XBee Connector User Guide

UgCS 3.1.871

Contents

1 XBee Connector User Guide

-

1.1 Legal notice

See Disclaimer.

XBee® and **Digi®** are registered trademarks of **Digi International Inc.**

ZigBee® is a registered trademark of **ZigBee Alliance**.

1.2 Introduction

ZigBee is a wireless network standard targeted at wide development in wireless control and monitoring applications. It suits well for unmanned vehicles remote controlling.

There are several advantages of ZigBee remote control:

- It is possible to control a number of vehicles from one node.
- Multiple vehicles are able to form `wireless mesh network`.
- You can create encrypted radio channel between ground station and vehicle (ZigBee networks are secured by 128 bit symmetric encryption keys).

You can use all these advanced features in UgCS by virtue of XBee Connector. It allows to connect any supported vehicle to ground station via ZigBee protocol using Digi XBee modules.

Some of XBee module specifications:

- **RF interaction distance between two XBee**: up to 3200 meters in a line of sight
- **Data rate**: up to 250 Kbps
- **Radio frequency**: 2.4 GHz
- **Operating Temperature**: -40º C to +85º C

Warning

Note that only Pixhawk with ArduPilot firmware is supported at the moment.

1.3 Interaction model

All vehicle specific data exchange in UgCS is performed by VSMs. Every VSM are able to connect to XBee Connector by TCP. The only parameters to be configured in particular VSM are the host address and port number of XBee Connector running instance.

XBee Connector is responsible for detecting remote ZigBee nodes and routing data flow to correspondent VSM.

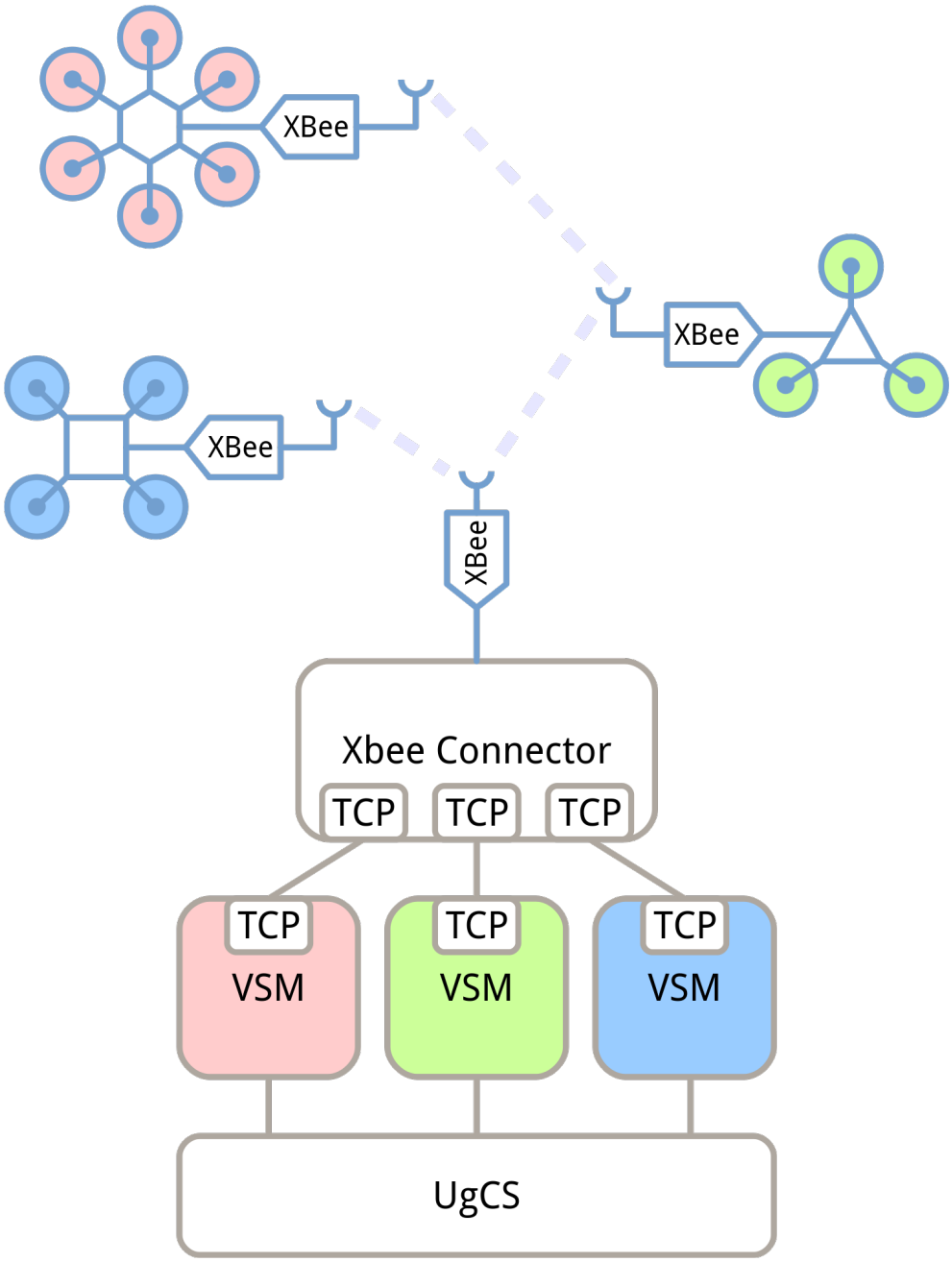

Figure 1: XBee interaction model

If one of the vehicles is not in a line of sight in terms of ZigBee link (e.g. due to being too far or behind an obstacle) then this vehicle can interact with ground station through other vehicle which is located in between.

1.4 Prerequisites

You need at least two Digi XBee ZigBee modules in order to connect your vehicle via ZigBee interface. You can choose options that would best fit your needs on manufacturer's website:

http://www.digi.com/products/xbee-rf-solutions/modules/xbee-zigbee

We recommend through-hole modules instead of SMT due to more simple connecting procedures.

Also you will need an appropriate USB-UART adapter for connecting XBee module to PC and autopilot. For example you can use DFRobot's one:

http://www.dfrobot.com/index.php?route=product/product&product_id=588

1.5 XBee configuration

There is a dedicated software for XBee modules configuration from Digi called **XCTU**. Please refer to Digi website in order to download it:

http://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu

After installing software you are to configure XBee modules.

1.5.1 Ground station side

First module should be appointed as **ZigBee Coordinator API**. Such kind of appointment is performed by firmware update:

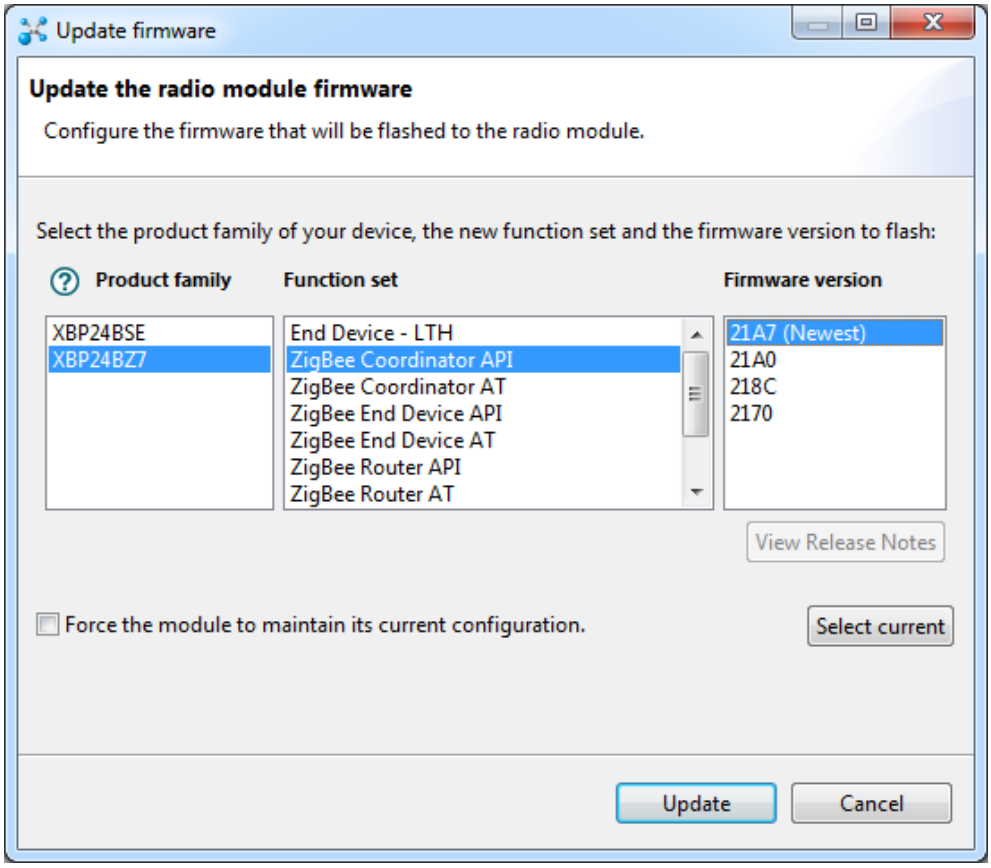

Figure 2: ZigBee Coordinator API firmware

This module is to be connected to PC via mentioned USB-UART adapter.

The next step is to configure Coordinator's parameters. Most important are:

- **ID** (PAN ID): **105**
- **NI** (Node Identifier): **ONGROUND**

Other parameters may be set to default values. Just in case check some of them:

- **PL** (Power Level): **Highest** (4)

- **BD** (Baud Rate): **115200** (7)

1.5.2 Vehicle side

A role of the second module (and subsequent if any) may be **ZigBee Router AT** as well as **ZigBee End Device AT.** ZigBee Router option is more preferrable because it brings more networking capabilities.

Choose correspondent firmware and update the module:

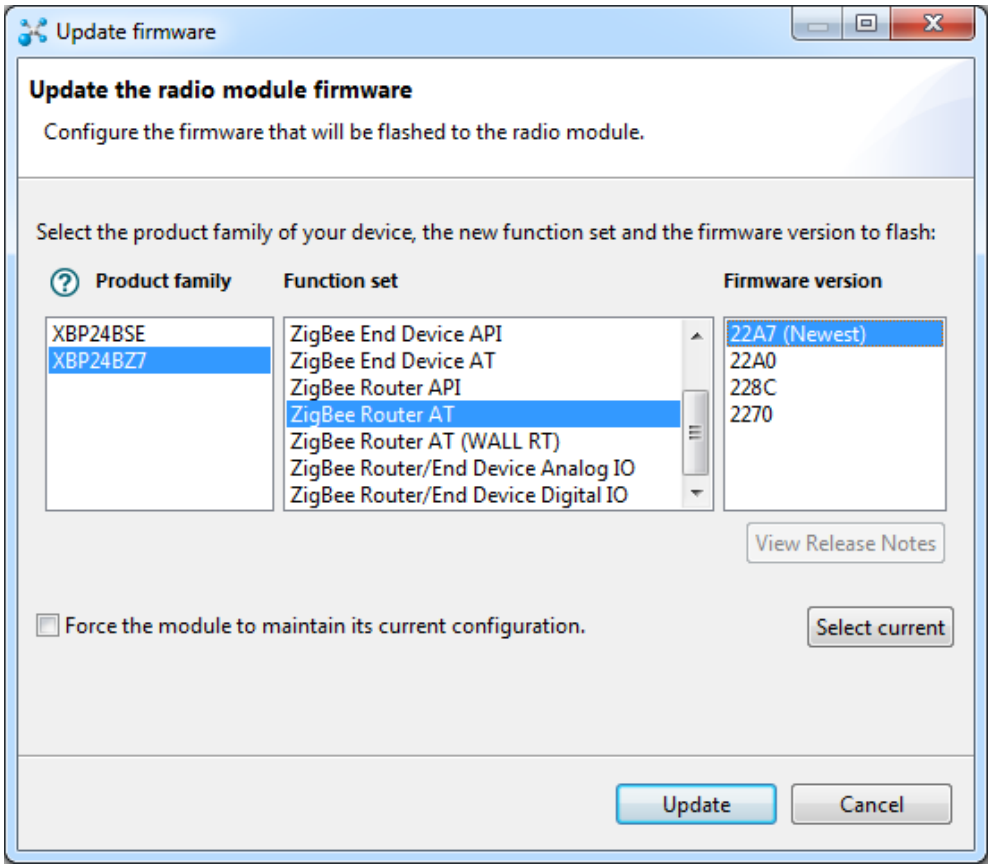

Figure 3: ZigBee Router AT firmware

Router's parameters should be as following:

- **ID** (PAN ID): **105** *(same value as on Coordinator)*
- **NI** (Node Identifier): **ONBOARD1** *(ONBOARD2, ONBOARD3 etc. for subsequent XBee modules)*
- **PL** (Power Level): **Highest** (4)
- **BD** (Baud Rate): **57600** (6)

1.6 Hardware connection

Connect XBee Coordinator to USB using USB-UART adapter mentioned above. Check out whether drivers are installed correctly and new serial port became available.

Connect XBee Router to Pixhawk TELEM1 connector. Connection diagram (in case of using DFRobot XBee USB Adapter V2):

Pixhawk TELEM1 (DF13-6)	Adapter UART (PLS-5)
1 (+5V)	1 (+5V)
2 (TX, out)	4 (RX, in)
3 (RX, in)	3 (TX, out)
6 (GND)	2 (GND)

After powering autopilot up you can use XCTU software for performing remote nodes discovery in order to make sure that wiring and settings are correct.

1.7 Configuration file

Default configuration file of the XBee Connector suits most needs and it is generally not necessary to modify it.

Configuration file location:

- **On Microsoft Windows:**

```
C:\Program Files (x86)\UgCS\bin\vsm-xbee.conf
```

- **On GNU/Linux:**

```
/etc/opt/ugcs/vsm-xbee.conf
```

- **On Apple OS X:**

```
/Users/[user name]/Library/Application Support/UGCS/configuration/vsm-xbee.conf
```

1.7.1 Serial port configuration

Typically xBee is connected to UgCS via serial port. See Serial port configuration for details.

- **Example:**

```
connection.serial.1.name = COM21
connection.serial.1.baud = 57600
```

1.7.2 Common parameters

XBee Connector shares a common set of configuration file parameters with VSMs which are described in Common configuration file parameters. XBee Connector configuration file prefix is:

```
xbee
```

1.7.3 TCP port number for incoming connections from VSMs

Mandatory.

- **Name:** xbee.tcp.[number].port = [port number]
- **Description:** This port number is used to establish TCP connections between XBee Connector and VSMs. It should have the same value as `vehicle.[VSM].tcp.[number].port` parameter of particular VSM configuration.
- **Example:**

```
xbee.tcp.1.port = 5566
```

1.8 Common configuration file parameters

VSM configuration file is a text file specified via command line argument - *-config* of the VSM application. Example:

```
--config /etc/opt/ugcs/vsm-ardupilot.conf
```

Each configuration parameter is defined as a line in the configuration file with the following structure:

```
name1.name2....nameX = value
```

where name1, name2 ... nameX are arbitrary names separated by dots to divide a variable into logical blocks and a value which can be a number value or a text string depending on the context. See below the description about common VSM configuration parameters.

1.8.1 UgCS server configuration

VSM can connect to UgCS in two different ways:

- Listen for connection form the UgCS server. See Listening address and Listening port.

  When VSM is configured in listening mode automatic VSM discovery can be set up, too. See Automatic service discovery

- Initiate connection to UgCS server. See UgCS server address and UgCS server port.

At least one of the above must be configured for VSM to work.

1.8.1.1 Listening address

Optional.

- **Name:** ucs.local_listening_address = [IP address]
- **Description:** Local address to listen for incoming connections from UgCS server.
- **Default:** 0.0.0.0 (listen on all local addresses)
- **Example:** ucs.local_listening_address = 10.0.0.2

1.8.1.2 Listening port

Optional.

- **Name:** ucs.local_listening_port = [port number]
- **Description:** Local TCP port to listen for incoming connections from UgCS server.
- **Example:** ucs.local_listening_port = 5556

1.8.1.3 UgCS server address

Optional.

- **Name:** ucs.address = [IP address]
- **Description:** UgCS server address to connect to.
- **Example:** ucs.address = 1.2.3.4

1.8.1.4 UgCS server port

Optional.

- **Name:** ucs.port = [port number]
- **Description:** UgCS server port.
- **Example:** ucs.port = 3335

1.8.1.5 Retry timeout

Optional.

- **Name:** ucs.retry_timeout = [seconds]
- **Description:** Retry timeout for outgoing server connections in seconds.
- **Default:** 10
- **Example:** retry_timeout = 11

1.8.2 Automatic service discovery

VSM can respond to automatic service discovery requests form UgCS server.

When this parameter is not configured, service discovery is disabled.

Optional.

- **Name:** service_discovery.vsm_name = [Service name]
- **Description:** Human readable service name.
- **Example:** service_discovery.vsm_name = Ardupilot VSM

1.8.3 Logging configuration

1.8.3.1 Level

Optional.

- **Name:** log.level = [error|warning|info|debug]
- **Description:** Logging level.
- **Default:** info
- **Example:** log.level = debug

1.8.3.2 File path

Optional.

- **Name:** log.file_path = [path to a file]
- **Description:** Absolute or relative (to the current directory) path to a logging file. Logging is disabled if logging file is not defined. File should be writable. Backslash should be escaped with a backslash.
- **Example:** log.file = /var/opt/ugcs/log/vsm-ardupilot/vsm-ardupilot.log
- **Example:** log.file = C:\\Users\\John\\AppData\\Local\\UGCS\\logs\\vsm-ardupilot\\vsm-ardupilot.log

1.8.3.3 Maximum single file size

Optional.

- **Name:** log.single_max_size = [size]
- **Description:** Maximum size of a single log file. When maximum size is exceeded, existing file is renamed by adding a time stamp and logging is continued into the empty file. [size] should be defined as a number postfixed by a case insensitive multiplier:
  - Gb, G, Gbyte, Gbytes: for Giga-bytes
  - Mb, M, Mbyte, Mbytes: for Mega-bytes
  - Kb, K, Kbyte, Kbytes: for Kilo-bytes
  - no postfix: for bytes
- **Default:** 100 Mb
- **Example:** log.single_max_size = 500 Mb

1.8.3.4 Maximum number of old log files

Optional.

- **Name:** log.max_file_count = [number]
- **Description:** Log rotation feature. Maximum number of old log files to keep. After reaching single_max_size of current log file VSM will rename it with current time in extension and start new one. VSM will delete older logs so the number of old logs does not exceed the max_file_count.
- **Default:** 1
- **Example:** log.max_file_count = 5
  -

1.8.4 Mission dump path

Optional.

- **Name:** [prefix].mission_dump_path = [path to a file]
- **Description:** File to dump all generated missions to. Timestamp is appended to the name. Delete the entry to disable mission dumping. All directories in the path to a file should be already created.
- **Example:** vehicle.ardupilot.mission_dump_path = C:\\tmp\\ardupilot_dump

1.8.5 Command execution control

When vehicle is connected via unreliable link VSM will retry each command several times before failing. This section describes the parameters which control the command execution.

1.8.5.1 Command try count

- **Name:** vechicle.command_try_count = <number of="" times>=""></number>
- **Description:** Number of times the command will be issued before declaring it as failed. Must be greater than zero.
- **Default:** 3
- **Example:** vechicle.command_try_count = 5

1.8.5.2 Command timeout

- **Name:** vechicle.command_timeout = <timeout in="" seconds>=""></timeout>
- **Description:** Time to wait for response on issued command before retrying.
- **Unit:** second
- **Default:** 1

1.9 Communication with devices

VSM can communicate with Vehicle over different communication channels

Currently supported channels:

- Serial port, see Serial port configuration for details.
- Outgoing TCP, see Outgoing TCP connection configuration for details.
- Incoming TCP, see Incoming TCP connection configuration for details.
- Outgoing UDP, see Outgoing UDP connection configuration for details.
- Incoming UDP, see Incoming UDP connection configuration for details.
- Incoming UDP (any peer), see Incoming UDP connection configuration (any peer) for details.
- Named pipe , see Named pipes for details.
- vsm-proxy (XBee), see Proxy configuration for details.

1.9.1 Serial port configuration

VSM which communicates with vehicles via serial ports should define at least one serial port, otherwise VSM will not try to connect to the vehicles. Port name and baud rate should be both defined.

1.9.1.1 Port name

Required.

- **Name:** connection.serial.[index].name = [regular expression]
- **Description:** Ports which should be used to connect to the vehicles by given VSM. Port names are defined by a [regular expression] which can be used to define just a single port or create a port filtering regular expression. Expression is case insensitive on Windows. [index] is a arbitrary port indexing name.
- **Example:** connection.serial.1.name = /dev/ttyUSB[0-9]+|com[0-9]+
- **Example:** connection.serial.2.name = com42

1.9.1.2 Port baud rate

Required.

- **Name:** connection.serial.[index].baud.[baud index] = [baud]
- **Description:** Baud rate for port opening. [baud index] is an optional arbitrary name used when it is necessary to open the same serial port using multiple baud rates. [index] is an arbitrary port indexing name.
- **Example:** connection.serial.1.baud.1 = 9600
- **Example:** connection.serial.1.baud.2 = 57600
- **Example:** connection.serial.2.baud = 38400

1.9.1.3 Excluded port name

Optional.

- **Name:** connection.serial.exclude.[exclude index] = [regular expression]
- **Description:** Ports which should not be used for vehicle access by this VSM. Port names are defined by a [regular expression] which can be used to define just a single port or create a port filtering regular expression. Filter is case insensitive on Windows. [exclude index] is a arbitrary indexing name used when more than one exclude names are defined.
- **Example:** connection.serial.exclude.1 = /dev/ttyS.∗
- **Example:** connection.serial.exclude = com1

1.9.1.4 Serial port arbiter

Optional.

- **Name:** connection.serial.use_arbiter = [yes|no]
- **Description:** Enable (yes) or disable (no) serial port access arbitration between VSMs running on the same machine. It is recommended to have it enabled to avoid situation when multiple VSMs try to open the same port simultaneously.
- **Default:** yes
- **Example:** connection.serial.use_arbiter = no

1.9.2 Outgoing TCP connection configuration

VSM can be configured to connect to the vehicle via TCP. VSM will try to establish connection to the specified address:port.

Used to connect to vehicle simulator or when vehicle is equipped with WiFi adapter.

1.9.2.1 Remote TCP port

Required.

- **Name:** connection.tcp_out.[index].port = [port number]
- **Description:** Remote port to connect to.
- **Example:** connection.tcp_out.1.port = 5762

1.9.2.2 IP-address for outgoing TCP connection

Required.

- **Name:** connection.tcp_out.[index].address = [IP-address]
- **Description:** IP-address of vehicle to connect to.
- **Example:** connection.tcp_out.1.address = 10.0.0.111

1.9.3 Incoming TCP connection configuration

VSM can be configured to listen for incoming TCP connections from the vehicle. Multiple vehicles are supported on the same port.

Used to connect to vehicle equipped with WiFi adapter.

1.9.3.1 Local listening TCP port

Required.

- **Name:** connection.tcp_in.[index].local_port = [port number]
- **Description:** Remote port to connect to.
- **Example:** connection.tcp_in.1.local_port = 5762

1.9.3.2 Local IP address

Optional.

- **Name:** connection.tcp_in.[index].local_address = [IP-address]
- **Description:** Local ip address to bind to.
- **Default:** 0.0.0.0 (all interfaces)
- **Example:** connection.tcp_in.1.local_address = 127.0.0.1

1.9.4 Outgoing UDP connection configuration

VSM can be configured to connect to the vehicle via UDP. VSM will try to establish UDP connection to the specified address:port.

1.9.4.1 Remote IP-address for UDP

Required.

- **Name:** connection.udp_out.[index].address = [IP-address]
- **Description:** Remote IP-address to send outgoing UDP packets to.
- **Example:** connection.udp_out.1.address = 192.168.1.1

1.9.4.2 Remote UDP port

Required.

- **Name:** connection.udp_out.[index].port = [port number]
- **Description:** Remote UDP port to send outgoing packets to.
- **Example:** connection.udp_out.1.port = 14551

1.9.4.3 Local IP-address for UDP

Optional.

- **Name:** connection.udp_out.[index].local_address = [IP-address]
- **Description:** Local ip address to bind to.
- **Default:** 0.0.0.0 (bind to all interfaces)
- **Example:** connection.udp_out.1.local_address = 0.0.0.0

1.9.4.4 Local UDP port

Optional.

- **Name:** connection.udp_out.[index].local_port = [port number]
- **Description:** Local UDP port to listen for incoming packets on.
- **Default:** 0 (bind to random port)
- **Example:** connection.udp_out.1.local_port = 14550

1.9.5 Incoming UDP connection configuration

VSM can be configured to listen for UDP conncections from the vehicle. Vehicle must be actively sending heartbeat/telemetry on specified UDP port before it can be detected by VSM. VSM will automatically detect multiple vehicles on the same port. This is very useful for "drone swarm" setups as there is no need to specify connector for each vehicle and no need to know the IP address of each vehicle in advance.

1.9.5.1 Local UDP port

Required.

- **Name:** connection.udp_in.[index].local_port = [port number]
- **Description:** Local UDP port to listen for incoming packets on.
- **Example:** connection.udp_in.1.local_port = 14550

1.9.5.2 Local IP-address for UDP

Optional.

- **Name:** connection.udp_in.[index].local_address = [IP-address]
- **Description:** Local ip address to bind to.
- **Default:** 0.0.0.0 (bind to all interfaces)
- **Example:** connection.udp_in.1.local_address = 0.0.0.0

1.9.6 Incoming UDP connection configuration (any peer)

This connection type is similar to "udp_in" with the exception that all incoming traffic will be recevied as one stream. It is used for special purpose connections and cannot be used to connect vehicles.

1.9.6.1 Local UDP port

Required.

- **Name:** connection.udp_any.[index].local_port = [port number]
- **Description:** Local UDP port to listen for incoming packets on.
- **Example:** connection.udp_any.1.local_port = 14550

1.9.6.2 Local IP-address for UDP

Optional.

- **Name:** connection.udp_any.[index].local_address = [IP-address]
- **Description:** Local ip address to bind to.
- **Default:** 0.0.0.0 (bind to all interfaces)
- **Example:** connection.udp_any.1.local_address = 0.0.0.0

1.9.7 Named pipes

VSM is able to communicate with vehicle over named pipe. The pipe must already exist.

- **Name:** connection.pipe.[index].port = pipe_name
- **Description:** Pipe name
- **Example:** connection.pipe.1.name = \\pipe\my_named_pipe

1.9.8 Proxy configuration

VSM is able to communicate with vehicle via proxy service which redirects dataflow received from vehicle through TCP connection to VSM and vice versa using specific protocol. In other words proxy service appears as a router between vehicle and VSM. At the moment there is one implementation of proxy in UgCS called XBee Connector which retranslates data from ZigBee network to respective VSM.

1.9.8.1 IP-address for proxy

Required.

- **Name:** connection.proxy.[index].address = [IP-address]
- **Description:** IP-address to connect proxy to. Specify local or remote address.
- **Example:** connection.proxy.1.address = 127.0.0.1

1.9.8.2 TCP port for proxy

Required.

- **Name:** connection.proxy.[index].port = [port number]
- **Description:** TCP port to be connected with proxy through. Should be the same as in congiguration on proxy side.
- **Example:** connection.proxy.1.port = 5566

2 Disclaimer

DISCLAIMER OF WARRANTIES AND LIMITATIONS ON LIABILITY.

(a) SMART PROJECTS HOLDINGS MAKE NO REPRESENTATIONS OR WARRANTIES REGARDING THE ACCURACY OR COMPLETENESS OF ANY CONTENT OR FUNCTIONALITY OF THE PRODUCT AND ITS DOCUMENTATION.

(b) SMART PROJECTS HOLDINGS DISCLAIM ALL WARRANTIES IN CONNECTION WITH THE PRODUCT, AND WILL NOT BE LIABLE FOR ANY DAMAGE OR LOSS RESULTING FROM YOUR USE OF THE PRODUCT. INCLUDING BUT NOT LIMITED TO INJURY OR DEATH OF USER OR ANY THIRD PERSONS OR DAMAGE TO PROPERTY.

(c) THE SOFTWARE IS SUPPLIED AS IS WITH NO WARRANTIES AND CAN BE USED ONLY AT USERS OWN RISK.