KIC – EIT ICT Labs
CALL FOR SME – Nov. 5th 2012

Java Gateway Abstraction Layer for ZigBee Pro and ZigBee IP networks

Contents

1 Introduction

This document is the call for the software implementation of the ZigBee Gateway Device.

Chapter 2 of the document reports all those non-technical aspects of the call, including eligibility criteria, expected timeline, traveling, IPR issues, and how to answer to the call.

Only SME (small and medium-sized enterprises) companies, as defined in the EU law by the European Commission document [5], are eligible to answer this call. Answers to the call are requested to specify at a minimum the workplan for the software development and to provide evidence of expertise of the contractor with the software activities that are subject of this call, in particular with the ZigBee specifications and with the OSGi specifications.

Chapter 3, instead, reports all the technical aspects that describe the requested software module, including a description of the context for the work and the end-to-end architecture where the software module will be integrated. The document does not cover generic information like scope or purpose of a ZigBee Gateway Device, since this is public information available in public ZigBee documents [2], but it rather focuses on the mandatory requested functionalities and requirements. Further to the mandatory requirements already described in the ZigBee Alliance documentation, this call requests some extra functionalities and implementation requirements, based upon the experience of Telecom Italia with the implementation of the GAL ZPRO (Gateway Abstraction Layer), a ZigBee PRO Gateway Device implemented by Telecom Italia in C and C++ code. Access to the source code of this software will be given to the selected contractor under a proper license and for the only purposes of executing this contract. By extremely simplifying, the requested activity is the porting of this C/C++ implementation into Java language, plus packaging as on OSGi bundle plus additional development of support for the ZigBee-IP protocol.

In the rest of the document the following convention is adopted to differentiate between mandatory and preferred requirements:

shall	A key word indicating a mandatory requirement. Solutions are *required* to meet all such mandatory requirements.
should	A key word indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase *is recommended*.

2 Non-technical aspects

2.1 Eligibility criteria

Only SME (small and medium-sized enterprises) companies, as defined in the EU law by the European Commission document [5], are eligible to participate to this call.

Under the condition that the activity subject to this call will be included in the EIT ICT Labs Business Plan 2013.

2.2 Timeline

All answers shall comply with the following timeline:

- Nov. 12th, 2012: publication of this call on the KIC EIT web site
- Nov. 19th, 2012: opening of question time. At this time questions about the call will be accepted, all questions and provided answers will be posted on this public web page

http://www.eitictlabs.eu/ict-labs/tenders-and-calls/ available to all participants. Any questions and/or clarifications shall be posted only via e-mail to this address eit-call@avalon.tilab.com. Requests submitted by different means will not be considered.

- Dec. 2nd, 2012: closure of the question time: no more questions will be accepted.
- Dec. 9th, 2012: closure of the call, all answers collected (4 weeks after the call publication). The deadline to submit answers to the call is 24:00 of Dec. 9th, 2012. Answers shall be sent via e-mail to this address **eit-call@avalon.tilab.com** Any variation of this deadline will be promptly communicated via this public web page http://www.eitictlabs.eu/ict-labs/tenders-and-calls/
- Jan. 9th, 2013: contractor selected
- Jan. 19th, 2013: selected contractor posted on the web site
- 21st January 2013: start of the activities (T0 below)
- T0 + 10: kick-off
- T0 + 60: test list of the software module finalized
- T0 + 180: first release of the software released
- T0 + 210: testing of Telecom Italia and comments sent to the contractor (30 days after the first release)
- T0 + 270: final release of the software released
- December 2013: end of the activities. At this time all documentation and software have to be completed, tested and approved.

2.3 Budget

The budget of the call is 30.000 Euro and the funding is 100% on a cost basis.

The contractor will be allowed to claim this funding up to the maximum of 30.000 Euro on basis of cost claim, under the condition that the activity subject to this call will be included in the EIT ICT Labs Business Plan 2013.

The participation to this call will be at the Company's complete risk and cost; thus the Company is not entitled to request Telecom Italia and/or KIC EIT for any reimbursement connected to its participation at the process.

2.4 Traveling and logistics

It is expected that most work will be done by electronic means (e-mail, telephone) but at least two travels to Torino, where the testing and validation lab is hosted, will be needed by the contractor:

1) kick-off of the activities including the in-detail presentation of the existing software and the target software architecture
2) testing and validation

All travel expenses are considered already included into the budget and extra travelling will not give right to additional cost reimbursements.

2.5 IPR issues

All the IPR of the implemented software shall be owned by Telecom Italia and shall be released to the KIC EIT ICT Labs and to Telecom Italia.

The contractor shall provide warranty that the implemented software is free of constraints and/or property rights of third parties.

Telecom Italia is going to provide to the selected contractor a software implementation of the ZigBee Gateway Device for ZigBee Pro stack. The software will be provided in source code and as-is, it is partly written in C language, partly in C++ language and partly in Java language. It is provided as a reference for the requested implementation under a proper license and for the only purposes of executing this contract.

2.6 How to answer to this call

All answers shall be submitted in written form in English language sent by e-mail to the following address:

eit-call@avalon.tilab.com

Only answers arriving before the deadline are considered valid. Any variation of this deadline will be promptly communicated via the same web site where the call was published.

Answers shall include at least the following information:

- Confirmation that the respondent is an SME as defined as defined in the EU law by the European Commission document [5]
- Workplan for the software development
- Software development Methodology
- Eventual additional functionalities that will be implemented even if not described in this call
- Information with evidence of the expertise in ZigBee domain, if any
- Information with evidence of the expertise in OSGi domain, if any
- CV of the relevant people that will be involved in the software development
- Contact point for the scope of the call
- Any other information that helps to qualify the expertise of the respondent

All submitted documentation will remain at Telecom Italia disposal for any internal elaboration and/or internal activities.

2.7 Evaluation Criteria

An advisory team will evaluate all received answers. The team is composed of:

- The Smart Energy Systems Action Line Leader
- One expert nominated by the Trento Node of the KIC EIT ICT Lab
- One expert internal to the the SecSES-EU Activity nominated by Telecom Italia

The winner of the call (in the following also referred as the **contractor**) will be selected on the basis of the following evaluation criteria:

- expertise in the field of ZigBee and OSGi
- quality of the workplan
- quality of software development methodology
- eventual additional functionalities proposed by the contractor

Each answer will receive a score and a consensus meeting will select the best answer.

We reserve the right to not consider Answers which are not consistent with the guidelines herein provided and/or are presented after the defined deadlines. This call does not bind Telecom Italia and the KIC EIT ICT Lab to carry on the evaluation process and to proceed with the award of any supply and we reserve the right to cancel at any moment the evaluation process.

3 Technical Aspects

3.1 General Architecture

The call fits within the scope of the SecSES-EU Activity Work Plan of the Action Line Smart Energy Systems of the KIC EIT. In particular, it fulfills some of the requirements of the goal of implementing a Secure Smart Home Energy Gateway for Smart Buildings.

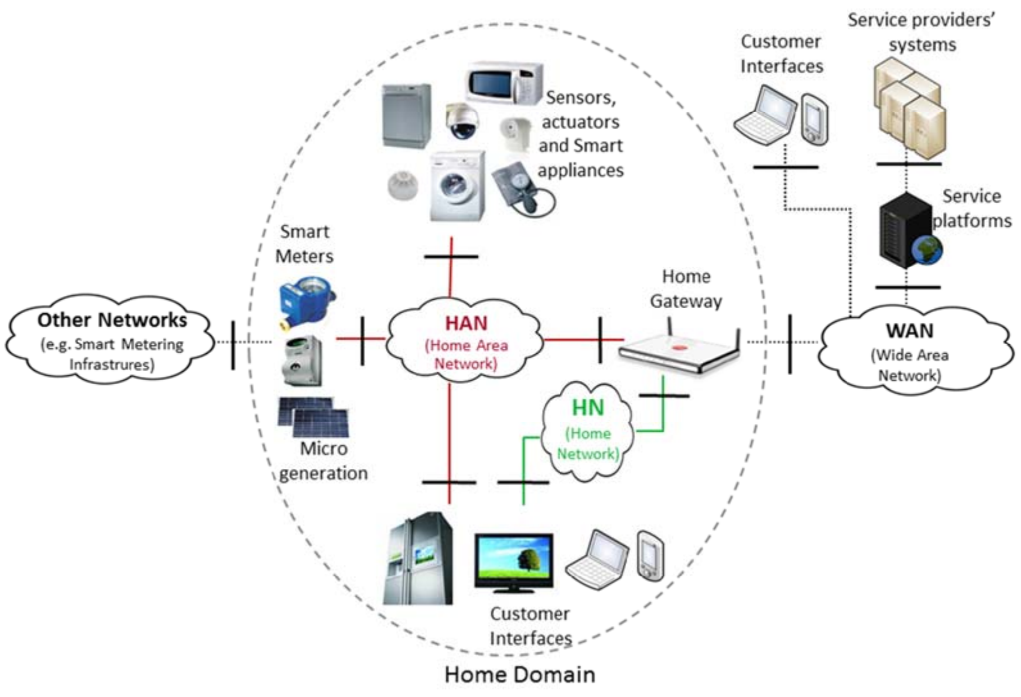

Figure 1 - End-to-end system architecture

The activity will implement security and privacy related features in a prototype of an Energy Box that is physically embedded into the Home Broadband Gateway at the users' premises. The gateway will support a variety of networking protocols, such as ZigBeePro, ZigBee-IP, and WiFi. The figure below illustrates an expected scenario, including smart meters, photovoltaic plants for micro-generation, electrical plants, smart plugs, smart appliances and different types of sensors and actuators, and of course customer interfaces. It also shows the communication with the service platforms in the cloud through a Wide Area Network. As shown in the figure, the system is complex and consists of many interacting components. The figure represents the user's Home Domain that includes both the Home Area Network (HAN) and the Home Network. In this domain all the home devices (i.e. Smart Appliances, Smart Plugs, Home Gateway, Smart Info and Customer Interfaces) will form a local network coordinated by the Home Gateway.

The Home Broadband Gateway will provide multiple network interfaces and it will host proper communication software to enable home devices to connect to the network. For the scope of this call, ZigBee Pro and ZigBee IP stacks will be provided by the gateway through two different commercial USB keys. The subject of this call is the implementation of the ZigBee Network Device software that will be hosted on the Home Broadband Gateway. **In the following this software will be generally referred with the term GAL (Gateway Abstraction Layer).**

The general architecture of the GAL is shown in the figure below: **it shall provide an abstraction to the application developers in order to deal with both ZigBee IP networks and ZigBee PRO networks**. The ZigBee PRO stack and ZigBee IP stack will run on a dedicated network processor provided as a module (microcontroller and IEEE 802.15.4 radio) or as an external USB dongle. The Java software module to be developed is indicated as GAL and it shall run on a host processor.

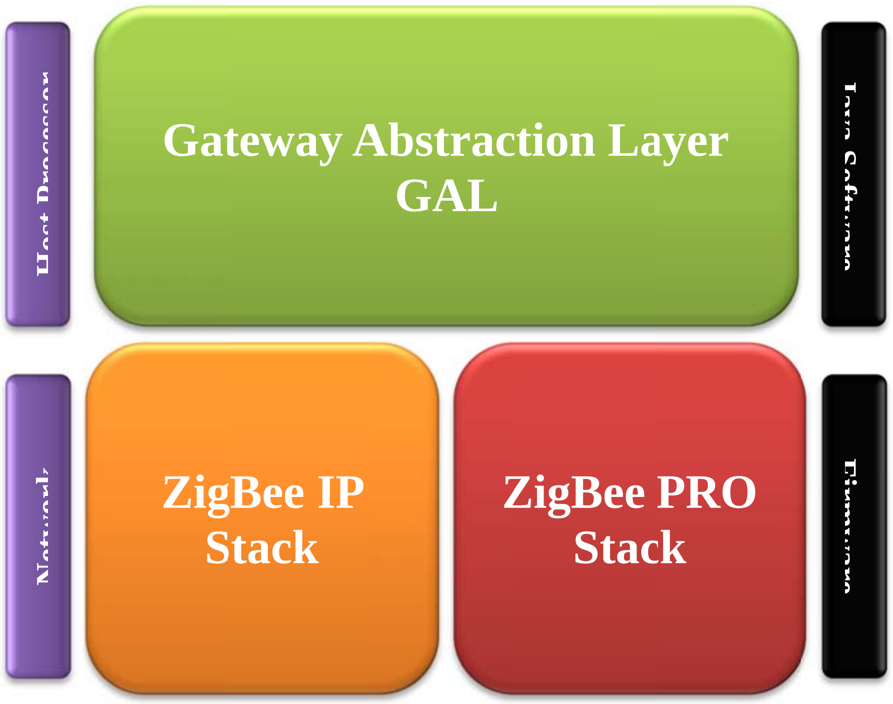

Figure 2 –General architecture

3.2 General Software Requirements

[R1] All the software shall be implemented in Java language, JDK 1.5

[R2] All the software shall be packaged as a Java library and it shall expose its services to enable the development of Java applications using ZigBee IP and/or ZigBee Pro networks and devices

[R3] All the software shall also be packaged as OSGi bundles by using the standard functionalities of OSGi R4.x core framework specifications, e.g. by using OSGi open source Equinox framework (see http://www.eclipse.org/equinox/)

[R4] Dependency of the delivered software on third-party libraries should be avoided or kept at a minimum. Any dependency shall be explicitly agreed between the contractor and the contractee.

[R5] The GAL software shall expose the Java jGAL interface reported in the Annex 1 of this document by respecting syntax and semantics. This interface will also be released to the contractor in source code in order to simplify the software development activity. Extensions to this interface are allowed to implement new functionalities, while modifications should be kept at a minimum and – any way – they shall be agreed upon with the contractee.

[R6] The GAL software shall be designed and structured to be independent of the underlying used USB key, even if the implementation should be provided and tested only for a single USB key. All the modules of the software which depend on the specific version or vendor of the underlying USB stack shall be packaged into an ad-hoc separated software module. This separated software module shall implement an harmonized well-defined Java interface that separates the USB-dependent software from the USB-independent software.

[R7] Memory requirement, footprint and CPU usage should be minimized.

[R8] All the software shall be tested and delivered in source code including javadoc documentation.

[R9] Test units should be implemented and provided as part of the contract.

3.3 ZigBee PRO requirements

3.3.1 Mandatory Requirements from the ZigBee Standard Technical Specifications

Unless it is specified in the following sections, the main reference guide to implement a ZigBee Gateway Device is the “Network Device Gateway Specification Document”, public available at the ZigBee web site (www.zigbee.org) [1]. For those one who are beginner with the ZigBee protocol and with the ZigBee Network Device working group (NDG), a good reference to read before to start is the document number 095465r12, “Understanding ZigBee Gateway”, public available as well.

In Figure 3 it is shown a generic ZigBee Gateway deployment, introduced at this stage only to point out at the following major requirements:

[R10] The GAL software shall implement the REST interface as defined into the ZigBee Gateway Device public specifications [1] and as reported in the detail in the following paragraphs.

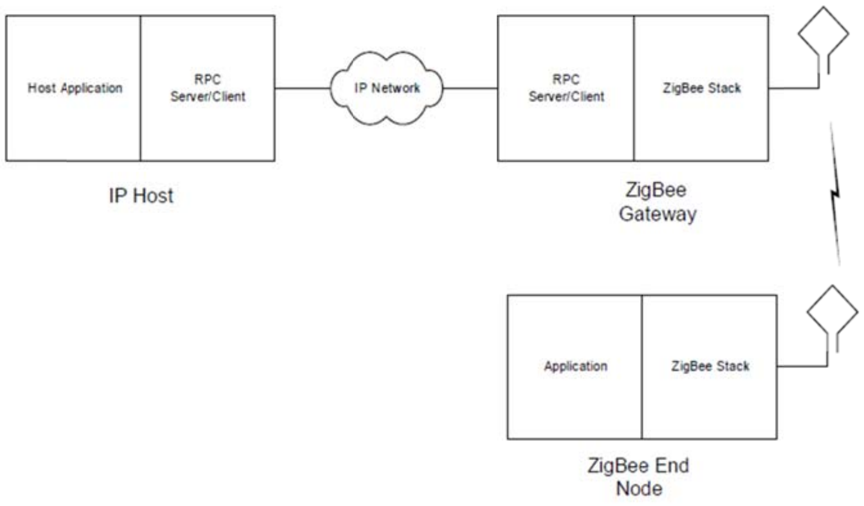

Figure 3 – Generic ZigBee Gateway deployment

Following what has been specified in the ZigBee Gateway Device document ([1]), this section report all the mandatory features that need to be implemented. Clearly, all the mandatory features need to be implemented in order to certify the device as ZigBee standard device. Regarding instead the optional feature, Telecom Italia selected some of them for its internal applications and those ones needs to be implemented as well: a complete and detailed list can be found in the PICS document for Gateway device, filled by Telecom Italia to certify a ZigBee Gateway Device (i.e. the “PICS_TI_WINDOWS-FREESCALE.doc” file).

[R11] The GAL shall implement the following ZGD features, as described on chapter 5 of [1] and as, for convenience, reported in Table 1

RequestIdentifier
CallbackDestination
Timeout
Status
Callback Handlers

Table 1 – General communications supported

[R12] The GAL shall implement all the mandatory Gateway Management Objects (GMO) procedures that are listed in Table 6.1 of [1] and that, for convenience, also reported below in Table 2.

GetVersion
Get procedure
CreateCallback
DeleteCallback
ListCallbacks
StartGatewayDevice
StartGatewayDeviceEvent
ListAddresses
ConfigureEndpoint
ClearEndpoint
SendAPSMessage
NofitySendAPSMessage
Leave
LeaveEvent
PermitJoin
PermitJoinEvent

Table 2 - GMO Functions

[R13] The GAL shall implement the ZigBee Device Profile (ZDP) commands and ZigBee Cluster Library (ZCL) commands and handlers, as defined in Section 6.5 and 6.6 of [1] and, for convenience, here reported in Table 3.

SendZDPCommand
NotifyZDPEvent
SendZCLCommand
NotifyZCLEvent

Table 3 – ZDP and ZCL procedures

[R14] The GAL shall implement the following ZigBee Application Support (APS) Sub-Layer procedures and callbacks, as defined in Section 6.7 of [1] and, for convenience, here reported in Table 4.

ConfigureEndpoint
ClearEndpoint
SendAPSMessage
NofitySendAPSMessage

Table 4 – ZigBee Application Support Sub-Layer functions

3.3.2 Other requirements

This section reports an additional list of requirements that are not defined as mandatory in the ZigBee specifications.

[R15] The GAL shall be based upon a ZigBee Pro stack implemented into a USB key.

[R16] The expected commissioning practices according to the specific use cases enabled by the NDG (Network Device Group) recommend that the ZGD shall be able to form and join any network using common security model (TC link key). In order to maintain compatibility with ZigBee profile interoperability ([3]), default TC link key need to be used to transmit the network key.

[R17] The GAL shall maintain a list of TC link keys derived by install codes of the devices in the network (see Smart Energy Profile 1.x).

[R18] The GAL shall be able to securely join an existing network and act as a ZigBee Router.

[R19] The GAL shall support packet fragmentation both in transmission and reception phases. Such a functionality shall be made transparent to the application using the GAL. When transmitting a large message, the GAL shall split the message into proper fragments adequate to the transmission. At the same way, when receiving a content split into several fragments, the GAL shall reorder and package the content as a single large message.

[R20] The GAL shall implement the following list of API commands as reported in the table where the last column “Standard” has the following meaning: “Yes” means that the command is part of the ZigBee standard and thus it shall be implemented; “No” means that it is a Telecom Italia extension and thus it is left as optional. The only two APIs that are required to be implemented are the "setGatewayEventListener" (this could be replaced with the implementation of the NodeDiscoveryEvent and StartGatewayDeviceEvent procedures, respectively described in paragraph 6.4.11 and 6.4.20 of [1]), and the "resetDongle" (required to allow the client to reset the gateway, which is the equivalent of going back to an initial state). Notice that the syntax of the API used in the table is purely indicative (please refer to Annex 1 to a proper Java syntax).

API command	**Description**	**Standard**
Void setGatewayEventListener(GatewayEventListener listener)	Function used to set a listener	NO
Version getVersion() throws IOException, Exception, GatewayException	See [1], paragraph 6.4.1	Yes
String getInfoBaseAttribute(short attrId) throws Exception, Exception, GatewayException	See [1], paragraph 6.4.2	Yes
void setInfoBaseAttribute(short attrId, String value) throws IOException, Exception, GatewayException	See [1], paragraph 6.4.3	Yes
long createCallback(Callback callback, APSMessageListener	See [1], paragraph 6.4.4	Yes

listener) throws IOException, Exception, GatewayException		
long createAPSCallback(short endpoint, APSMessageListener listener) throws IOException, Exception, GatewayException	This is a short version of the CreateCallback described in [1], paragraph 6.4.4	NO
long createAPSCallback(APSMessageListener listener) throws IOException, Exception, GatewayException	This is a short version of the CreateCallback described in [1], paragraph 6.4.4	NO
List<Long> listCallbacks() throws IOException, Exception, GatewayException	See [1], paragraph 6.4.7	Yes
void deleteCallback(long callId) throws IOException, Exception, GatewayException	See [1], paragraph 6.4.6	Yes
Aliases listAddresses() throws IOException, Exception, GatewayException	See [1], paragraph 6.4.25	Yes
void configureStartupAttributeSet(StartupAttributeInfo sai) throws IOException, Exception, GatewayException	See [1], paragraph 6.4.21	Yes
StartupAttributeInfo readStartupAttributeSet(short index) throws IOException, Exception, GatewayException	See [1], paragraph 6.4.22	Yes
void startGatewayDevice(long timeout, StartupAttributeInfo sai) throws IOException, Exception, GatewayException	See [1], paragraph 6.4.19	Yes
void startGatewayDevice(long timeout) throws IOException, Exception, GatewayException	Similar to the previous API (described in [1], paragraph 6.4.19) but without taking any input since it uses the config.ini settings.	NO
WSNNodeList readNodeCache() throws IOException, Exception, GatewayException	See [1], paragraph 6.4.13	Yes
void startNodeDiscovery(long timeout, int discoveryMask) throws IOException, Exception, GatewayException	See [1], paragraph 6.4.10	Yes
void subscribeNodeRemoval(long timeout, int discoveryMask) throws IOException, Exception, GatewayException	See [1], paragraph 6.4.12 and paragraph 6.4.27	Yes
NodeServices getLocalServices() throws IOException, Exception, GatewayException	Similar to the API described in [1], paragraph 6.4.18, but acting on the local node	NO
NodeServicesList readServicesCache() throws IOException, Exception, GatewayException	See [1], paragraph 6.4.18	Yes
void startServiceDiscovery(long timeout, Address addrOfInterest) throws IOException, Exception, GatewayException	See [1], paragraph 6.4.14	Yes
void getServiceDescriptor(long timeout, Address addrOfInterest, short endpoint) throws IOException, Exception, GatewayException	See [1], paragraph 6.4.16	Yes
void getNodeDescriptor(long timeout, Address addrOfInterest) throws IOException, Exception, GatewayException	See [1], paragraph 6.7.2	Yes
void sendZDPCommand(long timeout, ZDPCommand command) throws IOException, Exception, GatewayException	See [1], paragraph 6.5.1	Yes
short configureEndpoint(long timeout, SimpleDescriptor desc) throws IOException, Exception, GatewayException	See [1], paragraph 6.7.6	Yes
void clearEndpoint(short endpoint) throws IOException, Exception, GatewayException	See [1], paragraph 6.7.7	Yes
void leaveAll() throws IOException, Exception,	Similar to the API described	NO

GatewayException	in [1], paragraph 6.4.26, but with setting forcing to leave the network to all the nodes	🟥
void leave(long timeout, Address addrOfInterest) throws IOException, Exception, GatewayException	Similar to the API described in [1], paragraph 6.4.26, but with standard mask settings	NO
void leave(long timeout, Address addrOfInterest, int mask) throws IOException, Exception, GatewayException	See [1], paragraph 6.4.26	Yes
void permitJoinAll(long timeout, short duration) throws IOException, Exception, GatewayException	Similar to the API described in [1], paragraph 6.4.28, but with destination address set to 0xFFFF	NO
void permitJoin(long timeout, Address addrOfInterest, short duration) throws IOException, Exception, GatewayException	See [1], paragraph 6.4.28	Yes
void sendAPSMessage(APSMessage message) throws IOException, Exception, GatewayException	Similar to the API described in [1], paragraph 6.7.8, but in blocking mode	NO
void sendAPSMessage(long timeout, APSMessage message) throws IOException, Exception, GatewayException	See [1], paragraph 6.7.8	Yes
void sendZCLMessage(long timeout, ZCLMessage message) throws IOException, Exception, GatewayException	See [1], paragraph 6.6.1	Yes
void sendZDPCommand(long timeout, ZDPMessage message) throws IOException, Exception, GatewayException	See [1], paragraph 6.5.1	Yes
void resetDongle(long timeout, short mode) throws IOException, Exception, GatewayException	Force the gateway to restore all its default setting and come back to an initial state	NO

Table 5 – GAL APIs

[R21] Extra APIs from those one described in Table 5 shall be implemented:

- A GET to the resource "[root]/allwsnnodes/lqi" for the LQI monitoring (few more details can be found in the GAL's code when a NdA will be agreed.
- addBinding, removeBinding and getNodeBindings, three important APIs to respectively set, unset and get a list of Bindings. Please see Annex 1 for more details.
- A GET/POST to a dedicated REST resource to force the network to move to a different channel (FrequencyAgility). It is currently implemented as a GET on the resource "[root]/localnode/frequencyagility?timeout=0xfffff&scanDuration=10" but it shall became a POST and the new path shall be agreed with Telecom Italia

[R22] The GAL shall implement the discovery algorithm and the Freshness/ForcePing algorithm as described in Annex 2, where more detailed information are provided.

[R23] The GAL shall read a configuration file with the list of variables described in the following Table 6. Part of those settings can be reviewed in agreement with Telecom Italia.

Variable	**Description**	**Implementation status**
Platform	Hardware platform desired: integration (0), zsdio/EZSP/Freescale/MicroSD (1)	Since the gateway shall support different stack vendors, a variable is probably needed to do some specific calls due to the different internal architectures. Clearly this is a development-

		specific variable so it could be not inserted.
autostart	Autostart the ZigBee network: 0 to disable (it will wait for a StartGatewayDevice command), 1 to enabled	This in an important variable useful to automatically call the StartGatewayDevice procedure using the default parameters stored in the configuration file, enabling the gateway to startup without wait any external interaction.
startupMode	Three possible scenario to startup the network: #0 - StartupSetMode=0x18 (CommissioningMode) AND StartupControlMode=0x00 (Association) --> To be used the first time, when the network is created #1 - StartupSetMode=0x00 (use NVM) AND StartupControlMode=0x04 (SilentStart) --> To be used at running time #2 - StartupSetMode=0x18 (CommissioningMode) AND StartupControlMode=0x04 (SilentStart) --> To be used for TC Replacement	Depending of the stack vendor that will be supported, specific startup settings could be necessary: the user/administrator of the gateway shall have the possibility to decide if startup the network using a brand new ZigBee setting (commissioning mode) or if recycle an existing setting. This second part could differs from the current GAL and thus it shall be considered with attention: all the main ZigBee setting like channel radio, extendedPANId, PANId, DeviceType, etc… shall we stored in a configuration file together with an image file of the current status of the stack vendor's memory. It shall thus be possible to reused any time a configuration before a crash, using a Not-Volatile-Memory (NVM) if it is supported by the stack, or by "flashing" those setting in a way that the node does not start a new network.
localRegToDAnnce	Enable a local registration to device announcement events (1 to enable, 0 to disable)	The new gateway shall have this requirements always enabled so this variable can be eliminated. The target here is to be sure that any time a new node joins the network, an announcement event shall be generated to inform the upper level about this event.
keepAliveThreshold	Specify how long to wait (in seconds) before to start polling each node to verify if they're still alive ('0' to disable)	Please refer to paragraph **Error! Reference source not found.**.
keepAliveNumberOfAttempt	Once keepAliveThreshold is reached, GAL tries to ping the node for 'keepAliveNumberOfAttempt' before considering it dead	Please refer to paragraph **Error! Reference source not found.**.
forcePingTimeout	Specify in seconds a timeout before to force the gateway to send a ping (MGMT_LQI_Req) to the nodes (0 to disable)	Please refer to paragraph **Error! Reference source not found.**.
autoDiscoveryUnknownNodes	Automatic discovery unknown nodes before sending or after receiving a message (1 to enable, 0 to disable)	This is an important feature: every time the gateway receives messages from nodes that has not been yet recognized/included in the neighbors table entry shall be polled in order to find out more information (in particular all those details needed by the gateway to have a valid entry into its database)
gwStatusChanged_DefaultURIListener	[Optional] "Callback URI listener" address. It represent the "GW Status Changed event" default URI Listener	This is a development-specific variable inserted to inform an external application about the status of the gateway and its possible changes. It is an important feature but it could be implemented

		using API commands (a registration to this event would be enough)
apscMaxWindow Size	APS level parameters (required for fragmentation services)	This is a variable that solve a specific need (fragmentation) and thus probably needs to be maintained.
apsInterframeDel ay	APS level parameters (required for fragmentation services)	This is a variable that solve a specific need (fragmentation) and thus probably needs to be maintained.
debugEnabled (HTTPServerDL, HTTPSessionCon textDL, RESTClientDL, RESTBrokerDL, DiscoveryAgent DL, ManagementAge ntDL, LocalNodeDL)	To enable debug messages select "1" (0 to disable) Printing levels for class debugging - 0 (DEBUGGING), 1 (TEST), 3 (PRODUCTION)	This is a typical variable used for debugging purpose: it allows the developer to decide what to print and which specific classes to enable. The current gateway has 3 different debug levels, but at the end only two of them are mainly used.
DeviceType	Device Type (0 =Current device type; 1=Coordinator; 2=Router; 3=End Device)	Fundamental setting to decide if the gateway shall be a coordinator or router (end device is not a standard setting anymore)
ChannelMask	Default Channel Mask [valid range: 11 - 26]	ZigBee channel mask: an optional feature could be to insert a mask or an “OR” symbol to allow multiple channel choice, useful in case we are a router able to join any channel or a coordinator that shall create a network where the noise is the lowest.
PANId	ZigBee Network PANId ('0xFFFF' = generate random value)	Typical ZigBee setting (optionally it could be generated randomly)
ExtendedPANId	Extended PAN Id (big-endian order)	Typical ZigBee setting (optionally it could be generated randomly)
StartUpControl	StartupControl (0=Association; 1=Orphan Rejoin; 2=Network Rejoin; 3=Find and Rejoin; 4=Silent Start)	Typical ZigBee setting
SecurityLevel	SecurityLevel (0=Disabled, 5=Enabled)	This variable is not needed anymore since all the ZigBee PRO stack shall mandatory support the security
freescaleMACAd dressValue	MAC Address mode: 0=use manufacture address (stored within the dongle), 1= generate random address	Any gateway shall be produced with a specific and not-editable IEEE Address, but if the stack vendor allows changing it this setting could be used.
networkKey	Default Network Key	Typical ZigBee setting (optionally it could be generated randomly)
preconfiguredLin kKey	Default Trust Center Link Key - 'ZigBeeAlliance09'	Typical ZigBee setting (optionally it could be generated randomly)
DefaultEndPoint	Default End Point	This setting shall be expanded. In ZigBee an endpoint is linked with a SimpleDescriptor and thus this part shall be changed so that the user can specify the default simple descriptor to be used.

TCKeyTableSize (IEEEAddressXX, TCLinkKeyXX)	TCKeyTableSize - Zero to disable the TrustCenterLinkKeytable - Note: maximum number of entries is 12 for Ember, 9 for Freescale	Typical ZigBee setting. If the gateway is a coordinator, this feature gives the possibility to specify special security requirements for ad hoc nodes.
IEEE_ADDRESS	Insert the Trust Center IEEE Address	Trust Center setting that could be reviewed.
JOINREQUESTPROCESSING	Decide if the local ACL Conf shall be used (INSERT true) or if the WSN-C access is required (INSERT false)	Trust Center setting that could be reviewed.
LEAVENOTIFICATIONPROCESSING	Decide if process locally the leave notifications without warning the WSN-C (SELECT true). Otherwise insert false	Trust Center setting that could be reviewed.
ACLFILENAME	File name containing the Access Control List	Trust Center setting that could be reviewed.
NETWORKKEY	The network master key, 16-bytes long	Trust Center setting that could be reviewed.
APPSECURITYTIMEOUTPERIOD	The Security Timeout period (expressed in milliseconds)	Trust Center setting that could be reviewed.
UseDefaultNWKRootURI	Decide if the Network Root URI can be obtained by appending the net/default' suffix (SELECT 1), or by appending the net/<ExtendedPANId>' suffix (SELECT 0)	This setting could be avoided (unless it is required to certify a gateway device as standard) and keep by default the "/net/default" path since so far the gateway has been thought to be used only for one network at time.
HelloMsgNotifyPeriod, HelloMsg_WSNCAttached_NotifyPeriod	Gateway Descriptor Asynchronous notification period (expressed in seconds), respectively before and after a WSN-C is attached. Note: This is like an "Hello" message to the WSN-C platform address	This is an obsolete setting that could be implemented into a new API to periodically inform an application (registered to this event) about the status of the gateway. See also 'gwStatusChanged_DefaultURIListener' since this is a similar feature that could be implemented in a unique API.
serverPorts	Port server where to receive REST commands	This is an important setting since it specifies where the gateway expects to receive REST commands.
Ssl, sslCertFile, httpDigestAuth, httpDigestAuthPwdFile	RESTServer parameters	Development-specific settings that depending on the new architecture could be avoided to be inserted.
httpOptTimeout	libcurl HTTP option application timeout (in seconds) - Note: for remote connection between GW and IPHA insert a value higher than 1. Set to zero to completely disable caching	This is a libcurl setting that the new gateway could not use in case the developer decide to manage the HTTP communication in a different way.
txPeriod	Select the txPeriod used during the Performance tests	Obsolete setting used for performance tests, not needed anymore.

Table 6 – Configuration file

[R24] All the parameters described in Table 6 shall be able to be modified using ad hoc primitives (part of them could use the standard ZigBee API Get and Set). The intent is to have predefined values that can all be modified at run time.

3.4 ZigBee IP requirements

The following figure reports the protocol layers of the ZigBee IP stack. The expected architecture foresees the implementation of the applications (e.g. SEP2 profile) on the host processor while the IP stack is executed on a network processor, i.e. a commercial USB key.

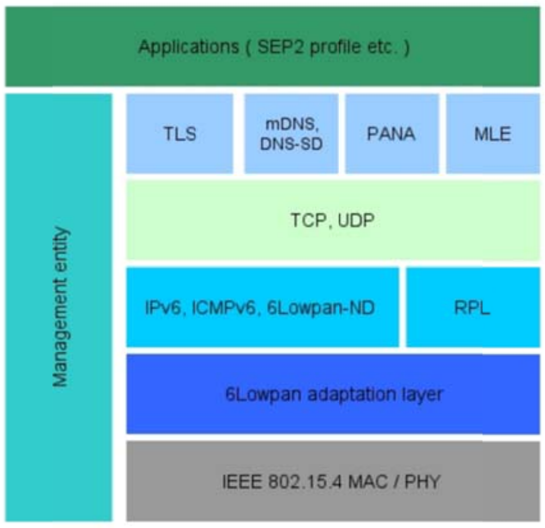

Figure 4 –ZigBee IP stack

[R25] The GAL for ZigBee IP shall implement the same functionalities reported above in the ZigBee Pro-related section.

[R26] The GAL shall implement communication with a commercial USB key where the ZigBee IP stack is executed. The implementation of the ZigBee IP stack is outside the scope of this call. The used commercial USB key will be agreed upon between the contractor and the contractee.

[R27] The ZigBee IP stack of the used commercial USB key shall comply with the public ZigBee IP specification https://docs.zigbee.org/zigbee-docs/dcn/09/docs-09-5023-20-0csg-zigbee-ip-2009-specification.pdf.

[R28] ZigBee IP stack is based upon IPv6 on IEEE 802.15.4, the GAL shall accordingly implement the device discovery and service discovery procedures in compliance with the SEP2 specification (mDNS, DNS-SD) [4].

[R29] The security should be managed directly by the ZigBee IP stack on the USB key by using TLS. Security related information, such as setting the network key to be transported and roll key timeouts, shall however be configurable at the application layer through the GAL.

4 Annex 1 – jGAL Java Interface

```
public interface GatewayInterface{
	/**
	 * Registration callback to receive notifications about events
	 * @param listener to receive notifications
	 */
	void setGatewayEventListener(GatewayEventListener listener);

	/**
	 * Retrieves the version and the main informations of the GAL.
	 * It can be used as a way to tell if and when the GAL is running as it does not
affect the status of the GAL and
	 *  it is a very light command
	 * @return
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
	 */
	Version getVersion() throws IOException, Exception, GatewayException;

	/**
	 * Retrieves a particular attribute of the database InfoBaseAttribute defined in
ZigBee Alliance
	 * @param attrId the ID of the attribute to retrieve
	 * @return
	 * @throws Exception
	 * @throws Exception
	 * @throws GatewayException
	 */
	String getInfoBaseAttribute(short attrId) throws Exception, Exception,
GatewayException;

	/**
	 * Allows the creation of a callback to receive APS/ZDP/ZCL messages using a class
of filters
	 *
	 * @param callback
	 * @param listener to receive notifications
	 * @return
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
	 */
	long createCallback(Callback callback, APSMessageListener listener) throws
IOException, Exception, GatewayException;

	/**
	 * Allows the creation of a callback to receive APS messages and specifying the
endPoint on which to listen.
	 * In fact represents a more fast version, compared to the previous function
createCallback that acts on all
	 * the endpoints
	 * @param endpoint on which to listen
	 * @param listener to receive notifications
	 * @return
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
	 */
	long createAPSCallback(short endpoint, APSMessageListener listener) throws
IOException, Exception, GatewayException;

	/**
	 * Allows the creation of a callback to receive APS messages. In fact represents a
more fast version,
	 * compared to the previous function createCallback that acts on all the endpoints
	 *
	 * @param listener to receive notifications
	 * @return
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
	 */
	long createAPSCallback(APSMessageListener listener) throws IOException, Exception,
GatewayException;

	/**
	 * Returns list of all callbacks to which you have previously registered
	 * @return
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
	 */
	List<Long> listCallbacks() throws IOException, Exception, GatewayException;

	/**
	 * Allows to remove a callback
	 * @param callId
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
```

```
         */
        void deleteCallback(long callId) throws IOException, Exception, GatewayException;

        /**
         * Returns the list of associated nodes in the network, and for each node gives
the short and
         * the IEEE Address
         * @return
         * @throws IOException
         * @throws Exception
         * @throws GatewayException
         */
        Aliases listAddresses() throws IOException, Exception, GatewayException;

        /**
         * Allows to configure a set of parameters throught the StartupAttributeInfo class
before
         * to launch the ZigBee network
         * @param sai the StartupAttributeInfo
         * @throws IOException
         * @throws Exception
         * @throws GatewayException
         */
        void configureStartupAttributeSet(StartupAttributeInfo sai) throws IOException,
Exception, GatewayException;

        /**
         * Allows to read a set of parameters throught the StartupAttributeInfo class
         * @param index
         * @return
         * @throws IOException
         * @throws Exception
         * @throws GatewayException
         */
        StartupAttributeInfo readStartupAttributeSet(short index) throws IOException,
Exception, GatewayException;

        /**
         * Allows to start/create a ZigBee network using the StartupAttributeInfo class as
parameter previously configured
         * @param timeout
         * @param sai the StartupAttributeInfo
         * @throws IOException
         * @throws Exception
         * @throws GatewayException
         */
        void startGatewayDevice(long timeout, StartupAttributeInfo sai) throws
IOException, Exception, GatewayException;

        /**
         * Allows to start/create a ZigBee network using a set of default values inside
the GAL
         * @param timeout
         * @throws IOException
         * @throws Exception
         * @throws GatewayException
         */
        void startGatewayDevice(long timeout) throws IOException, Exception,
GatewayException;

        /**
         * Returns the list of active nodes and connected to the ZigBee network from the
cache of the GAL
         * @return WSNNodeList the list of active nodes
         * @throws IOException
         * @throws Exception
         * @throws GatewayException
         */
        WSNNodeList readNodeCache() throws IOException, Exception, GatewayException;

        /**
         * Activation of the discovery procedures of the nodes in a ZigBee network.
         * Each node will produce a notification by the announcement
         * @param timeout
         * @param discoveryMask
         * @throws IOException
         * @throws Exception
         * @throws GatewayException
         */
        void startNodeDiscovery(long timeout, int discoveryMask) throws IOException,
Exception, GatewayException;

        /**
         * Allows the subscription to the event for which the node is no longer active
         * @param timeout
         * @param discoveryMask
         * @throws IOException
         * @throws Exception
         * @throws GatewayException
         */
        void subscribeNodeRemoval(long timeout, int discoveryMask) throws IOException,
Exception, GatewayException;

        /**
```

```
	 * Retrieves the local services (the endpoints) on which the GAL is running and
listening
	 * @return
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
	 */
	NodeServices getLocalServices() throws IOException, Exception, GatewayException;

	/**
	 * Returns the list of active endpoints from the cache of the GAL
	 * @return
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
	 */
	NodeServicesList readServicesCache() throws IOException, Exception,
GatewayException;

	/**
	 * Activation of the discovery procedures of the services (the endpoints) of a
node connected
	 * to the ZigBee network
	 * @param timeout
	 * @param addrOfInterest
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
	 */
	void startServiceDiscovery(long timeout, Address addrOfInterest) throws
IOException, Exception, GatewayException;

	/**
	 * Retrieves the informations about the ServiceDescriptor of a specific endpoint
of a ZigBee node
	 * @param timeout
	 * @param addrOfInterest
	 * @param endpoint
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
	 */
	void getServiceDescriptor(long timeout, Address addrOfInterest, short endpoint)
throws IOException, Exception, GatewayException;

	/**
	 * Retrieves the informations about the NodeDescriptor of a ZigBee node
	 * @param timeout
	 * @param addrOfInterest
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
	 */
	void getNodeDescriptor(long timeout, Address addrOfInterest) throws IOException,
Exception, GatewayException;

	/**
	 * Allows the creation of an endpoint to which is associated a SimpleDescriptor
	 * @param timeout
	 * @param desc the SimpleDescriptor
	 * @return
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
	 */
	short configureEndpoint(long timeout, SimpleDescriptor desc) throws IOException,
Exception, GatewayException;

	/**
	 * Allows to remove a SimpleDescriptor or an endpoint
	 * @param endpoint the endpoint to remove
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
	 */
	void clearEndpoint(short endpoint) throws IOException, Exception,
GatewayException;

	/**
	 * It’s a command to generate the disassociation of all the nodes from the network
ZigBee
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
	 */
	void leaveAll() throws IOException, Exception, GatewayException;

	/**
	 * It’s a command to generate the disassociation of a node from the network ZigBee
	 * @param timeout
	 * @param addrOfInterest
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
	 */
	void leave(long timeout, Address addrOfInterest) throws IOException, Exception, 
GatewayException;

	/**
	 * It’s a command to generate the disassociation of a node from the network 
ZigBee. Mask equals to 0x00 close the network, 0xff leaves the network open, and any 
other value leaves the network open for that number of seconds.
	 * @param timeout
	 * @param addrOfInterest
	 * @param mask
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
	 */
	void leave(long timeout, Address addrOfInterest, int mask) throws IOException, 
Exception, GatewayException;

	/**
	 * This command allows to create a binding of a remote node to a prefefined 
destination address node
	 * @param timeout
	 * @param binding
	 * @throws IOException
	 * @throws JAXBException
	 * @throws GatewayException
	 */
	void addBinding(long timeout, Binding binding) throws IOException, JAXBException, 
GatewayException;

	/**
	 * This command removes a previously created binding of a remote node
	 * @param timeout
	 * @param binding
	 * @throws IOException
	 * @throws JAXBException
	 * @throws GatewayException
	 */
	void removeBinding(long timeout, Binding binding) throws IOException, 
JAXBException, GatewayException;

	/**
	 * This command request a list of all the bindings stored in a remote node, 
starting from index zero
	 * @param timeout
	 * @param aoi
	 * @throws IOException
	 * @throws JAXBException
	 * @throws GatewayException
	 */
	void getNodeBindings(long timeout, Address aoi) throws IOException, JAXBException, 
GatewayException;

	/**
	 * This command request a list of all the bindings stored in a remote node
	 * @param timeout
	 * @param aoi
	 * @param index
	 * @throws IOException
	 * @throws JAXBException
	 * @throws GatewayException
	 */
	void getNodeBindings(long timeout, Address aoi, short index) throws IOException, 
JAXBException, GatewayException;

	/**
	 * Allows the opening of the ZigBee network to all nodes, and for a specified 
duration,
	 * to be able to associate new nodes
	 * @param timeout
	 * @param duration
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
	 */
	void permitJoinAll(long timeout, short duration) throws IOException, Exception, 
GatewayException;

	/**
	 * Allows the opening of the ZigBee network to a single node, and for a specified 
duration,
	 * to be able to associate new nodes
	 * @param timeout
	 * @param addrOfInterest
	 * @param duration
	 * @throws IOException
	 * @throws Exception
	 * @throws GatewayException
	 */
	void permitJoin(long timeout, Address addrOfInterest, short duration) throws 
IOException, Exception, GatewayException;

	/**
	 * Sends an APS message to a node in blocking mode
	 * @param message
	 * @throws IOException
```

```
         * @throws Exception
         * @throws GatewayException
         */
        void sendAPSMessage(APSMessage message) throws IOException, Exception,
GatewayException;

        /**
         * Sends an APS message to a node in an asynchronous mode
         * @param timeout
         * @param message
         * @throws IOException
         * @throws Exception
         * @throws GatewayException
         */
        void sendAPSMessage(long timeout, APSMessage message) throws IOException,
Exception, GatewayException;

        /**
         * Resets the GAl with the ability to set whether to delete the NonVolatileMemory
to the next reboot
         * @param timeout
         * @param mode
         * @throws IOException
         * @throws Exception
         * @throws GatewayException
         */
        void resetDongle(long timeout, short mode) throws IOException, Exception,
GatewayException;
}
```

5 Annex 2 – Discovery and Freshness Algorithms

5.1 Discovery Algorithm

One of the most important features that a gateway shall implement is the discovery mechanism. In a typical scenario the gateway is generally the coordinator of the network and thus it is generally the first node in the network. Subsequently the other nodes can joined the network after a permitJoin command[1] is issued: each of them send an announcement message when they are in, and this allows the gateway to update its internal table to have a whole view of the network status. However, different things happen during the life of a ZigBee network, like for example the GAL is not a coordinator but a router (different mechanisms are applied here) or it simply crash and restart, causing to ignore other nodes already present in the network, and thus here a discovery is needed to create an entry again in the internal database for each of them. Also, often happen that when a remote node is powered off, it doesn't have the time to send a leave command, and thus the gateway still believe that this node is active and present in the network, while instead is not.

The algorithm implemented by the GAL to discovery all the node relies on the MGMT_LQI request and response commands. In the previous release of the GAL, other commands have been implemented and used (they're still present in the code), but the new gateway shall mandatory implement this mechanism using the MGMT_LQI, leaving the other ones as optional. It is then important to verify with the stack vendor that the MGMT_LQI.Request and MGMT_LQI.Response are supported, since by standard ZigBee those two APIs are optional.

We're now going to explain how to implement the startNodeDiscovery and the readNodeCache procedure, but once the code written by Telecom Italia will be available everything could became more clear. As shown in [1], both those procedures rely on the same API path: they differ only on one extra parameter present in the REST resource to indicate a request for the cache (in this case the gateway shall return a list of the current database, without even interact with the ZigBee stack, i.e. no message shall be sent over the air). If this is not the case, then we're going to have a discovery[2]. Telecom Italia have decided to use the MGMT_LQI commands since they return a list of neighbor that each node can hear: this is a good way to

[1] The permitJoin command opens the network for a specific amount of seconds, enabling a node to join the network.
[2] There are still some cases where the user wants just to register to the freshness algorithm updates and/or the announcement of a new node.

find out easily other existing nodes. The strategy is to clear first the database's entries, so that we are sure to have an empty DB without any old nodes that could be now dead (not present anymore in the network). Starting from the coordinator, we ask an MGMT_LQI_Req command to ourselves, and in response the gateway will collect all its neighbors. Recursively, for each of those entries the gateway will send the MGMT_LQI_Req, in order to find out their children, and so on. At the end of this interaction, the gateway shall have a collection of all the active nodes that are in the network. Please note that we are sure about the existence of a node due to its MGMT_LQI_Rsp received: if this not happens probably the node is not part of the network anymore[3]. Different behavior shall be considered to handle end device and sleepy end device: the first group is nodes that do not have a neighbor's table since they communicate only with a parent, while instead sleepy end device could respond to the MGMT_LQI.Request with a delay that shall be taken in consideration by the developer when performing a discovery[4].

The algorithm that will be shown in the next session (called "freshness algorithm") 'completes' the discovery mechanism: since a call to the discovery procedure removed all the entries in the database, any incoming message by unknown nodes shall be handled as well but those nodes shall be discovered as soon as possible. Thus, it is responsibility of the freshness algorithm to automatically ping those nodes that have not been communicated yet with the gateway so that their response will be handled naturally by the discovery mechanism.

5.2 Freshness Algorithm

The freshness algorithm has been introduced in GAL in order to have a way to recognize the status of the nodes in the network, in particular trying to understand when those nodes are still active and when they're out of the network (i.e. the node is not reachable anymore or it has been unplugged/powered off without have sent a leave command).

In Figure 5 is shown a simplified version of the logical diagram followed by the freshness algorithm. This is a quite complex algorithm and thus it is described with more details below. However, we suggest reviewing the core parts directly into the code and use this document just as a reference to have a clear understanding of what has been done. Most (if not all) the code that implements this feature is present in CDiscoveryAgent.cpp, in a function called "`keepAliveAlgorithm`", periodically called by a separated and independent thread. This function is responsible to verify, when the gateway is running and for all the nodes present in the network, coordinator included, that the node presents some activity within a predefined period (by configuration file). If the node is not interacting for such threshold period, the gateway tries to ping the node with a specific ZDO-ZigBee command.

Before to enter into more details, it is clear that this algorithm cannot work with sleepyEndDevice, since they could be sleepy nodes for a period much longer that the threshold, and thus there is no way to ping them to verify if they're still alive. Telecom Italia have decided to assume those nodes always present in the network and thus this algorithm ignore them.

The base of the algorithm relies on a timer, called `KANodeTimer`, which is stored for each node in ZbNetDb.cpp, 'Node' class: every time the gateway receives a message from that node it updates its timer to the current '`gettimeofday`' timer. This is an independent feature that is applied as soon as we receive the message, in GalZbIf.cpp, while instead the "`keepAliveAlgorithm`" just check periodically if the timer goes above a specific threshold. When a specific node has a timer higher than the threshold, we enter in the "ping"

[3] "Probably" since a different scenario is applied for sleepy nodes.
[4] Those nodes could also not "hear" and receive the MGMT_LQI.Request, generally sent in broadcast, since their parent are not supposed by standard to handle that message for them.

mode: a MGMT_LQI_Req command is sent[5] and we expect that the user respond with a MGMT_LQI_Rsp, which will automatically update the timer.

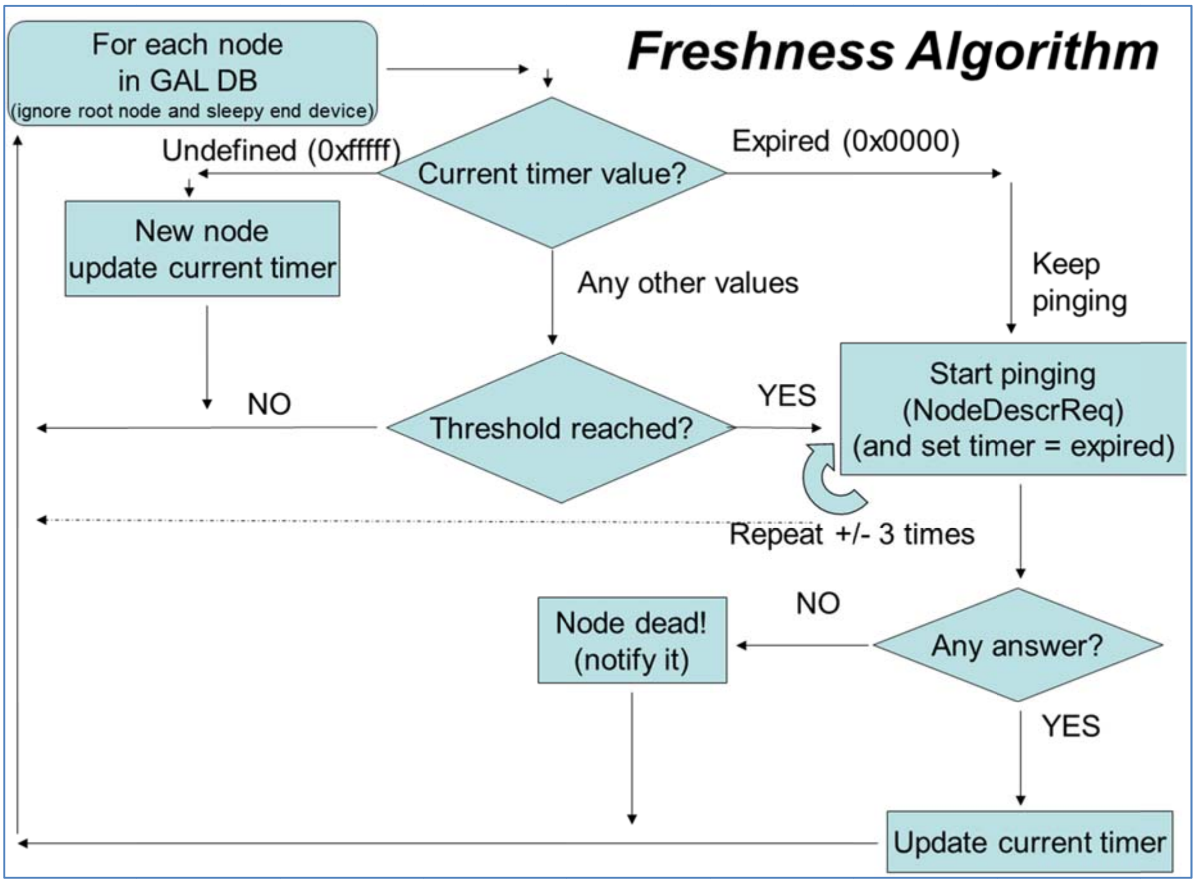

Figura 5 – Freshness algorithm

The next time the algorithm select this node, it will verify if the timer has been updated:

- If yes, the node will end its ‘ping’ mode procedure and it will be threaten as any other normal node
- If not, the gateway will try to send again a ping, and it will try until a “KAnumOfPings” is reached: after a certain number of retries the gateway shall announce the node as ‘dead’.

[5] Please note that other commands could be used as well, as long as they’re mandatory in the ZigBee specification.

6 References

[1] ZigBee Alliance. *Network Device Gateway Specification document*, document no. 075468r35

[2] ZigBee Alliance *Understanding ZigBee Gateway*, document no. 095465r12

[3] ZigBee Alliance *Profile Interoperability Requirements*, document no. 115487r08

[4] ZigBee Alliance *ZigBee IP Specification document*, document no. 095023r20

[5] European Commission. *The new SME definition. User guide and model declaration*, http://ec.europa.eu/enterprise/enterprise_policy/sme_definition/sme_user_guide.pdf