



ZigBee Device Service Specification

Draft

310 Pages

Abstract

This specification defines the Java API to discover, control and implement ZigBee devices on the OSGi platform and according to OSGi service design patterns. This API maps the representation of ZigBee entities defined by ZigBee Cluster Library into Java classes. OSGi service design patterns are used on the one hand for dynamic discovery, control and eventing of local and networked devices and on the other hand for dynamic network advertising and control of local OSGi services implementing this API.

0 Document Information

License

DISTRIBUTION AND FEEDBACK LICENSE, Version 2.0

The OSGi Alliance hereby grants you a limited copyright license to copy and display this document (the "Distribution") in any medium without fee or royalty. This Distribution license is exclusively for the purpose of reviewing and providing feedback to the OSGi Alliance. You agree not to modify the Distribution in any way and further agree to not participate in any way in the making of derivative works thereof, other than as a necessary result of reviewing and providing feedback to the Distribution. You also agree to cause this notice, along with the accompanying consent, to be included on all copies (or portions thereof) of the Distribution. The OSGi Alliance also grants you a perpetual, non-exclusive, worldwide, fully paid-up, royalty free, limited license (without the right to sublicense) under any applicable copyrights, to create and/or distribute an implementation of the Distribution that: (i) fully implements the Distribution including all its required interfaces and functionality; (ii) does not modify, subset, superset or otherwise extend the OSGi Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the OSGi Name Space other than those required and authorized by the Distribution. An implementation that does not satisfy limitations (i)-(ii) is not considered an implementation of the Distribution, does not receive the benefits of this license, and must not be described as an implementation of the Distribution. "OSGi Name Space" shall mean the public class or interface declarations whose names begin with "org.osgi" or any recognized successors or replacements thereof. The OSGi Alliance expressly reserves all rights not granted pursuant to these limited copyright licenses including termination of the license at will at any time.

EXCEPT FOR THE LIMITED COPYRIGHT LICENSES GRANTED ABOVE, THE OSGi ALLIANCE DOES NOT GRANT, EITHER EXPRESSLY OR IMPLIEDLY, A LICENSE TO ANY INTELLECTUAL PROPERTY IT, OR ANY THIRD PARTIES, OWN OR CONTROL. Title to the copyright in the Distribution will at all times remain with the OSGi Alliance. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted therein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

THE DISTRIBUTION IS PROVIDED "AS IS," AND THE OSGi ALLIANCE (INCLUDING ANY THIRD PARTIES THAT HAVE CONTRIBUTED TO THE DISTRIBUTION) MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DISTRIBUTION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

NEITHER THE OSGi ALLIANCE NOR ANY THIRD PARTY WILL BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE DISTRIBUTION.

Implementation of certain elements of this Distribution may be subject to third party intellectual property rights, including without limitation, patent rights (such a third party may or may not be a member of the OSGi Alliance). The OSGi Alliance is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

The Distribution is a draft. As a result, the final product may change substantially by the time of final publication, and you are cautioned against relying on the content of this Distribution. You are encouraged to update any implementation of the Distribution if and when such Distribution becomes a final specification.

The OSGi Alliance is willing to receive input, suggestions and other feedback (“Feedback”) on the Distribution. By providing such Feedback to the OSGi Alliance, you grant to the OSGi Alliance and all its Members a non-exclusive, non-transferable, worldwide, perpetual, irrevocable, royalty-free copyright license to copy, publish, license, modify, sublicense or otherwise distribute and exploit your Feedback for any purpose. Likewise, if incorporation of your Feedback would cause an implementation of the Distribution, including as it may be modified, amended, or published at any point in the future (“Future Specification”), to necessarily infringe a patent or patent application that you own or control, you hereby commit to grant to all implementers of such Distribution or Future Specification an irrevocable, worldwide, sublicenseable, royalty free license under such patent or patent application to make, have made, use, sell, offer for sale, import and export products or services that implement such Distribution or Future Specification. You warrant that (a) to the best of your knowledge you have the right to provide this Feedback, and if you are providing Feedback on behalf of a company, you have the rights to provide Feedback on behalf of your company; (b) the Feedback is not confidential to you and does not violate the copyright or trade secret interests of another; and (c) to the best of your knowledge, use of the Feedback would not cause an implementation of the Distribution or a Future Specification to necessarily infringe any third-party patent or patent application known to you. You also acknowledge that the OSGi Alliance is not required to incorporate your Feedback into any version of the Distribution or a Future Specification.

I HEREBY ACKNOWLEDGE AND AGREE TO THE TERMS AND CONDITIONS DELINEATED ABOVE.

Trademarks

OSGi™ is a trademark, registered trademark, or service mark of the OSGi Alliance in the US and other countries. Java is a trademark, registered trademark, or service mark of Oracle Corporation in the US and other countries. All other trademarks, registered trademarks, or service marks used in this document are the property of their respective owners and are hereby recognized.

Feedback

This document can be downloaded from the OSGi Alliance design repository at <https://github.com/osgi/design>. The public can provide feedback about this document by opening a bug at <https://www.osgi.org/bugzilla/>.

Table of Contents

0 Document Information.....	2
License.....	2
Trademarks.....	3
Feedback.....	3
Table of Contents.....	3
Terminology and Document Conventions.....	4
Revision History.....	5
 1 Introduction.....	 19
 2 Application Domain.....	 19
System Architecture.....	19
ZigBee Stack.....	20
Application Profiles and ZigBee Cluster Library (ZCL).....	21
 3 Problem Description.....	 22

4 Requirements.....	22
5 Technical Solution.....	23
Essentials.....	23
Entities.....	23
ZigBee Base Driver.....	26
ZigBee Node.....	27
ZigBee Endpoint.....	29
ZigBee Device Description.....	30
ZigBee Device Description Set.....	30
ZCL Cluster.....	31
ZCL Cluster Description.....	31
ZCL Global Cluster Description.....	31
ZigBee Command Description.....	31
ZigBee Attribute.....	32
ZigBee Attribute Description.....	32
ZCL Data Type Description.....	32
ZigBee Attribute Record.....	32
ZigBee Handler.....	32
ZigBee Data Types.....	32
Working With a ZigBee Endpoint.....	35
Implementing a ZigBee Endpoint.....	35
Event API.....	36
ZCL Exception.....	37
ZDP Exception.....	38
ZCL Frame.....	38
ZigBee Group.....	38
ZigBee Networking.....	39
Security.....	40
6 Javadoc.....	40
7 Considered Alternatives.....	306
Which entity has to be registered in the service registry? The ZigBeeEndpoint object and/or the ZigBeeNode object?.....	306
Why having startNetwork() and permitJoin(short duration)? (And not rely on bundle API). .	307
Configure reporting and the White Board Pattern.....	307
8 Security Considerations.....	307
9 Document Support.....	308
References.....	308
Author's Address.....	308
Acronyms and Abbreviations.....	310
End of Document.....	310

Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in 8.

Source code is shown in this typeface.

Revision History

The last named individual in this history is currently responsible for this document.

Revision	Date	Comments
Initial	May, 16 th , 2012	Andre Bottaro, Orange, andre.bottaro@orange.com
1 st Draft	September, 20 th , 2012	Bâle presentation
	October, 16 th , 2012	API Summary Initialized
	October, 18 th , 2012	ZigBeeClusterDescription and ZigBeeCommandDescription section initialized
	December, 14 th , 2012	Added details and references, cleared comments, fixed few mistakes.
v11-reg	January, 15 th	<p>Andre Bottaro, Orange Jean-Pierre Poutcheu, Orange</p> <ol style="list-style-type: none"> 1. ZigBeeDeviceDescription and ZigBeeDeviceDescriptionSet classes are added and the registration of device descriptions is explained. The base driver and any bundle are now able to register the set of ZigBeeDeviceDescription objects which they have the knowledge. Those sets are registered with ZigBeeDeviceDescriptionSet interface. 2. ZigBeeEvent.getCluster() added in order to be able to retrieve the ids of the devicenode, the endpoint and the cluster which attributes values are notified. 3. Masaki 1st point: ZigBeeEndpoint now provides getDeviceNode() method in ZigBeeEndpoint class without any input argument. 4. Masaki 2nd point: ZigBeeEndpoint class now provides a method to retrieve all available input ZigBeeCluster objects and a method to retrieve all output ones. 5. Whenever getXXXid() can be changed into getId() without ambiguity, the change is made. The same change is applied to getXXXName, getXXXVersion(). For instance, ZigBeeCluster.getClusterId() is changed into ZigBeeCluster.getId(). 6. ZigBeeDataType.getJavaDataType() has now the same signature as UpnPDataType.getJavaDataType(). 7. ZigBeeHost.getPanId() is removed and the method is added to the parent class: ZigBeeDeviceNode.getPanId() is added. 8. PAN_ID property was a property only specified for exported ZigBeeEndpoint services. It is now specified for all ZigBeeEndpoint services. Other properties are added to improve filtering features made on ZigBeeEndpoint services. 9. An Endpoint was able to be registered once and exported on several networks by distinct hosts. This lead to an issue: which host to return in ZigBeeEndpoint.getDeviceNode() method? Thus, the spec has been changed: a distinct ZigBeeEndpoint object has now to be created and registered for every distinct targeted network (identified by a distinct PAN_ID)

Revision	Date	Comments
v12-reg	January, 29 th	<p>Andre Bottaro, Orange Jean-Pierre Poutcheu, Orange</p> <ol style="list-style-type: none"> 1. Typed collections (Java 1.5) may remain in the javadoc. That's a bug and they are removed. (javadoc to be sent to the list later). 2. The link between ZigBeeDeviceDescription and ZigBeeClusterDescription was missing in the UML schema. It is now added. 3. Masaki's 4th point: Permit duration taken into account. 4. Standard properties are proposed for the ZigBeeDeviceDescriptionSet service. The right mapping with ZigBee standard names and the format of values is now applied. 5. The list of constant ZigBeeDataTypeDescription objects was missing. The developer needs to be able to retrieve those ZigBee constant objects. It is now specified in a new interface named "ZigBeeDataTypes". 6. Nicola's point on de/serialization of data types. isAnalog(), serialize/deserialize() method names taken into account. 7. Cardinality 0..1 is replaced by * when it involves a table or a vector of objects (attributedescs, clusterdescs, ...). 8. Masaki 3rd point: a method « void ZigBeeEndPoint.notExported(ZigBeeException ze) » is added. Explanations are now in the Export section. 9. The ZigBeeDeviceDescriptionSet class was missing in the UML schema. It is now added. (and the 'ZigBee Cluster Descriptor' implementation (grey blox) is removed).
v13-reg	February, 5 th	<p>Andre Bottaro, Orange Jean-Pierre Poutcheu, Orange</p> <ol style="list-style-type: none"> 1. The ZigBee Extended PAN ID is now mentioned and used in the specification.

Revision	Date	Comments
v14-reg	February, 25 th	<p>Andre Bottaro, Orange Jean-Pierre Poutcheu, Orange Thanks to Evgeni Grigorov's (Prosyst's) comments and Nicola Portinaro (Telecom Italia's) comments</p> <ol style="list-style-type: none"> 1. Added a 'leave()' method in ZigBeeDeviceNode javadoc for removing nodes to request the device to leave the network: void leave(boolean rejoin, boolean request, boolean removeChildren, ZigBeeHandler handler) 2. Added a 'checkValue(Object obj)' in ZigBeeParameterDescription which returns true if the parameter value is valid according to his description and possible value ranges and other specific information. 3. Added new filters in listener, with names closer to ZCL documentation ones ZigBeeAttribute.REPORTABLE_CHANGE, ZigBeeAttribute.MIN_REPORT_INTERVAL, ZigBeeAttribute.MAX_REPORT_INTERVAL, ZigBeeAttribute.TIMEOUT_PERIOD 4. Added a 'public void setValue(Object value, ZigBeeHandler handler) throws ZigBeeException' method in ZigBeeAttribute 5. Added a description in 'Implementing a ZigBee Endpoint' about the use case where, an exportable endpoint corresponds to two more than 1 ZigBeeHost, at this time a ZigBeeException is thrown. 6. Added a paragraph to tell the reader that EndPoint 0 and 255 are not registered in the registry. And that EndPoint 241-255 should not be registered since these numbers are said "reserved for future use" in the ZB spec.
v20-reg	May, 6 th	<p>Jean-Pierre Poutcheu, Orange Arnaud Rinquin, Orange</p> <ol style="list-style-type: none"> 1. ZigBeeAttributeHandler,notifyResponse(), use of Map instead of dictionary 2. Moved getAccessType() and isReportable() from ZigBeeAttribute to ZigBeeAttributeDescription 3. UNSIGNED_INTEGER_64 mapped with BigInteger Java class 4. ZigBeeCluster.readAttributeAsByte(...) has been removed 5. Added get and setChannel mask operations in ZigBeeHost

Revision	Date	Comments
v21-reg	May, 15 th	<p>Jean-Pierre Poutcheu, Orange Arnaud Rinquin, Orange</p> <ol style="list-style-type: none"> Added a new Exception ZigBeeNoDescriptionAvailableException Added a new class ZigBeeAttributeRecord Modified ZigBeeCluster.writeAttributes(...) to <ul style="list-style-type: none"> void writeAttributes(boolean undivided, int[] attributesIds, byte[] values, ZigBeeAttributesHandler handler) throws ZigBeeNoDescriptionAvailableException; void writeAttributes(boolean undivided, ZigBeeAttributeRecord[] attributes, ZigBeeAttributesHandler handler) Change ZigBeeHandler.notifyResponse to notifyResponse(int Status, Map values) Nicola's Point : New package org.osgi.service.zigbee.descriptors for all the descriptors Nicola's Point: ZigBeeException use hex value are used for constants. (Thx to Nicola) Evgeni's Point: Removed ZigBeeCoordinator.getLinkKey() and ZigBeeCoordinator.getMasterKey() methods
v22-reg	May, 22 th	<p>Jean-Pierre Poutcheu, Orange Arnaud Rinquin, Orange</p> <ol style="list-style-type: none"> Explain that 'no response' command are used when handler is null in writeAttributes commands Explain that map use attribute ids as key and objects as values
v23-reg	May, 29 th	<p>Jean-Pierre Poutcheu, Orange Arnaud Rinquin, Orange</p> <ol style="list-style-type: none"> Added a new section about ZigBeeAttributeRecord class Added getInvalidNumber() method in ZigBeeDataTypeDescription
v24-reg	June, 5 th	<p>Jean-Pierre Poutcheu, Orange Arnaud Rinquin, Orange</p> <ol style="list-style-type: none"> Moved getSimpleDescriptor() from ZigBee Node section to Endpoint section Changed getInputCluster()/getOutputCluster() by getServerCluster()/getClientCluster()

Revision	Date	Comments
v25-reg	June, 12 th	<p>Jean-Pierre Poutcheu, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Renamed ZigBeeDeviceNode interface by ZigBeeNode 2. Updated 'Operation Summary' section to take into account the registration of ZigBeeNode as an OSGi service by the base driver. 3. In ZigBeeEndpoint, getDeviceNode() replaced by getNodeAddress(), which returns the node IEEE Address 4. In ZigBeeNode interface, static field ID replaced by IEEE_ADDRESS 5. Updated figure 6.2: 'Device Node' → 'Node'
v26-reg	June, 19 th	<p>Jean-Pierre Poutcheu, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Updated figure 6.1: 'ZigBeeDeviceNode' → 'ZigBeeNode' 2. Moved refreshNetwork(ZigBeeHandler) from ZigBeeCoordinator to ZigBeeHost 3. Added start() in ZigBeeHost
v27-reg	June, 28 th	<p>Jean-Pierre Poutcheu, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Deleted all the mention about ZigBeeCoordinator 2. Updated figure 6.1: Removed ZigBeeCoordinator interface 3. More explanation in ZigBee Networking section about the role of ZigBeeHost
v28-reg	July, 3 rd	<p>Jean-Pierre Poutcheu, Orange André Bottaro, Orange</p> <p>ZigBeeHost.setOperationalMode(short) replaced by ZigBeeHost.setLogicalType(short)</p>
v29-reg	July, 17 th	<p>Jean-Pierre Poutcheu, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Deleted ZigBeeAttributesHandler interface 2. IEEE Address managed as a Long Java type
v30-reg	July, 25 th	<p>Jean-Pierre Poutcheu, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Added properties ZigBeeNode.HOST_PID and ZigBeeEndpoint.HOST_PID_TARGET 2. Updated endpoint export section to take into account HOST_PID_TARGET property when exporting an endpoint.

Revision	Date	Comments
v31-reg	August, 29 th	<p>Jean-Pierre Poutcheu, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> Added ZigBeeGlobalClusterDescription.getClusterFunctionalDomain() Added a table in ZigBeeHandler section describing Map parameter response for onSuccess(Map) and onFailure(Map)
v32-reg	September, 3 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> Fix typo, spelling, and grammar. Remove several methods duplicated from Javadoc. Enhance some sentences.
v33-reg	September, 9 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> Enhance ZigBee Handler part. Fix references. Merge import/export HOST_PID. Add some open questions as comments.
v34-reg	September, 17 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> Delete invoke(Object[] values, ZigBeeDataTypeDescription[] inputTypes, ZigBeeDataTypeDescription[] outputTypes, ZigBeeHandler handler) Add serialize, and deserialize methods to ZigBeeCommandDescription. These methods are designed to ease the use of ZigBeeCommand.invoke(byte[] bytes, ZigBeeHandler handler). Add ZigBeeHost.stop() method.

Revision	Date	Comments
v35-reg	September, 30 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. ZigBeeEvent: change Dictionary getAttributesEvents() to Object getValue() 2. Remove getDescription() from ZigBeeCluster, ZigBeeCommand, and ZigBeeAttributes. 3. In ZigBeeCluster, remove: public void readAttributesAsBytes(int[] attributesIds, ZigBeeHandler handler); and public void writeAttributes(boolean undivided, int[] attributesIds, byte[] values, ZigBeeHandler handler) throws ZigBeeNoDescriptionAvailableException; 4. Introduce ZigBeeCommandHandler 5. Rename ZigBeeException.ATTRIBUTE_NOT_SUPPORTED to UNSUPPORTED_ATTRIBUTE 6. Add constructor in ZigBeeAttributeRecord, and remove setters. 7. Remove setters/getters methods with bytes parameters in ZigBeeAttribute. 8. Introduce ZigBeeAttributesHandler. 9. ZigBee*Handler.onFailure now takes a ZigBeeException as parameter. 10. Remove getDeviceDescription() from ZigBeeEndpoint.
v36-reg	October, 2 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. ZigBeeEventListener: remove filter. 2. Event API: Specify mandatory, and optional pseudo properties for event filtering. Add ZigBeeEndpoint.ENDPOINT. 3. Add ZCL document version 4. ZigBeeEventListener: Add public void onFailure(ZigBeeException e); 5. ZigBeeEndpoint: move ENDPOINT – (zigbee.device.endpoint) to ZigBeeEndpoint.ID – (zigbee.endpoint.id); move zigbee.device.clusters.input to zigbee.endpoint.clusters.input; move zigbee.device.clusters.output to zigbee.endpoint.clusters.output. 6. ZigBeeCluster: move zigbee.listener.cluster.* to zigbee.cluster.* 7. ZigBeeNode: move zigbee.listener.node.ieee.address to zigbee.node.ieee.address 8. ZigBeeAttribute: move zigbee.listener.attribute.* to zigbee.attribute.*

Revision	Date	Comments
v37-reg	October, 7 th October, 10 th	Antonin Chazalet, Orange André Bottaro, Orange <ol style="list-style-type: none"> 1. Remove no longer needed ZigBeeNoDescriptionAvailableException 2. In the ZigBeeHandler section, Map took int as key, and now take Integer (This is a Java requirement). 3. Update ZCL document version (from 075123r01ZB to 075123r04ZB) 4. Add isPartOfAScene() method to ZigBeeAttributeDescription 5. Add a short explanation regarding ZigBeeEvent.
v38-reg	October, 18 th	Antonin Chazalet, Orange André Bottaro, Orange <ol style="list-style-type: none"> 1. ZigBeeAttributeRecord is now a final java class. 2. Remove PAN_ID and EXTENDED_PAN_ID from ZigBeeEventListener interface; the ones from ZigBeeEndpoint interface must be used. 3. Move listener static fields that are listener properties into ZigBeeEventListener: (REPORTABLE_CHANGE, MIN/MAX_INTERVAL, TIME_OUT). Keep the zigbee.attribute.prefix. 4. Event API: Specify mandatory, and optional pseudo-properties for event filtering. Add ATTRIBUTE_DATA_TYPE = "zigbee.attribute.datatype"
v39-reg	November, 4 th	Antonin Chazalet, Orange André Bottaro, Orange <ol style="list-style-type: none"> 1. Update some comments on ZigBeeEventListener's optional properties. 2. ZigBeeEndPoint: Add additional properties.
v40-reg	November, 8 th	Antonin Chazalet, Orange André Bottaro, Orange <ol style="list-style-type: none"> 1. Add EventListener.notifyTimeOut(int) 2. Update javadoc of ZigBeeAttribute.getDataType() 3. ZigBeeEventListener.ATTRIBUTE_DATA_TYPE is now mandatory. Add details on ZigBeeEventListener.MIN_REPORT_INTERVAL, MAX_REPORT_INTERVAL, and REPORTABLE_CHANGE.
v41-reg	November, 12 th	Antonin Chazalet, Orange André Bottaro, Orange <ol style="list-style-type: none"> 1. Minor enhancements. 2. Remove ZigBeeEventListener.TIMEOUT_PERIOD

Revision	Date	Comments
V42-reg	November, 22 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Add ZigBeeCommand.invoke(byte[] handler, bytes, String exportedServicePID) throws ZigBeeException. 2. Add ZigBeeEventListener, ZigBeeNode et ZigBeeDeviceDescriptionSet's properties types.
V43-reg	December, 5 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Update ZigBeeNode, and ZigBeeEndpoint.
V44-reg	December, 10 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Update filter part. 2. Update ZigBeeEventListener, ZigBeeNode et ZigBeeDeviceDescriptionSet's properties types.
V45-reg	December, 18 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Add ZigBeeEndPoint.bind(...), and unbind(...) methods. 2. Add ZigBeeEndPoint.getBoundEndPoints(...) method.
V46-reg	December, 23 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Update ZigBeeEndpoint's getBoundEndPoints method (see Java API).
V47-reg	January, 20 th	<p>André Bottaro, Orange Antonin Chazalet, Orange</p> <ol style="list-style-type: none"> 1. Update "Implementing a ZigBee Endpoint " section. 2. Clean up references section.
V48-reg	February, 7 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Add Stefano Lenzi as an author. 2. Add/update ZCLHeader, ZCLFrame, ZigBeeCluster, ZigBeeCommandDescription, and ZigBeeCommandHandler. 3. Remove ZigBeeCommand Java interface.
V49-reg	February, 14 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 4. Integrate last call decisions: Add ZigBeeGroup, and update ZigBeeCommandHandler javadoc.

Revision	Date	Comments
V50-reg	March, 3 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none">1. Rename ZigBeeCluster to ZCLCluster, ZigBeeClusterDescription to ZCLClusterDescription, and ZigBeeGlobalClusterDescription to ZCLGlobalClusterDescription.2. Remove getServerClusters(), and getServerCluster(...) from ZigBeeGroup.3. Add void invoke(ZCLFrame frame, ZigBeeCommandHandler handler) throws ZigBeeException, and void invoke(ZCLFrame frame, ZigBeeCommandHandler handler, String exportedServicePID) throws ZigBeeException in ZigBeeGroup.4. Add two broadcast methods on ZigBeeHost.5. Modify ZigBeeDataTypeDescription: serialize, and deserialize methods are now related to ZCLFrame object.6. Add getInputStream(), and getOutputStream() methods to ZCLFrame.
V51-reg	March, 4 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none">1. Update ZigBee Data Types.

Revision	Date	Comments
V52-reg	March, 10 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Replace “that occurred in the ZigBee stack.” by “that occurred in the ZigBee stack or internally by the ZigBee Base Driver or by the ZigBee network” 2. Replace “It must also expose, on the ZigBee Network, ZigBee Node services” by “It must also export, on the ZigBee Network, ZigBee Endpoint services” 3. Remove “ZigBeeNode provides getEndPoints() method which returns its associated endpoints.” 4. Rename “ZigBee hierarchy model” to “ZigBee Cluster Library model”. 5. Rephrase “Endpoint 0, also called ZDO” to “Endpoint 0, also called the ZigBee Device Object (ZDO)” 6. Rephrase “to describe the generic device capabilities,” to “for the management operations on both ZigBee node and ZigBee Endpoints”. 7. Rephrase “identifies the profile that is supported by this endpoint.” to “identifies the profile that the Endpoint belongs to. The profile can be either a ZigBee Alliance standard profile or a vendor-specific profile.”. 8. Rephrase “devices identifiers supported by the profile.” to “devices identifiers supported by the set.”. 9. Rephrase “readAttributes(int[] attributelds, ZigBeeAttributesHandler handler) – The read attributes command is generated when a device wishes to determine the value of one or more attributes located on another device.” to “The ZBD MAY (i.e. ZBD may cache request) generate the read attributes command on behalf of the OSGi application that is invoking the readAttributes method.”. 10. Rephrase “The write attributes command is generated when a device wishes to change the values of one or more attributes located on another device.” to “The ZBD generates the write attributes command on behalf of the OSGi application that is invoking the writeAttributes method.”.
V53-reg	March, 17 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Rename ZigBeeEndpoint.ID to ZigBeeEnd.ENDPOINT_ID. 2. Remove ZigBeeEndpoint.DEVICE_DESCRIPTION, and DEVICE_SERIAL. 3. Update data types.
V54-reg	March, 18 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Update ZigBeeDescriptionSet. 2. Move HOST_PID from ZigBeeNode to ZigBeeEndpoint, and update spec, and javadoc. 3. Remove HOST_PID_TARGET. 4. Replace ByteBuffer by Byte[]. 5. Update ZigBeeNode, and ZigBeeEndpoint properties.

Revision	Date	Comments
V55-reg	March, 24 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Reintroduce DEVICE_DESCRIPTION, and DEVICE_SERIAL in ZigBeeNode section in the RFC. 2. Remove no longer relevant: ZCLClusterDescriptionSet. 3. Remove org.osgi.service.zigbee sub section. 4. Update ZCLGlobalClusterDescription sentences in Entities section. 5. Remove no longer relevant ZigBeeCommand. 6. Update ZigBeeGroup. 7. Rename ZigBeeEventListener to ZCLEventListener. 8. Rename ZigBeeParameterDescription to ZCLParameterDescription. 9. Update ZCLCluster. 10. Rename ZigBeeException to ZCLException. 11. Add a ZigBeeException section (that now handles ZDP exceptions)
V56-reg	March, 28 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Replace Byte[] by byte[]. 2. ZigBeeHost.start(), and stop() now throws Exception instead of ZCLException. 3. Update "Scope"s sentences in Essentials section.
V57-reg	April, 7 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Update ZigBee Networking, and Network selection sections. 2. Update ZigBee Node section.
V58-reg	April, 14 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Update text font. 2. Update Network coordination sub-section. 3. Rename ZigBeeException by ZDPException.
V59-reg	April, 16 th	<p>André Bottaro, Orange Antonin Chazalet, Orange</p> <ol style="list-style-type: none"> 1. Updated Operation Summary and ZigBee Base Driver sections with more paragraphs and service diagrams. 2. Update fig. 3. 3. Fix a paragraph font.
V60-reg	April, 23 th	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Update fig. 3. 2. Add javadoc section.

Revision	Date	Comments
V61-reg	April, 27 th	André Bottaro, Orange Antonin Chazalet, Orange 1. Update figures with ZigBee endpoints instead of ZigBee devices. 2. Merge the document with a version with an application using directly the base driver instead of a refinement driver.
V62-reg	April, 29 th	Antonin Chazalet, Orange André Bottaro, Orange 1. Fix bug [reg] [2673] https://www.osgi.org/members/bugzilla/show_bug.cgi?id=2673
V63-reg	May, 5 th	Antonin Chazalet, Orange André Bottaro, Orange 1. Remove ZigBeeHandler's table (everything is now in the javadoc). 2. Remove ZigBee MapHandler.

1 Introduction

ZigBee [1]. is a standard wireless communication protocol designed for low-cost and low-power devices by 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 ZigBee Cluster Library [2].. The specification also describes the external API of a ZigBee Base Driver according to Device Access specification, the example made by UPnP Device Service specification and spread OSGi practices on residential market [3].[4]..

2 Application Domain

System Architecture

When installing a new ZigBee network into a residential network with a home gateway, there are 2 options. One is to add ZigBee communication capability to your home gateway with an additional hardware such as a USB device called "dongle". The other one is to replace the current home gateway with one which has ZigBee communication capability. In both cases OSGi applications call the ZigBee driver API to communicate with the ZigBee network (and its ZigBee devices) as shown in Erreur : source de la référence non trouvée.

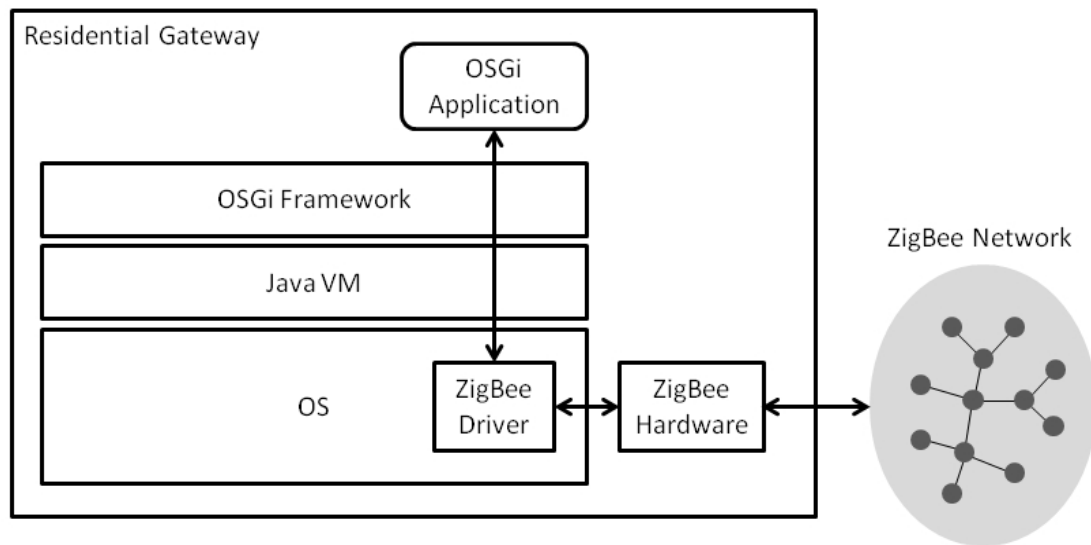


Figure 1 Communication with ZigBee devices through a ZigBee driver

The ZigBee specification defines three types of ZigBee devices: ZigBee Coordinator (ZC), ZigBee Router (ZR) and ZigBee End Device (ZED). In the above case the ZigBee hardware works as the ZigBee Coordinator and the other ZigBee devices are attached to the ZigBee network as ZigBee End Device or ZigBee Router.

- ZigBee Coordinator (ZC) is responsible for managing a ZigBee network and ZigBee devices on the network. There is one, and only one ZigBee Coordinator in each ZigBee network.
- ZigBee Router (ZR) is capable of extending a ZigBee network by relaying messages from other ZigBee devices.
- ZigBee End Device (ZED) has functionality to communicate with either ZigBee Coordinator or ZigBee Router.

ZigBee Stack

The ZigBee stack is shown in Erreur : source de la référence non trouvée. The two bottom layers, the PHY layer and the MAC layer, are defined by IEEE802.15.4 standard. The ZigBee standard defines network (NWK) layer, application (APL) layer and security layer on top of it. The NWK layer is responsible for managing the network formation and routing. The APL layer hosts application objects developed by manufacturers. The security service provider is responsible for encryption and authentication.

The application layer consists of three functional blocks: application support sub-layer, ZigBee Device Object (ZDO) and application framework. The application support sub-layer provides the transmission capability of data and management messages. The ZDO provides common functionality used by all applications. The application framework is the environment where application objects are hosted to control and manage the protocol layers.

There are two interfaces available to applications: APSDE-SAP and ZDO public interface. The APSDE-SAP

provides data transmission functionality between ZigBee devices. The ZDO public interface provides applications with management functionality such as device discovery, service discovery and network management.

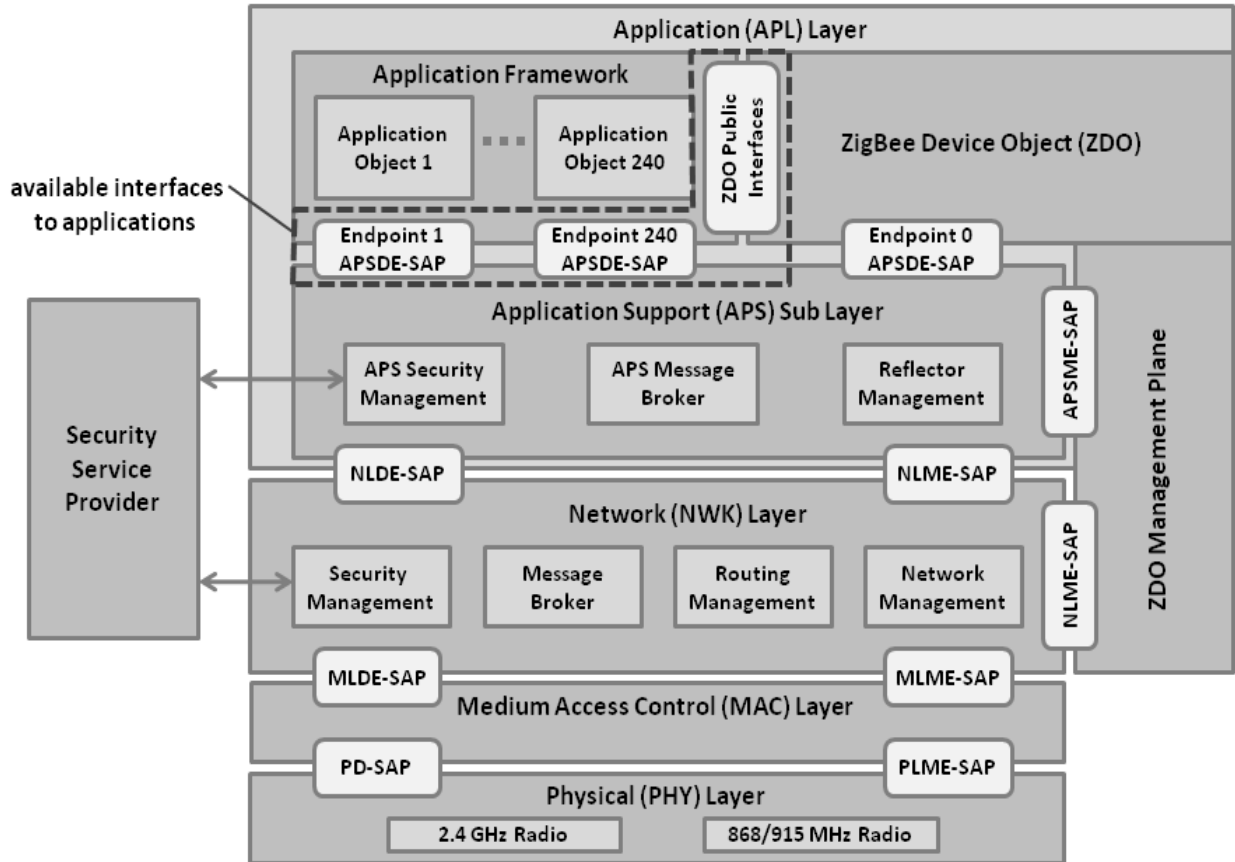


Figure 2: ZigBee Stack

Application Profiles and ZigBee Cluster Library (ZCL)

The application profiles allow interoperability between products developed by different vendors for a specific application. For example, in a light control scenario, switches developed by a vendor can turn on and turn off lights developed by another vendor if the both vendors take the same application profile. The ZigBee Alliance has defined nine public application profiles such as Home Automation (HA) and ZigBee Smart Energy (ZSE).

An application profile defines its application domain, a list of specific devices supported in the profile and a list of clusters supported by the devices. A cluster is a relevant collection of commands and attributes which together define an interface for a specific functionality. The clusters used in public application profiles are defined in the ZigBee Cluster Library (ZCL) specification. The ZCL specification defines a number of clusters and categories them into groups by their functionality.

3 Problem Description

As described in the section Erreur : source de la référence non trouvée, OSGi applications which communicate with ZigBee devices are supposed to call the API of the driver provided by the vendor. The API is proprietary and different vendor by vendor since it is not standardized in the ZigBee specification. This causes the following problems:

- 1) Application developers need to know which vendor's ZigBee hardware is used with the target residential gateway in advance before developing their applications.
- 2) An application which was developed for a certain environment may not work for other environments.

Those problems make it difficult for third parties to develop portable (OSGi) applications communicating with ZigBee devices.

The standard ZigBee API demanded in this RFP would give developers a unified way of communicating with ZigBee devices. The developers will no longer need to care about the proprietary API of drivers but will simply use the standard one.

4 Requirements

R1: The solution MUST provide an API for data transmission supported by APSDE-SAP.

R2: The solution MUST provide a base driver interface as an OSGi service for management operations supported by ZDO: device and service discovery, security management, network management, binding management, node management and group management.

R3: The solution SHOULD enable applications to trigger a re-scan of the network to refresh the registry with actual ZigBee device services.

R4: The solution MUST provide API for switching the type of the local ZigBee device among ZC, ZR and ZED.

R5: The solution MUST provide a mechanism which notifies OSGi applications of events occurred in the ZigBee network and devices.

R6: The solution MUST provide an installation capability of cluster libraries within OSGi service-oriented architecture.

R7: The solution MUST register a Device Service object representing each found ZigBee device into Service Registry and unregister the Device Service object when the ZigBee device is unavailable.

R8: The solution MAY define the driver provisioning process in accordance with the OSGi Device Access specification.

R9: The solution MUST be independent from the interface used to control the ZigBee network. The solution MUST likewise work with network controllers based on ZigBee built-in chips, ZigBee USB dongles and high level protocols offered by ZigBee Gateway Devices compliant with the ZigBee Alliance specification.

5 Technical Solution

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* – OSGi services implementing the API defined here and explicitly set to be *exported* should be made available to networks with ZigBee enabled endpoints in a transparent way.

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 by 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 these clusters are mandatory or not. This entity is represented by a **ZigBeeDeviceDescription** object.
- *ZigBee Cluster* – Represents a cluster existing in the network. It holds its cluster description, the current value for each attributes 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 hold either Client or 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 **ZigBeeCommandDescription** 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 **ZigBeeAttribute** 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 **ZigBeeAttributeDescription** 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 Handler* – A ZigBee handler is a helper that manages asynchronous communication with the base driver. This entity is represented, e.g. by **ZigBeeHandler**.
- *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.
- *ZCL Exception* – An exception that delivers errors that occurred in the ZigBee stack or internally by the ZigBee Base Driver or by the ZigBee network.
- *ZigBee Exception* – This class represents root exception for all the code related to ZigBee/ZDP (see Table 2.137 ZDP Enumerations Description in ZIGBEE SPECIFICATION: 1_053474r17ZB_TSC-ZigBee-Specification.pdf)
- *ZCL Frame* – A ZCL frame that must used when invoking a command.
- *ZCL Header* – A ZCL header that describes the header of a ZCL frame.
- *ZigBee Group* – Enables group management. It is registered as an OSGi service.

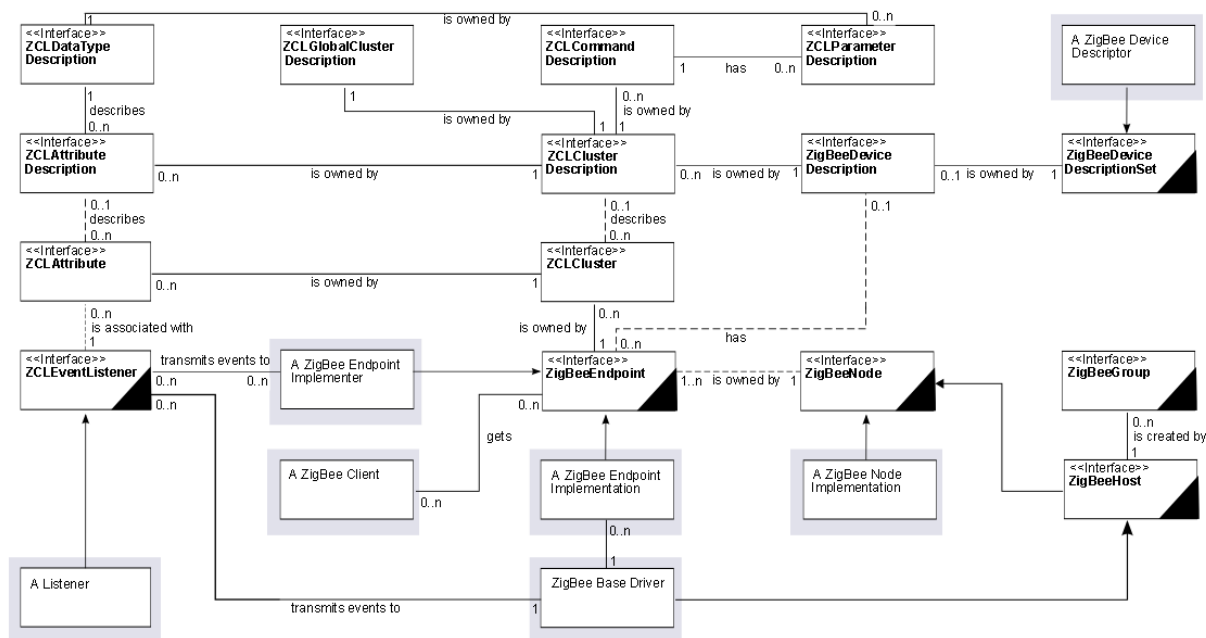


Figure 3 ZigBee Service Specification class Diagram org.osgi.service.zigbee package

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 4).

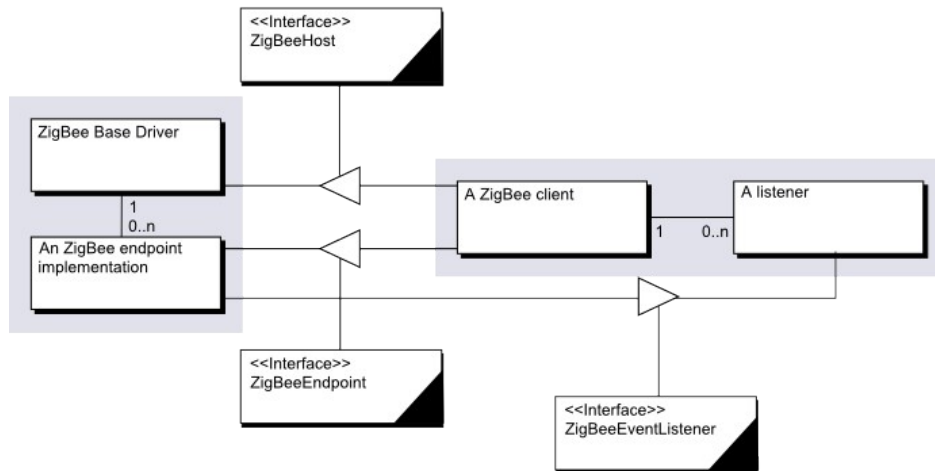


Figure 4: ZigBee device import

OSGi bundles may also expose framework-internal (local) **ZigBeeEndpoint** instances, registered within the framework (see Figure 5). The Base Driver then should emulate those objects as ZigBee endpoints associated to the ZigBee node represented by the underlying ZigBee hosts (ZigBee chip) on the ZigBee network. This is a *device export* situation. For more information about this process, please report to the “Exporting a ZigBee device” section below.

To control ZigBee

OSGi service registry and

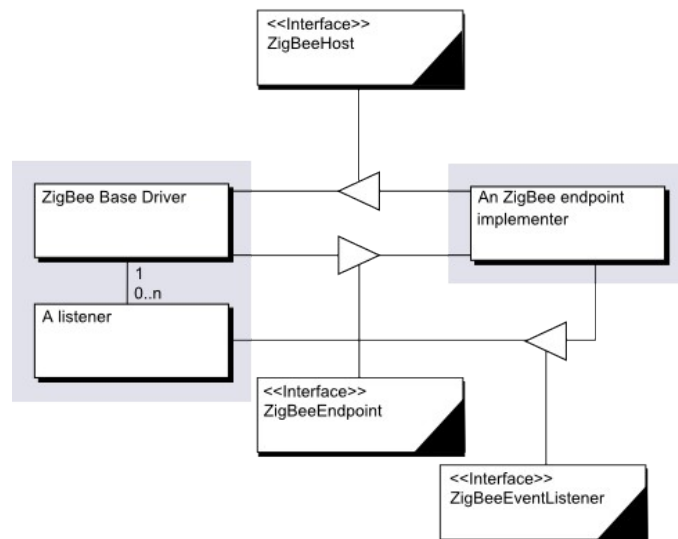


Figure 5: ZigBee device export

Several methods obey an asynchronous mechanism. For instance, ZigBee command invocation is made through the call to `ZigBeeCommand.invoke` method that returns nothing. A handler object – here a `ZigBeeCommandHandler` – is passed as an argument in this method call. When the command response is to be received, a callback – here **`notifyResponse()`** – is called on the handler to convey the command response frame. It is called by the base driver in the device import situation and it is called by the local `ZigBeeEndpoint` in the device export situation.

OSGi bundles – called listeners in Figure 3 – subscribe to attribute value changes through the White Board Pattern ([6]). They register an object under the `ZigBeeEventListener` interface with properties identifying a ZigBee attribute and a special event filter. This registration is conveyed as a configure report command on the ZigBee network in the device import situation. Reports are received by the base driver and transmitted as `ZigBeeEventListener.notifyEvent()` method calls on relevant `ZigBeeEventListener` services in this situation. Local `ZigBeeEndpoints` directly call these methods to notify listeners with reports in the export situation. The Base Driver conveys events received through listeners from local endpoints as reports to networked devices that have configured the relevant reporting.

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 6). 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.

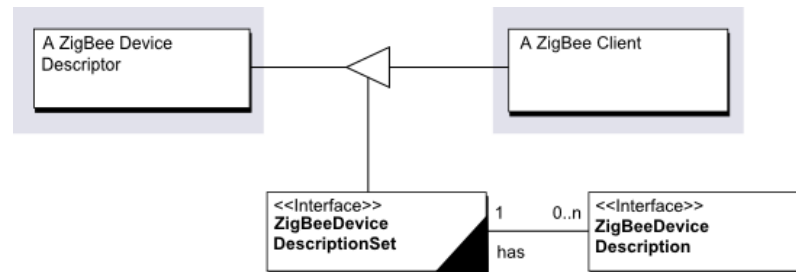


Figure 6: Using a set of device descriptions

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 OSGi registered ZigBee Node services. It must also *export*, on the ZigBee Network, `ZigBeeEndpoint` services (programmatically registered as OSGi services).

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 *refinement drivers* above a layer of *base drivers*. The refinement 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 7). In the case of a generic device abstraction layer, the model is agnostic to technologies.

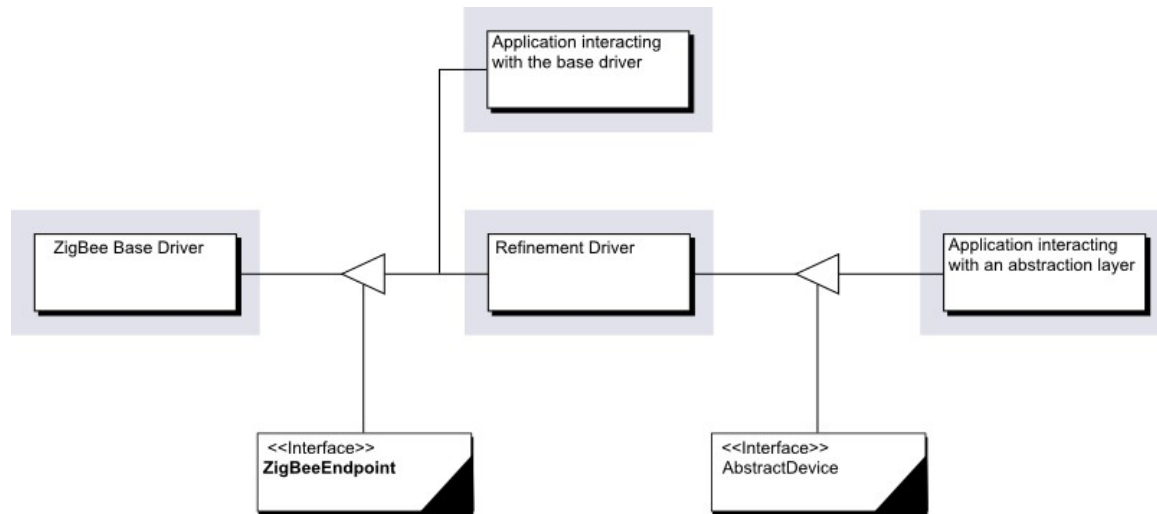


Figure 7: The ZigBee Base Driver and a refinement driver representing devices in an abstract model

The ZigBee Alliance defines their own abstract model with ZigBee Profiles, e.g., Home Automation, Lighting, and refinement drivers may provide the implementation of all ZigBee standard devices with ZigBee-specific Java interfaces. The AbstractDevice interface of Figure 7 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.

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/Long)
- **LOGICAL_TYPE** – (zigbee.node.description.type/Short) specifies the device type of the ZigBee node. The ZigBee specification defines three types of nodes: ZigBee coordinator, ZigBee router and ZigBee end device.
- **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, i.e. 3rd bit of "MAC Capabilities" in Node Descriptor. 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/Integer) represents the ZigBee receiver on when idle, i.e. 4th bit of "MAC Capabilities" in Node Descriptor. 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 `ZigBeeNode.getPanId()` method.
- **EXTENDED_PAN_ID** – (`zigbee.node.pan.extended.id/Long`) 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 previous conflict). `ZigBeeNode.getExtendedPanId()` returns the network extended PAN ID if specified.

Note that: **PAN_ID** and **EXTENDED_PAN_ID** are optional, but at least one of these properties MUST be specified.

- `org.osgi.service.device.Constants.DEVICE_CATEGORY` (see OSGi Compendium: 103 Device Access Specification) – (“DEVICE_CATEGORY”) describes a table of the categories to which the device belongs. One of the value MUST be “ZigBee” (`org.osgi.service.zigbee.ZigBeeEndpoint.DEVICE_CATEGORY`).

Additional properties (defined in Device Access – 103.2.1) 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 `ZigBeeComplexDescriptor.getModelName()`.
- **DEVICE_SERIAL** – if the complex descriptor of the device is available, the value MUST be set and MUST be the value returned by `ZigBeeComplexDescriptor.getSerialNumber()`.

Finally, `service.pid` property MUST be set.

ZigBee nodes describes themselves using descriptor data structures:

- `getNodeDescriptor()` – Returns a **ZigBeeNodeDescriptor** object representing Node Descriptor which contains information about the node capabilities.
- `getPowerDescriptor()` – Returns a **ZigBeePowerDescriptor** object representing Node Power Descriptor which gives a dynamic indication of the node power status.
- `getComplexDescriptor()` – Returns a **ZigBeeComplexDescriptor** object representing Complex Descriptor which contains extended information for each device descriptions contained in this node. Returns `null` if no Complex Descriptor is provided.
- `getUserDescriptor()` – Returns a **ZigBeeUserDescriptor** object representing User Descriptor which contains information that allows the user to identify the device using user-friendly character string. Returns `null` if no User Descriptor is provided.

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

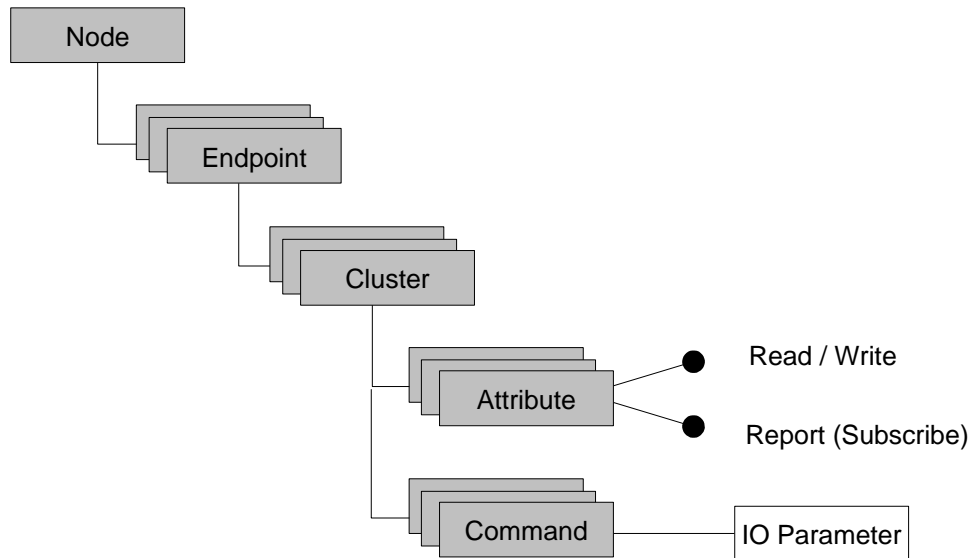


Figure 8: ZigBee Cluster Library model

- All interfaces corresponding to the ZigBee Cluster Library model (see Figure 5) 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.

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:

- **ENDPOINT_ID** – (`zigbee.endpoint.id/Integer`) 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 nodes 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 the profiles identifiers, the ZigBee specification defines several device identifiers, and some others are vendor specific.
- **DEVICE_VERSION** – (`zigbee.device.version/Integer`) specifies the device description version supported by this endpoint.
- **INPUT_CLUSTERS** – (`zigbee.endpoint.clusters.input/String[]`) specifies the list of input cluster ids supported by this endpoint. Input cluster at the end are called Server cluster.
- **OUTPUT_CLUSTERS** – (`zigbee.endpoint.clusters.output/String[]`) specifies the list of output cluster ids supported by this endpoint. Output cluster at the end are called Client cluster.
- **org.osgi.service.device.Constants.DEVICE_CATEGORY** (see OSGi Compendium: 103 Device Access Specification) – (“DEVICE_CATEGORY”) describes a table of the categories to which the device belongs. One of the value MUST be “ZigBee” (`org.osgi.service.zigbee.ZigBeeEndpoint.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. In endpoint export case, it is a free unique identifier that enables the base driver to identify any exported endpoint across local bundle restarts.

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

Each endpoint must provide a simple descriptor. `getSimpleDescriptor()` returns a **ZigBeeSimpleDescriptor** object which contains general information about the endpoint.

ZigBeeEndpoint interface provides two methods to bind and unbind ZigBee clusters: `bind(...)` and `unbind(...)`. 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(...)` method that provides the table of bound **ZigBeeEndpoints** identified by their service PIDs.

ZigBee Device Description

A ZigBee endpoint may have a description used to define and instantiate his input and output clusters, but also describes which of these clusters are mandatory or not.

ZigBeeDeviceDescription provides general information about an endpoint, and describes its inputs and outputs clusters.

ZigBee Device Description Set

ZigBeeDeviceDescriptionSet objects are registered as OSGi services by the Base Driver. A **ZigBeeDeviceDescriptionSet** provides `getDeviceSpecification(int deviceId, short version)` which returns the device description if provided, or null otherwise. **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_ID** property.
- **DEVICES** – (`zigbee.profile.devices/Integer[]`) A comma separated list of devices identifiers

supported by the set.

ZCL Cluster

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

ZCLCluster objects combine one or more ZigBee frame and ZigBeeAttribute.

ZCLCluster provides some methods for reading and writing attributes values:

- [readAttributes\(int\[\] attributelds, ZigBeeHandler handler\)](#) – The ZBD MAY (i.e. ZBD may cache request) generate the read attributes command on behalf of the OSGi application that is invoking the readAttributes method.
- [writeAttributes\(boolean undivided, ZigBeeAttributeRecord\[\] attributes, ZigBeeHandler handler\)](#) – The ZBD generates the write attributes command on behalf of the OSGi application that is invoking the writeAttributes method. If [handler](#) is set to [null](#), the base driver should use a 'no response' ZigBee general command (see Chapter 2.4 General Commands in ZCL specification). The [boolean undivided](#) parameter specifies that if any attribute cannot be written (e.g. 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 using a handler based response; operations parameters are attributes that will be manipulated:

- [invoke\(ZCLFrame frame, ZigBeeCommandHandler handler\)](#) – [bytes](#) is a sequence of byte and 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\(ZCLFrame, ZigBeeCommandHandler handler, String exportedServicePID\)](#) – [bytes](#) is a sequence of byte and 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 handler is provided to manage the command response asynchronously.

ZCL Cluster Description

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

Each cluster client and server may have attributes (see ZCL specifications chapter 2.2.1), received and generated commands. **ZCLClusterDescription** provides methods to describe commands, attributes and retrieve general cluster information.

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.

ZigBee Command Description

ZigBeeCommandDescription describes a ZigBee command.

ZigBeeCommandDescription contains **ZCLParameterDescription** objects which describe the command parameters. **ZCLParameterDescription** has, for instance, a [checkValue\(Object value\)](#) method which returns [true](#) if the parameter value is valid according to its description.

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

command.

Each 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.

ZigBeeCommandDescription also provides two methods for serializing (Java Values to bytes), and deserializing (bytes to Java Values). These bytes are, respectively, the parameters, and the returned value sent, respectively received when invoking a ZigBee command.

ZigBee Attribute

A ZigBee cluster is associated with a set of attributes. Each attribute is represented by a **ZigBeeAttribute** interface. **ZigBeeAttribute** provides [getValue\(ZigBeeAttributesHandler handler\)](#) and [setValue\(Object value, ZigBeeAttributesHandler handler\)](#) to retrieve and set current attribute value.

ZigBee Attribute Description

ZigBeeAttributeDescription describes information about a specific **ZigBeeAttribute**.

ZCL Data Type Description

ZigBeeAttributeDescription and **ZCLParameterDescription** provides a [getDataType\(\)](#) which returns a **ZCLDataTypeDescription** object. The interface provides, amongst others, the following methods:

- [void serialize\(Object param, ByteArrayOutputStream outdata\)](#) – Serializes a Java Object corresponding to the Java data type given by [getJavaDataType\(\)](#), and adds the result to the given [ByteArrayOutputStream](#) according to ZigBee Cluster Library.
- [Object deserialize\(ByteArrayInputStream data\)](#) – Deserializes the given data into a Java Object of the Java data type given by [getJavaDataType\(\)](#).

ZigBee Attribute Record

A **ZigBeeAttributeRecord** object holds some useful information about a given **ZigBee attribute**. It's mainly used by [ZCLCluster.writeAttributes\(...\)](#) for writing operations. **ZigBeeAttributeRecord** provides the following methods:

- [getId\(\)](#) – Returns the attribute identifier.
- [getDataType\(\)](#) – Returns the attribute data type.
- [getValue\(\)](#) – Returns the attribute object value.

ZigBee Handler

The ZigBee Handlers (i.e. **ZigBeeHandler**, and **ZigBeeCommandHandler**) help to manage asynchronous communication with the base driver. The defined interfaces are used when requesting the base driver and provide [onSuccess\(...\)](#) and [onFailure\(...\)](#) methods for managing responses.

This table shows the methods that uses a **Handler** and the following description of the map parameter:

ZigBee Data Types

ZigBeeDataTypes provides all standard ZigBee type descriptions as **ZCLDataTypeDescription** objects assigned to public final static fields (constants).

Here is the table of encoding relations between ZigBee types and Java types:

ZigBeeDataType constant	ZigBee type	Java type
--------------------------------	--------------------	------------------

SIGNED_INTEGER_8 BITMAP_8 GENERAL_DATA_8	Signed 8-bit integer 8-bit bitmap 8-bit data	Byte
SIGNED_INTEGER_16 BITMAP_16 GENERAL_DATA_16 UNSIGNED_INTEGER_8	Signed 16-bit integer 16-bit bitmap 16-bit data Unsigned 8-bit integer	Short
SIGNED_INTEGER_24 SIGNED_INTEGER_32 BITMAP_24 BITMAP_32 GENERAL_DATA_24 GENERAL_DATA_32 UNSIGNED_INTEGER_16 UNSIGNED_INTEGER_24	Signed 24-bit integer Signed 32-bit integer 24-bit bitmap 32-bit bitmap 24-bit data 32-bit data Unsigned 16-bit integer Unsigned 24-bit integer	Integer
SIGNED_INTEGER_40 SIGNED_INTEGER_48 SIGNED_INTEGER_56 SIGNED_INTEGER_64 BITMAP_40 BITMAP_48 BITMAP_56 BITMAP_64 GENERAL_DATA_40 GENERAL_DATA_48 GENERAL_DATA_56 GENERAL_DATA_64 UNSIGNED_INTEGER_32 UNSIGNED_INTEGER_40 UNSIGNED_INTEGER_48 UNSIGNED_INTEGER_56	Signed 40-bit integer Signed 48-bit integer Signed 56-bit integer Signed 64-bit integer 40-bit bitmap 48-bit bitmap 56-bit bitmap 64-bit bitmap 40-bit data 48-bit data 56-bit data 64-bit data Unsigned 32-bit integer Unsigned 40-bit integer Unsigned 48-bit integer Unsigned 56-bit integer	Long
UNSIGNED_INTEGER_64	Unsigned 64-bit integer	BigInteger
OCTET_STRING LONG_OCTET_STRING SECURITY_KEY	Octet string Long octet string 128-bit Security Key	byte[]
CHARACTER_STRING LONG_CHARACTER_STRING	Character string Long character string	String
BOOLEAN	Logical	Boolean
ENUMERATION_8	8-bit enumeration	Short
ENUMERATION_16 CLUSTER_ID ATTRIBUTE_ID	16-bit enumeration Unsigned 16-bit integer Unsigned 16-bit integer	Integer
BACNET_OID	BACnet OID*(Unsigned 32-bit integer)	Long
IEEE_ADDRESS	IEEE address (MAC-48,EUI-48/64)	Long
TIME_OF_DAY DATE UTC_TIME	Time of day Date UTCTime	Date
FLOATING_SEMI	Semi-precision	Float

FLOATING_SINGLE	Single precision	
FLOATING_DOUBLE	Double precision	Double
ARRAY STRUCTURE BAG	Array Structure Bag	List
SET	Set	Set
UNKNOWN	Unknown	

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

Working With a ZigBee Endpoint

All discovered ZigBee endpoints in the local networks are registered under **ZigBeeEndpoint** interface with the OSGi Framework. Every time a ZigBee endpoint 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 [3].

Using a remote ZigBee endpoint thus involves tracking ZigBee Endpoint 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 ZigBee Endpoint services and add them to a user interface, such as a remote controller application.

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

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** should 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** have priority on **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 could wait for a **ZigBeeHost** to appear in the OSGi service registry before setting these properties.

The Base Driver will export the endpoint on the ZigBee network associated to the ZigBee HOST PID, ZigBee 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 IDs properties set, the associated ZigBeeHost service is modified accordingly (getEndpoints() returns a different Enumeration object).

If an error occurs when exporting a ZigBee endpoint, then the base driver calls ZigBeeEndpoint.**notExported()** method with a relevant **ZCLErrorException** object as the input argument.

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 ZigBeeEndpoint.**notExported()** method with the **ZCLErrorException** object as the input argument with ZCLErrorException.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 ZigBee specification [1].

The reader must note that a same ZigBeeEndpoint object can not be registered several times with distinct PAN IDs since ZigBeeEndpoint.get**NodeAddress()** method can only return one ZigBee Node address..

If the PAN ID corresponds to more than one ZigBeeHost, the ZigBeeEndpoint MUST define the Extended PAN ID property which uniquely identifies a ZigBee network. The base driver will call ZigBeeEndpoint.**notExported()** with the error code ZCLErrorException.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 ZigBeeEndpoint.**notExported()** with the error code ZCLErrorException.OSGI_MULTIPLE_HOSTS.

Event API

There are two distinct event directions for the ZigBee Service specification.

- 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 from its own listeners.

ZigBee events are sent using the whiteboard pattern [6]., in which a bundle interested in receiving the ZigBee events registers an object implementing the **ZCLErrorListener** interface. The service MUST be registered with **PAN_ID** (zigbee.node.pan.id) and/or **EXTENDED_PAN_ID** (zigbee.node.extended.pan.id) properties. These properties indicates 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 ZigBee Event Listener for every attribute report configured in configure reporting commands it receives from the network.

The filter refers to the combination of the properties registered with the ZCLErrorListener service. The mandatory properties (i.e. each ZCLErrorListener MUST be registered with all the mandatory property) are:

- ZigBeeNode.**IEEE_ADDRESS** – (zigbee.node.ieee.address/Long) Only events generated by endpoints matching the specific node are delivered.
- ZigBeeEndpoint.**ID** – (zigbee.endpoint.id/Short) Only events matching a specific endpoint are delivered.
- ZCLCluster.**ID** – (zigbee.cluster.id/Integer) Only events generated by endpoints matching a specific cluster are delivered.
- ZigBeeAttribute.**ID** – (zigbee.attribute.id/Integer) Only events generated by endpoints matching a specific attribute are delivered.
- ZCLErrorListener.**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 ZCL – 2.4.7.1.4 Attribute Data Type Field).

The optional properties are:

- ZCLEventListener.**MIN_REPORT_INTERVAL** – (zigbee.attribute.min.report.interval/Integer) The minimum interval, in seconds, between issuing reports of the specified attribute. If it is not specified, then the base driver should set the value to 0x0000 in commands (see ZCL – 2.4.7.1.5).
- ZCLEventListener.**MAX_REPORT_INTERVAL** – (zigbee.attribute.max.report.interval/Integer) The maximum interval, in seconds, between issuing reports of the specified attribute. If it is not specified, then the base driver should set the value to 0xffff in commands (see 2.4.7.1.6).
- ZCLEventListener.**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 ZCLEventListener.**notifyTimeout(int)** is then called with the set '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]. 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 **notifyZigBeeEvent(ZigBeeEvent event)** 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 Life Cycle Layer chapter of the Core specification.

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

ZCL Exception

The **ZCLException** can be thrown and holds information about the different ZigBee ZCL layers. Error codes specified by ZigBee Alliance are conveyed by the **errorCode** field of ZCLException objects:

- **FAILURE** – (1) Operation was not successful.
- **MALFORMED_COMMAND** – (128) Wrong or missing field command.
- **CLUSTER_COMMAND_NOT_SUPPORTED** – (129) Cluster command not supported.
- **GENERAL_COMMAND_NOT_SUPPORTED** – (130) General command not supported.
- **MANUF_GENERAL_COMMAND_NOT_SUPPORTED** – (131) Manufacturer general command not supported.
- **MANUF_CLUSTER_COMMAND_NOT_SUPPORTED** – (132) Manufacturer cluster command not supported.
- **INVALID_FIELD** – (133) Invalid field.
- **UNSUPPORTED_ATTRIBUTE** – (134) Attribute not supported.
- **INVALID_VALUE** – (135) Invalid attribute value.
- **READ_ONLY** – (136) Read only attribute.
- **INSUFFICIENT_SPACE** – (137) Insufficient amount of free space.
- **DUPLICATE_EXISTS** – (138) Entry already exists in the table.
- **NOT_FOUND** – (139) Requested information can not be found.
- **UNREPORTABLE_TYPE** – (140) Attribute periodic reports cannot be issued.
- **INVALID_DATA_TYPE** – (141) Incorrect attribute data type.
- **HARDWARE_FAILURE** – (192) Operation unsuccessful due to a hardware failure.

- [SOFTWARE_FAILURE](#) – (193) Operation unsuccessful due to a software failure.
- [CALIBRATION_ERROR](#) – (194) An error occurred during calibration.

Some error codes are specified by the OSGi Alliance:

- [OSGI_EXISTING_ID](#) (16) – another endpoint exists with the same ID.
- [OSGI_MULTIPLE_HOSTS](#) (17) – several hosts exist for this PAN ID target or HOST_PID target.

ZDP Exception

The **ZDPException** can be thrown and holds information about the ZigBee ZDP layer. Error codes specified by ZigBee Alliance are conveyed by the `errorCode` field of **ZDPException** objects:

- [INV_REQUESTTYPE](#) – 0x80 The supplied request type was invalid.
- [DEVICE_NOT_FOUND](#) – 0x81 The requested device did not exist on a device following a child descriptor request to a parent.
- [INVALID_EP](#) – 0x82 The supplied endpoint was equal to 0x00 or between 0xf1 and 0xff.
- [NOT_ACTIVE](#) – 0x83 The requested endpoint is not described by a simple descriptor.
- [NOT_SUPPORTED](#) – 0x84 The requested optional feature is not supported on the target device.
- [TIMEOUT](#) – 0x85 A timeout has occurred with the requested operation.
- [NO_MATCH](#) – 0x86 The end device bind request was unsuccessful due to a failure to match any suitable clusters.
- [NO_ENTRY](#) – 0x88 The unbind request was unsuccessful due to the coordinator or source device not having an entry in its binding table to unbind.
- [NO_DESCRIPTOR](#) – 0x89 A child descriptor was not available following a discovery request to a parent.
- [INSUFFICIENT_SPACE](#) – 0x8a The device does not have storage space to support the requested operation.
- [NOT_PERMITTED](#) – 0x8b The device is not in the proper state to support the requested operation.
- [TABLE_FULL](#) – 0x8c The device does not have table space to support the operation.
- [NOT_AUTHORIZED](#) – 0x8d The permissions configuration table on the target indicates that the request is not authorized from this device.

Note that 0x01-0x7f, 0x87, and 0x8e-0xff are reserved.

ZCL Frame

The **ZCLFrame** contains a **ZCLHeader**, and a payload. It must be 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. That's why we standardize getters and we don't standardize setters.

ZigBee Group

The **ZigBeeGroup** enables group management (i.e. it provides list, join, and leave methods).

The creation of Groups is made through the **ZigBeeHost** `.createGroupService()` method.

ZigBeeGroup service should be registered with the following property:

- [ID](#) – (`zigbee.group.id/Integer`) The 16bits group address of the device.

And, the following **ZigBeeNode** properties:

- [DEVICE_CATEGORY](#)

- [INPUT_CLUSTERS](#)
- [HOST_PID](#)

ZigBee Networking

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. Constant value **ZigBeeNode.COORDINATOR** represents the ZigBee coordinator.
- ZigBee Router (ZR) – A router is capable of extending a ZigBee network by routing data from other ZigBee devices. Constant value **ZigBeeNode.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. Constant value **ZigBeeNode.END_DEVICE** represents a ZigBee end device.

Each discovered **ZigBeeNode** on the network must have a logical node type returned by **ZigBeeNode.getNodeDescriptor().getLogicalType()**.

Network selection

The base driver provides a ZigBee Host object for each 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].) This object must be registered under **ZigBeeHost** interface. ZigBee Host object enables to start, and stop the Host, stores the networking configuration information (channel, channel mask, logical type, PAN ID, Extended Pan ID, security level, network key), and provides a method to open the network for devices to join it (**permitJoin()**).

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 destined for the end device until the end device is able to wake up and receive the data.

Network coordination

In case **ZigBeeHost** is configured as the network coordinator, **ZigBeeHost.getNodeDescriptor().getLogicalType()** MUST return **ZigBeeNode.COORDINATOR** constant value. **ZigBeeHost** object will then be able to use the following operations for managing the network:

- [setChannel\(byte channel, ZigBeeHandler handler\)](#) – Sets the network channel.
- [setChannelMask\(int mask, ZigBeeHandler handler\)](#) – Sets a new configured channel mask.
- [refreshNetwork\(ZigBeeHandler handler\)](#) – 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.

Networking considerations

The Network Address is a 16 bits 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.

Security

Security management

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

Security amongst 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 amongst all devices in the network. The master key is only use as initial shared secret between two devices when they perform the Key Establishment to generate Link Keys.

Security configuration is provided by [getSecurityLevel\(\)](#) of **ZigBeeHost** object 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 of the network. The Trust Center stores all the shared network keys. **ZigBeeHost**.[getMasterKey\(\)](#) operation returns the network master key.

Conditional permission

When a bundle registers a ZigBee Endpoint OSGi service, then the basedriver exposes this Endpoint on the outside ZigBee network and this Endpoint has the ability to communicate with the others network devices. The basedriver 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.

6 Javadoc

OSGi Javadoc

06/05/14 10:25

Package Summary		Page
org.osgi.service.zigbee		42
org.osgi.service.zigbee.descriptions		112
org.osgi.service.zigbee.descriptors		130
org.osgi.service.zigbee.types		145

Package org.osgi.service.zigbee

Interface Summary		Page
<u>ZCLAttribute</u>	This interface represents a ZCLAttribute	48
<u>ZCLCluster</u>	This interface represents a ZCL Cluster	52
<u>ZCLCommandHandler</u>	Manage response of a command request to the Base Driver	56
<u>ZCLEventListener</u>	This interface represents a listener to events from ZigBee Device nodes	57
<u>ZCLFrame</u>	This interface represents a ZCL Frame.	64
<u>ZCLHeader</u>	This interface represents a ZCL Header.	66
<u>ZigBeeEndpoint</u>	This interface represents a ZigBee EndPoint.	83
<u>ZigBeeEvent</u>	This interface represents events generated by a ZigBee Device node	89
<u>ZigBeeGroup</u>	This interface represents a ZigBee Group	91
<u>ZigBeeHandler</u>	ZigBeeHandler manages response of a request to the Base Driver	94
<u>ZigBeeHost</u>	This interface represents the machine that hosts the code to run a ZigBee device or client.	95
<u>ZigBeeLinkQuality</u>	This interface represents an entry of the NeighborTableList (see Table 2.126 NeighborTableList Record Format in ZIGBEE SPECIFICATION: 1_053474r17ZB_TSC-ZigBee-Specification.pdf)	101
<u>ZigBeeNode</u>	This interface represents a ZigBee node, means a physical device that can communicate using the ZigBee protocol.	104
<u>ZigBeeRoute</u>	This interface represents an entry of the RoutingTableList (see Table 2.128 RoutingTableList Record Format in ZIGBEE SPECIFICATION: 1_053474r17ZB_TSC-ZigBee-Specification.pdf)	110

Class Summary		Page
<u>ZCLAttributeRecord</u>	This class represents a ZCLAttributeRecord	50
<u>ZigBeeDataTypes</u>	This interface represents all ZigBee data types, and contains the common serialize/deserialize methods for the org.osgi.service.zigbee.types.* Reference: 075366r04ZB_AFG-ZigBee_Cluster_Library_Public_download_version.pdf	72

Exception Summary		Page
<u>APSEException</u>	This exception class is specialized for the APS errors.	43
<u>ZCLException</u>	This class represents root exception for all the code related to ZigBee/ZCL.	59
<u>ZDPException</u>	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)	68

Class APSEException

org.osgi.service.zigbee

```

java.lang.Object
├── java.lang.Throwable
│   ├── java.lang.Exception
│   │   ├── java.lang.RuntimeException
│   │   └── org.osgi.service.zigbee.APSEException

```

All Implemented Interfaces:
Serializable

```

public class APSEException
extends RuntimeException

```

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.

Version:
1.0

Field Summary		Page
static int	ASDU_TOO_LONG A transmit request failed since the ASDU is too large and fragmentation is not supported.	44
static int	DEFRAG_DEFERRED A received fragmented frame could not be defragmented at the current time.	44
static int	DEFRAG_UNSUPPORTED A received fragmented frame could not be defragmented since the device does not support fragmentation.	45
static int	ILLEGAL_REQUEST A parameter value was out of range.	45
static int	INVALID_BINDING An APSME-UNBIND.request failed due to the requested binding link not existing in the binding table.	45
static int	INVALID_GROUP An APSME-REMOVE-GROUP.request has been issued with a group identifier that does not appear in the group table.	45
static int	INVALID_PARAMETER A parameter value was invalid or out of range.	45
static int	NO_ACK An APSDE-DATA.request requesting acknowledged transmission failed due to no acknowledgement being received.	45
static int	NO_BOUND_DEVICE An APSDE-DATA.request with a destination addressing mode set to 0x00 failed due to there being no devices bound to this device.	45
static int	NO_SHORT_ADDRESS 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.	45

static int	NOT_SUPPORTED An APSDE-DATA.request with a destination addressing mode set to 0x00 failed due to a binding table not being supported on the device.	46
static int	SECURED_LINK_KEY An ASDU was received that was secured using a link key.	46
static int	SECURED_NWK_KEY An ASDU was received that was secured using a network key.	46
static int	SECURITY_FAIL An APSDE-DATA.request requesting security has resulted in an error during the corresponding security processing.	46
static int	SUCCESS A request has been executed successfully.	44
static int	TABLE_FULL An APSME-BIND.request or APSME.ADDGROUP. request issued when the binding or group tables, respectively, were full.	46
static int	UNSECURED An ASDU was received without any security.	46
static int	UNSUPPORTED_ATTRIBUTE An APSME-GET.request or APSMESET. request has been issued with an unknown attribute identifier.	46

Constructor Summary		Page
APSEException (int errorCode, String errorDesc)		47
APSEException (String errordesc)		47

Method Summary		Page
int	getZigBeeErrorCode ()	47

Field Detail

SUCCESS

```
public static final int SUCCESS = 0
```

A request has been executed successfully.

ASDU_TOO_LONG

```
public static final int ASDU_TOO_LONG = 160
```

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

DEFRAG_DEFERRED

```
public static final int DEFRAG_DEFERRED = 161
```

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

DEFRAG_UNSUPPORTED

```
public static final int DEFRAG_UNSUPPORTED = 162
```

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

ILLEGAL_REQUEST

```
public static final int ILLEGAL_REQUEST = 163
```

A parameter value was out of range.

INVALID_BINDING

```
public static final int INVALID_BINDING = 164
```

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

INVALID_GROUP

```
public static final int INVALID_GROUP = 165
```

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

INVALID_PARAMETER

```
public static final int INVALID_PARAMETER = 166
```

A parameter value was invalid or out of range.

NO_ACK

```
public static final int NO_ACK = 167
```

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

NO_BOUND_DEVICE

```
public static final int NO_BOUND_DEVICE = 168
```

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

NO_SHORT_ADDRESS

```
public static final int NO_SHORT_ADDRESS = 169
```

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.

NOT_SUPPORTED

```
public static final int NOT_SUPPORTED = 170
```

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

SECURED_LINK_KEY

```
public static final int SECURED_LINK_KEY = 171
```

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

SECURED_NWK_KEY

```
public static final int SECURED_NWK_KEY = 172
```

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

SECURITY_FAIL

```
public static final int SECURITY_FAIL = 173
```

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

TABLE_FULL

```
public static final int TABLE_FULL = 174
```

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

UNSECURED

```
public static final int UNSECURED = 175
```

An ASDU was received without any security.

UNSUPPORTED_ATTRIBUTE

```
public static final int UNSUPPORTED_ATTRIBUTE = 176
```

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

Constructor Detail

APSEException

```
public APSEException(String errordesc)
```

Parameters:

`errordesc` - exception error description

APSEException

```
public APSEException(int errorCode,  
                     String errorDesc)
```

Parameters:

`errorCode` - An error code.

`errorDesc` - An error description which explain the type of problem.

Method Detail

getZigBeeErrorCode

```
public int getZigBeeErrorCode()
```

Returns:

A ZigBee error code defined a ZigBee Forum working committee or specified by a ZigBee vendor.

Interface ZCLAttribute

org.osgi.service.zigbee

public interface ZCLAttribute

This interface represents a ZCLAttribute

Version:
1.0

Field Summary		Page
String	ID Property key for the optional attribute id of a ZigBee Event Listener.	48

Method Summary		Page
ZCLDataTypeDescription	getDataType()	49
int	getId()	48
void	getValue(ZigBeeHandler handler) Gets the current value of the attribute.	48
void	setValue(Object value, ZigBeeHandler handler) Sets the current value of the attribute.	49

Field Detail

ID

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

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

Method Detail

getId

int getId()

Returns:
the attribute identifier (i.e. the attribute's ID)

getValue

void getValue([ZigBeeHandler](#) handler)
throws [ZCLException](#)

Gets the current value of the attribute. The response object given to the handler is the attribute's Java data type (see [getDataType\(\)](#) method) that will contain the current attribute value. In case of a failure, onFailure is called with a ZCLException.

Parameters:

handler - the handler

Throws:

[ZCLException](#)

setValue

```
void setValue(Object value,  
              ZigBeeHandler handler)
```

Sets the current value of the attribute. The response object given to the handler is a boolean set to true if the attribute value has been written or false otherwise. In case of a failure, onFailure is called with a ZCLException.

Parameters:

value - the Java value to set

handler - the handler

getDataType

```
ZCLDataTypeDescription getDataType()
```

Returns:

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

Class ZCLAttributeRecord

[org.osgi.service.zigbee](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.ZCLAttributeRecord
```

```
final public class ZCLAttributeRecord
extends Object
```

This class represents a ZCLAttributeRecord

Version:
1.0

Constructor Summary	Page
ZCLAttributeRecord (short id, ZCLDataTypeDescription dataType, Object value)	50

Method Summary	Page
ZCLDataTypeDescription getDataType ()	50
short getId ()	50
Object getValue ()	51

Constructor Detail

ZCLAttributeRecord

```
public ZCLAttributeRecord(short id,
                           ZCLDataTypeDescription dataType,
                           Object value)
```

Parameters:
id - the attribute record id
dataType - the attribute record dataType
value - the attribute record value

Method Detail

getId

```
public short getId()
```

Returns:
the id

getDataType

```
public ZCLDataTypeDescription getDataType()
```

Returns:
the dataType

getValue

```
public Object getValue()
```

Returns:
the value

Interface ZCLCluster

org.osgi.service.zigbee

```
public interface ZCLCluster
```

This interface represents a ZCL Cluster

Version:
1.0

Field Summary		Page
String	DOMAIN Property key for the optional cluster domain.	53
String	ID Property key for the optional cluster id.	52
String	NAME Property key for the optional cluster name.	53

Method Summary		Page
void	getAttribute (int attributeId, ZigBeeHandler handler) Get the cluster attribute identified corresponding to given attributeId.	53
void	getAttributes (ZigBeeHandler handler) Get an array of all this Cluster's Attributes.	53
void	getCommandIds (ZigBeeHandler handler) Get an array of all the commandIds of the ZigBeeCluster.	54
int	getId ()	53
void	invoke (ZCLFrame frame, ZCLCommandHandler handler) Invokes the action.	54
void	invoke (ZCLFrame frame, ZCLCommandHandler handler, String exportedServicePID) This method is to be used by applications when the targeted device has to distinguish between source endpoints of the message.	55
void	readAttributes (int[] attributesIds, ZigBeeHandler handler) Read a list of attributes.	53
void	writeAttributes (boolean undivided, ZCLAttributeRecord [] attributesRecords, ZigBeeHandler handler) Write a list of attributes.	54

Field Detail

ID

```
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.

DOMAIN

```
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.

NAME

```
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.

Method Detail

getId

```
int getId()
```

Returns:
the cluster identifier

getAttribute

```
void getAttribute(int attributeId,  
                  ZigBeeHandler handler)
```

Get the cluster attribute identified corresponding to given attributeId.

Parameters:
attributeId - an Attribute identifier
handler - the response handler

getAttributes

```
void getAttributes(ZigBeeHandler handler)
```

Get an array of all this Cluster's Attributes.

Parameters:
handler - the response handler

readAttributes

```
void readAttributes(int[] attributesIds,  
                   ZigBeeHandler handler)
```

Read a list of attributes. As described in "Table 2.11 APSME-GET.confirm Parameters" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a APSME-GET.confirm can have the following status: SUCCESS, or UNSUPPORTED_ATTRIBUTE (see [APSEException](#)).

The response object given to the handler is a Map. For each Map entry, the key is the attribute identifier of Integer type and the value is the associated attribute value of Object type (or null if an UNSUPPORTED_ATTRIBUTE occurred).

Parameters:

`attributesIds` - An array of attributes ids
`handler` - the response handler

writeAttributes

```
void writeAttributes(boolean undivided,  
                     ZCLAttributeRecord[] attributesRecords,  
                     ZigBeeHandler handler)
```

Write a list of attributes. As described in "Table 2.13 APSME-SET.confirm Parameters" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a APSME-SET.confirm can have the following status: SUCCESS, INVALID_PARAMETER or UNSUPPORTED_ATTRIBUTE (see [APSEException](#)). The response object given to the handler is a Map. For each Map entry, the key is the attribute identifier of Integer type and the value is the associated attribute status, i.e. SUCCESS, INVALID_VALUE, etc. In case undivided equals false, onSuccess() is always called to notify the response. In case undivided equals true and an error has occurred, onFailure is called with a ZCLException.

Parameters:

`undivided` - The write command is undivided or not
`attributesRecords` - An array of attributes records
`handler` - the response handler

getCommandIds

```
void getCommandIds(ZigBeeHandler handler)
```

Get an array of all the commandIds of the ZigBeeCluster. This method is implemented for devices implementing a version equal or later than 1.2 of the Home Automation Profile or other profiles that enable the discovery of command IDs as a general command. When the device implements a profile that does not support this feature, the method call throws a ZCLException with code GENERAL_COMMAND_NOT_SUPPORTED. The response object given to the handler is a List containing the commandIds.

Parameters:

`handler` - the response handler

invoke

```
void invoke(ZCLFrame frame,  
           ZCLCommandHandler handler)
```

Invokes the action. The handler will provide the invocation response in an asynchronously 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.

Parameters:

`frame` - a command frame sequence.

handler - The handler that manages the command response.

invoke

```
void invoke(ZCLFrame frame,  
           ZCLCommandHandler handler,  
           String exportedServicePID)
```

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.

Parameters:

frame - a command frame sequence.

handler - The handler that manages the command response.

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

Interface ZCLCommandHandler

org.osgi.service.zigbee

```
public interface ZCLCommandHandler
```

Manage response of a command request to the Base Driver

Version:

1.0

Method Summary

Page

void	notifyResponse (ZCLFrame frame, Exception e) Notifies the result (success or failure) of the call.	56
------	--	----

Method Detail

notifyResponse

```
void notifyResponse(ZCLFrame frame,  
                   Exception e)
```

Notifies the result (success or failure) of the call. The entity calling notifyresponse() (i.e., the base driver in the import situation) must not parse the ZCL frame payload. Thus, error codes that are conveyed in the ZCLFrame payload must not be turned into exceptions. The ZigBee Base Driver will release the handler object when he receives a null frame in a notifyResponse call or thanks to the an implementation specific timeout. The ZigBee Base Driver MUST discard the Default Response if the caller set the DisableDefaultReponse flag and the status of DefaultResponse command is SUCCESS. Multiple response management: Several responses MAY be sent to an endpoint. A handler could be called several times on a command handler.

Parameters:

frame - the ZCLFrame
e - the exception if any

Interface ZCLEventListener

org.osgi.service.zigbee

```
public interface ZCLEventListener
```

This interface represents a listener to events from ZigBee Device nodes

Version:
1.0

Field Summary		Page
String	ATTRIBUTE_DATA_TYPE Property key for the optional attribute data type of an attribute reporting configuration record, cf.	57
String	MAX_REPORT_INTERVAL Property key for the optional maximum interval, in seconds between issuing reports of the attribute.	58
String	MIN_REPORT_INTERVAL Property key for the optional minimum interval, in seconds between issuing reports of the attribute.	57
String	REPORTABLE_CHANGE Property key for the optional maximum change to the attribute that will result in a report being issued.	58

Method Summary		Page
void	notifyEvent (ZigBeeEvent event) Callback method that is invoked for received events.	58
void	notifyTimeOut (int timeout) TIMEOUT_PERIOD is sent from the attribute owner to the listening client to say that the interval between reports may exceed MAX_INTERVAL.	58
void	onFailure (ZCLException e) Notifies a failure, i.e. when either a ZCLException.UNSUPPORTED_ATTRIBUTE , or a ZCLException.UNREPORTABLE_ATTRIBUTE , or ZCLException.INVALID_VALUE , or ZCLException.INVALID_DATA_TYPE status occurs.	58

Field Detail

ATTRIBUTE_DATA_TYPE

```
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.

MIN_REPORT_INTERVAL

```
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.

MAX_REPORT_INTERVAL

```
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.

REPORTABLE_CHANGE

```
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.

Method Detail

notifyEvent

```
void notifyEvent(ZigBeeEvent event)
```

Callback method that is invoked for received events. This method must be called asynchronously.

Parameters:

event - a set of events

onFailure

```
void onFailure(ZCLException e)
```

Notifies a failure, i.e. when either a `ZCLException.UNSUPPORTED_ATTRIBUTE`, or a `ZCLException.UNREPORTABLE_ATTRIBUTE`, or `ZCLException.INVALID_VALUE`, or `ZCLException.INVALID_DATA_TYPE` status occurs.

Parameters:

e - the `ZCLException`.

notifyTimeOut

```
void notifyTimeOut(int timeout)
```

`TIMEOUT_PERIOD` is sent from the attribute owner to the listening client to say that the interval between reports may exceed `MAX_INTERVAL`.

Parameters:

timeout - in seconds

Class ZCLException

org.osgi.service.zigbee

```

java.lang.Object
├── java.lang.Throwable
│   ├── java.lang.Exception
│   │   ├── java.lang.RuntimeException
│   │   └── org.osgi.service.zigbee.ZCLException

```

All Implemented Interfaces:
Serializable

```

public class ZCLException
extends RuntimeException

```

This class represents root exception for all the code related to ZigBee/ZCL.

Version:
1.0

Field Summary		Page
static short	CALIBRATION_ERROR CALIBRATION_ERROR	62
static short	CLUSTER_COMMAND_NOT_SUPPORTED CLUSTER_COMMAND_NOT_SUPPORTED	60
static short	DUPLICATE_EXISTS DUPLICATE_EXISTS	61
static short	FAILURE FAILURE	60
static short	GENERAL_COMMAND_NOT_SUPPORTED GENERAL_COMMAND_NOT_SUPPORTED	60
static short	HARDWARE_FAILURE HARDWARE_FAILURE - in this case, an additional exception describing the problem can be nested.	62
static short	INSUFFICIENT_SPACE INSUFFICIENT_SPACE	61
static short	INVALID_DATA_TYPE INVALID_DATA_TYPE	62
static short	INVALID_FIELD INVALID_FIELD	61
static short	INVALID_VALUE INVALID_VALUE	61
static short	MALFORMED_COMMAND MALFORMED_COMMAND	60
static short	MANUF_CLUSTER_COMMAND_NOT_SUPPORTED MANUF_CLUSTER_COMMAND_NOT_SUPPORTED	61
static short	MANUF_GENERAL_COMMAND_NOT_SUPPORTED MANUF_GENERAL_COMMAND_NOT_SUPPORTED	61
static short	NOT_FOUND NOT_FOUND	62

static short	READ_ONLY READ_ONLY	61
static short	SOFTWARE_FAILURE SOFTWARE_FAILURE - in this case, an additional exception describing the problem can be nested.	62
static short	SUCCESS SUCCESS	60
static short	UNREPORTABLE_TYPE UNREPORTABLE_TYPE	62
static short	UNSUPPORTED_ATTRIBUTE UNSUPPORTED_ATTRIBUTE	61

Constructor Summary		Page
ZCLEException (int errorCode, String errorDesc)		63
ZCLEException (String errordesc)		62

Method Summary		Page
int	getZigBeeErrorCode ()	63

Field Detail

SUCCESS

```
public static final short SUCCESS = 0
```

SUCCESS

FAILURE

```
public static final short FAILURE = 1
```

FAILURE

MALFORMED_COMMAND

```
public static final short MALFORMED_COMMAND = 128
```

MALFORMED_COMMAND

CLUSTER_COMMAND_NOT_SUPPORTED

```
public static final short CLUSTER_COMMAND_NOT_SUPPORTED = 129
```

CLUSTER_COMMAND_NOT_SUPPORTED

GENERAL_COMMAND_NOT_SUPPORTED

```
public static final short GENERAL_COMMAND_NOT_SUPPORTED = 130
```

GENERAL_COMMAND_NOT_SUPPORTED

MANUF_CLUSTER_COMMAND_NOT_SUPPORTED

```
public static final short MANUF_CLUSTER_COMMAND_NOT_SUPPORTED = 131
```

MANUF_CLUSTER_COMMAND_NOT_SUPPORTED

MANUF_GENERAL_COMMAND_NOT_SUPPORTED

```
public static final short MANUF_GENERAL_COMMAND_NOT_SUPPORTED = 132
```

MANUF_GENERAL_COMMAND_NOT_SUPPORTED

INVALID_FIELD

```
public static final short INVALID_FIELD = 133
```

INVALID_FIELD

UNSUPPORTED_ATTRIBUTE

```
public static final short UNSUPPORTED_ATTRIBUTE = 134
```

UNSUPPORTED_ATTRIBUTE

INVALID_VALUE

```
public static final short INVALID_VALUE = 135
```

INVALID_VALUE

READ_ONLY

```
public static final short READ_ONLY = 136
```

READ_ONLY

INSUFFICIENT_SPACE

```
public static final short INSUFFICIENT_SPACE = 137
```

INSUFFICIENT_SPACE

DUPLICATE_EXISTS

```
public static final short DUPLICATE_EXISTS = 138
```

DUPLICATE_EXISTS

NOT_FOUND

```
public static final short NOT_FOUND = 139
```

NOT_FOUND

UNREPORTABLE_TYPE

```
public static final short UNREPORTABLE_TYPE = 140
```

UNREPORTABLE_TYPE

INVALID_DATA_TYPE

```
public static final short INVALID_DATA_TYPE = 141
```

INVALID_DATA_TYPE

HARDWARE_FAILURE

```
public static final short HARDWARE_FAILURE = 192
```

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

SOFTWARE_FAILURE

```
public static final short SOFTWARE_FAILURE = 193
```

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

CALIBRATION_ERROR

```
public static final short CALIBRATION_ERROR = 194
```

CALIBRATION_ERROR

Constructor Detail

ZCLException

```
public ZCLException(String errordesc)
```

Parameters:

errordesc - exception error description

ZCLException

```
public ZCLException(int errorCode,  
                    String errorDesc)
```

Parameters:

`errorCode` - An error code.

`errorDesc` - An error description which explain the type of problem.

Method Detail

getZigBeeErrorCode

```
public int getZigBeeErrorCode()
```

Returns:

A ZigBee error code defined a ZigBee Forum working committee or specified by a ZigBee vendor.

Interface ZCLFrame

org.osgi.service.zigbee

public interface ZCLFrame

This interface represents a ZCL Frame.

Method Summary		Page
ZCLHeader	getHeader() Get this ZCLFrame's header	64
ByteArrayInputStream	getInputStream()	64
ByteArrayOutputStream	getOutputStream()	64
byte[]	getPayload() Get (a copy of this ZCLFrame) payload	64

Method Detail

getHeader

[ZCLHeader](#) getHeader()

Get this ZCLFrame's header

Returns:
the header

getPayload

byte[] getPayload()

Get (a copy of this ZCLFrame) payload

Returns:
a copy of the payload

getInputStream

ByteArrayInputStream getInputStream()

Returns:
this ZCLFrame's inputstream that can be used in order to obtain this ZCLFrame's input bytes.

getOutputStream

ByteArrayOutputStream getOutputStream()

Returns:

this ZCLFrame's outputstream that can be used in order to write data/bytes to this ZCLFrame.

Interface ZCLHeader

org.osgi.service.zigbee

```
public interface ZCLHeader
```

This interface represents a ZCL Header.

Method Summary		Page
int	getCommandId() Get this ZCLHeader's command id	66
int	getManufacturerCode() Get manufacturerCode Default value is: -1 (no code)	66
boolean	isClientServerDirection()	67
boolean	isClusterSpecificCommand()	66
boolean	isDefaultResponseEnabled()	67
boolean	isManufacturerSpecific()	66

Method Detail

getCommandId

```
int getCommandId()
```

Get this ZCLHeader's command id

Returns:
the commandId

getManufacturerCode

```
int getManufacturerCode()
```

Get manufacturerCode Default value is: -1 (no code)

Returns:
the manufacturerCode

isClusterSpecificCommand

```
boolean isClusterSpecificCommand()
```

Returns:
the isClusterSpecificCommand value

isManufacturerSpecific

```
boolean isManufacturerSpecific()
```

Returns:
the isManufacturerSpecific value

isClientServerDirection

`boolean isClientServerDirection()`

Returns:
the isClientServerDirection value

isDefaultResponseEnabled

`boolean isDefaultResponseEnabled()`

Returns:
the isDefaultResponseEnabled value

Class ZDPException

org.osgi.service.zigbee

```
java.lang.Object
├── java.lang.Throwable
│   ├── java.lang.Exception
│   │   ├── java.lang.RuntimeException
│   │   └── org.osgi.service.zigbee.ZDPException
```

All Implemented Interfaces:
Serializable

```
public class ZDPException
extends RuntimeException
```

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)

Version:
1.0

Field Summary		Page
static short	DEVICE_NOT_FOUND The requested device did not exist on a device following a child descriptor request to a parent.	69
static short	INSUFFICIENT_SPACE The device does not have storage space to support the requested operation.	70
static short	INV_REQUESTTYPE The supplied request type was invalid.	69
static short	INVALID_EP The supplied endpoint was equal to 0x00 or between 0xf1 and 0xff.	69
static short	NO_DESCRIPTOR A child descriptor was not available following a discovery request to a parent.	70
static short	NO_ENTRY The unbind request was unsuccessful due to the coordinator or source device not having an entry in its binding table to unbind.	70
static short	NO_MATCH The end device bind request was unsuccessful due to a failure to match any suitable clusters.	70
static short	NOT_ACTIVE The requested endpoint is not described by a simple descriptor.	69
static short	NOT_AUTHORIZED The permissions configuration table on the target indicates that the request is not authorized from this device.	71
static short	NOT_PERMITTED The device is not in the proper state to support the requested operation.	70
static short	NOT_SUPPORTED The requested optional feature is not supported on the target device.	70
static short	OSGI_EXISTING_ID OSGI_EXISTING_ID (16) â€œ another endpoint exists with the same ID.	71

static short	OSGI_MULTIPLE_HOSTS OSGI_MULTIPLE_HOSTS (17) â€œ several hosts exist for this PAN ID target or HOST_PID target.	71
static short	SUCCESS The requested operation or transmission was completed successfully.	69
static short	TABLE_FULL The device does not have table space to support the operation.	70
static short	TIMEOUT A timeout has occurred with the requested operation.	70

Constructor Summary		Page
ZDPEException (int errorCode, String errorDesc)		71
ZDPEException (String errordesc)		71

Method Summary		Page
int getZigBeeErrorCode ()		71

Field Detail

SUCCESS

```
public static final short SUCCESS = 0
```

The requested operation or transmission was completed successfully.

INV_REQUESTTYPE

```
public static final short INV_REQUESTTYPE = 128
```

The supplied request type was invalid.

DEVICE_NOT_FOUND

```
public static final short DEVICE_NOT_FOUND = 129
```

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

INVALID_EP

```
public static final short INVALID_EP = 130
```

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

NOT_ACTIVE

```
public static final short NOT_ACTIVE = 131
```

The requested endpoint is not described by a simple descriptor.

NOT_SUPPORTED

```
public static final short NOT_SUPPORTED = 132
```

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

TIMEOUT

```
public static final short TIMEOUT = 133
```

A timeout has occurred with the requested operation.

NO_MATCH

```
public static final short NO_MATCH = 134
```

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

NO_ENTRY

```
public static final short NO_ENTRY = 136
```

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

NO_DESCRIPTOR

```
public static final short NO_DESCRIPTOR = 137
```

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

INSUFFICIENT_SPACE

```
public static final short INSUFFICIENT_SPACE = 138
```

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

NOT_PERMITTED

```
public static final short NOT_PERMITTED = 139
```

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

TABLE_FULL

```
public static final short TABLE_FULL = 140
```

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

NOT_AUTHORIZED

```
public static final short NOT_AUTHORIZED = 141
```

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

OSGI_EXISTING_ID

```
public static final short OSGI_EXISTING_ID = 16
```

OSGI_EXISTING_ID (16) â€” another endpoint exists with the same ID.

OSGI_MULTIPLE_HOSTS

```
public static final short OSGI_MULTIPLE_HOSTS = 17
```

OSGI_MULTIPLE_HOSTS (17) â€” several hosts exist for this PAN ID target or HOST_PID target.

Constructor Detail

ZDPEException

```
public ZDPEException(String errordesc)
```

Parameters:

errordesc - exception error description

ZDPEException

```
public ZDPEException(int errorCode,  
                     String errorDesc)
```

Parameters:

errorCode - An error code.

errorDesc - An error description which explain the type of problem.

Method Detail

getZigBeeErrorCode

```
public int getZigBeeErrorCode()
```

Returns:

A ZigBee error code defined a ZigBee Forum working committee or specified by a ZigBee vendor.

Class ZigBeeDataTypes

org.osgi.service.zigbee

```
java.lang.Object
└─ org.osgi.service.zigbee.ZigBeeDataTypes
```

```
public class ZigBeeDataTypes
extends Object
```

This interface represents all ZigBee data types, and contains the common serialize/deserialize methods for the `org.osgi.service.zigbee.types.*` Reference: `075366r04ZB_AFG-ZigBee_Cluster_Library_Public_download_version.pdf`

Version:
1.0

Field Summary		Page
static short	ARRAY 2.5.2.15 Array An array is an ordered sequence of zero or more elements, all of the same data type.	79
static short	ATTRIBUTE_ID 2.5.2.23 Attribute ID This type represents an attribute identifier as defined in spec.	81
static short	BACNET_OID 2.5.2.24 BACnet OID (Object Identifier) The BACnet OID data type is included to allow interworking with BACnet.	81
static short	BAG 2.5.2.18 Bag A bag behaves exactly the same as a set, except that the restriction that no two elements may have the same value is removed.	80
static short	BITMAP_16	76
static short	BITMAP_24	76
static short	BITMAP_32	76
static short	BITMAP_40	76
static short	BITMAP_48	76
static short	BITMAP_56	76
static short	BITMAP_64	76
static short	BITMAP_8 2.5.2.4 Bitmap (8, 16, 24, 32, 40, 48, 56 and 64-bit) The Bitmap type holds 8, 16, 24, 32, 40, 48, 56 or 64 logical values, one per bit, depending on its length.	76
static short	BOOLEAN 2.5.2.3 Boolean The Boolean type represents a logical value, either FALSE (0x00) or TRUE (0x01).	75
static short	CHARACTER_STRING 2.5.2.12 Character String The character string data type contains data octets encoding characters according to the language and character set field of the complex descriptor.	79
static short	CLUSTER_ID 2.5.2.22 Cluster ID This type represents a cluster identifier as defined in spec.	80

static short	DATE 2.5.2.20 Date The Time of day data type shall be formatted as illustrated in spec.	80
static short	ENUMERATION_16	78
static short	ENUMERATION_8 2.5.2.7 Enumeration (8-bit, 16-bit) The Enumeration type represents an index into a lookup table to determine the final value.	78
static short	FLOATING_DOUBLE 2.5.2.10 Double Precision The format of the double precision data type is based on the IEEE 754 standard for binary floating-point arithmetic.	79
static short	FLOATING_SEMI 2.5.2.8 Semi-precision The ZigBee semi-precision number format is based on the IEEE 754 standard for binary floating-point arithmetic.	78
static short	FLOATING_SINGLE 2.5.2.9 Single Precision The format of the single precision data type is based on the IEEE 754 standard for binary floating-point arithmetic.	78
static short	GENERAL_DATA_16	75
static short	GENERAL_DATA_24	75
static short	GENERAL_DATA_32	75
static short	GENERAL_DATA_40	75
static short	GENERAL_DATA_48	75
static short	GENERAL_DATA_56	75
static short	GENERAL_DATA_64	75
static short	GENERAL_DATA_8 2.5.2.2 General Data (8, 16, 24, 32, 40, 48, 56 and 64-bit) This type has no rules about its use, and may be used when a data element is needed but its use does not conform to any of the standard types.	75
static short	IEEE_ADDRESS 2.5.2.25 IEEE Address The IEEE Address data type is a 64-bit IEEE address that is unique to every ZigBee device.	81
static short	LONG_CHARACTER_STRING 2.5.2.14 Long Character String The long character string data type contains data octets encoding characters according to the language and character set field of the complex descriptor.	79
static short	LONG_OCTET_STRING 2.5.2.13 Long Octet String The long octet string data type contains data in an application-defined format, not defined in this specification.	79
static short	NO_DATA 2.5.2.1 No Data Type The no data type is a special type to represent an attribute with no associated data.	74
static short	OCTET_STRING 2.5.2.11 Octet String The octet string data type contains data in an application-defined format, not defined in this specification.	79
static short	ONE_HUNDRED_TWENTY_EIGHT_BIT_SECURITY_KEY 2.5.2.26 128-bit Security Key The 128-bit Security Key data type is for use in ZigBee security, and may take any 128-bit value.	81
static short	SET 2.5.2.17 Set A set is a collection of elements with no associated order.	80
static short	SIGNED_INTEGER_16	77
static short	SIGNED_INTEGER_24	77

static short	SIGNED_INTEGER_32	78
static short	SIGNED_INTEGER_40	78
static short	SIGNED_INTEGER_48	78
static short	SIGNED_INTEGER_56	78
static short	SIGNED_INTEGER_64	78
static short	SIGNED_INTEGER_8 2.5.2.6 Signed Integer (8, 16, 24, 32, 40, 48, 56 and 64-bit) This 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.	77
static short	STRUCTURE 2.5.2.16 Structure A structure is an ordered sequence of elements, which may be of different data types.	79
static short	TIME_OF_DAY 2.5.2.19 Time of Day The Time of Day data type shall be formatted as illustrated in spec.	80
static short	UNSIGNED_INTEGER_16	77
static short	UNSIGNED_INTEGER_24	77
static short	UNSIGNED_INTEGER_32	77
static short	UNSIGNED_INTEGER_40	77
static short	UNSIGNED_INTEGER_48	77
static short	UNSIGNED_INTEGER_56	77
static short	UNSIGNED_INTEGER_64	77
static short	UNSIGNED_INTEGER_8 2.5.2.5 Unsigned Integer (8, 16, 24, 32, 40, 48, 56 and 64-bit) This 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.	76
static short	UTC_TIME 2.5.2.21 UTCTime 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).	80

Constructor Summary		Pag e
ZigBeeDataTypes ()		81

Method Summary		Pag e
static Object	decode (short type, ByteArrayInputStream value)	81
static void	encode (short type, Object value, ByteArrayOutputStream outdata)	81

Field Detail

NO_DATA

```
public static final short NO_DATA = 0
```

2.5.2.1 No Data Type The no data type is a special type to represent an attribute with no associated data.

GENERAL_DATA_8

```
public static final short GENERAL_DATA_8 = 8
```

2.5.2.2 General Data (8, 16, 24, 32, 40, 48, 56 and 64-bit) This type has no rules about its use, and may be used when a data element is needed but its use does not conform to any of the standard types.

GENERAL_DATA_16

```
public static final short GENERAL_DATA_16 = 9
```

GENERAL_DATA_24

```
public static final short GENERAL_DATA_24 = 10
```

GENERAL_DATA_32

```
public static final short GENERAL_DATA_32 = 11
```

GENERAL_DATA_40

```
public static final short GENERAL_DATA_40 = 12
```

GENERAL_DATA_48

```
public static final short GENERAL_DATA_48 = 13
```

GENERAL_DATA_56

```
public static final short GENERAL_DATA_56 = 14
```

GENERAL_DATA_64

```
public static final short GENERAL_DATA_64 = 15
```

BOOLEAN

```
public static final short BOOLEAN = 16
```

2.5.2.3 Boolean 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.

BITMAP_8

```
public static final short BITMAP_8 = 24
```

2.5.2.4 Bitmap (8, 16, 24, 32, 40, 48, 56 and 64-bit) The Bitmap type holds 8, 16, 24, 32, 40, 48, 56 or 64 logical values, one per bit, depending on its length. There is no value that represents an invalid value of this type.

BITMAP_16

```
public static final short BITMAP_16 = 25
```

BITMAP_24

```
public static final short BITMAP_24 = 26
```

BITMAP_32

```
public static final short BITMAP_32 = 27
```

BITMAP_40

```
public static final short BITMAP_40 = 28
```

BITMAP_48

```
public static final short BITMAP_48 = 29
```

BITMAP_56

```
public static final short BITMAP_56 = 30
```

BITMAP_64

```
public static final short BITMAP_64 = 31
```

UNSIGNED_INTEGER_8

```
public static final short UNSIGNED_INTEGER_8 = 32
```

2.5.2.5 Unsigned Integer (8, 16, 24, 32, 40, 48, 56 and 64-bit) This 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, 0xfffffff, 0xffffffff, 0xffffffff, 0xfffffffffff and 0xffffffffffff respectively.

UNSIGNED_INTEGER_16

```
public static final short UNSIGNED_INTEGER_16 = 33
```

UNSIGNED_INTEGER_24

```
public static final short UNSIGNED_INTEGER_24 = 34
```

UNSIGNED_INTEGER_32

```
public static final short UNSIGNED_INTEGER_32 = 35
```

UNSIGNED_INTEGER_40

```
public static final short UNSIGNED_INTEGER_40 = 36
```

UNSIGNED_INTEGER_48

```
public static final short UNSIGNED_INTEGER_48 = 37
```

UNSIGNED_INTEGER_56

```
public static final short UNSIGNED_INTEGER_56 = 38
```

UNSIGNED_INTEGER_64

```
public static final short UNSIGNED_INTEGER_64 = 39
```

SIGNED_INTEGER_8

```
public static final short SIGNED_INTEGER_8 = 40
```

2.5.2.6 Signed Integer (8, 16, 24, 32, 40, 48, 56 and 64-bit) This 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.

SIGNED_INTEGER_16

```
public static final short SIGNED_INTEGER_16 = 41
```

SIGNED_INTEGER_24

```
public static final short SIGNED_INTEGER_24 = 42
```

SIGNED_INTEGER_32

```
public static final short SIGNED_INTEGER_32 = 43
```

SIGNED_INTEGER_40

```
public static final short SIGNED_INTEGER_40 = 44
```

SIGNED_INTEGER_48

```
public static final short SIGNED_INTEGER_48 = 45
```

SIGNED_INTEGER_56

```
public static final short SIGNED_INTEGER_56 = 46
```

SIGNED_INTEGER_64

```
public static final short SIGNED_INTEGER_64 = 47
```

ENUMERATION_8

```
public static final short ENUMERATION_8 = 48
```

2.5.2.7 Enumeration (8-bit, 16-bit) 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.

ENUMERATION_16

```
public static final short ENUMERATION_16 = 49
```

FLOATING_SEMI

```
public static final short FLOATING_SEMI = 56
```

2.5.2.8 Semi-precision The ZigBee semi-precision number format is based on the IEEE 754 standard for binary floating-point arithmetic. This number format should be used very sparingly, when absolutely necessary, keeping in mind the code and processing required supporting it. See reference on top of this class.

FLOATING_SINGLE

```
public static final short FLOATING_SINGLE = 57
```

2.5.2.9 Single Precision The format of the single precision data type is based on the IEEE 754 standard for binary floating-point arithmetic. This number format should be used very sparingly, when absolutely necessary, keeping in mind the code and processing required supporting it. See reference on top of this class.

FLOATING_DOUBLE

```
public static final short FLOATING_DOUBLE = 58
```

2.5.2.10 Double Precision The format of the double precision data type is based on the IEEE 754 standard for binary floating-point arithmetic. This number format should be used very sparingly, when absolutely necessary, keeping in mind the code and processing required supporting it. See reference on top of this class.

OCTET_STRING

```
public static final short OCTET_STRING = 65
```

2.5.2.11 Octet String The octet string data type contains data in an application-defined format, not defined in this specification. See reference on top of this class.

CHARACTER_STRING

```
public static final short CHARACTER_STRING = 66
```

2.5.2.12 Character String The character string data type contains data octets encoding characters according to the language and character set field of the complex descriptor. See reference on top of this class.

LONG_OCTET_STRING

```
public static final short LONG_OCTET_STRING = 67
```

2.5.2.13 Long Octet String The long octet string data type contains data in an application-defined format, not defined in this specification. See reference on top of this class.

LONG_CHARACTER_STRING

```
public static final short LONG_CHARACTER_STRING = 68
```

2.5.2.14 Long Character String The long character string data type contains data octets encoding characters according to the language and character set field of the complex descriptor. See reference on top of this class.

ARRAY

```
public static final short ARRAY = 72
```

2.5.2.15 Array 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, and may be further limited by any relevant profile or application. See reference on top of this class.

STRUCTURE

```
public static final short STRUCTURE = 76
```

2.5.2.16 Structure 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, and may be further limited by any relevant profile or application. See reference on top of this class.

SET

```
public static final short SET = 80
```

2.5.2.17 Set 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, and may be further limited by any relevant profile or application. See reference on top of this class.

BAG

```
public static final short BAG = 81
```

2.5.2.18 Bag A bag behaves exactly the same as a set, except that the restriction that no two elements may have the same value is removed.

TIME_OF_DAY

```
public static final short TIME_OF_DAY = 224
```

2.5.2.19 Time of Day The Time of Day data type shall be formatted as illustrated in spec. See reference on top of this class.

DATE

```
public static final short DATE = 225
```

2.5.2.20 Date The Time of day data type shall be formatted as illustrated in spec. See reference on top of this class.

UTC_TIME

```
public static final short UTC_TIME = 226
```

2.5.2.21 UTCTime 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. Note that UTCTime does not hold a standard textual representation of Universal Coordinated Time (UTC). However, UTC (to a precision of one second) may be derived from it.

CLUSTER_ID

```
public static final short CLUSTER_ID = 232
```

2.5.2.22 Cluster ID This type represents a cluster identifier as defined in spec. See reference on top of this class.

ATTRIBUTE_ID

```
public static final short ATTRIBUTE_ID = 233
```

2.5.2.23 Attribute ID This type represents an attribute identifier as defined in spec. See reference on top of this class.

BACNET_OID

```
public static final short BACNET_OID = 234
```

2.5.2.24 BACnet OID (Object Identifier) The BACnet OID data type is included to allow interworking with BACnet. The format is described in the referenced standard. See reference on top of this class.

IEEE_ADDRESS

```
public static final short IEEE_ADDRESS = 240
```

2.5.2.25 IEEE Address The IEEE Address data type is a 64-bit IEEE address that is unique to every ZigBee device. A value of 0xffffffffffffff indicates that the address is unknown.

ONE_HUNDRED_TWENTY_EIGHT_BIT_SECURITY_KEY

```
public static final short ONE_HUNDRED_TWENTY_EIGHT_BIT_SECURITY_KEY = 241
```

2.5.2.26 128-bit Security Key The 128-bit Security Key data type is for use in ZigBee security, and may take any 128-bit value.

Constructor Detail

ZigBeeDataTypes

```
public ZigBeeDataTypes()
```

Method Detail

encode

```
public static void encode(short type,
                          Object value,
                          ByteArrayOutputStream outdata)
```

Parameters:

type - the value's type

value - the Java value

outdata - `ByteArrayOutputStream` in which the array of bytes that represents the encoded value of param will be added.

decode

```
public static Object decode(short type,
                            ByteArrayInputStream value)
```

Parameters:

`type` - the value's type

`value` - the `ByteArrayInputStream` value

Returns:

the given value decoded as a Java Object

Interface ZigBeeEndpoint

org.osgi.service.zigbee

```
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.

Version:
1.0

Field Summary		Page
String	DEVICE_CATEGORY Constant used by all ZigBee devices indicating the device category.	85
String	DEVICE_ID Key of the String property containing the DeviceId of the device It is mandatory property for this service	85
String	DEVICE_VERSION Key of the String property containing the DeviceVersion of the device It is mandatory property for this service	85
String	ENDPOINT_ID Key of the String property containing the EndPoint Address of the device It is mandatory property for this service	84
String	HOST_PID Key of String containing the ZigBeeHost 's pid.	84
String	INPUT_CLUSTERS Key of the int array of containing the ids of each input cluster It is mandatory property for this service	85
String	OUTPUT_CLUSTERS Key of the int array of containing the ids of each output cluster It is mandatory property for this service	85
String	PROFILE_ID Key of the String property containing the profile id implemented by the device.	84
String	ZIGBEE_EXPORT Key of the String property mentioning that an endpoint is an exported one or not.	85

Method Summary		Page
void	bind (String servicePid, int clusterId, ZigBeeHandler handler) This method modify the <i>Binding Table</i> of physical device by adding the following entry: <pre>this.getNodeAddress(), this.getId(), clusterId, device.getNodeAddress(), device.getId()</pre> 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: SUCCESS, ILLEGAL_REQUEST, TABLE_FULL, NOT_SUPPORTED (see APSEException).	87
void	getBoundEndPoints (int clusterId, ZigBeeHandler handler) This method is used to get bound endpoints (identified by their service PIDs).	88

ZCLCluster	getClientCluster (int clientClusterId)	86
ZCLCluster []	getClientClusters ()	86
int	getId ()	85
Long	getNodeAddress ()	86
ZCLCluster	getServerCluster (int serverClusterId)	86
ZCLCluster []	getServerClusters ()	86
void	getSimpleDescriptor (ZigBeeHandler handler) 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_desc request can have the following status: SUCCESS, INVALID_EP, NOT_ACTIVE, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR.	86
void	notExported (ZCLEException e) This method is used to get details about problems when an error occurs during exporting an endpoint	87
void	unbind (String servicePid, int clusterId, ZigBeeHandler handler) This method modify the <i>Binding Table</i> of physical device by removing the entry if exists: <pre>this.getNodeAddress(), this.getId(), clusterId, device.getNodeAddress(), device.getId()</pre> 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: SUCCESS, ILLEGAL_REQUEST, INVALID_BINDING (see APSEException).	87

Field Detail

ENDPOINT_ID

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

Key of the String property containing the EndPoint Address of the device
It is mandatory property for this service

PROFILE_ID

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

Key of the String property containing the profile id implemented by the device.
It is mandatory property for this service

HOST_PID

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

Key of String containing the [ZigBeeHost](#)'s pid.
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 nodes that belong to a specific network MUST specify the value of the associated host number. It is mandatory for imported endpoints, optional for exported endpoints.

DEVICE_ID

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

Key of the String property containing the DeviceId of the device
It is mandatory property for this service

DEVICE_VERSION

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

Key of the String property containing the DeviceVersion of the device
It is mandatory property for this service

INPUT_CLUSTERS

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

Key of the int array of containing the ids of each input cluster
It is mandatory property for this service

OUTPUT_CLUSTERS

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

Key of the int array of containing the ids of each output cluster
It is mandatory property for this service

ZIGBEE_EXPORT

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

Key of the String property mentioning that an endpoint is an exported one or not. It is an optional property for this service.

DEVICE_CATEGORY

```
public static final String DEVICE_CATEGORY = "ZigBee"
```

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

Method Detail

getId

```
int getId()
```

Returns:

identifier of the endpoint represented by this object, value ranges from 1 to 240.

getNodeAddress

Long getNodeAddress()

Returns:

The IEEE Address of the node containing this endpoint

getSimpleDescriptor

void getSimpleDescriptor([ZigBeeHandler](#) handler)

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_desc request can have the following status: SUCCESS, INVALID_EP, NOT_ACTIVE, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR.

Parameters:

handler - that will be used in order to return the node simple descriptor [ZigBeeSimpleDescriptor](#).

getServerClusters

[ZCLCluster](#)[] getServerClusters()

Returns:

An array of servers(inputs) clusters, returns an empty array if does not provides any servers clusters.

getServerCluster

[ZCLCluster](#) getServerCluster(int serverClusterId)

Parameters:

serverClusterId - The server(input) cluster identifier

Returns:

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

getClientClusters

[ZCLCluster](#)[] getClientClusters()

Returns:

An array of clients(outputs) clusters, returns an empty array if does not provides any clients clusters.

getClientCluster

[ZCLCluster](#) getClientCluster(int clientClusterId)

Parameters:

clientClusterId - The client(output) cluster identifier

Returns:

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

bind

```
void bind(String servicePid,  
          int clusterId,  
          ZigBeeHandler handler)
```

This method modify the *Binding Table* of physical device by adding the following entry:

```
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: SUCCESS, ILLEGAL_REQUEST, TABLE_FULL, NOT_SUPPORTED (see [APSEException](#)). The response object given to the handler is an int that corresponds to one of the APSEException' codes.

Parameters:

servicePid - to bound to

clusterId - the cluster identifier to bound to

unbind

```
void unbind(String servicePid,  
            int clusterId,  
            ZigBeeHandler handler)
```

This method modify the *Binding Table* of physical device by removing the entry if 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: SUCCESS, ILLEGAL_REQUEST, INVALID_BINDING (see [APSEException](#)). The response object given to the handler is an int that corresponds to one of the APSEException' codes.

Parameters:

servicePid - to unbound from

clusterId - The cluster identifier to unbound from

notExported

```
void notExported(ZCLEException e)
```

This method is used to get details about problems when an error occurs during exporting an endpoint

Parameters:

e - A device [ZCLEException](#) the occurred exception

getBoundEndpoints

```
void getBoundEndpoints(int clusterId,  
                       ZigBeeHandler handler)
```

This method is used to get bound endpoints (identified by their service PIDs). It is implemented on the base driver with Mgmt_Bind_req command. It is implemented without a command request in local endpoints. If the local method or command request is not supported, then an exception with the following reason is thrown: GENERAL_COMMAND_NOT_SUPPORTED. If the method fails to retrieve the full binding table (that could require several Mgmt_Bind_req command), then an exception with the error code that was sent on the last response is thrown. 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: NOT_SUPPORTED or any status code returned from the APSME-GET.confirm primitive (see [APSEException](#)).

The response object given to the handler is a List containing the service.PIDs.

Interface ZigBeeEvent

org.osgi.service.zigbee

public interface ZigBeeEvent

This interface represents events generated by a ZigBee Device node

Version:

1.0

Method Summary

	Pag e
int getAttributeId()	89
int getClusterId()	89
int getEndpointId()	89
Long getIEEEAddress()	89
Object getValue()	90

Method Detail

getIEEEAddress

Long [getIEEEAddress\(\)](#)

Returns:

The ZigBee device node IEEE Address.

getEndpointId

int [getEndpointId\(\)](#)

Returns:

The endpoint identifier.

getClusterId

int [getClusterId\(\)](#)

Returns:

The cluster id.

getAttributeId

int [getAttributeId\(\)](#)

Returns:

the attribute identifier (i.e. the attribute's ID)

getValue

Object getValue()

Returns:

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

Interface ZigBeeGroup

org.osgi.service.zigbee

```
public interface ZigBeeGroup
```

This interface represents a ZigBee Group

Version:

1.0

Field Summary

		Page
String	ID	91
Key of the String containing the Group Address of the device.		

Method Summary

		Page
short	getGroupAddress()	91
void	invoke (Integer clusterId, ZCLFrame frame, ZCLCommandHandler handler)	92
Invokes the action on a Group.		
void	invoke (Integer clusterId, ZCLFrame frame, ZCLCommandHandler handler, String exportedServicePID)	93
This method is to be used by applications when the targeted device has to distinguish between source endpoints of the message.		
void	joinGroup (String pid, ZCLCommandHandler handler)	92
This method is used for adding an Endpoint to a Group, it may be invoked on exported Endpoint or even on import Endpoint.		
void	leaveGroup (String pid, ZCLCommandHandler handler)	92
This method is used for adding an Endpoint to a Group, it may be invoked on exported Endpoint or even on import Endpoint.		

Field Detail

ID

```
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.

Method Detail

getGroupAddress

```
short getGroupAddress()
```

Returns:
The 16bit group address.

joinGroup

```
void joinGroup(String pid,  
               ZCLCommandHandler handler)
```

This method is used for adding an Endpoint to a Group, it may be invoked on exported Endpoint or even on import Endpoint. In the former case, the ZigBee Base Driver should rely on the *APSME-ADD-GROUP* API defined by the ZigBee Specification, in the former case 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, a `add_group` request can have the following status: `SUCCESS`, `INVALID_PARAMETER` or `TABLE_FULL` (see [APSEException](#)).

Parameters:

`pid` - String representing the PID (see `org.osgi.framework.Constants.SERVICE_PID`) of the [ZigBeeEndpoint](#) that we want add to this Group.
`handler` - the handler that will notified of the result of "joining".

leaveGroup

```
void leaveGroup(String pid,  
                ZCLCommandHandler handler)
```

This method is used for adding an Endpoint to a Group, it may be invoked on exported Endpoint or even on import Endpoint. In the former case, the ZigBee Base Driver should rely on the *APSME-REMOVE-GROUP* API defined by the ZigBee Specification, in the former case 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: `SUCCESS`, `INVALID_GROUP` or `INVALID_PARAMETER` (see [APSEException](#)).

Parameters:

`pid` - String representing the PID (see `org.osgi.framework.Constants.SERVICE_PID`) of the [ZigBeeEndpoint](#) that we want leave to this Group.
`handler` - the handler that will notified of the result of "leaving".

invoke

```
void invoke(Integer clusterId,  
            ZCLFrame frame,  
            ZCLCommandHandler handler)
```

Invokes the action on a Group. The handler will provide the invocation response in an asynchronously 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.

Parameters:

`clusterId` - a cluster identifier.
`frame` - a command frame sequence.
`handler` - The handler that manages the command response.

invoke

```
void invoke(Integer clusterId,  
            ZCLFrame frame,  
            ZCLCommandHandler handler,  
            String exportedServicePID)
```

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.

Parameters:

`clusterId` - a cluster identifier.

`frame` - a command frame sequence.

`handler` - The handler that manages the command response.

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

Interface ZigBeeHandler

org.osgi.service.zigbee

public interface ZigBeeHandler

ZigBeeHandler manages response of a request to the Base Driver

Version:

1.0

Method Summary

		Page
void	onFailure (Exception e) Notifies the failure result of the call.	94
void	onSuccess (Object response) Notifies the success result of the call.	94

Method Detail

onSuccess

void onSuccess(Object response)

Notifies the success result of the call. This method is used when the handler command result is a success.

Parameters:

response - contains the results of the call.

onFailure

void onFailure(Exception e)

Notifies the failure result of the call. This method is used when the handler command result is a failure.

Parameters:

e - the exception.

Interface ZigBeeHost

org.osgi.service.zigbee

All Superinterfaces:

[ZigBeeNode](#)

```
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 basedriver (below/under the OSGi execution environment).

ZigBeeHost is more than a ZigBeeNode. It must be registered as a OSGi service.

Version:

1.0

Fields inherited from interface [org.osgi.service.zigbee.ZigBeeNode](#)

[COORDINATOR_TYPE](#), [END_DEVICE_TYPE](#), [EXTENDED_PAN_ID](#), [IEEE_ADDRESS](#), [LOGICAL_TYPE](#), [MANUFACTURER_CODE](#), [PAN_ID](#), [POWER_SOURCE](#), [RECEIVER_ON_WHEN_IDLE](#), [ROUTER_TYPE](#)

Method Summary

		Page
void	broadcast (Integer clusterID, ZCLFrame frame, ZCLCommandHandler handler) Enable to broadcast a given frame on a given cluster.	100
void	broadcast (Integer clusterID, ZCLFrame frame, ZCLCommandHandler handler, String exportedServicePID) Enable to broadcast a given frame on a given cluster.	100
void	createGroupService (String pid, int groupAddress, ZCLCommandHandler handler) This method is used for creating a ZigBeeGroup service that has not yet been discovered by the ZigBee Base Driver or that does not exist on the ZigBee network yet.	99
int	getChannel ()	98
int	getChannelMask ()	98
String	getNetworkKey ()	99
int	getSecurityLevel ()	99
boolean	isStarted () Get the host's start/stop state.	96
void	permitJoin (short duration) Indicates if a ZigBee device can join the network.	97
void	refreshNetwork () Updates the list of devices in the network by adding the new devices that joined the network and removing the devices that left the network since the last refresh.	99
void	setChannel (byte channel) Sets the network channel. 802.15.4 and ZigBee break the 2.4Ghz band into 16 channels, numbered from 11 to 26.	98
void	setChannelMask (int mask) Set a new configured channel mask.	98
void	setExtendedPanId (long extendedPanId) Set the extendedPanId.	97

void	setLogicalType (short logicalNodeType) Sets the host logical node type.	97
void	setPanId (int panId) Set the panId.	97
void	start () Starts the host.	96
void	stop () Stops the host.	96

Methods inherited from interface org.osgi.service.zigbee.[ZigBeeNode](#)

[getComplexDescriptor](#), [getExtendedPanId](#), [getHostPid](#), [getIEEEAddress](#), [getLinksQuality](#),
[getNetworkAddress](#), [getNodeDescriptor](#), [getPanId](#), [getPowerDescriptor](#), [getRoutingTable](#),
[getUserDescriptor](#), [leave](#), [leave](#)

Method Detail**start**

```
void start()  
    throws Exception
```

Starts the host. If the host is a Coordinator, then it can be started with or without PAN_ID and Extended PAN_ID (i.e. 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 router, or an end device, then the host may start without a registered PAN_ID property; the property will be set when the host will find and join a ZigBee network. The host status must be persistent, i.e. 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

stop

```
void stop()  
    throws Exception
```

Stops the host.

Throws:
 Exception

isStarted

```
boolean isStarted()
```

Get the host's start/stop state.

Returns:
 true if the host is started.

setPanId

```
void setPanId(int panId)
```

Set the panId.

Parameters:

panId - The network Personal Area Network identifier (PAND ID)

setExtendedPanId

```
void setExtendedPanId(long extendedPanId)
```

Set the extendedPanId.

Parameters:

extendedPanId - The network Extended PAN identifier(EPID)

permitJoin

```
void permitJoin(short duration)  
    throws Exception
```

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: SUCCESS, INVALID_REQUEST, NOT_AUTHORIZED or any status code returned from the NLMEPERMITJOINING.confirm primitive.

Parameters:

duration - The time during which associations are permitted.

Throws:

Exception

setLogicalType

```
void setLogicalType(short logicalNodeType)  
    throws Exception
```

Sets the host logical node type. The logical type will then be available when the host will restart. ZigBee defines three different types of node, coordinator(-> [COORDINATOR](#)), router([ROUTER](#)) and end device(-> [END_DEVICE](#))

Parameters:

logicalNodeType - The logical node type.

Throws:

Exception

getChannel

```
int getChannel()  
    throws Exception
```

Returns:

The current network channel.

Throws:

Exception

setChannel

```
void setChannel(byte channel)  
    throws IOException,  
        IllegalStateException
```

Sets the network channel. 802.15.4 and ZigBee break the 2.4Ghz band into 16 channels, numbered from 11 to 26. 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: SUCCESS, INVALID_PARAMETER or UNSUPPORTED_ATTRIBUTE (see [APSEException](#)).

Parameters:

channel - The network channel.

Throws:

IOException - for serial communication exception.

IllegalStateException - for state issues in the dongle depending if it is a coordinator and of coordinator capabilities.

getChannelMask

```
int getChannelMask()  
    throws Exception
```

Returns:

The currently configured channel mask.

Throws:

Exception

setChannelMask

```
void setChannelMask(int mask)  
    throws IOException,  
        IllegalStateException
```

Set 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: SUCCESS, INVALID_PARAMETER or UNSUPPORTED_ATTRIBUTE (see [APSEException](#)).

Parameters:

mask - A value representing the channel mask.

Throws:

IOException - for serial communication exception.

IllegalStateException - for state issues in the dongle depending if it is a coordinator and of coordinator capabilities.

refreshNetwork

```
void refreshNetwork()  
    throws Exception
```

Updates the list of devices in the network by adding the new devices that joined the network and removing the devices that left the network since the last refresh.

Throws:
Exception

getSecurityLevel

```
int getSecurityLevel()  
    throws Exception
```

Returns:
The network security level, i.e. 0 if security is disabled, an int code if enabled.

Throws:
Exception

getNetworkKey

```
String getNetworkKey()  
    throws Exception
```

Returns:
The current Network key.

Throws:
Exception

createGroupService

```
void createGroupService(String pid,  
                        int groupAddress,  
                        ZCLCommandHandler handler)
```

This method is used for creating a [ZigBeeGroup](#) service that has not yet been discovered by the ZigBee Base Driver or that does not exist on the ZigBee network yet. The method may be invoked on exported endpoint or even on import endpoint. In the former case, the ZigBee Base Driver should rely on the *APSME-ADD-GROUP* API defined by the ZigBee Specification, in the former case it will use the proper commands of the *Groups* cluster of the ZigBee Specification Library

Parameters:

pid - String representing the PID (see `org.osgi.framework.Constants.SERVICE_PID`) of the [ZigBeeEndpoint](#) that we want leave to this Group.
groupAddress - the address of the group to create.
handler - the [ZCLCommandHandler](#) that will be notified of the result of "creation".

broadcast

```
void broadcast(Integer clusterID,  
               ZCLFrame frame,  
               ZCLCommandHandler handler)
```

Enable to broadcast a given frame on a given cluster.

Parameters:

`clusterID` - the cluster ID.
`frame` - a command frame sequence.
`handler` - The handler that manages the command response.

broadcast

```
void broadcast(Integer clusterID,  
               ZCLFrame frame,  
               ZCLCommandHandler handler,  
               String exportedServicePID)
```

Enable to broadcast a given frame on a given cluster.

Parameters:

`clusterID` - the cluster ID.
`frame` - a command frame sequence.
`handler` - The handler that manages the command response.
`exportedServicePID` - : the source endpoint of the command request. In targeted situations, the source endpoint is the valid service PID of an exported endpoint.

Interface ZigBeeLinkQuality

org.osgi.service.zigbee

```
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)

Version:

1.0

Field Summary			Page
int	CHILD_NEIGHBOR	Constant value representing a child relationship between current ZigBeeNode and the neighbor	101
int	OTHERS_NEIGHBOR	Constant value representing a others relationship between current ZigBeeNode and the neighbor	102
int	PARENT_NEIGHBOR	* Constant value representing a parent relationship between current ZigBeeNode and the neighbor	101
int	PREVIOUS_CHILD_NEIGHBOR	Constant value representing a previous child relationship between current ZigBeeNode and the neighbor	102
int	SIBLING_NEIGHBOR	Constant value representing a sibling relationship between current ZigBeeNode and the neighbor	102

Method Summary			Page
int	getDepth()	See the Depth field of the (NeighborTableList Record Format).	102
int	getLQI()	See the LQI field of the (NeighborTableList Record Format).	102
String	getNeighbor()		102
int	getRelationship()	See the Relationship field of the (NeighborTableList Record Format).	103

Field Detail

PARENT_NEIGHBOR

```
public static final int PARENT_NEIGHBOR = 0
```

* Constant value representing a parent relationship between current [ZigBeeNode](#) and the neighbor

CHILD_NEIGHBOR

```
public static final int CHILD_NEIGHBOR = 1
```

Constant value representing a child relationship between current [ZigBeeNode](#) and the neighbor

SIBLING_NEIGHBOR

```
public static final int SIBLING_NEIGHBOR = 2
```

Constant value representing a sibling relationship between current [ZigBeeNode](#) and the neighbor

OTHERS_NEIGHBOR

```
public static final int OTHERS_NEIGHBOR = 3
```

Constant value representing a others relationship between current [ZigBeeNode](#) and the neighbor

PREVIOUS_CHILD_NEIGHBOR

```
public static final int PREVIOUS_CHILD_NEIGHBOR = 4
```

Constant value representing a previous child relationship between current [ZigBeeNode](#) and the neighbor

Method Detail

getNeighbor

```
String getNeighbor()
```

Returns:

the Service.PID referring to the [ZigBeeNode](#) representing neighbor

getLQI

```
int getLQI()
```

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\(\)](#)

getDepth

```
int getDepth()
```

See the Depth field of the (NeighborTableList Record Format).

Returns:

the tree-depth of device

getRelationship

```
int getRelationship()
```

See the Relationship field of the (NeighborTableList Record Format).

Returns:

the relationship between [ZigBeeNode](#) returning this and the [ZigBeeNode](#) identified by the [getNeighbor\(\)](#)

Interface ZigBeeNode

org.osgi.service.zigbee

All Known Subinterfaces:

[ZigBeeHost](#)

```
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
-	16-bit	ZigBee	Network	Address

Version:

1.0

Field Summary			Page
short	COORDINATOR_TYPE ZigBee coordinator type		106
short	END_DEVICE_TYPE ZigBee end device type		106
String	EXTENDED_PAN_ID Key of String containing the device node network extended PAN ID.		106
String	IEEE_ADDRESS Property key for the mandatory node IEEE Address representing node MAC address.		105
String	LOGICAL_TYPE Property key for the device logical type		105
String	MANUFACTURER_CODE Property key for a manufacturer code that is allocated by the ZigBee Alliance, relating the manufacturer to the device.		106
String	PAN_ID Key of String containing the device node network PAN ID		106
String	POWER_SOURCE ZigBee power source, i.e. 3rd bit of "MAC Capabilities" in Node Descriptor.		106
String	RECEIVER_ON_WHEN_IDLE ZigBee receiver on when idle, i.e. 4th bit of "MAC Capabilities" in Node Descriptor.		106
short	ROUTER_TYPE ZigBee router type		106

Method Summary			Page
void	getComplexDescriptor (ZigBeeHandler handler) As described in "Table 2.96 Fields of the Complex_Desc_rsp Command" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a complex_desc request can have the following status: SUCCESS, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR.		108

long	getExtendedPanId()	107
String	getHostPid()	107
Long	getIEEEAddress()	107
void	getLinksQuality() (ZigBeeHandler handler) The ZigBee Base Drive may use the Mgmt_Lqi_req / Mgmt_Lqi_rsp messages to retrieve the Link Quality table (i.e also known as NeighborTableList in the ZigBee Specification).	108
int	getNetworkAddress()	107
void	getNodeDescriptor() (ZigBeeHandler handler) As described in "Table 2.91 Fields of the Node_Desc_rsp Command" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a node_desc request can have the following status: SUCCESS, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR.	107
int	getPanId()	107
void	getPowerDescriptor() (ZigBeeHandler handler) As described in "Table 2.92 Fields of the Power_Desc_rsp Command" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a power_desc request can have the following status: SUCCESS, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR.	108
void	getRoutingTable() (ZigBeeHandler handler) The ZigBee Base Drive may use the Mgmt_Rtg_req / Mgmt_Rtg_rsp messages to retrieve the Routing Table (i.e also known as RoutingTableList in the ZigBee Specification).	109
void	getUserDescriptor() (ZigBeeHandler handler) As described in "Table 2.97 Fields of the User_Desc_rsp Command" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a user_desc request can have the following status: SUCCESS, NOT_SUPPORTED, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR.	108
void	leave() (boolean rejoin, boolean removeChildren, ZigBeeHandler handler) Requests the device to leave the network.	109
void	leave() (ZigBeeHandler handler) Request to leave the network.	109

Field Detail

IEEE_ADDRESS

```
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. There is no need to use 0x hexadecimal notation. *i.e* `zigbee.node.ieee.address="00:25:96:AB:37:56"` for a 48-bit address and *i.e* `zigbee.node.ieee.address="00:25:96:FF:FE:AB:37:56"` for a 64-bit address A ZigBee Event Listener service can announce for what ZigBee device nodes it wants notifications.

LOGICAL_TYPE

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

Property key for the device logical type

MANUFACTURER_CODE

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

Property key for a manufacturer code that is allocated by the ZigBee Alliance, relating the manufacturer to the device.

PAN_ID

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

Key of String containing the device node network PAN ID

EXTENDED_PAN_ID

```
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.

POWER_SOURCE

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

ZigBee power source, i.e. 3rd bit of "MAC Capabilities" in Node Descriptor. Set to 1 if the current power source is mains power, set to 0 otherwise.

RECEIVER_ON_WHEN_IDLE

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

ZigBee receiver on when idle, i.e. 4th bit of "MAC Capabilities" in Node Descriptor. Set to 1 if the device does not disable its receiver to conserve power during idle periods, set to 0 otherwise.

COORDINATOR_TYPE

```
public static final short COORDINATOR_TYPE = 0
```

ZigBee coordinator type

ROUTER_TYPE

```
public static final short ROUTER_TYPE = 1
```

ZigBee router type

END_DEVICE_TYPE

```
public static final short END_DEVICE_TYPE = 2
```

ZigBee end device type

Method Detail

getIEEEAddress

```
Long getIEEEAddress()
```

Returns:
The ZigBee device node IEEE Address.

getNetworkAddress

```
int getNetworkAddress()
```

Returns:
The ZigBee device node current network address.

getHostPid

```
String getHostPid()
```

Returns:
The ZigBee Host OSGi service PID.

getPanId

```
int getPanId()
```

Returns:
The network Personal Area Network identifier(PAND ID)

getExtendedPanId

```
long getExtendedPanId()
```

Returns:
The network Extended PAN identifier(EPID)

getNodeDescriptor

```
void getNodeDescriptor(ZigBeeHandler handler)
```

As described in "Table 2.91 Fields of the Node_Desc_rsp Command" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a node_desc request can have the following status: SUCCESS, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR.

Parameters:
handler - that will be used in order to return the node descriptor [ZigBeeNodeDescriptor](#).

getPowerDescriptor

```
void getPowerDescriptor(ZigBeeHandler handler)
```

As described in "Table 2.92 Fields of the Power_Desc_rsp Command" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a power_desc request can have the following status: SUCCESS, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR.

Parameters:

handler - that will be used in order to return the node power descriptor [ZigBeePowerDescriptor](#).

getComplexDescriptor

```
void getComplexDescriptor(ZigBeeHandler handler)
```

As described in "Table 2.96 Fields of the Complex_Desc_rsp Command" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a complex_desc request can have the following status: SUCCESS, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR.

Parameters:

handler - that will be used in order to return the node complex descriptor [ZigBeeComplexDescriptor](#). Can be null if complex descriptor is not provided.

getUserDescriptor

```
void getUserDescriptor(ZigBeeHandler handler)
```

As described in "Table 2.97 Fields of the User_Desc_rsp Command" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a user_desc request can have the following status: SUCCESS, NOT_SUPPORTED, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR.

Parameters:

handler - that will be used in order to return the node user descriptor [ZigBeeUserDescriptor](#). Can be null if user descriptor is not provided.

getLinksQuality

```
void getLinksQuality(ZigBeeHandler handler)
```

The ZigBee Base Drive may use the Mgmt_Lqi_req / Mgmt_Lqi_rsp messages to retrieve the Link Quality table (i.e also known as NeighborTableList in the ZigBee Specification). The method limit the Link Quality table to the [ZigBeeNode](#) service discovered. The target device may report error code NOT_SUPPORTED, or UNSUPPORTED_ATTRIBUTE in case of failure that will be notified to the handler.

Parameters:

handler - that will be notified with the result of this operation. In case of success, the object notified with [ZigBeeHandler.onSuccess\(Object\)](#) will be a Map containing the Service.PID as String key of the [ZigBeeNode](#) service and the value the [ZigBeeLinkQuality](#) for that node.

getRoutingTable

```
void getRoutingTable(ZigBeeHandler handler)
```

The ZigBee Base Drive may use the Mgmt_Rtg_req / Mgmt_Rtg_rsp messages to retrieve the Routing Table (i.e also known as RoutingTableList in the ZigBee Specification). The target device may report error code NOT_SUPPORTED, or UNSUPPORTED_ATTRIBUTE in case of failure that will be notified to the handler.

Parameters:

handler - that will notified with the result of this operation. In case of success, the object notified with [ZigBeeHandler.onSuccess\(Object\)](#) will be a Map containing the Service.PID as String key of the [ZigBeeNode](#) service and the value the [ZigBeeRoute](#) for that node.

leave

```
void leave(ZigBeeHandler handler)
```

Request to leave the network. As described in "Table 2.131 Fields of the Mgmt_Leave_rsp Command" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a mgmt_leave request can have the following status: NOT_SUPPORTED, NOT_AUTHORIZED or any status code returned from the NLMELEAVE.confirm primitive (see [ZDPEException](#)). The response object given to the handler is a short that corresponds to one of the ZDPEException' codes.

leave

```
void leave(boolean rejoin,  
           boolean removeChildren,  
           ZigBeeHandler handler)
```

Requests the device to leave the network. The ZigBeeHandler onSuccess method is called if and only if the ZigBeeDeviceNode has been removed. As described in "Table 2.131 Fields of the Mgmt_Leave_rsp Command" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a mgmt_leave request can have the following status: NOT_SUPPORTED, NOT_AUTHORIZED or any status code returned from the NLMELEAVE.confirm primitive (see [ZDPEException](#)). The response object given to the handler is a short that corresponds to one of the ZDPEException' codes.

Parameters:

rejoin - This field has a value of 1 if the device being asked to leave from the current parent is requested to rejoin the network. Otherwise, it has a value of 0.
removeChildren - This field has a value of 1 if the device being asked to leave the network is also being asked to remove its child devices, if any. Otherwise, it has a value of 0.
handler - The handler

Interface ZigBeeRoute

org.osgi.service.zigbee

```
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)

Version:

1.0

Field Summary		Page
int	ACTIVE Constant value representing an active route	110
int	DISCOVERY_FAILED Constant value representing a failed route discovery	110
int	DISCOVERY_UNDERWAY Constant value representing a route that is under discovery	110
int	INACTIVE Constant value representing an inactive route	111
int	VALIDATION_UNDERWAY Constant value representing a route which is under validation	111

Method Summary		Page
String	getDestination()	111
String	getNextHop()	111
int	getStatus()	111

Field Detail

ACTIVE

```
public static final int ACTIVE = 0
```

Constant value representing an active route

DISCOVERY_UNDERWAY

```
public static final int DISCOVERY_UNDERWAY = 1
```

Constant value representing a route that is under discovery

DISCOVERY_FAILED

```
public static final int DISCOVERY_FAILED = 2
```

Constant value representing a failed route discovery

INACTIVE

```
public static final int INACTIVE = 3
```

Constant value representing an inactive route

VALIDATION_UNDERWAY

```
public static final int VALIDATION_UNDERWAY = 4
```

Constant value representing a route which is under validation

Method Detail

getDestination

```
String getDestination()
```

Returns:

the Service.PID of the [ZigBeeNode](#) as destination of this route entry

getNextHop

```
String getNextHop()
```

Returns:

the Service.PID of the [ZigBeeNode](#) to send the data for reaching the destination

getStatus

```
int getStatus()
```

Returns:

the status of the RoutingLink as defined by ZigBee Specification: ACTIVE, DISCOVERY_UNDERWAY, DISCOVERY_FAILED, INACTIVE, VALIDATION_UNDERWAY

Package org.osgi.service.zigbee.descriptions

Interface Summary		Page
<u>ZCLAttributeDescription</u>	This interface represents a ZCLAttributeDescription	113
<u>ZCLClusterDescription</u>	This interface represents a ZCL Cluster description	116
<u>ZCLCommandDescription</u>	This interface represents a ZCLCommandDescription	118
<u>ZCLDataTypeDescription</u>	This interface represents the ZCL data type abstraction.	121
<u>ZCLGlobalClusterDescription</u>	This interface represents Cluster global description	123
<u>ZCLParameterDescription</u>	This interface represents a ZigBee parameter description	125
<u>ZigBeeDeviceDescription</u>	This interface represents a ZigBee device description	126
<u>ZigBeeDeviceDescriptionSet</u>	This interface represents a ZigBee Device description Set.	128

Interface ZCLAttributeDescription

[org.osgi.service.zigbee.descriptions](#)

```
public interface ZCLAttributeDescription
```

This interface represents a ZCLAttributeDescription

Version:

1.0

Method Summary		Page
boolean	checkValue (Object value) checks whether the value object is conform to the attribute data type description	114
ZCLDataTypeDescription	getDataTypeDescription ()	114
Object	getDefaultValue ()	114
int	getId ()	113
String	getName ()	113
String	getShortDescription ()	113
boolean	isMandatory ()	114
boolean	isPartOfAScene ()	115
boolean	isReadOnly ()	114
boolean	isReportable ()	114

Method Detail

getId

```
int getId()
```

Returns:
the attribute identifier

getName

```
String getName()
```

Returns:
The attribute name

getShortDescription

```
String getShortDescription()
```

Returns:
The Attribute functional description

getDefaultValue

Object `getDefaultValue()`

Returns:
The attribute default value

isMandatory

boolean `isMandatory()`

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

isReportable

boolean `isReportable()`

Returns:
the true if and only if the attribute support subscription

isReadOnly

boolean `isReadOnly()`

Returns:
true if the attribute is read only, false otherwise (i.e. if the attribute is read/write or optionnaly writable (R*W))

getDataTypeDescription

[ZCLDataTypeDescription](#) `getDataTypeDescription()`

Returns:
A [ZCLDataTypeDescription](#) representing the attribute data type

checkValue

boolean `checkValue(Object value)`

checks whether the value object is conform to the attribute data type description

Parameters:
value - The value to check

Returns:
true if value is conform otherwise returns false

isPartOfAScene

`boolean isPartOfAScene()`

Returns:

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

Interface ZCLClusterDescription

org.osgi.service.zigbee.descriptions

public interface ZCLClusterDescription

This interface represents a ZCL Cluster description

Version:
1.0

Method Summary		Page
ZCLAttributeDescription[]	getAttributeDescriptions()	116
ZCLCommandDescription[]	getGeneratedCommandDescriptions()	116
ZCLGlobalClusterDescription	getGlobalClusterDescription()	117
int	getId()	116
ZCLCommandDescription[]	getReceivedCommandDescriptions()	116

Method Detail

getId

int getId()

Returns:
the cluster identifier

getGeneratedCommandDescriptions

[ZCLCommandDescription\[\]](#) getGeneratedCommandDescriptions()

Returns:
an array of cluster's generated command description

getReceivedCommandDescriptions

[ZCLCommandDescription\[\]](#) getReceivedCommandDescriptions()

Returns:
an array of cluster's received command description

getAttributeDescriptions

[ZCLAttributeDescription\[\]](#) getAttributeDescriptions()

Returns:
an array of cluster's Attributes description

getGlobalClusterDescription

[ZCLGlobalClusterDescription](#) getGlobalClusterDescription()

Returns:
an array of cluster's Commands description

Interface ZCLCommandDescription

org.osgi.service.zigbee.descriptions

public interface ZCLCommandDescription

This interface represents a ZCLCommandDescription

Version:
1.0

Method Summary		Page
Object[]	deserialize (ZCLFrame frame) Deserialize ZCLFrame to javaValues.	119
int	getId ()	118
int	getManufacturerCode () Get manufacturerCode Default value is: -1 (no code)	120
String	getName ()	118
ZCLParameterDescription []	getParameterDescriptions ()	119
String	getShortDescription ()	118
boolean	isClientServerDirection ()	120
boolean	isClusterSpecificCommand ()	119
boolean	isMandatory ()	119
ZCLFrame	serialize (ZCLHeader header, Object[] javaValues) Serialize javaValues to a ZCLFrame that can them be used in invocations (e.g. via ZigBeeCluster, or ZigBeeGroup).	119

Method Detail

getId

int getId()

Returns:
the command identifier

getName

String getName()

Returns:
the command name

getShortDescription

String getShortDescription()

Returns:
the command functional description

isMandatory

`boolean isMandatory()`

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

getParameterDescriptions

`ZCLParameterDescription[] getParameterDescriptions()`

Returns:
an array of command's parameters description

serialize

`ZCLFrame serialize(ZCLHeader header,
Object[] javaValues)`

Serialize javaValues to a ZCLFrame that can then be used in invocations (e.g. via ZigBeeCluster, or ZigBeeGroup).

Parameters:
header - the ZCLFrame's header
javaValues - ordered java values

Returns:
serialized javaValues as a byte[]

deserialize

`Object[] deserialize(ZCLFrame frame)`

Deserialize ZCLFrame to javaValues. This ZCLFrame is expected to be a result of an invocation. (e.g. via ZigBeeCluster, or ZigBeeGroup).

Parameters:
frame - the ZCLFrame

Returns:
deserialized Object[] as javaValues

isClusterSpecificCommand

`boolean isClusterSpecificCommand()`

Returns:
the isClusterSpecificCommand value

getManufacturerCode

```
int getManufacturerCode()
```

Get manufacturerCode Default value is: -1 (no code)

Returns:
the manufacturerCode

isClientServerDirection

```
boolean isClientServerDirection()
```

Returns:
the isClientServerDirection value

Interface ZCLDataTypeDescription

org.osgi.service.zigbee.descriptions

All Known Implementing Classes:

[ZCLAttributeID](#), [ZigBeeArray](#), [ZigBeeBacnetOID](#), [ZigBeeBag](#), [ZigBeeBitmap16](#), [ZigBeeBitmap24](#), [ZigBeeBitmap32](#), [ZigBeeBitmap40](#), [ZigBeeBitmap48](#), [ZigBeeBitmap56](#), [ZigBeeBitmap64](#), [ZigBeeBitmap8](#), [ZigBeeBoolean](#), [ZigBeeCharacterString](#), [ZigBeeClusterID](#), [ZigBeeDate](#), [ZigBeeEnumeration16](#), [ZigBeeEnumeration8](#), [ZigBeeFloatingDouble](#), [ZigBeeFloatingSemi](#), [ZigBeeFloatingSingle](#), [ZigBeeGeneralData16](#), [ZigBeeGeneralData24](#), [ZigBeeGeneralData32](#), [ZigBeeGeneralData40](#), [ZigBeeGeneralData48](#), [ZigBeeGeneralData56](#), [ZigBeeGeneralData64](#), [ZigBeeGeneralData8](#), [ZigBeeIEEEADDRESS](#), [ZigBeeLongCharacterString](#), [ZigBeeLongOctetString](#), [ZigBeeOctetString](#), [ZigBeeSet](#), [ZigBeeSignedInteger16](#), [ZigBeeSignedInteger24](#), [ZigBeeSignedInteger32](#), [ZigBeeSignedInteger40](#), [ZigBeeSignedInteger48](#), [ZigBeeSignedInteger56](#), [ZigBeeSignedInteger64](#), [ZigBeeSignedInteger8](#), [ZigBeeStructure](#), [ZigBeeTimeOfDay](#), [ZigBeeUnsignedInteger16](#), [ZigBeeUnsignedInteger24](#), [ZigBeeUnsignedInteger32](#), [ZigBeeUnsignedInteger40](#), [ZigBeeUnsignedInteger48](#), [ZigBeeUnsignedInteger56](#), [ZigBeeUnsignedInteger64](#), [ZigBeeUnsignedInteger8](#), [ZigBeeUTCTime](#)

```
public interface ZCLDataTypeDescription
```

This interface represents the ZCL data type abstraction.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	122
short	getId ()	121
Object	getInvalidNumber ()	122
Class	getJavaDataType ()	122
String	getName ()	121
boolean	isAnalog ()	122
void	serialize (Object param, ByteArrayOutputStream outdata)	122

Method Detail

getId

```
short getId()
```

Returns:
The data type identifier.

getName

```
String getName()
```

Returns:
The associated data type name string.

getInvalidNumber

`Object getInvalidNumber()`

Returns:

The data type invalid number if exists, otherwise returns null.

isAnalog

`boolean isAnalog()`

Returns:

true, if the data type is analog.

getJavaDataType

`Class getJavaDataType()`

Returns:

The corresponding Java type class.

serialize

`void serialize(Object param,
 ByteArrayOutputStream outdata)`

Parameters:

`param` - Object to be serialized using the associated type.

`outdata` - `ByteArrayOutputStream` in which the array of bytes that represents the serialized value of `param` will be added.

deserialize

`Object deserialize(ByteArrayInputStream data)`

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Interface ZCLGlobalClusterDescription

org.osgi.service.zigbee.descriptions

public interface ZCLGlobalClusterDescription

This interface represents Cluster global description

Version:
1.0

Method Summary		Page
ZCLClusterDescription	getClientClusterDescription()	124
String	getClusterDescription()	123
String	getClusterFunctionalDomain()	123
int	getClusterId()	123
String	getClusterName()	123
ZCLClusterDescription	getServerClusterDescription()	124

Method Detail

getClusterId

int getClusterId()

Returns:
the cluster identifier

getClusterName

String getClusterName()

Returns:
the cluster name

getClusterDescription

String getClusterDescription()

Returns:
the cluster functional description

getClusterFunctionalDomain

String getClusterFunctionalDomain()

Returns:
the cluster functional domain

getClientClusterDescription

[ZCLClusterDescription](#) getClientClusterDescription()

Returns:
a ZCLClusterDescription representing the client cluster description

getServerClusterDescription

[ZCLClusterDescription](#) getServerClusterDescription()

Returns:
a ZCLClusterDescription representing the server cluster description

Interface ZCLParameterDescription

org.osgi.service.zigbee.descriptions

```
public interface ZCLParameterDescription
```

This interface represents a ZigBee parameter description

Version:
1.0

Method Summary		Page
boolean	checkValue (Object value) checks whether the value object is conform to the parameter data type description	125
ZCLDataTypeDescription	getDataTypeDescription ()	125

Method Detail

getDataTypeDescription

```
ZCLDataTypeDescription getDataTypeDescription()
```

Returns:
the parameter data type

checkValue

```
boolean checkValue(Object value)
```

checks whether the value object is conform to the parameter data type description

Parameters:
value - The value to check

Returns:
true if value is conform otherwise returns false

Interface ZigBeeDeviceDescription

org.osgi.service.zigbee.descriptions

public interface ZigBeeDeviceDescription

This interface represents a ZigBee device description

Version:
1.0

Method Summary		Page
ZCLClusterDescription n[]	getClientClustersDescriptions()	127
int	getId()	126
String	getName()	126
int	getProfileId()	126
ZCLClusterDescription n[]	getServerClustersDescriptions()	127
Integer	getVersion()	126

Method Detail

getId

int getId()

Returns:
The device identifier.

getName

String getName()

Returns:
The device name.

getVersion

Integer getVersion()

Returns:
The device version.

getProfileId

int getProfileId()

Returns:
The profile identifier.

getServerClustersDescriptions

[ZCLClusterDescription](#)[] getServerClustersDescriptions()

Returns:
An array of server cluster description.

getClientClustersDescriptions

[ZCLClusterDescription](#)[] getClientClustersDescriptions()

Returns:
an array of client cluster description.

Interface ZigBeeDeviceDescriptionSet

org.osgi.service.zigbee.descriptions

```
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.

Version:
1.0

Field Summary		Page
String	DEVICES Property key for a comma separated list of devices identifiers supported by the set.	128
String	PROFILE_NAME Property key for a profile name.	128
String	VERSION Property key for a version of the application profile.	128

Method Summary		Page
ZigBeeDeviceDescription	getDeviceSpecification (int deviceId, short version)	129

Field Detail

VERSION

```
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.

PROFILE_NAME

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

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

DEVICES

```
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.

Method Detail

getDeviceSpecification

[ZigBeeDeviceDescription](#) getDeviceSpecification(int deviceId,
short version)

Parameters:

deviceId - Identifier of the device.

version - The version of the application profile.

Returns:

The associated device description.

Package org.osgi.service.zigbee.descriptors

Interface Summary		Page
<u>ZigBeeComplexDescriptor</u>	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.	131
<u>ZigBeeNodeDescriptor</u>	This interface represents a Node Descriptor as described in the ZigBee Specification The Node Descriptor contains information about the capabilities of the node.	133
<u>ZigBeePowerDescriptor</u>	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.	138
<u>ZigBeeSimpleDescriptor</u>	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.	142
<u>ZigBeeUserDescriptor</u>	This interface represents a User Descriptor as described in the ZigBee Specification The User Descriptor contains information that allows the user to identify the device using user-friendly character string.	144

Interface ZigBeeComplexDescriptor

org.osgi.service.zigbee.descriptors

```
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.

Version:

1.0

Method Summary		Page
String	getCharacterSetIdentifier()	131
String	getDeviceURL()	132
byte[]	getIcon()	132
String	getIconURL()	132
String	getLanguageCode()	131
String	getManufacturerName()	131
String	getModelName()	132
String	getSerialNumber()	132

Method Detail

getLanguageCode

```
String getLanguageCode()
```

Returns:

the language code used for character strings.

getCharacterSetIdentifier

```
String getCharacterSetIdentifier()
```

Returns:

the encoding used by characters in the character set.

getManufacturerName

```
String getManufacturerName()
```

Returns:

the manufacturer name field.

getModelName

`String getModelName()`

Returns:
the model name field

getSerialNumber

`String getSerialNumber()`

Returns:
the serial number field.

getDeviceURL

`String getDeviceURL()`

Returns:
the Device URL field.

getIcon

`byte[] getIcon()`

Returns:
the icon field.

getIconURL

`String getIconURL()`

Returns:
the icon field URL.

Interface ZigBeeNodeDescriptor

org.osgi.service.zigbee.descriptors

```
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.

Version:
1.0

Field Summary		Page
int	BACKUP_BINDING_TABLE_CACHE_MASK Backup Discovery Cache server mask.	134
int	BACKUP_DISCOVERY_CACHE_MASK Network Manager server mask.	135
int	BACKUP_TRUST_CENTER_MASK Backup Trust Center server mask.	134
short	FREQUENCY_RANGE_2400_2483_MASK Frequency mask for the range 2400 - 2483.5 MHz	134
short	FREQUENCY_RANGE_868_MASK Frequency mask for the range 868 - 868.6 MHz.	134
short	FREQUENCY_RANGE_902_928_MASK Frequency mask for the range 902 - 928 MHz	134
int	NETWORK_MANAGER_MASK Frequency mask for the range 2400 - 2483.5 MHz	135
int	PRIMARY_BINDING_TABLE_CACHE_MASK Primary Binding Table Cache server mask.	134
int	PRIMARY_DISCOVERY_CACHE_MASK Frequency mask for the range 2400 - 2483.5 MHz	135
int	PRIMARY_TRUST_CENTER_MASK Primary Trust Center server mask.	134

Method Summary		Page
short	getFrequencyBand()	135
Short	getLogicalType()	135
int	getManufacturerCode()	136
int	getMaxBufferSize()	136
int	getMaxIncomingTransferSize()	136
int	getMaxOutgoingTransferSize()	136
int	getServerMask()	136
boolean	isAddressAllocate()	137
boolean	isAlternatePANCoordinator()	136
boolean	isComplexDescriptorAvailable()	135
boolean	isExtendedActiveEndpointListAvailable()	137
boolean	isExtendedSimpleDescriptorListAvailable()	137
boolean	isFullFunctionDevice()	136

boolean	isMainsPower()	137
boolean	isReceiverOnWhenIdle()	137
boolean	isSecurityCapable()	137
boolean	isUserDescriptorAvailable()	135

Field Detail

FREQUENCY_RANGE_868_MASK

```
public static final short FREQUENCY_RANGE_868_MASK = 1
```

Frequency mask for the range 868 - 868.6 MHz.

FREQUENCY_RANGE_902_928_MASK

```
public static final short FREQUENCY_RANGE_902_928_MASK = 4
```

Frequency mask for the range 902 - 928 MHz

FREQUENCY_RANGE_2400_2483_MASK

```
public static final short FREQUENCY_RANGE_2400_2483_MASK = 8
```

Frequency mask for the range 2400 - 2483.5 MHz

PRIMARY_TRUST_CENTER_MASK

```
public static final int PRIMARY_TRUST_CENTER_MASK = 1
```

Primary Trust Center server mask.

BACKUP_TRUST_CENTER_MASK

```
public static final int BACKUP_TRUST_CENTER_MASK = 2
```

Backup Trust Center server mask.

PRIMARY_BINDING_TABLE_CACHE_MASK

```
public static final int PRIMARY_BINDING_TABLE_CACHE_MASK = 4
```

Primary Binding Table Cache server mask.

BACKUP_BINDING_TABLE_CACHE_MASK

```
public static final int BACKUP_BINDING_TABLE_CACHE_MASK = 8
```

Backup Discovery Cache server mask.

PRIMARY_DISCOVERY_CACHE_MASK

```
public static final int PRIMARY_DISCOVERY_CACHE_MASK = 16
```

Frequency mask for the range 2400 - 2483.5 MHz

BACKUP_DISCOVERY_CACHE_MASK

```
public static final int BACKUP_DISCOVERY_CACHE_MASK = 32
```

Network Manager server mask.

NETWORK_MANAGER_MASK

```
public static final int NETWORK_MANAGER_MASK = 64
```

Frequency mask for the range 2400 - 2483.5 MHz

Method Detail

getLogicalType

```
Short getLogicalType()
```

Returns:

one of: ZigBeeNode.COORDINATOR, ZigBeeNode.ROUTER, ZigBeeNode.END_DEVICE.

isComplexDescriptorAvailable

```
boolean isComplexDescriptorAvailable()
```

Returns:

true if a complex descriptor is available or false otherwise.

isUserDescriptorAvailable

```
boolean isUserDescriptorAvailable()
```

Returns:

true if a user descriptor is available or false otherwise.

getFrequencyBand

```
short getFrequencyBand()
```

Returns:

an int corresponding to the 5 bits which indicate the frequency range. use the frequency mask constants to get info, which are the ranges actually supported.

getManufacturerCode

```
int getManufacturerCode()
```

Returns:
the manufacurer code field.

getMaxBufferSize

```
int getMaxBufferSize()
```

Returns:
the maximum buffer size field.

getMaxIncomingTransferSize

```
int getMaxIncomingTransferSize()
```

Returns:
the maximum incoming transfer size field.

getMaxOutgoingTransferSize

```
int getMaxOutgoingTransferSize()
```

Returns:
the maximum outgoing transfer size field.

getServerMask

```
int getServerMask()
```

Returns:
the server mask field.

isAlternatePANCoordinator

```
boolean isAlternatePANCoordinator()
```

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

isFullFunctionDevice

```
boolean isFullFunctionDevice()
```

Returns:
true if this node a full function device false otherwise.

isMainsPower

`boolean isMainsPower()`

Returns:

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

isReceiverOnWhenIdle

`boolean isReceiverOnWhenIdle()`

Returns:

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

isSecurityCapable

`boolean isSecurityCapable()`

Returns:

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

isAddressAllocate

`boolean isAddressAllocate()`

Returns:

true if the device is address allocate or false otherwise.

isExtendedActiveEndpointListAvailable

`boolean isExtendedActiveEndpointListAvailable()`

Returns:

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

isExtendedSimpleDescriptorListAvailable

`boolean isExtendedSimpleDescriptorListAvailable()`

Returns:

true if extended simple descriptor is available or false otherwise.

Interface ZigBeePowerDescriptor

org.osgi.service.zigbee.descriptors

```
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.

Version:
1.0

Field Summary		Page
short	CONSTANT_POWER Power source: Constant (mains) power.	139
short	CRITICAL_LEVEL Current power source level: critical.	139
short	DISPOSABLE_BATTERY Power source: Disposable battery.	140
short	FULL_LEVEL Current power source level: 100%.	139
short	LOW_LEVEL Current power source level: 33%.	139
short	MIDDLE_LEVEL Current power source level: 66%.	139
short	POWER_MODE_PERIODIC The current power mode.	139
short	POWER_MODE_STIMULATED The current power mode.	139
short	POWER_MODE_SYNC Receiver synchronized with the receiver on when idle subfield of the node descriptor.	138
short	RECHARGEABLE_BATTERY Power source: Rechargeable battery.	139

Method Summary		Page
short	getCurrentPowerMode()	140
short	getCurrentPowerSource()	140
short	getCurrentPowerSourceLevel()	140
boolean	isConstantMainsPowerAvailable()	140
boolean	isDisposableBatteryAvailable()	140
boolean	isRechargeableBatteryAvailable()	140

Field Detail

POWER_MODE_SYNC

```
public static final short POWER_MODE_SYNC = 0
```

Receiver synchronized with the receiver on when idle subfield of the node descriptor.

POWER_MODE_PERIODIC

```
public static final short POWER_MODE_PERIODIC = 1
```

The current power mode.

POWER_MODE_STIMULATED

```
public static final short POWER_MODE_STIMULATED = 2
```

The current power mode.

CRITICAL_LEVEL

```
public static final short CRITICAL_LEVEL = 0
```

Current power source level: critical.

LOW_LEVEL

```
public static final short LOW_LEVEL = 4
```

Current power source level: 33%.

MIDDLE_LEVEL

```
public static final short MIDDLE_LEVEL = 8
```

Current power source level: 66%.

FULL_LEVEL

```
public static final short FULL_LEVEL = 12
```

Current power source level: 100%.

CONSTANT_POWER

```
public static final short CONSTANT_POWER = 1
```

Power source: Constant (mains) power.

RECHARGEABLE_BATTERY

```
public static final short RECHARGEABLE_BATTERY = 2
```

Power source: Rechargeable battery.

DISPOSABLE_BATTERY

```
public static final short DISPOSABLE_BATTERY = 4
```

Power source: Disposable battery.

Method Detail

getCurrentPowerMode

```
short getCurrentPowerMode()
```

Returns:
the current power mode.

getCurrentPowerSource

```
short getCurrentPowerSource()
```

Returns:
the current power source.

getCurrentPowerSourceLevel

```
short getCurrentPowerSourceLevel()
```

Returns:
the current power source level.

isConstantMainsPowerAvailable

```
boolean isConstantMainsPowerAvailable()
```

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

isDisposableBatteryAvailable

```
boolean isDisposableBatteryAvailable()
```

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

isRechargeableBatteryAvailable

```
boolean isRechargeableBatteryAvailable()
```

Returns:

true if rechargeable battery is available or false otherwise.

Interface ZigBeeSimpleDescriptor

org.osgi.service.zigbee.descriptors

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.

Version:
1.0

Method Summary		Page
int	getApplicationDeviceId()	142
byte	getApplicationDeviceVersion()	142
int	getApplicationProfileId()	142
short	getEndpoint()	142
int[]	getInputClusters()	143
int[]	getOutputClusters()	143
boolean	providesInputCluster(int clusterId)	143
boolean	providesOutputCluster(int clusterId)	143

Method Detail

getApplicationProfileId

int getApplicationProfileId()

Returns:
the application profile id.

getApplicationDeviceId

int getApplicationDeviceId()

Returns:
device id as defined per profile.

getEndpoint

short getEndpoint()

Returns:
the endpoint for which this descriptor is defined.

getApplicationDeviceVersion

byte getApplicationDeviceVersion()

Returns:
the version of the application.

getInputClusters

```
int[] getInputClusters()
```

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

getOutputClusters

```
int[] getOutputClusters()
```

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

providesInputCluster

```
boolean providesInputCluster(int clusterId)
```

Parameters:
clusterId - the cluster identifier

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

providesOutputCluster

```
boolean providesOutputCluster(int clusterId)
```

Parameters:
clusterId - the cluster identifier

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

Interface ZigBeeUserDescriptor

org.osgi.service.zigbee.descriptors

```
public interface ZigBeeUserDescriptor
```

This interface represents a User Descriptor as described in the ZigBee Specification. The User Descriptor contains information that allows the user to identify the device using user-friendly character string. The use of the User Descriptor is optional.

Version:
1.0

Method Summary		Page
String	getUserDescriptor()	144
void	setUserDescriptor (String userDescriptor, ZigBeeHandler handler) As described in "Table 2.137 ZDP Enumerations Description" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a set user desc request may throw: NOT_SUPPORTED, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR.	144

Method Detail

getUserDescriptor

```
String getUserDescriptor()
```

Returns:
a user-friendly that identify the device, such as 'Bedroom TV' or 'Stairs light'

setUserDescriptor

```
void setUserDescriptor(String userDescriptor,  
                       ZigBeeHandler handler)
```

As described in "Table 2.137 ZDP Enumerations Description" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a set user desc request may throw: NOT_SUPPORTED, DEVICE_NOT_FOUND, INV_REQUESTTYPE or NO_DESCRIPTOR.

Parameters:
userDescriptor - the user descriptor
handler - the response handler

Package org.osgi.service.zigbee.types

Class Summary		Page
ZCLAttributeID	This interface represents a ZCLAttributeID as described in the ZigBee Specification.	148
ZigBeeArray	This interface represents a ZigBeeArray as described in the ZigBee Specification.	151
ZigBeeBacnetOID	This interface represents a ZigBeeBacnetOID as described in the ZigBee Specification.	154
ZigBeeBag	This interface represents a ZigBeeBag as described in the ZigBee Specification.	157
ZigBeeBitmap16	This interface represents a ZigBeeBitmap16 as described in the ZigBee Specification.	160
ZigBeeBitmap24	This interface represents a ZigBeeBitmap24 as described in the ZigBee Specification.	163
ZigBeeBitmap32	This interface represents a ZigBeeBitmap32 as described in the ZigBee Specification.	166
ZigBeeBitmap40	This interface represents a ZigBeeBitmap40 as described in the ZigBee Specification.	169
ZigBeeBitmap48	This interface represents a ZigBeeBitmap48 as described in the ZigBee Specification.	172
ZigBeeBitmap56	This interface represents a ZigBeeBitmap56 as described in the ZigBee Specification.	175
ZigBeeBitmap64	This interface represents a ZigBeeBitmap64 as described in the ZigBee Specification.	178
ZigBeeBitmap8	This interface represents a ZigBeeBitmap8 as described in the ZigBee Specification.	181
ZigBeeBoolean	This interface represents a ZigBeeBoolean as described in the ZigBee Specification.	184
ZigBeeCharacterString	This interface represents a ZigBeeCharacterString as described in the ZigBee Specification.	187
ZigBeeClusterID	This interface represents a ZigBeeClusterID as described in the ZigBee Specification.	190
ZigBeeDate	This interface represents a ZigBeeDate as described in the ZigBee Specification.	193
ZigBeeEnumeration16	This interface represents a ZigBeeEnumeration16 as described in the ZigBee Specification.	196
ZigBeeEnumeration8	This interface represents a ZigBeeEnumeration8 as described in the ZigBee Specification.	199
ZigBeeFloatingDouble	This interface represents a ZigBeeFloatingDouble as described in the ZigBee Specification.	202
ZigBeeFloatingSemi	This interface represents a ZigBeeFloatingSemi as described in the ZigBee Specification.	205
ZigBeeFloatingSingle	This interface represents a ZigBeeFloatingSingle as described in the ZigBee Specification.	208
ZigBeeGeneralData16	This interface represents a ZigBeeGeneralData16 as described in the ZigBee Specification.	211
ZigBeeGeneralData24	This interface represents a ZigBeeGeneralData24 as described in the ZigBee Specification.	214
ZigBeeGeneralData32	This interface represents a ZigBeeGeneralData32 as described in the ZigBee Specification.	217
ZigBeeGeneralData40	This interface represents a ZigBeeGeneralData40 as described in the ZigBee Specification.	220

ZigBeeGeneralData48	This interface represents a ZigBeeGeneralData48 as described in the ZigBee Specification.	223
ZigBeeGeneralData56	This interface represents a ZigBeeGeneralData56 as described in the ZigBee Specification.	226
ZigBeeGeneralData64	This interface represents a ZigBeeGeneralData64 as described in the ZigBee Specification.	229
ZigBeeGeneralData8	This interface represents a ZigBeeGeneralData8 as described in the ZigBee Specification.	232
ZigBeeIEEEADDRESS	This interface represents a ZigBeeIEEEADDRESS as described in the ZigBee Specification.	235
ZigBeeLongCharacterString	This interface represents a ZigBeeLongCharacterString as described in the ZigBee Specification.	238
ZigBeeLongOctetString	This interface represents a ZigBeeLongOctetString as described in the ZigBee Specification.	241
ZigBeeOctetString	This interface represents a ZigBeeOctetString as described in the ZigBee Specification.	244
ZigBeeSet	This interface represents a ZigBeeSet as described in the ZigBee Specification.	247
ZigBeeSignedInteger16	This interface represents a ZigBeeSignedInteger16 as described in the ZigBee Specification.	250
ZigBeeSignedInteger24	This interface represents a ZigBeeSignedInteger24 as described in the ZigBee Specification.	253
ZigBeeSignedInteger32	This interface represents a ZigBeeSignedInteger32 as described in the ZigBee Specification.	256
ZigBeeSignedInteger40	This interface represents a ZigBeeSignedInteger40 as described in the ZigBee Specification.	259
ZigBeeSignedInteger48	This interface represents a ZigBeeSignedInteger48 as described in the ZigBee Specification.	262
ZigBeeSignedInteger56	This interface represents a ZigBeeSignedInteger56 as described in the ZigBee Specification.	265
ZigBeeSignedInteger64	This interface represents a ZigBeeSignedInteger64 as described in the ZigBee Specification.	268
ZigBeeSignedInteger8	This interface represents a ZigBeeSignedInteger8 as described in the ZigBee Specification.	271
ZigBeeStructure	This interface represents a ZigBeeStructure as described in the ZigBee Specification.	274
ZigBeeTimeOfDay	This interface represents a ZigBeeTimeOfDay as described in the ZigBee Specification.	277
ZigBeeUnsignedInteger16	This interface represents a ZigBeeUnsignedInteger16 as described in the ZigBee Specification.	280
ZigBeeUnsignedInteger24	This interface represents a ZigBeeUnsignedInteger24 as described in the ZigBee Specification.	283
ZigBeeUnsignedInteger32	This interface represents a ZigBeeUnsignedInteger32 as described in the ZigBee Specification.	286
ZigBeeUnsignedInteger40	This interface represents a ZigBeeUnsignedInteger40 as described in the ZigBee Specification.	289
ZigBeeUnsignedInteger48	This interface represents a ZigBeeUnsignedInteger48 as described in the ZigBee Specification.	292
ZigBeeUnsignedInteger56	This interface represents a ZigBeeUnsignedInteger56 as described in the ZigBee Specification.	295

ZigBeeUnsignedInteger64	This interface represents a ZigBeeUnsignedInteger64 as described in the ZigBee Specification.	298
ZigBeeUnsignedInteger8	This interface represents a ZigBeeUnsignedInteger8 as described in the ZigBee Specification.	301
ZigBeeUTCTime	This interface represents a ZigBeeUTCTime as described in the ZigBee Specification.	304

Class ZCLAttributeID

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZCLAttributeID
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZCLAttributeID
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZCLAttributeID as described in the ZigBee Specification.

Version:
1.0

Field Summary		Page
static short	ID	148

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	150
short	getId ()	149
static ZCLAttributeID	getInstance ()	148
Object	getInvalidNumber ()	149
Class	getJavaDataType ()	149
String	getName ()	149
boolean	isAnalog ()	149
void	serialize (Object param, ByteArrayOutputStream outdata)	149

Field Detail

ID

```
public static final short ID = 9
```

Method Detail

getInstance

```
public static ZCLAttributeID getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:

[isAnalog](#) in interface [ZCLDataTypeDescription](#)

Returns:

true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - `ByteArrayOutputStream` in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

data - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeArray

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeArray
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#)

```
public class ZigBeeArray
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeArray as described in the ZigBee Specification.

Version:

1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	152
short	getId ()	152
static ZigBeeArray	getInstance ()	151
Object	getInvalidNumber ()	152
Class	getJavaDataType ()	152
String	getName ()	152
boolean	isAnalog ()	151
void	serialize (Object param, ByteArrayOutputStream outdata)	152

Method Detail

getInstance

```
public static ZigBeeArray getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)

Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```


Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeBacnetOID

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeBacnetOID
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#)

```
public class ZigBeeBacnetOID
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeBacnetOID as described in the ZigBee Specification.

Version:

1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	155
short	getId ()	155
static ZigBeeBacnetOID	getInstance ()	154
Object	getInvalidNumber ()	155
Class	getJavaDataType ()	155
String	getName ()	155
boolean	isAnalog ()	154
void	serialize (Object param, ByteArrayOutputStream outdata)	155

Method Detail

getInstance

```
public static ZigBeeBacnetOID getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)

Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeBag

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeBag
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#)

```
public class ZigBeeBag
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeBag as described in the ZigBee Specification.

Version:

1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	158
short	getId ()	158
static ZigBeeBag	getInstance ()	157
Object	getInvalidNumber ()	158
Class	getJavaDataType ()	158
String	getName ()	158
boolean	isAnalog ()	157
void	serialize (Object param, ByteArrayOutputStream outdata)	158

Method Detail

getInstance

```
public static ZigBeeBag getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeBitmap16

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeBitmap16
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeBitmap16
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeBitmap16 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	161
short	getId ()	161
static ZigBeeBitmap16	getInstance ()	160
Object	getInvalidNumber ()	161
Class	getJavaDataType ()	161
String	getName ()	161
boolean	isAnalog ()	160
void	serialize (Object param, ByteArrayOutputStream outdata)	161

Method Detail

getInstance

```
public static ZigBeeBitmap16 getInstance ()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog ()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeBitmap24

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeBitmap24
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#)

```
public class ZigBeeBitmap24
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeBitmap24 as described in the ZigBee Specification.

Version:

1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	164
short	getId ()	164
static ZigBeeBitmap24	getInstance ()	163
Object	getInvalidNumber ()	164
Class	getJavaDataType ()	164
String	getName ()	164
boolean	isAnalog ()	163
void	serialize (Object param, ByteArrayOutputStream outdata)	164

Method Detail

getInstance

```
public static ZigBeeBitmap24 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)

Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeBitmap32

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeBitmap32
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeBitmap32
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeBitmap32 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	167
short	getId ()	167
static ZigBeeBitmap32	getInstance ()	166
Object	getInvalidNumber ()	167
Class	getJavaDataType ()	167
String	getName ()	167
boolean	isAnalog ()	166
void	serialize (Object param, ByteArrayOutputStream outdata)	167

Method Detail

getInstance

```
public static ZigBeeBitmap32 getInstance ()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog ()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeBitmap40

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeBitmap40
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeBitmap40
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeBitmap40 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	170
short	getId ()	170
static ZigBeeBitmap40	getInstance ()	169
Object	getInvalidNumber ()	170
Class	getJavaDataType ()	170
String	getName ()	170
boolean	isAnalog ()	169
void	serialize (Object param, ByteArrayOutputStream outdata)	170

Method Detail

getInstance

```
public static ZigBeeBitmap40 getInstance ()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog ()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeBitmap48

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeBitmap48
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeBitmap48
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeBitmap48 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	173
short	getId ()	173
static ZigBeeBitmap48	getInstance ()	172
Object	getInvalidNumber ()	173
Class	getJavaDataType ()	173
String	getName ()	173
boolean	isAnalog ()	172
void	serialize (Object param, ByteArrayOutputStream outdata)	173

Method Detail

getInstance

```
public static ZigBeeBitmap48 getInstance ()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog ()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeBitmap56

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeBitmap56
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeBitmap56
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeBitmap56 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	176
short	getId ()	176
static ZigBeeBitmap56	getInstance ()	175
Object	getInvalidNumber ()	176
Class	getJavaDataType ()	176
String	getName ()	176
boolean	isAnalog ()	175
void	serialize (Object param, ByteArrayOutputStream outdata)	176

Method Detail

getInstance

```
public static ZigBeeBitmap56 getInstance ()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog ()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```


Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeBitmap64

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeBitmap64
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeBitmap64
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeBitmap64 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	179
short	getId ()	179
static ZigBeeBitmap64	getInstance ()	178
Object	getInvalidNumber ()	179
Class	getJavaDataType ()	179
String	getName ()	179
boolean	isAnalog ()	178
void	serialize (Object param, ByteArrayOutputStream outdata)	179

Method Detail

getInstance

```
public static ZigBeeBitmap64 getInstance ()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog ()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeBitmap8

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeBitmap8
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#)

```
public class ZigBeeBitmap8
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeBitmap8 as described in the ZigBee Specification.

Version:

1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	182
short	getId ()	182
static ZigBeeBitmap8 ap8	getInstance ()	181
Object	getInvalidNumber ()	182
Class	getJavaDataType ()	182
String	getName ()	182
boolean	isAnalog ()	181
void	serialize (Object param, ByteArrayOutputStream outdata)	182

Method Detail

getInstance

```
public static ZigBeeBitmap8 getInstance ()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog ()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)

Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeBoolean

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeBoolean
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#)

```
public class ZigBeeBoolean
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeBoolean as described in the ZigBee Specification.

Version:

1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	185
short	getId ()	185
static ZigBeeBoolean	getInstance ()	184
Object	getInvalidNumber ()	185
Class	getJavaDataType ()	185
String	getName ()	185
boolean	isAnalog ()	184
void	serialize (Object param, ByteArrayOutputStream outdata)	185

Method Detail

getInstance

```
public static ZigBeeBoolean getInstance ()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog ()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)

Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeCharacterString

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeCharacterString
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeCharacterString
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeCharacterString as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	188
short	getId ()	188
static ZigBeeCharacterString	getInstance ()	187
Object	getInvalidNumber ()	188
Class	getJavaDataType ()	188
String	getName ()	188
boolean	isAnalog ()	187
void	serialize (Object param, ByteArrayOutputStream outdata)	188

Method Detail

getInstance

```
public static ZigBeeCharacterString getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeClusterID

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeClusterID
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeClusterID
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeClusterID as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	191
short	getId ()	191
static ZigBeeClusterID	getInstance ()	190
Object	getInvalidNumber ()	191
Class	getJavaDataType ()	191
String	getName ()	191
boolean	isAnalog ()	190
void	serialize (Object param, ByteArrayOutputStream outdata)	191

Method Detail

getInstance

```
public static ZigBeeClusterID getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeDate

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeDate
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#)

```
public class ZigBeeDate
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeDate as described in the ZigBee Specification.

Version:

1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	194
short	getId ()	194
static ZigBeeDate	getInstance ()	193
Object	getInvalidNumber ()	194
Class	getJavaDataType ()	194
String	getName ()	194
boolean	isAnalog ()	193
void	serialize (Object param, ByteArrayOutputStream outdata)	194

Method Detail

getInstance

```
public static ZigBeeDate getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeEnumeration16

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeEnumeration16
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#)

```
public class ZigBeeEnumeration16
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeEnumeration16 as described in the ZigBee Specification.

Version:

1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	197
short	getId ()	197
static ZigBeeEnumeration16	getInstance ()	196
Object	getInvalidNumber ()	197
Class	getJavaDataType ()	197
String	getName ()	197
boolean	isAnalog ()	196
void	serialize (Object param, ByteArrayOutputStream outdata)	197

Method Detail

getInstance

```
public static ZigBeeEnumeration16 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)

Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeEnumeration8

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeEnumeration8
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeEnumeration8
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeEnumeration8 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	200
short	getId ()	200
static ZigBeeEnumeration8	getInstance ()	199
Object	getInvalidNumber ()	200
Class	getJavaDataType ()	200
String	getName ()	200
boolean	isAnalog ()	199
void	serialize (Object param, ByteArrayOutputStream outdata)	200

Method Detail

getInstance

```
public static ZigBeeEnumeration8 getInstance ()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog ()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```


Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeFloatingDouble

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeFloatingDouble
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeFloatingDouble
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeFloatingDouble as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	203
short	getId ()	203
static ZigBeeFloatingDouble	getInstance ()	202
Object	getInvalidNumber ()	203
Class	getJavaDataType ()	203
String	getName ()	203
boolean	isAnalog ()	202
void	serialize (Object param, ByteArrayOutputStream outdata)	203

Method Detail

getInstance

```
public static ZigBeeFloatingDouble getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeFloatingSemi

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeFloatingSemi
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeFloatingSemi
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeFloatingSemi as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	206
short	getId ()	206
static ZigBeeFloatingSemi	getInstance ()	205
Object	getInvalidNumber ()	206
Class	getJavaDataType ()	206
String	getName ()	206
boolean	isAnalog ()	205
void	serialize (Object param, ByteArrayOutputStream outdata)	206

Method Detail

getInstance

```
public static ZigBeeFloatingSemi getInstance ()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog ()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeFloatingSingle

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeFloatingSingle
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#)

```
public class ZigBeeFloatingSingle
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeFloatingSingle as described in the ZigBee Specification.

Version:

1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	209
short	getId ()	209
static ZigBeeFloatingSingle	getInstance ()	208
Object	getInvalidNumber ()	209
Class	getJavaDataType ()	209
String	getName ()	209
boolean	isAnalog ()	208
void	serialize (Object param, ByteArrayOutputStream outdata)	209

Method Detail

getInstance

```
public static ZigBeeFloatingSingle getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)

Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeGeneralData16

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeGeneralData16
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#)

```
public class ZigBeeGeneralData16
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeGeneralData16 as described in the ZigBee Specification.

Version:

1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	212
short	getId ()	212
static ZigBeeGeneralData16	getInstance ()	211
Object	getInvalidNumber ()	212
Class	getJavaDataType ()	212
String	getName ()	212
boolean	isAnalog ()	211
void	serialize (Object param, ByteArrayOutputStream outdata)	212

Method Detail

getInstance

```
public static ZigBeeGeneralData16 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)

Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeGeneralData24

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeGeneralData24
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeGeneralData24
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeGeneralData24 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	215
short	getId ()	215
static ZigBeeGeneralData24	getInstance ()	214
Object	getInvalidNumber ()	215
Class	getJavaDataType ()	215
String	getName ()	215
boolean	isAnalog ()	214
void	serialize (Object param, ByteArrayOutputStream outdata)	215

Method Detail

getInstance

```
public static ZigBeeGeneralData24 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeGeneralData32

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeGeneralData32
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeGeneralData32
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeGeneralData32 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	218
short	getId ()	218
static ZigBeeGeneralData32	getInstance ()	217
Object	getInvalidNumber ()	218
Class	getJavaDataType ()	218
String	getName ()	218
boolean	isAnalog ()	217
void	serialize (Object param, ByteArrayOutputStream outdata)	218

Method Detail

getInstance

```
public static ZigBeeGeneralData32 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeGeneralData40

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeGeneralData40
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeGeneralData40
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeGeneralData40 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	221
short	getId ()	221
static ZigBeeGeneralData40	getInstance ()	220
Object	getInvalidNumber ()	221
Class	getJavaDataType ()	221
String	getName ()	221
boolean	isAnalog ()	220
void	serialize (Object param, ByteArrayOutputStream outdata)	221

Method Detail

getInstance

```
public static ZigBeeGeneralData40 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeGeneralData48

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeGeneralData48
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeGeneralData48
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeGeneralData48 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	224
short	getId ()	224
static ZigBeeGeneralData48	getInstance ()	223
Object	getInvalidNumber ()	224
Class	getJavaDataType ()	224
String	getName ()	224
boolean	isAnalog ()	223
void	serialize (Object param, ByteArrayOutputStream outdata)	224

Method Detail

getInstance

```
public static ZigBeeGeneralData48 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```


Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeGeneralData56

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeGeneralData56
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeGeneralData56
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeGeneralData56 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	227
short	getId ()	227
static ZigBeeGeneralData56	getInstance ()	226
Object	getInvalidNumber ()	227
Class	getJavaDataType ()	227
String	getName ()	227
boolean	isAnalog ()	226
void	serialize (Object param, ByteArrayOutputStream outdata)	227

Method Detail

getInstance

```
public static ZigBeeGeneralData56 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeGeneralData64

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeGeneralData64
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeGeneralData64
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeGeneralData64 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	230
short	getId ()	230
static ZigBeeGeneralData64	getInstance ()	229
Object	getInvalidNumber ()	230
Class	getJavaDataType ()	230
String	getName ()	230
boolean	isAnalog ()	229
void	serialize (Object param, ByteArrayOutputStream outdata)	230

Method Detail

getInstance

```
public static ZigBeeGeneralData64 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeGeneralData8

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeGeneralData8
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeGeneralData8
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeGeneralData8 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	233
short	getId ()	233
static ZigBeeGeneralData8	getInstance ()	232
Object	getInvalidNumber ()	233
Class	getJavaDataType ()	233
String	getName ()	233
boolean	isAnalog ()	232
void	serialize (Object param, ByteArrayOutputStream outdata)	233

Method Detail

getInstance

```
public static ZigBeeGeneralData8 getInstance ()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog ()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeIEEEADDRESS

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeIEEEADDRESS
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeIEEEADDRESS
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeIEEEADDRESS as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	236
short	getId ()	236
static ZigBeeIEEEADDRESS	getInstance ()	235
Object	getInvalidNumber ()	236
Class	getJavaDataType ()	236
String	getName ()	236
boolean	isAnalog ()	235
void	serialize (Object param, ByteArrayOutputStream outdata)	236

Method Detail

getInstance

```
public static ZigBeeIEEEADDRESS getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeLongCharacterString

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeLongCharacterString
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#)

```
public class ZigBeeLongCharacterString
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeLongCharacterString as described in the ZigBee Specification.

Version:

1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	239
short	getId ()	239
static ZigBeeLongCharacterString	getInstance ()	238
Object	getInvalidNumber ()	239
Class	getJavaDataType ()	239
String	getName ()	239
boolean	isAnalog ()	238
void	serialize (Object param, ByteArrayOutputStream outdata)	239

Method Detail

getInstance

```
public static ZigBeeLongCharacterString getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)

Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeLongOctetString

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeLongOctetString
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#)

```
public class ZigBeeLongOctetString
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeLongOctetString as described in the ZigBee Specification.

Version:

1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	242
short	getId ()	242
static ZigBeeLongOctetString	getInstance ()	241
Object	getInvalidNumber ()	242
Class	getJavaDataType ()	242
String	getName ()	242
boolean	isAnalog ()	241
void	serialize (Object param, ByteArrayOutputStream outdata)	242

Method Detail

getInstance

```
public static ZigBeeLongOctetString getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)

Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeOctetString

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeOctetString
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeOctetString
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeOctetString as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	245
short	getId ()	245
static ZigBeeOctetString	getInstance ()	244
Object	getInvalidNumber ()	245
Class	getJavaDataType ()	245
String	getName ()	245
boolean	isAnalog ()	244
void	serialize (Object param, ByteArrayOutputStream outdata)	245

Method Detail

getInstance

```
public static ZigBeeOctetString getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeSet

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSet
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#)

```
public class ZigBeeSet
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeSet as described in the ZigBee Specification.

Version:

1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	248
short	getId ()	248
static ZigBeeSet	getInstance ()	247
Object	getInvalidNumber ()	248
Class	getJavaDataType ()	248
String	getName ()	248
boolean	isAnalog ()	247
void	serialize (Object param, ByteArrayOutputStream outdata)	248

Method Detail

getInstance

```
public static ZigBeeSet getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```


Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeSignedInteger16

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSignedInteger16
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeSignedInteger16
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeSignedInteger16 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	251
short	getId ()	251
static ZigBeeSignedInteger16	getInstance ()	250
Object	getInvalidNumber ()	251
Class	getJavaDataType ()	251
String	getName ()	251
boolean	isAnalog ()	250
void	serialize (Object param, ByteArrayOutputStream outdata)	251

Method Detail

getInstance

```
public static ZigBeeSignedInteger16 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeSignedInteger24

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSignedInteger24
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeSignedInteger24
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeSignedInteger24 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	254
short	getId ()	254
static ZigBeeSignedInteger24	getInstance ()	253
Object	getInvalidNumber ()	254
Class	getJavaDataType ()	254
String	getName ()	254
boolean	isAnalog ()	253
void	serialize (Object param, ByteArrayOutputStream outdata)	254

Method Detail

getInstance

```
public static ZigBeeSignedInteger24 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeSignedInteger32

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSignedInteger32
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeSignedInteger32
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeSignedInteger32 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	257
short	getId ()	257
static ZigBeeSignedInteger32	getInstance ()	256
Object	getInvalidNumber ()	257
Class	getJavaDataType ()	257
String	getName ()	257
boolean	isAnalog ()	256
void	serialize (Object param, ByteArrayOutputStream outdata)	257

Method Detail

getInstance

```
public static ZigBeeSignedInteger32 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeSignedInteger40

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSignedInteger40
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeSignedInteger40
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeSignedInteger40 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	260
short	getId ()	260
static ZigBeeSignedInteger40	getInstance ()	259
Object	getInvalidNumber ()	260
Class	getJavaDataType ()	260
String	getName ()	260
boolean	isAnalog ()	259
void	serialize (Object param, ByteArrayOutputStream outdata)	260

Method Detail

getInstance

```
public static ZigBeeSignedInteger40 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeSignedInteger48

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSignedInteger48
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeSignedInteger48
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeSignedInteger48 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	263
short	getId ()	263
static ZigBeeSignedInteger48	getInstance ()	262
Object	getInvalidNumber ()	263
Class	getJavaDataType ()	263
String	getName ()	263
boolean	isAnalog ()	262
void	serialize (Object param, ByteArrayOutputStream outdata)	263

Method Detail

getInstance

```
public static ZigBeeSignedInteger48 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeSignedInteger56

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSignedInteger56
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeSignedInteger56
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeSignedInteger56 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	266
short	getId ()	266
static ZigBeeSignedInteger56	getInstance ()	265
Object	getInvalidNumber ()	266
Class	getJavaDataType ()	266
String	getName ()	266
boolean	isAnalog ()	265
void	serialize (Object param, ByteArrayOutputStream outdata)	266

Method Detail

getInstance

```
public static ZigBeeSignedInteger56 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeSignedInteger64

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSignedInteger64
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeSignedInteger64
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeSignedInteger64 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	269
short	getId ()	269
static ZigBeeSignedInteger64	getInstance ()	268
Object	getInvalidNumber ()	269
Class	getJavaDataType ()	269
String	getName ()	269
boolean	isAnalog ()	268
void	serialize (Object param, ByteArrayOutputStream outdata)	269

Method Detail

getInstance

```
public static ZigBeeSignedInteger64 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeSignedInteger8

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSignedInteger8
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeSignedInteger8
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeSignedInteger8 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	272
short	getId ()	272
static ZigBeeSignedInteger8	getInstance ()	271
Object	getInvalidNumber ()	272
Class	getJavaDataType ()	272
String	getName ()	272
boolean	isAnalog ()	271
void	serialize (Object param, ByteArrayOutputStream outdata)	272

Method Detail

getInstance

```
public static ZigBeeSignedInteger8 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```


Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeStructure

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeStructure
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeStructure
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeStructure as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	275
short	getId ()	275
static ZigBeeStructure	getInstance ()	274
Object	getInvalidNumber ()	275
Class	getJavaDataType ()	275
String	getName ()	275
boolean	isAnalog ()	274
void	serialize (Object param, ByteArrayOutputStream outdata)	275

Method Detail

getInstance

```
public static ZigBeeStructure getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeTimeOfDay

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeTimeOfDay
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeTimeOfDay
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeTimeOfDay as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	278
short	getId ()	278
static ZigBeeTimeOfDay	getInstance ()	277
Object	getInvalidNumber ()	278
Class	getJavaDataType ()	278
String	getName ()	278
boolean	isAnalog ()	277
void	serialize (Object param, ByteArrayOutputStream outdata)	278

Method Detail

getInstance

```
public static ZigBeeTimeOfDay getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeUnsignedInteger16

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeUnsignedInteger16
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#)

```
public class ZigBeeUnsignedInteger16
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeUnsignedInteger16 as described in the ZigBee Specification.

Version:

1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	281
short	getId ()	281
static ZigBeeUnsignedInteger16	getInstance ()	280
Object	getInvalidNumber ()	281
Class	getJavaDataType ()	281
String	getName ()	281
boolean	isAnalog ()	280
void	serialize (Object param, ByteArrayOutputStream outdata)	281

Method Detail

getInstance

```
public static ZigBeeUnsignedInteger16 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)

Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeUnsignedInteger24

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeUnsignedInteger24
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeUnsignedInteger24
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeUnsignedInteger24 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	284
short	getId ()	284
static ZigBeeUnsignedInteger24	getInstance ()	283
Object	getInvalidNumber ()	284
Class	getJavaDataType ()	284
String	getName ()	284
boolean	isAnalog ()	283
void	serialize (Object param, ByteArrayOutputStream outdata)	284

Method Detail

getInstance

```
public static ZigBeeUnsignedInteger24 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeUnsignedInteger32

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeUnsignedInteger32
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeUnsignedInteger32
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeUnsignedInteger32 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	287
short	getId ()	287
static ZigBeeUnsignedInteger32	getInstance ()	286
Object	getInvalidNumber ()	287
Class	getJavaDataType ()	287
String	getName ()	287
boolean	isAnalog ()	286
void	serialize (Object param, ByteArrayOutputStream outdata)	287

Method Detail

getInstance

```
public static ZigBeeUnsignedInteger32 getInstance ()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog ()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeUnsignedInteger40

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeUnsignedInteger40
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeUnsignedInteger40
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeUnsignedInteger40 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	290
short	getId ()	290
static ZigBeeUnsignedInteger40	getInstance ()	289
Object	getInvalidNumber ()	290
Class	getJavaDataType ()	290
String	getName ()	290
boolean	isAnalog ()	289
void	serialize (Object param, ByteArrayOutputStream outdata)	290

Method Detail

getInstance

```
public static ZigBeeUnsignedInteger40 getInstance ()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog ()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeUnsignedInteger48

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeUnsignedInteger48
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeUnsignedInteger48
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeUnsignedInteger48 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	293
short	getId ()	293
static ZigBeeUnsignedInteger48	getInstance ()	292
Object	getInvalidNumber ()	293
Class	getJavaDataType ()	293
String	getName ()	293
boolean	isAnalog ()	292
void	serialize (Object param, ByteArrayOutputStream outdata)	293

Method Detail

getInstance

```
public static ZigBeeUnsignedInteger48 getInstance ()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog ()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeUnsignedInteger56

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeUnsignedInteger56
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeUnsignedInteger56
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeUnsignedInteger56 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	296
short	getId ()	296
static ZigBeeUnsignedInteger56	getInstance ()	295
Object	getInvalidNumber ()	296
Class	getJavaDataType ()	296
String	getName ()	296
boolean	isAnalog ()	295
void	serialize (Object param, ByteArrayOutputStream outdata)	296

Method Detail

getInstance

```
public static ZigBeeUnsignedInteger56 getInstance ()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog ()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```


Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeUnsignedInteger64

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeUnsignedInteger64
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeUnsignedInteger64
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeUnsignedInteger64 as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	299
short	getId ()	299
static ZigBeeUnsignedInteger64	getInstance ()	298
Object	getInvalidNumber ()	299
Class	getJavaDataType ()	299
String	getName ()	299
boolean	isAnalog ()	298
void	serialize (Object param, ByteArrayOutputStream outdata)	299

Method Detail

getInstance

```
public static ZigBeeUnsignedInteger64 getInstance ()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog ()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeUnsignedInteger8

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeUnsignedInteger8
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#)

```
public class ZigBeeUnsignedInteger8
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeUnsignedInteger8 as described in the ZigBee Specification.

Version:

1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	302
short	getId ()	302
static ZigBeeUnsignedInteger8	getInstance ()	301
Object	getInvalidNumber ()	302
Class	getJavaDataType ()	302
String	getName ()	302
boolean	isAnalog ()	301
void	serialize (Object param, ByteArrayOutputStream outdata)	302

Method Detail

getInstance

```
public static ZigBeeUnsignedInteger8 getInstance()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)

Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Class ZigBeeUTCTime

[org.osgi.service.zigbee.types](#)

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeUTCTime
```

All Implemented Interfaces:
[ZCLDataTypeDescription](#)

```
public class ZigBeeUTCTime
extends Object
implements ZCLDataTypeDescription
```

This interface represents a ZigBeeUTCTime as described in the ZigBee Specification.

Version:
1.0

Method Summary		Page
Object	deserialize (ByteArrayInputStream data)	305
short	getId ()	305
static ZigBeeUTCTime	getInstance ()	304
Object	getInvalidNumber ()	305
Class	getJavaDataType ()	305
String	getName ()	305
boolean	isAnalog ()	304
void	serialize (Object param, ByteArrayOutputStream outdata)	305

Method Detail

getInstance

```
public static ZigBeeUTCTime getInstance ()
```

Returns:
the singleton instance.

isAnalog

```
public boolean isAnalog()
```

Specified by:
[isAnalog](#) in interface [ZCLDataTypeDescription](#)
Returns:
true, if the data type is analog.

getName

```
public String getName()
```

Specified by:

[getName](#) in interface [ZCLDataTypeDescription](#)

Returns:

The associated data type name string.

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

[getJavaDataType](#) in interface [ZCLDataTypeDescription](#)

Returns:

The corresponding Java type class.

getInvalidNumber

```
public Object getInvalidNumber()
```

Specified by:

[getInvalidNumber](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type invalid number if exists, otherwise returns null.

getId

```
public short getId()
```

Specified by:

[getId](#) in interface [ZCLDataTypeDescription](#)

Returns:

The data type identifier.

serialize

```
public void serialize(Object param,  
                      ByteArrayOutputStream outdata)
```

Specified by:

[serialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

param - Object to be serialized using the associated type.

outdata - ByteArrayOutputStream in which the array of bytes that represents the serialized value of param will be added.

deserialize

```
public Object deserialize(ByteArrayInputStream data)
```

Specified by:

[deserialize](#) in interface [ZCLDataTypeDescription](#)

Parameters:

`data` - `ByteArrayInputStream` Array of bytes to be deserialized using associated type.

Returns:

An object that represents the deserialized value of data.

Java API documentation generated with [DocFlex/Doclet](#) v1.5.6

[DocFlex/Doclet](#) is both a multi-format Javadoc doclet and a free edition of [DocFlex/Javadoc](#). If you need to customize your Javadoc without writing a full-blown doclet from scratch, [DocFlex/Javadoc](#) may be the only tool able to help you! Find out more at www.docflex.com

7 Considered Alternatives

- A **ZigBeeAttribute** object can also implement the **ZigBeeLocalAttribute** interface if the device is implemented locally. That is, the device is not imported from the network. The **ZigBeeLocalAttribute** interface provides a [getCurrentValue\(\)](#) method that provides direct access to the actual value of the attribute.
- In Java, primitives types are not objects and the generic function `decode(byte[])` returns an `Object` type. That's why Java objects types instead of primitives are used to represents ZigBee types.
- Is it possible to change the logical node type, e.g., an end device becoming a coordinator with a `setLogicalType`? Those changes are not described in ZigBee specifications and sound to be complex. So there is not setter for the operational mode in this specification.

Which entity has to be registered in the service registry? The ZigBeeEndpoint object and/or the ZigBeeNode object?

First, a decision has been taken (to be re-thought?) during Basel meeting (September 2012) on the number of objects to be registered: In order to avoid a burst events from 2 entities that are hierarchically related, it is decided only to register one object or the other.

Before arguing between the registration of ZigBeeEndpoint objects or the registration of ZigBeeNode objects, let's describe the two main use cases:

- 1st use case is associated to a special application like a light switch client: The client will search for light switch servers (standardized ZigBee endpoints) in the service registry before interacting with them. The bundle associated to the application will search for light switches and only for this type of services in the registry.
- 2nd use case is associated to a ZigBee network administrator (e.g., the user) who wants to explore the network and all the ZigBee devices and embedded services. The application or HMI will dynamically represent to the administrator all the devices that are available on the ZigBee network. So the application looks for ZigBee nodes in the service registry before exploring the endpoints, clusters, commands and attributes that are hierarchically hosted by these nodes.

Arguments in favor of the registration of ZigBeeEndpoint objects:

- The Endpoint brings more metadata and the information on the real functions brought by ZigBee devices. They are the first entity whose instances are standardized in terms of device profiles (e.g., ZigBee Home Automation profile standardizes light switch endpoints whereas nodes are not standardized). So the registration of this entity makes applications benefit from full OSGi service filter features to search for the right ZigBee services (Endpoints). The first use case is then easier in this case. The second use case will be slightly less easy since the application will have to ask for the node id of any endpoint and filter the list of the available unique nodes.

- Declarative Services lazy mode will be possible and very efficient for the first use case. The application will declare a service dependency towards endpoints that are light switch servers. Declarative Services lazy mode will build the service component only when light switches are available and will save hardware resources (cpu, memory) in when light switches are not available on the ZigBee network (and the OSGi service registry).

Arguments in favor of the registration of ZigBeeNode objects:

- The ZigBeeNode is the root object of the object graph of a ZigBee device. The registration of the ZigBeeNode object is thus enough to represent ZigBee network dynamicity and would avoid the multiplicity of events coming from the registration of all ZigBeeEndpoint objects. The discovery phase of the second case will be immediate to implement. However, in the first use case, the application will have to ask any node whether it hosts a light switch server. Declarative Services lazy mode will not be usable in that case.

Why having startNetwork() and permitJoin(short duration)? (And not rely on bundle API)

Every ZigBee chip/network has to be started in an independent way while the Base Driver maintains the bindings with available ZigBeeEndpoints to be exported (and that could be exported on a chip that is already started and on a chip that is not started). Relying on bundle start and stop would not make this distinction. This is why startNetwork() and permitJoin() methods are needed in the ZigBeeHost class.

Configure reporting and the White Board Pattern

ConfigureReporting command is a general command. Like every general command, it is implemented through a specific object design pattern. (e.g., Read/Write attribute are implemented with Attribute.get/SetValue() method calls)

Here, the Configure Reporting command enables an application (a client) to subscribe to application-specific events notified by a ZigBee device. In Java, you have 3 patterns available to implement eventing: Observer, WhiteBoard Pattern, Publish Subscribe (from the less to the most loosely coupling pattern). In OSGi, the Observer is not an option. Event Admin is the recommended one when it is relevant. The use of Event Admin, because it totally uncouples Publishers and Subscribers, is not possible for ZigBee eventing. That is why the use of Event Admin is not specified. Actually, ZigBee devices adapt their notification to client needs in attributes, frequency and considered range values. For ZigBee devices need to detect client needs, the Whiteboard pattern is the relevant model. We then have applied the WhiteBoard pattern like it was applied first in UPnP Device Service specification.

In brief:

- Applications interested in attribute reporting (ZigBeeEvents) register ZCLEventListener objects into the registry. The Attribute IDs, the frequency, attribute relevant value ranges are configurable into service properties.
- The Base Driver (for imported Endpoints) and locally implemented Endpoints request relevant listeners (relevance through service filtering) and read subscription information into service properties. Then, whenever an event matches a subscription, they call notifyEvent() method on every relevant registered listener.

Thus, registering a ZCLEventListener triggers 'Configure Reporting' commands sent by the base driver on networked devices. See 'Event API' section in ZigBee RFC and the javadoc for the detailed API specification.

8 Security Considerations

Description of all known vulnerabilities this may either introduce or address as well as scenarios of how the weaknesses could be circumvented.

9 Document Support

References

- [1]. ZigBee Alliance, ZigBee 2007 specification, 2007.
- [2]. ZigBee Alliance, ZigBee Cluster Library, document 075123r04ZB, 2007.
- [3]. André Bottaro, Anne Gérodolle, Philippe Lalanda, "Pervasive Service Composition in the Home Network", 21st IEEE International Conference on Advanced Information Networking and Applications (AINA-07), Niagara Falls, Canada, May 2007.
- [4]. Pavlin Dobrev, David Famolari, Christian Kurzke, Brent A. Miller, "Device and Service Discovery in Home Networks with OSGi", 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]. Peter Kriens, BJ Hargrave for the OSGi Alliance, "Listeners considered harmful: The whiteboard pattern", Technical Whitepaper, August 2004.
- [7]. ZigBee Alliance, ZigBee Gateway, 2011.
- [8]. Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, RFC2119, March 1997.
- [9]. Michael Jackson, "Software Requirements & Specifications", ISBN 0-201-87712-0, 1995.

Author's Address

Name	Andre Bottaro
Company	Orange
Address	28 Chemin du Vieux Chêne, Meylan, France
Voice	+33 4 76 76 41 03
e-mail	andre.bottaro@orange.com

Name	Arnaud Rinquin
Company	Orange
Address	28 Chemin du Vieux Chêne, Meylan, France
Voice	+33 4 76 76 45 59
e-mail	arnaud.rinquin@orange.com

Name	Jean-Pierre Poutcheu
Company	Orange
Address	28 Chemin du Vieux Chêne, Meylan, France
Voice	+33 4 76 76 41 03
e-mail	jeanpierre.poutcheu@orange.com

Name	Fabrice Blache
Company	Orange
Address	28 Chemin du Vieux Chêne, Meylan, France
Voice	+33 4 76 76 41 03
e-mail	fabrice.blache@orange.com

Name	Christophe Demottie
Company	Orange
Address	28 Chemin du Vieux Chêne, Meylan, France
Voice	+33 4 76 76 41 03
e-mail	christophe.demottie@orange.com

Name	Antonin Chazalet
Company	Orange
Address	28 Chemin du Vieux chêne, 38240 Meylan, France
Voice	+33 4 76 76 41 03
e-mail	antonin.chazalet@orange.com

Name	Evgeni Grigorov
Company	ProSyst Software
Address	222, 50935 Cologne, Germany
Voice	+49 221 6604 501
e-mail	e.grigorov@prosyst.com

Name	Nicola Portinaro
Company	Telecom Italia
Address	Via G. Reiss Romoli, 274 – 10148 Turin, Italy
Voice	+39 011 228 5635
e-mail	nicola.portinaro@telecomitalia.it

Name	Stefano Lenzi
Company	Consiglio Nazionale delle Ricerche - Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Address	Via G. Moruzzi 1, 56124 Pisa, Italy
Voice	+39 050 621 2844
e-mail	stefano.lenzi@isti.cnr.it

Acronyms and Abbreviations

End of Document
