which data type serialization is not supported (that is, ZCLDataTypeDescription.getJavaDataType() returns null), the value is of type byte[].

When the list of attributes do not fit into a single ZCLFrame, ZigBee clusters truncate the list of attributes returned in the response. The client has to check the Map of results to send a new request for the attributes which values are missing. In export situations, the base driver may truncate the read attribute command response sent to networked devices in order to obey the rules.

NOTE: According to the ZigBee Specification all the attributes must be standard attributes or belong to the same manufacturer code, otherwise the promise must fail with a IllegalArgumentException exception.

Returns A promise representing the completion of this asynchronous call. The promise may fail with an IllegalArgumentException if the array size is 0 or if one of the array entries is `null` or not valid. An IllegalArgumentException is also thrown if some of ZCLAttributeInfo are manufacturer specific and other are standard, or even if there are mix of attributes with different manufacturer specific code. If the passed argument is `null` the promise must fail with a `NullPointerException`.

149.31.5.13 public Promise writeAttributes(boolean undivided,Map attributesAndValues)

undivided `true` if an undivided write attributes command is requested, `false` if not.

attributesAndValues A `Map<ZCLAttributeInfo, Object>` of attributes and values to be written. For ZCLAttributeInfo objects which serialization is not supported (that is, `getDataType().getJavaDataType()` returns null), the value must be of type byte[].

□ Writes a set of attributes on the cluster using the ZCL *Write Attributes* or the *Write Attributes Undivided* commands, according to the passed `undivided` parameter.

The promise resolves with a `Map<Integer, Integer>`. If all the attributes have been written successfully, the map is empty. In case of failure in writing specific attribute(s), the map is filled with entries related to those attributes. Every key is set with the id of an attribute that was not written successfully, every value with the status returned in the associated *write attribute response record* accordingly re-mapped to one of the constants defined in the ZCLException class.

According to the ZigBee Specification all the attributes must be standard attributes or, if manufacturer-specific they must have the same manufacturer code, otherwise an IllegalArgumentException occurs.

Returns A promise representing the completion of this asynchronous call. If resolved successfully the promise may return an empty `Map<Integer, Integer>`. Otherwise the map will be filled with the status information about the attributes that were not written. The key represents the attributeID and the value the status present in the corresponding attribute record returned by the ZCL Write Attributes response message. The original ZCL status values must be re-mapped to the list of status values listed in the ZCLException class. The promise may fail with an IllegalArgumentException if some of ZCLAttributeInfo are manufacturer specific and other are standard, or even if there are mix of attributes with different manufacturer specific code.

149.31.6 public interface ZCLCommandResponse

A response event for a ZCLCommandResponseStream.

149.31.6.1 public Promise getResponse()

□ Returns a promise holding the response.

Returns A Promise holding the ZCLFrame response, or a failure exception if this is not a success response.

149.31.6.2 public boolean isEnd()

□ Checks if this is a terminal close event.

Returns `true` if this is a terminal close event.

149.31.7 public interface ZCLCommandResponseStream

This type represents a stream of responses to a broadcast operation. It can be closed by the client using the close method is called. The ZCLCommandResponseStream is used to process a stream of responses from a ZigBee network. Responses are consumed by registering a handler with forEach(Predicate). Responses received before a handler is registered are buffered until a handler is registered, or until the close method is called. A handler consumes events returning `true` to continue delivery. At some point the ZigBee service invocation will terminate event delivery by sending a close event (a ZCLCommandResponse which returns `true` from ZCLCommandResponse.isEnd(). After a close event the handler function will be dereferenced.

149.31.7.1 public void close()

□ Closes this response, indicating that no further responses are needed. Any buffered responses will be discarded, and a close event will be sent to a handler if it is registered.

149.31.7.2 public void forEach(Predicate handler)

handler A handler to process ZCLCommandResponse objects

□ Registers a handler that will be called for each of the received responses. Only one handler may be registered. Any responses that arrive before a handler is registered will be buffered and pushed into the handler when it is registered. If the handler returns `false` from its accept method then the handler will be released and no further events will be delivered. Any remaining buffered events will be discarded, and this object marked as closed. If the handler does not close the stream early then the ZigBee service implementation will eventually send a close event.

Throws `IllegalStateException`– if a handler has already been registered, or if this object has been closed (a ZCLCommandResponse which returns `true` from ZCLCommandResponse.isEnd(). After a close event the handler function will be dereferenced.

149.31.8 public interface ZCLEventListener

This interface represents a listener to events from ZigBee Device nodes.

149.31.8.1 public static final String ATTRIBUTE_DATA_TYPE = "zigbee.attribute.datatype"

Property key for the optional attribute data type of an attribute reporting configuration record, cf. ZCL Figure 2.16 Format of the Attribute Reporting Configuration Record.

149.31.8.2 public static final String MAX_REPORT_INTERVAL = "zigbee.attribute.max.report.interval"

Property key for the optional maximum interval, in seconds between issuing reports of the attribute. A ZigBee Event Listener service can declare the maximum frequency at which events it wants notifications.

149.31.8.3 public static final String MIN_REPORT_INTERVAL = "zigbee.attribute.min.report.interval"

Property key for the optional minimum interval, in seconds between issuing reports of the attribute. A ZigBee Event Listener service can declare the minimum frequency at which events it wants notifications.

149.31.8.4 public static final String REPORTABLE_CHANGE = "zigbee.attribute.reportable.change"

Property key for the optional maximum change to the attribute that will result in a report being issued. A ZigBee Event Listener service can declare the maximum frequency at which events it wants notifications.

149.31.8.5 public void notifyEvent(ZigBeeEvent event)

event a set of events.

□ Notifies the reception of an event. This method is called asynchronously.

149.31.8.6 public void notifyTimeOut(int timeout)

timeout the timeout in seconds.

□ Notifies that the timeout is elapsed. No event will be received in the interval.

149.31.8.7 public void onFailure(ZCLException e)

e the ZCLException.

□ Notifies that a failure has occurred.

That is, when either a ZCLException.UNSUPPORTED_ATTRIBUTE, ZCLException.UNREPORTABLE_TYPE, ZCLException.INVALID_VALUE, or ZCLException.INVALID_DATA_TYPE status occurs.

149.31.9 public class ZCLException extends ZigBeeException

This class represents root exception for all the code related to ZigBee/ZCL. The provided constants names, but not the values, maps to the ZCL error codes defined in the ZCL specification.

149.31.9.1 public static final int CALIBRATION_ERROR = 18

ZCL Calibration Error error code.

149.31.9.2 public static final int CLUSTER_COMMAND_NOT_SUPPORTED = 3

ZCL Cluster Command Not Supported error code.

149.31.9.3 public static final int DUPLICATE_EXISTS = 12

ZCL Duplicate Exists error code.

149.31.9.4 **public static final int FAILURE = 1**

ZCL Failure error code.

149.31.9.5 **public static final int GENERAL_COMMAND_NOT_SUPPORTED = 4**

ZCL General Command Not Supported error code.

149.31.9.6 **public static final int HARDWARE_FAILURE = 16**

HARDWARE_FAILURE - in this case, an additional exception describing the problem can be nested.

149.31.9.7 **public static final int INSUFFICIENT_SPACE = 11**

ZCL Insufficient Space error code.

149.31.9.8 **public static final int INVALID_DATA_TYPE = 15**

ZCL Invalid Data Type error code.

149.31.9.9 **public static final int INVALID_FIELD = 7**

ZCL Invalid Field error code.

149.31.9.10 **public static final int INVALID_VALUE = 9**

ZCL Invalid Value error code.

149.31.9.11 **public static final int MALFORMED_COMMAND = 2**

ZCL Malformed Command error code.

149.31.9.12 **public static final int MANUF_CLUSTER_COMMAND_NOT_SUPPORTED = 5**

ZCL Manuf Cluster Command Not Supported error code.

149.31.9.13 **public static final int MANUF_GENERAL_COMMAND_NOT_SUPPORTED = 6**

ZCL Manuf General Command Not Supported error code.

149.31.9.14 **public static final int NOT_FOUND = 13**

ZCL Not Found error code.

149.31.9.15 **public static final int READ_ONLY = 10**

ZCL Read Only error code.

149.31.9.16 **public static final int SOFTWARE_FAILURE = 17**

Software Failure error code - in this case, an additional exception describing the problem can be nested.

149.31.9.17 **public static final int SUCCESS = 0**

ZCL Success error code.

149.31.9.18 **public static final int UNREPORTABLE_TYPE = 14**

Unreportable Type error code.

149.31.9.19 **public static final int UNSUPPORTED_ATTRIBUTE = 8**

ZCL Unsupported Attribute error code.

149.31.9.20 **public ZCLException(String errorDesc)**

errorDesc exception error description.

□ Creates a ZCLException containing only a description, but no error codes. If issued on this exception the getErrorCode() and getZigBeeErrorCode() methods return the UNKNOWN_ERROR constant.

149.31.9.21 public ZCLException(int errorCode,String errorDesc)

errorCode One of the error codes defined in this interface or UNKNOWN_ERROR if the actual error is not listed in this interface. In this case if the native ZigBee error code is known, it is preferred to use the ZCLException(int, int, String) constructor, passing UNKNOWN_ERROR as first parameter and the native ZigBee error as the second.

errorDesc An error description which explain the type of problem.

□ Creates a ZCLException containing a specific `errorCode`. Using this constructor with `errorCode` set to UNKNOWN_ERROR is equivalent to call ZCLException(String).

149.31.9.22 public ZCLException(int errorCode,int zigBeeErrorCode,String errorDesc)

errorCode One of the error codes defined in this interface or UNKNOWN_ERROR the actual error is not covered in this interface. In this case the `zigBeeErrorCode` parameter must be the actual status code returned by the ZigBee stack.

zigBeeErrorCode The actual ZCL status code or UNKNOWN_ERROR if this status is unknown.

errorDesc An error description which explain the type of problem.

□ Creates a ZCLException containing a specific `errorCode` or `zigBeeErrorCode`. Using this constructor with both the `errorCode` and `zigBeeErrorCode` set to UNKNOWN_ERROR is equivalent to call ZCLException(String).

149.31.10 public interface ZCLFrame

This interface models the ZigBee Cluster Library Frame.

149.31.10.1 public byte[] getBytes()

□ Returns a byte array containing the raw ZCL frame, suitable to be sent on the wire. The returned byte array contains the whole ZCL Frame, including the ZCL Frame Header and the ZCL Frame payload.

Returns a byte array containing a raw ZCL frame, suitable to be sent on the wire. Any modifications issued on the returned array must not affect the internal representation of the ZCLFrame interface implementation.

149.31.10.2 public int getBytes(byte[] buffer)

buffer The buffer where to copy the raw ZCL frame.

□ Copy in the passed array the internal raw ZCLFrame.

Returns The actual number of bytes copied.

149.31.10.3 public ZigBeeDataInput getDataInput()

□ Returns ZigBeeDataInput for reading the ZCLFrame payload content. Every call to this method returns a different instance. The returned instances must not share the current position to the underlying ZCLFrame payload.

Returns a DataInput for the payload of the ZCLFrame. This method does not generate a copy of the payload.

Throws `IllegalStateException`– if the InputStream is not available.

149.31.10.4 public ZCLHeader getHeader()

□ Returns the header of this frame.

Returns the header of this frame.

149.31.10.5 public int getSize()

□ Retrieve the current size of the internal raw frame (that is the size of the byte[] that would be returned if calling the getBytes() method.

Returns The size of the raw ZCL frame.

149.31.11 public interface ZCLHeader

This interface represents the ZCL Frame Header.

149.31.11.1 public short getCommandId()

□ Returns the command identifier of this frame.

Returns the command identifier of this frame.

149.31.11.2 public short getFrameControlField()

□ Returns the Frame Control field of this frame.

Returns the frame control field of this frame.

149.31.11.3 public int getManufacturerCode()

□ Returns the manufacturer code of this frame.

Returns the manufacturer code if the ZCL Frame is manufacturer specific, otherwise returns -1.

149.31.11.4 public byte getSequenceNumber()

□ Returns the transaction Sequence Number of this frame.

Returns the transaction sequence number of this frame.

149.31.11.5 public boolean isClientServerDirection()

□ Checks the client server direction of the frame.

Returns the isClientServerDirection value.

149.31.11.6 public boolean isClusterSpecificCommand()

□ Checks the frame Type Sub-field of the frame control field.

Returns true if the frame control field states that the command is cluster specific. Returns false otherwise.

149.31.11.7 public boolean isDefaultResponseDisabled()

□ Checks if the default response is disabled.

Returns `true` if the ZCL Header Frame Control Field "Disable Default Response Sub-field" is 1. Returns `false` otherwise.

149.31.11.8 public boolean isManufacturerSpecific()

□ Checks if the frame is manufacturer specific.

Returns true if the ZCL frame is manufacturer specific (that is, the Manufacturer Specific Sub-field of the ZCL Frame Control Field is 1.

149.31.12 public interface ZCLReadStatusRecord

This interface the reading result of ZCLCluster.readAttributes(ZCLAttributeInfo[]).

149.31.12.1 public ZCLAttributeInfo getAttributeInfo()

□ Returns the ZCLAttributeInfo related to the reading operation.

Returns the ZCLAttributeInfo related to the reading operation.

149.31.12.2 **public ZigBeeException getFailure()**

□ Returns the potential failure of the reading operation.

Returns null in case of success, otherwise the ZigBeeException specifying the failing of the reading.

149.31.12.3 **public Object getValue()**

□ Returns the value of the related read attribute.

Returns null in case of failure or invalid data, otherwise the Java Object representing the ZigBee value.

149.31.13 public class ZDPException extends ZigBeeException

This class represents root exception for all the code related to ZDP.

See Table 2.137 ZDP Enumerations Description in ZIGBEE SPECIFICATION: 1_053474r17ZB_TSC-ZigBee-Specification.pdf.

149.31.13.1 **public static final int DEVICE_NOT_FOUND = 34**

The requested device did not exist on a device following a child descriptor request to a parent.

149.31.13.2 **public static final int INSUFFICIENT_SPACE = 42**

The device does not have storage space to support the requested operation.

149.31.13.3 **public static final int INV_REQUESTTYPE = 33**

The supplied request type was invalid.

149.31.13.4 **public static final int INVALID_EP = 35**

The supplied endpoint was equal to 0x00 or between 0xf1 and 0xff.

149.31.13.5 **public static final int NO_DESCRIPTOR = 41**

A child descriptor was not available following a discovery request to a parent.

149.31.13.6 **public static final int NO_ENTRY = 40**

The unbind request was unsuccessful due to the coordinator or source device not having an entry in its binding table to unbind.

149.31.13.7 **public static final int NO_MATCH = 39**

The end device bind request was unsuccessful due to a failure to match any suitable clusters.

149.31.13.8 **public static final int NOT_ACTIVE = 36**

The requested endpoint is not described by a simple descriptor.

149.31.13.9 **public static final int NOT_AUTHORIZED = 45**

The permissions configuration table on the target indicates that the request is not authorized from this device.

149.31.13.10 **public static final int NOT_PERMITTED = 43**

The device is not in the proper state to support the requested operation.

149.31.13.11 **public static final int NOT_SUPPORTED = 37**

The requested optional feature is not supported on the target device.

149.31.13.12 **public static final int SUCCESS = 0**

The requested operation or transmission was completed successfully.

149.31.13.13 public static final int TABLE_FULL = 44

The device does not have table space to support the operation.

149.31.13.14 public static final int TIMEOUT = 38

A timeout has occurred with the requested operation.

149.31.13.15 public ZDPException(String errorDesc)

errorDesc exception error description.

□ Creates a ZDPException containing only a description, but no error codes. If issued on this exception the getErrorCode() and getZigBeeErrorCode() methods return the UNKNOWN_ERROR constant.

149.31.13.16 public ZDPException(int errorCode,String errorDesc)

errorCode One of the error codes defined in this interface or UNKNOWN_ERROR if the actual error is not listed in this interface. In this case if the native ZigBee error code is known, it is preferred to use the ZDPException(int, int, String) constructor, passing UNKNOWN_ERROR as first parameter and the native ZigBee error as the second.

errorDesc An error description which explain the type of problem.

□ Creates a ZDPException containing a specific errorCode. Using this constructor with errorCode set to UNKNOWN_ERROR is equivalent to call ZDPException(String).

149.31.13.17 public ZDPException(int errorCode,int zigBeeErrorCode,String errorDesc)

errorCode One of the error codes defined in this interface or UNKNOWN_ERROR the actual error is not covered in this interface. In this case the zigBeeErrorCode parameter must be the actual status code returned by the ZigBee stack.

zigBeeErrorCode The actual ZDP status code or UNKNOWN_ERROR if this status is unknown.

errorDesc An error description which explain the type of problem.

□ Creates a ZDPException containing a specific errorCode or zigBeeErrorCode. Using this constructor with both the errorCode and zigBeeErrorCode set to UNKNOWN_ERROR is equivalent to call ZDPException(String).

149.31.14 public interface ZDPFrame

This interface represents a ZDP frame.

See Figure 2.19 Format of the ZDP Frame ZIGBEE SPECIFICATION: 1_053474r17ZB_TSC-ZigBee-Specification.pdf.

This interface MUST be implemented by the developer invoking the ZigBeeNode.invoke(int, int, ZDPFrame) method.

Notes:

- This interface hides on purpose the Transaction Sequence Number field because it MUST be handled internally by the ZigBee Base Driver
- The interface does not provide any method for writing the payload because the ZigBee Base Driver needs only to read the payload.

149.31.14.1 public ZigBeeDataInput getDataInput()

□ Returns the ZigBeeDataInput of the payload of this frame.

Returns the ZigBeeDataInput of the payload of the ZDPFrame. This method, in contrary to getPayload(), doesn't require to create a copy of the payload.

Throws IllegalStateException– if a ZigBeeDataInput stream cannot be returned because the underlying ZDPFrame implementation was not correctly initialized.

149.31.14.2 public byte[] getPayload()

□ Returns a copy of the payload of this frame.

Returns A copy of the payload of this frame.

149.31.15 public interface ZDPResponse

This type represents a successful ZDP invocation. Note that the underlying call may not have succeeded, The ZDPFrame frame must be introspected to identify the response from the ZigBeeNode.

149.31.15.1 public int getClusterId()

□ Returns the clusterId this response refers to.

Returns the clusterId this response refers to.

149.31.15.2 public ZDPFrame getFrame()

□ Returns the ZDPFrame containing the response.

Returns the ZDPFrame containing the response.

149.31.16 public interface ZigBeeDataInput

The ZigBeeDataInput interface is designed for converting a series of bytes in Java data types. The purpose of this interface is the same as the DataInput interface available in the standard Java library, with the difference that in this interface, byte ordering is little endian, whereas in the DataInput interface is big endian.

Each method provided by this interface read one or more bytes from the underlying stream, combine them, and return a Java data type. The pointer to the stream is then moved immediately after the last byte read. If this pointer past the available buffer bounds, a subsequent call to one of these methods will throw a EOFException.

149.31.16.1 public byte readByte() throws IOException

□ Reads a byte from the DataInput Stream.

Returns the byte read from the data input.

Throws EOFException– When the end of the input has been reached and there are no more data to read.

IOException– If an I/O error occurs.

149.31.16.2 public byte[] readBytes(int len) throws IOException

len the number of bytes to read.

□ Reads the specified amount of bytes from the underlying stream and return a copy of them. If the number of available bytes is less than the requested len, it throws an EOFException.

Returns return a copy of the bytes contained in the stream.

Throws EOFException– if there are not at least len bytes left on the data input.

IOException– If an I/O error occurs.

149.31.16.3 public double readDouble() throws IOException

□ Reads a number of type Double.

Returns a decoded double.

Throws EOFException– if there are not at least size 8 bytes left on the data input.

IOException– If an I/O error occurs.

149.31.16.4 public float readFloat(int size) throws IOException

size expected value for this parameter are 2 or 4 depending if reading ZigBeeDataTypes.FLOATING_SEMI or ZigBeeDataTypes.FLOATING_SINGLE.

□ Reads a number of type Float.

Returns The float number read from the data input.

Throws EOFException– if there are not at least size bytes left on the data input.

IOException– If an I/O error occurs.

IllegalArgumentException– If the passed size is not in the allowed range.

149.31.16.5 public int readInt(int size) throws IOException

size the number of bytes that have to be read. Allowed values for this parameter are in the range [1, 4].

□ Reads an integer of the specified size. The sign bit of the size-bytes integer is left-extended. In other words if a readInt(2) is issued and the byte read are 0x01, 0x02 and 0xf0, the method returns 0xfff00201. For this reason if the 4 bytes read from the stream represent an unsigned value, to get the expected value the and bitwise operator must be used:

```
int u = readInt(3) & oxffffff;
```

Returns the integer read from the data input.

Throws EOFException– When the end of the input has been reached and there are no more data to read.

IOException– If an I/O error occurs.

IllegalArgumentException– If the passed size is not in the allowed range.

149.31.16.6 public long readLong(int size) throws IOException

size the number of bytes that have to be read. Allowed values for this parameter are in the range [1, 8].

□ Reads a certain amount of bytes and returns a long. The sign bit of the read size-bytes long is left-extended. In other words if a readLong(2) is issued and the byte read are 0x01 and 0xf0, the method returns 0xfffffffffffff001L. For this reason if the 2 bytes read from the stream represent an unsigned value, to get the expected value the and bitwise operator must be used:

```
long u = readLong(2) & oxffff;
```

Returns The long value read from the data input.

Throws EOFException– if there are not at least size bytes left on the data input.

IOException– If an I/O error occurs.

IllegalArgumentException– If the passed size is not in the allowed range.

149.31.17 public interface ZigBeeDataOutput

The ZigBeeDataOutput interface is designed for converting Java data types into a series of bytes. The purpose of this interface is the same as the DataOutput interface provided by Java, with the difference that in this interface, the generated bytes ordering is little endian, whereas in the DataOutput is big endian.

149.31.17.1 public void writeByte(byte value)

value The value to append.

□ Appends a byte to the data output.

To avoid losing information, the passed value must be in the range [-128, 127] for signed numbers and [0, 255] for unsigned numbers.

149.31.17.2 public void writeBytes(byte[] bytes,int length) throws IOException

bytes A buffer containing the bytes to append to the data output stream.

length The length in bytes that have to be actually appended.

□ Appends on the Data Output Stream a byte array. The byte array is written on the data output starting from the byte at index 0.

Throws IOException– If an I/O error occurs.

IllegalArgumentException– If the passed buffer is null or shorter than length bytes.

149.31.17.3 public void writeDouble(double value) throws IOException

value The double value to append.

□ Appends on the Data Output Stream a double value.

Throws IOException– If an I/O error occurs.

149.31.17.4 public void writeFloat(float value,int size) throws IOException

value The float value to append.

size The size in bytes that have to be actually appended. The size must be 2 for semi precision floats or 4 for standard precision floats (see the ZigBee Cluster Library specifications).

□ Appends on the Data Output Stream a float value.

Throws IOException– If an I/O error occurs.

IllegalArgumentException– If the passed size is not within the allowed range.

149.31.17.5 public void writeInt(int value,int size) throws IOException

value The integer value to append

size The size in bytes that have to be actually appended. The size must be in the range [1,4].

□ Appends an int value to the data output.

To avoid losing information, according to the size argument, the passed long value if it represents a signed number must fit in the range [-2^(size * 8 - 1), -2^(size * 8 - 1) - 1].

For unsigned numbers it should fit in the range [0, -2^(size * 8) - 1].

For instance if size is 2 the correct range for signed numbers is [0xffff8000, 0x7fff] (that is, [-32768, +32767]), whereas for unsigned numbers is [0L, 0xffff].

Although this method allows write even 1 byte of the passed int value, it is suggested to use the writeByte(byte) because this latter could be implemented in a more efficient way.

Throws IOException– If an I/O error occurs.

IllegalArgumentException– If the passed size is not within the allowed range.

149.31.17.6 public void writeLong(long value,int size) throws IOException

value The long value to append

size The size in bytes that have to be actually appended. The size must be in the range [1,8].

□ Appends a long to the data output.

To avoid losing information, according to the size argument, the passed long value if it represents a signed number must fit in the range [-2^(size * 8 - 1), -2^(size * 8 - 1) - 1].

For unsigned numbers it should fit in the range [0, -2^(size * 8) - 1].

For instance if size is 3 the correct range for signed numbers is [0xffffffffff800000L, 0x7fffffL] (that is, [-21474836448, +2147483647]), whereas for unsigned numbers is [0L, 0xffffffL].

Although this method allows write even 1 byte of the passed long value, it is suggested to use the writeByte(byte) because this latter could be implemented in a more efficient way.

Throws IOException– If an I/O error occurs.

IllegalArgumentException– If the passed size is not within the allowed range.

149.31.18 public class ZigBeeDataTypes

This class contains the constants that are used internally by these API to represent the ZCL data types.

These constants do not match the values used in the ZigBee specification, but follow the rules below:

- *bit 0-3*: if bit 6 is one, these bits represents the size of the data type in bytes.
- *bit 6*: if set to 1 bits 0-3 represents the size of the data type in bytes.

Related documentation: [1] ZigBee Cluster Library specification, Document 075123r04ZB, May 29, 2012.

149.31.18.1 public static final short ARRAY = 16

According to ZigBee Cluster Library [1], an Array is an ordered sequence of zero or more elements, all of the same data type. This data type may be any ZCL defined data type, including Array, Structure, Bag or Set. The total nesting depth is limited to 15.

149.31.18.2 public static final short ATTRIBUTE_ID = 6

The type of an attribute identifier.

149.31.18.3 public static final short BACNET_OID = 7

According to ZigBee Cluster Library [1], the BACnet OID data type is included to allow interworking with BACnet. The format is described in the referenced standard.

149.31.18.4 public static final short BAG = 19

According to ZigBee Cluster Library [1], a Bag behaves exactly the same as a Set, except that two elements may have the same value.

149.31.18.5 public static final short BITMAP_16 = 89

Bitmap16-bit

149.31.18.6 public static final short BITMAP_24 = 90

Bitmap 24-bit

149.31.18.7 public static final short BITMAP_32 = 91

Bitmap 32-bit

149.31.18.8 public static final short BITMAP_40 = 92

Bitmap 40-bit

149.31.18.9 public static final short BITMAP_48 = 93

Bitmap 48-bit

149.31.18.10 public static final short BITMAP_56 = 94

Bitmap 56-bit

149.31.18.11 public static final short BITMAP_64 = 95

Bitmap 64-bit

149.31.18.12 public static final short BITMAP_8 = 88

According to ZigBee Cluster Library [1], the Bitmap type holds logical values, one per bit, depending on its length. There is no value that represents an invalid value of this type. The Bitmap type is defined with several sizes: 8, 16, 24, 32, 40, 48, 56 and 64 bits.

149.31.18.13 public static final short BOOLEAN = 1

According to ZigBee Cluster Library [1], the Boolean type represents a logical value, either FALSE (0x00) or TRUE (0x01). The value 0xff represents an invalid value of this type. All other values of this type are forbidden.

149.31.18.14 public static final short CHARACTER_STRING = 121

According to ZigBee Cluster Library [1], the Character String data type contains data octets encoding characters according to the language and character set field of the complex descriptor.

149.31.18.15 public static final short CLUSTER_ID = 5

The type of a cluster identifier.

149.31.18.16 public static final short DATE = 3

The Date data type format is specified in section 2.5.2.20 of ZigBee Cluster Specification [1].

149.31.18.17 public static final short ENUMERATION_16 = 113

Enumeration 16-bit

149.31.18.18 public static final short ENUMERATION_8 = 112

According to ZigBee Cluster Library [1], the Enumeration type represents an index into a lookup table to determine the final value. The values 0xff and 0xffff represent invalid values of the 8-bit and 16-bit types respectively.

149.31.18.19 public static final short FLOATING_DOUBLE = 250

According to ZigBee Cluster Library [1], the format of the double precision data type is based on the IEEE 754 standard for binary floating-point arithmetic.

149.31.18.20 public static final short FLOATING_SEMI = 248

According to ZigBee Cluster Library [1], the ZigBee semi-precision number format is based on the IEEE 754 standard for binary floating-point arithmetic.

149.31.18.21 public static final short FLOATING_SINGLE = 249

According to ZigBee Cluster Library [1], the format of the single precision data type is based on the IEEE 754 standard for binary floating-point arithmetic.

149.31.18.22 public static final short GENERAL_DATA_16 = 81

General Data 16-bit

149.31.18.23 public static final short GENERAL_DATA_24 = 82

General Data 24-bit

149.31.18.24 public static final short GENERAL_DATA_32 = 83

General Data 32-bit

149.31.18.25 public static final short GENERAL_DATA_40 = 84

General Data 40-bit

149.31.18.26 public static final short GENERAL_DATA_48 = 85

General Data 48-bit

149.31.18.27 public static final short GENERAL_DATA_56 = 86

General Data 56-bit

149.31.18.28 public static final short GENERAL_DATA_64 = 87

General Data 64-bit

149.31.18.29 public static final short GENERAL_DATA_8 = 80

According to ZigBee Cluster Library [1], the General Data type may be used when a data element is needed but its use does not conform to any of other types. The General Data type is defined with several sizes: 8, 16, 24, 32, 40, 48, 56 and 64 bits.

149.31.18.30 public static final short IEEE_ADDRESS = 8

According to ZigBee Cluster Library [1], the IEEE Address data type is a 64-bit IEEE address that is unique to every ZigBee device. A value of 0xffffffffffffffff indicates that the address is unknown.

149.31.18.31 public static final short LONG_CHARACTER_STRING = 123

According to ZigBee Cluster Library [1], the Long Character String data type contains data octets encoding characters according to the language and character set field of the complex descriptor.

149.31.18.32 public static final short LONG_OCTET_STRING = 122

According to ZigBee Cluster Library [1], the Long Octet String data type contains data in application-defined formats.

149.31.18.33 public static final short NO_DATA = 0

According to ZigBee Cluster Library [1], the no data type represents an attribute with no associated data.

149.31.18.34 public static final short OCTET_STRING = 120

According to ZigBee Cluster Library [1], the Octet String data type contains data in application-defined formats.

149.31.18.35 public static final short SECURITY_KEY_128 = 9

According to ZigBee Cluster Library [1], the 128-bit Security Key data type is for use in ZigBee security, and may take any 128-bit value.

149.31.18.36 public static final short SET = 18

According to ZigBee Cluster Library [1], a Set is a collection of elements with no associated order. Each element has the same data type, which may be any ZCL defined data type, including Array, Structure, Bag or Set. The nesting depth is limited to 15.

149.31.18.37 public static final short SIGNED_INTEGER_16 = 225

Signed Integer 16-bit

149.31.18.38 public static final short SIGNED_INTEGER_24 = 226

Signed Integer 24-bit

149.31.18.39 public static final short SIGNED_INTEGER_32 = 227

Signed Integer 32-bit

149.31.18.40 public static final short SIGNED_INTEGER_40 = 228

Signed Integer 40-bit

149.31.18.41 public static final short SIGNED_INTEGER_48 = 229

Signed Integer 48-bit

149.31.18.42 public static final short SIGNED_INTEGER_56 = 230

Signed Integer 56-bit

149.31.18.43 public static final short SIGNED_INTEGER_64 = 231

Signed Integer 64-bit

149.31.18.44 public static final short SIGNED_INTEGER_8 = 224

According to ZigBee Cluster Library [1], the Signed Integer type represents a signed integer with a decimal range of -(2^7-1) to 2^7-1, - (2^15-1) to 2^15-1, -(2^23-1) to 2^23-1, -(2^31-1) to 2^31-1, -(2^39-1) to 2^39-1, -(2^47-1) to 2^47-1, -(2^55-1) to 2^55-1, or -(2^63-1) to 2^63-1, depending on its length. The values that represents an invalid value of this type are 0x80, 0x8000, 0x800000, 0x80000000, 0x8000000000, 0x800000000000, 0x80000000000000 and 0x8000000000000000 respectively. This type is defined with several sizes: 8, 16, 24, 32, 40, 48, 56 and 64 bits.

149.31.18.45 public static final short STRUCTURE = 17

According to ZigBee Cluster Library [1], a Structure is an ordered sequence of elements, which may be of different data types. Each data type may be any ZCL defined data type, including Array, Structure, Bag or Set. The total nesting depth is limited to 15.

149.31.18.46 public static final short TIME_OF_DAY = 2

The Time of Day data type format is specified in section 2.5.2.19 of ZCL specification [1].

149.31.18.47 public static final short UNKNOWN = 255

The UNKNOWN type is used when the data type is unknown.

149.31.18.48 public static final short UNSIGNED_INTEGER_16 = 97

Unsigned Integer 16-bit

149.31.18.49 public static final short UNSIGNED_INTEGER_24 = 98

Unsigned Integer 24-bit

149.31.18.50 public static final short UNSIGNED_INTEGER_32 = 99

Unsigned Integer 32-bit

149.31.18.51 public static final short UNSIGNED_INTEGER_40 = 100

Unsigned Integer 40-bit

149.31.18.52 public static final short UNSIGNED_INTEGER_48 = 101

Unsigned Integer 48-bit

149.31.18.53 public static final short UNSIGNED_INTEGER_56 = 102

Unsigned Integer 56-bit

149.31.18.54 public static final short UNSIGNED_INTEGER_64 = 103

Unsigned Integer 64-bit

149.31.18.55 public static final short UNSIGNED_INTEGER_8 = 96

According to ZigBee Cluster Library [1], the Unsigned Integer type represents an unsigned integer with a decimal range of 0 to 2^8-1, 0 to 2^16-1, 0 to 2^24-1, 0 to 2^32-1, 0 to 2^40-1, 0 to 2^48-1, 0 to 2^56-1, or 0 to 2^64-1, depending on its length. The values that represents an invalid value of this type are 0xff, 0xffff, 0xffffff, 0xffffffff, 0xffffffffff, 0xffffffffffff, 0xffffffffffffff and 0xffffffffffffffff respectively. This type is defined with several sizes: 8, 16, 24, 32, 40, 48, 56 and 64 bits.

149.31.18.56 public static final short UTC_TIME = 4

According to ZigBee Cluster Library [1], UTCTime is an unsigned 32-bit value representing the number of seconds since 0 hours, 0 minutes, 0 seconds, on the 1st of January, 2000 UTC (Universal Coordinated Time). The value that represents an invalid value of this type is 0xffffffff.

149.31.19 public interface ZigBeeEndpoint

This interface represents a ZigBee EndPoint. A ZigBeeEndpoint must be registered as a OSGi service with ZigBeeNode.IEEE_ADDRESS, and ZigBeeEndpoint.ENDPOINT_ID properties.

149.31.19.1 public static final String DEVICE_CATEGORY = "ZigBee"

Constant used by all ZigBee devices indicating the device category. It is a **mandatory** service property for this service.

149.31.19.2 public static final String DEVICE_ID = "zigbee.device.id"

Property containing the application device identifier. This identifier is also contained in the ZigBee Simple Descriptor. This property is of type Integer.

It is **mandatory** property for this service.

149.31.19.3 public static final String DEVICE_VERSION = "zigbee.device.version"

Property containing the application device version. The application device version is also contained in the ZigBee endpoint Simple Descriptor. This property is of type Byte.

It is **mandatory** property for this service.

149.31.19.4 public static final String ENDPOINT_ID = "zigbee.endpoint.id"

Property containing the EndPoint ID of the device. This property is of type Short and its value must be in the range allowed by the ZigBee specifications for Zigbee endpoints identifiers.

It is **mandatory** service property for ZigBeeEndpoint services.

149.31.19.5 public static final String HOST_PID = "zigbee.endpoint.host.pid"

Property containing the ZigBeeHost's pid. This property is of type String.

The ZigBee local host identifier is intended to uniquely identify the ZigBee local host, since there could be many hosts on the same platform.

All the endpoints that belong to a specific network MUST specify the value of the associated host pid. It is mandatory for imported endpoints, optional for exported endpoints.

149.31.19.6 public static final String INPUT_CLUSTERS = "zigbee.endpoint.clusters.input"

Property containing a list of input clusters. This list is contained also in the ZigBee Simple Descriptor returned by the ZigBeeEndpoint service. This property is of type int[].

It is **mandatory** service property for this service.

149.31.19.7 public static final String OUTPUT_CLUSTERS = "zigbee.endpoint.clusters.output"

Property containing a list of output clusters. This list is contained also in the ZigBee Simple Descriptor of the ZigBeeEndpoint service. This property is of type int[].

It is a **mandatory** service property for this service.

149.31.19.8 public static final String PROFILE_ID = "zigbee.device.profile.id"

Property containing the application profile identifier also contained in the ZigBee Simple Descriptor. This property is of type Integer.

It is **mandatory** service property for this service.

149.31.19.9 public static final String ZIGBEE_EXPORT = "zigbee.export"

Property used to mark if a ZigBeeEndPoint service is an exported one or not. Imported endpoints do not have this property set. This service property requires no specific values.

149.31.19.10 public Promise bind(String servicePid,int clusterId)

servicePid the PID of the endpoint to bind to

clusterId the cluster identifier to bind to

□ Adds the following entry in the *Binding Table* of the device:

`this.getNodeAddress(), this.getId(), clusterId, device.getNodeAddress(), device.getId()`

As described in "Table 2.7 APSME-BIND.confirm Parameters" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a binding request can have the following results: APSException.SUCCESS, APSException.ILLEGAL_REQUEST, APSException.TABLE_FULL, APSException.NOT_SUPPORTED.

Returns A promise representing the completion of this asynchronous call. Promise.getFailure() returns null if the cluster has been successfully bound. The adequate ZigBeeEndpoint is returned otherwise.

149.31.19.11 public Promise getBoundEndPoints(int clusterId)

clusterId the cluster identifier of the targeted bindings.

□ Returns bound endpoints (identified by their service PIDs) on a specific cluster ID. It is implemented on the base driver with Mgmt_Bind_req command. It is implemented without a command request in local endpoints.

As described in "Table 2.129 Fields of the Mgmt_Bind_rsp Command" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a Mgmt_Bind_rsp command can have the following status: APSException.NOT_SUPPORTED or any status code returned from the APSME-GET.confirm primitive (see APSException).

Returns A promise representing the completion of this asynchronous call. Promise.getValue() returns a List of the bound endpoint service PIDs if the command is successful. The response object is null and the adequate APSException is returned by Promise.getFailure() otherwise.

149.31.19.12 public ZCLCluster getClientCluster(int clientClusterId)

clientClusterId The client(output) cluster identifier.

□ Returns the client cluster identified by the cluster identifier.

Returns the client(output) cluster identified by the cluster identifier, or null if the given id is not listed in the simple descriptor.

Throws `IllegalArgumentException` – If the passed argument is outside the range [0, 0xffff].

149.31.19.13 public ZCLCluster[] getClientClusters()

□ Returns an array of client (output) clusters.

Returns an array of client (output) clusters, returns an empty array if does not provides any clients clusters.

149.31.19.14 public short getId()

□ Returns the identifier of this endpoint, that is the Endpoint ID.

Returns the identifier of this endpoint, value ranges from 1 to 240.

149.31.19.15 public BigInteger getNodeAddress()

□ Returns the IEEE Address of the node containing this endpoint.

Returns the IEEE Address of the node containing this endpoint.

149.31.19.16 public ZCLCluster getServerCluster(int serverClusterId)

serverClusterId The server(input) cluster identifier.

□ Returns the server (input) cluster identified by the given identifier.

Returns the server (input) cluster identified by the given identifier, or null if the given id is not listed in the simple descriptor.

Throws `IllegalArgumentException`– If the passed argument is outside the range [0, 0xffff].

149.31.19.17 public ZCLCluster[] getServerClusters()

□ Returns an array of server (input) clusters.

Returns an array of server (input) clusters, returns an empty array if it does not provide any server cluster.

149.31.19.18 public Promise getSimpleDescriptor()

□ Returns the simple descriptor of this endpoint. As described in "Table 2.93 Fields of the Simple_Desc_rsp Command" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a simple_decr request can have the following status: ZDPException.SUCCESS, ZDPException.INVALID_EP, ZDPException.NOT_ACTIVE, ZDPException.DEVICE_NOT_FOUND, ZDPException.INV_REQUESTTYPE or ZDPException.NO_DESCRIPTOR.

Returns A promise representing the completion of this asynchronous call. Promise.getValue() returns the node simple descriptor ZigBeeSimpleDescriptor in case of success and Promise.getFailure() returns the adequate ZDPException otherwise.

149.31.19.19 public void notExported(ZigBeeException e)

e A device ZigBeeException the occurred exception.

□ Notifies that the base driver is unable to export this endpoint. This method is called by the base driver and used to give details about issues preventing the export of an endpoint.

149.31.19.20 public Promise unbind(String servicePid,int clusterId)

servicePid The pid of the service to unbind.

clusterId The cluster identifier to unbind.

□ Removes the following entry in the *Binding Table* of the device if it exists:

`this.getNodeAddress(), this.getId(), clusterId, device.getNodeAddress(), device.getId()`

As described in "Table 2.9 APSME-UNBIND.confirm Parameters" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, an unbind request can have the following results: APSException.SUCCESS, APSException.ILLEGAL_REQUEST, APSException.INVALID_BINDING.

Returns A promise representing the completion of this asynchronous call. Promise.getFailure() returns null if the cluster has been successfully bound. The adequate APSException is returned otherwise.

149.31.20 public interface ZigBeeEvent

This interface represents events generated by a ZigBee Device node.

149.31.20.1 public int getAttributeId()

□ Returns the attribute identifier (that is, the attribute's ID).

Returns the attribute identifier (that is, the attribute's ID).

149.31.20.2 public int getClusterId()

□ Returns the cluster id associated to this ZigBeeEvent.

Returns the cluster id.

149.31.20.3 public short getEndpointId()

□ Returns the endpoint identifier.

Returns the endpoint identifier.

149.31.20.4 public BigInteger getIEEEAddress()

□ Returns the ZigBee device node IEEE Address.

Returns the ZigBee device node IEEE Address.

149.31.20.5 public Object getValue()

□ Returns an object containing the new value of the related ZigBee attribute.

Returns an object containing the new value for the ZigBee attribute that has changed.

149.31.21 public class ZigBeeException extends RuntimeException

This class represents root exception for all the code related to ZigBee. The provided constants names, but not the values.

149.31.21.1 protected final int errorCode

The error code associated to this exception.

See Also getErrorCode()

149.31.21.2 public static final int OSGI_EXISTING_ID = 48

The error code used when another endpoint exists with the same ID.

149.31.21.3 public static final int OSGI_MULTIPLE_HOSTS = 49

The error code used when several hosts exist for this PAN ID target or HOST_PID target.

149.31.21.4 public static final int TIMEOUT = 50

The error code used when the timeout of ZigBee asynchronous exchange is reached.

149.31.21.5 public static final int UNKNOWN_ERROR = -1

This error code is used if the ZigBee error returned is not covered by this API specification.

149.31.21.6 protected final int zigBeeErrorCode

The actual error code returned by the ZigBee node.

See Also ZigBeeException.getZigBeeErrorCode()

149.31.21.7 public ZigBeeException(String errorDesc)

errorDesc exception error description.

□ Creates a ZigBeeException containing only a description, but no error codes. If issued on this exception the getErrorCode() and getZigBeeErrorCode() methods return the UNKNOWN_ERROR constant.

149.31.21.8 public ZigBeeException(int errorCode,String errorDesc)

errorCode One of the error codes defined in this interface or UNKNOWN_ERROR if the actual error is not listed in this interface.

errorDesc An error description which explain the type of problem.

□ Creates a ZigBeeException containing a specific `errorCode`. Using this constructor with `errorCode` set to UNKNOWN_ERROR is equivalent to call ZigBeeException(String).

149.31.21.9 public ZigBeeException(int errorCode,int zigBeeErrorCode,String errorDesc)

errorCode One of the error codes defined in this interface or UNKNOWN_ERROR the actual error is not covered in this interface.

zigBeeErrorCode The actual status code or UNKNOWN_ERROR if this status is unknown.

errorDesc An error description which explain the type of problem.

□ Creates a ZigBeeException containing a specific `errorCode` or `zigBeeErrorCode`. Using this constructor with both the `errorCode` and `zigBeeErrorCode` set to UNKNOWN_ERROR is equivalent to call ZigBeeException(String).

149.31.21.10 public int getErrorCode()

□ Returns the error code.

Returns the error code.

149.31.21.11 public int getZigBeeErrorCode()

□ Returns the potential ZigBee error code.

Returns One of the error codes defined above. If the returned error code is UNKNOWN_ERROR and the hasZigBeeErrorCode() returns `true` then the getZigBeeErrorCode() provides the actual ZigBee error code returned by the device.

149.31.21.12 public boolean hasZigBeeErrorCode()

□ Checks if this exception has a ZigBee error code.

Returns true if the ZigBeeException convey also the actual error code returned by the ZigBee stack.

149.31.22 public interface ZigBeeGroup

This interface represents a ZigBee Group.

No Implement Consumers of this API must not implement this interface

149.31.22.1 public static final String ID = "zigbee.group.id"

Key of the String containing the Group Address of the device.

It is a **mandatory** property for this service.

149.31.22.2 public int getGroupAddress()

□ Returns the 16-bit group address.

Returns the 16-bit group address.

149.31.22.3 public ZCLCommandResponseStream groupcast(int clusterId,ZCLFrame frame)

clusterId a cluster identifier.

frame a command frame sequence.

□ Sends a ZCL frame to the group represented by this service. The returned stream will provide the invocation response(s) in an asynchronous way.

The source endpoint is not specified in this method call. To send the appropriate message on the network, the base driver must generate a source endpoint. The latter must not correspond to any exported endpoint.

Returns a ZCLCommandResponseStream to collect every ZCL frame one after the other in case of multiple responses.

149.31.22.4 public ZCLCommandResponseStream groupcast(int clusterId,ZCLFrame frame,String exportedServicePID)

clusterId a cluster identifier.

frame a command frame sequence.

exportedServicePID : the source endpoint of the command request. In targeted situations, the source endpoint is the valid service PID of an exported endpoint.

□ Sends a ZCL frame to the ZigBee group represented by this service. The returned stream will provide the invocation response(s) in an asynchronous way.

This method is to be used by applications when the targeted device has to distinguish between source endpoints of the message. For instance, alarms cluster (see 3.11 Alarms Cluster in [ZCL]) generated events are differently interpreted if they come from the oven or from the intrusion alert system.

Returns a ZCLCommandResponseStream to collect every ZCL frame one after the other in case of multiple responses.

149.31.22.5 public Promise joinGroup(String pid)

pid String representing the service PID of the ZigBeeEndpoint to add to this Group.

□ Requests an endpoint to join this group. This method may be invoked on exported and imported endpoints. In the former case, the ZigBee Base Driver should rely on the *APSME-ADD-GROUP* API defined by the ZigBee Specification, or it will use the proper commands of the *Groups* cluster of the ZigBee Specification Library. As described in "Table 2.15 APSME-ADD-GROUP.confirm Parameters" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, an add_group request can have the following status: APSException.SUCCESS, APSException.INVALID_PARAMETER or APSException.TABLE_FULL. When the joining is performed remotely on an imported ZigBeeEndpoint, it may also fail because the command is not supported by the remote endpoint, or because the remote device cannot perform the operation at the moment (see ZCLException).

Returns A promise representing the completion of this asynchronous call. Promise.getFailure() returns null if the cluster has been successfully bound. The adequate ZigBeeException is returned otherwise.

149.31.22.6 public Promise leaveGroup(String pid)

pid String representing the service PID of the ZigBeeEndpoint to remove from this Group.

□ Requests an endpoint to leave this group. This method may be invoked on exported and imported endpoints. In the former case, the ZigBee Base Driver should rely on the *APSME-REMOVE-GROUP* API defined by the ZigBee Specification, or it will use the proper commands of the *Groups* cluster of the ZigBee Specification Library. As described in "Table 2.17 APSME-REMOVE-GROUP.confirm Parameters" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a remove_group request can have the following status: APSException.SUCCESS, APSException.INVALID_GROUP or

APSException.INVALID_PARAMETER. When the command is invoked remotely on an imported ZigBeeEndpoint, it may also fail because the command is not supported by the remote endpoint, or because the remote device cannot perform the operation at the moment (see ZCLException).

Returns A promise representing the completion of this asynchronous call. Promise.getFailure() returns null if the cluster has been successfully bound. The adequate ZigBeeException is returned otherwise.

149.31.23 public interface ZigBeeHost extends ZigBeeNode

This interface represents the machine that hosts the code to run a ZigBee device or client. This machine is, for example, the ZigBee chip/dongle that is controlled by the base driver (below/under the OSGi execution environment).

ZigBeeHost is more than a ZigBeeNode.

It must be registered as a OSGi service.

Even if not specified explicitly in the javadoc, any method of this interface must throw an IllegalArgumentException exception if a or one of the passed arguments has a value not admitted by the method.

No Implement Consumers of this API must not implement this interface

149.31.23.1 public static final short UNLIMITED_BROADCAST_RADIUS = 255

Value constant to set an unlimited broadcast radius.

149.31.23.2 public ZCLCommandResponseStream broadcast(int clusterID,ZCLFrame frame)

clusterID The cluster ID this ZCL frame must be sent to.

frame A ZCL Frame.

□ Broadcasts a ZCL frame to the cluster ID of all the nodes of the ZigBee network. The setBroadcastRadius(short) method, may be used to limit the broadcast radius used in the subsequent broadcast calls.

Returns a response stream instance that collects and allows the caller to be asynchronously notified about the ZCLFrame responses sent back by the ZigBee nodes.

149.31.23.3 public ZCLCommandResponseStream broadcast(int clusterID,ZCLFrame frame,String exportedServicePID)

clusterID The cluster ID.

frame A ZCL Frame.

exportedServicePID the source endpoint of the command request. In targeted situations, the source endpoint is the valid service PID of an exported endpoint.

□ Broadcasts a ZCL frame to the cluster ID of all the nodes of the ZigBee network. The passed `exportedServicePID` allows to force the source endpoint of the message sent to be the endpoint id of the exported ZigBeeEndPoint service having the specified service.pid property.

Returns a response stream instance that collects and allows the caller to be asynchronously notified about the ZCLFrame responses sent back by the ZigBee nodes.

See Also Setting the broadcast radius.

149.31.23.4 public void createGroupService(int groupAddress) throws Exception

groupAddress the address of the group to create.

□ Creates a ZigBeeGroup service that has not yet been discovered by the ZigBee Base Driver or that does not exist on the ZigBee network yet.

Throws `Exception`– when a ZigBeeGroup service with the same groupAddress already exists.

149.31.23.5 public short getBroadcastRadius()

□ Returns the current broadcast radius value.

Returns the current broadcast radius value.

149.31.23.6 public int getChannel() throws Exception

□ Returns the current network channel.

Returns the current network channel.

Throws `Exception`– Any exception related to the communication with the chip.

149.31.23.7 public int getChannelMask() throws Exception

□ Returns the currently configured channel mask.

Returns the currently configured channel mask.

Throws `Exception`– Any exception related to the communication with the chip.

149.31.23.8 public long getCommunicationTimeout()

□ Returns the current value set for the communication timeout.

Returns the current value set for the communication timeout expressed in milliseconds.

149.31.23.9 public String getPreconfiguredLinkKey() throws Exception

□ Returns the current preconfigured link key.

Returns the current preconfigured link key.

Throws `Exception`– Any exception related to the communication with the chip.

149.31.23.10 public int getSecurityLevel() throws Exception

□ Returns the network security level.

Returns the network security level, that is, 0 if security is disabled, an int code if enabled (see "Table 4.38 Security Levels Available to the NWK, and APS Layers" of the ZigBee specification").

Throws `Exception`– Any exception related to the communication with the chip.

149.31.23.11 public boolean isStarted()

□ Checks the host's start/stop state.

Returns true if the host is started.

149.31.23.12 public void permitJoin(short duration) throws Exception

duration The time during which associations are permitted.

□ Indicates if a ZigBee device can join the network.

Broadcasts a Mgmt_Permit_req to all routers and the coordinator. If the duration argument is not equal to zero or 0xFF, the argument is a number of seconds and joining is permitted until it counts down to zero, after which time, joining is not permitted. If the duration is set to zero, joining is not permitted. If set to 0xFF, joining is permitted indefinitely or until another Mgmt_Permit_Joining_req is received by the coordinator.

As described in "Table 2.133 Fields of the Mgmt_Permit_Joining_rsp Command" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a permitJoin request can have the following status: ZDPException.SUCCESS, ZDPException.INV_REQUESTTYPE, ZDPException.NOT_AUTHORIZED or any status code returned from the NLMEPERMITJOINING.confirm primitive.

Throws Exception– Any exception related to the communication with the chip.

149.31.23.13 public Promise refreshNetwork() throws Exception

□ Forces a new network scan. It checks that the ZigBeeNode services are still representing an available node on the network. It also updates the whole representation of all nodes (endpoints, clusters, descriptors, attributes).

Returns A promise representing the completion of this asynchronous call. In case of success the promise will resolve with Boolean.TRUE otherwise the promise is failed with an exception.

Throws Exception– Any exception related to the communication with the chip.

149.31.23.14 public void setBroadcastRadius(short broadcastRadius)

broadcastRadius - is the number of routers that the messages are allowed to cross. Radius value is in the range from 0 to 0xff.

□ Sets the broadcast radius value. By default the ZigBeeHost must use UNLIMITED_BROADCAST_RADIUS as default value for the broadcast.

Throws IllegalArgumentException– if set with a value out of the expected range.

IllegalStateException– if set when the ZigBeeHost is "running".

149.31.23.15 public void setChannelMask(int mask) throws IOException

mask A value representing the channel mask.

□ Sets a new configured channel mask.

As described in "Table 2.13 APSME-SET.confirm Parameters" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a set request can have the following status: APSException.SUCCESS, APSException.INVALID_PARAMETER or APSException.UNSUPPORTED_ATTRIBUTE.

Throws IllegalStateException– If the host is already started.

IOException– for serial communication exception.

149.31.23.16 public void setCommunicationTimeout(long timeout)

timeout the number of milliseconds before firing a timeout exception.

□ Sets the timeout for the communication sent through this device.

149.31.23.17 public void setExtendedPanId(BigInteger extendedPanId)

extendedPanId The network Extended PAN identifier(EPID)

□ Sets the extendedPanId.

Throws IllegalStateException– If the host is already started.

149.31.23.18 public void setLogicalType(short logicalNodeType) throws Exception

logicalNodeType The logical node type.

□ Sets the host logical node type. ZigBee defines three different types of node: ZigBeeNode.COORDINATOR, ZigBeeNode.ROUTER and ZigBeeNode.ZED.

Throws IllegalStateException– If the host is already started.

Exception– Any exception related to the communication with the chip.

149.31.23.19 public void setPanId(int panId)

panId The network Personal Area Network identifier (PAN ID)

□ Sets the panId.

Throws IllegalArgumentException– if set with a value out of the expected range [0x0000, 0xffff].

IllegalStateException– If the host is already started.

149.31.23.20 public void start() throws Exception

□ Starts the host. If the host is a ZigBeeNode.COORDINATOR, then it can be started with or without ZigBeeNode.PAN_ID and ZigBeeNode.EXTENDED_PAN_ID (that is, if no PAN_ID, and Extended PAN_ID are given, then they will be automatically generated and then added to the service properties).

If the host is a ZigBeeNode.ROUTER, or a ZigBeeNode.ZED, then the host may start without a registered ZigBeeNode.PAN_ID property; the property will be set when the host will find and join a ZigBee network.

The host status must be persistent, that is, if the host was started, then the host must starts again when the bundle restarts. In addition, the values of channel, pan id, extended pan id, and host PID must remain the same.

Throws Exception– Any exception related to the communication with the chip.

149.31.23.21 public void stop() throws Exception

□ Stops the host.

Throws Exception– Any exception related to the communication with the chip.

149.31.23.22 public void updateNetworkChannel(byte channel) throws IOException

channel The network channel.

□ Updates the network channel. 802.15.4 and ZigBee divide the 2.4GHz band into 16 channels, numbered from 11 to 26.

As described in "Table 2.4.3.3.9 Mgmt_NWK_Update_req" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, this request is sent as broadcast by the network manager with a ScanDuration to be set with the channel parameter.

Throws IllegalStateException– If the host is started, or the host is not a network manager.

IOException– for serial communication exception.

149.31.24 public interface ZigBeeLinkQuality

This interface represents an entry of the NeighborTableList.

See Table 2.126 NeighborTableList Record Format in ZIGBEE SPECIFICATION: 1_053474r17ZB_TSC-ZigBee-Specification.pdf.

No Implement Consumers of this API must not implement this interface

149.31.24.1 public static final int CHILD_NEIGHBOR = 241

Constant value representing a child relationship between current ZigBeeNode and the neighbor.

149.31.24.2 public static final int OTHERS_NEIGHBOR = 243

Constant value representing a others relationship between current ZigBeeNode and the neighbor.

149.31.24.3 public static final int PARENT_NEIGHBOR = 240

Constant value representing a parent relationship between current ZigBeeNode and the neighbor.

149.31.24.4 public static final int PREVIOUS_CHILD_NEIGHBOR = 244

Constant value representing a previous child relationship between current ZigBeeNode and the neighbor.

149.31.24.5 public static final int SIBLING_NEIGHBOR = 242

Constant value representing a sibling relationship between current ZigBeeNode and the neighbor.

149.31.24.6 public int getDepth()

- Returns the depth field of the NeighborTableList Record Format.

Returns the tree-depth of device.

149.31.24.7 public int getLQI()

- Returns the Link Quality Indicator. See the LQI field of the NeighborTableList Record Format.

Returns the Link Quality Indicator estimated by ZigBeeNode returning this for communicating with ZigBeeNode identified by the getNeighbor().

149.31.24.8 public String getNeighbor()

- Returns the Service.PID referring to the ZigBeeNode representing a neighbor.

Returns the Service.PID referring to the ZigBeeNode representing a neighbor.

149.31.24.9 public int getRelationship()

- Returns the relationship with the neighbor. See the Relationship field of the NeighborTableList Record Format.

Returns the relationship between ZigBeeNode returning this LQI and the ZigBeeNode identified by the getNeighbor().

149.31.25 public interface ZigBeeNode

This interface represents a ZigBee node, means a physical device that can communicate using the ZigBee protocol.

Each physical device may contain up 240 logical devices which are represented by the ZigBeeEndpoint class.

Each logical device is identified by an *EndPoint* address, but shares:

- either the *64-bit 802.15.4 IEEE Address*
- or the *16-bit ZigBee Network Address*.

No Implement Consumers of this API must not implement this interface

149.31.25.1 public static final short COORDINATOR = 2

Constant value used as logical type value when the ZigBee device is a Coordinator.

149.31.25.2 public static final String EXTENDED_PAN_ID = "zigbee.node.extended.pan.id"

Key of String containing the device node network extended PAN ID. If the device type is "Coordinator", the extended pan id may be available only after the network is started. It means that internally the ZigBeeHost interface must update the service properties.

This property is of type BigInteger

149.31.25.3 public static final String IEEE_ADDRESS = "zigbee.node.ieee.address"

Property key for the mandatory node IEEE Address representing node MAC address. MAC Address is a 12-digit(48-bit) or 16-digit(64-bit) hexadecimal numbers.

149.31.25.4 public static final String LOGICAL_TYPE = "zigbee.node.description.node.type"

Property name for the device logical type. The property value is of type Short.

149.31.25.5 public static final String MANUFACTURER_CODE = "zigbee.node.description.manufacturer.code"

Property name for a manufacturer code that is allocated by the ZigBee Alliance, relating the manufacturer to the device. The property is of type Integer.

149.31.25.6 public static final String PAN_ID = "zigbee.node.pan.id"

Property containing the ZigBee network PAN ID. The property is of type Integer.

149.31.25.7 public static final String POWER_SOURCE = "zigbee.node.power.source"

ZigBee power source, that is, 3rd bit of "MAC Capabilities" in Node Descriptor. Set to true if the current power source is mains power, set to false, otherwise.

This property is of type Boolean.

149.31.25.8 public static final String RECEIVER_ON_WHEN_IDLE = "zigbee.node.receiver.on.when.idle"

ZigBee receiver on when idle, that is, 4th bit of "MAC Capabilities" in Node Descriptor. Set to true if the device does not disable its receiver to conserve power during idle periods, set to false otherwise.

This property is of type Boolean.

149.31.25.9 public static final short ROUTER = 3

Constant value used as logical type value when the ZigBee device is a Router.

149.31.25.10 public static final short ZED = 1

Constant value used as logical type value when the ZigBee device is an End Device.

149.31.25.11 public ZCLCommandResponseStream broadcast(int clusterID,ZCLFrame frame)

clusterID the cluster ID the broadcast message is directed.

frame a ZCL Frame that contains the command that have to be broadcast to the specific cluster of all the endpoints running on the node.

□ Broadcasts a given ZCL Frame to cluster clusterID on all the ZigBeeEndpoint that are running on this node (endpoint broadcasting).

Returns a response stream instance that collects and allows a client to be asynchronously notified about the ZCLFrame responses sent back by the ZigBee nodes.

149.31.25.12 public ZCLCommandResponseStream broadcast(int clusterID,ZCLFrame frame,String exportedServicePID)

clusterID the cluster ID the broadcast message is directed.

frame a ZCL Frame that contains the command that have to be broadcast to the specific cluster of all the endpoints running on the node.

exportedServicePID the source endpoint of the command request. In targeted situations, the source endpoint is the valid service PID of an exported endpoint.

□ Broadcasts a given ZCL Frame to cluster clusterID on all the ZigBeeEndpoint that are running on this node (endpoint broadcasting). The source endpoint of the APS message sent, is set to the endpoint identifier of the exportedServicePID service.

Returns a response stream instance that collects and allows the caller to be asynchronously notified about the ZCLFrame responses sent back by the ZigBee nodes.

149.31.25.13 public Promise getComplexDescriptor()

□ Retrieves the ZigBee node Complex Descriptor.

As described in *Table 2.92 Fields of the Complex_Desc_rsp Command* of the ZigBee specification, a *Complex_Desc_rsp* command can return with the following status codes:

- ZDPException.SUCCESS
- ZDPException.DEVICE_NOT_FOUND
- ZDPException.INV_REQUESTTYPE
- ZDPException.NO_DESCRIPTOR

Returns A promise representing the completion of this asynchronous call. It will be used in order to return the complex descriptor ZigBeeComplexDescriptor. If the ZDP *Complex_Desc_rsp* do not return success, the promise must fail with a ZDPException exception with the correct status code.

149.31.25.14 public ZigBeeEndpoint[] getEndpoints()

□ Returns the array of the endpoints hosted by this node.

Returns the array of the endpoints hosted by this node, returns an empty array if this node does not host any endpoint.

149.31.25.15 public BigInteger getExtendedPanId()

□ Returns the network Extended PAN identifier (EPID).

Returns the network Extended PAN identifier (EPID).

149.31.25.16 public String getHostPid()

□ Returns the OSGi service PID of the ZigBee Host that is on the network of this node.

Returns the OSGi service PID of the ZigBee Host that is on the network of this node.

149.31.25.17 public BigInteger getIEEEAddress()

□ Returns the ZigBee device node IEEE Address of this node.

Returns the ZigBee device node IEEE Address of this node.

149.31.25.18 public Promise getLinksQuality()

□ Retrieves the link quality information to the neighbor nodes.

An implementation of this method may use the *Mgmt_Lqi_req* and *Mgmt_Lqi_rsp* messages to retrieve the Link Quality table (also known as NeighborTableList in the ZigBee Specification).

The method limit the Link Quality table to the ZigBeeNode service discovered.

In case of failure, the target device may report error code ZDPException.NOT_SUPPORTED.

Returns A promise representing the completion of this asynchronous call. It will be resolved with the result of this operation. In case of success the resolved value will be a Map containing the Service.PID as String key of the ZigBeeNode service and the value the ZigBeeLinkQuality for that node. In case of errors the promise must fail with the correct ZDPException.

149.31.25.19 public int getNetworkAddress()

□ Returns the current network address (alias short-address) of this node.

Returns the current network address of this node.

149.31.25.20 public Promise getNodeDescriptor()

□ Retrieves the ZigBee node Node Descriptor. As described in *Table 2.91 Fields of the Node_Desc_rsp Command* of the ZigBee specification, a *Node_Desc_rsp* command can return with the following status codes:

- ZDPException.SUCCESS
- ZDPException.DEVICE_NOT_FOUND
- ZDPException.INV_REQUESTTYPE

- ZDPException.NO_DESCRIPTOR

Returns A promise representing the completion of this asynchronous call. It will be used in order to return the node descriptor ZigBeeNodeDescriptor. If the ZDP *Node_Desc_rsp* do not return success, the promise must fail with a ZDPException exception with the correct status code.

149.31.25.21 public int getPanId()

□ Returns the network Personal Area Network identifier (PAN ID).

Returns the network Personal Area Network identifier (PAN ID).

149.31.25.22 public Promise getPowerDescriptor()

□ Retrieves the ZigBee node Power Descriptor. As described in *Table 2.92 Fields of the Power_Desc_rsp Command* of the ZigBee specification, a *Power_Desc_rsp* command can return with the following status codes:

- ZDPException.SUCCESS
- ZDPException.DEVICE_NOT_FOUND
- ZDPException.INV_REQUESTTYPE
- ZDPException.NO_DESCRIPTOR

Returns A promise representing the completion of this asynchronous call. It will be used in order to return the node power descriptor ZigBeePowerDescriptor. If the ZDP *Power_Desc_rsp* do not return success, the promise must fail with a ZDPException exception with the correct status code.

149.31.25.23 public Promise getRoutingTable()

□ Retrieves the routing table information of the node. This routing table is also known as RoutingTableList in the ZigBee Specification.

An implementation of this method may use the *Mgmt_Rtg_req* ZDP command to retrieve the Routing Table .

The target device may report a status code ZDPException.NOT_SUPPORTED in case of failure.

Returns A promise representing the completion of this asynchronous call. In case of success, the resolved value will be a Map containing the Service.PID as String key of the ZigBeeNode service and the value the ZigBeeRoute for that node. In case of failure a ZDPException exception with the correct status code must be used to fail the promise.

149.31.25.24 public Promise getUserDescription()

□ Returns the user description of this node. As described in *Table 2.97 Fields of the User_Desc_rsp Command* of the ZigBee specification, a User_Desc_rsp may return the following status:

- ZDPException.SUCCESS
- ZDPException.NOT_SUPPORTED
- ZDPException.DEVICE_NOT_FOUND
- ZDPException.INV_REQUESTTYPE
- ZDPException.NO_DESCRIPTOR

Returns A promise representing the completion of this asynchronous call. It will be used in order to return the node user description (String). In case of errors the promise will fail with a ZDPException exception containing the response status code value.

149.31.25.25 public Promise invoke(int clusterIdReq,int expectedClusterIdRsp,ZDPFrame message)

clusterIdReq the cluster Id of the ZDPFrame that will be sent to the device.

expectedClusterIdRsp the expected cluster Id of the response to the ZDPFrame sent.

message the ZDPFrame containing the message.

□ Sends the ZDPFrame to this ZigBeeNode with the specified cluster id. This method expects a specific cluster in the response to the request.

Returns A promise representing the completion of this asynchronous call. In case of success the promise resolves with the response ZDPFrame.

149.31.25.26 public Promise invoke(int clusterIdReq,ZDPFrame message)

clusterIdReq the cluster Id of the ZDPFrame that will be sent to the device.

message the ZDPFrame containing the message.

□ Sends the ZDPFrame to this ZigBeeNode with the specified cluster id. This method expects a specific cluster in the response to the request. This method considers that the 0x8000 + clusterIdReq is the clusterId expected from messaged received for the message sent by this request.

Returns A promise representing the completion of this asynchronous call. In case of success the promise resolves with the response ZDPFrame.

149.31.25.27 public Promise leave()

□ Requests this node to leave the ZigBee network.

As described in *Table 2.131 Fields of the Mgmt_Leave_rsp Command* of the ZigBee specification, a Mgmt_Leave_rsp ZDP command may return the following status values:

- ZDPException.SUCCESS
- ZDPException.NOT_SUPPORTED
- ZDPException.NOT_AUTHORIZED
- any status code returned from the NLMELEAVE.confirm primitive

Returns A promise representing the completion of this asynchronous call. In case of success, the promise is resolved with a `null` value, otherwise with the correct ZDPException exception.

149.31.25.28 public Promise leave(boolean rejoin,boolean removeChildren)

rejoin `true` if the device being asked to leave from the current parent is requested to rejoin the network. Otherwise, `false`.

removeChildren `true` if the device being asked to leave the network is also being asked to remove its child devices, if any. Otherwise, `false`.

□ Requests the device to leave the network.

As described in *Table 2.131 Fields of the Mgmt_Leave_rsp Command* of the ZigBee specification, a Mgmt_Leave_rsp command could return the following status values:

- ZDPException.SUCCESS
- ZDPException.NOT_SUPPORTED
- ZDPException.NOT_AUTHORIZED
- any status code returned from the NLMELEAVE.confirm primitive

Returns A promise representing the completion of this asynchronous call. In case of success, the ZigBeeNode service must be unregistered, first and then the promise may be resolved with a `null` value, otherwise with the correct ZDPException exception.

149.31.25.29 public Promise setUserDescription(String userDescription)

userDescription the user description.

□ Sets the user description of this node. As described in *Table 2.137 ZDP Enumerations Description* of the ZigBee specification, a Set_User_Desc_rsp request may return the following status:

- ZDPException.SUCCESS
- ZDPException.DEVICE_NOT_FOUND
- ZDPException.INV_REQUESTTYPE
- ZDPException.NO_DESCRIPTOR

Returns A promise representing the completion of this asynchronous call. In case of success the promise returns a null value. In case of errors the promise must fail with a ZDPException exception containing the response status code value.

149.31.26 public interface ZigBeeRoute

This interface represents an entry of the RoutingTableList

See Table 2.128 RoutingTableList Record Format in ZIGBEE SPECIFICATION: 1_053474r17ZB_TSC-ZigBee-Specification.pdf.

No Implement Consumers of this API must not implement this interface

149.31.26.1 public static final int ACTIVE = 240

Constant value representing an active route.

149.31.26.2 public static final int DISCOVERY_FAILED = 242

Constant value representing a failed route discovery.

149.31.26.3 public static final int DISCOVERY_UNDERWAY = 241

Constant value representing a route that is under discovery.

149.31.26.4 public static final int INACTIVE = 243

Constant value representing an inactive route.

149.31.26.5 public static final int VALIDATION_UNDERWAY = 244

Constant value representing a route which is under validation.

149.31.26.6 public String getDestination()

□ Returns the service PID of the ZigBeeNode as destination of this route entry.

Returns the service PID of the ZigBeeNode as destination of this route entry.

149.31.26.7 public String getNextHop()

□ Returns the service PID of the ZigBeeNode to send the data for reaching the destination.

Returns the service PID of the ZigBeeNode to send the data for reaching the destination.

149.31.26.8 public int getStatus()

□ Returns the status of this route.

Returns the status of this route (or routing link) as defined by ZigBee Specification: ACTIVE, DISCOVERY_UNDERWAY, DISCOVERY_FAILED, INACTIVE, VALIDATION_UNDERWAY.

149.32 org.osgi.service.zigbee.descriptions

Device Service Specification for ZigBee Technology Descriptions.

This package contains the interfaces for descriptions. The latter may be used to embed meta information about the ZigBee devices, and in other words a meta description of each device type present in a ZCL profile, or even custom devices.

It is not mandatory to provide this meta model for being able to interact with a specific device, but the presence of this meta model would make much easier to implement, for example user interfaces.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. This package has two types of users: the consumers that use the API in this package and the providers that implement the API in this package.

Example import for consumers using the API in this package:

```
Import-Package: org.osgi.service.zigbee.descriptions; version="[1.0,2.0)"
```

Example import for providers implementing the API in this package:

```
Import-Package: org.osgi.service.zigbee.descriptions; version="[1.0,1.1)"
```

149.32.1 Summary

- `ZCLAttributeDescription` - This interface represents a ZCLAttributeDescription.
- `ZCLClusterDescription` - This interface represents a ZCL Cluster description.
- `ZCLCommandDescription` - This interface represents a ZCLCommandDescription.
- `ZCLDataTypeDescription` - This interface is used for representing any of the ZigBee Data Types defined in the ZCL.
- `ZCLGlobalClusterDescription` - This interface represents Cluster global description.
- `ZCLParameterDescription` - This interface represents a ZigBee parameter description.
- `ZCLSimpleTypeDescription` - This interface is used for representing any of the simple ZigBee Data Types defined in the ZCL.
- `ZigBeeDeviceDescription` - This interface represents a ZigBee device description.
- `ZigBeeDeviceDescriptionSet` - This interface represents a ZigBee Device description Set.

149.32.2 public interface ZCLAttributeDescription extends ZCLAttributeInfo

This interface represents a ZCLAttributeDescription.

149.32.2.1 public Object getDefaultValue()

□ Returns the attribute default value.

Returns the attribute default value.

149.32.2.2 public String getName()

□ Returns the attribute name.

Returns the attribute name.

149.32.2.3 public String getShortDescription()

□ Returns the attribute functional description.

Returns the attribute functional description.

149.32.2.4 public boolean isMandatory()

□ Checks if this attribute is mandatory.

Returns true, if and only if the attribute is mandatory.

149.32.2.5 public boolean isPartOfAScene()

□ Checks if this attribute is part of a scene.

Returns true if the attribute is part of a scene (cluster), false otherwise.

149.32.2.6 public boolean isReadOnly()

□ Checks if this attribute is read-only.

Returns true if the attribute is read only, false otherwise (that is, if the attribute is read/write or optionally writable (R∗W)).

149.32.2.7 public boolean isReportable()

□ Checks if this attribute is reportable.

Returns true if and only if the attribute support subscription.

149.32.3 public interface ZCLClusterDescription

This interface represents a ZCL Cluster description.

149.32.3.1 public ZCLAttributeDescription[] getAttributeDescriptions()

□ Returns an array of the attribute descriptions.

Returns an array of the attribute descriptions.

149.32.3.2 public ZCLCommandDescription[] getGeneratedCommandDescriptions()

□ Returns an array of the generated command descriptions.

Returns an array of the generated command descriptions.

149.32.3.3 public ZCLGlobalClusterDescription getGlobalClusterDescription()

□ Returns an array of the command descriptions.

Returns an array of the command descriptions.

149.32.3.4 public int getId()

Returns the cluster identifier.

149.32.3.5 public ZCLCommandDescription[] getReceivedCommandDescriptions()

□ Returns an array of the received command description.

Returns an array of the received command description.

149.32.4 public interface ZCLCommandDescription

This interface represents a ZCLCommandDescription.

149.32.4.1 public short getId()

□ Returns the command identifier.

Returns the command identifier.

149.32.4.2 public int getManufacturerCode()

□ Returns the manufacturer code. Default value is: -1 (no code).

Returns the manufacturer code.

149.32.4.3 public String getName()

□ Returns the command name.

Returns the command name.

149.32.4.4 public ZCLParameterDescription[] getParameterDescriptions()

□ Returns an array of the parameter descriptions.

Returns an array of the parameter descriptions.

149.32.4.5 public String getShortDescription()

□ Returns the command functional description.

Returns the command functional description.

149.32.4.6 public boolean isClientServerDirection()

□ Checks if this is a server-side command (that is going from the client to server direction).

Returns the isClientServerDirection value.

149.32.4.7 public boolean isClusterSpecificCommand()

Returns the isClusterSpecificCommand value.

149.32.4.8 public boolean isMandatory()

□ Checks if this command it mandatory.

Returns true, if and only if the command is mandatory.

149.32.4.9 public boolean isManufacturerSpecific()

□ Checks if the command is manufacturer specific.

Returns `true` if end only if getManufacturerCode() is not. -1.

149.32.5 public interface ZCLDataTypeDescription

This interface is used for representing any of the ZigBee Data Types defined in the ZCL. Each of these data types has a set of associated information that this interface definition permits to retrieve using the specific methods.

- The data type identifier
- The data type name
- The data type is analog or digital
- The Java class used to represent the data type.

No Implement Consumers of this API must not implement this interface

149.32.5.1 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.32.5.2 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.32.5.3 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.32.5.4 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.32.6 public interface ZCLGlobalClusterDescription

This interface represents Cluster global description.

149.32.6.1 public ZCLClusterDescription getClientClusterDescription()

□ Returns the client cluster description.

Returns the client cluster description.

149.32.6.2 public String getClusterDescription()

□ Returns the cluster functional description.

Returns the cluster functional description.

149.32.6.3 public String getClusterFunctionalDomain()

□ Returns the cluster functional domain.

Returns the cluster functional domain.

149.32.6.4 public int getClusterId()

□ Returns the cluster identifier.

Returns the cluster identifier.

149.32.6.5 public String getClusterName()

□ Returns the cluster name.

Returns the cluster name.

149.32.6.6 public ZCLClusterDescription getServerClusterDescription()

□ Returns the server cluster description.

Returns the server cluster description.

149.32.7 public interface ZCLParameterDescription

This interface represents a ZigBee parameter description.

149.32.7.1 public ZCLDataTypeDescription getDataTypeDescription()

□ Returns the parameter data type.

Returns the parameter data type.

149.32.8 public interface ZCLSimpleTypeDescription extends ZCLDataTypeDescription

This interface is used for representing any of the simple ZigBee Data Types defined in the ZCL.

The interface extends the ZCLDataTypeDescription by providing serialize and deserialize methods to marshal and unmarshal the data into the ZigBeeDataInput and from ZigBeeDataOutput streams.

Related documentation: [1] ZigBee Cluster Library specification, Document 075123r04ZB, May 29, 2012.

No Implement Consumers of this API must not implement this interface

149.32.8.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws NullPointerException– If ZigBeeDataInput parameter is null.

IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.32.8.2 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

NullPointerException– If ZigBeeDataOutput parameter is null.

IllegalArgumentException– If the passed value parameter does not belong to the expected class or its value exceeds the possible values allowed (range or length).

149.32.9 public interface ZigBeeDeviceDescription

This interface represents a ZigBee device description.

149.32.9.1 public ZCLClusterDescription[] getClientClustersDescriptions()

□ Returns an array of client cluster descriptions.

Returns an array of client cluster descriptions.

149.32.9.2 public int getId()

□ Returns the device identifier.

Returns the device identifier.

149.32.9.3 public String getName()

□ Returns the device name.

Returns the device name.

149.32.9.4 public int getProfileId()

□ Returns the profile identifier.

Returns the profile identifier.

149.32.9.5 public ZCLClusterDescription[] getServerClustersDescriptions()

□ Returns an array of server cluster descriptions.

Returns an array of server cluster descriptions.

149.32.9.6 public Integer getVersion()

□ Returns the device version.

Returns the device version.

149.32.10 public interface ZigBeeDeviceDescriptionSet

This interface represents a ZigBee Device description Set. A Set is registered as an OSGi Service that provides method to retrieve endpoint descriptions. In addition to the ZigBeeDeviceDescriptionSet's (OSGi service) properties; ZigBeeDeviceDescriptionSet is also expected to be registered as an OSGi service with the following ZigBeeEndpoint.PROFILE_ID, and ZigBeeNode.MANUFACTURER_CODE properties.

149.32.10.1 public static final String DEVICES = "zigbee.profile.devices"

Property key for a comma separated list of devices identifiers supported by the set. This property is **mandatory**.

149.32.10.2 public static final String PROFILE_NAME = "zigbee.profile.name"

Property key for a profile name. This property is **mandatory**.

149.32.10.3 public static final String VERSION = "zigbee.profile.version"

Property key for a version of the application profile. The format is 'major.minor' with major and minor being integers. This property is **mandatory**.

149.32.10.4 public ZigBeeDeviceDescription getDeviceSpecification(int deviceId,short version)

deviceId Identifier of the device.

version The version of the application profile.

□ Returns the description of a device identified by its identifier and its version.

Returns The associated device description.

149.33 org.osgi.service.zigbee.descriptors

Device Service Specification for ZigBee Technology Descriptors.

This package contains the interfaces representing the ZigBee descriptors and the fields defined inside some of them.

An interface for modeling the ZigBee User Descriptor is missing because this descriptor has only one field (the UserDescription). Therefore this field can be read and written using respectively the org.osgi.service.zigbee.ZigBeeNode.getUserDescription() and the org.osgi.service.zigbee.ZigBeeNode.setUserDescription(String) methods.

The org.osgi.service.zigbee.descriptors.ZigBeeNodeDescriptor, org.osgi.service.zigbee.descriptors.ZigBeePowerDescriptor and the org.osgi.service.zigbee.descriptors.ZigBeeComplexDescriptor are read using the appropriate methods in the org.osgi.service.zigbee.ZigBeeNode interface, whereas the ZigBeeSimpleDescriptor can be read using the appropriate method of the org.osgi.service.zigbee.ZigBeeEndpoint services registered in the framework.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. This package has two types of users: the consumers that use the API in this package and the providers that implement the API in this package.

Example import for consumers using the API in this package:

```
Import-Package: org.osgi.service.zigbee.descriptors; version="[1.0,2.0)"
```

Example import for providers implementing the API in this package:

```
Import-Package: org.osgi.service.zigbee.descriptors; version="[1.0,1.1)"
```

149.33.1 Summary

- `ZigBeeComplexDescriptor` - This interface represents a Complex Descriptor as described in the ZigBee Specification.
- `ZigBeeFrequencyBand` - This interface represents a the frequency band field.
- `ZigBeeMacCapabiliyFlags` - This interface represents the Node Descriptor MAC Capability Flags as described in the ZigBee Specification.
- `ZigBeeNodeDescriptor` - This interface represents a Node Descriptor as described in the ZigBee Specification.
- `ZigBeePowerDescriptor` - This interface represents a power descriptor as described in the ZigBee Specification.
- `ZigBeeServerMask` - Represents the ZigBee Server Mask field of the ZigBee Node Descriptor.
- `ZigBeeSimpleDescriptor` - This interface represents a simple descriptor as described in the ZigBee Specification.

149.33.2 public interface ZigBeeComplexDescriptor

This interface represents a Complex Descriptor as described in the ZigBee Specification.

The Complex Descriptor contains extended information for each of the device descriptions contained in the node. The use of the Complex Descriptor is optional.

No Implement Consumers of this API must not implement this interface

149.33.2.1 public String getCharacterSetIdentifier()

□ Returns the encoding used by characters in the character set.

Returns the encoding used by characters in the character set.

149.33.2.2 public String getDeviceURL()

□ Returns the Device URL.

Returns the Device URL.

149.33.2.3 public byte[] getIcon()

□ Returns the icon field.

Returns the icon field.

149.33.2.4 public String getIconURL()

□ Returns the icon URL.

Returns the icon URL.

149.33.2.5 public String getLanguageCode()

□ Returns the language code used for character strings.

Returns the language code used for character strings.

149.33.2.6 public String getManufacturerName()

□ Returns the manufacturer name.

Returns the manufacturer name.

149.33.2.7 public String getModelName()

□ Returns the model name.

Returns the model name.

149.33.2.8 public String getSerialNumber()

□ Returns the serial number.

Returns the serial number.

149.33.3 public interface ZigBeeFrequencyBand

This interface represents a the frequency band field.

No Implement Consumers of this API must not implement this interface

149.33.3.1 public boolean is2400()

□ Checks if the radio band is 2.4GHz.

Returns true if and only if the radio is operating in the frequency band 2400MHz to 2483MHz.

149.33.3.2 public boolean is868()

□ Checks if the radio band is 868MHz.

Returns true if and only if the radio is operating in the frequency band 868 to 868.6 MHz.

149.33.3.3 public boolean is915()

□ Checks if the radio band is 900MHz.

Returns true if and only if the radio is operating in the frequency band 908MHz to 928MHz.

149.33.4 public interface ZigBeeMacCapabiliyFlags

This interface represents the Node Descriptor MAC Capability Flags as described in the ZigBee Specification.

No Implement Consumers of this API must not implement this interface

149.33.4.1 public boolean isAddressAllocate()

□ Checks if the device is address allocate.

Returns true if the device is address allocate or false otherwise.

149.33.4.2 public boolean isAlternatePANCoordinator()

□ Checks if this node is capable of becoming PAN coordinator.

Returns true if this node is capable of becoming PAN coordinator or false otherwise.

149.33.4.3 public boolean isFullFunctionDevice()

□ Checks if this node a Full Function Device (FFD).

Returns true if this node a Full Function Device (FFD), false otherwise (it is a Reduced Function Device, RFD).

149.33.4.4 public boolean isMainsPower()

□ Checks if the current power source is mains power.

Returns true if the current power source is mains power or false otherwise.

149.33.4.5 public boolean isReceiverOnWhenIdle()

□ Checks if the device does not disable its receiver to conserve power during idle periods.

Returns true if the device does not disable its receiver to conserve power during idle periods or false otherwise.

149.33.4.6 public boolean isSecurityCapable()

□ Checks if the device is capable of sending and receiving secured frames

Returns true if the device is capable of sending and receiving secured frames or false otherwise.

149.33.5 public interface ZigBeeNodeDescriptor

This interface represents a Node Descriptor as described in the ZigBee Specification.

The Node Descriptor contains information about the capabilities of the node.

No Implement Consumers of this API must not implement this interface

149.33.5.1 public ZigBeeFrequencyBand getFrequencyBand()

□ Returns the radio frequency band the node is currently operating on.

Returns returns the information about the radio frequency band the node is currently operating on.

149.33.5.2 public short getLogicalType()

□ Returns the logical type of the described node.

Returns one of: ZigBeeNode.COORDINATOR, ZigBeeNode.ROUTER, ZigBeeNode.ZED.

149.33.5.3 public ZigBeeMacCapabiliyFlags getMacCapabilityFlags()

□ Returns the MAC Capability Flags field information.

Returns the MAC Capability Flags field information.

149.33.5.4 public int getManufacturerCode()

□ Returns the manufacturer code of the described node.

Returns the manufacturer code of the described node.

149.33.5.5 public int getMaxBufferSize()

□ Returns the maximum buffer size of the described node.

Returns the maximum buffer size of the described node.

149.33.5.6 public int getMaxIncomingTransferSize()

□ Returns the maximum incoming transfer size of the described node.

Returns the maximum incoming transfer size of the described node.

149.33.5.7 public int getMaxOutgoingTransferSize()

□ Returns the maximum outgoing transfer size of the described node.

Returns the maximum outgoing transfer size of the described node.

149.33.5.8 public ZigBeeServerMask getServerMask()

□ Returns the server mask of the described node.

Returns the server mask of the described node.

149.33.5.9 **public boolean isComplexDescriptorAvailable()**

□ Checks if a complex descriptor is available.

Returns true if a complex descriptor is available or false otherwise.

149.33.5.10 **public boolean isExtendedActiveEndpointListAvailable()**

□ Checks if extended active endpoint list is available.

Returns true if extended active endpoint list is available or false otherwise.

149.33.5.11 **public boolean isExtendedSimpleDescriptorListAvailable()**

□ Checks if extended simple descriptor is available.

Returns true if extended simple descriptor is available or false otherwise.

149.33.5.12 **public boolean isUserDescriptorAvailable()**

□ Checks if a user descriptor is available.

Returns true if a user descriptor is available or false otherwise.

149.33.6 public interface ZigBeePowerDescriptor

This interface represents a power descriptor as described in the ZigBee Specification.

The Power Descriptor gives a dynamic indication of the power status of the node.

No Implement Consumers of this API must not implement this interface

149.33.6.1 **public static final short CRITICAL_LEVEL = 0**

Current power source level: critical.

149.33.6.2 **public static final short FULL_LEVEL = 3**

Current power source level: 100%.

149.33.6.3 **public static final short LOW_LEVEL = 1**

Current power source level: 33%.

149.33.6.4 **public static final short MIDDLE_LEVEL = 2**

Current power source level: 66%.

149.33.6.5 **public short getCurrentPowerMode()**

□ Returns the current power mode.

Returns the current power mode.

149.33.6.6 **public short getCurrentPowerSource()**

□ Returns the current power source field of the Power Descriptor.

Returns the current power source field of the Power Descriptor.

149.33.6.7 **public short getCurrentPowerSourceLevel()**

□ Returns the current power source level.

Returns the current power source level. May be one of CRITICAL_LEVEL, LOW_LEVEL, MIDDLE_LEVEL, FULL_LEVEL.

149.33.6.8 public boolean isConstantMainsPowerAvailable()

□ Checks if constant (mains) power is available.

Returns true if constant (mains) power is available or false otherwise.

149.33.6.9 public boolean isDisposableBattery()

□ Checks if the currently selected power source is the disposable battery.

Returns `true` if the currently selected power source is the disposable battery.

149.33.6.10 public boolean isDisposableBatteryAvailable()

□ Checks if a disposable battery is available.

Returns true if a disposable battery is available or false otherwise.

149.33.6.11 public boolean isMainsPower()

□ Checks if the currently selected power source is the mains power.

Returns `true` if the currently selected power source is the mains power.

149.33.6.12 public boolean isOnWhenStimulated()

□ Checks if the receiver is on when the device is simulated.

Returns `true` if the Current Power Mode field tells that the receiver is on when the device is stimulated by pressing a button, for instance.

149.33.6.13 public boolean isPeriodicallyOn()

□ Checks if the Current Power Mode field is periodically on.

Returns `true` if the Current Power Mode field is periodically on.

149.33.6.14 public boolean isRechargableBattery()

□ Checks if the currently selected power source is the rechargeable battery.

Returns `true` if the currently selected power source is the rechargeable battery.

149.33.6.15 public boolean isRechargableBatteryAvailable()

□ Checks if a rechargeable battery is available.

Returns true if a rechargeable battery is available or false otherwise.

149.33.6.16 public boolean isSyncronizedWithOnIdle()

□ Checks if synchronized with the receiver on-when-idle subfield of the node descriptor.

Returns `true` if the Current Power Mode field is synchronized on idle.

149.33.7 public interface ZigBeeServerMask

Represents the ZigBee Server Mask field of the ZigBee Node Descriptor.

No Implement Consumers of this API must not implement this interface

149.33.7.1 public boolean isBackupBindingTableCache()

□ Checks if the server is a Backup Binding Table Cache.

Returns `true` if and only if the server is a Backup Binding Table Cache.

149.33.7.2 public boolean isBackupDiscoveryCache()

□ Checks if the server is a Backup Discovery Cache.

Returns true if and only if the server is a Backup Discovery Cache.

149.33.7.3 public boolean isBackupTrustCenter()

□ Checks if the server is a Backup Trust Center.

Returns true if and only if the server is a Backup Trust Center.

149.33.7.4 public boolean isNetworkManager()

□ Checks if the server is a Network Manager.

Returns true if and only if the server is a Network Manager.

149.33.7.5 public boolean isPrimaryBindingTableCache()

□ Checks if the server is a Primary Binding Table Cache.

Returns true if and only if the server is a Primary Binding Table Cache.

149.33.7.6 public boolean isPrimaryDiscoveryCache()

□ Checks if the server is a Primary Discovery Cache.

Returns true if and only if the server is a Primary Discovery Cache.

149.33.7.7 public boolean isPrimaryTrustCenter()

□ Checks if the server is a Primary Trust Center.

Returns true if and only if the server is a Primary Trust Center.

149.33.8 public interface ZigBeeSimpleDescriptor

This interface represents a simple descriptor as described in the ZigBee Specification.

The Simple Descriptor contains information specific to each endpoint present in the node.

149.33.8.1 public int getApplicationDeviceId()

□ Returns the application device id as defined per profile.

Returns the application device id as defined per profile.

149.33.8.2 public byte getApplicationDeviceVersion()

□ Returns the version of the endpoint application.

Returns the version of the endpoint application.

149.33.8.3 public int getApplicationProfileId()

□ Returns the application profile id.

Returns the application profile id.

149.33.8.4 public short getEndpoint()

□ Returns the endpoint for which this descriptor is defined.

Returns the endpoint for which this descriptor is defined.

149.33.8.5 public int[] getInputClusters()

□ Returns an array of input (server) cluster identifiers.

Returns an array of input (server) cluster identifiers, returns an empty array if does not provides any input (server) clusters.

149.33.8.6 public int[] getOutputClusters()

□ Returns an array of output (client) cluster identifiers.

Returns an array of output (client) cluster identifiers, returns an empty array if does not provides any output (client) clusters.

149.33.8.7 public boolean providesInputCluster(int clusterId)

clusterId the cluster identifier.

□ Checks if this endpoint implements the given cluster id as an input cluster.

Returns true if and only if this endpoint implements the given cluster id as an input cluster.

149.33.8.8 public boolean providesOutputCluster(int clusterId)

clusterId the cluster identifier.

□ Checks if this endpoint implements the given cluster id as an output cluster.

Returns true if and only if this endpoint implements the given cluster id as an output cluster.

149.34 org.osgi.service.zigbee.types

Device Service Specification for ZigBee Technology Data Types.

Utility classes modeling the ZCL data types. Each class provides the static getInstance() method for retrieving a singleton instance of the class itself.

Every class contains methods for getting information about the data type like its ID and name. It is also possible to know if the data type is analog or digital or get the Java class it is mapped in.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. This package has two types of users: the consumers that use the API in this package and the providers that implement the API in this package.

Example import for consumers using the API in this package:

`Import-Package: org.osgi.service.zigbee.types; version="[1.0,2.0)"`

Example import for providers implementing the API in this package:

`Import-Package: org.osgi.service.zigbee.types; version="[1.0,1.1)"`

See Also org.osgi.service.zigbee.descriptions.ZCLDataTypeDescription

149.34.1 Summary

- `ZigBeeArray` - A singleton class that represents the 'Array' data type, as it is defined in the ZigBee Cluster Library specification.
- `ZigBeeAttributeID` - A singleton class that represents the 'Attribute ID' data type, as it is defined in the ZigBee Cluster Library specification.
- `ZigBeeBACnet` - A singleton class that represents the 'Unsigned Integer 32-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- `ZigBeeBag` - A singleton class that represents the 'Bag' data type, as it is defined in the ZigBee Cluster Library specification.
- `ZigBeeBitmap16` - A singleton class that represents the 'Bitmap 16-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- `ZigBeeBitmap24` - A singleton class that represents the 'Bitmap 24-bit' data type, as it is defined in the ZigBee Cluster Library specification.

- ZigBeeBitmap32 - A singleton class that represents the 'Bitmap 32-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeBitmap40 - A singleton class that represents the 'Bitmap 40-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeBitmap48 - A singleton class that represents the 'Bitmap 48-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeBitmap56 - A singleton class that represents the 'Bitmap 56-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeBitmap64 - A singleton class that represents the 'Bitmap 64-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeBitmap8 - A singleton class that represents the 'Bitmap 8-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeBoolean - A singleton class that represents the 'Boolean' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeCharacterString - A singleton class that represents the 'Character String' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeClusterID - A singleton class that represents the 'Cluster ID' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeDate - A singleton class that represents the 'Date' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeEnumeration16 - A singleton class that represents the 'Enumeration 16-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeEnumeration8 - A singleton class that represents the 'Enumeration 8-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeFloatingDouble - A singleton class that represents the 'Floating Double' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeFloatingSemi - A singleton class that represents the 'Floating Semi' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeFloatingSingle - A singleton class that represents the 'Floating Single' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeGeneralData16 - A singleton class that represents the 'General Data 16-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeGeneralData24 - A singleton class that represents the 'General Data 24-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeGeneralData32 - A singleton class that represents the 'General Data 32-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeGeneralData40 - A singleton class that represents the 'General Data 40-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeGeneralData48 - A singleton class that represents the 'General Data 48-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeGeneralData56 - A singleton class that represents the 'General Data 56-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeGeneralData64 - A singleton class that represents the 'General Data 64-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeGeneralData8 - A singleton class that represents the 'General Data 8-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeIEEE_ADDRESS - A singleton class that represents the 'IEEE ADDRESS' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeLongCharacterString - A singleton class that represents the 'Long Character String' data type, as it is defined in the ZigBee Cluster Library specification.

- ZigBeeLongOctetString - A singleton class that represents the 'Long Octet String' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeOctetString - A singleton class that represents the 'Octet String' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeSecurityKey128 - A singleton class that represents the 'Security Key 128' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeSet - A singleton class that represents the 'Set' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeSignedInteger16 - A singleton class that represents the 'Signed Integer 16-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeSignedInteger24 - A singleton class that represents the 'Signed Integer 24-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeSignedInteger32 - A singleton class that represents the 'Signed Integer 32-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeSignedInteger40 - A singleton class that represents the 'Signed Integer 40-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeSignedInteger48 - A singleton class that represents the 'Signed Integer 48-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeSignedInteger56 - A singleton class that represents the 'Signed Integer 56-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeSignedInteger64 - A singleton class that represents the 'Signed Integer 64-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeSignedInteger8 - A singleton class that represents the 'Signed Integer 8-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeStructure - A singleton class that represents the 'Structure' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeTimeOfDay - A singleton class that represents the 'Time Of Day' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeUnsignedInteger16 - A singleton class that represents the 'Unsigned Integer 16-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeUnsignedInteger24 - A singleton class that represents the 'Unsigned Integer 24-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeUnsignedInteger32 - A singleton class that represents the 'Unsigned Integer 32-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeUnsignedInteger40 - A singleton class that represents the 'Unsigned Integer 40-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeUnsignedInteger48 - A singleton class that represents the 'Unsigned Integer 48-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeUnsignedInteger56 - A singleton class that represents the 'Unsigned Integer 56-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeUnsignedInteger64 - A singleton class that represents the 'Unsigned Integer 64-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeUnsignedInteger8 - A singleton class that represents the 'Unsigned Integer 8-bit' data type, as it is defined in the ZigBee Cluster Library specification.
- ZigBeeUTCTime - A singleton class that represents the 'UTC Time' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.2 public class ZigBeeArray implements ZCLDataTypeDescription

A singleton class that represents the 'Array' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.2.1 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.2.2 public static ZigBeeArray getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.2.3 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.2.4 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.2.5 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.3 public class ZigBeeAttributeID implements ZCLSimpleTypeDescription

A singleton class that represents the 'Attribute ID' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.3.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns `null` if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws `IOException`– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.3.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.3.3 public static ZigBeeAttributeID getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.3.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.3.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.3.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.3.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be `null`. If `null` a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a `null` value as the request to serialize the so called *Invalid Value*.

Throws `IOException`– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.4 public class ZigBeeBACnet implements ZCLSimpleTypeDescription

A singleton class that represents the 'Unsigned Integer 32-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.4.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns `null` if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws `IOException`– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.4.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.4.3 public static ZigBeeBACnet getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.4.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.4.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.4.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.4.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be `null`. If `null` a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a `null` value as the request to serialize the so called *Invalid Value*.

Throws `IOException`– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.5 public class ZigBeeBag implements ZCLDataTypeDescription

A singleton class that represents the 'Bag' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.5.1 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.5.2 public static ZigBeeBag getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.5.3 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.5.4 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.5.5 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.6 public class ZigBeeBitmap16 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Bitmap 16-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.6.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.6.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.6.3 public static ZigBeeBitmap16 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.6.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.6.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.6.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.6.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.7 public class ZigBeeBitmap24 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Bitmap 24-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.7.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.7.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.7.3 public static ZigBeeBitmap24 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.7.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.7.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.7.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.7.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.8 public class ZigBeeBitmap32 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Bitmap 32-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.8.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.8.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.8.3 public static ZigBeeBitmap32 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.8.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.8.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.8.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.8.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.9 public class ZigBeeBitmap40 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Bitmap 40-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.9.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.9.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.9.3 public static ZigBeeBitmap40 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.9.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.9.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.9.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.9.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.10 public class ZigBeeBitmap48 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Bitmap 48-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.10.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.10.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.10.3 public static ZigBeeBitmap48 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.10.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.10.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.10.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.10.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.11 public class ZigBeeBitmap56 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Bitmap 56-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.11.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.11.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.11.3 public static ZigBeeBitmap56 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.11.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.11.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.11.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.11.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.12 public class ZigBeeBitmap64 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Bitmap 64-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.12.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.12.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.12.3 public static ZigBeeBitmap64 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.12.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.12.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.12.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.12.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.13 public class ZigBeeBitmap8 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Bitmap 8-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.13.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.13.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.13.3 public static ZigBeeBitmap8 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.13.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.13.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.13.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.13.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.14 public class ZigBeeBoolean implements ZCLSimpleTypeDescription

A singleton class that represents the 'Boolean' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.14.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.14.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.14.3 public static ZigBeeBoolean getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.14.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.14.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.14.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.14.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.15 public class ZigBeeCharacterString implements ZCLSimpleTypeDescription

A singleton class that represents the 'Character String' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.15.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.15.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.15.3 public static ZigBeeCharacterString getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.15.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.15.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.15.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.15.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.16 public class ZigBeeClusterID implements ZCLSimpleTypeDescription

A singleton class that represents the 'Cluster ID' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.16.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.16.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.16.3 public static ZigBeeClusterID getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.16.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.16.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.16.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.16.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.17 public class ZigBeeDate implements ZCLSimpleTypeDescription

A singleton class that represents the 'Date' data type, as it is defined in the ZigBee Cluster Library specification.

The ZigBee data type is mapped to a byte[4] array where byte[0] must contain the Year field (be careful that in the ZCL specification this byte do not contain the actual year, but an offset) whereas byte[3] the Day of Week. The array is marshaled/unmarshaled starting from byte[0].

149.34.17.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.17.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.17.3 public static ZigBeeDate getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.17.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.17.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.17.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.17.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.18 public class ZigBeeEnumeration16 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Enumeration 16-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.18.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.18.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.18.3 public static ZigBeeEnumeration16 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.18.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.18.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.18.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.18.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.19 public class ZigBeeEnumeration8 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Enumeration 8-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.19.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.19.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.19.3 public static ZigBeeEnumeration8 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.19.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.19.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.19.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.19.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.20 public class ZigBeeFloatingDouble implements ZCLSimpleTypeDescription

A singleton class that represents the 'Floating Double' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.20.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.20.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.20.3 public static ZigBeeFloatingDouble getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.20.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.20.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.20.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.20.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.21 public class ZigBeeFloatingSemi implements ZCLSimpleTypeDescription

A singleton class that represents the 'Floating Semi' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.21.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.21.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.21.3 public static ZigBeeFloatingSemi getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.21.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.21.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.21.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.21.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.22 public class ZigBeeFloatingSingle implements ZCLSimpleTypeDescription

A singleton class that represents the 'Floating Single' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.22.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.22.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.22.3 public static ZigBeeFloatingSingle getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.22.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.22.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.22.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.22.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.23 public class ZigBeeGeneralData16 implements ZCLSimpleTypeDescription

A singleton class that represents the 'General Data 16-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.23.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.23.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.23.3 public static ZigBeeGeneralData16 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.23.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.23.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.23.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.23.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.24 public class ZigBeeGeneralData24 implements ZCLSimpleTypeDescription

A singleton class that represents the 'General Data 24-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.24.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.24.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.24.3 public static ZigBeeGeneralData24 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.24.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.24.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.24.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.24.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.25 public class ZigBeeGeneralData32 implements ZCLSimpleTypeDescription

A singleton class that represents the 'General Data 32-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.25.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.25.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.25.3 public static ZigBeeGeneralData32 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.25.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.25.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.25.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.25.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.26 public class ZigBeeGeneralData40 implements ZCLSimpleTypeDescription

A singleton class that represents the 'General Data 40-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.26.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.26.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.26.3 public static ZigBeeGeneralData40 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.26.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.26.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.26.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.26.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.27 public class ZigBeeGeneralData48 implements ZCLSimpleTypeDescription

A singleton class that represents the 'General Data 48-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.27.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.27.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.27.3 public static ZigBeeGeneralData48 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.27.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.27.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.27.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.27.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.28 public class ZigBeeGeneralData56 implements ZCLSimpleTypeDescription

A singleton class that represents the 'General Data 56-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.28.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.28.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.28.3 public static ZigBeeGeneralData56 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.28.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.28.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.28.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.28.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.29 public class ZigBeeGeneralData64 implements ZCLSimpleTypeDescription

A singleton class that represents the 'General Data 64-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.29.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.29.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.29.3 public static ZigBeeGeneralData64 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.29.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.29.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.29.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.29.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.30 public class ZigBeeGeneralData8 implements ZCLSimpleTypeDescription

A singleton class that represents the 'General Data 8-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.30.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.30.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.30.3 public static ZigBeeGeneralData8 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.30.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.30.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.30.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.30.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.31 public class ZigBeeIEEE_ADDRESS implements ZCLSimpleTypeDescription

A singleton class that represents the 'IEEE ADDRESS' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.31.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.31.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.31.3 public static ZigBeeIEEE_ADDRESS getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.31.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.31.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.31.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.31.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.32 public class ZigBeeLongCharacterString implements ZCLSimpleTypeDescription

A singleton class that represents the 'Long Character String' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.32.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.32.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.32.3 public static ZigBeeLongCharacterString getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.32.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.32.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.32.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.32.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.33 public class ZigBeeLongOctetString implements ZCLSimpleTypeDescription

A singleton class that represents the 'Long Octet String' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.33.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.33.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.33.3 public static ZigBeeLongOctetString getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.33.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.33.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.33.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.33.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a `null` value as the request to serialize the so called *Invalid Value*.

Throws `IOException`– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.34 public class ZigBeeOctetString implements ZCLSimpleTypeDescription

A singleton class that represents the 'Octet String' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.34.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns `null` if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws `IOException`– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.34.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.34.3 public static ZigBeeOctetString getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.34.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.34.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.34.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.34.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be `null`. If `null` a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a `null` value as the request to serialize the so called *Invalid Value*.

Throws `IOException`– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.35 public class ZigBeeSecurityKey128 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Security Key 128' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.35.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns `null` if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws `IOException`– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.35.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.35.3 public static ZigBeeSecurityKey128 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.35.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.35.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.35.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.35.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be `null`. If `null` a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.36 public class ZigBeeSet implements ZCLDataTypeDescription

A singleton class that represents the 'Set' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.36.1 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.36.2 public static ZigBeeSet getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.36.3 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.36.4 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.36.5 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.37 public class ZigBeeSignedInteger16 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Signed Integer 16-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.37.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.37.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.37.3 public static ZigBeeSignedInteger16 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.37.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.37.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.37.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.37.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.38 public class ZigBeeSignedInteger24 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Signed Integer 24-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.38.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.38.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.38.3 public static ZigBeeSignedInteger24 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.38.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.38.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.38.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.38.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.39 public class ZigBeeSignedInteger32 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Signed Integer 32-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.39.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.39.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.39.3 public static ZigBeeSignedInteger32 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.39.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.39.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.39.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.39.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value.*

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.40 public class ZigBeeSignedInteger40 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Signed Integer 40-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.40.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.40.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.40.3 public static ZigBeeSignedInteger40 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.40.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.40.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.40.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.40.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.41 public class ZigBeeSignedInteger48 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Signed Integer 48-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.41.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.41.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.41.3 public static ZigBeeSignedInteger48 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.41.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.41.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.41.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.41.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.42 public class ZigBeeSignedInteger56 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Signed Integer 56-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.42.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.42.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.42.3 public static ZigBeeSignedInteger56 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.42.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.42.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.42.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.42.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.43 public class ZigBeeSignedInteger64 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Signed Integer 64-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.43.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.43.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.43.3 public static ZigBeeSignedInteger64 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.43.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.43.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.43.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.43.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.44 public class ZigBeeSignedInteger8 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Signed Integer 8-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.44.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.44.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.44.3 public static ZigBeeSignedInteger8 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.44.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.44.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.44.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.44.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.45 public class ZigBeeStructure implements ZCLDataTypeDescription

A singleton class that represents the 'Structure' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.45.1 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.45.2 public static ZigBeeStructure getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.45.3 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.45.4 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.45.5 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.46 public class ZigBeeTimeOfDay implements ZCLSimpleTypeDescription

A singleton class that represents the 'Time Of Day' data type, as it is defined in the ZigBee Cluster Library specification. The ZigBee data type is mapped to a byte[4] array where byte[0] must contain the Hour field and byte[3] the Hundredths of seconds. The array is marshaled/unmarshaled starting from byte[0].

149.34.46.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns `null` if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws `IOException`– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.46.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.46.3 public static ZigBeeTimeOfDay getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.46.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.46.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.46.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.46.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.47 public class ZigBeeUnsignedInteger16 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Unsigned Integer 16-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.47.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.47.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.47.3 public static ZigBeeUnsignedInteger16 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.47.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.47.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.47.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.47.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.48 public class ZigBeeUnsignedInteger24 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Unsigned Integer 24-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.48.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.48.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.48.3 public static ZigBeeUnsignedInteger24 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.48.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.48.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.48.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.48.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.49 public class ZigBeeUnsignedInteger32 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Unsigned Integer 32-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.49.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.49.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.49.3 public static ZigBeeUnsignedInteger32 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.49.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.49.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.49.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.49.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.50 public class ZigBeeUnsignedInteger40 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Unsigned Integer 40-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.50.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.50.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.50.3 public static ZigBeeUnsignedInteger40 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.50.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.50.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.50.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.50.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.51 public class ZigBeeUnsignedInteger48 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Unsigned Integer 48-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.51.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.51.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.51.3 public static ZigBeeUnsignedInteger48 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.51.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.51.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.51.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.51.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.52 public class ZigBeeUnsignedInteger56 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Unsigned Integer 56-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.52.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.52.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.52.3 public static ZigBeeUnsignedInteger56 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.52.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.52.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.52.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.52.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.53 public class ZigBeeUnsignedInteger64 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Unsigned Integer 64-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.53.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.53.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.53.3 public static ZigBeeUnsignedInteger64 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.53.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.53.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.53.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.53.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.54 public class ZigBeeUnsignedInteger8 implements ZCLSimpleTypeDescription

A singleton class that represents the 'Unsigned Integer 8-bit' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.54.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.54.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.54.3 public static ZigBeeUnsignedInteger8 getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.54.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.54.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.54.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.54.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws IOException– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.34.55 public class ZigBeeUTCTime implements ZCLSimpleTypeDescription

A singleton class that represents the 'UTC Time' data type, as it is defined in the ZigBee Cluster Library specification.

149.34.55.1 public Object deserialize(ZigBeeDataInput is) throws IOException

is the ZigBeeDataInput from where the value of data type is read from.

□ Deserializes a value from the passed ZigBeeDataInput stream.

Returns An object that represents the deserialized value of data. Returns null if the read value represents the *Invalid Value* for the specific ZigBee data type.

Throws IOException– If an I/O error occurs while reading the ZigBeeDataInput. An EOFException is thrown if the data input stream end is reached while deserializing the data type.

149.34.55.2 public short getId()

□ Returns the data type identifier.

Returns the data type identifier. The data types identifiers supported by this specification are defined in the ZigBeeDataTypes interface.

149.34.55.3 public static ZigBeeUTCTime getInstance()

□ Gets a singleton instance of this class.

Returns the singleton instance.

149.34.55.4 public Class getJavaDataType()

□ Returns the corresponding Java type class.

Returns the corresponding Java type class.

149.34.55.5 public String getName()

□ Returns the associated data type name.

Returns the associated data type name string.

149.34.55.6 public boolean isAnalog()

□ Checks if the data type is analog.

Returns true, if the data type is Analog, otherwise is Discrete.

149.34.55.7 public void serialize(ZigBeeDataOutput os,Object value) throws IOException

os a ZigBeeDataOutput stream where to the passed value will be appended. This parameter cannot be null. If null a NullPointerException must be thrown.

value The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the data type do not allow any invalid value and the passed value is null an IllegalArgumentException is thrown.

□ Serializes a ZigBee data type into a ZigBeeDataOutput stream. An implementation of this method must throw an IllegalArgumentException if the passed value does not belong to the expected class or its value exceeds the possible values allowed (in terms of range or length).

An implementation of this method must interpret (where it makes sense) a null value as the request to serialize the so called *Invalid Value*.

Throws `IOException`– If an I/O error occurs while writing on the ZigBeeDataOutput. The EOFException may be thrown if there is no more space on the data output for serializing the passed value.

149.35 References

[1] *ZigBee Specification*
Document 053474r17, ZigBee Alliance, October 19, 2007.

[2] *ZigBee Cluster Library Specification*
Document 075123r04ZB, ZigBee Alliance, May 29, 2012.

[3] *Pervasive Service Composition in the Home Network*
André Bottaro, Anne Gérodolle, Philippe Lalanda, 21st IEEE International Conference on Advanced Information Networking and Applications (AINA-07), Niagara Falls, Canada, May 2007.

[4] *Device and Service Discovery in Home Networks with OSGi*
Pavlin Dobrev, David Famolari, Christian Kurzke, Brent A. Miller, IEEE Communications magazine, Volume 40, Issue 8, pp. 86-92, August 2002.

[5] *ASHRAE 135-2004 Standard*
Data Communication Protocol for Building Automation and Control Networks.

[6] *Listeners considered harmful: The whiteboard pattern*
Peter Kriens, BJ Hargrave for the OSGi Alliance, Technical Whitepaper, August 2004.
https://www.osgi.org/wp-content/uploads/whiteboard1.pdf

[7] *ZigBee Gateway*
ZigBee Alliance, 2011.

End Of Document