IOT Vega Server

Manual

Interruption

There is description of the IOT Vega Server abilities, ways to communicate with it and also settings for the first launch

Contents

IOT Vega Server description 3
Abilities 4
Installation 5
Settings 7
IOT Vega AdminTool 11
Attachment A. Description of the database structure 17
Attachment B. Structure and composition of the main messages in the console 21

IOT Vega Server description

The IOT Vega Server is a tool for organizing LoRaWAN networks of any scale.

It is intended for control of the base network of base stations operating under the control of the Packet forwarder software from Semtech, receiving data from the end devices and transferring them to external applications, as well as transferring data from external applications to LoRaWAN devices.

The server operates according to the LoRaWAN 1.02 specification and supports any end devices that operate under this version of the specification.

All data received from terminals is stored in the database built into the IOT Vega Server and is always available for external applications.

An open API based on Web Socket technology allows you to connect external applications to IOT Vega Server and use LoRaWAN's network capabilities in your projects.

IOT Vega Server is released as a console application for Windows, Linux and Linux-ARM (assembly for launching on gateways platform) operating systems.

To manage the server, IOT Vega AdminTool application is used with a simple user-friendly interface. AdminTool opens to the server administrator a wide range of options for managing the LoRaWAN network. With AdminTool, you can add new LoRaWAN endpoints to the network, view the network map, monitor the base stations, and manage user rights. IOT Vega AdminTool is provided as a Web application.

To work with devices, IOT Vega Pulse client application is used with a wide range of data extraction, processing and presentation capabilities in various formats (table, graph, report, diagram).

Abilities

- Support of all LoRaWAN 1.0.1 end devices
- Support of Class A and C end devices
- Integrated database
- Support of work with an external database
- User-friendly administrator application
- Network mapping
- Managing of network users
- Flexible alignment of devices connected to the server
- Ability to set a custom frequency plan
- Packets online browsing from each device
- Communication plots for each device within the network

Installation

Before the server's updating it is recommend making the reserve copy of the database (DB) and the configuration file for the following reasons:

- The new version of the server modifies DB structure, therefore in the case of the necessary to back to the previous version the server will cannot working with modified DB;
- The configuration file is replaced by the configuration file from the installation package, thereby settings return to the default values. The updating of the configuration file is necessary because the new version of the server may contain the new settings.

The **version for Windows** does not require installation. You need to unzip the archive and run the executable.

For the correct operation of the server, it is necessary to install the both libraries that are located in the **IOT Vega Server/msvc c ++ 2013**. Then you can start working with the server.

The **Linux version** comes in the form of an installation DEB package for 32-bit and 64-bit systems.

The process of installing the application on Linux:

- download file to PC
- install by running the command: **sudo dpkg -i /path/to /file/iot-vega-server-1.1.5.deb**;
- to configure, you need to change the contents of the file with the settings **/opt/iot-vega-server/settings.conf**;
- to start the server you must execute a script: **sudo ./iot-vega-server.sh** in the directory **/opt/iot-vega-server** , or write a new script specifying the path to the libraries (installation directory).

For correct working of MySQL plugin in **Linux** operation system it is necessary to make sure that **libssl** library (version less than 1.1.0) presented and symbolic links are valid. The default links supplies with the server's packet but in real either library versions or installation paths are different. So server launching is invalid sometimes. For links recovering you need to do the follow steps:

- go to directory with the server files **cd /opt/iot-vega-server**;
- run the command **ls –al**. Directory content will be shown as result. Pay attention to the two symbolic links:

  - libcrypto.so.10 -> /lib/x86_64-linux-gnu/libcrypto.so.1.0.0
  - libssl.so.10 -> /lib/x86_64-linux-gnu/libssl.so.1.0.0

  If the linked files exist in the system then that are real addresses specified after arrows. Server may be launching in that case.
- If that files are not found in the system then find them (command **find / -name "libcrypto*"**) and create symbolic links (command **ln –s "path_to_library" libcrypto.so.10**) for both of files. After that run the command **ls –al** again and make sure that both links became are correct.
- If the server is not launching after all steps it is necessary to go to the **sqldrivers** directory (command **cd sqldrivers**). Then run the command: **LD_LIBRARY_PATH=/opt/iot-vega-server/ ldd libqsqlmysql.so**

Approximate result is:

```
linux-vdso.so.1 =>  (0x00007ffd035fe000)
libQt5Sql.so.5 => /opt/iot-vega-server/libQt5Sql.so.5 (0x00007f9a76730000)
libQt5Core.so.5 => /opt/iot-vega-server/libQt5Core.so.5 (0x00007f9a76010000)
libmysqlclient.so.18 => /opt/iot-vega-server/libmysqlclient.so.18 (0x00007f9a75a40000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f9a75822000)
libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1 (0x00007f9a75609000)
libssl.so.10 => /opt/iot-vega-server/libssl.so.10 (0x00007f9a753aa000)
libcrypto.so.10 => /opt/iot-vega-server/libcrypto.so.10 (0x00007f9a74fce000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f9a74dca000)
libstdc++.so.6 => /usr/lib/x86_64-linux-gnu/libstdc++.so.6 (0x00007f9a74ac6000)
libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f9a747c0000)
libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007f9a745aa000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f9a741e1000)
libicui18n.so.56 => /opt/iot-vega-server/libicui18n.so.56 (0x00007f9a73d48000)
libicuuc.so.56 => /opt/iot-vega-server/libicuuc.so.56 (0x00007f9a73990000)
libicudata.so.56 => /opt/iot-vega-server/libicudata.so.56 (0x00007f9a71fad000)
librt.so.1 => /lib/x86_64-linux-gnu/librt.so.1 (0x00007f9a71da5000)
libgthread-2.0.so.0 => /usr/lib/x86_64-linux-gnu/libgthread-2.0.so.0 (0x00007f9a71ba3000)
libglib-2.0.so.0 => /lib/x86_64-linux-gnu/libglib-2.0.so.0 (0x00007f9a7189b000)
/lib64/ld-linux-x86-64.so.2 (0x00007f9a76b8a000)
libpcre.so.3 => /lib/x86_64-linux-gnu/libpcre.so.3 (0x00007f9a7165d000)
```

That command displays dependencies for MySQL database plugin. If case of empty lines presence please contact to our technical support.

The **version for Linux-ARM** does not require installation. You need to unzip the archive **iot-vega-server (armhf) v1.1.5.tar.gz** at the **/opt/** directory by the command **tar –xvf iot-vega-server (armhf) v1.1.5.tar.gz**.

Then you need to change the configuration file up to your requirements. Server launched by the **iot-vega-server.sh** script.

Work with external database is not supported.

Settings

The server is autonomous, only the initial configuration is required before the server starting. To perform the initial configuration of the server, you need to go to the **IOT Vega Server** folder and open the **settings.conf** file using any text editor (for example, Notepad).

When editing the settings.conf file, you cannot use the "!" symbol - an exclamation point

The contents of the file are as follows:

```
# Host connection settings
[host]
        # IP-address for UDP connection (gateway connection)
ip=127.0.0.1
        # Port for UDP connection (gateway connection)
udpPort=8001
        # Port for TCP (WebSocket) connection
tcpPort=8002
        # "path" part of webSocket address
webSocketPath=/
        # Flag of using SSL encryption for WebSocket
useSSL=0
        # SSL certificate filename (certificate must be in server's directory)
certFileName=cert.crt
        # SSL key filename (key must be in server's directory)
keyFileName=key.key

# LoRaWAN network settings
[lora]
        # LoRaWAN network identifier (should be random between 1 and 127)
networkID=1
        # Flag for using Plug-and-Play gateways function.
        # If this value is 1, server would automatically append all gateways which connected to
one
usePnPGateway=1

# Super user options
[root]
        # Login for super user
root=root
        # Password for super user (recommendation: change this password to your own)
password=123

# Console settings (volume of debug information)
[console]
        # Maximum level of console messages that will be shown (levels of messages represented
below)
maxMsgLevel=20
        # Maximum level of console messages that will be saved into LOG file (levels of messages
represented below)
maxLogMsgLevel=0
        #       Console message levels:
        # errors          = 0
```

```
        # uplink        = 1
        # downlink      = 2
        # warning       = 3
        # info          = 4
        # debug         = 20

    # External DataBase settings
[external_db]
        # Flag of using external DB
useExternalDb=0
        # Type of external DB. Supported only next types:
        #       MYSQL
        #       SQLITE
typeExternalDb=MYSQL
        # Name of external DB (schema's name for MYSQL)
nameExternalDb=server
        # IP and port of DB's server ("localhost" is supported)
ipExternalDb=127.0.0.1
portExternalDb=5505
        # User login and password (user should have maximum level of privileges)
userExternalDb=admin
passwordExternalDb=admin

    # self-test settings
[selftest]
        # Flag for enabling anonymous reports with possible server errors
enableReports=1
```

The [host] section contains port and network connection settings:
ip - IP address through which the base stations will work

This parameter must be changed to the correct IP address through which the base stations will work. Otherwise, there will be no communication with the base stations

udpPort – port for the base stations (BS);
tcpPort – port for external applications (including for AdminTool);
webSocketPath - the contents of the "path" field in the WebSocket address. If you specify /ws in this field, then when you connect from AdminTool you will need to specify ws://address: port/ws;
useSSL - flag to enable SSL encryption for the WebSocket;
certFileName - the name of the file that contains the SSL certificate (with the extension);
keyFileName - the name of the file that contains the SSL key (with the extension).

To establish the correct operation of SSL encryption, you must perform the following actions:

- Obtain an SSL certificate (and a key) signed by a trusted certification authority (this can also be a free trusted center, for example, sslforfree or letsencrypt);
- Install OpenSSL version 1.0.2 (the server does not work with the version of OpenSSL 1.1.0);
- Copy the certificate and key files (for linux it is enough to create links) in the directory with the server;
- Allow SSL encryption, prescribe the appropriate file names (with a certificate and a key) in the configuration file;

- Start the server, monitor for possible errors.

OpenSSL for Windows is available on iotvega.ru by link http://en.iotvega.com/content/ru/soft/server/Win32OpenSSL-1_0_2n.zip.

The [lora] section contains the LoRa network's identification dates.
networkID is parameter which determinate the LoRaWAN network's identificator. Before running the server, you must set a random value in the range from 1 to 127 inclusive.

If several LoRaWAN networks use the same NetworkID in the immediate vicinity - each of the networks will continue to work in normal mode, but periodically there will be errors about the incorrectness of the NwkSKey of this or that device: "INVALID_DEVICE_NETWORK_SESSION_KEY"

usePnPGateway - the permission flag to automatically add the BS to the server. If this option is enabled, the server will add any unknown BS to the list of registered BS with default parameters.

The [root] section defines the root user's identification, where **root** is the login, and **password** is the password.

The [console] section specifies the amount of information to be output to the console and stores it in the LOG file (the parameters of this section are updated every minute, so you do not need to restart the server to change the output level):

maxMsgLevel - the maximum level of messages (inclusive) that will be displayed in the server console. The decoding of message levels is given below;

maxLogMsgLevel - the maximum level of messages that will be stored in the LOG file.

Console message levels:

- 0 = messages with critical errors;
- 1 = uplink messages;
- 2 = downlink messages;
- 3 = warning messages;
- 4 = information messages (often with debugging information);
- 20 = messages with debugging information.

The [external_db] section contains external database (DB) work settings – after inserting changes need to restart server:
useExternalDb – flag permitting work with external DB. May be equal to 1 or 0;
typeExternalDb – parameter defines type of the external DB stroke. There are two DB types supported at present time:

- MYSQL;
- SQLITE – work and create DB file with name that is different from default name («server.db»).

nameExternalDb – DB naming. For SQLITE it corresponds to file name, for MYSQL is to scheme name;
ipExternalDb – IP address of the external DBMS server (don't use for SQLITE) as a stroke (supported value «localhost»);
portExternalDb – server port of the external DBMS (don't use for SQLITE);
userExternalDb – username for authorization on the external DBMS server (don't use for SQLITE);
passwordExternalDb – password for authorization on the external DBMS server (don't use for SQLITE).

User must have the maximum rights to insert changes at the DB composition

When you switch from a local database to an external database, when you start the server for the first time, the data from the local database will be moved to an external database

The [selftest] section contains the permission flag to send anonymous messages with possible detected bugs.

The server in the work process accumulates information about possible failures and can send data reports to Vega-Absolute developers once a day. This information will be extremely useful while searching and solving possible server failures. The sent information does not contain any confidential data.

Commands for the server are given in the **API IOT Vega Server Rev21.pdf** file.

IOT Vega AdminTool

IOT Vega AdminTool is a convenient Web application for server administration and allows you to add new LoRaWAN end devices to the network, view the network map, monitor the base stations, and manage user rights.

Let's see **an example of connecting a new or editing the parameters of an existing BS** on the server (Figure 1).

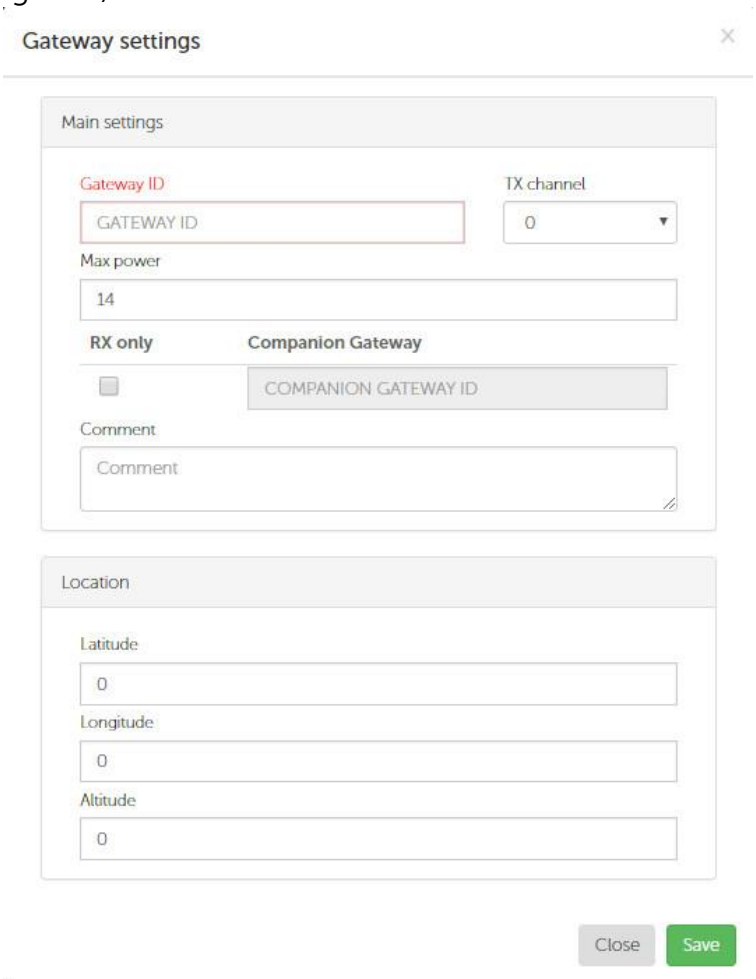

Fig. 1. The connecting window the BS to the server.

Required settings:

Gateway ID - identifier of the BS (16 hexadecimal symbols - 8 bytes);

TX channel is the BS channel used to send messages to the end devices (downlink). This parameter is specified in the software settings "packet_forwarder" on the BS (usually in the file "global_conf.json"). The default is 0 channel;

Transmit power - broadcast power BS. The maximum broadcasting power is usually determined by the circuitry of the BS and is limited in the software "packet_forwarder"; if this parameter is exceeded, the BS will return an error with the corresponding code;

Here it is necessary to specify the criteria by which the required value of the broadcast power is selected.

Let's look at the file "global_conf.json" from the settings of the software "packet_forwarder" on the BS.

The list of valid sets of broadcast parameters "tx_lut_ .." contains a set of powers on which the BS can transmit. For a standard list, this range is from -6 to 27 dBm with a different pitch (16 in total).

There is also an "antenna_gain" parameter, which determines the antenna gain (in dBm).

As a result, the difference in the values of "Transmit power" and "antenna_gain" should correspond to one of the values from the list "tx_lut_ ..". For example, "antenna_gain" = 3 dBm and it is planned to transmit data at a power of 10 dBm ("Transmit power"). The difference between "Transmit power" and "antenna_gain"; is 7dBm, but this value is not in the list "tx_lut_ .." (and if you use this set of parameters, when you try to send data, the BS will send a message with an error about the incorrect power). The closest allowed values from "tx_lut_ .." are 6 and 10 dBm, respectively, for this configuration the allowed values of "Transmit power" will be 9 or 13 dBm.

RX only - flag indicating that the BS is only receiving. When this flag is set, data transfer to the end devices via the corresponding BS is prohibited. This option is introduced for the organization of a "full duplex BS": this is an option when two BSs are installed in the immediate vicinity and one of them works only for reception, and the second works in the usual mode. At the same time, a continuity of listening to the radio is achieved;
Companion Gateway - the identifier of the BS of the companion working in a normal mode at the organization of "full duplex BS";
Comment - a field for comments (for example, the name of the BS);

Area Location - contains fields for inputting the location coordinates of the BS: latitude, longitude, altitude. If the GPS has a built-in GPS receiver, the entered coordinates will be updated with the current values automatically.

Let's analyze an example of registering a new device or changing the parameters of an existing one (Figure 2 and Figure 3).

Figure 2 shows a screenshot of the form for registering the device with basic parameters.

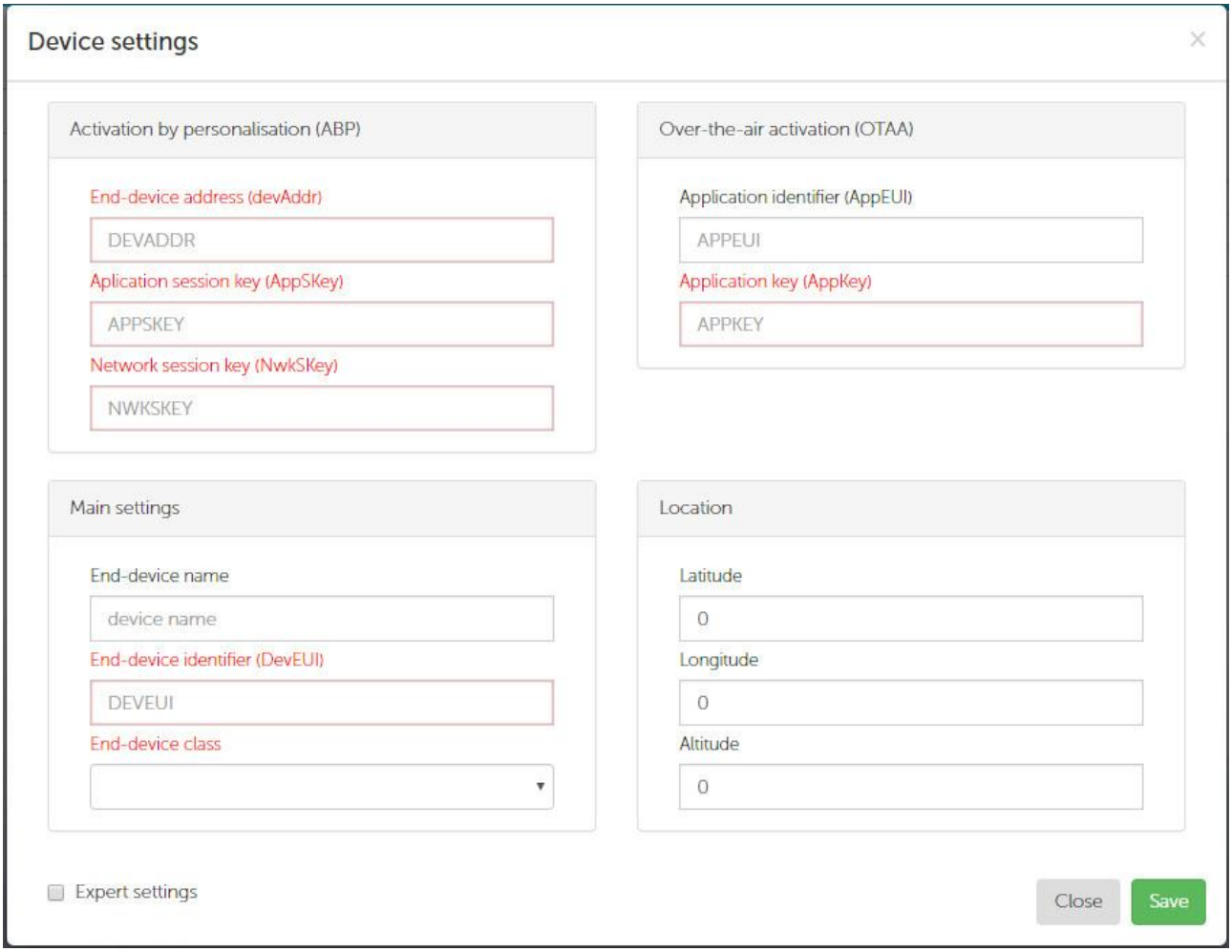

Fig. 2. The main parameters for registering the device on the server.

The **Activation by personalization (ABP)** area contains the parameters required to register the device on the server via ABP:
devAddr is the device address in the LoRaWAN network. This is a 32-bit number in hexadecimal form, for example 012345AB;
AppSKey - session key of the application is a string of 32 hexadecimal characters;
NwkSKey - network session key is a string of 32 hexadecimal characters.

The **Over-the-air activation (OTAA)** area contains the parameters required to register the device on the server through OTAA:
AppEUI - EUI device application identifier - a string of 16 hexadecimal characters;
AppKey - the device application key is a string of 32 hexadecimal characters.

To register the device, it is sufficient to specify at least one of the activation types or both.

The **Main settings** area contains the following parameters:
End-device name - the user name of the device;
DevEui - EUI device identifier (unique device number) - a string of 16 hexadecimal characters;
End-device class - class of the device. This parameter can be equal to the two values: CLASS_A or CLASS_C. Support for CLASS_B devices is under development.

The **Location** area contains fields for specifying the location coordinates of the device: latitude, longitude, altitude.

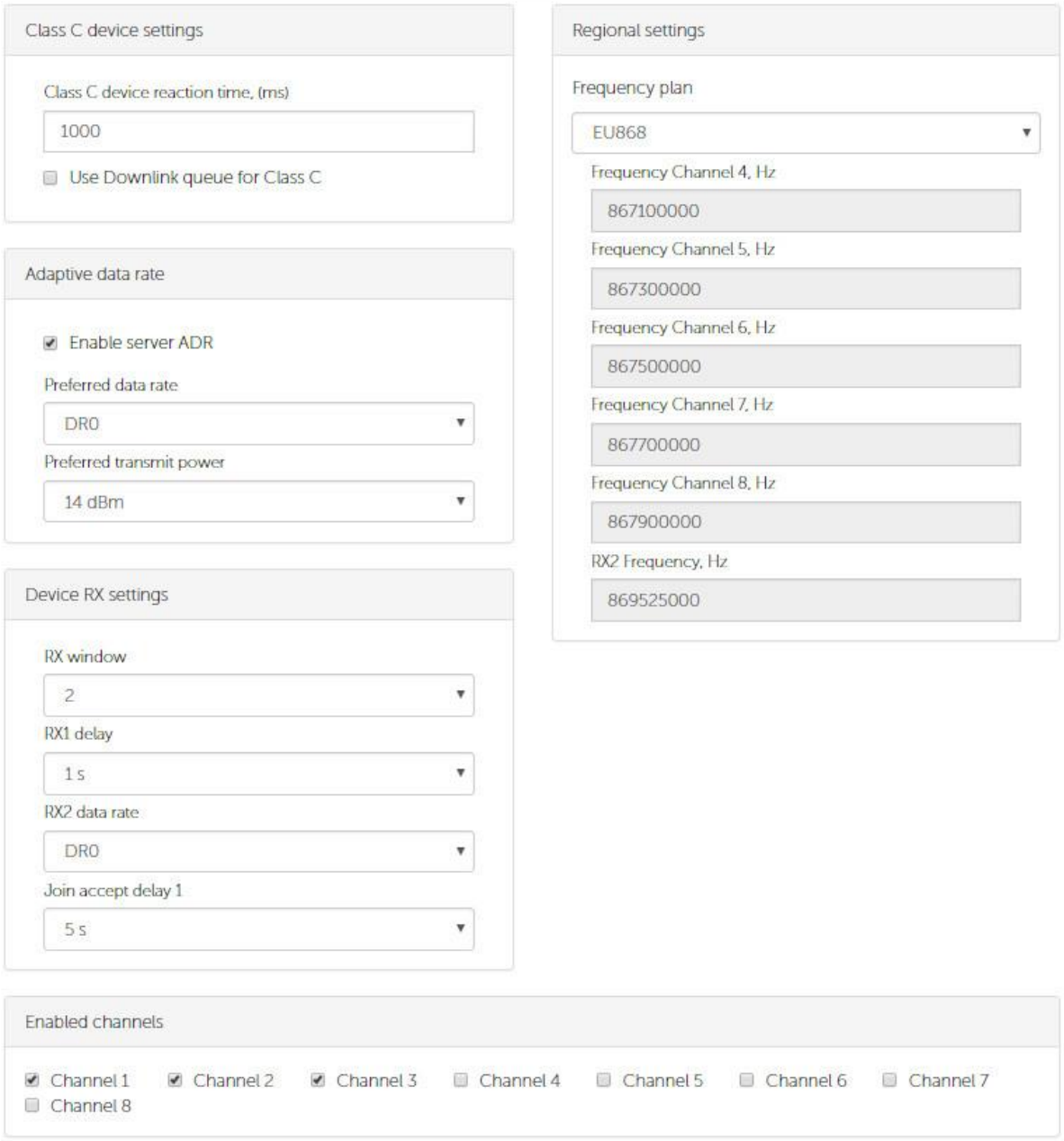

Fig. 3. Expert parameters for registering the device on the server.

Clicking on the checkbox **Expert settings**, you can access the expert settings of the device (see Figure 3).

The **Class C device settings** area is only available to CLASS_C devices. The following options are available:

Class C reaction time is the reaction time of the CLASS_C device. This is the time between the end of the reception of the server message by the device and the beginning of the sending of a possible response from the device (ie, in fact, the time to prepare a possible response). The parameter is introduced to improve the quality of work with messages from the server to devices of class C;

Use downlink queue for Class C - flag allowing to place packets for sending to the sending queue. Often CLASS_C devices work in "online" mode and if the package is not delivered to the device at the moment, its delivery is no longer required - in this case the queue of packages is not needed. Quite the contrary, the queue will play a negative role, as a drive of already obsolete information, which will be transmitted without fail, occupying airtime. By default, the option is disabled.

The Adaptive data rate area contains settings for the ADR algorithm designed to automatically change the speed of the broadcasts of devices depending on the quality of the connection (in good reception conditions, the speed will increase, thereby reducing the transmission time of packets and increasing the battery life of the device):
Enable server ADR - a flag that allows the ADR algorithm to work on the server for a specific device (if the checkbox is cleared, the server will not adjust the transfer rate of the device, even if the ADR algorithm on the device itself is activated);
Preferred data rate - the value of DR (speed) to which the ADR algorithm of the server will strive;
Preferred transmit power - the power of the device's transmitter that the server will set to it in the next session.

The parameter 'Preferred transmit power' should be changed (reduced relative to the default value) in the case that the quality of the connection for the device at maximum speed is satisfactory

The Device RX settings area contains settings for the device receiving windows:
RX window - the number of the receiving window (1 or 2) through which the server will transmit data by default. If you set the first window, then if the server cannot send data to the 1st receiving window (for example, there is no free BS), an attempt will be made to send data to the 2nd receiving window;
RX1 delay - the delay of the device opening the first receiving window (the default is 1 second). The 2nd receiving window is always opened (if the data was not received at the first receiving window) 1 second after the 1st;
RX2 data rate - data transfer rate for the 2nd receiving window;
Join accept delay 1 - the delay of the device opening the first receiving window for obtaining registration information when activated on the network by the OTAA method (by default 5 seconds).

If the gateway operates via the mobile Internet, then the delay in delivering packets to such a gateway can reach significant values. Therefore, in order to avoid problems with the operation of devices (sending confirms and MAC commands), it is necessary to shift the opening time of the first receiving window by more than 1s. Usually 3s interval is sufficient. In some cases, you will need to shift to a larger interval - you need to monitor the stability of the network.

The Regional settings area contains the frequency plan settings for the corresponding device. Here we offer a choice of two existing sets (the official European frequency plan and one of the variants of the Russian frequency plan), and there is also the possibility to configure a unique set of frequencies.
Each of the frequency sets consists of:

- Three channels with fixed frequencies (these channels are fixed in the device without the possibility of changing them through the LoRaWAN protocol);
- Five channels for receiving and transmitting messages. If the channel is not used, you must set the frequency to zero and disable the corresponding check box in the Enabled settings;
- One channel that determines the frequency of data reception for the second receiving window.

The active channel mask (the **Enabled** flags list) is located to the left of the frequency input fields and contains the disable/enable flags of the corresponding channels (frequencies) for data transfer on the device. This parameter is passed to the device at each JOIN procedure and in each message of the ADR algorithm (if ADR is enabled).

Attachment A. Description of the database structure

IOT Vega Server constantly uses the database (built-in SQLite or external) in the course of its work. To get started, you need minimal settings for accessing the external database, and for the built-in database settings are not required at all. The database structure is automatically created in both cases. It is necessary to exclude the influence of the user directly on the database structure, so as not to disrupt the server.

However, there is an option to work with the database in read-only mode. For example, while reading the accumulated data from the device. This approach is designed to remove the load from the server to read data from devices through the API and generally speed up the system work with user requests.

The database consists of the next tables:

- «bs» - list of connected gateways;
- «devices» - list of connected end-devices;
- «rawdata» - data received from devices and transferred to devices;
- «queuetransmit» - queue for sending packets to the device;
- «coveragemap» - the list of gateways through which data was received from the corresponding device. According to this table, you can build a network coverage map;
- «deviceattributes» - property list of the corresponding device;
- «users» - a list of users;
- «userdevices» - list of devices available to the appropriate user.

Description of «queuetransmit», «deviceattributes», «users» and «userdevices» tables will not be given, because these tables either store buffer information, or information in a specific format. It is assumed that this information does not require direct access for reading and is always available through fast executable API commands

Table 1 – Fields structure of the «bs» database table

Field	Data type	Description
id	INTEGER	Identifier (is not used)
mac	TEXT	Gateway ID
comment	TEXT	Comment
latency	INTEGER	Current network latency gateway-server
downport	INTEGER	Port for data transfer to the gateway
hostaddress	TEXT	Gateway IP address for data transfer
longitude	REAL / DOUBLE	GW coordinate – longitude
latitude	REAL / DOUBLE	GW coordinate – latitude
altitude	INTEGER	GW coordinate – altitude
active	INTEGER	GW activity flag (on-line)
rxonly	INTEGER	GW operation flag is only for reception
complimentarybsmac	TEXT	ID of the GW-companion through which it is allowed to transmit data if the current GW only works for reception
downlinkchannel	INTEGER	Data transmission channel
maxpower	INTEGER	Transmission power through the GW
lastonline	INTEGER	Time of the last activity of the GW

Table 2 – Fields structure of the «rawdata» database table

Field	Data type	Description
data	BLOB	Data in the HEX array
macbs	TEXT	The gateways ID through which the packet was transmitted is combined through the "+" sign
deveui	TEXT	Device ID
rssi	INTEGER	RSSI
snr	REAL / DOUBLE	SNR
freq	INTEGER	Frequency in Hz
sf	INTEGER	SF
time	INTEGER	Time in ms format that has passed since the beginning of the UNIX era
fcntup	INTEGER	Packet number
port	INTEGER	Number of the used port
ack	INTEGER	Presence of the ACK flag in the package
macdata	BLOB	MAC commands
type	INTEGER	Packet type (see command «get_data» description in API)
direction	INTEGER	Direction of transmission: «UPLINK» or «DOWNLINK»
status	TEXT	Status (see command «get_data» description in API)
activationside	INTEGER	Flag, shows the type of activation used while transmitting the current packet

Table 3 - Fields structure of the «coveragemap» database table

Field	Data type	Description
id	INTEGER	Identifier (is not used)
deveui	TEXT	Device ID
macbs	TEXT	Gateway ID
snr	REAL / DOUBLE	SNR
rssi	INTEGER	RSSI
time	INTEGER	Time in ms format that has passed since the beginning of the UNIX era

Table 4 - Fields structure of the «devices» database table

Field	Data type	Description
deveui	TEXT	Device ID
devname	TEXT	Name of the device
appeui	TEXT	Device AppEUI
appkey	BLOB	AppKEY
devaddrabp	INTEGER	DevAddr for ABP activation type
nwkskeyabp	BLOB	NwkSKEY for ABP activation type
appskeyabp	BLOB	AppSKEY for ABP activation type
devaddrotaa	INTEGER	DevAddr, generated by the server with OTAA activation type
nwkskeyotaa	BLOB	NwkSKEY, generated by the server with OTAA activation type
appskeyotaa	BLOB	AppSKEY, generated by the server with OTAA activation type
devnonce	INTEGER	Last DevNonce in JOIN request
lastjoints	INTEGER	Last JOIN request in ms format, that has passed since the beginning of the UNIX era
class	INTEGER	Device class: 1 = «A», 3 = «C»
rxwinnum	INTEGER	The number of the receiving window on the device
timingprof	INTEGER	Is not used, may be absent
lastrssi	INTEGER	Last RSSI at the device packet
lastsnr	INTEGER	Last SNR at the device packet
lastpower	INTEGER	Transmission power of the device
lastsf	INTEGER	Last SF at the device packet
lastch	INTEGER	Is not used
preferdr	INTEGER	Max DR with ADR work
preferpower	INTEGER	Is not used
fcntup	INTEGER	Last received packet number
fcntdown	INTEGER	Last transmitted packet number
dutycycle	INTEGER	Is not used
chanmask	INTEGER	16-bit channel mask

battery	INTEGER	Is not used
adr	INTEGER	The permission flag to use ADR (device)
macbs	TEXT	Is not used
lastdatats	INTEGER	The last device activity time in ms
longitude	REAL / DOUBLE	Device coordinate – longitude
latitude	REAL / DOUBLE	Device coordinate – latitude
altitude	INTEGER	Device coordinate – altitude
freqplan	INTEGER	Is not used, may be absent
reactiontime	INTEGER	Device response time
lastactivatside	INTEGER	The activation type used in the last received packet
rx1delay	INTEGER	Delay of opening the first receiving window, s
rx2delay	INTEGER	Delay of opening the second receiving window, s
join1delay	INTEGER	Delay of opening the first receiving window with JOIN, s
join2delay	INTEGER	Delay of opening the second receiving window with JOIN, s
rx2dr	INTEGER	DR the second receiving window of the device
rx2freq	INTEGER	Frequency of the second receiving window, Hz
freq4	INTEGER	Frequency of the channel №4, Hz
freq5	INTEGER	Frequency of the channel №5, Hz
freq6	INTEGER	Frequency of the channel №6, Hz
freq7	INTEGER	Frequency of the channel №7, Hz
freq8	INTEGER	Frequency of the channel №8, Hz
usedownqueue	INTEGER	Package queue packet send flag
adrthrescounter	INTEGER	The buffer parameter
serveradrenable	INTEGER	The permission flag to use ADR (server)

Attachment B. Structure and composition of the main messages in the console

IOT Vega Server application in the process of working displays some operational information in the console. The amount of displayed information depends on the corresponding server settings (see description of the configuration file).

In the console, messages may appear with debugging information, the contents of which may change. But the basic types of messages are unchanged. Below are descriptions of their structure and composition.

1. A message about an unsupported version of the protocol used in the PacketForwarder software on the GW.

The message is highlighted in red and starts with a line «**WARNING! Skip gateway msg, invalid protocol version:**». Next information is about the GW IP-address and the protocol version used;

2. A message is about an error that occurred when the packet was transferred to the device via the GW.

The message is highlighted in lilac and has a format: «**<< %1 | TX_ACK %2 | %3**», where

%1 – gateway ID;
%2 – error reception time;
%3 – error code as a string.
Possible error codes:

- «**TOO_LARGE_GW_PING_ERR**» - ping to GW is too big;
- «**COLLISION_ERR**» - the packet time specified in the packet is already taken up by another message;
- «**POWER_ERR**» - in the last packet an incorrect transmission power value indicated;
- «**FREQ_ERR**» - in the last packet an incorrect transmission frequency value indicated;
- «**NO_VACANT_GW**» - no vacant gateway;
- «**PAYLOAD_SIZE_ERR**» - it is impossible to send a packet of specified length.

3. Message about accepted JOIN request.

The message is highlighted in yellow and has a format: «**>> GW-%1: JOIN_REQ | %2 | %3 | %4 | SF%5 | RSSI:%6 | %7 | %8**», where

%1 – GW ID;
%2 – device ID (DevEUI);
%3 – date and time of message reception, accurate to ms;
%4 – frequency in MHz, accurate to 100 kHz;
%5 – SF value;
%6 – RSSI value;
%7 – SNR value;
%8 – result of message processing. Can take one of the following values:

- «**VALIDATED**» - the request has been processed correctly, a response will be sent;

- «**UNKNOWN DEVICE | DEVNONCE=0x%1**» - unknown device, where **%1** – is DevNonce used in the package in HEX;
- «**REPETITION DEVNONCE**» - repeated request reception;
- «**INVALID APPEUI**» - the AppEUI value from the request does not match what was specified when registering the device.

4. The message about the sent answer to the JOIN request.

The message is highlighted in blue and has a format: «**>> GW-%1: JOIN_REQ | %2 | %3 | %4 | SF%5 | VALIDATED**», where

%1 – gateway ID;
%2 – device ID (DevEUI);
%3 – date and time of message reception, accurate to ms;
%4 – frequency in MHz, accurate to 100 kHz;
%5 – SF value.

5. Notification of the received packet from the device.

The message is highlighted in green and has a format: «**>> GW-%1: %2 | %3 | %4 | %5 | SF%6 | RSSI:%7 | %8 | CNT:%9 | PORT:%10 | %11**», где

%1 – gateway ID;
%2 – packet type. «**CONF_UP**» or «**UNCOF_UP**» is used. If there are MAC data in the packet, the «**+M**» characters are added to the type;
%3 – device ID (DevEUI);
%4 – date and time of message reception, accurate to ms;
%5 – frequency in MHz, accurate to 100 kHz;
%6 – SF value;
%7 – RSSI value;
%8 – SNR value;
%9 – packet number;
%10 – port number;
%11 – additional information. May consist of the next information:

- if the packet is repeated from the device, this field contains the code «**REPETITION PACKET**» and the message text color turns navy blue;
- otherwise it contains information about the size of the received message in the form of «**[%1]**», where **%1** – is the size of the message in bytes. If in addition the MAC data is contained, then their size is added in the form «**| M [%1]**», where **%1** – is the size of the MAC data in bytes.

6. The message about the sent package for the device.

The message is highlighted in violet and has a format: «**>> GW-%1: %2 | %3 | %4 | %5 | SF%6 | CNT:%7 | PORT:%8 | %9**», where

%1 – GW ID;
%2 – packet type. «**CONF_DOWN**» or «**UNCOF_DOWN**» is used. If there are MAC data in the packet, the «**+M**» characters are added to the type;
%3 – device ID (DevEUI);
%4 – date and time of arrival of the message sending confirmation (not to be confused with the actual sending time);
%5 – frequency in MHz, accurate to 100 kHz;
%6 – SF value;
%7 – packet number;
%8 – port number;

%9 – contains information about the size of the transmitted message in the form «**[%1]**», where **%1** – is the size of the message in bytes. If in addition the MAC data is contained, then their size is added in the form «**| M [%1]**», where **%1** – is the size of the MAC data in bytes.

7. Reporting errors when sending data.

The message is highlighted in red and has a format: «**>> GW-%1: %2 | %3 | %4 | %5 |**», where

%1 – GW ID;

%2 – packet type. «**UNCONF**», «**CONFRM**» or «**UNCKNOWN**» is used. If there are MAC data in the packet, the «**+M**» characters are added to the type;

%3 – device ID (DevEUI);

%4 – date and time of attempted unsuccessful sending;

%5 – error code as a string.

Possible error codes:

- «**TOO_LARGE_GW_PING_ERR**» - ping to GW is too big;
- «**COLLISION_ERR**» - the packet time specified in the packet is already taken up by another message;
- «**POWER_ERR**» - in the last packet an incorrect transmission power value indicated;
- «**FREQ_ERR**» - in the last packet an incorrect transmission frequency value indicated;
- «**NO_VACANT_GW**» - no vacant gateway;
- «**PAYLOAD_SIZE_ERR**» - it is impossible to send a packet of specified length.

Document Information	
Title	IOT Vega Server
Document type	Manual – Translation from Russian
Document number	B02-server-01
Revision and date	10 - 02.12.2018

This document applies to the following products:

Product name	Type number
Software	IOT Vega Server
	IOT Vega Admin Tool

Revision history

Revision	Date	Name	Comments
01	06.05.2017	KEV	Document creation date
02	06.06.2017	MAI	Minor edits
03	06.16.2017	KEV	Minor edits
04	07.10.2017	MAI	Part 'Installation' was added, part 'Settings' was corrected
05	08.14.2017	KEV	Minor edits
06	08.31.2017	MAI	Part 'IOT Vega AdminTool' was added, part 'Settings' was corrected
07	09.27.2017	MAI	Minor edits in the part 'IOT Vega AdminTool'
08	12.20.2017	MAI	Changes in the part "Settings"
09	01.15.2018	MAI	Changes in the part "Installation" and "Settings" due to Linux-ARM version additional
10	01.22.2018	BIY	At the "Settings" part were added recommendations for using valid characters while correcting settings.conf and link on openssl
11	02.12.2018	MAI	Attachments A and B were added

vega-absolute.ru