



Device Service Specification for ZigBee Technology

Draft

321 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 Handler.....	32
ZigBee Data Types.....	32
Working With a ZigBee Endpoint.....	34
Implementing a ZigBee Endpoint.....	34
Event API.....	35
ZCL Exception.....	36
ZDP Exception.....	37
ZCL Frame.....	37
ZigBee Group.....	37
ZigBee Networking.....	37
Security.....	38
6 Javadoc.....	39
7 Considered Alternatives.....	315
Which entity has to be registered in the service registry? The ZigBeeEndpoint object and/or the ZigBeeNode object?.....	315
Why having startNetwork() and permitJoin(short duration)? (And not rely on bundle API)...	316
Configure reporting and the White Board Pattern.....	316
8 Security Considerations.....	316
9 Document Support.....	317
References.....	317
Author's Address.....	317
Acronyms and Abbreviations.....	319
End of Document.....	319

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 .

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 , 2013	<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 , 2013	<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 , 2013	<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 , 2013	<p>Andre Bottaro, Orange Jean-Pierre Poutcheu, Orange</p> <p>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 , 2013	<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 , 2013	<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 , 2013	<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 , 2013	<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 , 2013	<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 , 2013	<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 , 2013	<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 , 2013	<p>Jean-Pierre Poutcheu, Orange André Bottaro, Orange</p> <p>ZigBeeHost.setOperationalMode(short) replaced by ZigBeeHost.setLogicalType(short)</p>
v29-reg	July, 17 th , 2013	<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 , 2013	<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 , 2013	<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 , 2013	<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 , 2013	<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 , 2013	<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 , 2013	<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 , 2013	<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 , 2013 October, 10 th , 2013	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 , 2013	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 , 2013	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 , 2013	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 , 2013	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 , 2013	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Add <code>ZigBeeCommand.invoke(byte[] handler, bytes, String exportedServicePID)</code> throws <code>ZigBeeException</code>. 2. Add <code>ZigBeeEventListener</code>, <code>ZigBeeNode</code> et <code>ZigBeeDeviceDescriptionSet</code>'s properties types.
V43-reg	December, 5 th , 2013	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Update <code>ZigBeeNode</code>, and <code>ZigBeeEndpoint</code>.
V44-reg	December, 10 th , 2013	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Update filter part. 2. Update <code>ZigBeeEventListener</code>, <code>ZigBeeNode</code> et <code>ZigBeeDeviceDescriptionSet</code>'s properties types.
V45-reg	December, 18 th , 2013	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Add <code>ZigBeeEndPoint.bind(...)</code>, and <code>unbind(...)</code> methods. 2. Add <code>ZigBeeEndPoint.getBoundEndPoints(...)</code> method.
V46-reg	December, 23 th , 2013	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Update <code>ZigBeeEndpoint</code>'s <code>getBoundEndPoints</code> method (see Java API).
V47-reg	January, 20 th , 2014	<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 , 2014	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Add Stefano Lenzi as an author. 2. Add/update <code>ZCLHeader</code>, <code>ZCLFrame</code>, <code>ZigBeeCluster</code>, <code>ZigBeeCommandDescription</code>, and <code>ZigBeeCommandHandler</code>. 3. Remove <code>ZigBeeCommand</code> Java interface.
V49-reg	February, 14 th , 2014	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 4. Integrate last call decisions: Add <code>ZigBeeGroup</code>, and update <code>ZigBeeCommandHandler</code> javadoc.

Revision	Date	Comments
V50-reg	March, 3 th , 2014	<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 , 2014	<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 , 2014	<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 , 2014	<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 , 2014	<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 , 2014	<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 , 2014	<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 , 2014	<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 , 2014	<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 , 2014	<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 , 2014	<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 , 2014	<p>André Bottaro, Orange Antonin Chazalet, Orange</p> <ol style="list-style-type: none"> 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 refining driver.
V62-reg	April, 29 th , 2014	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Fix bug [reg] [2673] https://www.osgi.org/members/bugzilla/show_bug.cgi?id=2673
V63-reg	May, 5 th , 2014	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Remove ZigBeeHandler's table (everything is now in the javadoc). 2. Remove ZigBee MapHandler.
V64-reg	May, 21 th , 2014	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. ZigBee Attribute Record. 2. Update ZigBee, and ZCL specifications. 3. Update javadoc of ZCLAttribute.get/setValue, and ZCLCluster.read/writeAttributes. 4. Rename ZigBeeHost.getNetworkKey to getPreconfiguredLinkKey.
V65-reg	August, 22 th , 2014	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. ZigBeeNode.IEEEADDRESS is now of type BigInteger. 2. Extended pan id is now of type BigInteger. 3. Endpoint id is now of type short. 4. Nota: javadoc is to be updated.
V66-reg	December, 10 th 2014	<p>Antonin Chazalet, Orange André Bottaro, Orange</p> <ol style="list-style-type: none"> 1. Update RFC according to the changes on ZCLAttribute, and ZCLCluster. 2. Update the Javadoc section/link, and therefore the javadoc itself.
v67-reg	May, 7 th 2016	<p>André Bottaro, Orange</p> <p>updated references (removed errors).</p>

Revision	Date	Comments
v68-reg	May, 18 th 2016	André Bottaro, Orange Described broadcast and groupcast communication in ZigBeeNode, ZigBeeGroup, ZigBeeHost sections.
v69-reg	May, 22 nd 2016	André Bottaro, Orange Reviewed the RFC for consistency and put the version on OSGi Alliance repository.

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 Figure 1.

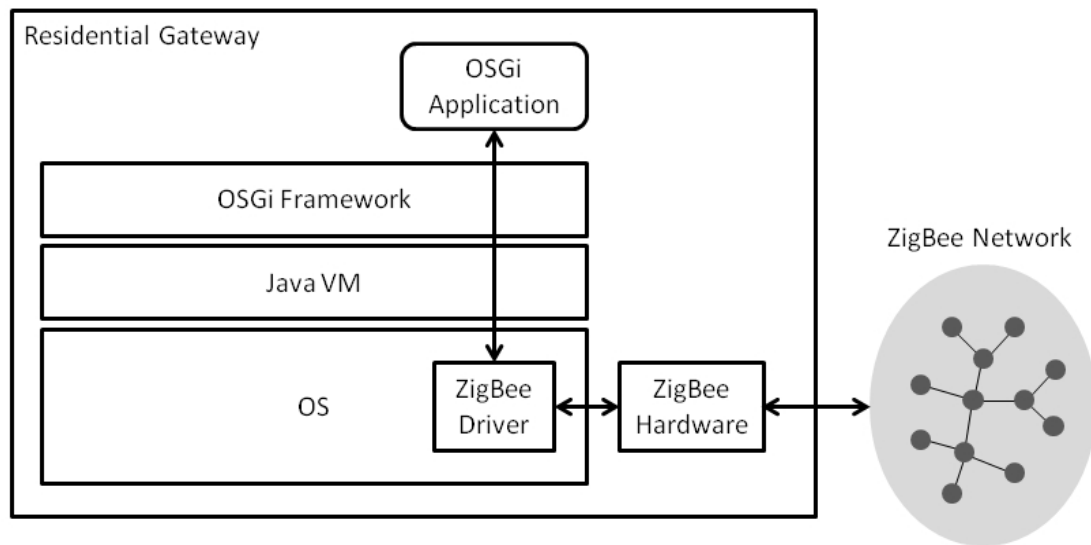


Figure 1 Communication with ZigBee devices through a ZigBee driver

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 Figure 2. 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.

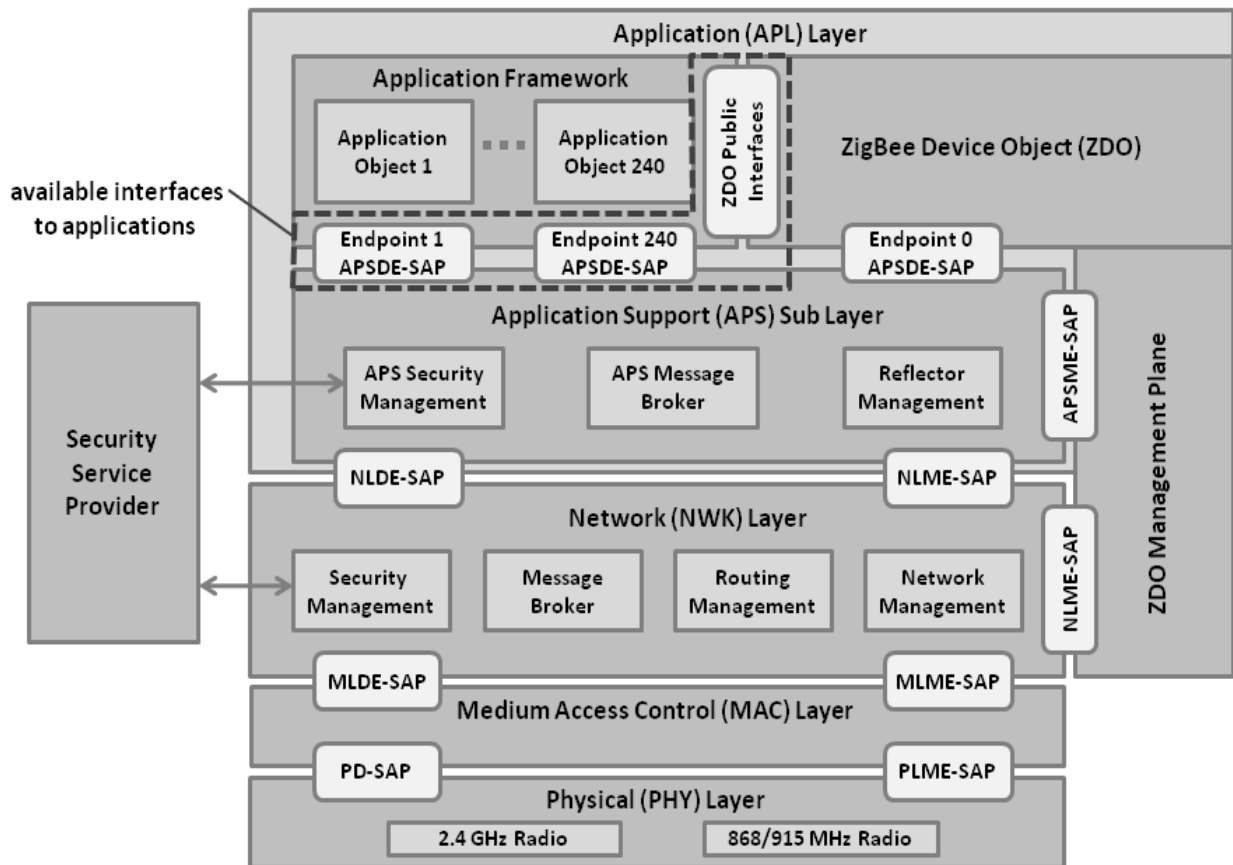


Figure 2: ZigBee Stack

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 2, 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 on the Framework) or external (implemented by another device on the network).
- *ZigBee Device Description* – Statically describes a ZigBee endpoint by providing its input/output clusters and specifies which of these clusters are mandatory or not. This entity is represented by a ZigBeeDeviceDescription object.
- *ZigBee Cluster* – Represents a ZigBee cluster entity, i.e., a set of attributes and commands. It allows the read and write of attribute values, and allows command invocation. This concept is represented by a ZCLCluster object.
- *ZigBee Cluster Description* – Cluster description provides details about available commands and attributes for a specific Cluster. A cluster description should be constant. A cluster description holds either a Client or a Server Cluster description and refers to a global cluster description.

- *ZigBee Global Cluster Description* – Global cluster description holds the server and client cluster description as well as common information such as cluster id, description and name. This concept is represented by a ZCLGlobalClusterDescription object.
- *ZigBee Command Description* – Statically describes a specific cluster command by giving its name, id, parameters. This entity is represented by a ZCLCommandDescription object.
- *ZigBee Parameter Description* – A ZigBee parameter description has a name, a range and a data type. This entity description is represented by a ZCLParameterDescription object.
- *ZigBee Attribute* – Holds the current value of an existing cluster attribute, it allows easy (de)encoding. This concept is represented by a ZCLAttribute object.
- *ZigBee Attribute Description* – Statically describes a ZigBee Attributes (data type, name, default value). It does not hold any current value. This concept is represented by a ZCLAttributeDescription object.
- *ZigBee Event Listener Service* – A service that listens to events coming from ZigBee devices.
- *ZigBee Event* – An event generated by a ZigBee node. It contains a modified attribute value of a specific cluster. This concept is represented by a ZigBeeEvent object.
- *ZigBee 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 error codes related to ZCL, APS, ZDP layers (see Table 2.137 ZDP Enumerations Description in [1].).
- *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.

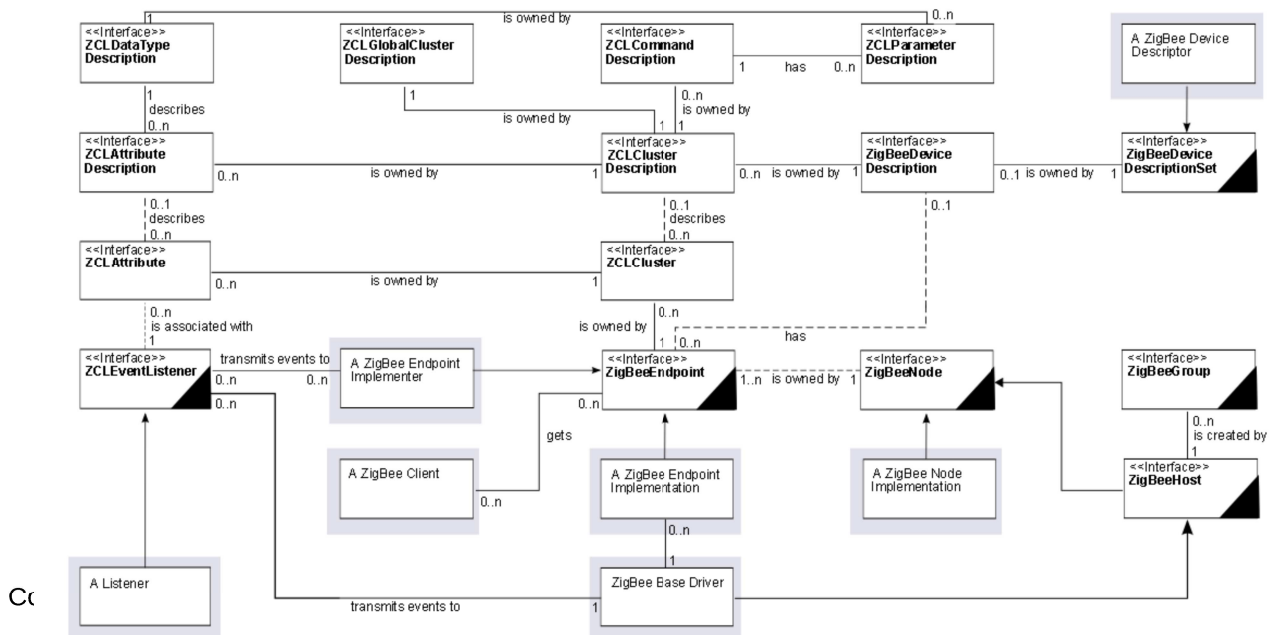


Figure 3 ZigBee Service Specification class Diagram org.osgi.service.zigbee package

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).

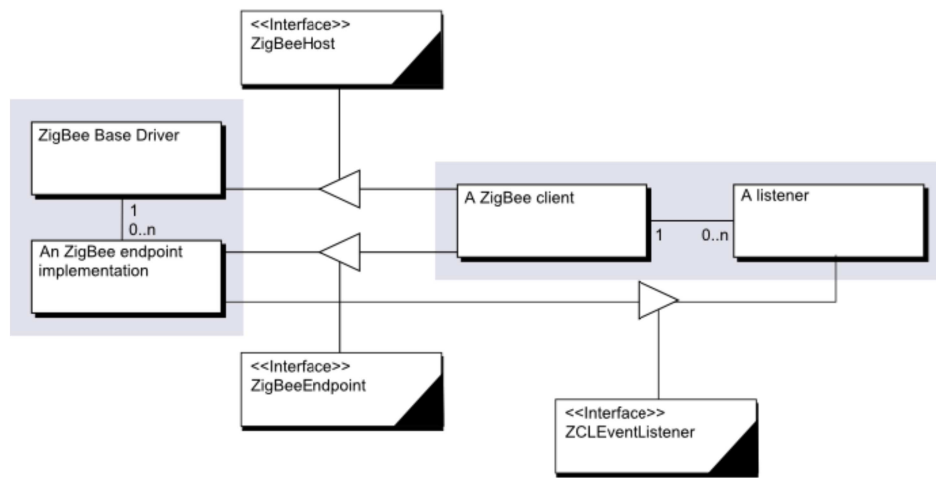


Figure 4: ZigBee device import

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 host (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.

Draft

June 1, 2016

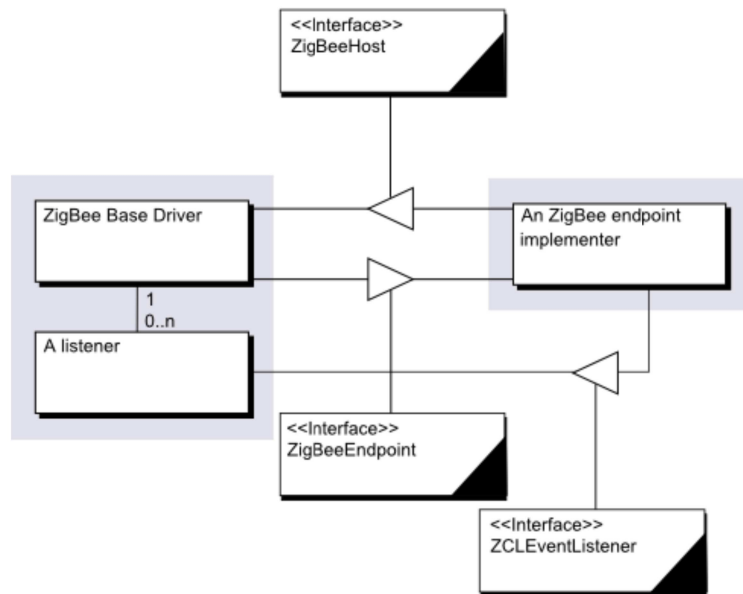


Figure 5: ZigBee device export

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

Several methods obey an asynchronous mechanism. For instance, ZigBee command invocation is made through the call to `ZCLCommand` invoke method that returns nothing. A handler object – here a `ZCLCommandHandler` – 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 `ZCLEventListener` interface with properties identifying a ZigBee attribute and a special event filter. This registration is conveyed as a ZigBee configure report command on the ZigBee network in the device import situation. Reports are received by the base driver and transmitted as `ZCLEventListener.notifyEvent()` method calls on relevant `ZCLEventListener` services in this situation. Local `ZigBeeEndpoint` objects directly call these methods to notify listeners with reports in the export situation. The Base Driver conveys events received through 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.

Draft

June 1, 2016

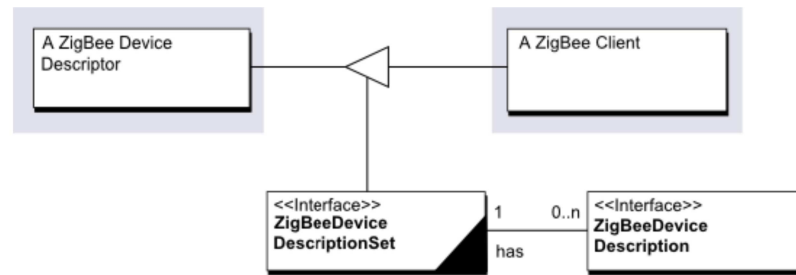


Figure 6: Using a set of device descriptions

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 an OSGi registered `ZigBeeNode` service. 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 *refining drivers* above a layer of *base drivers*. The refining driver is responsible for adapting technology-specific device services registered by the base driver into device services of another model (see `AbstractDevice` interface in Figure 7). In the case of a generic device abstraction layer, the model is agnostic to technologies.

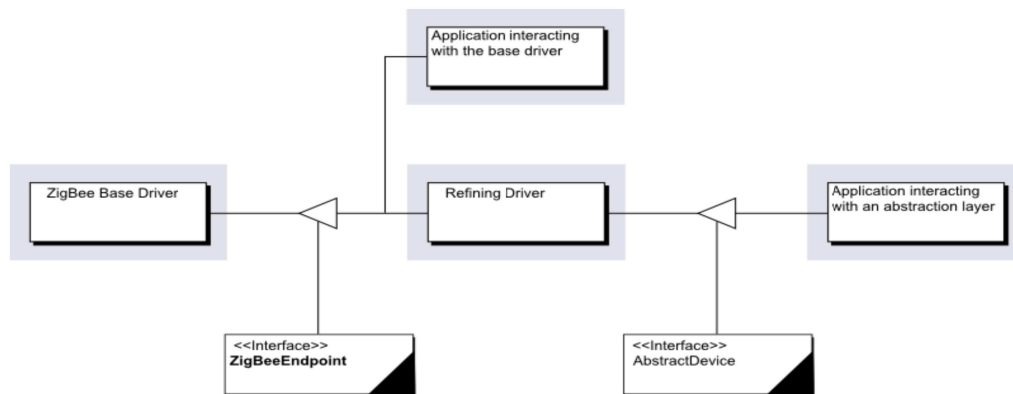


Figure 7: The ZigBee Base Driver and a refining driver representing devices in an abstract model

The ZigBee Alliance defines their own abstract model with ZigBee Profiles, e.g., Home Automation, Lighting, and refining drivers may provide the implementation of all ZigBee standard devices with ZigBee-specific Java interfaces. The `AbstractDevice` interface of Figure 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/BigInteger`)
- `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, which is set to 1 if the current power source is mains power, set to 0 otherwise.
- `RECEIVER_ON_WHEN_IDLE` – (`zigbee.node.receiver.on.when.idle/Integer`) represents the ZigBee receiver on when idle, i.e. 4th bit of "MAC Capabilities" in Node Descriptor, which is set to 1 if the device does not disable its receiver to conserve power during idle periods, set to 0 otherwise.
- `PAN_ID` – (`zigbee.node.pan.id/Integer`) (Personal Area Network Identifier) is a 16-bit value that identifies a ZigBee network. Every `ZigBeeNode` object is associated to a PAN ID, which can be retrieved through the `ZigBeeNode.getPanId()` method.
- `EXTENDED_PAN_ID` – (`zigbee.node.pan.extended.id/BigInteger`) Extended PAN ID is a 64-bit numbers that uniquely identify a PAN. It is intended to enhance selection of a PAN and enable recognition of network after PAN ID change (due to a previous conflict). `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 `ZigBeeHandler` object that is asynchronously notified with a `ZigBeeNodeDescriptor` object representing the Node Descriptor which contains information about the node capabilities. On failure, an exception is returned.
- `getPowerDescriptor()` – Returns a `ZigBeeHandler` object that is asynchronously notified with a `ZigBeePowerDescriptor` object representing the Node Power Descriptor which gives a dynamic

indication of the node power status. On failure, an exception is returned.

- `getComplexDescriptor()` – Returns a `ZigBeeHandler` object that is asynchronously notified with a `ZigBeeComplexDescriptor` object representing the Complex Descriptor which contains extended information for each device descriptions contained in this node. On failure, an exception is returned, esp. an exception with `NO_DESCRIPTOR` error code if no Complex Descriptor is provided.
- `getUserDescription()` – Returns a `ZigBeeHandler` object that is asynchronously notified with the unique field named “User description” of the User Descriptor, which contains information that allows the user to identify the device using user-friendly character string. On failure, an exception is returned, esp. an exception with `NO_DESCRIPTOR` error code if no User Descriptor is provided.

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

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

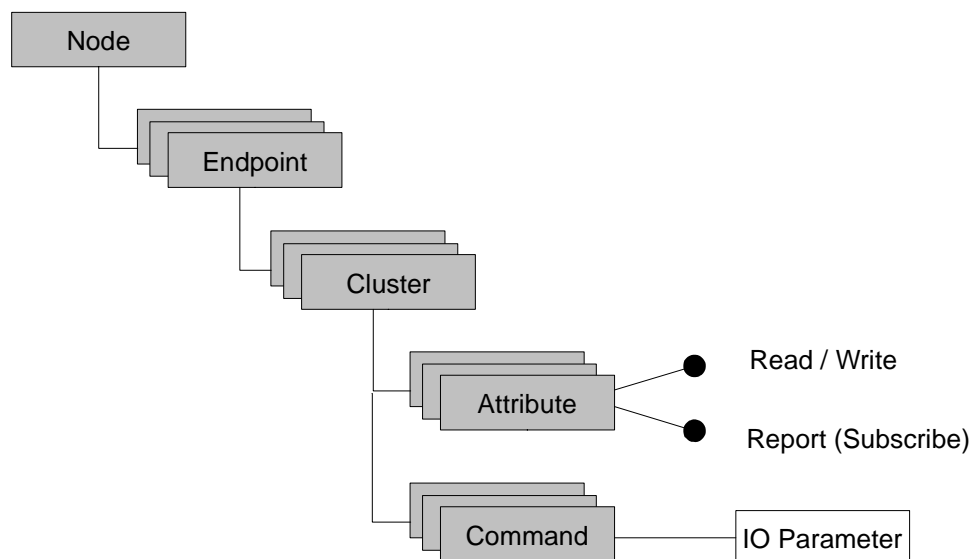


Figure 8: ZigBee Cluster Library model

All interfaces corresponding to the ZigBee Cluster Library model (see Figure 8) 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/short`) specifies the endpoint address within the node. Applications shall only use endpoints 1-240.
- **PROFILE_ID** – (`zigbee.device.profile.id/Integer`) identifies the profile that the endpoint belongs to. The profile can be either a ZigBee Alliance standard profile or a vendor-specific profile. The ZigBee specification defines several profile identifiers, and some others are vendor specific.
- **HOST_PID** (`zigbee.endpoint.host.pid/String`) – The ZigBee local host identifier is intended to uniquely identify the ZigBee local host, since there could be many hosts on the same platform. All the endpoints that belong to a specific network MUST specify the value of the associated host number.
- **DEVICE_ID** – (`zigbee.device.id/Integer`) identifies the device description supported by this endpoint. Like 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 are called Server cluster.
- **OUTPUT_CLUSTERS** – (`zigbee.endpoint.clusters.output/String[]`) specifies the list of output cluster ids supported by this endpoint. Output cluster 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.

Every endpoint must provide a simple descriptor. `getSimpleDescriptor()` returns a `ZigBeeHandler` object that is asynchronously notified with 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

ZigBeeEndpoint objects identified by their service PIDs.

ZigBee Device Description

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

ZigBee Device Description Set

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

- `VERSION` – (`zigbee.profile.version/Short`) The application profile version.
- `PROFILE_ID` – see `ZigBeeEndpoint.PROFILE_ID` property.
- `PROFILE_NAME` – (`zigbee.profile.name/String`) The profile name.
- `MANUFACTURER_CODE` – see `ZigBeeNode.MANUFACTURER_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 of the device. For example, ZigBee Home Automation profile provides a cluster dedicated to the control of lighting subsystems. Clusters are represented under `ZCLCluster` interface.

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

`ZCLCluster` provides some methods for reading and writing attributes values:

- `readAttributes(ZCLAttribute[] attributes, ZigBeeHandler handler)` – The ZigBee Base driver MAY generate the read attributes command on behalf of the OSGi application that is invoking `readAttributes()` method.
- `writeAttributes(boolean undivided, Map attributesAndValues, ZigBeeHandler handler)` – The ZigBee Base driver generates the write attributes command on behalf of the OSGi application that is invoking `writeAttributes()` method. If the 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 [2]). 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:

- `invoke(ZCLFrame frame, ZCLCommandHandler handler)` – a sequence of byte 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, ZCLCommandHandler handler, String exportedServicePID)` – a sequence of bytes represents the command frame, and `exportedServicePID` is the source endpoint of the command request. In targeted situations, the source endpoint is the valid service PID of an exported endpoint.

A handler is provided to manage the command response asynchronously.

ZCL Cluster Description

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

Every cluster client and server may have attributes (see [2], 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

`ZCLCommandDescription` describes a ZigBee command.

`ZCLCommandDescription` contains `ZCLParameterDescription` objects which describe the command parameters. `ZCLParameterDescription` has, for instance, `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.

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

`ZCLCommandDescription` 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. Every attribute is represented by an object implementing `ZCLAttribute` interface extending `ZCLAttributeInfo`. `ZCLAttribute` provides `getValue()` and `setValue()` to retrieve and set current attribute value with the use of a handler, which returns the response asynchronously.

ZigBee Attribute Description

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

ZCL Data Type Description

`ZCLAttributeInfo` and `ZCLParameterDescription` provide `getDataType()` method which returns a `ZCLDataTypeDescription` object. One object is associated to every ZigBee data type, see `ZigBeeDataTypes` constants in ZigBee Data Types section below.

ZCL Simple Type Description

`ZCLSimpleTypeDescription` extends `ZCLDataTypeDescription` interface to provide the following methods:

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

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

ZigBee Handlers

The ZigBee Handlers (i.e. `ZigBeeHandler`, `ZDPHandler` and `ZCLCommandHandler`) 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.

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	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	Long

UNSIGNED_INTEGER_48	Unsigned 48-bit integer	
UNSIGNED_INTEGER_56	Unsigned 56-bit integer	
UNSIGNED_INTEGER_64	Unsigned 64-bit integer	BigInteger
OCTET_STRING	Octet string	byte[]
LONG_OCTET_STRING	Long octet string	
SECURITY_KEY	128-bit Security Key	
CHARACTER_STRING	Character string	String
LONG_CHARACTER_STRING	Long character string	
BOOLEAN	Logical	Boolean
ENUMERATION_8	8-bit enumeration	Short
ENUMERATION_16	16-bit enumeration	Integer
CLUSTER_ID	Unsigned 16-bit integer	
ATTRIBUTE_ID	Unsigned 16-bit integer	
BACNET_OID	BACnet OID*(Unsigned 32-bit integer)	Long
IEEE_ADDRESS	IEEE address (MAC-48,EUI-48/64)	BigInteger
TIME_OF_DAY	Time of day	Date
DATE	Date	
UTC_TIME	UTCTime	
FLOATING_SEMI	Semi-precision	Float
FLOATING_SINGLE	Single precision	
FLOATING_DOUBLE	Double precision	Double
ARRAY	Array	List
STRUCTURE	Structure	
BAG	Bag	
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 within 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 `ZigBeeEndpoint` 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 `ZigBeeEndpoint` 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` service 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 array of `ZigBeeEndpoint` objects).

If an error occurs when exporting a ZigBee endpoint, then the base driver calls `ZigBeeEndpoint.notExported()` method with a relevant `ZigBeeException` 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 `ZigBeeException` object as the input argument with `ZigBeeException.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.getNodeAddress()` method can only return one ZigBee node address.

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

Event API

Eventing is available in import and export situations:

- External events from the network must be dispatched to listeners inside the OSGi Service Platform. The ZigBee Base driver is responsible for mapping the network events to internal listener events.
- Implementations of ZigBee endpoints must send out events to local listeners. The ZigBee Base driver dispatches events 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 `ZCLEventListener` 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 `ZCLEventListener` service 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 `ZCLEventListener` service. The mandatory properties (i.e. each `ZCLEventListener` **MUST** be registered with all the mandatory property) are:

- `ZigBeeNode.IEEE_ADDRESS` – (`zigbee.node.ieee.address/BigInteger`) 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.
- `ZCLAttribute.ID` – (`zigbee.attribute.id/Integer`) Only events generated by endpoints matching a specific attribute are delivered.
- `ZCLEventListener.ATTRIBUTE_DATA_TYPE` – (`zigbee.attribute.datatype/Short`) The Attribute data type field contains the data type of the attribute that is to be reported (see [2]. – 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 (see [2]. – 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 (see [2]. 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 `notifyEvent(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 request) 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` – Operation was not successful.
- `MALFORMED_COMMAND` – Wrong or missing field command.
- `CLUSTER_COMMAND_NOT_SUPPORTED` – Cluster command not supported.
- `GENERAL_COMMAND_NOT_SUPPORTED` – General command not supported.
- `MANUF_GENERAL_COMMAND_NOT_SUPPORTED` – Manufacturer general command not supported.
- `MANUF_CLUSTER_COMMAND_NOT_SUPPORTED` – Manufacturer cluster command not supported.
- `INVALID_FIELD` – Invalid field.
- `UNSUPPORTED_ATTRIBUTE` – Attribute not supported.
- `INVALID_VALUE` – Invalid attribute value.
- `READ_ONLY` – Read only attribute.
- `INSUFFICIENT_SPACE` – Insufficient amount of free space.
- `DUPLICATE_EXISTS` – Entry already exists in the table.
- `NOT_FOUND` – Requested information can not be found.
- `UNREPORTABLE_TYPE` – Attribute periodic reports cannot be issued.
- `INVALID_DATA_TYPE` – Incorrect attribute data type.
- `HARDWARE_FAILURE` – Operation unsuccessful due to a hardware failure.
- `SOFTWARE_FAILURE` – Operation unsuccessful due to a software failure.
- `CALIBRATION_ERROR` – An error occurred during calibration.

ZigBee Exception

Some error codes are specified by the OSGi Alliance:

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

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` – The supplied request type was invalid.
- `DEVICE_NOT_FOUND` – The requested device did not exist on a device following a child descriptor request to a parent.
- `INVALID_EP` – The supplied endpoint was equal to 0x00 or between 0xf1 and 0xff.
- `NOT_ACTIVE` – The requested endpoint is not described by a simple descriptor.
- `NOT_SUPPORTED` – The requested optional feature is not supported on the target device.
- `TIMEOUT` – A timeout has occurred with the requested operation.
- `NO_MATCH` – The end device bind request was unsuccessful due to a failure to match any suitable clusters.
- `NO_ENTRY` – The unbind request was unsuccessful due to the coordinator or source device not having an entry in its binding table to unbind.

- `NO_DESCRIPTOR` – A child descriptor was not available following a discovery request to a parent.
- `INSUFFICIENT_SPACE` – The device does not have storage space to support the requested operation.
- `NOT_PERMITTED` – The device is not in the proper state to support the requested operation.
- `TABLE_FULL` – The device does not have table space to support the operation.
- `NOT_AUTHORIZED` – The permissions configuration table on the target indicates that the request is not authorized from this device.

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. Therefore only getters are specified.

ZigBee Group

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

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

A `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`

A `ZigBeeGroup` service enables the ZigBee groupcasting of command invocation thanks to `ZigBeeGroup.invoke()` method. A groupcasted message is received by the endpoints that are members of the targeted group.

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.

Every discovered `ZigBeeNode` on the network has a logical node type returned by `ZigBeeNode.getNodeDescriptor().getLogicalType()`.

Network selection

The base driver provides a `ZigBeeHost` object for every available ZigBee local host. A ZigBee local host can represent a ZigBee chip on a USB dongle, a ZigBee built-in chip or a ZigBee Gateway Device (see [7].) This object must be registered under `ZigBeeHost` interface. The latter 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()`).

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

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

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

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.

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

they perform the Key Establishment to generate Link Keys.

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

6 Javadoc

OSGi Javadoc

01/06/16 23:23

Package Summary		Page
<u>org.osgi.service.zigbee</u>	This is the main package of this specification.	43
<u>org.osgi.service.zigbee.descriptions</u>	This package contains the interfaces for descriptions.	125
<u>org.osgi.service.zigbee.descriptors</u>	This package contains the interfaces representing the ZigBee descriptors and the fields defined inside some of them.	144
<u>org.osgi.service.zigbee.types</u>	Utility classes modeling the ZCL data types.	160

Package org.osgi.service.zigbee

This is the main package of this specification.

See:

[Description](#)

Interface Summary		Page
ZCLAttribute	This interface represents a ZCLAttribute and adds to the ZCLAttributeInfo interface the methods to read and write the ZCL attribute from and to the ZigBee node with respectively the getValue(ZigBeeHandler) and setValue(Object, ZigBeeHandler) methods	50
ZCLAttributeInfo	This interface represents ZCLAttribute information.	52
ZCLCluster	This interface represents a ZCL Cluster	54
ZCLCommandHandler	Manage response of a command request to the Base Driver	59
ZCLEventListener	This interface represents a listener to events from ZigBee Device nodes	60
ZCLFrame	This interface models the ZigBee Cluster Library Frame.	67
ZCLHeader	This interface represents the ZCL Frame Header.	69
ZDPFrame	This interface represents a ZDP frame (see Figure 2.19 Format of the ZDP Frame ZIGBEE SPECIFICATION: 1_053474r17ZB_TSC-ZigBee-Specification.pdf).	75
ZDPHandler	Callback for managing response to ZDPFrame sent by ZigBeeNode.invoke(int, int, ZDPFrame, ZDPHandler)	76
ZigBeeDataInput	The ZigBeeDataInput interface is designed for converting a series of bytes in java data types.	77
ZigBeeDataOutput	The ZigBeeDataOutput interface is designed for converting java data types into a series of bytes.	80
ZigBeeEndpoint	This interface represents a ZigBee EndPoint.	92
ZigBeeEvent	This interface represents events generated by a ZigBee Device node	97
ZigBeeGroup	This interface represents a ZigBee Group	102
ZigBeeHandler	ZigBeeHandler manages response of a request to the Base Driver	105
ZigBeeHost	This interface represents the machine that hosts the code to run a ZigBee device or client.	106
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)	113
ZigBeeNode	This interface represents a ZigBee node, means a physical device that can communicate using the ZigBee protocol.	115
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)	123

Class Summary	Page
---------------	------

ZigBeeDataTypes	This interface represents all ZigBee data types, and contains the common serialize/deserialize methods for the org.osgi.service.zigbee.types.* This constants are not the same provided by the ZigBee specification, and follows the rules below: bit 0-3: if bit 6 is one, these bits represents the size of the data type in bytes.	82
---------------------------------	--	----

Exception Summary		Page
APSEException	This exception class is specialized for the APS errors.	45
ZCLEException	This class represents root exception for all the code related to ZigBee/ZCL.	62
ZDPEException	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)	71
ZigBeeException	This class represents root exception for all the code related to ZigBee.	99

Package org.osgi.service.zigbee Description

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

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

`ZigBeeEndpoint` interface provides the `ZigBeeEndpoint.getServerCluster(serverClusterId)` method to get an interface reference to a `ZCLCluster` object.

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

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

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

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

Example import for consumers using the API in this package:

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

Example import for providers implementing the API in this package:

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

Class APSException

org.osgi.service.zigbee

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

All Implemented Interfaces:
Serializable

```
public class APSException
extends ZigBeeException
```

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

Field Summary		Page
static int	ASDU_TOO_LONG A transmit request failed since the ASDU is too large and fragmentation is not supported.	46
static int	DEFRAG_DEFERRED A received fragmented frame could not be defragmented at the current time.	46
static int	DEFRAG_UNSUPPORTED A received fragmented frame could not be defragmented since the device does not support fragmentation.	47
static int	ILLEGAL_REQUEST A parameter value was out of range.	47
static int	INVALID_BINDING An APSME-UNBIND.request failed due to the requested binding link not existing in the binding table.	47
static int	INVALID_GROUP An APSME-REMOVE-GROUP.request has been issued with a group identifier that does not appear in the group table.	47
static int	INVALID_PARAMETER A parameter value was invalid or out of range.	47
static int	NO_ACK An APSDE-DATA.request requesting acknowledged transmission failed due to no acknowledgement being received.	47
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.	47
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.	47
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.	47

static int	SECURED LINK KEY An ASDU was received that was secured using a link key.	47
static int	SECURED NWK KEY An ASDU was received that was secured using a network key.	48
static int	SECURITY FAIL An APSDE-DATA.request requesting security has resulted in an error during the corresponding security processing.	48
static int	SUCCESS A request has been executed successfully.	46
static int	TABLE FULL An APSME-BIND.request or APSME.ADDGROUP. request issued when the binding or group tables, respectively, were full.	48
static int	UNSECURED An ASDU was received without any security.	48
static int	UNSUPPORTED ATTRIBUTE An APSME-GET.request or APSMESET. request has been issued with an unknown attribute identifier.	48

Fields inherited from class org.osgi.service.zigbee. ZigBeeException
errorCode , OSGI_EXISTING_ID , OSGI_MULTIPLE_HOSTS , UNKNOWN_ERROR , zigBeeErrorCode

Constructor Summary	Page
APSEException (int errorCode, int zigBeeErrorCode, String errorDesc) Create a ZCLEException containing a specific errorCode or zigBeeErrorCode.	49
APSEException (int errorCode, String errorDesc) Create a ZCLEException containing a specific errorCode.	48
APSEException (String errorDesc) Create a ZCLEException containing only a description, but no error codes.	48

Methods inherited from class org.osgi.service.zigbee. ZigBeeException
getErrorCode , getZigBeeErrorCode , hasZigbeeErrorCode

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 = 65
```

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

DEFRAG_DEFERRED

```
public static final int DEFRAG_DEFERRED = 66
```

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

DEFRAG_UNSUPPORTED

```
public static final int DEFRAG_UNSUPPORTED = 67
```

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

ILLEGAL_REQUEST

```
public static final int ILLEGAL_REQUEST = 68
```

A parameter value was out of range.

INVALID_BINDING

```
public static final int INVALID_BINDING = 69
```

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 = 70
```

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 = 71
```

A parameter value was invalid or out of range.

NO_ACK

```
public static final int NO_ACK = 72
```

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

NO_BOUND_DEVICE

```
public static final int NO_BOUND_DEVICE = 73
```

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

NO_SHORT_ADDRESS

```
public static final int NO_SHORT_ADDRESS = 74
```

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

NOT_SUPPORTED

```
public static final int NOT_SUPPORTED = 75
```

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

SECURED_LINK_KEY

```
public static final int SECURED_LINK_KEY = 76
```

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

SECURED_NWK_KEY

```
public static final int SECURED_NWK_KEY = 77
```

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

SECURITY_FAIL

```
public static final int SECURITY_FAIL = 78
```

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

TABLE_FULL

```
public static final int TABLE_FULL = 79
```

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

UNSECURED

```
public static final int UNSECURED = 80
```

An ASDU was received without any security.

UNSUPPORTED_ATTRIBUTE

```
public static final int UNSUPPORTED_ATTRIBUTE = 81
```

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

Constructor Detail

APSEException

```
public APSEException(String errorDesc)
```

Create a [ZCLEException](#) containing only a description, but no error codes. If issued on this exception the [ZigBeeException.getErrorCode\(\)](#) and [ZigBeeException.getZigBeeErrorCode\(\)](#) methods return the [ZigBeeException.UNKNOWN_ERROR](#) constant.

Parameters:

errorDesc - exception error description

APSEException

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

Create a [ZCLEException](#) containing a specific errorCode. Using this constructor with errorCode set to [ZigBeeException.UNKNOWN_ERROR](#) is equivalent to call [APSEException\(String\)](#).

Parameters:

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

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

APSException

```
public APSException(int errorCode,
                    int zigBeeErrorCode,
                    String errorDesc)
```

Create a [ZCLException](#) containing a specific errorCode or zigBeeErrorCode. Using this constructor with both the errorCode and zigBeeErrorCode set to [ZigBeeException.UNKNOWN_ERROR](#) is equivalent to call [APSException\(String\)](#).

Parameters:

errorCode - One of the error codes defined in this interface or

[ZigBeeException.UNKNOWN_ERROR](#) the actual error is not covered in this interface. In this case the zigBeeErrorCode parameter must be the actual status code returned by the ZigBe stack.

zigBeeErrorCode - The actual APS status code or [ZigBeeException.UNKNOWN_ERROR](#) if this status is unknown.

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

Interface ZCLAttribute

org.osgi.service.zigbee

All Superinterfaces:

[ZCLAttributeInfo](#)

```
public interface ZCLAttribute
extends ZCLAttributeInfo
```

This interface represents a ZCLAttribute and adds to the ZCLAttributeInfo interface the methods to read and write the ZCL attribute from and to the ZigBee node with respectively the [getValue\(ZigBeeHandler\)](#) and [setValue\(Object, ZigBeeHandler\)](#) methods

Field Summary		Page
String	ID Property key for the optional attribute id of a ZigBee Event Listener.	50

Method Summary		Page
void	getValue (ZigBeeHandler handler) Gets the current value of the attribute.	50
void	setValue (Object value, ZigBeeHandler handler) Sets the current value of the attribute.	51

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

[getDataType](#), [getId](#), [getManufacturerCode](#), [isManufacturerSpecific](#)

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

getValue

```
void getValue(ZigBeeHandler handler)
```

Gets the current value of the attribute.

As described in "2.4.1.3 Effect on Receipt" chapter of the ZCL, a "read attribute" can have the following status: SUCCESS, or UNSUPPORTED_ATTRIBUTE (see [ZCLException](#)).

The response object given to the handler is the attribute's Java data type (see [ZCLAttributeInfo.getDataType\(\)](#) method) that will contain the current attribute value (or null if an UNSUPPORTED_ATTRIBUTE occurred or in case of an invalid value).

Parameters:

handler - the handler

setValue

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

Sets the current value of the attribute.

As described in "2.4.3.3 Effect on Receipt" chapter of the ZCL, a "write attribute" can have the following status: SUCCESS, UNSUPPORTED_ATTRIBUTE, INVALID_DATA_TYPE, READ_ONLY, INVALID_VALUE (see [ZCLException](#)), or NOT_AUTHORIZED (see [ZDPException](#)).

The response object given to the handler is a Boolean set to true if the attribute value has been written. A null value is processed as an invalid number. In case of an error has occurred, onFailure is called with a ZCLException.

Parameters:

- value - the Java value to set
- handler - the handler

Interface ZCLAttributeInfo

org.osgi.service.zigbee

All Known Subinterfaces:

[ZCLAttribute](#), [ZCLAttributeDescription](#)

```
public interface ZCLAttributeInfo
```

This interface represents ZCLAttribute information.

Field Summary		Page
String	ID Property key for the optional attribute id of a ZigBee Event Listener.	52

Method Summary		Page
ZCLDataTypeDescription	getDataType()	53
int	getId()	52
int	getManufacturerCode()	52
boolean	isManufacturerSpecific()	52

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

isManufacturerSpecific

```
boolean isManufacturerSpecific()
```

Returns:

true if and only if this attribute is related to a Manufacturer extension

getManufacturerCode

```
int getManufacturerCode()
```

Returns:

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

getId

```
int getId()
```

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

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).

Interface ZCLCluster

org.osgi.service.zigbee

```
public interface ZCLCluster
```

This interface represents a ZCL Cluster

Field Summary		Page
String	DOMAIN Property key for the optional cluster domain.	55
String	ID Property key for the optional cluster id.	54
String	NAME Property key for the optional cluster name.	55

Method Summary		Page
void	getAttribute (int attributeId, int code, ZigBeeHandler handler) Get the cluster attribute identified corresponding to given attributeld of a specific Manufacturer or the standard attribute	55
void	getAttribute (int attributeId, ZigBeeHandler handler) Get the cluster attribute identified corresponding to given attributeld.	55
void	getAttributes (int code, ZigBeeHandler handler) Get an array of all this Cluster's Attributes.	56
void	getAttributes (ZigBeeHandler handler) Get an array of all this Cluster's Attributes.	55
void	getCommandIds (ZigBeeHandler handler) Get an array of all the commandIds of the ZCLCluster.	57
int	getId ()	55
void	invoke (ZCLFrame frame, ZCLCommandHandler handler) Invokes the action.	57
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.	57
void	readAttributes (ZCLAttributeInfo [] attributes, ZigBeeHandler handler) Read a list of attributes.	56
void	writeAttributes (boolean undivided, Map attributesAndValues, ZigBeeHandler handler) Write a list of attributes.	56

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

See Also:

[To get Manufacturer specific attribute use ZCLCluster#getAttribute\(int, int, ZigBeeHandler\)](#)

getAttribute

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

Get the cluster attribute identified corresponding to given attributeId of a specific Manufacturer or the standard attribute

Parameters:

attributeId - an Attribute identifier
code - the int representing the Manufacturer code for getting the vendor specific attribute, use -1 if looking for standard attribute
handler - the response handler

getAttributes

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

Get an array of all this Cluster's Attributes. This method returns only standard attributes

Parameters:

handler - the response handler

See Also:

[To get Manufacturer specific attribute use ZCLCluster#getAttributes\(int, ZigBeeHandler\)](#)

getAttributes

```
void getAttributes(int code,  
                  ZigBeeHandler handler)
```

Get an array of all this Cluster's Attributes. This method returns only standard attributes when using -1 as code or vendor specific attribute when invoked with the proper code.

Parameters:

code - the int representing the Manufacturer code for getting the vendor specific attribute, use -1 if looking for standard attribute

handler - the response handler

readAttributes

```
void readAttributes(ZCLAttributeInfo[] attributes,  
                  ZigBeeHandler handler)
```

Read a list of attributes.

As described in "2.4.1.3 Effect on Receipt" chapter of the ZCL, a "read attribute" can have the following status: SUCCESS, or UNSUPPORTED_ATTRIBUTE (see [ZCLException](#)).

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 in the corresponding Java wrapper type (or null if an UNSUPPORTED_ATTRIBUTE occurred or in case of an invalid value).

NOTE Considering the ZigBee Specification all the attributes must be standard attributes or belong to the same Manufacturer otherwise `IllegalArgumentException` will be thrown

Parameters:

attributes - An array of [ZCLAttributeInfo](#)

handler - the response handler

Throws:

`NullPointerException` - the attribute array cannot be null

`IllegalArgumentException` - if some of [ZCLAttributeInfo](#) are manufacturer specific and other are standard, or even if there are mix of attributes with different manufacturer specific code, Or if the attributes array is empty

writeAttributes

```
void writeAttributes(boolean undivided,  
                    Map attributesAndValues,  
                    ZigBeeHandler handler)
```

Write a list of attributes.

As described in "2.4.3.3 Effect on Receipt" chapter of the ZCL, a "write attribute" can have the following status: SUCCESS, UNSUPPORTED_ATTRIBUTE, INVALID_DATA_TYPE, READ_ONLY, INVALID_VALUE (see [ZCLException](#)), or NOT_AUTHORIZED (see [ZDPException](#)).

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 (see above). Every null value in the Map is considered as an invalid number. 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.

NOTE Considering the ZigBee Specification all the attributes must be standard attributes or belong to the same Manufacturer otherwise `IllegalArgumentException` will be thrown

Parameters:

`undivided` - The write command is undivided or not
`attributesAndValues` - A Map of attributes, and values to be written.
`handler` - the response handler

Throws:

`IllegalArgumentException` - if some of [ZCLAttributeInfo](#) are manufacturer specific and other are standard, or even if there are mix of attributes with different manufacturer specific code

getCommandIds

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

Get an array of all the commandIds of the ZCLCluster.

This method is implemented for ZCL devices compliant version equal or later than 1.2 of the Home Automation Profile or other profiles that 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 `ZCLException.GENERAL_COMMAND_NOT_SUPPORTED`.

The response object given to the handler is an array containing the commandIds. Each commandId is of Integer type.

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

Method Summary		Page
void	notifyResponse (ZCLFrame frame, Exception e) Notifies the result (success or failure) of the call.	59

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 (As described in "Table 2.17 Enumerated Status Values Used in the ZCL" of the ZCL specification).

Interface ZCLEventListener

org.osgi.service.zigbee

```
public interface ZCLEventListener
```

This interface represents a listener to events from ZigBee Device nodes

Field Summary		Page
String	ATTRIBUTE_DATA_TYPE Property key for the optional attribute data type of an attribute reporting configuration record, cf.	60
String	MAX_REPORT_INTERVAL Property key for the optional maximum interval, in seconds between issuing reports of the attribute.	61
String	MIN_REPORT_INTERVAL Property key for the optional minimum interval, in seconds between issuing reports of the attribute.	60
String	REPORTABLE_CHANGE Property key for the optional maximum change to the attribute that will result in a report being issued.	61

Method Summary		Page
void	notifyEvent (ZigBeeEvent event) Callback method that is invoked for received events.	61
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.	61
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.	61

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.ZigBeeException
│   │   │   └── org.osgi.service.zigbee.ZCLException
```

All Implemented Interfaces:
Serializable

```
public class ZCLException
extends ZigBeeException
```

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

Field Summary		Page
static int	CALIBRATION_ERROR CALIBRATION_ERROR	65
static int	CLUSTER_COMMAND_NOT_SUPPORTED CLUSTER_COMMAND_NOT_SUPPORTED	63
static int	DUPLICATE_EXISTS DUPLICATE_EXISTS	64
static int	FAILURE FAILURE	63
static int	GENERAL_COMMAND_NOT_SUPPORTED GENERAL_COMMAND_NOT_SUPPORTED	64
static int	HARDWARE_FAILURE HARDWARE_FAILURE - in this case, an additional exception describing the problem can be nested.	65
static int	INSUFFICIENT_SPACE INSUFFICIENT_SPACE	64
static int	INVALID_DATA_TYPE INVALID_DATA_TYPE	65
static int	INVALID_FIELD INVALID_FIELD	64
static int	INVALID_VALUE INVALID_VALUE	64
static int	MALFORMED_COMMAND MALFORMED_COMMAND	63
static int	MANUF_CLUSTER_COMMAND_NOT_SUPPORTED MANUF_CLUSTER_COMMAND_NOT_SUPPORTED	64
static int	MANUF_GENERAL_COMMAND_NOT_SUPPORTED MANUF_GENERAL_COMMAND_NOT_SUPPORTED	64

static int	NOT_FOUND NOT_FOUND	64
static int	READ_ONLY READ_ONLY	64
static int	SOFTWARE_FAILURE SOFTWARE_FAILURE - in this case, an additional exception describing the problem can be nested.	65
static int	SUCCESS SUCCESS	63
static int	UNREPORTABLE_TYPE UNREPORTABLE_TYPE	64
static int	UNSUPPORTED_ATTRIBUTE UNSUPPORTED_ATTRIBUTE	64

Fields inherited from class org.osgi.service.zigbee.[ZigBeeException](#)

[errorCode](#), [OSGI_EXISTING_ID](#), [OSGI_MULTIPLE_HOSTS](#), [UNKNOWN_ERROR](#), [zigBeeErrorCode](#)

Constructor Summary

Pag
e

[ZCLException](#)(int errorCode, int zigBeeErrorCode, String errorDesc)

Create a [ZCLException](#) containing a specific errorCode OR zigBeeErrorCode.

65

[ZCLException](#)(int errorCode, String errorDesc)

Create a [ZCLException](#) containing a specific errorCode.

65

[ZCLException](#)(String errorDesc)

Create a [ZCLException](#) containing only a description, but no error codes.

65

Methods inherited from class org.osgi.service.zigbee.[ZigBeeException](#)

[getErrorCode](#), [getZigBeeErrorCode](#), [hasZigbeeErrorCode](#)

Field Detail

SUCCESS

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

SUCCESS

FAILURE

```
public static final int FAILURE = 1
```

FAILURE

MALFORMED_COMMAND

```
public static final int MALFORMED_COMMAND = 2
```

MALFORMED_COMMAND

CLUSTER_COMMAND_NOT_SUPPORTED

```
public static final int CLUSTER_COMMAND_NOT_SUPPORTED = 3
```

CLUSTER_COMMAND_NOT_SUPPORTED

~~**GENERAL_COMMAND_NOT_SUPPORTED**~~

```
public static final int GENERAL_COMMAND_NOT_SUPPORTED = 4
```

GENERAL_COMMAND_NOT_SUPPORTED

~~**MANUF_CLUSTER_COMMAND_NOT_SUPPORTED**~~

```
public static final int MANUF_CLUSTER_COMMAND_NOT_SUPPORTED = 5
```

MANUF_CLUSTER_COMMAND_NOT_SUPPORTED

~~**MANUF_GENERAL_COMMAND_NOT_SUPPORTED**~~

```
public static final int MANUF_GENERAL_COMMAND_NOT_SUPPORTED = 6
```

MANUF_GENERAL_COMMAND_NOT_SUPPORTED

~~**INVALID_FIELD**~~

```
public static final int INVALID_FIELD = 7
```

INVALID_FIELD

~~**UNSUPPORTED_ATTRIBUTE**~~

```
public static final int UNSUPPORTED_ATTRIBUTE = 8
```

UNSUPPORTED_ATTRIBUTE

~~**INVALID_VALUE**~~

```
public static final int INVALID_VALUE = 9
```

INVALID_VALUE

~~**READ_ONLY**~~

```
public static final int READ_ONLY = 10
```

READ_ONLY

~~**INSUFFICIENT_SPACE**~~

```
public static final int INSUFFICIENT_SPACE = 11
```

INSUFFICIENT_SPACE

~~**DUPLICATE_EXISTS**~~

```
public static final int DUPLICATE_EXISTS = 12
```

DUPLICATE_EXISTS

~~**NOT_FOUND**~~

```
public static final int NOT_FOUND = 13
```

NOT_FOUND

~~**UNREPORTABLE_TYPE**~~

```
public static final int UNREPORTABLE_TYPE = 14
```


UNREPORTABLE_TYPE

INVALID_DATA_TYPE

```
public static final int INVALID_DATA_TYPE = 15
```

INVALID_DATA_TYPE

HARDWARE_FAILURE

```
public static final int HARDWARE_FAILURE = 16
```

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

SOFTWARE_FAILURE

```
public static final int SOFTWARE_FAILURE = 17
```

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

CALIBRATION_ERROR

```
public static final int CALIBRATION_ERROR = 18
```

CALIBRATION_ERROR

Constructor Detail

ZCLErrorException

```
public ZCLErrorException(String errorDesc)
```

Create a [ZCLErrorException](#) containing only a description, but no error codes. If issued on this exception the [ZigBeeException.getErrorCode\(\)](#) and [ZigBeeException.getZigBeeErrorCode\(\)](#) methods return the [ZigBeeException.UNKNOWN_ERROR](#) constant.

Parameters:

errorDesc - exception error description

ZCLErrorException

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

Create a [ZCLErrorException](#) containing a specific errorCode. Using this constructor with errorCode set to [ZigBeeException.UNKNOWN_ERROR](#) is equivalent to call [ZCLErrorException\(String\)](#).

Parameters:

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

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

ZCLErrorException

```
public ZCLErrorException(int errorCode,  
                        int zigBeeErrorCode,  
                        String errorDesc)
```

Create a [ZCLException](#) containing a specific `errorCode` or `zigBeeErrorCode`. Using this constructor with both the `errorCode` and `zigBeeErrorCode` set to [ZigBeeException.UNKNOWN_ERROR](#) is equivalent to call [ZCLException\(String\)](#).

Parameters:

`errorCode` - One of the error codes defined in this interface or [ZigBeeException.UNKNOWN_ERROR](#) the actual error is not covered in this interface. In this case the `zigBeeErrorCode` parameter must be the actual status code returned by the ZigBe stack.

`zigBeeErrorCode` - The actual ZCL status code or [ZigBeeException.UNKNOWN_ERROR](#) if this status is unknown.

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

Interface ZCLFrame

org.osgi.service.zigbee

```
public interface ZCLFrame
```

This interface models the ZigBee Cluster Library Frame.

Method Summary		Page
<code>byte[]</code>	getBytes() Returns a byte array containing the raw ZCL frame, suitable to be sent on the wire.	67
ZigBeeDataInput	getDataInput() Returns <code>ZigBeeDataInput</code> for reading the ZCLFrame payload content.	67
ZCLHeader	getHeader() Returns the ZCLFrame header.	67

Method Detail

getHeader

[ZCLHeader](#) `getHeader()`

Returns the ZCLFrame header.

Returns:
the header

getBytes

`byte[]` `getBytes()`

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

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

getDataInput

[ZigBeeDataInput](#) `getDataInput()`

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

Returns:
a `DataInput` for the payload of the [ZCLFrame](#). This method does not generate a copy of the payload.

Throws:

`IllegalStateException` - if the `InputStream` is not available.

Interface ZCLHeader

org.osgi.service.zigbee

```
public interface ZCLHeader
```

This interface represents the ZCL Frame Header.

Method Summary		Pag e
int	getCommandId() Get this ZCLHeader's command id	69
short	getFrameControlField() Returns the Frame Control field of the ZCLHeader	70
int	getManufacturerCode() Get manufacturerCode of the ZCL Frame Header	69
byte	getSequenceNumber() The ZCL Frame Header Transaction Sequence Number	70
boolean	isClientServerDirection()	70
boolean	isClusterSpecificCommand() Returns the Frame Type Sub-field of the Frame Control Field	69
boolean	isDefaultResponseDisabled()	70
boolean	isManufacturerSpecific()	70

Method Detail

getCommandId

```
int getCommandId()
```

Get this ZCLHeader's command id

Returns:
the commandId

getManufacturerCode

```
int getManufacturerCode()
```

Get manufacturerCode of the ZCL Frame Header

Returns:
the manufacturerCode if the ZCL Frame is maufacturer specific, otherwise returns -1

isClusterSpecificCommand

```
boolean isClusterSpecificCommand()
```

Returns the Frame Type Sub-field of the Frame Control Field

Returns:

true if the Frame Control Field states that the command is Cluster Specific. Returns false otherwise

isManufacturerSpecific

```
boolean isManufacturerSpecific()
```

Returns:

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

isClientServerDirection

```
boolean isClientServerDirection()
```

Returns:

the isClientServerDirection value

isDefaultResponseDisabled

```
boolean isDefaultResponseDisabled()
```

Returns:

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

getSequenceNumber

```
byte getSequenceNumber()
```

The ZCL Frame Header Transaction Sequence Number

Returns:

the transaction sequence number

getFrameControlField

```
short getFrameControlField()
```

Returns the Frame Control field of the ZCLHeader

Returns:

the frame control field.

Class ZDPException

org.osgi.service.zigbee

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

All Implemented Interfaces:
Serializable

```
public class ZDPException
extends ZigBeeException
```

This class represents root exception for all the code related to ZDP (see Table 2.137 ZDP Enumerations Description in ZIGBEE SPECIFICATION: 1_053474r17ZB_TSC-ZigBee-Specification.pdf)

Field Summary		Pag e
static int	DEVICE_NOT_FOUND The requested device did not exist on a device following a child descriptor request to a parent.	72
static int	INSUFFICIENT_SPACE The device does not have storage space to support the requested operation.	73
static int	INV_REQUESTTYPE The supplied request type was invalid.	72
static int	INVALID_EP The supplied endpoint was equal to 0x00 or between 0xf1 and 0xff.	72
static int	NO_DESCRIPTOR A child descriptor was not available following a discovery request to a parent.	73
static int	NO_ENTRY The unbind request was unsuccessful due to the coordinator or source device not having an entry in its binding table to unbind.	73
static int	NO_MATCH The end device bind request was unsuccessful due to a failure to match any suitable clusters.	73
static int	NOT_ACTIVE The requested endpoint is not described by a simple descriptor.	72
static int	NOT_AUTHORIZED The permissions configuration table on the target indicates that the request is not authorized from this device.	73
static int	NOT_PERMITTED The device is not in the proper state to support the requested operation.	73
static int	NOT_SUPPORTED The requested optional feature is not supported on the target device.	72
static int	SUCCESS The requested operation or transmission was completed successfully.	72

static int	TABLE_FULL The device does not have table space to support the operation.	73
static int	TIMEOUT A timeout has occurred with the requested operation.	73

Fields inherited from class org.osgi.service.zigbee.[ZigBeeException](#)

[errorCode](#), [OSGI_EXISTING_ID](#), [OSGI_MULTIPLE_HOSTS](#), [UNKNOWN_ERROR](#), [zigBeeErrorCode](#)

Constructor Summary

	Page
ZDPEException (int errorCode, int zigBeeErrorCode, String errorDesc) Create a ZCLEException containing a specific errorCode OR zigBeeErrorCode.	74
ZDPEException (int errorCode, String errorDesc) Create a ZCLEException containing a specific errorCode.	74
ZDPEException (String errorDesc) Create a ZCLEException containing only a description, but no error codes.	73

Methods inherited from class org.osgi.service.zigbee.[ZigBeeException](#)

[getErrorCode](#), [getZigBeeErrorCode](#), [hasZigbeeErrorCode](#)

Field Detail

SUCCESS

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

The requested operation or transmission was completed successfully.

INV_REQUESTTYPE

```
public static final int INV_REQUESTTYPE = 33
```

The supplied request type was invalid.

DEVICE_NOT_FOUND

```
public static final int DEVICE_NOT_FOUND = 34
```

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

INVALID_EP

```
public static final int INVALID_EP = 35
```

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

NOT_ACTIVE

```
public static final int NOT_ACTIVE = 36
```

The requested endpoint is not described by a simple descriptor.

NOT_SUPPORTED

```
public static final int NOT_SUPPORTED = 37
```


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

TIMEOUT

```
public static final int TIMEOUT = 38
```

A timeout has occurred with the requested operation.

NO_MATCH

```
public static final int NO_MATCH = 39
```

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

NO_ENTRY

```
public static final int NO_ENTRY = 40
```

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

NO_DESCRIPTOR

```
public static final int NO_DESCRIPTOR = 41
```

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

INSUFFICIENT_SPACE

```
public static final int INSUFFICIENT_SPACE = 42
```

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

NOT_PERMITTED

```
public static final int NOT_PERMITTED = 43
```

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

TABLE_FULL

```
public static final int TABLE_FULL = 44
```

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

NOT_AUTHORIZED

```
public static final int NOT_AUTHORIZED = 45
```

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

Constructor Detail

ZDPEException

```
public ZDPEException(String errorDesc)
```

Create a [ZCLEException](#) containing only a description, but no error codes. If issued on this exception the [ZigBeeException.getErrorCode\(\)](#) and [ZigBeeException.getZigBeeErrorCode\(\)](#) methods return the [ZigBeeException.UNKNOWN_ERROR](#) constant.

Parameters:

`errorDesc` - exception error description

ZDPEException

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

Create a [ZCLEException](#) containing a specific `errorCode`. Using this constructor with `errorCode` set to [ZigBeeException.UNKNOWN_ERROR](#) is equivalent to call [ZDPEException\(String\)](#).

Parameters:

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

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

ZDPEException

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

Create a [ZCLEException](#) containing a specific `errorCode` or `zigBeeErrorCode`. Using this constructor with both the `errorCode` and `zigBeeErrorCode` set to [ZigBeeException.UNKNOWN_ERROR](#) is equivalent to call [ZDPEException\(String\)](#).

Parameters:

`errorCode` - One of the error codes defined in this interface or [ZigBeeException.UNKNOWN_ERROR](#) the actual error is not covered in this interface. In this case the `zigBeeErrorCode` parameter must be the actual status code returned by the ZigBe stack.

`zigBeeErrorCode` - The actual ZDP status code or [ZigBeeException.UNKNOWN_ERROR](#) if this status is unknown.

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

Interface ZDPFrame

org.osgi.service.zigbee

```
public interface ZDPFrame
```

This interface represents a ZDP frame (see Figure 2.19 Format of the ZDP Frame ZIGBEE SPECIFICATION: 1_053474r17ZB_TSC-ZigBee-Specification.pdf).

This interface **MUST** be implemented by the developer invoking the [ZigBeeNode.invoke\(int, int, ZDPFrame, ZDPHandler\)](#)

Notes

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

Method Summary		Page
ZigBeeDataInput	getDataInput()	75
byte[]	getPayload() Get (a copy of this ZDP) payload	75

Method Detail

getPayload

```
byte[] getPayload()
```

Get (a copy of this ZDP) payload

Returns:
a copy of the payload

getDataInput

```
ZigBeeDataInput getDataInput()  
throws IllegalStateException
```

Returns:
an [ZigBeeDataInput](#) for the payload of the [ZDPFrame](#). This method, in contrary to [getPayload\(\)](#), doesn't require to create a copy of the payload.

Throws:
`IllegalStateException` - if the `InputStream` is not available.

Interface ZDPHandler

org.osgi.service.zigbee

```
public interface ZDPHandler
```

Callback for managing response to [ZDPFrame](#) sent by [ZigBeeNode.invoke\(int, int, ZDPFrame, ZDPHandler\)](#)

Method Summary		Page
void	zdoResponse (int clusterId, ZDPFrame frame, Exception e) Notifies the result (success or failure) of the call.	76

Method Detail

zdoResponse

```
void zdoResponse(int clusterId,  
                 ZDPFrame frame,  
                 Exception e)
```

Notifies the result (success or failure) of the call. This method is invoked by the entity that registered the [ZigBeeNode](#), and it is expected that only the ZigBee Base Driver register it.

The [ZDPHandler](#) MUST be invoked with `null` value for the Exception parameter in case of success.

On the contrary, the [ZDPFrame](#) MUST be contain the message received from the [ZigBeeNode](#) even in case of failure so that the implementor can analyze the content of the message to better understand the failure.

Parameters:

`clusterId` - the clusterId of the response

`frame` - the [ZDPFrame](#) containing the response, in case of failure the value MAY be `null`, if it is not the [ZDPFrame](#)

`e` - is any exception related to ZigBee communication failure, in case of success the value is `null`

Interface ZigBeeDataInput

org.osgi.service.zigbee

```
public interface ZigBeeDataInput
```

The `ZigBeeDataInput` interface is designed for converting a series of bytes in java data types. The purpose of this interface is the same as the `DataInput` interface that is in the java library, with the difference that in this interface, byte ordering is little endian, whereas in the `DataInput` interface is big endian. Each method provided by this interface read one or more bytes from the underlying stream, combine them, and return a java data type. The pointer to the stream is then moved immediately after the last byte read. If this pointer past the available buffer bounds, a subsequent call to one of these methods will throw a `EOFException`.

Method Summary		Page
byte	readByte() Read a byte from the <code>DataInput</code> Stream.	77
byte[]	readBytes(int len) Read the specified amount of bytes from the underlying stream and return a copy of them.	78
double	readDouble()	78
float	readFloat(int size)	78
int	readInt(int size) Read an an integer of the specified <code>size</code> .	77
long	readLong(int size) Read a certain amount of bytes and returns a long.	78

Method Detail

readByte

```
byte readByte()  
    throws IOException
```

Read a byte from the `DataInput` Stream.

Returns:
the byte read from the data input.

Throws:
`IOException` - If an I/O error occurs.
`EOFException` - When the end of the input has been reached and there are no more data to read.

readInt

```
int readInt(int size)  
    throws IOException
```

Read an an integer of the specified `size`.

Parameters:
`size` - the number of bytes that have to be read. Allowed values for this parameter are in the range (1, 4]. If `b1` is the first read byte and `b4` is the last (supposing that `size` is 4) then:

```
int = (b1 & 0xff) | ((b2 & 0xff) << 8) | ((b3 & 0xff) << 16) | ((b4 & 0xff) << 24)
```

Returns:

the integer read.

Throws:

`IOException` - If an I/O error occurs.

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

readLong

```
long readLong(int size)
    throws IOException
```

Read a certain amount of bytes and returns a long.

Parameters:

`size` - the number of bytes that have to be read. Allowed values for this parameter are in the range (5, 8].

Returns:

the long resulting from the bytes read.

Throws:

`IOException` - If an I/O error occurs.

`EOFException` - if there are not at least `size` bytes left on the data input.

readFloat

```
float readFloat(int size)
    throws IOException
```

Parameters:

`size` - expected value for this parameter are 2 or 4 depending if reading [ZigBeeDataTypes.FLOATING_SEMI](#) or [ZigBeeDataTypes.FLOATING_SINGLE](#)

Returns:

a decoded float

Throws:

`IOException` - If an I/O error occurs.

`EOFException` - if there are not at least `size` bytes left on the data input.

readDouble

```
double readDouble()
    throws IOException
```

Returns:

a decoded double.

Throws:

`IOException` - If an I/O error occurs.

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

readBytes

```
byte[] readBytes(int len)
    throws IOException
```

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

Parameters:

`len` - the number of bytes to read.

Returns:

return a copy of the byte contained in the stream

Throws:

`IOException` - If an I/O error occurs.

`EOFException` - if there are not at least `len` bytes left on the data input.

Interface ZigBeeDataOutput

org.osgi.service.zigbee

```
public interface ZigBeeDataOutput
```

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

Method Summary		Page
void	writeByte (byte value) Appends a byte to the data output	80
void	writeBytes (byte[] bytes, int length) Appends on the Data Output Stream a bytes value	81
void	writeDouble (double value) Appends on the Data Output Stream a double value	81
void	writeFloat (float value, int size) Appends on the Data Output Stream a float value	81
void	writeInt (int value, int size) Appends an int value to the data output.	80
void	writeLong (long value, int size) Appends a long to to the data output.	81

Method Detail

`writeByte`

```
void writeByte(byte value)
```

Appends a byte to the data output

Parameters:

`value` - The value to append

`writeInt`

```
void writeInt(int value,  
              int size)
```

Appends an int value to the data output.

Parameters:

`value` - The int value to append

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

writeLong

```
void writeLong(long value,  
               int size)
```

Appends a long to to the data output.

Parameters:

value - The long value to append

size - The size in bytes that have to be actually appended.

writeFloat

```
void writeFloat(float value,  
                int size)
```

Appends on the Data Output Stream a float value

Parameters:

value - The float value to append

size - The size in bytes that have to be actually appended.

writeDouble

```
void writeDouble(double value)
```

Appends on the Data Output Stream a double value

Parameters:

value - The double value to append

writeBytes

```
void writeBytes(byte[] bytes,  
                int length)
```

Appends on the Data Output Stream a bytes value

Parameters:

bytes - The bytes value to append

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

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.*`. This constants are not the same provided by the ZigBee specification, and follows the rules below:

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

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

bit 7: if one the data type represents a unsigned value, otherwise it is signed.

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

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.	88
static short	ATTRIBUTE_ID 2.5.2.23 Attribute ID This type represents an attribute identifier as defined in spec.	89
static short	BACNET_OID 2.5.2.24 BACnet OID (Object Identifier) The BACnet OID data type is included to allow interworking with BACnet.	89
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.	89
static short	BITMAP_16	86
static short	BITMAP_24	86
static short	BITMAP_32	86
static short	BITMAP_40	86
static short	BITMAP_48	86
static short	BITMAP_56	86
static short	BITMAP_64	86
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.	85
static short	BOOLEAN 2.5.2.3 Boolean The Boolean type represents a logical value, either FALSE (0x00) or TRUE (0x01).	85

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.	88
static short	CLUSTER_ID 2.5.2.22 Cluster ID This type represents a cluster identifier as defined in spec.	89
static short	DATE 2.5.2.20 Date The Time of day data type shall be formatted as illustrated in spec.	89
static short	ENUMERATION_16	87
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.	87
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.	88
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.	87
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.	87
static short	GENERAL_DATA_16	85
static short	GENERAL_DATA_24	85
static short	GENERAL_DATA_32	85
static short	GENERAL_DATA_40	85
static short	GENERAL_DATA_48	85
static short	GENERAL_DATA_56	85
static short	GENERAL_DATA_64	85
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.	85
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.	89
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.	88
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.	88
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.	85
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.	88

static short	SECURITY KEY 128 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.	90
static short	SET 2.5.2.17 Set A set is a collection of elements with no associated order.	89
static short	SIGNED INTEGER 16	87
static short	SIGNED INTEGER 24	87
static short	SIGNED INTEGER 32	87
static short	SIGNED INTEGER 40	87
static short	SIGNED INTEGER 48	87
static short	SIGNED INTEGER 56	87
static short	SIGNED INTEGER 64	87
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 $-(2^{n-1})$ to $2^{n-1}-1$, where n is the number of bits. For example, for 8-bit, the range is $-(2^7-1)$ to 2^7-1 , for 16-bit, the range is $-(2^{15}-1)$ to $2^{15}-1$, for 24-bit, the range is $-(2^{23}-1)$ to $2^{23}-1$, for 32-bit, the range is $-(2^{31}-1)$ to $2^{31}-1$, for 40-bit, the range is $-(2^{39}-1)$ to $2^{39}-1$, for 48-bit, the range is $-(2^{47}-1)$ to $2^{47}-1$, for 56-bit, the range is $-(2^{55}-1)$ to $2^{55}-1$, or for 64-bit, the range is $-(2^{63}-1)$ to $2^{63}-1$, depending on its length.	87
static short	STRUCTURE 2.5.2.16 Structure A structure is an ordered sequence of elements, which may be of different data types.	88
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.	89
static short	UNKNOWN UNKNOWN = 0xff	90
static short	UNSIGNED INTEGER 16	86
static short	UNSIGNED INTEGER 24	86
static short	UNSIGNED INTEGER 32	86
static short	UNSIGNED INTEGER 40	86
static short	UNSIGNED INTEGER 48	86
static short	UNSIGNED INTEGER 56	86
static short	UNSIGNED INTEGER 64	87
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^n-1 , where n is the number of bits. For example, for 8-bit, the range is 0 to 2^8-1 , for 16-bit, the range is 0 to $2^{16}-1$, for 24-bit, the range is 0 to $2^{24}-1$, for 32-bit, the range is 0 to $2^{32}-1$, for 40-bit, the range is 0 to $2^{40}-1$, for 48-bit, the range is 0 to $2^{48}-1$, for 56-bit, the range is 0 to $2^{56}-1$, or for 64-bit, the range is 0 to $2^{64}-1$, depending on its length.	86
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).	89

Constructor Summary	Page
ZigBeeDataTypes ()	90

Method Summary		Page
static Object	deserializeDataType (ZigBeeDataInput is, short dataType)	90
static void	serializeDataType (ZigBeeDataOutput os, short dataType, Object value) Marshal the passed value into a ZigBeeDataOutput stream, according the dataType argument.	90

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 = 80
```

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 = 81
```

GENERAL_DATA_24

```
public static final short GENERAL_DATA_24 = 82
```

GENERAL_DATA_32

```
public static final short GENERAL_DATA_32 = 83
```

GENERAL_DATA_40

```
public static final short GENERAL_DATA_40 = 84
```

GENERAL_DATA_48

```
public static final short GENERAL_DATA_48 = 85
```

GENERAL_DATA_56

```
public static final short GENERAL_DATA_56 = 86
```

GENERAL_DATA_64

```
public static final short GENERAL_DATA_64 = 87
```

BOOLEAN

```
public static final short BOOLEAN = 1
```

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 = 88
```

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 = 89
```

BITMAP_24

```
public static final short BITMAP_24 = 90
```

BITMAP_32

```
public static final short BITMAP_32 = 91
```

BITMAP_40

```
public static final short BITMAP_40 = 92
```

BITMAP_48

```
public static final short BITMAP_48 = 93
```

BITMAP_56

```
public static final short BITMAP_56 = 94
```

BITMAP_64

```
public static final short BITMAP_64 = 95
```

UNSIGNED_INTEGER_8

```
public static final short UNSIGNED_INTEGER_8 = 96
```

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, 0xffffffffff, 0xffffffff, 0xfffffffff and 0xfffffffffff respectively.

UNSIGNED_INTEGER_16

```
public static final short UNSIGNED_INTEGER_16 = 97
```

UNSIGNED_INTEGER_24

```
public static final short UNSIGNED_INTEGER_24 = 98
```

UNSIGNED_INTEGER_32

```
public static final short UNSIGNED_INTEGER_32 = 99
```

UNSIGNED_INTEGER_40

```
public static final short UNSIGNED_INTEGER_40 = 100
```

UNSIGNED_INTEGER_48

```
public static final short UNSIGNED_INTEGER_48 = 101
```

UNSIGNED_INTEGER_56

```
public static final short UNSIGNED_INTEGER_56 = 102
```

UNSIGNED_INTEGER_64

```
public static final short UNSIGNED_INTEGER_64 = 103
```

SIGNED_INTEGER_8

```
public static final short SIGNED_INTEGER_8 = 224
```

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 = 225
```

SIGNED_INTEGER_24

```
public static final short SIGNED_INTEGER_24 = 226
```

SIGNED_INTEGER_32

```
public static final short SIGNED_INTEGER_32 = 227
```

SIGNED_INTEGER_40

```
public static final short SIGNED_INTEGER_40 = 228
```

SIGNED_INTEGER_48

```
public static final short SIGNED_INTEGER_48 = 229
```

SIGNED_INTEGER_56

```
public static final short SIGNED_INTEGER_56 = 230
```

SIGNED_INTEGER_64

```
public static final short SIGNED_INTEGER_64 = 231
```

ENUMERATION_8

```
public static final short ENUMERATION_8 = 112
```

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 = 113
```

FLOATING_SEMI

```
public static final short FLOATING_SEMI = 248
```

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 = 249
```

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 = 250
```

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 = 120
```

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 = 121
```

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 = 122
```

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 = 123
```

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 = 16
```

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 = 17
```

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.

SFT

```
public static final short SET = 18
```

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 = 19
```

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 = 2
```

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 = 3
```

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 = 4
```

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 = 5
```

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 = 6
```

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 = 7
```

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 = 8
```

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.

SECURITY_KEY_128

```
public static final short SECURITY_KEY_128 = 9
```

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.

UNKNOWN

```
public static final short UNKNOWN = 255
```

UNKNOWN = 0xff

Constructor Detail

ZigBeeDataTypes

```
public ZigBeeDataTypes()
```

Method Detail

serializeDataType

```
public static void serializeDataType(ZigBeeDataOutput os,  
                                     short dataType,  
                                     Object value)
```

Marshal the passed value into a `ZigBeeDataOutput` stream, according the `dataType` argument. An `IllegalArgumentException` is thrown when the the passed value does not belong to the class allowed for the `dataType`.

If the data type allows that, a null value is marshaled into ZCL Invalid Value for that data type.

Parameters:

`os` - a [ZigBeeDataOutput](#) stream where to stream the value. This parameter cannot be null. If it is nul a `NullPointerException` is thrown.

`dataType` - The data type that have to be marshaled on the output stream.

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

Throws:

`IllegalArgumentException` - Thrown when the the passed value does not belong to the allowed class for the `dataType` as described in the specification or when the value exceed the range allowed by that type (i.e. length for Octet String data types).

deserializeDataType

```
public static Object deserializeDataType(ZigBeeDataInput is,  
                                         short dataType)  
    throws IOException
```

Parameters:

`is` - A valid [ZigBeeDataInput](#) stream instance. This parameter cannot be null.

`dataType` - The data type that have to be deserialized. This value must be one of the valid data types constants defined at the beginning of this interface, otherwise an `IllegalArgumentException` exception is thrown.

Returns:

The deserialized object. The returned value is null if the deserialized value is equal to the *Invalid Value* for the specified dataType.

Throws:

IOException - in case of problems while deserializing the ZigBeeDataInput. For instance this could happen if in the stream the remaining bytes are not enough to be able to unmarshal the requested data type.

IllegalArgumentException - if the passed dataType is not supported.

NullPointerException - if the passed ZigBeeDataInput is null

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.

Field Summary		Page
String	DEVICE_CATEGORY Constant used by all ZigBee devices indicating the device category.	94
String	DEVICE_ID Key of the String property containing the DeviceId of the device It is mandatory property for this service	93
String	DEVICE_VERSION Key of the String property containing the DeviceVersion of the device It is mandatory property for this service	94
String	ENDPOINT_ID Key of the String property containing the EndPoint Address of the device It is mandatory property for this service	93
String	HOST_PID Key of String containing the ZigBeeHost 's pid.	93
String	INPUT_CLUSTERS Key of the int array of containing the ids of each input cluster It is mandatory property for this service	94
String	OUTPUT_CLUSTERS Key of the int array of containing the ids of each output cluster It is mandatory property for this service	94
String	PROFILE_ID Key of the String property containing the profile id implemented by the device.	93
String	ZIGBEE_EXPORT Key of the String property mentioning that an endpoint is an exported one or not.	94

Method Summary		Page
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).	95
void	getBoundEndpoints (int clusterId, ZigBeeHandler handler) This method is used to get bound endpoints (identified by their service PIDs).	96

ZCLCluster	getClientCluster (int clientClusterId)	95
ZCLCluster []	getClientClusters ()	95
short	getId ()	94
BigInteger	getNodeAddress ()	94
ZCLCluster	getServerCluster (int serverClusterId)	95
ZCLCluster []	getServerClusters ()	95
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.	94
void	notExported (ZigBeeException e) This method is used to get details about problems when an error occurs during exporting an endpoint	96
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).	96

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

```
short getId()
```

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

getNodeAddress

```
BigInteger 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_descr 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 it does not provide any server cluster.

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 [APSException](#)).

The response object given to the handler is a Boolean set to true if the binding succeeds. In case of an error has occurred, onFailure is called with a APSEException.

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 a Boolean set to true if the unbinding succeeds. In case of an error has occurred, onFailure is called with a APSEException.

Parameters:

servicePid - to unbound from
clusterId - The cluster identifier to unbound from

notExported

```
void notExported(ZigBeeException e)
```

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

Parameters:

e - A device [ZigBeeException](#) 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 bound endpoint service PIDs.

Interface ZigBeeEvent

org.osgi.service.zigbee

```
public interface ZigBeeEvent
```

This interface represents events generated by a ZigBee Device node

Method Summary		Page
int	getAttributeId()	97
int	getClusterId()	97
short	getEndpointId()	97
BigInteger	getIEEEAddress()	97
Object	getValue()	97

Method Detail

getIEEEAddress

```
BigInteger getIEEEAddress()
```

Returns:

The ZigBee device node IEEE Address.

getEndpointId

```
short 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.

Class ZigBeeException

org.osgi.service.zigbee

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

All Implemented Interfaces:

Serializable

Direct Known Subclasses:

[APSEException](#), [ZCLEException](#), [ZDPEException](#)

```
public class ZigBeeException
extends RuntimeException
```

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

Field Summary		Page
protected int	errorCode The error code associated to this exception	100
static int	OSGI_EXISTING_ID OSGI_EXISTING_ID (16) â€” another endpoint exists with the same ID.	100
static int	OSGI_MULTIPLE_HOSTS OSGI_MULTIPLE_HOSTS (17) â€” several hosts exist for this PAN ID target or HOST_PID target.	100
static int	UNKNOWN_ERROR This error code is used if the ZigBee error returned is not covered by this API specification.	100
protected int	zigBeeErrorCode The actual error code returned by the ZigBee node.	100

Constructor Summary		Page
ZigBeeException (int errorCode, int zigBeeErrorCode, String errorDesc)	Create a ZigBeeException containing a specific errorCode OR zigBeeErrorCode.	101
ZigBeeException (int errorCode, String errorDesc)	Create a ZigBeeException containing a specific errorCode.	100
ZigBeeException (String errorDesc)	Create a ZigBeeException containing only a description, but no error codes.	100

Method Summary		Page
int	getErrorCode ()	101
int	getZigBeeErrorCode ()	101
boolean	hasZigbeeErrorCode ()	101

Field Detail

OSGI_EXISTING_ID

```
public static final int OSGI_EXISTING_ID = 48
```

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

OSGI_MULTIPLE_HOSTS

```
public static final int OSGI_MULTIPLE_HOSTS = 49
```

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

UNKNOWN_ERROR

```
public static final int UNKNOWN_ERROR = -1
```

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

errorCode

```
protected final int errorCode
```

The error code associated to this exception

See Also:

[getErrorCode\(\)](#)

zigBeeErrorCode

```
protected final int zigBeeErrorCode
```

The actual error code returned by the ZigBee node.

See Also:

[getZigBeeErrorCode\(\)](#)

Constructor Detail

ZigBeeException

```
public ZigBeeException(String errorDesc)
```

Create a [ZigBeeException](#) containing only a description, but no error codes. If issued on this exception the [getErrorCode\(\)](#) and [getZigBeeErrorCode\(\)](#) methods return the [UNKNOWN_ERROR](#) constant.

Parameters:

errorDesc - exception error description

ZigBeeException

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

Create a [ZigBeeException](#) containing a specific errorCode. Using this constructor with errorCode set to [UNKNOWN_ERROR](#) is equivalent to call [ZigBeeException\(String\)](#).

Parameters:

errorCode - One of the error codes defined in this interface or [UNKNOWN_ERROR](#) if the actual error is not listed in this interface.

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

ZigBeeException

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

Create a [ZigBeeException](#) containing a specific **errorCode** or **zigBeeErrorCode**. Using this constructor with both the **errorCode** and **zigBeeErrorCode** set to [UNKNOWN_ERROR](#) is equivalent to call [ZigBeeException\(String\)](#).

Parameters:

errorCode - One of the error codes defined in this interface or [UNKNOWN_ERROR](#) the actual error is not covered in this interface.

zigBeeErrorCode - The actual status code or [UNKNOWN_ERROR](#) if this status is unknown.

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

Method Detail

getZigBeeErrorCode

```
public int getZigBeeErrorCode()
```

Returns:

One of the error codes defined above. If the returned error code is [UNKNOWN_ERROR](#) and the [hasZigbeeErrorCode\(\)](#) returns **true** then the [getZigBeeErrorCode\(\)](#) provides the actual ZigBee error code returned by the device.

getErrorCode

```
public int getErrorCode()
```

Returns:

the error code.

hasZigbeeErrorCode

```
public boolean hasZigbeeErrorCode()
```

Returns:

true if the [ZigBeeException](#) convey also the actual error code returned by the ZigBee stack.

Interface ZigBeeGroup

org.osgi.service.zigbee

```
public interface ZigBeeGroup
```

This interface represents a ZigBee Group

Field Summary		Page
String	ID Key of the String containing the Group Address of the device.	102

Method Summary		Page
int	getGroupAddress()	102
void	invoke (Integer clusterId, ZCLFrame frame, ZCLCommandHandler handler) Invokes the action on a Group.	103
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.	104
void	joinGroup (String pid, ZigBeeHandler handler) This method is used for adding an Endpoint to a Group, it may be invoked on exported Endpoint or even on imported Endpoint.	102
void	leaveGroup (String pid, ZigBeeHandler handler) This method is used for adding an Endpoint to a Group, it may be invoked on exported Endpoint or even on imported Endpoint.	103

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

```
int getGroupAddress()
```

Returns:
The 16bit group address.

joinGroup

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

This method is used for adding an Endpoint to a Group, it may be invoked on exported Endpoint or even on imported Endpoint. In the former case, the ZigBee Base Driver should rely on the *APSME-ADD-GROUP* API defined by the ZigBee Specification, or it will use the proper commands of the *Groups* cluster of the ZigBee Specification Library. As described in "Table 2.15 APSME-ADD-GROUP.confirm Parameters" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a `add_group` request can have the following status: SUCCESS, INVALID_PARAMETER or TABLE_FULL (see [APSEException](#)).

Parameters:

`pid` - String representing the service PID (see `org.osgi.framework.Constants.SERVICE_PID`) of the [ZigBeeEndpoint](#) to add to this Group.
`handler` - the handler that will notified of the result of "joining". The expected object is always a Boolean indicating a failure or a success

Throws:

[APSEException](#) - when the joining is performed locally on an exported [ZigBeeEndpoint](#) and it fails either with error code INVALID_PARAMETER or TABLE_FULL. This exception is also generated when the joining is performed remotely on an imported [ZigBeeEndpoint](#) and the communication with it fails
[ZCLEException](#) - when the joining is performed remotely on an imported [ZigBeeEndpoint](#) and it fails either because the command is not supported by the remote End Point, or the remote device cannot perform the operation at the moment.

leaveGroup

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

This method is used for adding an Endpoint to a Group, it may be invoked on exported Endpoint or even on imported Endpoint. In the former case, the ZigBee Base Driver should rely on the *APSME-REMOVE-GROUP* API defined by the ZigBee Specification, or it will use the proper commands of the *Groups* cluster of the ZigBee Specification Library. As described in "Table 2.17 APSME-REMOVE-GROUP.confirm Parameters" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, a `remove_group` request can have the following status: SUCCESS, INVALID_GROUP or INVALID_PARAMETER (see [APSEException](#)).

Parameters:

`pid` - String representing the service PID (see `org.osgi.framework.Constants.SERVICE_PID`) of the [ZigBeeEndpoint](#) to remove from this Group.
`handler` - the handler that will notified of the result of "joining". The expected object is always a Boolean indicating a failure or a success

Throws:

[APSEException](#) - when the joining is performed locally on an exported [ZigBeeEndpoint](#) and it fails either with error code INVALID_PARAMETER or INVALID_GROUP. This exception is also generated when the joining is performed remotely on an imported [ZigBeeEndpoint](#) and the communication with it fails
[ZCLEException](#) - when the joining is performed remotely on an imported [ZigBeeEndpoint](#) and it fails either because the command is not supported by the remote End Point, or the remote device cannot perform the operation at the moment.

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

Method Summary		Page
void	onFailure (Exception e) Notifies the failure result of the call.	105
void	onSuccess (Object response) Notifies the success result of the call.	105

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.

Field Summary		Page
short	UNLIMITED_BROADCAST_RADIUS UNLIMITED_BROADCAST_RADIUS	107

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

[COORDINATOR](#), [EXTENDED PAN ID](#), [IEEE ADDRESS](#), [LOGICAL TYPE](#), [MANUFACTURER CODE](#), [PAN ID](#), [POWER SOURCE](#), [RECEIVER ON WHEN IDLE](#), [ROUTER](#), [ZED](#)

Method Summary		Page
void	broadcast (int clusterID, ZCLFrame frame, ZCLCommandHandler handler) Enable to broadcast a given frame on a given cluster.	111
void	broadcast (int clusterID, ZCLFrame frame, ZCLCommandHandler handler, String exportedServicePID) Enable to broadcast a given frame on a given cluster.	111
void	createGroupService (int groupAddress) 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.	111
short	getBroadcastRadius ()	112
int	getChannel ()	109
int	getChannelMask ()	109
String	getPreconfiguredLinkKey ()	110
int	getSecurityLevel ()	110
boolean	isStarted () Get the host's start/stop state.	108
void	permitJoin (short duration) Indicates if a ZigBee device can join the network.	108
void	refreshNetwork (ZigBeeHandler handler) The method forces a new scan.	110
void	setBroadcastRadius (short broadcastRadius) By default the ZigBeeHost must use UNLIMITED_BROADCAST_RADIUS as default value for the broadcast	112

void	setChannelMask (int mask) Set a new configured channel mask.	110
void	setExtendedPanId (long extendedPanId) Set the extendedPanId.	108
void	setLogicalType (short logicalNodeType) Sets the host logical node type.	109
void	setPanId (int panId) Set the panId.	108
void	start () Starts the host.	107
void	stop () Stops the host.	107
void	updateNetworkChannel (byte channel) Updates the network channel. 802.15.4 and ZigBee divide the 2.4Ghz band into 16 channels, numbered from 11 to 26.	109

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

[getComplexDescriptor](#), [getEndpoints](#), [getExtendedPanId](#), [getHostPid](#), [getIEEEAddress](#), [getLinksQuality](#), [getNetworkAddress](#), [getNodeDescriptor](#), [getPanId](#), [getPowerDescriptor](#), [getRoutingTable](#), [getUserDescription](#), [invoke](#), [invoke](#), [leave](#), [leave](#), [setUserDescription](#)

Field Detail

UNLIMITED_BROADCAST_RADIUS

```
public static final short UNLIMITED_BROADCAST_RADIUS = 255
```

UNLIMITED_BROADCAST_RADIUS

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`
`Exception`, - any exception related to the communication with the chip.

stop

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

Stops the host.

Throws:

`Exception`

`Exception`, - any exception related to the communication with the chip.

isStarted

```
boolean isStarted()
```

Get the host's start/stop state.

Returns:

true if the host is started.

setPanId

```
void setPanId(int panId)
    throws IllegalStateException
```

Set the panId.

Parameters:

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

Throws:

`IllegalStateException`

`IllegalStateException`, - is thrown in case the host is still started.

setExtendedPanId

```
void setExtendedPanId(long extendedPanId)
    throws IllegalStateException
```

Set the extendedPanId.

Parameters:

`extendedPanId` - The network Extended PAN identifier(EPID)

Throws:

`IllegalStateException`

`IllegalStateException`, - is thrown in case the host is still started.

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

Exception, - any exception related to the communication with the chip.

setLogicalType

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

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

Parameters:

logicalNodeType - The logical node type.

Throws:

IllegalStateException

Exception

IllegalStateException, - is thrown in case the host is still started.

Exception, - any exception related to the communication with the chip.

getChannel

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

Returns:

The current network channel.

Throws:

Exception

Exception, - any exception related to the communication with the chip.

updateNetworkChannel

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

Updates the network channel. 802.15.4 and ZigBee divide the 2.4Ghz band into 16 channels, numbered from 11 to 26. As described in "Table 2.4.3.3.9 Mgmt_NWK_Update_req" of the ZigBee specification 1_053474r17ZB_TSC-ZigBee-Specification.pdf, this request is sent as broadcast by the network manager with a ScanDuration to be set with the channel parameter.

Parameters:

channel - The network channel.

Throws:

IllegalStateException

IOException - for serial communication exception.

IllegalStateException, - is thrown in case the host is still started, or in case the host is not a network manager.

getChannelMask

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

Returns:

The currently configured channel mask.

Throws:

`Exception`

`Exception`, - any exception related to the communication with the chip.

setChannelMask

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

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:

`IllegalStateException`

`IOException` - for serial communication exception.

`IllegalStateException`, - is thrown in case the host is still started.

refreshNetwork

```
void refreshNetwork(ZigBeeHandler handler)
    throws Exception
```

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

Parameters:

`handler` - in case of success `handler.onSuccess(true)` is called, `handler.onFailure(any Exception)` is called otherwise.

Throws:

`Exception`

`Exception`, - any exception related to the communication with the chip.

getSecurityLevel

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

Returns:

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

Throws:

`Exception`

`Exception`, - any exception related to the communication with the chip.

getPreconfiguredLinkKey

```
String getPreconfiguredLinkKey()
    throws Exception
```

Returns:

The current preconfigured link key.

Throws:

[Exception](#)

Exception, - any exception related to the communication with the chip.

createGroupService

```
void createGroupService(int groupAddress)
    throws Exception
```

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.

Parameters:

groupAddress - the address of the group to create.

Throws:

Exception - when a ZigBeeGroup service with the same groupAddress already exists.

broadcast

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

Enable to broadcast a given frame on a given cluster.

Specified by:

[broadcast](#) in interface [ZigBeeNode](#)

Parameters:

clusterID - the cluster ID.

frame - a command frame sequence.

handler - The handler that manages the command response.

See Also:

[for setting the broadcast radius](#)

broadcast

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

Enable to broadcast a given frame on a given cluster.

Specified by:

[broadcast](#) in interface [ZigBeeNode](#)

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.

See Also:

[for setting the broadcast radius](#)

getBroadcastRadius

```
short getBroadcastRadius()
```

Returns:
the current broadcastradius value.

setBroadcastRadius

```
void setBroadcastRadius(short broadcastRadius)
    throws IllegalArgumentException,
        IllegalStateException
```

By default the [ZigBeeHost](#) must use [UNLIMITED_BROADCAST_RADIUS](#) as default value for the broadcast

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

Throws:
IllegalArgumentException - if set with a value out of the expected range.
IllegalStateException - if set when the ZigBeeHost is "running".

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)

Field Summary		Page
int	CHILD_NEIGHBOR Constant value representing a child relationship between current ZigBeeNode and the neighbor	113
int	OTHERS_NEIGHBOR Constant value representing a others relationship between current ZigBeeNode and the neighbor	114
int	PARENT_NEIGHBOR * Constant value representing a parent relationship between current ZigBeeNode and the neighbor	113
int	PREVIOUS_CHILD_NEIGHBOR Constant value representing a previous child relationship between current ZigBeeNode and the neighbor	114
int	SIBLING_NEIGHBOR Constant value representing a sibling relationship between current ZigBeeNode and the neighbor	114

Method Summary		Page
int	getDepth() See the Depth field of the (NeighborTableList Record Format).	114
int	getLQI() See the LQI field of the (NeighborTableList Record Format).	114
String	getNeighbor()	114
int	getRelationship() See the Relationship field of the (NeighborTableList Record Format).	114

Field Detail

PARENT_NEIGHBOR

```
public static final int PARENT_NEIGHBOR = 240
```

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

CHILD_NEIGHBOR

```
public static final int CHILD_NEIGHBOR = 241
```

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

SIBLING_NEIGHBOR

```
public static final int SIBLING_NEIGHBOR = 242
```

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

OTHERS_NEIGHBOR

```
public static final int OTHERS_NEIGHBOR = 243
```

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

PREVIOUS_CHILD_NEIGHBOR

```
public static final int PREVIOUS_CHILD_NEIGHBOR = 244
```

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

Field Summary		Page
short	COORDINATOR The Node is a ZigBee Coordinator	118
String	EXTENDED PAN ID Key of String containing the device node network extended PAN ID.	117
String	IEEE ADDRESS Property key for the mandatory node IEEE Address representing node MAC address.	117
String	LOGICAL TYPE Property key for the device logical type	117
String	MANUFACTURER CODE Property key for a manufacturer code that is allocated by the ZigBee Alliance, relating the manufacturer to the device.	117
String	PAN ID Key of String containing the device node network PAN ID	117
String	POWER SOURCE ZigBee power source, i.e. 3rd bit of "MAC Capabilities" in Node Descriptor.	117
String	RECEIVER ON WHEN IDLE ZigBee receiver on when idle, i.e. 4th bit of "MAC Capabilities" in Node Descriptor.	117
short	ROUTER The Node is a ZigBee Router	118
short	ZED The Node is a ZigBee End Device	117

Method Summary		Page
void	broadcast (int clusterID, ZCLFrame frame, ZCLCommandHandler handler) Enable to broadcast a given frame of a specific cluster to all the ZigBeeEndpoint that are running on this node.	121

void	<u>broadcast</u> (int clusterID, <u>ZCLFrame</u> frame, <u>ZCLCommandHandler</u> handler, String exportedServicePID) Enable to broadcast a given frame of a specific cluster to all the <u>ZigBeeEndpoint</u> that are running on this node from a specific exported endpoint.	121
void	<u>getComplexDescriptor</u> (<u>ZigBeeHandler</u> 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.	119
<u>ZigBeeEndpoint</u> []	<u>getEndpoints</u> ()	118
BigInteger	<u>getExtendedPanId</u> ()	118
String	<u>getHostPid</u> ()	118
BigInteger	<u>getIEEEAddress</u> ()	118
void	<u>getLinksQuality</u> (<u>ZigBeeHandler</u> 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).	119
int	<u>getNetworkAddress</u> ()	118
void	<u>getNodeDescriptor</u> (<u>ZigBeeHandler</u> 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.	119
int	<u>getPanId</u> ()	118
void	<u>getPowerDescriptor</u> (<u>ZigBeeHandler</u> 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.	119
void	<u>getRoutingTable</u> (<u>ZigBeeHandler</u> 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).	120
void	<u>getUserDescription</u> (<u>ZigBeeHandler</u> 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.	121
void	<u>invoke</u> (int clusterIdReq, int expectedClusterIdRsp, <u>ZDPFrame</u> message, <u>ZDPHandler</u> handler) This method sends the <u>ZDPFrame</u> to this <u>ZigBeeNode</u> with the specified cluster id and it will expect a specific cluster as response to the request.	120
void	<u>invoke</u> (int clusterIdReq, <u>ZDPFrame</u> message, <u>ZDPHandler</u> handler) This method sends the <u>ZDPFrame</u> to this <u>ZigBeeNode</u> with the specified cluster id and it will expect a specific cluster as response to the request This method considers that the 0x8000 + clusterIdReq is the clusterId expected from messaged received for the message sent by this request.	121
void	<u>leave</u> (boolean rejoin, boolean removeChildren, <u>ZigBeeHandler</u> handler) Requests the device to leave the network.	120
void	<u>leave</u> (<u>ZigBeeHandler</u> handler) Request to leave the network.	120

void	setUserDescription (String userDescription, 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.	122
------	---	-----

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.

ZED

```
public static final short ZED = 1
```

The Node is a ZigBee End Device

COORDINATOR

```
public static final short COORDINATOR = 2
```

The Node is a ZigBee Coordinator

ROUTER

```
public static final short ROUTER = 3
```

The Node is a ZigBee Router

Method Detail

getIEEEAddress

```
BigInteger 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

```
BigInteger getExtendedPanId()
```

Returns:

The network Extended PAN identifier(EPID)

getEndpoints

```
ZigBeeEndpoint[] getEndpoints()
```

Returns:

An array of embedded endpoints, returns an empty array if it does not provide any endpoint.

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.

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 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 [ZDPException](#)).

The response object given to the handler is a Boolean set to true if the leave succeeds. In case of an error has occurred, onFailure is called with a ZCLException.

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 [ZDPException](#)).

The response object given to the handler is a Boolean set to true if the leave succeeds. In case of an error has occurred, onFailure is called with a ZCLException.

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

invoke

```
void invoke(int clusterIdReq,  
            int expectedClusterIdRsp,  
            ZDPFrame message,  
            ZDPHandler handler)
```

This method sends the [ZDPFrame](#) to this [ZigBeeNode](#) with the specified cluster id and it will expect a specific cluster as response to the request.

Parameters:

clusterIdReq - the cluster Id of the [ZDPFrame](#) that will be sent to the device.

expectedClusterIdRsp - the expected cluster Id of the response to the [ZDPFrame](#) sent.
message - the [ZDPFrame](#) containing the message.
handler - The handler for the response to the [ZDPFrame](#).

invoke

```
void invoke(int clusterIdReq,  
            ZDPFrame message,  
            ZDPHandler handler)
```

This method sends the [ZDPFrame](#) to this [ZigBeeNode](#) with the specified cluster id and it will expect a specific cluster as response to the request. This method considers that the 0x8000 + clusterIdReq is the clusterId expected from message received for the message sent by this request.

Parameters:

clusterIdReq - the cluster Id of the [ZDPFrame](#) that will be sent to the device
message - the [ZDPFrame](#) containing the message
handler - The handler for the response to the [ZDPFrame](#)

broadcast

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

Enable to broadcast a given frame of a specific cluster to all the [ZigBeeEndpoint](#) that are running on this node.

Parameters:

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

broadcast

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

Enable to broadcast a given frame of a specific cluster to all the [ZigBeeEndpoint](#) that are running on this node from a specific exported endpoint.

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.

getUserDescription

```
void getUserDescription(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. These constants are defined in [ZDPException](#).

Parameters:

`handler` - that will be used in order to return the node user description. Return an exception with NO_DESCRIPTOR (via `handler.onFailure(...)`) whether no user descriptor is available.

setUserDescription

```
void setUserDescription(String userDescription,  
                        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. These constants are defined in [ZDPException](#).

Parameters:

`userDescription` - the user description
`handler` - the response 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)

Field Summary		Page
int	ACTIVE Constant value representing an active route	123
int	DISCOVERY_FAILED Constant value representing a failed route discovery	123
int	DISCOVERY_UNDERWAY Constant value representing a route that is under discovery	123
int	INACTIVE Constant value representing an inactive route	123
int	VALIDATION_UNDERWAY Constant value representing a route which is under validation	124

Method Summary		Page
String	getDestination()	124
String	getNextHop()	124
int	getStatus()	124

Field Detail

ACTIVE

```
public static final int ACTIVE = 240
```

Constant value representing an active route

DISCOVERY_UNDERWAY

```
public static final int DISCOVERY_UNDERWAY = 241
```

Constant value representing a route that is under discovery

DISCOVERY_FAILED

```
public static final int DISCOVERY_FAILED = 242
```

Constant value representing a failed route discovery

INACTIVE

```
public static final int INACTIVE = 243
```

Constant value representing an inactive route

VALIDATION_UNDERWAY

```
public static final int VALIDATION_UNDERWAY = 244
```

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_UNDERDAY

Package org.osgi.service.zigbee.descriptions

This package contains the interfaces for descriptions.

See:

[Description](#)

Interface Summary		Page
ZCLAttributeDescription	This interface represents a ZCLAttributeDescription	126
ZCLClusterDescription	This interface represents a ZCL Cluster description	128
ZCLCommandDescription	This interface represents a ZCLCommandDescription	130
ZCLDataTypeDescription	This interface is used for representing any of the ZigBee Data Types defined in the ZCL.	133
ZCLGlobalClusterDescription	This interface represents Cluster global description	135
ZCLParameterDescription	This interface represents a ZigBee parameter description	137
ZCLSimpleTypeDescription	This interface is used for representing any of the simple ZigBee Data Types defined in the ZCL.	138
ZigBeeDeviceDescription	This interface represents a ZigBee device description	140
ZigBeeDeviceDescriptionSet	This interface represents a ZigBee Device description Set.	142

Package org.osgi.service.zigbee.descriptions Description

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

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

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

Example import for consumers using the API in this package:

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

Example import for providers implementing the API in this package:

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

Interface ZCLAttributeDescription

org.osgi.service.zigbee.descriptions

All Superinterfaces:

[ZCLAttributeInfo](#)

```
public interface ZCLAttributeDescription
extends ZCLAttributeInfo
```

This interface represents a ZCLAttributeDescription

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

[ID](#)

Method Summary

Page

boolean	checkValue (Object value) checks whether the value object is conform to the attribute data type description	127
Object	getDefaultValue ()	126
String	getName ()	126
String	getShortDescription ()	126
boolean	isMandatory ()	127
boolean	isPartOfAScene ()	127
boolean	isReadOnly ()	127
boolean	isReportable ()	127

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

[getDataType](#), [getId](#), [getManufacturerCode](#), [isManufacturerSpecific](#)

Method Detail

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 optionally writable (R*W))

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

Method Summary		Page
ZCLAttributeDescription[]	getAttributeDescriptions()	128
ZCLCommandDescription[]	getGeneratedCommandDescriptions()	128
ZCLGlobalClusterDescription	getGlobalClusterDescription()	129
int	getId()	128
ZCLCommandDescription[]	getReceivedCommandDescriptions()	128

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

Method Summary		Page
Object[]	deserialize (ZCLFrame frame) Deserialize ZCLFrame to javaValues.	131
int	getId ()	130
int	getManufacturerCode () Get manufacturerCode Default value is: -1 (no code)	131
String	getName ()	130
ZCLParameterDescription []	getParameterDescriptions ()	131
String	getShortDescription ()	130
boolean	isClientServerDirection ()	132
boolean	isClusterSpecificCommand ()	131
boolean	isMandatory ()	131
boolean	isManufacturerSpecific ()	132
ZCLFrame	serialize (ZCLHeader header, Object[] javaValues) Serialize javaValues to a ZCLFrame that can then be used in invocations (e.g. via ZCLCluster, or ZigBeeGroup).	131

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 ZCLCluster, 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 ZCLCluster, 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

isManufacturerSpecific

```
boolean isManufacturerSpecific()
```

Returns:
true if end only if [getManufacturerCode\(\)](#) is not -1

isClientServerDirection

```
boolean isClientServerDirection()
```

Returns:
the isClientServerDirection value

Interface ZCLDataTypeDescription

org.osgi.service.zigbee.descriptions

All Known Subinterfaces:

[ZCLSimpleTypeDescription](#)

All Known Implementing Classes:

[ZigBeeArray](#), [ZigBeeAttributeID](#), [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](#), [ZigBeeSecurityKey128](#), [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 is used for representing any of the ZigBee Data Types defined in the ZCL. Each of these data types has a set of associated information that this interface definition permit to retrieve using the specific methods.

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

Method Summary		Page
short	getId()	133
Class	getJavaDataType()	134
String	getName()	133
boolean	isAnalog()	134

Method Detail

getId

```
short getId()
```

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

getName

```
String getName()
```

Returns:
The associated data type name string.

isAnalog

```
boolean isAnalog()
```

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

getJavaDataType

```
Class getJavaDataType()
```

Returns:
The corresponding Java type class.

Interface ZCLGlobalClusterDescription

org.osgi.service.zigbee.descriptions

```
public interface ZCLGlobalClusterDescription
```

This interface represents Cluster global description

Method Summary		Page
ZCLClusterDescription	getClientClusterDescription()	136
String	getClusterDescription()	135
String	getClusterFunctionalDomain()	135
int	getClusterId()	135
String	getClusterName()	135
ZCLClusterDescription	getServerClusterDescription()	136

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

Method Summary		Page
boolean	checkValue (Object value) checks whether the value object is conform to the parameter data type description	137
ZCLDataTypeDescription	getDataTypeDescription ()	137

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 ZCLSimpleTypeDescription

org.osgi.service.zigbee.descriptions

All Superinterfaces:

[ZCLDataTypeDescription](#)

All Known Implementing Classes:

[ZigBeeAttributeID](#), [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](#), [ZigBeeSecurityKey128](#), [ZigBeeSignedInteger16](#), [ZigBeeSignedInteger24](#), [ZigBeeSignedInteger32](#), [ZigBeeSignedInteger40](#), [ZigBeeSignedInteger48](#), [ZigBeeSignedInteger56](#), [ZigBeeSignedInteger64](#), [ZigBeeSignedInteger8](#), [ZigBeeTimeOfDay](#), [ZigBeeUnsignedInteger16](#), [ZigBeeUnsignedInteger24](#), [ZigBeeUnsignedInteger32](#), [ZigBeeUnsignedInteger40](#), [ZigBeeUnsignedInteger48](#), [ZigBeeUnsignedInteger56](#), [ZigBeeUnsignedInteger64](#), [ZigBeeUnsignedInteger8](#), [ZigBeeUTCTime](#)

```
public interface ZCLSimpleTypeDescription
extends ZCLDataTypeDescription
```

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

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

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

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	139
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	138

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

[getId](#), [getJavaDataType](#), [getName](#), [isAnalog](#)

Method Detail

serialize

```
void serialize(ZigBeeDataOutput os,
              Object value)
```

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

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

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

Throws:

`IllegalArgumentException` - Must be thrown if the passed value does not belong to the expected class or its value exceeds the possible values allowed (range or length).

deserialize

```
Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Parameters:

is - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Interface ZigBeeDeviceDescription

org.osgi.service.zigbee.descriptions

```
public interface ZigBeeDeviceDescription
```

This interface represents a ZigBee device description

Method Summary		Page
ZCLClusterDescription n[]	getClientClustersDescriptions()	141
int	getId()	140
String	getName()	140
int	getProfileId()	140
ZCLClusterDescription n[]	getServerClustersDescriptions()	141
Integer	getVersion()	140

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.

Field Summary		Page
String	DEVICES Property key for a comma separated list of devices identifiers supported by the set.	142
String	PROFILE_NAME Property key for a profile name.	142
String	VERSION Property key for a version of the application profile.	142

Method Summary		Page
ZigBeeDeviceDescription	getDeviceSpecification (int deviceId, short version)	142

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

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

See:

[Description](#)

Interface Summary		Page
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.	145
ZigBeeFrequencyBand	This interface represents a frequency band.	147
ZigBeeMacCapabilityFlags	This interface represents the Node Descriptor MAC Capability Flags as described in the ZigBee Specification.	148
ZigBeeNodeDescriptor	This interface represents a Node Descriptor as described in the ZigBee Specification The Node Descriptor contains information about the capabilities of the node.	150
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.	153
ZigBeeServerMask		156
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.	158

Package org.osgi.service.zigbee.descriptors Description

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

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

The `ZigBeeNodeDescriptor`, `ZigBeePowerDescriptor` and the `ZigBeeComplexDescriptor` are read using the appropriate methods in the interface, whereas the `ZigBeeSimpleDescriptor` can be read using the appropriate method of the services registered in the framework.

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

Example import for consumers using the API in this package:

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

Example import for providers implementing the API in this package:

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


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.

Method Summary		Page
String	getCharacterSetIdentifier()	145
String	getDeviceURL()	146
byte[]	getIcon()	146
String	getIconURL()	146
String	getLanguageCode()	145
String	getManufacturerName()	145
String	getModelName()	145
String	getSerialNumber()	146

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 ZigBeeFrequencyBand

org.osgi.service.zigbee.descriptors

```
public interface ZigBeeFrequencyBand
```

This interface represents a frequency band.

Method Summary

		Page
boolean	is2400 ()	147
boolean	is868 ()	147
boolean	is915 ()	147

Method Detail

is868

```
boolean is868()
```

Returns:

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

is915

```
boolean is915()
```

Returns:

true if and only if the radio is operating in the frequency band 908Mhz to 928Mhz

is2400

```
boolean is2400()
```

Returns:

true if and only if the radio is operating in the frequency band 2400Mhz to 2483Mhz

Interface ZigBeeMacCapabiliyFlags

org.osgi.service.zigbee.descriptors

```
public interface ZigBeeMacCapabiliyFlags
```

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

Method Summary		Page
boolean	isAddressAllocate()	149
boolean	isAlternatePANCoordinator()	148
boolean	isFullFunctionDevice()	148
boolean	isMainsPower()	148
boolean	isReceiverOnWhenIdle()	148
boolean	isSecurityCapable()	149

Method Detail

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 (FFD), false otherwise (it is a Reduced Function Device, RFD)

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.

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.

Method Summary		Page
ZigBeeFrequencyBand	getFrequencyBand()	151
short	getLogicalType()	150
ZigBeeMacCapabilityFlags	getMacCapabilityFlags()	151
int	getManufacturerCode()	151
int	getMaxBufferSize()	151
int	getMaxIncomingTransferSize()	151
int	getMaxOutgoingTransferSize()	151
ZigBeeServerMask	getServerMask()	151
boolean	isComplexDescriptorAvailable()	150
boolean	isExtendedActiveEndpointListAvailable()	151
boolean	isExtendedSimpleDescriptorListAvailable()	152
boolean	isUserDescriptorAvailable()	150

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

[ZigBeeFrequencyBand](#) getFrequencyBand()

Returns:

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

getMacCapabilityFlags

[ZigBeeMacCapabiliyFlags](#) getMacCapabilityFlags()

Returns:

the MAC Capability Flags field information

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

[ZigBeeServerMask](#) getServerMask()

Returns:

the server mask field.

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.

Field Summary		Page
short	CRITICAL_LEVEL Current power source level: critical.	153
short	FULL_LEVEL Current power source level: 100%.	154
short	LOW_LEVEL Current power source level: 33%.	153
short	MIDDLE_LEVEL Current power source level: 66%.	154

Method Summary		Page
short	getCurrentPowerMode()	154
short	getCurrentPowerSource()	154
short	getCurrentPowerSourceLevel()	154
boolean	isConstantMainsPowerAvailable()	155
boolean	isDisposableBattery()	154
boolean	isDisposableBatteryAvailable()	155
boolean	isMainsPower()	154
boolean	isOnWhenStimulated()	155
boolean	isPeriodicallyOn()	155
boolean	isRechargableBattery()	154
boolean	isRechargableBatteryAvailable()	155
boolean	isSynchronizedWithOnIdle() Returns true if synchronized with the receiver on-when-idle subfield of the node descriptor.	155

Field Detail

CRITICAL_LEVEL

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

Current power source level: critical.

LOW_LEVEL

```
public static final short LOW_LEVEL = 1
```

Current power source level: 33%.

MIDDLE_LEVEL

```
public static final short MIDDLE_LEVEL = 2
```

Current power source level: 66%.

FULL_LEVEL

```
public static final short FULL_LEVEL = 3
```

Current power source level: 100%.

Method Detail

getCurrentPowerMode

```
short getCurrentPowerMode()
```

Returns:
the current power mode.

getCurrentPowerSource

```
short getCurrentPowerSource()
```

Returns:
the current power source field of the Power Descriptor

isMainsPower

```
boolean isMainsPower()
```

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

isDisposableBattery

```
boolean isDisposableBattery()
```

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

isRechargeableBattery

```
boolean isRechargeableBattery()
```

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

getCurrentPowerSourceLevel

```
short getCurrentPowerSourceLevel()
```

Returns:
the current power source level. May be one of [CRITICAL_LEVEL](#), [LOW_LEVEL](#), [MIDDLE_LEVEL](#), [FULL_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.

isSynchronizedWithOnIdle

`boolean isSynchronizedWithOnIdle()`

Returns true if synchronized with the receiver on-when-idle subfield of the node descriptor.

Returns:

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

isPeriodicallyOn

`boolean isPeriodicallyOn()`

Returns:

true if the Current Power Mode field is periodically on.

isOnWhenStimulated

`boolean isOnWhenStimulated()`

Returns:

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

Interface ZigBeeServerMask

org.osgi.service.zigbee.descriptors

```
public interface ZigBeeServerMask
```

Method Summary		Page
boolean	isBackupBindingTableCache()	156
boolean	isBackupDiscoveryCache()	157
boolean	isBackupTrustCenter()	156
boolean	isNetworkManager()	157
boolean	isPrimaryBindingTableCache()	156
boolean	isPrimaryDiscoveryCache()	156
boolean	isPrimaryTrustCenter()	156

Method Detail

isPrimaryTrustCenter

```
boolean isPrimaryTrustCenter()
```

Returns:

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

isBackupTrustCenter

```
boolean isBackupTrustCenter()
```

Returns:

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

isPrimaryBindingTableCache

```
boolean isPrimaryBindingTableCache()
```

Returns:

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

isBackupBindingTableCache

```
boolean isBackupBindingTableCache()
```

Returns:

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

isPrimaryDiscoveryCache

```
boolean isPrimaryDiscoveryCache()
```

Returns:

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

isBackupDiscoveryCache

```
boolean isBackupDiscoveryCache()
```

Returns:

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

isNetworkManager

```
boolean isNetworkManager()
```

Returns:

true if and only if the server is a Network Manager

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.

Method Summary		Page
int	getApplicationDeviceId()	158
byte	getApplicationDeviceVersion()	158
int	getApplicationProfileId()	158
short	getEndpoint()	158
int[]	getInputClusters()	159
int[]	getOutputClusters()	159
boolean	providesInputCluster(int clusterId)	159
boolean	providesOutputCluster(int clusterId)	159

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

Utility classes modeling the ZCL data types.

See:

[Description](#)

Class Summary		Page
ZigBeeArray	This interface represents the 'Array' Data Type, as described in the ZigBee Specification	163
ZigBeeAttributeID	This interface represents the 'Attribute ID' Data Type, as described in the ZigBee Specification	165
ZigBeeBag	This interface represents the 'Bag' Data Type, as described in the ZigBee Specification	168
ZigBeeBitmap16	This interface represents the 'Bitmap 16-bits' Data Type, as described in the ZigBee Specification	170
ZigBeeBitmap24	This interface represents the 'Bitmap 24-bits' Data Type, as described in the ZigBee Specification	173
ZigBeeBitmap32	This interface represents the 'Bitmap 32-bits' Data Type, as described in the ZigBee Specification	176
ZigBeeBitmap40	This interface represents the 'Bitmap 40-bits' Data Type, as described in the ZigBee Specification	179
ZigBeeBitmap48	This interface represents the 'Bitmap 48-bits' Data Type, as described in the ZigBee Specification	182
ZigBeeBitmap56	This interface represents the 'Bitmap 56-bits' Data Type, as described in the ZigBee Specification	185
ZigBeeBitmap64	This interface represents the 'Bitmap 64-bits' Data Type, as described in the ZigBee Specification	188
ZigBeeBitmap8	This interface represents the 'Bitmap 8-bits' Data Type, as described in the ZigBee Specification	191
ZigBeeBoolean	This interface represents the 'Boolean' Data Type, as described in the ZigBee Specification	194
ZigBeeCharacterString	This interface represents the 'Character String' Data Type, as described in the ZigBee Specification	197
ZigBeeClusterID	This interface represents the 'Cluster ID' Data Type, as described in the ZigBee Specification	200
ZigBeeDate	This interface represents the 'Date' Data Type, as described in the ZigBee Specification	203
ZigBeeEnumeration16	This interface represents the 'Enumeration 16-bits' Data Type, as described in the ZigBee Specification	206
ZigBeeEnumeration8	This interface represents the 'Enumeration 8-bits' Data Type, as described in the ZigBee Specification	209
ZigBeeFloatingDouble	This interface represents the 'Floating Double' Data Type, as described in the ZigBee Specification	212
ZigBeeFloatingSemi	This interface represents the 'Floating Semi' Data Type, as described in the ZigBee Specification	215
ZigBeeFloatingSingle	This interface represents the 'Floating Single' Data Type, as described in the ZigBee Specification	218
ZigBeeGeneralData16	This interface represents the 'General Data 16-bits' Data Type, as described in the ZigBee Specification	221

<u>ZigBeeGeneralData24</u>	This interface represents the 'General Data 24-bits' Data Type, as described in the ZigBee Specification	224
<u>ZigBeeGeneralData32</u>	This interface represents the 'General Data 32-bits' Data Type, as described in the ZigBee Specification	227
<u>ZigBeeGeneralData40</u>	This interface represents the 'General Data 40-bits' Data Type, as described in the ZigBee Specification	230
<u>ZigBeeGeneralData48</u>	This interface represents the 'General Data 48-bits' Data Type, as described in the ZigBee Specification	233
<u>ZigBeeGeneralData56</u>	This interface represents the 'General Data 56-bits' Data Type, as described in the ZigBee Specification	236
<u>ZigBeeGeneralData64</u>	This interface represents the 'General Data 64-bits' Data Type, as described in the ZigBee Specification	239
<u>ZigBeeGeneralData8</u>	This interface represents the 'General Data 8-bits' Data Type, as described in the ZigBee Specification	242
<u>ZigBeeIEEEADDRESS</u>	This interface represents the 'IEEE ADDRESS' Data Type, as described in the ZigBee Specification	245
<u>ZigBeeLongCharacterString</u>	This interface represents the 'Long Character String' Data Type, as described in the ZigBee Specification	248
<u>ZigBeeLongOctetString</u>	This interface represents the 'Long Octet String' Data Type, as described in the ZigBee Specification	251
<u>ZigBeeOctetString</u>	This interface represents the 'Octet String' Data Type, as described in the ZigBee Specification	254
<u>ZigBeeSecurityKey128</u>	This interface represents the 'Security Key 128' Data Type, as described in the ZigBee Specification	257
<u>ZigBeeSet</u>	This interface represents the 'Set' Data Type, as described in the ZigBee Specification	260
<u>ZigBeeSignedInteger16</u>	This interface represents the 'Signed Integer 16-bits' Data Type, as described in the ZigBee Specification	262
<u>ZigBeeSignedInteger24</u>	This interface represents the 'Signed Integer 24-bits' Data Type, as described in the ZigBee Specification	265
<u>ZigBeeSignedInteger32</u>	This interface represents the 'Signed Integer 32-bits' Data Type, as described in the ZigBee Specification	268
<u>ZigBeeSignedInteger40</u>	This interface represents the 'Signed Integer 40-bits' Data Type, as described in the ZigBee Cluster Library Specification	271
<u>ZigBeeSignedInteger48</u>	This interface represents the 'Signed Integer 48-bits' Data Type, as described in the ZigBee Specification	274
<u>ZigBeeSignedInteger56</u>	This interface represents the 'Signed Integer 56-bits' Data Type, as described in the ZigBee Specification	277
<u>ZigBeeSignedInteger64</u>	This interface represents the 'Signed Integer 64-bits' Data Type, as described in the ZigBee Specification	280
<u>ZigBeeSignedInteger8</u>	This interface represents the 'Signed Integer 8-bits' Data Type, as described in the ZigBee Specification	283
<u>ZigBeeStructure</u>	This interface represents the 'Structure' Data Type, as described in the ZigBee Specification	286
<u>ZigBeeTimeOfDay</u>	This interface represents the 'Time Of Day' Data Type, as described in the ZigBee Specification	288
<u>ZigBeeUnsignedInteger16</u>	This interface represents the 'Unsigned Integer 16-bits' Data Type, as described in the ZigBee Specification	291
<u>ZigBeeUnsignedInteger24</u>	This interface represents the 'Unsigned Integer 24-bits' Data Type, as described in the ZigBee Specification	294

ZigBeeUnsignedInteger32	This interface represents the 'Unsigned Integer 32-bits' Data Type, as described in the ZigBee Specification	297
ZigBeeUnsignedInteger40	This interface represents the 'Unsigned Integer 40-bits' Data Type, as described in the ZigBee Specification	300
ZigBeeUnsignedInteger48	This interface represents the 'Unsigned Integer 48-bits' Data Type, as described in the ZigBee Specification	303
ZigBeeUnsignedInteger56	This interface represents the 'Unsigned Integer 56-bits' Data Type, as described in the ZigBee Specification	306
ZigBeeUnsignedInteger64	This interface represents the 'Unsigned Integer 64-bits' Data Type, as described in the ZigBee Specification	309
ZigBeeUnsignedInteger8	This interface represents the 'Unsigned Integer 8-bits' Data Type, as described in the ZigBee Specification	312
ZigBeeUTCTime	This interface represents the 'UTC Time' Data Type, as described in the ZigBee Specification	315

Package org.osgi.service.zigbee.types Description

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

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

See Also:

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

Example import for consumers using the API in this package:

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

Example import for providers implementing the API in this package:

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

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 the 'Array' Data Type, as described in the ZigBee Specification

Constructor Summary

[ZigBeeArray\(\)](#)

Page

163

Method Summary

short [getId\(\)](#)

Page

164

static [getInstance\(\)](#)

Get a Singleton instance of this class

163

Class [getJavaDataType\(\)](#)

164

String [getName\(\)](#)

163

boolean [isAnalog\(\)](#)

164

Constructor Detail

ZigBeeArray

```
public ZigBeeArray()
```

Method Detail

getInstance

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

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:
The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

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

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

getJavaDataType

```
public Class getJavaDataType()
```

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

Returns:
The corresponding Java type class.

getId

```
public short getId()
```

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

Returns:
The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

Class ZigBeeAttributeID

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeAttributeID
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Attribute ID' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeAttributeID()	165

Method Summary	Page
Object deserialize (ZigBeeDataInput is)	167
Method for deserializing a value from the passed ZigBeeDataInput stream.	
short getId ()	166
static ZigBeeAttributeID getInstance ()	165
Get a Singleton instance of this class	
Class getJavaDataType ()	166
String getName ()	166
boolean isAnalog ()	166
void serialize (ZigBeeDataOutput os, Object value)	166
Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	

Constructor Detail

ZigBeeAttributeID

```
public ZigBeeAttributeID()
```

Method Detail

getInstance

```
public static ZigBeeAttributeID getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

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 the 'Bag' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeBag()	168

Method Summary	Page
short getId()	169
static ZigBeeBag getInstance() Get a Singleton instance of this class	168
Class getJavaDataType()	169
String getName()	168
boolean isAnalog()	169

Constructor Detail

ZigBeeBag

```
public ZigBeeBag()
```

Method Detail

getInstance

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

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

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

Returns:
The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

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

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

getJavaDataType

```
public Class getJavaDataType()
```

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

Returns:
The corresponding Java type class.

getId

```
public short getId()
```

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

Returns:
The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

Class ZigBeeBitmap16

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypedescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeBitmap16
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Bitmap 16-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeBitmap16()	170

Method Summary	Page
Object deserialize (ZigBeeDataInput is)	172
Method for deserializing a value from the passed ZigBeeDataInput stream.	
short getId ()	171
static ZigBeeBitmap16 getInstance ()	170
Get a Singleton instance of this class	
Class getJavaDataType ()	171
String getName ()	171
boolean isAnalog ()	171
void serialize (ZigBeeDataOutput os, Object value)	171
Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	

Constructor Detail

ZigBeeBitmap16

```
public ZigBeeBitmap16()
```

Method Detail

[getInstance](#)

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

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

`os` - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

`value` - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeBitmap24

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypedescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeBitmap24
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Bitmap 24-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeBitmap24()	173

Method Summary	Page
Object deserialize (ZigBeeDataInput is)	175
Method for deserializing a value from the passed ZigBeeDataInput stream.	
short getId ()	174
static ZigBeeBitmap24 getInstance ()	173
Get a Singleton instance of this class	
Class getJavaDataType ()	174
String getName ()	174
boolean isAnalog ()	174
void serialize (ZigBeeDataOutput os, Object value)	174
Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	

Constructor Detail

ZigBeeBitmap24

```
public ZigBeeBitmap24()
```

Method Detail

[getInstance](#)

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

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

`os` - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

`value` - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeBitmap32

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypedescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeBitmap32
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Bitmap 32-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Pag e
ZigBeeBitmap32()	176

Method Summary	Pag e
Object deserialize (ZigBeeDataInput is)	178
Method for deserializing a value from the passed ZigBeeDataInput stream.	
short getId ()	177
static ZigBeeBitmap32 getInstance ()	176
Get a Singleton instance of this class	
Class getJavaDataType ()	177
String getName ()	177
boolean isAnalog ()	177
void serialize (ZigBeeDataOutput os, Object value)	177
Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	

Constructor Detail

ZigBeeBitmap32

```
public ZigBeeBitmap32()
```

Method Detail

[getInstance](#)

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

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeBitmap40

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypedescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeBitmap40
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Bitmap 40-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Pag e
ZigBeeBitmap40()	179

Method Summary	Pag e
Object deserialize (ZigBeeDataInput is)	181
Method for deserializing a value from the passed ZigBeeDataInput stream.	
short getId ()	180
static ZigBeeBitmap40 getInstance ()	179
Get a Singleton instance of this class	
Class getJavaDataType ()	180
String getName ()	180
boolean isAnalog ()	180
void serialize (ZigBeeDataOutput os, Object value)	180
Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	

Constructor Detail

ZigBeeBitmap40

```
public ZigBeeBitmap40()
```

Method Detail

[getInstance](#)

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

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeBitmap48

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypedescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeBitmap48
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Bitmap 48-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeBitmap48()	182

Method Summary	Page
Object deserialize (ZigBeeDataInput is)	184
Method for deserializing a value from the passed ZigBeeDataInput stream.	
short getId ()	183
static ZigBeeBitmap48 getInstance ()	182
Get a Singleton instance of this class	
Class getJavaDataType ()	183
String getName ()	183
boolean isAnalog ()	183
void serialize (ZigBeeDataOutput os, Object value)	183
Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	

Constructor Detail

ZigBeeBitmap48

```
public ZigBeeBitmap48()
```

Method Detail

[getInstance](#)

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

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeBitmap56

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypedescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeBitmap56
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Bitmap 56-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeBitmap56()	185

Method Summary	Page
Object deserialize (ZigBeeDataInput is)	187
Method for deserializing a value from the passed ZigBeeDataInput stream.	
short getId ()	186
static ZigBeeBitmap56 getInstance ()	185
Get a Singleton instance of this class	
Class getJavaDataType ()	186
String getName ()	186
boolean isAnalog ()	186
void serialize (ZigBeeDataOutput os, Object value)	186
Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	

Constructor Detail

ZigBeeBitmap56

```
public ZigBeeBitmap56()
```

Method Detail

[getInstance](#)

```
public static ZigBeeBitmap56 getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeBitmap64

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeBitmap64
```

All Implemented Interfaces:

[ZCLDataTypedescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeBitmap64
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Bitmap 64-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeBitmap64()	188

Method Summary	Page
Object deserialize (ZigBeeDataInput is)	190
Method for deserializing a value from the passed ZigBeeDataInput stream.	
short getId ()	189
static ZigBeeBitmap64 getInstance ()	188
Get a Singleton instance of this class	
Class getJavaDataType ()	189
String getName ()	189
boolean isAnalog ()	189
void serialize (ZigBeeDataOutput os, Object value)	189
Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	

Constructor Detail

ZigBeeBitmap64

```
public ZigBeeBitmap64()
```

Method Detail

[getInstance](#)

```
public static ZigBeeBitmap64 getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeBitmap8

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeBitmap8
```

All Implemented Interfaces:

[ZCLDataTypedescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeBitmap8
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Bitmap 8-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeBitmap8()	191

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	193
short	getId ()	192
static ZigBeeBitmap8	getInstance () Get a Singleton instance of this class	191
Class	getJavaDataType ()	192
String	getName ()	192
boolean	isAnalog ()	192
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	192

Constructor Detail

ZigBeeBitmap8

```
public ZigBeeBitmap8()
```

Method Detail

[getInstance](#)

```
public static ZigBeeBitmap8 getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeBoolean

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeBoolean
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Boolean' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeBoolean()	194

Method Summary	Page
Object deserialize (ZigBeeDataInput is)	196
Method for deserializing a value from the passed ZigBeeDataInput stream.	
short getId ()	195
static getInstance ()	194
Get a Singleton instance of this class	
Class getJavaDataType ()	195
String getName ()	195
boolean isAnalog ()	195
void serialize (ZigBeeDataOutput os, Object value)	195
Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	

Constructor Detail

ZigBeeBoolean

```
public ZigBeeBoolean()
```

Method Detail

[getInstance](#)

```
public static ZigBeeBoolean getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeCharacterString

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeCharacterString
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Character String' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeCharacterString ()	197

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	199
short	getId ()	198
static ZigBeeCharacterString	getInstance () Get a Singleton instance of this class	197
Class	getJavaDataType ()	198
String	getName ()	198
boolean	isAnalog ()	198
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	198

Constructor Detail

ZigBeeCharacterString

```
public ZigBeeCharacterString()
```

Method Detail

getInstance

```
public static ZigBeeCharacterString getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeClusterID

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeClusterID
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Cluster ID' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeClusterID()	200

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	202
short	getId ()	201
static ZigBeeClusterID	getInstance () Get a Singleton instance of this class	200
Class	getJavaDataType ()	201
String	getName ()	201
boolean	isAnalog ()	201
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	201

Constructor Detail

ZigBeeClusterID

```
public ZigBeeClusterID()
```

Method Detail

[getInstance](#)

```
public static ZigBeeClusterID getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeDate

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeDate
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Date' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeDate()	203

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	205
short	getId ()	204
static ZigBeeDate	getInstance () Get a Singleton instance of this class	203
Class	getJavaDataType ()	204
String	getName ()	204
boolean	isAnalog ()	204
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	204

Constructor Detail

ZigBeeDate

```
public ZigBeeDate()
```

Method Detail

getInstance

```
public static ZigBeeDate getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeEnumeration16

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeEnumeration16
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeEnumeration16
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Enumeration 16-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeEnumeration16()	206

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	208
short	getId ()	207
static ZigBeeEnumeration16	getInstance () Get a Singleton instance of this class	206
Class	getJavaDataType ()	207
String	getName ()	207
boolean	isAnalog ()	207
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	207

Constructor Detail

ZigBeeEnumeration16

```
public ZigBeeEnumeration16()
```

Method Detail

getInstance

```
public static ZigBeeEnumeration16 getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

`os` - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

`value` - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeEnumeration8

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeEnumeration8
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeEnumeration8
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Enumeration 8-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeEnumeration8()	209

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	211
short	getId ()	210
static ZigBeeEnumeration8	getInstance () Get a Singleton instance of this class	209
Class	getJavaDataType ()	210
String	getName ()	210
boolean	isAnalog ()	210
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	210

Constructor Detail

ZigBeeEnumeration8

```
public ZigBeeEnumeration8()
```

Method Detail

getInstance

```
public static ZigBeeEnumeration8 getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeFloatingDouble

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeFloatingDouble
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Floating Double' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeFloatingDouble ()	212

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	214
short	getId ()	213
static ZigBeeFloatingDouble	getInstance () Get a Singleton instance of this class	212
Class	getJavaDataType ()	213
String	getName ()	213
boolean	isAnalog ()	213
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	213

Constructor Detail

ZigBeeFloatingDouble

```
public ZigBeeFloatingDouble()
```

Method Detail

[getInstance](#)

```
public static ZigBeeFloatingDouble getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeFloatingSemi

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeFloatingSemi
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Floating Semi' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeFloatingSemi ()	215

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	217
short	getId ()	216
static ZigBeeFloatingSemi	getInstance () Get a Singleton instance of this class	215
Class	getJavaDataType ()	216
String	getName ()	216
boolean	isAnalog ()	216
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	216

Constructor Detail

ZigBeeFloatingSemi

```
public ZigBeeFloatingSemi()
```

Method Detail

[getInstance](#)

```
public static ZigBeeFloatingSemi getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeFloatingSingle

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeFloatingSingle
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Floating Single' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeFloatingSingle ()	218

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	220
short	getId ()	219
static ZigBeeFloatingSingle	getInstance () Get a Singleton instance of this class	218
Class	getJavaDataType ()	219
String	getName ()	219
boolean	isAnalog ()	219
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	219

Constructor Detail

ZigBeeFloatingSingle

```
public ZigBeeFloatingSingle()
```

Method Detail

[getInstance](#)

```
public static ZigBeeFloatingSingle getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeGeneralData16

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeGeneralData16
```

All Implemented Interfaces:

[ZCLDataTypedescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeGeneralData16
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'General Data 16-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeGeneralData16()	221

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	223
short	getId ()	222
static ZigBeeGeneralData16	getInstance () Get a Singleton instance of this class	221
Class	getJavaDataType ()	222
String	getName ()	222
boolean	isAnalog ()	222
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	222

Constructor Detail

ZigBeeGeneralData16

```
public ZigBeeGeneralData16()
```

Method Detail

getInstance

```
public static ZigBeeGeneralData16 getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeGeneralData24

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeGeneralData24
```

All Implemented Interfaces:

[ZCLDataTypedescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeGeneralData24
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'General Data 24-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeGeneralData24 ()	224

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	226
short	getId ()	225
static ZigBeeGeneralData24	getInstance () Get a Singleton instance of this class	224
Class	getJavaDataType ()	225
String	getName ()	225
boolean	isAnalog ()	225
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	225

Constructor Detail

ZigBeeGeneralData24

```
public ZigBeeGeneralData24()
```

Method Detail

getInstance

```
public static ZigBeeGeneralData24 getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeGeneralData32

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeGeneralData32
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeGeneralData32
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'General Data 32-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeGeneralData32()	227

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	229
short	getId ()	228
static ZigBeeGeneralData32	getInstance () Get a Singleton instance of this class	227
Class	getJavaDataType ()	228
String	getName ()	228
boolean	isAnalog ()	228
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	228

Constructor Detail

ZigBeeGeneralData32

```
public ZigBeeGeneralData32()
```

Method Detail

getInstance

```
public static ZigBeeGeneralData32 getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeGeneralData40

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeGeneralData40
```

All Implemented Interfaces:

[ZCLDataTypedescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeGeneralData40
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'General Data 40-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeGeneralData40()	230

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	232
short	getId ()	231
static ZigBeeGeneralData40	getInstance () Get a Singleton instance of this class	230
Class	getJavaDataType ()	231
String	getName ()	231
boolean	isAnalog ()	231
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	231

Constructor Detail

ZigBeeGeneralData40

```
public ZigBeeGeneralData40()
```

Method Detail

getInstance

```
public static ZigBeeGeneralData40 getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeGeneralData48

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeGeneralData48
```

All Implemented Interfaces:

[ZCLDataTypedescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeGeneralData48
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'General Data 48-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeGeneralData48()	233

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	235
short	getId ()	234
static ZigBeeGeneralData48	getInstance () Get a Singleton instance of this class	233
Class	getJavaDataType ()	234
String	getName ()	234
boolean	isAnalog ()	234
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	234

Constructor Detail

ZigBeeGeneralData48

```
public ZigBeeGeneralData48()
```

Method Detail

getInstance

```
public static ZigBeeGeneralData48 getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeGeneralData56

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeGeneralData56
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeGeneralData56
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'General Data 56-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeGeneralData56()	236

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	238
short	getId ()	237
static ZigBeeGeneralData56	getInstance () Get a Singleton instance of this class	236
Class	getJavaDataType ()	237
String	getName ()	237
boolean	isAnalog ()	237
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	237

Constructor Detail

ZigBeeGeneralData56

```
public ZigBeeGeneralData56()
```

Method Detail

getInstance

```
public static ZigBeeGeneralData56 getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeGeneralData64

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeGeneralData64
```

All Implemented Interfaces:

[ZCLDataTypedescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeGeneralData64
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'General Data 64-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeGeneralData64()	239

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	241
short	getId ()	240
static ZigBeeGeneralData64	getInstance () Get a Singleton instance of this class	239
Class	getJavaDataType ()	240
String	getName ()	240
boolean	isAnalog ()	240
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	240

Constructor Detail

ZigBeeGeneralData64

```
public ZigBeeGeneralData64()
```

Method Detail

getInstance

```
public static ZigBeeGeneralData64 getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeGeneralData8

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeGeneralData8
```

All Implemented Interfaces:

[ZCLDataTypedescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeGeneralData8
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'General Data 8-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeGeneralData8()	242

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	244
short	getId ()	243
static ZigBeeGeneralData8	getInstance () Get a Singleton instance of this class	242
Class	getJavaDataType ()	243
String	getName ()	243
boolean	isAnalog ()	243
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	243

Constructor Detail

ZigBeeGeneralData8

```
public ZigBeeGeneralData8()
```

Method Detail

getInstance

```
public static ZigBeeGeneralData8 getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

`os` - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

`value` - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeIEEEADDRESS

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeIEEEADDRESS
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'IEEE ADDRESS' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeIEEEADDRESS ()	245

Method Summary	Page
Object deserialize (ZigBeeDataInput is)	247
Method for deserializing a value from the passed ZigBeeDataInput stream.	
short getId ()	246
static ZigBeeIEEEADDRESS getInstance ()	245
Get a Singleton instance of this class	
Class getJavaDataType ()	246
String getName ()	246
boolean isAnalog ()	246
void serialize (ZigBeeDataOutput os, Object value)	246
Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	

Constructor Detail

ZigBeeIEEEADDRESS

```
public ZigBeeIEEEADDRESS()
```

Method Detail

getInstance

```
public static ZigBeeIEEEADDRESS getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeLongCharacterString

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeLongCharacterString
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Long Character String' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeLongCharacterString ()	248

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	250
short	getId ()	249
static ZigBeeLongCharacterString	getInstance () Get a Singleton instance of this class	248
Class	getJavaDataType ()	249
String	getName ()	249
boolean	isAnalog ()	249
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	249

Constructor Detail

ZigBeeLongCharacterString

```
public ZigBeeLongCharacterString()
```

Method Detail

getInstance

```
public static ZigBeeLongCharacterString getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

`os` - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

`value` - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeLongOctetString

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeLongOctetString
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Long Octet String' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeLongOctetString ()	251

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	253
short	getId ()	252
static ZigBeeLongOctetString	getInstance () Get a Singleton instance of this class	251
Class	getJavaDataType ()	252
String	getName ()	252
boolean	isAnalog ()	252
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	252

Constructor Detail

ZigBeeLongOctetString

```
public ZigBeeLongOctetString()
```

Method Detail

getInstance

```
public static ZigBeeLongOctetString getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeOctetString

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeOctetString
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Octet String' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeOctetString ()	254

Method Summary	Page
Object deserialize (ZigBeeDataInput is)	256
Method for deserializing a value from the passed ZigBeeDataInput stream.	
short getId ()	255
static ZigBeeOctetString getInstance ()	254
Get a Singleton instance of this class	
Class getJavaDataType ()	255
String getName ()	255
boolean isAnalog ()	255
void serialize (ZigBeeDataOutput os, Object value)	255
Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	

Constructor Detail

ZigBeeOctetString

```
public ZigBeeOctetString()
```

Method Detail

getInstance

```
public static ZigBeeOctetString getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeSecurityKey128

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSecurityKey128
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeSecurityKey128
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Security Key 128' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeSecurityKey128 ()	257

Method Summary	Page
Object deserialize (ZigBeeDataInput is)	259
Method for deserializing a value from the passed ZigBeeDataInput stream.	
short getId ()	258
static getInstance ()	257
Get a Singleton instance of this class	
Class getJavaDataType ()	258
String getName ()	258
boolean isAnalog ()	258
void serialize (ZigBeeDataOutput os, Object value)	258
Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	

Constructor Detail

ZigBeeSecurityKey128

```
public ZigBeeSecurityKey128()
```

Method Detail

getInstance

```
public static ZigBeeSecurityKey128 getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

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 the 'Set' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeSet()	260

Method Summary	Page
short getId()	261
static ZigBeeSet getInstance() Get a Singleton instance of this class	260
Class getJavaDataType()	261
String getName()	260
boolean isAnalog()	261

Constructor Detail

ZigBeeSet

```
public ZigBeeSet()
```

Method Detail

getInstance

```
public static ZigBeeSet getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

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

Returns:
The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

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

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

getJavaDataType

```
public Class getJavaDataType()
```

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

Returns:
The corresponding Java type class.

getId

```
public short getId()
```

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

Returns:
The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

Class ZigBeeSignedInteger16

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSignedInteger16
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeSignedInteger16
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Signed Integer 16-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeSignedInteger16()	262

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	264
short	getId ()	263
static ZigBeeSignedInteger16	getInstance () Get a Singleton instance of this class	262
Class	getJavaDataType ()	263
String	getName ()	263
boolean	isAnalog ()	263
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	263

Constructor Detail

ZigBeeSignedInteger16

```
public ZigBeeSignedInteger16()
```

Method Detail

getInstance

```
public static ZigBeeSignedInteger16 getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeSignedInteger24

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSignedInteger24
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeSignedInteger24
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Signed Integer 24-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeSignedInteger24 ()	265

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	267
short	getId ()	266
static ZigBeeSignedInteger24	getInstance () Get a Singleton instance of this class	265
Class	getJavaDataType ()	266
String	getName ()	266
boolean	isAnalog ()	266
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	266

Constructor Detail

ZigBeeSignedInteger24

```
public ZigBeeSignedInteger24()
```

Method Detail

getInstance

```
public static ZigBeeSignedInteger24 getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeSignedInteger32

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSignedInteger32
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeSignedInteger32
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Signed Integer 32-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeSignedInteger32()	268

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	270
short	getId ()	269
static ZigBeeSignedInteger32	getInstance () Get a Singleton instance of this class	268
Class	getJavaDataType ()	269
String	getName ()	269
boolean	isAnalog ()	269
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	269

Constructor Detail

ZigBeeSignedInteger32

```
public ZigBeeSignedInteger32()
```

Method Detail

getInstance

```
public static ZigBeeSignedInteger32 getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeSignedInteger40

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSignedInteger40
```

All Implemented Interfaces:

[ZCLDataTypedescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeSignedInteger40
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Signed Integer 40-bits' Data Type, as described in the ZigBee Cluster Library Specification

Constructor Summary	Page
ZigBeeSignedInteger40()	271

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	273
short	getId ()	272
static ZigBeeSignedInteger40	getInstance () Get a Singleton instance of this class	271
Class	getJavaDataType ()	272
String	getName ()	272
boolean	isAnalog ()	272
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	272

Constructor Detail

ZigBeeSignedInteger40

```
public ZigBeeSignedInteger40()
```

Method Detail

getInstance

```
public static ZigBeeSignedInteger40 getInstance()
```

Get a Singleton instance of this class

Returns:

the singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeSignedInteger48

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSignedInteger48
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeSignedInteger48
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Signed Integer 48-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeSignedInteger48()	274

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	276
short	getId ()	275
static ZigBeeSignedInteger48	getInstance () Get a Singleton instance of this class	274
Class	getJavaDataType ()	275
String	getName ()	275
boolean	isAnalog ()	275
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	275

Constructor Detail

ZigBeeSignedInteger48

```
public ZigBeeSignedInteger48()
```

Method Detail

getInstance

```
public static ZigBeeSignedInteger48 getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeSignedInteger56

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSignedInteger56
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeSignedInteger56
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Signed Integer 56-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeSignedInteger56()	277

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	279
short	getId ()	278
static ZigBeeSignedInteger56	getInstance () Get a Singleton instance of this class	277
Class	getJavaDataType ()	278
String	getName ()	278
boolean	isAnalog ()	278
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	278

Constructor Detail

ZigBeeSignedInteger56

```
public ZigBeeSignedInteger56()
```

Method Detail

getInstance

```
public static ZigBeeSignedInteger56 getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeSignedInteger64

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSignedInteger64
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeSignedInteger64
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Signed Integer 64-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeSignedInteger64()	280

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	282
short	getId ()	281
static ZigBeeSignedInteger64	getInstance () Get a Singleton instance of this class	280
Class	getJavaDataType ()	281
String	getName ()	281
boolean	isAnalog ()	281
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	281

Constructor Detail

ZigBeeSignedInteger64

```
public ZigBeeSignedInteger64()
```

Method Detail

getInstance

```
public static ZigBeeSignedInteger64 getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeSignedInteger8

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeSignedInteger8
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeSignedInteger8
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Signed Integer 8-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeSignedInteger8()	283

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	285
short	getId ()	284
static ZigBeeSignedInteger8	getInstance () Get a Singleton instance of this class	283
Class	getJavaDataType ()	284
String	getName ()	284
boolean	isAnalog ()	284
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	284

Constructor Detail

ZigBeeSignedInteger8

```
public ZigBeeSignedInteger8()
```

Method Detail

getInstance

```
public static ZigBeeSignedInteger8 getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

`os` - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

`value` - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

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 the 'Structure' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeStructure ()	286

Method Summary	Page
short getId ()	287
static ZigBeeStructure getInstance () Get a Singleton instance of this class	286
Class getJavaDataType ()	287
String getName ()	286
boolean isAnalog ()	287

Constructor Detail

ZigBeeStructure

```
public ZigBeeStructure()
```

Method Detail

getInstance

```
public static ZigBeeStructure getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

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

Returns:
The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

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

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

getJavaDataType

```
public Class getJavaDataType()
```

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

Returns:
The corresponding Java type class.

getId

```
public short getId()
```

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

Returns:
The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

Class ZigBeeTimeOfDay

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeTimeOfDay
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Time Of Day' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeTimeOfDay()	288

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	290
short	getId ()	289
static ZigBeeTimeOfDay	getInstance () Get a Singleton instance of this class	288
Class	getJavaDataType ()	289
String	getName ()	289
boolean	isAnalog ()	289
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	289

Constructor Detail

ZigBeeTimeOfDay

```
public ZigBeeTimeOfDay()
```

Method Detail

getInstance

```
public static ZigBeeTimeOfDay getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

`os` - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

`value` - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeUnsignedInteger16

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeUnsignedInteger16
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeUnsignedInteger16
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Unsigned Integer 16-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeUnsignedInteger16()	291

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	293
short	getId ()	292
static ZigBeeUnsignedInteger16	getInstance () Get a Singleton instance of this class	291
Class	getJavaDataType ()	292
String	getName ()	292
boolean	isAnalog ()	292
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	292

Constructor Detail

ZigBeeUnsignedInteger16

```
public ZigBeeUnsignedInteger16()
```

Method Detail

getInstance

```
public static ZigBeeUnsignedInteger16 getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeUnsignedInteger24

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeUnsignedInteger24
```

All Implemented Interfaces:

[ZCLDataTypedescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeUnsignedInteger24
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Unsigned Integer 24-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeUnsignedInteger24()	294

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	296
short	getId ()	295
static ZigBeeUnsignedInteger24	getInstance () Get a Singleton instance of this class	294
Class	getJavaDataType ()	295
String	getName ()	295
boolean	isAnalog ()	295
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	295

Constructor Detail

ZigBeeUnsignedInteger24

```
public ZigBeeUnsignedInteger24()
```

Method Detail

getInstance

```
public static ZigBeeUnsignedInteger24 getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeUnsignedInteger32

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeUnsignedInteger32
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeUnsignedInteger32
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Unsigned Integer 32-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeUnsignedInteger32()	297

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	299
short	getId ()	298
static ZigBeeUnsignedInteger32	getInstance () Get a Singleton instance of this class	297
Class	getJavaDataType ()	298
String	getName ()	298
boolean	isAnalog ()	298
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	298

Constructor Detail

ZigBeeUnsignedInteger32

```
public ZigBeeUnsignedInteger32()
```

Method Detail

getInstance

```
public static ZigBeeUnsignedInteger32 getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeUnsignedInteger40

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeUnsignedInteger40
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeUnsignedInteger40
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Unsigned Integer 40-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeUnsignedInteger40()	300

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	302
short	getId ()	301
static ZigBeeUnsignedInteger40	getInstance () Get a Singleton instance of this class	300
Class	getJavaDataType ()	301
String	getName ()	301
boolean	isAnalog ()	301
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	301

Constructor Detail

ZigBeeUnsignedInteger40

```
public ZigBeeUnsignedInteger40()
```

Method Detail

getInstance

```
public static ZigBeeUnsignedInteger40 getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeUnsignedInteger48

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeUnsignedInteger48
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeUnsignedInteger48
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Unsigned Integer 48-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeUnsignedInteger48()	303

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	305
short	getId ()	304
static ZigBeeUnsignedInteger48	getInstance () Get a Singleton instance of this class	303
Class	getJavaDataType ()	304
String	getName ()	304
boolean	isAnalog ()	304
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	304

Constructor Detail

ZigBeeUnsignedInteger48

```
public ZigBeeUnsignedInteger48()
```

Method Detail

getInstance

```
public static ZigBeeUnsignedInteger48 getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeUnsignedInteger56

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeUnsignedInteger56
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeUnsignedInteger56
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Unsigned Integer 56-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeUnsignedInteger56()	306

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	308
short	getId ()	307
static ZigBeeUnsignedInteger56	getInstance () Get a Singleton instance of this class	306
Class	getJavaDataType ()	307
String	getName ()	307
boolean	isAnalog ()	307
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	307

Constructor Detail

ZigBeeUnsignedInteger56

```
public ZigBeeUnsignedInteger56()
```

Method Detail

getInstance

```
public static ZigBeeUnsignedInteger56 getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

`os` - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

`value` - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeUnsignedInteger64

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeUnsignedInteger64
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeUnsignedInteger64
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Unsigned Integer 64-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeUnsignedInteger64()	309

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	311
short	getId ()	310
static ZigBeeUnsignedInteger64	getInstance () Get a Singleton instance of this class	309
Class	getJavaDataType ()	310
String	getName ()	310
boolean	isAnalog ()	310
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	310

Constructor Detail

ZigBeeUnsignedInteger64

```
public ZigBeeUnsignedInteger64()
```

Method Detail

getInstance

```
public static ZigBeeUnsignedInteger64 getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeUnsignedInteger8

org.osgi.service.zigbee.types

```
java.lang.Object
└─ org.osgi.service.zigbee.types.ZigBeeUnsignedInteger8
```

All Implemented Interfaces:

[ZCLDataTypeDescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeUnsignedInteger8
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'Unsigned Integer 8-bits' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeUnsignedInteger8()	312

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	314
short	getId ()	313
static ZigBeeUnsignedInteger8	getInstance () Get a Singleton instance of this class	312
Class	getJavaDataType ()	313
String	getName ()	313
boolean	isAnalog ()	313
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	313

Constructor Detail

ZigBeeUnsignedInteger8

```
public ZigBeeUnsignedInteger8()
```

Method Detail

getInstance

```
public static ZigBeeUnsignedInteger8 getInstance()
```

Get a Singleton instance of this class

Returns:
the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

Class ZigBeeUTCTime

org.osgi.service.zigbee.types

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

All Implemented Interfaces:

[ZCLDataTypedescription](#), [ZCLSimpleTypeDescription](#)

```
public class ZigBeeUTCTime
extends Object
implements ZCLSimpleTypeDescription
```

This interface represents the 'UTC Time' Data Type, as described in the ZigBee Specification

Constructor Summary	Page
ZigBeeUTCTime ()	315

Method Summary		Page
Object	deserialize (ZigBeeDataInput is) Method for deserializing a value from the passed ZigBeeDataInput stream.	317
short	getId ()	316
static ZigBeeUTCTime	getInstance () Get a Singleton instance of this class	315
Class	getJavaDataType ()	316
String	getName ()	316
boolean	isAnalog ()	316
void	serialize (ZigBeeDataOutput os, Object value) Method for serializing a ZigBee data type into a ZigBeeDataOutput stream.	316

Constructor Detail

ZigBeeUTCTime

```
public ZigBeeUTCTime()
```

Method Detail

[getInstance](#)

```
public static ZigBeeUTCTime getInstance()
```

Get a Singleton instance of this class

Returns:

the Singleton instance

getName

```
public String getName()
```

Specified by:

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

Returns:

The associated data type name string.

isAnalog

```
public boolean isAnalog()
```

Specified by:

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

Returns:

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

getJavaDataType

```
public Class getJavaDataType()
```

Specified by:

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

Returns:

The corresponding Java type class.

getId

```
public short getId()
```

Specified by:

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

Returns:

The data type identifier. The currently supported data types ids are in file [ZigBeeDataTypes](#)

serialize

```
public void serialize(ZigBeeDataOutput os,  
                     Object value)
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

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

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

Specified by:

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

Parameters:

os - a [ZigBeeDataOutput](#) stream where to the passed value will be appended. This parameter cannot be null. If null a `NullPointerException` must be thrown.

value - The value that have to be serialized on the output stream. If null is passed this method outputs on the stream the ZigBee invalid value related the specific data type. If the

data type do not allow any invalid value and the passed value is null an `IllegalArgumentException` is thrown.

deserialize

```
public Object deserialize(ZigBeeDataInput is)
    throws IOException
```

Description copied from interface: [ZCLSimpleTypeDescription](#)

Method for deserializing a value from the passed [ZigBeeDataInput](#) stream.

Specified by:

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

Parameters:

`is` - the [ZigBeeDataInput](#) from where the value of data type is read from.

Returns:

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

Throws:

`IOException` - if an I/O error occurs while reading the `ZigBeeDataInput`

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 specification, Document 053474r17, October 19, 2007.
- [2]. ZigBee Alliance, ZigBee Cluster Library specification, Document 075123r04ZB, May 29, 2012.
- [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	Nicolas Lingois
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