

The trx-control Integration Guide

Marc Balmer HB9SSB <info@hb9ssb.ch>

Table of Contents

General Principles	2
Units	3
Frequencies	3
Altitudes	3
Position data	3
Request format	4
Destinations	4
General Requests	5
Controlling Transceivers	6
GPIO	8
GPS/NMEA	9
Extensions	12
Common commands	12
config	12
ping	12
keepalive	13
memory	13
hamqth	15
qrz	19
dxcluster	20
tasmota	21

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 response.

Units

Frequencies

Frequencies are expressed as integers in hertz. So 14.285 MHz must be expressed as 14285000.

Altitudes

Altitudes are expressed as integers in meters above sea level.

Position data

Position data is expressed as degrees with sign.

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

Usually parameters are sent back in a response.

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

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



Replies or status updates from a transceiver controller will always contain the field **from** which contains the name of the destination.

GPIO

GPS/NMEA

If configured, the `nmea` destination can be used to query the current fix data, i.e. date, time, position, locator etc.

Request	Purpose	Parameters	Response
get-fix	Get the last fix received from the connected device.		<p>fix An array with the following fields:</p> <p>date The current date in the format DD.MM.YYYY.</p> <p>time The current UTC time in the format HH:MM:SS</p> <p>longitude The longitude in degrees.</p> <p>latitude The latitude in degrees.</p> <p>altitude The altitude in m above sea level.</p> <p>variation The magnetic variation in degrees.</p> <p>course The course in degrees.</p> <p>speed</p>

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	Response
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	Response
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	Response
ping	Check connection		response The string pong trxd An array containing: version The trxd version

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	Response
listen	Start sending keepalives		
unlisten	Stop sending keepalives		

memory

The memory extension manages memories and memory groups that are stored in a PostgreSQL database.

Example configuration

```
extensions:  
  memory:  
    script: memory  
  configuration:  
    connStr: dbname=trx-control  
    datestyle: German
```

Request	Purpose	Parameters	Response
getTopLevel	Get the toplevel list of memory groups and memory entries		<p>An array of entries containing the following fields:</p> <p>entry Either group or memory</p> <p>id The id (UUID) of the memory or memory group</p> <p>name Name of the entry</p> <p>supplement Any supplement to the name</p> <p>descr A longer description of the entry</p> <p>The following entries are only present when entry is set to memory:</p> <p>type The type of the memory entry, can be quick, call, channel, or, repeater</p> <p>rx The RX frequency</p> <p>tx The TX frequency</p> <p>shift A TX shift, including sign</p> <p>mode The operating mode</p>

Request	Purpose	Parameters	Response
getMemoryGroup	Get the entries of a specific memory group	group The id (UUID) of the memory group to retrieve	Identical to the getTopLevel request
addMemoryGroup	Add a new memory group	name The name of the new group supplement A supplement to the name descr A longer description of the group	id The id (UUID) of the newly created group
addMemory	Add a new memory entry	name The name of the new memory supplement A supplement to the name descr A longer description of the memory entry rx The RX frequency tx The TX frequency shift A frequency shift including sign, if any mode The operating mode	id The id (UUID) of the newly created memory entry

hamqth

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

Example configuration

```
extensions:  
  hamqth:  
    script: hamqth  
    configuration:  
      username: MYCALLSIGN  
      password: sicrit
```


Request	Purpose	Parameters	Response
lookup	Lookup a callsign in the HamQTH.com database or the local cache	callsign The callsign to be looked up	<p>An array containing the following fields:</p> <p>callsign The callsign</p> <p>nick The name used on the air</p> <p>qth Callsign's QTH</p> <p>country The country</p> <p>adif ADIF ID of country related to callsign</p> <p>itu ITU zone</p> <p>cq CQ (WAZ) zone</p> <p>grid Station locator</p> <p>adr_city City address</p> <p>adr_zip Postcode</p> <p>adr_country Country related to the address</p> <p>adr_adif ADIF ID of address country</p> <p>lotw Y - uses LoTW, N - doesn't use LoTW, ? - unknown</p>

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

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](https://shop.qrz.com/collections/subscriptions) 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  
      password: sicrit
```

Request	Purpose	Parameters	Response
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	Response
getSpots	Get recent spots	<i>maxSpots</i> 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	Response
power	Query power state		power Either on or off
power	Set power state	state Either on or off	power Either on or off