

## NAME

**RoWAN** is a network–stream relay (server) over Wide Area Network.

## DESCRIPTION

**RoWAN** relays video streams from both local and remote sources to local multicast/unicast destinations.

### Rationale

**RoWAN** is a server that enables a video stream to cross over **WAN**, from a remote network to the site closest to consumers. Imagine, if you will, that a satellite dish (or any other source) is generating a feed to be served to clients hundreds of miles (kilometers) away. A network link must exist between the two endpoints, as long as there are subscribers to the feed. This application's purpose is to establish and maintain a constant stream of data between two vastly separated endpoints, using the most suitable network protocols.

**RoWAN** considers maintaining a data stream (reading from point **A** and writing to **B**) as a single **task**. At point A (source), the accepted protocols are: **HTTP** and **UDP**, at point B it's always **UDP**. Tasks are submitted either statically, via a file (where each line specifies a task), or dynamically, via HTTP request.

As a server, **RoWAN** is designed to handle (simultaneously) as many tasks as machine resources would allow, add and remove tasks, report on the status, etc. For the complete list of features, please refer to the man pages of specific components.

### Modules

**rwn(1)** and **iga(1)** are the modules of **RoWAN**. **rwn** is the core module, handling data I/O tasks. **iga** is an IGMP listener/agent that allows to submit tasks based on IGMPv2 **JOIN** and **LEAVE** messages.

### Task types

**rwn(1)** supports the following types of tasks:

- U: UDP–to–UDP tasks, where either side could be unicast or multicast.
- H: HTTP–to–UDP tasks, where source is obtained via an arbitrary HTTP request and destination is UDP (unicast or multicast).

**Important:** HTTP source **must** be **MPEG-TS** so that it could be parsed into packets.

- D: Daisy–chained UDP–to–UDP task. The source triggers a request to a *remote* **rwn(1)** instance that relays a *remote* stream to the *local* **rwn** instance.

### Simple relay

This is the mode of operation, when both source and destination are directly reachable from the local network. U–tasks and H–tasks operate in this mode. As soon as **rwn** receives the two endpoint addresses and, in case of an H–task, completes HTTP handshake, it starts relaying data from source to destination.

### Daisy–chained relay

This mode allows one to relay data from a remote network to the local one and requires a **rwn** module on each side (*local* and *remote*). The local **rwn** has a list of *routes*, each route maps a local address to the remote side to receive the source stream from. For each submitted request, **rwn** checks if the *source* maps to a route and if so, uses the route entry to request source stream from the *remote* side.

A straightforward use case for a daisy–chained task would be relay multicast stream over WAN.

**Example A:**

A client requests a relay from 225.0.4.4:4040 to 10.0.22.14:3030. **rwn** looks up the following route entry:

```
225.0.4.4:4040 192.168.1.162:4056 225.3.3.1:1212 192.168.1.30:20111
```

This route maps local multicast group 225.0.4.4:4040 as a remote source. Local **rwn**, rather than attempting to subscribe to this group locally, would send an HTTP request to the remote service at 192.168.1.162:4056, asking to subscribe (on its side) to remote group 225.3.3.1:1212 and send the stream to 192.168.1.30:20111. After successfully making the request, the local **rwn** instance would start relaying data from 192.168.1.30:20111 to 10.0.22.14:3030 (client's destination).

In a schematic way:

```
Local [request] -> [http://192.168.1.162:4056] Remote [225.3.3.1:1212] -> Local
[192.168.1.30:20111] -> [destination]
```

For a complete definition of routes and their format please refer to **rwn(1)** manpage.

**IGA-backed daisy-chained relay**

Subscribing to a remote group/stream via **rwn** requires an initial HTTP request to local **rwn**, which could be an issue if a subscription is made by a program you cannot (or would not) modify. What if one could just request a local multicast group and get the remote stream started, transparently? Sure. This is exactly what **iga(1)** is for: relaying **IGMP** requests to **rwn**.

**iga(1)** listens on IGMP traffic and, when it sees a request relating to a group in its scope, forwards it to local **rwn**. In our previous scenario, all one would need to do is to subscribe to 225.0.4.4:4040 using any third-party software. Once all subscribers are gone, the **LEAVE** message would be relayed to **rwn** to stop the stream

**Task lifecycle**

**RoWAN** tasks are normally **not** tied to the clients that submit them: the real consumers are still a layer apart. Initial request starts a task or, if such a task already exists, increments the client count. The same kind of request, augmented with a special *command* suffix, needs to go out in order to stop the task (or decrement the counter). A task could also be started in a bound or **keep-alive** mode. In such a mode **rwn** does not close the HTTP session started by the client's request, so the session goes on until the client terminates the HTTP session. This mode makes it simpler to submit tasks where client wants exclusive control over the initiated stream.

**Submitting tasks**

**rwn(1)** listens on specified port(s) for user-submitted HTTP requests in the format:

```
http://{addr}:{port}/src/{source-uri}/dst/{destination-uri}/{command}
```

*WHERE*

```
{addr}:{port}    ::= IPv4/6 address of the user-request listener;
{source-uri}    ::= URI for the source stream;
{destination-uri} ::= URI for the destination stream;
{command}       ::= <none>|keep-alive|drop|kill
```

URI format: {protocol}://{addr}:{port}. Protocol part (along with double slash) could be omitted for UDP.

{protocol} ::= *UDP* or *HTTP* for source and strictly *UDP* for destination.

**Commands:**

**<none>**: means no command at all, as in `http://localhost:4056/src/224.0.2.24:1212/dst/192.168.0.20:40394` – no slash after **dst**;

**keep-alive**: start a client-bound (or keep-alive) task, **rwn** would not close the HTTP-request connection until the client does;

**drop**: stop the matching task (if any) or decrement client counter if other clients exist;

**kill**: stop the matching task, regardless of any other clients still using the stream.

**Sample requests:**

**A:** `http://localhost:4056/src/224.0.2.24:1212/dst/192.168.0.20:40394` specifies a UDP-to-UDP relay task (*U-task*) from multicast group `224.0.2.24:1212` to unicast address `192.168.0.20:40394`.

**B:** `http://localhost:4056/src/http://10.1.1.4:8088/udp/226.0.0.2:4040/dst/226.1.1.16:2020` specifies an HTTP-to-UDP relay task (*H-task*) from video stream from `http://10.1.1.4:8088/udp/226.0.0.2:4040` to multicast address `226.1.1.16:2020`.

**C:** `http://localhost:4056/src/224.0.2.24:1212/dst/192.168.0.20:40394/keep-alive` same as **A** but asks for a client-bound session.

**D:** `http://localhost:4056/src/224.0.2.24:1212/dst/192.168.0.20:40394/drop` requests to stop (or decrement client counter) of the running *U-task*.

**Submitting requests in batch mode**

**rwn(1)** can read task requests from a text file in a pre-defined format (see **rwn(1)** manpage for details). The file is read at the application launch, all submitted tasks start immediately.

**Task report**

**rwn(1)** can periodically dump its task list along with basic statistics to a text file, in JSON format. This mechanism has been picked over on-demand HTTP reporting in order to simplify the internal logic and minimize the latency.

**AUTHORS**

Pavel V. Cherenkov

**SEE ALSO**

**rwn(1)**, **iga(1)**, **rwn.conf(5)**

**NAME**

**rwn** – data-relay component of **rowan(1)**

**SYNOPSIS**

**rwn** OPTIONS

**DESCRIPTION**

**rwn** relays video streams as per requests, submitted via HTTP or **iga(1)** communications channel. The format of HTTP requests is described in **rowan(1)** manpage. **rwn** listens for client and **iga** requests on dedicated port(s). Whenever it receives a request, it processes it and starts (or stops) a streaming task.

**rwn** reads its configurations from a config file (in *libconfig* format) specified as a command-line parameter (see **OPTIONS** below). For full description of configuration parameters please see **rwn.conf(5)** manpage.

**OPTIONS**

**rwn** accepts the following options:

- h, --help, -?, --options**  
output brief option guide.
- C, --config path**  
specify configuration file.
- l, --logfile path**  
specify log file.
- L, --level crit|err|norm|info|debug**  
specify log level.
- p, --pidfile path**  
specify pid file.
- T, --term**  
run as a terminal (non-daemon) application. This is the default behavior when **gws** is run by a non-privileged user. **-T** could be specified when run as root in order **NOT** to become a daemon, for instance, for debugging purposes.
- t, --tasks path**  
specify file with task specifications for batch mode (see the format below).
- r, --routes path**  
specify file with route specifications (see the format below).
- V, --version**  
output application's version and quit.
- q, --quiet**  
send no output to terminal. This is to suppress any output normally sent to standard output or error streams. Unless specified, when run from a non-privileged account, **gws** will **mirror** diagnostic messages sent to the log (as specified with the **-l** option) to standard output.
- P, --cpu**  
Set CPU affinity for the process. This option allows to restrict the process to the given CPU/core (numbered from *0* to *N-1*).
- M, --maxpkts**  
Set the maximum number of datagrams/packets to process in a single I/O event. This setting regulates how much data can be accumulated before it is sent out. The permissible range is 1–64. The setting affects all types of tasks.

**-K,--syskey**

Generate **host key** to identify this computer/VM when generating a license.

**TASK LIST FORMAT**

Task-list is a text file where each line specifies a single task. Each line has exactly **two** fields (*source* and *destination*) separated by a *single* space character. Each field is a URL, containing the schema (protocol), address and port of the corresponding endpoint.

Task specification examples:

udp://224.0.2.26:5050 udp://192.168.1.50:4040

is a spec for a UDP-to-UDP task (which may or may not be daisy-chained).

http://192.168.1.30:4044/udp/226.2.2.16:5050 udp://224.0.2.28:5050

is a spec for an HTTP-to-UDP task.

**ROUTE LIST FORMAT**

Route-list is a text file where each line specifies a single route. Each line has exactly **four** fields: local-source, remote-host, remote-source and relay-destination, where:

*local-source*

is the address/port combination for the locally-requested (UDP) stream, that may be prefixed by the (percentage sign) character. This field is used as the key in the look-up table for the routes. When there is the percentage prefix, only the *address part* of the field is used (see **IGA requests** below).

*remote-host*

is the address/port of the remote service (most likely an **rwn**) that can respond to **RoWAN** task requests. An HTTP session will be established with the host if the route used.

*remote-source*

is the address/port of the remote stream to request from *remote-host*. A group that subscribers know as 224.0.2.24:1212 on your network, for instance, could be 226.12.26.3:4141 on the remote network. **NB:** a dash instead of the address/port would indicate that remote-source in this case **matches** local-source

*relay-destination*

is the address/port at your *local server*, where you need the stream to be forwarded from the remote-host. This is the destination (second part) that **rwn(1)** uses in its request to *remote-host*, using *remote-source* as the first (source) part.

Route specification examples:

224.0.3.16:4040 192.168.1.50:4056 - 10.16.10.30:6766

Matches the same multicast group (224.0.3.16:4040) locally and at the network local to 192.168.1.50, whence the stream is to be forwarded to 10.16.10.30:6766

%224.0.3.17:6060 192.168.1.162:4056 224.0.3.17:5050 192.168.1.30:22111

Matches requests made (most likely by **iga(1)** to get 224.0.3.17 (**yes, no port!**). The request goes out to 192.168.1.162:4056 from where the stream at 224.0.3.17:5050 is read and relayed to 192.168.1.30:22111.

## IGA REQUESTS

**iga(1)** relays IGMP requests to **rwn** which dedicates a special channel of communication for it. Whenever **iga** catches an IGMP message relating to a group of interest (defined via **iga** command line options), it forwards a **JOIN** or a **LEAVE** message (via UDP) to its 'master' (**rwn**). Since IGMP does not recognize ports, **iga** uses addresses without one. This is where 'percentage-prefixed' routes come into play. **rwn** takes and address supplied by **iga** and matches it to a route entry, then creates a streaming task.

**iga(1)** JOIN request only carries the *local-source*, so the destination (normally specified in the *dst* section) is assumed to be equivalent to *local-source* with the **port included**. Therefore, when a third-party application subscribes to 224.0.3.17:6060, **iga** relays the subscription to **rwn** which streams the remote group (224.0.3.17:5050) to 224.0.3.17:6060.

**NB:** Please mind that **iga** would do the same thing for subscription involving any other port (than 6060), yet, by the route configuration, **rwn** would always relay the same stream to a specific local address/port – 224.0.3.17:6060.

For details on **iga(1)** and its options and configuration, please see the corresponding manpage.

## TASK REPORTS

**rwn(1)** can generate reports on the tasks it is currently running. A report is written into a designated text file (specified in the config), formatted as a JSON hierarchy.

### Global fields:

report\_time [string]:

time the report was generated, for instance, 2016-04-12 19:31:02.798137 EST

pid [number]:

process id of the **rwn** instance that generated the report.

task\_count [number]:

number of tasks currently running

### Task-record fields:

tid [string]:

task ID shown in **rwn** logs: the first letter of *tid* identifies task type, as in 'U0014'.

key [string]:

the hash-table key used for the task: includes source and destination URLs.

keep-alive [string]:

"yes" for client-bound tasks, "no" otherwise.

uptime [number]:

task uptime in seconds

in-kb [number]:

total received within the task (in Kb = 1024 bytes)

out-kb [number]:  
total sent within the task (in Kb = 1024 bytes)

avg-ikps [number]:  
average read speed in Kb/sec.

avg-okps [number]:  
average write speed in Kb/sec.

**Example:**

```
{ "report_time": "2016-04-12 19:31:02.798137 CET" "pid": 12611 "task_count": 1 "task_list": [
  { "tid": "U0003", "key": "udp://224.0.2.26:5050|udp://192.168.1.50:6060", "keep-alive": "no", "uptime":
  46.91, "in-Kb": 103.12, "out-Kb": 101.95, "avg-ikps": 2.20, "avg-okps": 2.17, } ] }
```

**AUTHORS**

Pavel V. Cherenkov

**SEE ALSO**

**rowan(1),iga(1),rwn.conf(5)**

## NAME

rwn.conf – RoWAN data-relay component (rwn) configuration file

## DESCRIPTION

The configuration file contains the parameters read by **rwn** at launch. The file is in **libconfig** format. An example of a **rwn.conf** is provided with the installation.

## CONFIG ENTRIES

**rwn(1)** settings begin with the **rowan.** prefix, as in 'rowan.max\_out\_packets'. No other sections are supposed to be in the file, yet **rwn** should be agnostic to those if they are included.

**rowan.log.level = err|crit|warn|norm|info|debug [info]**

Defines the level of verbosity for the application log.

**rowan.log.file = path**

Full path to the log. If the path is prefixed by a | (pipe) symbol, the log is treated as a stream opened with **popen(3)** call.

**rowan.log.max\_size\_mb = num [16]**

Maximum file size (in Mb, i.e. 1048576-byte chunks). Log is rotated when this size is exceeded.

**rowan.log.max\_files = num [16]**

Maximum number of files to rotate to. The next rotation after this limit removes the oldest rotated log.

**rowan.listener.user.\***

The following are the settings equally applying to [up to 16] listeners of *user* requests (the only listener type currently supported in the *listener.\** section). An example of multi-listener configuration is provided with the installation.

**rowan.listener.\*.alias = unique-alias [{ifc}:{port}]**

Unique human-readable identifier for the given listener. Populated by default by interface name and port (see below) separated by colon. For ifc=eth0 and port=3030, the alias, unless specified otherwise, would be set to *eth0:3030*.

**rowan.listener.\*.ifc = interface [any]**

Name or the address of the network interface for the listener of requests. **any**, **all** signifies the 'anonymous' interface with the address of 0, which means that the first eligible network interface will be picked by your OS.

**rowan.listener.\*.port = number**

Port number for the listener.

**rowan.listener.\*.default\_af = inet | inet6 [inet]**

is the address family to be used when an interface cannot be uniquely linked to a family. For instance, an interface could have both IPv4 and IPv6 addresses associated with it.

**rowan.listener.max\_to\_accept = num [128]**

is the maximum number of new connection to accept in a single I/O event. IMO, if you have more than a hundred sockets queued up, you're under a DOS attack.



**rowan.listener.user\_request\_timeout = num [5000]**

maximum time (in milliseconds) for any user request to last.

**rowan.runtime.\***

is the section that defines parameters pertinent to running as a *daemon*.

**rowan.runtime.run\_as\_user = username []**

Run as the specified *user* in daemon mode, if empty run as root (**not** recommended).

**rowan.runtime.non\_daemon = true/false [false]**

If *true*, event if started as root, DO NOT become a daemon. The standard behavior is to become a daemon if started with `EID == 0`.

**rowan.runtime.pidfile = path []**

Full path to the application pidfile. Please note that the hosting directory must be writable by the user the daemon would run as.

**rowan.multicast\_ifc\_in = name []**

Interface to use receiving multicast data.

**rowan.multicast\_ifc\_out = name []**

Interface to use sending multicast data.

**rowan.max\_pkts = num [4]**

Set the maximum number of datagrams/packets to process in a single I/O event. This setting regulates how much data can be accumulated before it is sent out. The permissible range is 1–64. The setting affects all types of tasks: for UDP the count is in **UDP datagrams**, for HTTP (TCP) it is in **MPEG-TS packets**. The optimal value for this setting (on both ends of the tunnel) is crucial for maintaining quality of the displayed stream.

**rowan.task\_list = path []**

Path to the task list (see **rwn(1)** for format).

**rowan.route\_list = path []**

Path to the route list (see **rwn(1)** for format).

**rowan.max\_tasks = num [1024]**

Maximum number of tasks to handle simultaneously.

**rowan.idle\_task\_timeout = ms [5000]**

Maximum time for a task to exist without data I/O.

**rowan.iga.\***

This section defines the settings needed to communicate with **iga(1)** IGMP agent.

**rowan.iga.udp-listener = address:port []**

Address and port (UDP) to listen on for messages from **iga**. No listener if empty.

**rowan.report.\***

This section defines parameters for the task report generated by **rwn**.

**rowan.report.path = path []**

Path to the task report generated by **rwn**. If no path is specified, reports do NOT get generated.

**rowan.report.force\_refresh\_sec = num [0]**

If set to a value greater than zero, **rwn** would generate a task report every N seconds. If set to zero, **rwn** would only generate a report **whenever a task has been added or removed**, in which case cumulative statistics would only be reflected at those *change points*.

**AUTHORS**

Pavel V. Cherenkov

**SEE ALSO**

**rowan(1),iga(1),rwn(1)**

**NAME**

**iga** – IGMP agent of **rwn(1)**

**SYNOPSIS**

**iga** OPTIONS

**DESCRIPTION**

**iga** monitors IGMPv2 traffic and reports on relevant (multicast group) addresses to **rwn(1)**.

**iga** considers an address relevant if it is within a specified range or is listed in a text file (one address per line).

**iga(1)** intercepts IGMP traffic on the given network interface and relays information on relevant addresses to **rwn(1)**

**OPTIONS**

**iga** accepts the following options:

- ?** Output brief option guide.
- i interface**  
Network interface to listen on.
- R address:port**  
UDP address:port (**rowan.iga.udp\_listener**) to report to.
- g min-group**  
Minimum multicast group to relay.
- G max-group**  
Maximum multicast group to relay.
- F path** List of relevant groups, one address per line, **no ports**.
- l path** Log file.
- L crit|err|norm|info|debug**  
Log level.
- p path** PID file.
- T** Run as a non-daemon application when starting as root.
- u username**  
User to run as in daemon mode.

**AUTHORS**

Pavel V. Cherenkov

**SEE ALSO**

**rowan(1),rwn(1),rwn.conf(5)**