The trx-control Integration Guide

Marc Balmer HB9SSB <info@hb9ssb.ch>

Table of Contents

General Principles . . . 2
Request format . . . 3
  Destinations . . . 3
General Requests . . . 4
Controlling Transceivers . . . 5
GPIO . . . 7
Extensions . . . 8
  Common commands . . . 8
  config . . . 8
  ping . . . 8
  keepalive . . . 8
  qrz . . . 9
  dxcluster . . . 10
  tasmota . . . 11

This guide is intended for developers who want to integrate trx-control with their own software, e.g. logging or contesting software.

General Principles

A client accesses the trx-control daemon trxd(8) over TCP/IP using either a plain IP socket or a WebSocket. Both IPv4 and IPv6 are supported.

All data exchange is done using JSON formatted data packages. If a plain socket is used, the so-called NDJSON (Newline Delimited JSON) format is to be used, i.e. the JSON data is sent as one line, delimited by a newline character.

All requests will be answered by a reply.

Request format

Every request must contain at least a **request** field. Depending on the request type, one or more mandatory or optional fields must be present.

The reply to each request will at least contain a **status** and a **reply** field. The **status** indicates success (**Ok**) or failure (**Error**). The **reply** field is usually set to the value of the **request** field.

Usually parameters are sent back in a reply.

Request is empty or contains invalid JSON data:

```
{
  "status": "Error",
  "reason": "Invalid input data or no input data at all"
}
```

Request does not contain a request field:

```
{
  "status": "Error",
  "reason": "No request"
}
```

Client sent an unknown request:

```
{
  "status": "Error",
  "reason": "Unknown request",
  "request: "<whatever the client sent as request>"
}
```

Destinations

As trx-control supports an unlimited number of devices, extensions etc. the concept of destinations is used. Each transceiver, device, or, extension represents a named destination. With every request you can send the optional "to" field, indicating the destination the request should be dispatched to.

A destination can be set by sending JSON data containing only the **to** field, without a request. Once set, the destination will be used for all subsequent requests until a new destination is specified. If no destination has ever been specified, the request will be dispatched to the default transceiver.

The list of available destinations can be queried with the **list-destination** request.

General Requests

These requests are processed by trxd(8) and are not routed to a device or extension.

Request	Purpose	Parameters	Reply
list-destination	List all available destinations		An array containing **name** and **type** fields
to	Select a destination		

Controlling Transceivers

Parameters in *italic* are optional.

Request	Purpose	Parameters	Reply
get-info	Request information about a transceiver		**name** The name of the transceiver. **frequencyRange** An array containing **min** and **max** numbers indicating the minimum and maximum operating frequency. **operatingModes** An array of strings containing the operating modes.
set-frequency	Set the operating frequency	**frequency** The operating frequency	
get-frequency	Get the operating frequency		**frequency** The operating frequency
set-mode	Set the operating mode	**mode** The operating mode *band* The band, either `main` or `sub`	
get-mode	Get the operating mode		**mode** The operating mode **band** The band, either `main` or `sub`
lock-trx	Lock the transceiver		
unlock-trx	Unlock the transceiver		

Request	Purpose	Parameters	Reply
start-status-updates	Start sending automatic status updates		Upon status changes, a JSON package with the following fields is sent to the client: **request** Contains the string `status-update` **status** An array with the following fields: **frequency** The operating frequency **mode** The operating mode
stop-status-updates	Stop sending automatic status updates		

GPIO

Extensions

Common commands

All extensions understand the **listen** and **unlisten** command to listen to status updates (or stop listening for them). What a status update actually means is of course depending on the extension.

Request	Purpose	Parameters	Reply
listen	Start listening for status updates		
unlisten	Stop listening for status updates		

config

The config extension can be used to access the trxd(8) configuration. Currently it can only be used for read access.

Request	Purpose	Parameters	Reply
getConfiguration	Get the current configuration		An array containing all configuration elements

Be aware that the configuration might contain sensitive information such as password or database connection strings!

ping

The ping extension serves the sole purpose to check the liveliness of trx-control.

Request	Purpose	Parameters	Reply
ping	Check connection		**reply** The string `pong`

The ping extension does not send status updates, even if you sent a **listen** request.

keepalive

The keepalive extension is meant for situations where trxd(8) runs behind a NAT-gateway or a firewall that kills idle TCP/IP connections. In this case the keepalive extension sends a keepalive package to connected clients at a set interval.

Example configuration

```
extensions:
  keepalive:
    script: keepalive
    callable: false
    configuration:
      timeout: 300
```

Request	Purpose	Parameters	Reply
listen	Start sending keepalives		
unlisten	Stop sending keepalives		

qrz

The qrz extension can be used to make online queries to the QRZ.com database. All callsigns are locally cached for quicker access if queried multiple times or by different clients.

A QRZ.com subscription is needed for the full functionality of this extension. Without a subscription, you will only get limited data and the number of lookups is limited to 100/day.
See https://shop.qrz.com/collections/subscriptions for all QRZ.com subscription options.

Example configuration

```
extensions:
  qrz:
    script: qrz
    configuration:
      username: MYCALLSIGN
      passwort: sicrit
```

Request	Purpose	Parameters	Reply
lookup	Lookup a callsign in the QRZ.COM database or the local cache	**callsign** The callsign to be looked up	An array containing the following fields: **call** The callsign **name** The name **fname** The surname **addr2** Address information **country** The country **source** Either `qrz` or `cache`, depending whether the information was retrieved from QRZ.com or taken from the local cache

The qrz extension does not send status updates, even if you sent a **listen** request.

dxcluster

The dxcluster extension can be used for DXCluster spots as well as for SOTA-Cluster spots.

Example Configuration

The example configuration will run two instances of the dxcluster extension, one to get DXCluster spots and one to get SOTA spots:

```
extensions:
  dxcluster:
    script: dxcluster
      configuration:
        host: wr3d.dxcluster.net
        port: 7300
        callsign: MYCALLSIGN

  sotacluster:
```

```
    script: dxcluster
      configuration:
        host: cluster.sota.co.uk
        port: 7300
        callsign: MYCALLSIGN
        source: sotacluster
```

Request	Purpose	Parameters	Reply
getSpots	Get recent spots	*maxSpots* The maximum number of spots to return	An array named `dxcluster` or whatever is configured in the **source** config parameter containing spot information as follows: **spotter** Who spotted the callsign **frequency** The frequency **spotted** The callsign being spotted **message** The optional message **time** The timestamp in UTC

When you send the **listen** request, the dxcluster extension will send new spots automatically as they arrive in a JSON package with the same fields as for the **getSpots** request.

tasmota

The tasmota extension can control power plugs with the alternative Tasmota firmware found at https://github.com/arendst/Tasmota.

Example configuration

```
extensions:
  tasmota:
    script: tasmota
```

```
      configuration:
        address: 192.168.4.1
```

Request	Purpose	Parameters	Reply
power	Query power state		**power** Either `on` or `off`
power	Set power state	**state** Either `on` or `off`	**power** Either `on` or `off`