LoRaWAN Troubleshooting guide

Conduit AEP v1.4.16 / mLinux v3.3.23
LoRa Network Server – v2.0.19

Table of Contents

1 Network Components

1.1 End-devices

The end-devices in a LoRaWAN network are used to send small infrequent packets from remote sensors or other devices. The low datarate and high latency are designed for low throughput applications.

Multitech mDot and xDot or 3rd party LoRaWAN modules can connect to the network server through a Multitech Conduit gateway. As long as the devices adhere to the LoRaWAN protocol they should be inter-operable.

1.2 Gateways

A LoRaWAN gateway listens for packets on a set of configured channels, received uplink packets will be forwarded to a network server. When the network server send a downlink packet to a device it will queue a packet to be transmitted by the gateway in a receive window.

Multitech Conduit or Conduit AP can be used to communicate with the network server on Conduit or a a 3rd party network server. The network server on Conduit requires a gateway to use a Semtech packet forwarder.

3rd party LoRaWAN gateways have not been tested for use with the network server on Conduit. If a Semtech packet forwarder is installed on the gateway it should work.

A single gateway card can be configured to listen on 10 channels. 8 Channels are multi-sf channels and can receive any 125 kHz datarate on the configured frequencies. The other 2 channels use specific datarates, one is for FSK packets and the other can be configured for a set LoRa bandwidth and spreading factor.

1.3 Network Servers

A network server maintains end-device session information included end-device address, packet counters and session keys. The address, counters and keys are used to authenticate received packets to be forwarded to an application.

The network server is responsible for de-duplicating packets received from multiple gateways or retransmitted by the end-device. An application will expect to see a single frame counter from the network server only once for each end-device session.

The Multitech network server on Conduit is proprietary software licensed only for use on Multitech products.

3rd party network servers may have a packet forwarder that will need to be installed on the gateway.

1.4 Join Server

The join server is responsible for creating the session keys. It will derive unique session keys using a

nonce value determined by the end-device and an application nonce created at the join server. The nonce value used by the join server is returned to the end-device in an encrypted packet. Only the end-device with the pre-shared key can decrypt the payload and create the session keys used to send and receive packets during the session.

1.5 Applications

An application connects to a network server expecting to receive packets from end-device connected to the network server. When the application needs to send a downlink to the end-device it will queue a packet with the network server. If a Class A end-device has just sent an uplink and the receive windows are pending the packet can be sent to the end-device, otherwise the packet will wait in the queue until the next uplink.

The network server is responsible for de-duplicating packets received from multiple gateways or retransmitted by the end-device. An application will expect to see a single frame counter from the network server only once for each end-device session.

2 Logging

Logs are available detailing the inner working of the code. End-device state and information about packets received and sent are available from the logs. Logging has several levels of detail available. Setting the log level to TRACE or MAXIMUM will give the most information.

Levels available:

OFF, FATAL, ERROR, WARNING, INFO, DEBUG, TRACE

NOTE: *Logs are held in a RAM disk. If custom logs are created be sure to configure logrotate to manage their size. Otherwise logging can run the system out of memory. Rotation is configured for any log file in /var/log/ that starts with "lora-". Logs are lost on reboot.*

2.1 Dot Log Settings

The AT+LOG command controls the details of the logging output on the debug serial interface.

Setting to 0 will disable all logging, 3 will enable some informational messages about the state of the dot such as joining, sending and receiving packets. Setting to 6 will enable the highest level of detail.

For production it is recommended to disable logging. Enabled logging will affect performance of the Dots and battery life even when there is no receiver for the output.

2.1.1 Example Log

```
[TRACE] Initiating join...
[TRACE] Join Network - OTA
[INFO] Disabling auto sleep mode
```

The join request will expect a join accept message either Rx window opened with the following settings. The end-device will wait RxDelay seconds after end of Tx before opening the first Rx window

using Rx1Offset on the datarate. If join accept is not received in the first window it will open the second Rx window using Rx2Freq and Rx2Dr.

```
[INFO] Send join request RxDelay: 5 Rx1Offset: 0 Rx2Freq: 923300000 Rx2Dr: 8
[TRACE] DevEUI: 008000000400706f
[TRACE] AppEUI: 16ea76f6ab663d80
[TRACE] AppKey: abea38f5de2ab839e399f238a071ed1a
[TRACE] Sending 23 bytes
```

A random channel will be chosen that supports the current datarate. If duty-cycle limits are required the channels available may be reduced.

```
[TRACE] Number of available channels: 8
[DEBUG] Using channel 7 : 903700000
```

The radio datarate settings are configured again after setting the frequency.

```
[INFO] Configure radio for TX
```

The radio Tx Power is computed using the maximum allowed by the channel plan. The current Dot hardware can output a maximum conducted power of +19 dBm. Antenna gain is added to give the total power.

```
[DEBUG] Session pwr: 30 ant: 3 max: 26
[DEBUG] Radio Power index: 20 output: 19 total: 22
```

The complete radio Tx settings are given here. Tx power, datarate (0-15), spreading factor (7-12), bandwidth (0:125,1:250,2:500), coderate (1:4/5), preamble length, CRC, and IQ settings.

```
[DEBUG] TX PWR: 20 DR: 0 SF: 10 BW: 0 CR: 1 PL: 8 CRC: 1 IQ: 0
[DEBUG] mDotEvent - TxStart
[DEBUG] mDotEvent - TxDone
[TRACE] Event: OK
[TRACE] Flags Tx: 1 Rx: 0 RxData: 0 RxSlot: 0 LinkCheck: 0 JoinAccept: 0
[TRACE] Info: Status: 0 ACK: 0 Retries: 0 TxDR: 0 RxPort: 0 RxSize: 0 RSSI: 0 SNR: 0 Energy:
0 Margin: 0 Gateways: 0
[DEBUG] RadioEvent – TxDone
```

Once transmission is complete the first Rx window is opened after waiting RxDelay seconds.

```
[TRACE] RX1 on freq: 927500000
[TRACE] RX DR: 10 SF: 10 BW: 2 CR: 1 PL: 8 STO: 12 CRC: 0 IQ: 1
```

Total number of uplink packets sent, downlink packets received, retried confirmed packets, and packets received with CRC errors.

```
[TRACE] Stats: Up: 53080 Down: 49779 DupTx: 0 CRC Errors: 43
```

Rx Window has completed and a packet was received, the RSSI in dB and SNR in cB from the radio are displayed. Divide the SNR value by 10 for a value in dB. If the SNR value is less than 0 then the RSSI is reduced by the SNR value in dB.

```
[INFO] Rx Window 1
```

```
[INFO] RxDone 17 bytes RSSI: -29 dB SNR: 67 cB
```

The received packet is displayed, the first byte is a packet type and the last 4 bytes are the MIC.

```
[TRACE] Payload: 20b6eeda12045d1a90bef076f6d5270e81
[INFO] Network joined
[DEBUG] Computing session keys...
```

The join accept message contains downlink settings the value is pulled from the buffer after being decrypted.

```
[DEBUG] DL Settings: 0x08 0x08
[DEBUG] RxOffset: 0 Rx2Index: 8
```

Session keys are generated from the nonce values of the Join Request and Join Accept messages.

```
[TRACE] NETWSKEY: 9a4d73b2a7d152c937a7250f6def2c0f
[TRACE] DATASKEY: 084f12e7086e11b0e5593f513c8a900b
```

The LoRaWAN NetID and DevAddr were sent in the Join Accept message

```
[TRACE] NetID: 0x000000
[TRACE] DevAddr: 0x012ACAA8
```

Rx Delay, Rx 1 offset and Rx2 datarate index to be used in future Rx windows are displayed.

```
[TRACE] Rx1DatarateOffset: 0 Rx2Datarate: 8 rxDelay: 1
[DEBUG] mDotEvent - JoinAccept
[TRACE] Event: OK
[TRACE] Flags Tx: 0 Rx: 0 RxData: 0 RxSlot: 0 LinkCheck: 0 JoinAccept: 1
[TRACE] Info: Status: 0 ACK: 0 Retries: 0 TxDR: 0 RxPort: 0 RxSize: 0 RSSI: 0 SNR: 0 Energy:
0 Margin: 0 Gateways: 0
[DEBUG] RadioEvent - JoinAccept
[INFO] Packet RSSI: -29 dB SNR: 67 cB
[DEBUG] mDotEvent - RxDone
[DEBUG] RadioEvent – RxDone
```

An AT command is entered to send an uplink packet. The same steps are taken for tranmission, the new Rx settings from the Join Accept message will be used.

```
[DEBUG] Send with tx timeout 20000
[INFO] Preparing frame
[TRACE] Adding MIC to frame
[TRACE] Sending 12 bytes
[TRACE] Number of available channels: 8
[DEBUG] Using channel 7 : 903700000
[INFO] Configure radio for TX
[DEBUG] Session pwr: 30 ant: 3 max: 21
[DEBUG] Radio Power index: 20 output: 19 total: 22
```

```
[DEBUG] TX PWR: 20 DR: 0 SF: 10 BW: 0 CR: 1 PL: 8 CRC: 1 IQ: 0
[INFO] Configure radio for TX
[DEBUG] Session pwr: 30 ant: 3 max: 21
[DEBUG] Radio Power index: 20 output: 19 total: 22
[DEBUG] TX PWR: 20 DR: 0 SF: 10 BW: 0 CR: 1 PL: 8 CRC: 1 IQ: 0
[DEBUG] mDotEvent - TxStart
[DEBUG] mDotEvent - TxDone
[TRACE] Event: OK
[TRACE] Flags Tx: 1 Rx: 0 RxData: 0 RxSlot: 0 LinkCheck: 0 JoinAccept: 0
[TRACE] Info: Status: 0 ACK: 0 Retries: 0 TxDR: 0 RxPort: 0 RxSize: 0 RSSI: 0 SNR: 0 Energy:
0 Margin: 0 Gateways: 0
[DEBUG] RadioEvent - TxDone
[TRACE] RX1 on freq: 927500000
[TRACE] RX DR: 10 SF: 10 BW: 2 CR: 1 PL: 8 STO: 12 CRC: 0 IQ: 1
[TRACE] Stats: Up: 53081 Down: 49780 DupTx: 0 CRC Errors: 43
[INFO] Rx Window 1
[INFO] RxDone 28 bytes RSSI: -27 dB SNR: 72 cB
```

The payload now has a header with DevAddr and FCnt values.

```
[TRACE] Payload: a0a8ca2a01200000007b6f0817cd9bd9615f6128341f80ad904dcfba
[INFO] Packet for 012acaa8
```

The FCnt value is used to authenticate the packet using the Network Session Key

```
[TRACE] Check downlink counter 00000000
[TRACE] Ack received
```

The packet was received on App Port 0, the payload will contain MAC commands.

```
[INFO] Packet Received : Port: 0 FCnt: 00000000 Size: 15 ACK: 1 DUP: 0
```

MAC commands found in the packet are processed. The first downlink will contain MAC commands to setup addition channels in EU,IN,AS or the supported channel mask in US and AU.

```
[DEBUG] mac commands in payload
[TRACE] CMDS: 03000f00710300ffff010300ffff11
[TRACE] Mac start processing ADR commands
[DEBUG] Mac command index: 0 cmd: 0x03
[DEBUG] ADR DR: 0 PWR: 0 Ctrl: 07 Mask: 000f NbRep: 1 Stat: 07
[TRACE] Mac processing ADR commands - partial status 0x07, last command status 0x07
[DEBUG] Mac command index: 5 cmd: 0x03
[DEBUG] ADR DR: 0 PWR: 0 Ctrl: 00 Mask: ffff NbRep: 1 Stat: 07
[TRACE] Mac processing ADR commands - partial status 0x07, last command status 0x07
[DEBUG] Mac command index: 10 cmd: 0x03
[DEBUG] ADR DR: 0 PWR: 0 Ctrl: 01 Mask: ffff NbRep: 1 Stat: 07
```

```
[TRACE] Mac finished processing 3 ADR commands - partial status 0x07

[TRACE] Mac validated 3 ADR commands - validation status 0x07

[TRACE] Mac 3 ADR commands - final status 0x07

[DEBUG] mDotEvent - PacketRx ADDR: 012acaa8

[TRACE] Payload: 03000f00710300ffff010300ffff11

[TRACE] Event: OK

[TRACE] Flags Tx: 0 Rx: 1 RxData: 1 RxSlot: 1 LinkCheck: 0 JoinAccept: 0

[TRACE] Info: Status: 0 ACK: 1 Retries: 1 TxDR: 0 RxPort: 0 RxSize: 15 RSSI: -27 SNR: 72
Energy: 0 Margin: 0 Gateways: 0

[INFO] Packet RSSI: -27 dB SNR: 72 cB

[DEBUG] mDotEvent - RxDone

[DEBUG] RadioEvent – RxDone
```

There was no application payload in the packet, the mDot library marks as error from call to Recv.

```
[ERROR] No packet received
```

2.2 Conduit Log Settings

The logs on Conduit AEP default by outputting to syslog /var/log/messages file. This file is rotated and contains messages from other processes running on Conduit.

The logging can be configured to output to a separate file /var/log/lora-network-server.log is the default log path. This may be easier for debugging as there are only messages specific to the network server present.

It is recommended to set the logging back to syslog for production deployments.

2.2.1 Example Log

A frame is received by the network server from a gateway. The gateway ID is shown in the middle of each line, following “GW:”. A message type is given on each line to aid log filtering.

```
13:54:30:507|INFO| GW:00-80-00-00-00-00-00-87|FRAME-RX|Parsing 1 packets

13:54:30:509|DEBUG| GW:00-80-00-00-00-00-00-87|FRAME-RX|DATA:
00803d66abf676ea166f70000400008000b89bf9615db6

13:54:30:510|DEBUG| GW:00-80-00-00-00-00-00-87|FRAME-RX|FREQ: 911.700000 MHz DR0 RSSI: -122 dB SNR: -90
cB
```

The packet is a join request, requests will first be processed by the local Join Server before being forwarded to a cloud Join Server if enabled. The local Join Server can match a Join Request comparing the the Join EUI against the NetworkID setting. If this matches the NetworkKey will be tried to authenitcate the Join Request. If this check fails then cloud Join Server will be forwarded the request.

```
13:54:30:511|DEBUG| GW:00-80-00-00-00-00-00-87|FRAME-RX|TYPE: Join Request

13:54:30:513|DEBUG| GW:00-80-00-00-00-00-00-87|JOINREQ-RX|00803d66abf676ea166f70000400008000b89b

13:54:30:514|DEBUG| GW:00-80-00-00-00-00-00-87|DEV-EUI|00-80-00-00-04-00-70-6f

13:54:30:514|DEBUG| GW:00-80-00-00-00-00-00-87|JOIN-EUI|16-ea-76-f6-ab-66-3d-80
```

```
13:54:30:514|DEBUG| GW:00-80-00-00-00-00-00-87|DEV-NONCE|9bb8
```

The request is sent the the configured Join Server after failing to authenticate with the local Join Server.

```
13:54:30:516|DEBUG| ED:00-80-00-00-04-00-70-6f|FWD-JOIN|Sending request to Join Server
```

This request is rejected by the cloud Join Server with a “Gatway Mismatch” response. The end-device is not allowed to join the Conduit that received this packet.

```
13:54:30:954|TRACE| POST code: 200 response:[{"AppSKey":
{"AESKey":"*","KeyLabel":"AppSKey"},"Lifetime":1,"NwkSKey":{"AESKey":"*","KeyLabel":"NwkSKey"},"PHY

Payload":"*","ProtocolVersion":"1.0","Result":{"Description":"Gateway
Mismatch","ResultCode":"JoinReqFailed"},"VSExtension":{}}

13:54:30:955|DEBUG| Join response received 443 ms

13:54:30:955|WARNING| Join Server rejected Join Request
```

The frame as reported by the packet forwarder is output.

```
13:54:30:958|DEBUG| GW:00-80-00-00-00-00-00-87|FRAME-RX|JSON:
{"tmst":16678436,"chan":7,"rfch":1,"freq":911.7,"stat":1,"modu":"LORA","datr":"SF10BW125","codr":"4/5",
"lsnr":-9.0,"rssi":-122,"size":23,"data":"AIA9Zqv2duoWb3AABAAAgAC4m/lhXbY="}
```

Another packet is received. This one cannot be processed because of an error detected by CRC. Random packets may be received with CRC error, they may be caused by environmental noise or faint LoRa packets nearly out of range.

```
13:58:8:637|INFO| GW:00-80-00-00-00-00-00-87|FRAME-RX|Parsing 1 packets

13:58:8:638|WARNING| GW:00-80-00-00-00-00-00-87|FRAME-RX|CRC-ERROR
```

Another packet is received. This one is another Join Request.

```
14:1:33:337|INFO| GW:00-80-00-00-00-00-00-87|FRAME-RX|Parsing 1 packets

14:1:33:340|DEBUG| GW:00-80-00-00-00-00-00-87|FRAME-RX|DATA:
00803d66abf676ea166f70000400008000821ac7eec7c3

14:1:33:341|DEBUG| GW:00-80-00-00-00-00-00-87|FRAME-RX|FREQ: 910.500000 MHz DR0 RSSI: -123 dB SNR: -92
cB

14:1:33:343|DEBUG| GW:00-80-00-00-00-00-00-87|FRAME-RX|TYPE: Join Request

14:1:33:345|DEBUG| GW:00-80-00-00-00-00-00-87|JOINREQ-RX|00803d66abf676ea166f70000400008000821a

14:1:33:346|DEBUG| GW:00-80-00-00-00-00-00-87|DEV-EUI|00-80-00-00-04-00-70-6f
```

The Join Request Join EUI matches the NetworkID.

```
14:1:33:346|DEBUG| GW:00-80-00-00-00-00-00-87|JOIN-EUI|16-ea-76-f6-ab-66-3d-80

14:1:33:347|DEBUG| GW:00-80-00-00-00-00-00-87|DEV-NONCE|1a82
```

This Join Request is authenticated successfully using the NetworkID and NetworkKey.

The end-device DevEUI is output on each line of this request and response.

```
14:1:33:348|DEBUG| ED:00-80-00-00-04-00-70-6f|CHECK-KEY|MIC Valid

14:1:33:349|DEBUG| ED:00-80-00-00-04-00-70-6f|APP-NONCE|3845b0
```

This end-device has joined before, a DevAddr was found in the database.

```
14:1:33:363|INFO| ED:00-80-00-00-04-00-70-6f|DEV-ADDR|Found End Device in DB 12acaa8

14:1:33:376|INFO| ED:00-80-00-00-04-00-70-6f|QUEUE-TX|JOIN SIZE: 17

14:1:33:376|INFO| ED:00-80-00-00-04-00-70-6f|JOINREQ-OK|JOINS: 5069 UP: 125003

14:1:33:389|DEBUG| ED:00-80-00-00-04-00-70-6f|PACKET-RX|GW:00-80-00-00-00-00-00-87 Time_us:255549580
```

The Join Accept packet is queued to be sent by the radio.

```
14:1:33:393|DEBUG| ED:00-80-00-00-04-00-70-6f|PACKET-RX|Downlink Packets Queued: 1

14:1:33:394|DEBUG| ReceivedFrame ID: 0
```

The time-on-air is checked against the available duty-cycle if applicable in the configured channel plan.

```
14:1:33:413|DEBUG| getTimeOnAirMs SF: 10 BW: 500 PL: 8 SZ: 16 TOA: 82

14:1:33:413|INFO| ED:00-80-00-00-04-00-70-6f|JOIN-ACCEPT|DevAddr: 012ACAA8
```

The packet is scheduled to be sent in the first Rx window.

```
14:1:33:414|INFO| ED:00-80-00-00-04-00-70-6f|SCHED-TX|Use RX1 TOA:132 ms
```

The frame as reported by the packet forwarder is output.

```
14:1:33:431|DEBUG| GW:00-80-00-00-00-00-00-87|FRAME-RX|JSON:
{"tmst":255549580,"chan":1,"rfch":0,"freq":910.5,"stat":1,"modu":"LORA","datr":"SF10BW125","codr":"4/5"
,"lsnr":-9.2,"rssi":-123,"size":23,"data":"AIA9Zqv2duoWb3AABAAAgACCGsfux8M="}
```

When the scheduled packet time is nearly expired the packet will be sent to the radio. The frame waits at the network server until the last moment, this allows the application to queue a payload in response to the uplink for normal packets. In this case the Join Accept message sent 5 seconds after the Join Request was received. Comparing the timestamps, 14:1:33:431 and 14:1:38:120, verifies the RxDelay setting.

```
14:1:38:120|INFO| GW:00-80-00-00-00-00-00-87|FRAME-TX|IP: 127.0.0.1:58039 CH: LC2 DEV: 01-2a-ca-a8
FCNT: 00000000 REPEAT: 0

14:1:38:120|INFO| ED:00-80-00-00-04-00-70-6f|SCHED-TX|Q-SIZE: 1 PKT-ROOM: 242

14:1:38:121|DEBUG| GW:00-80-00-00-00-00-00-87|FRAME-TX|DATA: 2028e22bc8aa42095ea77ce4d2b11cb604
```

The 5 second RxDelay is displayed.

```
14:27:26:82|DEBUG| GW:00-80-00-00-00-00-00-87|PACKET-TX|RX1 OFFSET: 5000000
```

The packet is sent to the packet forwarder via UDP

```
14:1:38:136|DEBUG| GW:00-80-00-00-00-00-00-87|FRAME-TX|JSON: {"txpk":{"appeui":"16-ea-76-f6-ab-66-3d-
80","codr":"4/5","data":"ICjiK8iqQglep3zk0rEctgQ=","da tr":"SF10BW500","deveui":"00-80-00-00-04-00-70-
6f","freq":923.89999999999998,"gweui":"00-80-00-00-00-00-00-
87","ipol":true,"modu":"LORA","ncrc":true,"powe" :26,"rfch":0,"size":17,"tmst":260549580,"twnd":1}}
```

The UDP packet size is 300 bytes.

```
14:1:38:136|INFO| GW:00-80-00-00-00-00-00-87|UDP-TX|JSON-SIZE:300
```

ACK is received from the packet forwarder that the packet is scheduled to be transmitted.

```
14:1:38:139|INFO| GW:00-80-00-00-00-00-00-87|TX-ACK|OK
```

3 Network Communications

Before secure communications between end-device and the gateway are possible, information must be shared between the two end-points. This information cannot be shared over the air as a third party would be able to intercept the messages and have access to later communications.

Using pre-shared AES-128 keys the messages can be secured after each side is provisioned. A frame counter is used in each direction to ensure each packet is uniquely encrypted and authenticated, this keeps the packets from revealing too much information where the AES-128 may then be found through

brute force.

3.1 Join Process

Two method are available for end-devices to join the network, activation by personalization (ABP) and over-the-air activation (OTAA).

With ABP a single session is created by sharing two AES-128 keys with each side. One session key, Network Session Key, is used to authenticate using a 4-byte MIC attached to each packet, the second key, Application Session Key, is used to encrypt the application payload. The ABP session has two 32-bit counters for uplink and downlink packets. Each side should never repeat a packet with a counter value to keep the keys secured. Only 16-bits are sent with each packet so the upper 16-bits of the counter are maintained on each end.

Using OTA, sessions can be created by sharing only one key and exchanging nonce values to generate the session keys on each end. In this way many sessions can be created over the life of the end-device and state can be reset if the session is lost on either side or the counter values exceed the allowed 16k MAX_FCNT_GAP.

3.1.1 Configuration

3.1.1.1 OTAA

An OTAA end-device does not need to pre-register with the Conduit if it uses the NetworkID and Network Key or is provisioned in the configured cloud Join Server.

The DevEUI and AppKey are used to identify and authenticate the Join Request and author the Join Accept messages.

The end-device record will be created on the Conduit and the device Class will be A unless the Join Server is configured with a Class setting.

To have the end-device join and be in Class C mode the end-device record can be created before an OTA join is performed by the end-device. On the LoRaWAN > Device Configuration page click the Add New button. Enter the DevEUI, Friendly Name and select Class C. Enter the build information if desired and click Finish. Now when the end-device joins the network server will know it is a Class C device and downlinks can be sent at any time. Another option is to change the end-device record after the end-device has joined, this could be automated through application code when a “joined” event is received.

If Local Keys are used for the Join Server and unique DevEUI/AppKeys are entered in the list on the Key Management page then the Class setting associated with the keys will be used as the default Class setting.

3.1.1.2 ABP

To configure an ABP session the DevAddr, Network Session Key and Application Session Keys must be configured to match on both sides.

On a Dot use AT+NA, AT+NSK, and AT+DSK to configure the address and keys.

On the Conduit first create an end-device record on the LoRaWAN > Device Configuration page. Although the DevEUI is not needed to transmit packet between the gateway and end-device, it is still used to identify the device in the network server and when forwarding packets to the application.

Enter the DevEUI, Friendly Name, Class and build information into the dialog box and click Finish. The new end-device should appear in the list.

Go to the LoRaWAN > Device Sessions page and click the Add New button to add the session information.

Enter the same DevEUI as used to configure the device record, enter a DevAddr (4-byte HEX), enter an AppEUI that will be attached to received packets and a Join EUI, the Join EUI can be any EUI for ABP sessions it is normally sent in the OTA Join Request. Enter a NetID (3-byte HEX), and Session Keys (16-byte HEX). App Session Key must match the AT+DSK setting and Net Session Key must match the AT+NSK setting. Click Finish.

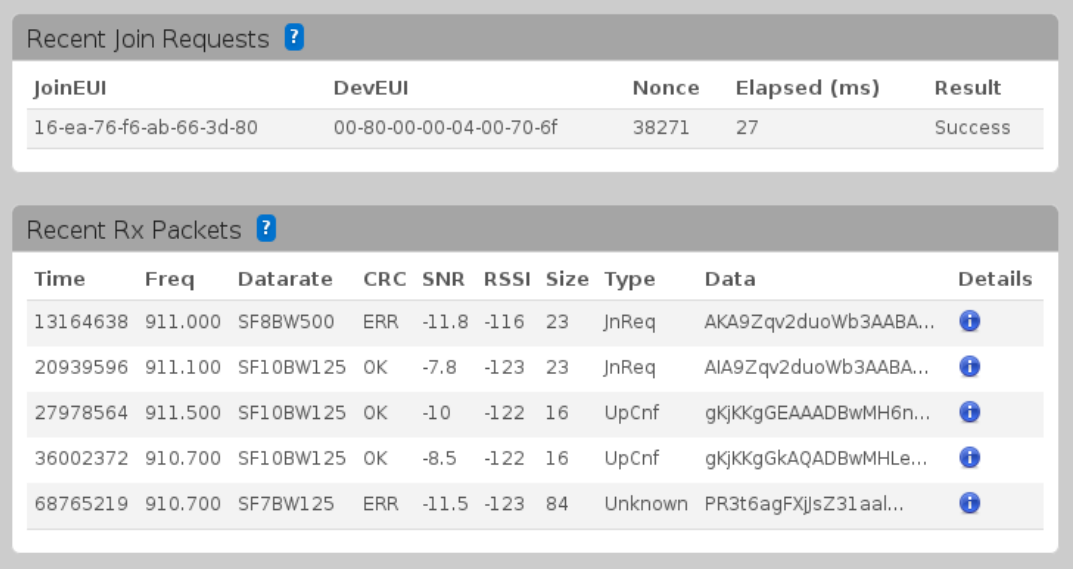
The counters will start at 0 and the device should now be able to send packets to the network server.

3.1.2 Join Request

The Join Request is sent from the end-device it initiate a LoRaWAN session. It contains a JoinEUI, DevEUI, Nonce and MIC fields.

The DevEUI is used to identify the joining end-device, the MIC is used to authenticate the end-device by generating the same MIC using the packet with the AppKey looked up by the DevEUI. If generated MIC matches the MIC received in the packet then the Join Request is accepted.

Received Join Requests can be seen on the Packets page as shown below. The list is held in memory with a maximum of 30 requests and cleared each time it is accessed. The Recent Rx Packets list is also cleared and holds a maximum of 30 packets. Failed Join Requests are not saved into the database.



Recent Join Requests

JoinEUI	DevEUI	Nonce	Elapsed (ms)	Result
16-ea-76-f6-ab-66-3d-80	00-80-00-00-04-00-70-6f	38271	27	Success

Recent Rx Packets

Time	Freq	Datarate	CRC	SNR	RSSI	Size	Type	Data	Details
13164638	911.000	SF8BW500	ERR	-11.8	-116	23	JnReq	AKA9Zqv2duoWb3AABA...	
20939596	911.100	SF10BW125	OK	-7.8	-123	23	JnReq	AIA9Zqv2duoWb3AABA...	
27978564	911.500	SF10BW125	OK	-10	-122	16	UpCnf	gKjKKgGEAAADBwMH6n...	
36002372	910.700	SF10BW125	OK	-8.5	-122	16	UpCnf	gKjKKgGkAQADBwMHLe...	
68765219	910.700	SF7BW125	ERR	-11.5	-123	84	Unknown	PR3t6agFXjJsZ31aal...	

Illustration 1: Recent Join Requests and Rx Packets

3.1.3 Join Accept

After the Join Request is accepted a Join Accept message is created and encrypted with the AppKey and sent to the end-device. The Join Accept message contains the DevAddr to be used in for the session, an app nonce value used to generate session keys, the NetID, dowlink settings, an optional list of additional channel frequencies (CFList) and MIC. The Join Accept is 17 or 33 bytes in length depending on the inclusion of the optional CFList.

When the Join Accept message is received by the end-device it will be decrypted and the MIC will be validated using the AppKey. Then the session keys are created using the dev nonce and app nonce values.

Completed Join Request and Join Accept packets will be seen in the Packets table on the Packets page as shown below. The Type field show JnReq and JnAcc of the join packets, the packet frequency and datarate are also shown and must match the settings of the receive window used to on the end-device.

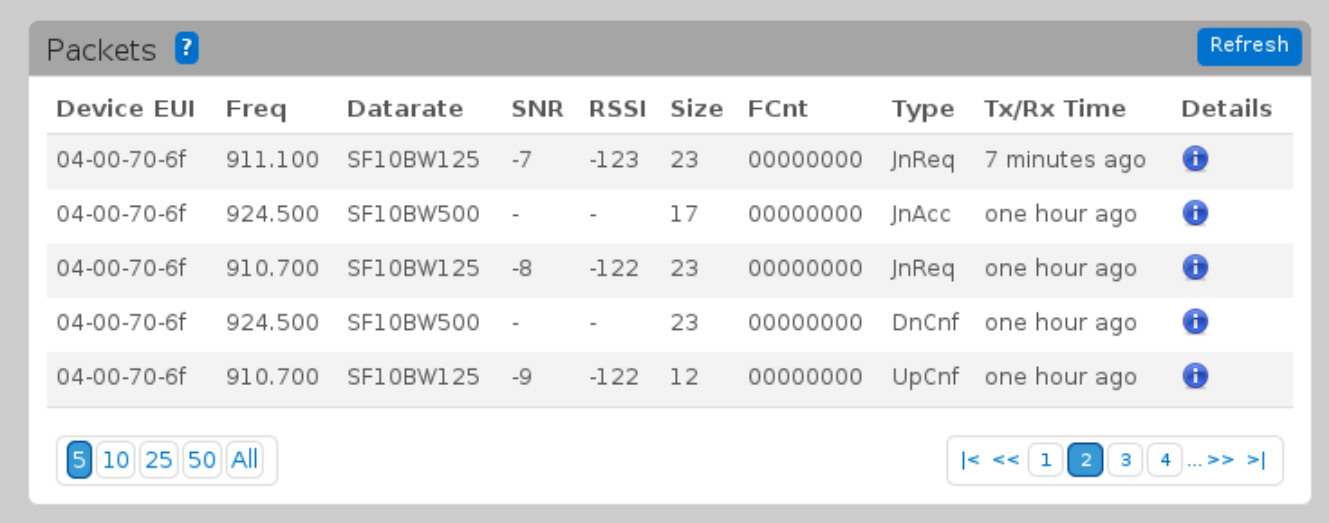


Illustration 2: Uplink and Downlink Packets Table

3.1.4 Join Receive Windows

The Join Receive windows use a default 5 second delay from the end of the uplink transmission. These values must match on both sides.

The Join Delay can be checked on the end-device using the AT+JD command. The join delay of the network server is configured on the Network Settings page.

3.2 Sending and Receiving

After the end-device has joined uplink and downlink packets can be sent. In Class A the end-device can only receive packets after an uplink is sent. This is the only time the end-device opens Rx Windows.

In Class C mode the end-device will open an Rx window using Rx2 datarate and frequency while idle.

The first downlink will contain MAC commands to setup addition channels in EU,IN,AS or the supported channel mask in US and AU. This packet is sent as a confirmed frame with App Port 0 and no application payload will be sent. The packet will be sent as a downlink until ACK is received from the end-device. This is to insure the end-device is using to correct frequencies and has all expected channels defined.

Any application payload queued in response to the first uplink will wait in the queue for the next uplink or be sent in a Class C window.

3.2.1 Sending

The end-device can send a packet at any time the the LoRaWAN ALOHA protocol network. The gateway is always listening for packets on all supported frequencies and datarates. If multiple uplinks are received at the same time the network server can schedule packets for Rx1 and Rx2 windows to respond to both end-devices if needed.

3.2.1.1 Frequencies

Enabled frequencies are controlled using channel mask. AT+CHM will display and modify the channel mask, US and AU 915 have a 80-bit channel mask for 72 usable channels and EU,AS,IN,KR have a 16-bit mask for 16 usable channels.

The channel mask can be controlled on the end-device from the network server by sending ADR MAC commands.

The list of channels can be displayed by the AT+TXCH command.

3.2.1.2 Datarates

The datarate on the Dot can be set using the AT+TXDR command this will set the default session datarate, if the datarate is modified by ADR then the current session datarate, AT+SDR, may be different. For each datarate there must be a channel defined that supports it.

The list of channels can be displayed by the AT+TXCH command.

In US915 there are 64 channels for DR0-3 and 8 channels for DR4. The R2 channel shows the Rx2 window settings.

```
at+txch
Index   Frequency  DR Max Min  On
  0     910300000      3   0    1
  1     910500000      3   0    1
  2     910700000      3   0    1
  3     910900000      3   0    1
  4     911100000      3   0    1
  5     911300000      3   0    1
  6     911500000      3   0    1
  7     911700000      3   0    1
  U     911000000      4   4    1
  R2    923300000      8   8
```

3.2.1.3 Duty-Cycle

The EU868 channel plan will limit transmissions according to duty-cycle regulations. Most channels have a 1% duty-cycle, 869.525 used for downlink in the second Rx window has a 10% duty-cycle allowed. The frequencies are separated into several bands of frequencies with different properties.

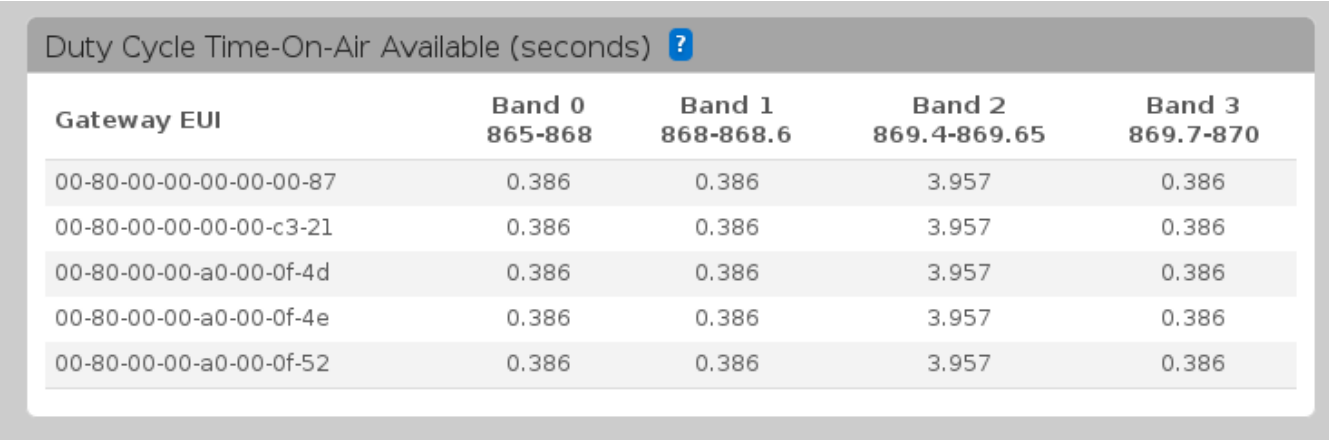
Band0: 865-870 – 1%

Band1: 868-868.6 – 1%

Band2: 869.4-869.65 – 10%

Band3: 869.7-870 – 1% unless EIRP is less than +7dBm then 100%

The gateways use a sliding window to accumulate duty-cycle in a time period. This allows the gateway to respond to end-device when needed instead of waiting after each transmission. When EU868 channel plan is selected the duty-cycle available for each gateway will be displayed on the Gateways page.



Gateway EUI	Band 0 865-868	Band 1 868-868.6	Band 2 869.4-869.65	Band 3 869.7-870
00-80-00-00-00-00-00-87	0.386	0.386	3.957	0.386
00-80-00-00-00-00-c3-21	0.386	0.386	3.957	0.386
00-80-00-00-a0-00-0f-4d	0.386	0.386	3.957	0.386
00-80-00-00-a0-00-0f-4e	0.386	0.386	3.957	0.386
00-80-00-00-a0-00-0f-52	0.386	0.386	3.957	0.386

Illustration 3: Duty-cycle per Gateway

The Dots use a time-on/time-off air duty-cycle algorithm. After each transmission on a channel it will be disabled until the time-off has expired. The time-off is the time-on-air of the transmitted packet times the inverse of the duty-cycle. Therefore a 1% channel must wait 100x longer than the last transmission before using a channel in the band again. With the lowest datarate at 2.5s time-on is possible resulting in 250s of time-off for the band.

3.2.2 Receiving

Class A end-device receive only after an uplink, Class B can received downlink periodically synchronized with a beacon from the gateways, and Class C can received downlinks at any time it is not transmitting.

3.2.2.1 Receive Windows

Receive windows are opened after an configured RxDelay seconds following the end of the uplink transmission. The window will be open for a short period depending on the symbol timeout and configured datarate. Low datarates have long symbol times resulting in longer windows. Tuning the Rx2 window to a higher datarate can then save on battery life if the lowest datarate setting is not needed in the network topography.

Receive window settings are configured on the Network Settings page. Join Delay and Rx Delay configure the time offset, Rx1 DR Offset and Rx2 Datarate are used for Rx windows after the OTA join. OTA Join windows always use the default settings according the the channel plan.

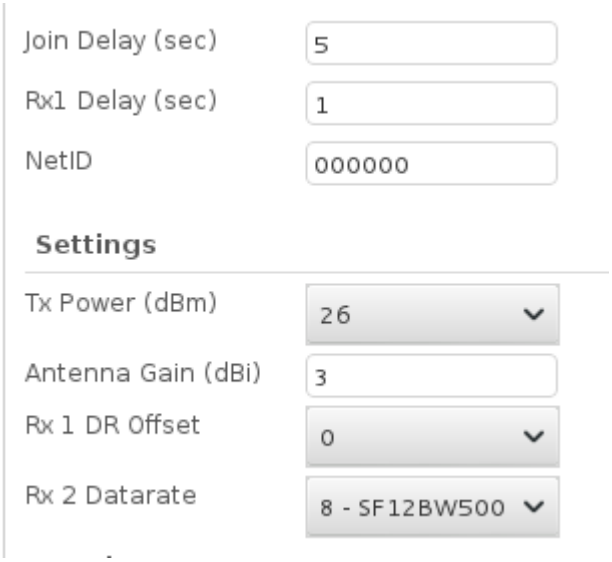


Illustration 4: Network Settings for Rx Window

3.3 Possible Issues

3.3.1 Cannot Join

1. Check for received Join Requests on LoRaWAN > Packets page
    1. Try a few join requests and refresh the page after each one
    2. If nothing shows up in the Recent Join Requests table to go step 2.
    3. If join requests appear in the table then go to step 4
2. Check Public/Private setting
3. Check Channel Plan
    1. AT+FREQ
    2. LoRaWAN > Network Settings > Channel Plan
4. Check NetworkId/NetworkKey for Local Network Settings
    1. AT+NI
    2. AT+NK
    3. LoRaWAN > Key Management
5. Check DevEUI/AppKey for Local Keys
    1. AT+DI
    2. AT+NK

3. LoRaWAN > Key Management

3.3.2 Cannot Receive Uplinks

1. Check Public/Private setting
2. Check Channel Plan
   1. AT+FREQ
   2. LoRaWAN > Network Settings > Channel Plan
3. Ensure AT+TXF=0 and AT+RXF=0
4. Check Tx settings on the Dot in logs
   - [DEBUG] Using channel 42 : 910700000
     [DEBUG] TX PWR: 20 DR: 0 SF: 10 BW: 0 CR: 1 PL: 8 CRC: 1 IQ: 0

3.3.3 Cannot Receive Downlinks

1. Ensure AT+TXF=0 and AT+RXF=0
2. LoRaWAN > Network Settings > Join Delay, Rx1 Delay Rx1 DR Offset and Rx2 Datarate
3. Check Tx settings in network server logs
   - 14:1:33:413|DEBUG| getTimeOnAirMs SF: 10 BW: 500 PL: 8 SZ: 16 TOA: 82
     14:1:38:136|DEBUG| GW:00-80-00-00-00-00-00-87|FRAME-TX|JSON: {"txpk":
     {"appeui":"16-ea-76-f6-ab-66-3d-80","codr":"4/5",
     "data":"ICjiK8iqQglep3zk0rEctgQ=","datr":"SF10BW500",
     "deveui":"00-80-00-00-04-00-70-6f","freq":923.89999999999998,
     "gweui":"00-80-00-00-00-00-00-87","ipol":true,"modu":"LORA",
     "ncrc":true,"powe" :26,"rfch":0,"size":17,"tmst":260549580,"twnd":1}}
4. Check Rx windows on the Dot in logs
   - [TRACE] RX1 on freq: 923900000
     [TRACE] RX DR: 10 SF: 10 BW: 2 CR: 1 PL: 8 STO: 12 CRC: 0 IQ: 1

3.3.4 If an End-Device resets the Uplink Counter

The following shows a log of a Joined End-Device and the network server log result of the uplink counter being reset.

18:6:23:655|INFO| GW:00-80-00-XX-XX-XX-XX-13|FRAME-RX|Parsing 1 packets

18:6:23:656|DEBUG| GW:00-80-00-XX-XX-XX-XX-13|FRAME-RX|DATA:
40ee07000000370002ed0ad2bac3b7a723ab8f43c797a4369dbd7fbdfdddc5

18:6:23:656|DEBUG| GW:00-80-00-XX-XX-XX-XX-13|FRAME-RX|FREQ: 868.300000 MHz DR4 RSSI: -31 dB SNR: 108
cB

18:6:23:657|DEBUG| GW:00-80-00-XX-XX-XX-XX-13|FRAME-RX|TYPE: Unconfirmed Up

18:6:23:657|DEBUG| GW:00-80-00-XX-XX-XX-XX-13|PACKET-RX|ADDR: 00:00:07:ee FCnt:0037

```
18:6:23:663|TRACE| AUTH KEY: 64.18ba.43.72.31.cb.46.24.96.c9.de.fe.c5.3b.b3

18:6:23:663|TRACE| PMIC: bdfdddc5 CMIC: bdfdddc5

18:6:23:663|INFO| ED:XX-XX-XX-XX-XX-XX-XX-XX|CHECK-PKT|FCNT: 00000037 LAST-FCNT: 00000036 Duplicate: no

18:6:23:664|INFO| ED:XX-XX-XX-XX-XX-XX-XX-XX|CHECK-MIC|ADDR: 00-00-07-ee passed

18:6:23:664|INFO| ED:XX-XX-XX-XX-XX-XX-XX-XX|PER|10.810811%

18:6:23:665|DEBUG| ED:XX-XX-XX-XX-XX-XX-XX-XX|PACKET-RX|GW:00-80-00-00-a0-00-06-13 Time_us:205704196

18:6:23:665|DEBUG| ED:XX-XX-XX-XX-XX-XX-XX-XX|PACKET-RX|Downlink Packets Queued: 0

18:6:23:666|INFO| ED:XX-XX-XX-XX-XX-XX-XX-XX|FCTRL|ADR:0 ADRACK:0 ACK:0 CLASS:A OPTS:0


--


18:6:55:445|TRACE| TX-ACK|127.0.0.1:36071 4 bytes 01446801

18:6:55:446|TRACE| GW:00-80-00-XX-XX-XX-XX-13|SEEN|PUSH-DATA|127.00.1:36071

18:6:55:447|INFO| GW:00-80-00-XX-XX-XX-XX-13|FRAME-RX|Parsing 1 packets

18:6:55:448|DEBUG| GW:00-80-00-XX-XX-XX-XX-13|FRAME-RX|DATA:
40ee070000000100020d0619526b9935507b7962ce547d848cec22b84266e5

18:6:55:448|DEBUG| GW:00-80-00-XX-XX-XX-XX-13|FRAME-RX|FREQ: 867.300000 MHz DR4 RSSI: -47 dB SNR: 85 cB

18:6:55:449|DEBUG| GW:00-80-00-XX-XX-XX-XX-13|FRAME-RX|TYPE: Unconfirmed Up

18:6:55:449|DEBUG| GW:00-80-00-XX-XX-XX-XX-13|PACKET-RX|ADDR: 00:00:07:ee FCnt:0001

18:6:55:450|WARNING| ED:XX-XX-XX-XX-XX-XX-XX-XX|CHECK-PKT|MAX_FCNT_GAP exceeded diff:
13167262864924278730

18:6:55:450|INFO| ED:XX-XX-XX-XX-XX-XX-XX-XX|CHECK-PKT|FCNT: 00010001 LAST-FCNT: 00000037 Duplicate: no

18:6:55:456|INFO| ED:XX-XX-XX-XX-XX-XX-XX-XX|CHECK-MIC|ADDR: 00-00-07-ee failed

18:6:55:456|WARNING| ED:XX-XX-XX-XX-XX-XX-XX-XX|DROP-PKT|Addr: 00:00:07:ee Duplicate: no
```

1. The FCNT is expected to be only increasing per LoRaWAN. An FCNT should only be used once per set of session keys. The FCNT is only reset when OTA join is received.

   Or an application can reset the FCNT for an end-device.

   Reset Uplink Counter
   # lora-query -x device update <DEV-EUI> ulc 0

   Reset Downlink Counter
   # lora-query -x device update <DEV-EUI> dlc 0

2. Is there an option to disable the FCNT check?

   Yes, there is a way to disable the strict counter validation.
   This causes the system to be susceptible to replay attacks.
   Any previous FCNT can be replayed by an attacker.

   http://www.multitech.net/developer/forums/topic/disable-enablestrictcountervalidation/

Disabling the Strict Counter Validation for AEP

- `curl 127.0.0.1/api/loraNetwork -X PUT --data '{"lora": {"enableStrictCounterValidation":false}}' -H "Content-Type: application/json"`
- `Save and Restart the Conduit`

For mLinux add `"enableStrictCounterValidation":false` to the lora-network-server.conf "lora" section.

3. Why the CHECK-PKT|FCNT is +10000 after the reset?
   The server is checking if the 16-bit counter has rolled-over since the last packet.

4 LoRa Modulation

4.1 SNR and RSSI

SNR is the indicator for LoRa signal quality. The RSSI is a measure of all energy on the tuned frequency, including noise and signal, therefore a noisy channel may have high RSSI but low SNR.

An SNR value of 0 dB is the LoRa demodulation noise floor. There is an equal amount of noise and signal.

The SNR can be as low as -20 dB when using SF12BW125. SNR can be reported up to 20 dB.

4.2 Transmit Power

The transmit power of gateways and end-devices is determined by regional regulations. This regulation vary in the manner of limitation such as EIRP, ERP or conducted power. These variations affect how the software applies the max power and antenna gain settings.

4.2.1 US and AU

US and AU regulations limit the transmit power based on conducted power limits.

The AT+TXP and Tx Power settings on the Dot and Conduit represent the conducted power of the end-device.

If the power and antenna settings combine to exceed the regulated limit the software will reduce the radio power to maintain compliance based on the antenna gain setting.

FCC dictates a maximum conducted output for a transmitter in the ISM band is +30 dBm, up to +6 dBm antenna gain is allowed. If the total EIRP exceeds +36 dBm the conducted power must be reduced to stay below the maximum.

In the case of Conduit +27 dBm can be output with the MTAC-LORA-H card. Therefore a +9 dBm

antenna may be used. If a +10 dBm antenna is installed and configured, the conducted power will be reduced to +26 dBm.

4.2.2 EU, AS, KR and IN

EU, AS, KR and IN regions limit the EIRP of the transmitter. The software must take account the antenna gain.

The AT+TXP and Tx Power settings on the Dot and Conduit represent the EIRP of the end-device, the antenna gain will be used to reduce the power of the radio to meet the regulated limit at the selected frequency. If the power and antenna settings combine to exceed the regulated limit the software will reduce the radio power to maintain compliance based on the antenna gain setting.

ETSI dictates maximum EIRP for a transmitter in the ISM band is +14 dBm for most of the band with exception of +27 dBm at 869.4-869.65 MHz.

In the case of Conduit +27 dBm can be output with the MTAC-LORA-H card. Therefore if +3 dBm antenna is installed and configured the conducted output power will be limited to +24 dBm when using the 869.525 MHz channel and +11 dBm at channels below 869.4 MHz or above 869.65 mHz.

AS923 channel plan can configure a MaxEIRP setting on Conduit to be sent to end-device to limit the radio output.

Operation in Japan should configure for +16 dBm EIRP for both end-devices and gateway. With a +3 dBm antennas installed the Tx Power should be set to +13 dBm and the MaxEIRP set to +16 dBm.

4.3 Operating End-Devices too close to a Gateway

In close proximity the end-device may over-drive the gateway hardware causing ghost packets to be received on adjacent channels. These ghost packets can also be seen at time on channels 1 MHz or higher or lower than the intended channel due to harmonic and sometimes 4MHz and 11MHz above or below the primary signals.

Increase the distance between end-devices or lower the power. The antenna may also be removed if the end-device are only close to the gateway during testing and development.

4.3.1 Packet Forwarder Log

```
JSON up: {"rxpk":
[{"tmst":18641388,"chan":3,"rfch":0,"freq":868.800000,"stat":1,"modu":"LORA","datr":"SF12BW125","codr":
"4/5","lsnr":-14.8,"rssi":-115,"size":23,"data":"QAgAAACAzyoBcFxYzit7lLVueB/dsSI="},
{"tmst":18641396,"chan":1,"rfch":0,"freq":868.300000,"stat":1,"modu":"LORA","datr":"SF12BW125","codr":"
4/5","lsnr":9.2,"rssi":-57,"size":23,"data":"QAgAAACAzyoBcFxYzit7lLVueB/dsSI="},
{"tmst":18641404,"chan":0,"rfch":0,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF12BW125","codr":"
4/5","lsnr":-12.0,"rssi":-115,"size":23,"data":"QAgAAACAzyoBcFxYzit7lLVueB/dsSI="},
{"tmst":18641404,"chan":2,"rfch":0,"freq":868.500000,"stat":1,"modu":"LORA","datr":"SF12BW125","codr":"
4/5","lsnr":-11.0,"rssi":-113,"size":23,"data":"QAgAAACAzyoBcFxYzit7lLVueB/dsSI="}]}
```

4.3.2 Network Server Log

```
17:43:3:775|INFO|GW:00:80:00:00:a0:00:0c:1e|FRAME-RX|Parsing 4 packets

17:43:3:775|WARNING|Removed duplicate packet received on wrong channel, end-device may be operating
```

```
close to gateway

17:43:3:776|WARNING|Removed duplicate packet received on wrong channel, end-device may be operating
close to gateway

17:43:3:776|WARNING|Removed duplicate packet received on wrong channel, end-device may be operating
close to gateway
```

5 Processes

Several processes are used in the system depending on the options enabled. For lora services there is the Packet Forwarder, Network Server and Lens Cloud Server. Binaries for each service is installed in the /opt/lora/ directory.

An angel process is also provided on AEP build to ensure the service is running. If the process exits for any reason, the angel will restart it.

On mLinux build the angel process is not installed, a monit utility is provided for mLinux that can be configured to monitor and restart system processes. This utility is not configured by default.

To check that the services are running, grep the lora service and angel processes with the following command.

```
admin@mtcdt:~# ps ax | grep lora

28349 ?        S      0:03 /sbin/angel /opt/lora/lora-network-server -c /var/config/lora/lora-network-
server.conf --lora-eui 00:80:00:00:00:00:C3:21 --lora-hw-1 MTAC-LORA-1.0 --lora

-prod-1 MTAC-LORA-915 --lora-path /var/run/lora --noconsole

28351 ?        S      0:03 /sbin/angel /opt/lora/lora-lens-server -c /var/config/lora/lora-network-
server.conf --noconsole

28354 ?        S<l  390:38 /opt/lora/lora-network-server -c /var/config/lora/lora-network-server.conf
--lora-eui 00:80:00:00:00:00:C3:21 --lora-hw-1 MTAC-LORA-1.0 --lora-prod-1 MTAC

-LORA-915 --lora-path /var/run/lora --noconsole

28355 ?        R     76:18 /opt/lora/lora-lens-server -c /var/config/lora/lora-network-server.conf
--noconsole

28361 ?        S      0:04 /sbin/angel /var/run/lora/1/lora_pkt_fwd

28362 ?        S<l  682:20 /var/run/lora/1/lora_pkt_fwd

30879 ttyS0    R+     0:00 grep lora
```

Note the angel process PID should be close to the PID of the monitored processes. If there is a gap in the PID values it is an indicator that the process has been restarted by the angel.

5.1 Packet Forwarder (lora_pkt_fwd)

This process runs in the /var/run/lora/1 directory. Also in the directory is the configuration file read by the process to setup the channels in the radio and the forwarding address of the network server.

5.1.1 Source Repositories

Source code and protocol documentation can be found in these repositories.

https://github.com/Lora-net/packet_forwarder

https://github.com/Lora-net/lora_gateway

5.1.2 USB vs SPI MTAC-LORA cards

The USB card has more system overhead with the SPI->USB->SPI conversion of instructions.

The USB card idle CPU usage of the process is about 25%, under load the CPU usage can actually be lower, under 10% in some circumstances.

The SPI card CPU usage is ~2-4% at idle and under load.

5.1.3 Remote connection

The packet forwarder uses outboud UDP sockets to communicate with the network server. The network server responds to the packet forwarder by using these client socket ports. Firewalls should allow the packets from the network server through without needing to open the ports for inbound traffic.

5.1.4 Checking Hardware

5.1.4.1 MTAC-LORA-1.5

mts-io-sysfs lora options

- lora/product-id
- lora/eui
- lora/device-id
- lora/vendor-id
- lora/cdone
- lora/reset
- lora/hw-version
- lora/creset

5.1.4.1.1 Manually reset the card

mts-io-sysfs store lora/reset 0

mts-io-sysfs store lora/reset 1

5.1.4.1.2 Manually reload the FPGA firmware

mts-io-sysfs store lora/creset 0

mts-io-sysfs show lora/cdone

0 <<-- Expected output

mts-io-sysfs store lora/creset 1

mts-io-sysfs show lora/cdone

1 <<-- Expected output

5.1.4.2 Utilities

5.1.5 Running manually

Some output is available from the packet forwarder that is not logged to a file. It may be desired to run the packet forwarder to see that it is receiving and transmitting packets.

If there are issues communicating with the hardware they may be detailed in the packet forwarder error messages.

1. Kill the angel and lora_pkt_fwd process
    - `# ps ax | grep lora`
        - `28361 ?        S      0:04 /sbin/angel /var/run/lora/1/lora_pkt_fwd`
        - `28362 ?        S<l  682:20 /var/run/lora/1/lora_pkt_fwd`
    - `# kill 28361`
2. Start the packet forwarder
    - # cd /var/run/lora/1
    - # ./lora_pkt_fwd
3. When finished restart the packet forwarder service
    - AEP – the network server init script reads the config from the API and starts the appropriate services
        - # /etc/init.d/lora-network-server
    - mLinux
        - # /etc/init.d/lora-packet-forwarder

Example Output

Version Information

```
*** Basic Packet Forwarder for Lora Gateway ***
```

```
Version: 1.4.1
*** Lora concentrator HAL library version info ***
Version: 1.7.0; Options: ftdi sx1301 sx1257 full mtac-lora private;
***
INFO: Little endian host
```

Configuration options are read from the configuration files

```
INFO: found global configuration file global_conf.json, parsing it
INFO: global_conf.json does contain a JSON object named SX1301_conf, parsing SX1301 parameters
INFO: radio 0 enabled, center frequency 907500000
lgw_rxrf_setconf:894: Note: rf_chain 0 configuration; en:1 freq:907500000
INFO: radio 1 enabled, center frequency 908300000
lgw_rxrf_setconf:894: Note: rf_chain 1 configuration; en:1 freq:908300000
INFO: Lora multi-SF channel 0>  radio 0, IF -400000 Hz, 125 kHz bw, SF 7 to 12
lgw_rxif_setconf:998: Note: LoRa 'multi' if_chain 0 configuration; en:1 freq:-400000 SF_mask:0x7e
INFO: Lora multi-SF channel 1>  radio 0, IF -200000 Hz, 125 kHz bw, SF 7 to 12
lgw_rxif_setconf:998: Note: LoRa 'multi' if_chain 1 configuration; en:1 freq:-200000 SF_mask:0x7e
INFO: Lora multi-SF channel 2>  radio 0, IF 0 Hz, 125 kHz bw, SF 7 to 12
lgw_rxif_setconf:998: Note: LoRa 'multi' if_chain 2 configuration; en:1 freq:0 SF_mask:0x7e
INFO: Lora multi-SF channel 3>  radio 0, IF 200000 Hz, 125 kHz bw, SF 7 to 12
lgw_rxif_setconf:998: Note: LoRa 'multi' if_chain 3 configuration; en:1 freq:200000 SF_mask:0x7e
INFO: Lora multi-SF channel 4>  radio 1, IF -400000 Hz, 125 kHz bw, SF 7 to 12
lgw_rxif_setconf:998: Note: LoRa 'multi' if_chain 4 configuration; en:1 freq:-400000 SF_mask:0x7e
INFO: Lora multi-SF channel 5>  radio 1, IF -200000 Hz, 125 kHz bw, SF 7 to 12
lgw_rxif_setconf:998: Note: LoRa 'multi' if_chain 5 configuration; en:1 freq:-200000 SF_mask:0x7e
INFO: Lora multi-SF channel 6>  radio 1, IF 0 Hz, 125 kHz bw, SF 7 to 12
lgw_rxif_setconf:998: Note: LoRa 'multi' if_chain 6 configuration; en:1 freq:0 SF_mask:0x7e
INFO: Lora multi-SF channel 7>  radio 1, IF 200000 Hz, 125 kHz bw, SF 7 to 12
lgw_rxif_setconf:998: Note: LoRa 'multi' if_chain 7 configuration; en:1 freq:200000 SF_mask:0x7e
INFO: Lora std channel> radio 0, IF 300000 Hz, 500000 Hz bw, SF 8
lgw_rxif_setconf:972: Note: LoRa 'std' if_chain 8 configuration; en:1 freq:300000 bw:1 dr:4
INFO: FSK channel 8 disabled
lgw_rxif_setconf:919: Note: if_chain 9 disabled
```

Server information is parsed from the configuration

```
INFO: global_conf.json does contain a JSON object named gateway_conf, parsing gateway parameters
INFO: gateway MAC address is configured to 008000000000C321
INFO: server hostname or IP address is configured to "127.0.0.1"
INFO: upstream port is configured to "1780"
INFO: downstream port is configured to "1782"
```

```
INFO: synch word is configured to 12
INFO: downstream keep-alive interval is configured to 10 seconds
INFO: statistics display interval is configured to 20 seconds
INFO: upstream PUSH_DATA time-out is configured to 120 ms
INFO: packets received with a valid CRC will be forwarded
INFO: packets received with a CRC error will be forwarded
INFO: packets received with no CRC will NOT be forwarded
```

Radio hardware is verified, hardware is started and calibrated

```
setup_sx125x:673: Note: SX125x #0 version register returned 0x21
setup_sx125x:678: Note: SX125x #0 clock output enabled
setup_sx125x:721: Note: SX125x #0 PLL start (attempt 1)
setup_sx125x:673: Note: SX125x #1 version register returned 0x21
setup_sx125x:678: Note: SX125x #1 clock output enabled
setup_sx125x:721: Note: SX125x #1 PLL start (attempt 1)
lgw_start:1120: Note: calibration started (time: 3000 ms)
lgw_start:1141: Note: calibration finished (status = 191)
Info: Initialising AGC firmware...
Info: loading custom TX gain table
Info: putting back original RADIO_SELECT value
INFO: [main] concentrator started, packet can now be received
```

Keepalive packet is sent to network server and ACK is received

```
INFO: [down] PULL_ACK received in 0 ms

INFO: Start of downstream thread
INFO: [down] PULL_ACK received in 2 ms
```

Periodic stat messages are displayed logging received and forwarded message counts

```
##### 2018-01-15 17:30:50 GMT #####
### [UPSTREAM] ###
# RF packets received by concentrator: 0
# CRC_OK: 0.00%, CRC_FAIL: 0.00%, NO_CRC: 0.00%
# RF packets forwarded: 0 (0 bytes)
# PUSH_DATA datagrams sent: 0 (0 bytes)
# PUSH_DATA acknowledged: 0.00%
### [DOWNSTREAM] ###
# PULL_DATA sent: 2 (100.00% acknowledged)
# PULL_RESP(onse) datagrams received: 0 (0 bytes)
# RF packets sent to concentrator: 0 (0 bytes)
```

```
# TX errors: 0

##### END #####

INFO: [down] PULL_ACK received in 2 ms
```

The first five FIFO buffer bytes are displayed for each received packet, indicating size and CRC status.

/* 0: number of packets available in RX data buffer */

/* 1,2: start address of the current packet in RX data buffer */

/* 3: CRC status of the current packet, 5:OK, 7:BAD, 1:NO-CRC */

/* 4: size of the current packet payload in bytes */

```
lgw_receive:1423: FIFO content: 1 10 0 7 34

lgw_receive:1438: [6 17]

Note: LoRa packet

INFO: [down] PULL_ACK received in 1 ms
```

Packets transmitted with USB packet forwarder

```
Info: packet will be sent without CRC

e6.d3.33.b6.1.0.0.f.0.1c.1c.12.0.8.0.0.60.38.1c.cb.1.10.14.0.2.78.5b.c5.40.a4.95.a6.77.4a.67.1e.7c.70.c
b.9.3b.95.28.e3.end

lgw_send:1687: DEBUG: Tx pow_index 15, rf_power 26

Info: packet will be sent without CRC

e6.d3.33.b6.1.0.0.f.0.1c.1f.12.0.8.0.0.60.38.1c.cb.1.10.15.0.2.9e.2e.50.ab.83.50.68.56.95.60.54.28.75.5
b.97.1a.bc.ed.c7.83.70.3c.end

lgw_send:1687: DEBUG: Tx pow_index 15, rf_power 26

Info: packet will be sent without CRC

e6.d3.33.b6.1.0.0.f.0.1c.26.12.0.8.0.0.60.38.1c.cb.1.10.16.0.3.44.e6.a9.9a.69.66.38.fe.36.86.e7.c2.9a.5
8.b9.10.6b.48.31.dd.1a.61.ec.f9.24.57.38.c9.52.end
```

Packets received and transmitted with SPI packet forwarder

```
INFO: Received pkt from mote: 01CB1C38 (fcnt=3)


JSON up: {"rxpk":
[{"tmst":8590500,"chan":3,"rfch":0,"freq":902.900000,"stat":1,"modu":"LORA","datr":"SF10BW125","codr":"
4/5","lsnr":-14.2,"rssi":-113,"size":12,"data":"QDgcywEAAwBVUZCv"}]}

INFO: [down] PULL_RESP received  - token[0:0] :)


JSON down: {"txpk":{"appeui":"16-ea-76-f6-ab-66-3d-
80","codr":"4/5","data":"YDgcywEQLAACV05qTZKaJxb7WAjfHBcE5qVuZNHfQg==","datr":"SF12BW500","deveui":"00-
80-00-00-04-00-70-6f","freq":923.29999999999995,"gweui":"00-80-00-00-a0-00-0f-
4d","imme":true,"ipol":true,"modu":"LORA","ncrc":true,"powe":26,"rfch":0,"size":31,"twnd":2}}

Info: packet will be sent without CRC

a6.d3.33.0.b5.15.75.e.0.1c.26.12.0.8.0.0.60.38.1c.cb.1.10.2e.0.4.fc.58.9b.31.e7.5f.66.b5.25.a7.ea.1b.53
.3.5a.5b.de.41.c6.5.4.87.ad.2.ad.e4.39.3a.57.end
```

5.1.6 TCP Dump

Packet forwarder communications can be intercepted with tcpdump utility.

5.1.6.1 Uplink Packets and Stats

Only ASCII data

tcpdump -i lo udp port 1780 -vv -A

ASCII and HEX

tcpdump -i lo udp port 1780 -vv -X

```
admin@mtcdt:/var/config/lora# tcpdump -i lo udp port 1780 -vv -X
device lo entered promiscuous mode
tcpdump: listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes
15:28:21.503711 IP (tos 0x0, ttl 64, id 23647, offset 0, flags [DF], proto UDP (17), length 254)
    localhost.43801 > localhost.1780: [bad udp cksum 0xfefd -> 0x745c!] UDP, length 226
        0x0000:  4500 00fe 5c5f 4000 4011 df8d 7f00 0001  E...\_@.@.......
        0x0010:  7f00 0001 ab19 06f4 00ea fefd 0265 6600  .............ef.
        0x0020:  0080 0000 a000 0f52 7b22 7278 706b 223a  .......R{"rxpk":
        0x0030:  5b7b 2274 6d73 7422 3a32 3630 3731 3230  [{"tmst":2607120
        0x0040:  342c 2274 696d 6522 3a22 3230 3138 2d30  4,"time":"2018-0
        0x0050:  332d 3031 5431 353a 3238 3a32 312e 3537  3-01T15:28:21.57
        0x0060:  3830 3135 5a22 2c22 6368 616e 223a 362c  8015Z","chan":6,
        0x0070:  2272 6663 6822 3a31 2c22 6672 6571 223a  "rfch":1,"freq":
        0x0080:  3932 322e 3830 3030 3030 2c22 7374 6174  922.800000,"stat
        0x0090:  223a 312c 226d 6f64 7522 3a22 4c4f 5241  ":1,"modu":"LORA
        0x00a0:  222c 2264 6174 7222 3a22 5346 3942 5731  ","datr":"SF9BW1
        0x00b0:  3235 222c 2263 6f64 7222 3a22 342f 3522  25","codr":"4/5"
        0x00c0:  2c22 6c73 6e72 223a 3131 2e38 2c22 7273  ,"lsnr":11.8,"rs
        0x00d0:  7369 223a 2d34 392c 2273 697a 6522 3a31  si":-49,"size":1
        0x00e0:  322c 2264 6174 6122 3a22 6750 7236 2f77  2,"data":"gPr6/w
        0x00f0:  4741 4351 424e 4139 6569 227d 5d7d       GACQBNA9ei"}]}
15:28:22.090432 IP (tos 0x0, ttl 64, id 23671, offset 0, flags [DF], proto UDP (17), length 183)
    localhost.43801 > localhost.1780: [bad udp cksum 0xfeb6 -> 0x45be!] UDP, length 155
        0x0000:  4500 00b7 5c77 4000 4011 dfbc 7f00 0001  E...\w@.@.......
        0x0010:  7f00 0001 ab19 06f4 00a3 feb6 0221 e400  .............!..
        0x0020:  0080 0000 a000 0f52 7b22 7374 6174 223a  .......R{"stat":
        0x0030:  7b22 7469 6d65 223a 2232 3031 382d 3033  {"time":"2018-03
        0x0040:  2d30 3120 3135 3a32 383a 3232 2047 4d54  -01.15:28:22.GMT
        0x0050:  222c 226c 6174 6922 3a34 352e 3039 3933  ","lati":45.0993
        0x0060:  382c 226c 6f6e 6722 3a2d 3933 2e31 3936  8,"long":-93.196
        0x0070:  3137 2c22 616c 7469 223a 3239 342c 2272  17,"alti":294,"r
        0x0080:  786e 6222 3a31 2c22 7278 6f6b 223a 312c  xnb":1,"rxok":1,
        0x0090:  2272 7866 7722 3a31 2c22 6163 6b72 223a  "rxfw":1,"ackr":
        0x00a0:  302e 302c 2264 776e 6222 3a30 2c22 7478  0.0,"dwnb":0,"tx
        0x00b0:  6e62 223a 307d 7d                        nb":0}}
```

5.1.6.2 Downlink Packets

Only ASCII data

tcpdump -i lo udp port 1782 -vv -A

ASCII and HEX

tcpdump -i lo udp port 1782 -vv -X

```
admin@mtcdt:/var/config/lora# tcpdump -i lo udp port 1782 -vv -X
device lo entered promiscuous mode
tcpdump: listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes
[INFO] mts-io:mts_attr_store_radio_reset:271: radio reset timer is running.

15:28:56.878104 IP (tos 0x0, ttl 64, id 25096, offset 0, flags [DF], proto UDP (17), length 32)
    localhost.1782 > localhost.43801: [bad udp cksum 0xfe1f -> 0x12d9!] UDP, length 4
        0x0000:  4500 0020 6208 4000 4011 dac2 7f00 0001  E...b.@.@.......
        0x0010:  7f00 0001 06f6 ab19 000c fe1f 02ea 3a01  ..............:.
15:28:57.685400 IP (tos 0x0, ttl 64, id 25121, offset 0, flags [DF], proto UDP (17), length 322)
    localhost.1782 > localhost.45851: [bad udp cksum 0xff41 -> 0x9c3a!] UDP, length 294
        0x0000:  4500 0142 6221 4000 4011 d987 7f00 0001  E..Bb!@.@.......
        0x0010:  7f00 0001 06f6 b31b 012e ff41 0200 0003  ...........A....
        0x0020:  7b22 7478 706b 223a 7b22 6170 7065 7569  {"txpk":{"appeui
        0x0030:  223a 2231 362d 6561 2d37 362d 6636 2d61  ":"16-ea-76-f6-a
        0x0040:  622d 3636 2d33 642d 3830 222c 2263 6f64  b-66-3d-80","cod
        0x0050:  7222 3a22 342f 3522 2c22 6461 7461 223a  r":"4/5","data":
        0x0060:  2259 5072 362f 7745 6743 6743 7730 3453  "YPr6/wEgCgCw04S
        0x0070:  4822 2c22 6461 7472 223a 2253 4639 4257  H","datr":"SF9BW
        0x0080:  3132 3522 2c22 6465 7665 7569 223a 2234  125","deveui":"4
        0x0090:  382d 3833 2d63 372d 6466 2d33 302d 3031  8-83-c7-df-30-01
        0x00a0:  2d31 612d 3663 222c 2266 7265 7122 3a39  -1a-6c","freq":9
        0x00b0:  3232 2e32 3030 3030 3030 3030 3030 3030  22.2000000000000
        0x00c0:  352c 2267 7765 7569 223a 2230 302d 3830  5,"gweui":"00-80
        0x00d0:  2d30 302d 3030 2d61 302d 3030 2d30 662d  -00-00-a0-00-0f-
        0x00e0:  3532 222c 2269 706f 6c22 3a74 7275 652c  52","ipol":true,
        0x00f0:  226d 6f64 7522 3a22 4c4f 5241 222c 226e  "modu":"LORA","n
        0x0100:  6372 6322 3a74 7275 652c 2270 6f77 6522  crc":true,"powe"
        0x0110:  3a32 362c 2272 6663 6822 3a30 2c22 7369  :26,"rfch":0,"si
        0x0120:  7a65 223a 3132 2c22 746d 7374 223a 3632  ze":12,"tmst":62
        0x0130:  3434 3634 3238 2c22 7477 6e64 223a 317d  446428,"twnd":1}
        0x0140:  7d0a                                     }.
15:28:57.694975 IP (tos 0x0, ttl 64, id 25122, offset 0, flags [DF], proto UDP (17), length 40)
    localhost.45851 > localhost.1782: [bad udp cksum 0xfe27 -> 0x95da!] UDP, length 12
        0x0000:  4500 0028 6222 4000 4011 daa0 7f00 0001  E..(b"@.@.......
        0x0010:  7f00 0001 b31b 06f6 0014 fe27 0200 0005  ...........'....
        0x0020:  0080 0000 a000 0f52                      .......R
```

5.1.7 Configuration Options

Two sections are available in the configuration: radio settings (SX1301_conf) and server settings (gateway_conf).

5.1.7.1 SX1301_conf

- lorawan_public – set sync word to false: 0x12 or true: 0x34
- clksrc – must be 0 for Multitech gateway hardware
- antenna_gain – configure the installed antenna gain if the network server sends EIRP value with downlinks. The Multitech network server will account for antenna gain before sending tx power to the forwarder.

5.1.7.1.1 Radios

The radio_0 and radio_1 settings configure the two front-end radios on the MTAC cards.

- enable – true if radio is enabled for use
- type – type of radio chip installed SX1257 or SX1255, frequency range differs
- freq – center frequency the radio is configured to listen for packets
- tx_enable – true if radio can be used for transmissions
- tx_freq_min – minimum frequency to be allowed for transmissions
- tx_freq_max – maximum frequency to be allowed for transmissions
- rssi_offset – offset in dBm to adjust the radio RSSI reading

5.1.7.1.2 Channels

Eight channels can be configured to receive LoRa packets using 125 kHz bandwith. These channels are configured in the **chan_multiSF_x** settings. Each channel has an enable boolean, a selected radio and an intermediate frequency. The channel frequency will **freq** setting of the radio plus the **if** setting of the channel.

- enable – true if this channel is enabled for use
- radio – selected radio to listen for packets
- if – intermediate frequency offset applied to the selected radio “freq” setting

Two additional channels can be configured with the **chan_Lora_std** and **chan_FSK** settings.

- enable – true if this channel is enabled for use
- radio – selected radio to listen for packets
- if – intermediate frequency offset applied to the selected radio “freq” setting
- bandwidth – channel bandwidth
- spread_factor (LoRa) – channel spreading factor (7-12)
- datarate (FSK) – channel datarate in bps

Example

Configure Channel 8 - 916.6 MHz on Radio 1

- Radio 1 is set to a center frequency of 916.4 MHz
- Configure the offset to +200 KHz and select Radio 1

```
"chan_multiSF_7" :
{
        "enable" : true,
        "if" : 200000,
        "radio" : 1
},
```

```
"radio_1" :
{
        "enable" : true,
        "freq" : 916400000,
        "rssi_offset" : -162,
        "tx_enable" : false,
        "type" : "SX1257"
},
```

5.1.7.1.3 Look-up-table (LUT)

There are sixteen power settings that can be configured, these settings have been selected during calibration for each hardware. The power sent from the network server is looked up in this table by the **rf_power** setting. The corresponding **pa_gain**, **mix_gain** and **dig_gain** settings are then input into the radio for the transmission. These settings should not be adjusted.

5.1.7.2 gateway_conf

- gateway_ID – gateway identifier sent in each message to the network server
- server_address – address of the network server
- serv_port_up – port for sending uplink packets to the network server
- serv_port_down – port to ping network server and receive downlink packets
- keepalive_interval – interval to ping the network server
- stat_interval – interval to send stat messages to the network server
- push_timeout_ms – socket timeout when publishing messages to the network server
- autoquit_threshold – number of keepalive messages without response to wait before quitting
- forward_crc_valid – enable to forward valid packets to the network server, default: true
- forward_crc_error – enable to forward CRC failed packets to the network server, default: true. The network server will reject packets with failed CRC, it may not be necessary to forward the packets except for a statistic of local RF quality or to monitor the gateway performance over time. Some random CRC failed packets are expected to be received from random noise.
- forward_crc_disabled – enable to forward packets without CRC enabled to the network server, default: false. LoRaWAN protocol expects uplink packets to have CRC enabled.

5.2 Network Server (lora-network-server)

The network server will be passed the /var/config/lora/lora-network-server.conf file location to read on start-up. On AEP this file is created from the API settings at /api/loraNetwork path. In mLinux the file contents are used directly.

If there may be an issue with the configuration, the network server can be started manually.

```
# /etc/init.d/lora-network-server stop
```

```
# /opt/lora/lora-network-server -c /var/config/lora/lora-network-server.conf
```

Any errors that occur on start-up will be displayed in the console. The packet forwarder will not be started with the network server process. It could also be started manually to serve packets to the network server process.

To restart the service after testing.

```
# /etc/init.d/lora-network-server start
```

The network server process will log to a file or syslog depending on selected options. AEP logs to syslog by default. The mLinux sample configuration has configuration for output to file. The default file location is /var/log/lora-network-server.log.

5.3 Lens Cloud Server (lora-lens-server)

The Lens Cloud Server accepts UDP messages from the network server to forward to the Lens Cloud over HTTP REST API.

The Lens cloud server process will log to a file or syslog depending on selected options. AEP logs to syslog by default. The mLinux sample configuration has configuration for output to file. The default file location is /var/log/lora-lens-server.log.