DMIX128 WebSocket Protocol Documentation

Overview

The DMIX128 uses a WebSocket-based communication protocol built on top of **Socket.IO** for real-time bidirectional communication between the web GUI and the C++ backend server (`mixer-web`).

Server: C++ application **Client**: JavaScript GUI **Default Port**: 80 (HTTP/WebSocket)
Secondary Port: 7712 (TCP for VU meters and direct communication)

Connection Establishment

Socket.IO Connection

The client connects using Socket.IO v4+ with the following configuration:

```
io(serverAddress, {
    timeout: 4000,
    reconnectionDelay: 500,
    reconnectionDelayMax: 2500,
    randomizationFactor: 0.05,
    transports: ['websocket']
})
```

Connection Flow

1. **HTTP Polling Handshake** (optional, Socket.IO upgrade mechanism):

   ```
   GET /socket.io/?transport=polling
   ```

   Response:

   ```
   0{"sid":"<20-char-random-id>","upgrades":
   ["websocket"],"pingInterval":25000,"pingTimeout":20000,"maxPayload":1000000}
   ```

2. **WebSocket Upgrade**:
   - Client connects via WebSocket
   - Server sends session initialization:

     ```
     0{"sid":"<session-id>","upgrades":
     [],"pingInterval":25000,"pingTimeout":5000}
     ```

```
40
```

3. **Client Requests Initial Data**:

```
{"cmd":"INIT","id":12345,"args":{}}
```

4. **Server Responds with Complete State**:

```
{"cmd":"INIT","id":12345,"data":{<all mixer parameters>}}
```

Message Format

Socket.IO Message Wrapping

All messages are wrapped in Socket.IO packet format:

From Client:

```
42["message",<JSON_PAYLOAD>]
```

From Server to Client:

```
42["message",<JSON_PAYLOAD>]
```

Heartbeat:

- Client sends: `"2"`
- Server responds: `"3"`

JSON Payload Structure

1. Command Messages (with response expected)

```
{
  "cmd": "<COMMAND_NAME>",
  "id": <unique_numeric_id>,
  "args": {
    <command_arguments>
  }
}
```

2. Command Responses

```
{
  "cmd": "<COMMAND_NAME>",
  "id": <matching_request_id>,
  "data": <response_data>
}
```

3. State Updates (key-value pairs, no response needed)

```
{
  "key1": value1,
  "key2": value2,
  ...
}
```

Core Commands

INIT - Initialize/Get Complete State

Request:

```
{
  "cmd": "INIT",
  "id": 12345,
  "args": {}
}
```

Response:

```
{
  "cmd": "INIT",
  "id": 12345,
  "data": {
    "show": "Default",
    "snapshot": "",
    "bpm": 120,
    "i.0.mix": 0.7647,
    "i.0.pan": 0.5,
    "i.0.solo": 0,
    "i.0.mute": 0,
    ... // All mixer parameters (thousands of keys)
  }
}
```

VU - VU Meter Data (Broadcast, No Response)

From Server:

```
{
  "cmd": "VU",
  "id": 0,
  "data": {
    "in.pre": "<base64_encoded_vu_data>",
    "in.post": "<base64_encoded_vu_data>",
    "in.comp": "<base64_encoded_vu_data>",
    "fx.pre": "<base64_encoded_vu_data>",
    ... // VU data for all channel types
  }
}
```

VU Data Structure:

- Base64-encoded binary data
- Each byte represents level for one channel (0-255)
- Pickup types: `pre`, `post`, `gain`, `comp`, `compin`, `compout`, `mcomp`, `gate`, `gatein`, `gateout`, `dynin`, `dynout`, `ds`, `dsin`, `dsout`, `eqdyn`

RTA - Real-Time Analyzer Data

From Server:

```
{
  "cmd": "RTA",
  "id": 0,
  "data": {
    "ch": "i.5",
    "rta": "<base64_encoded_spectrum_data>"
  }
}
```

BMSG - Broadcast Message (Forwarded to All Clients)

Request:

```
{
  "cmd": "BMSG",
  "id": 0,
  "args": {
    "custom_field": "value",
    ... // Any custom data
```

```
  }
}
```

Behavior: Server broadcasts this message to all connected clients except sender.

Show & Snapshot Management

SHOW_LIST - List All Shows

Request:

```
{
  "cmd": "SHOW_LIST",
  "id": 123,
  "args": {}
}
```

Response:

```
{
  "cmd": "SHOW_LIST",
  "id": 123,
  "data": ["Show 1", "Show 2", "Default"]
}
```

SHOW_SAVE - Save Current State as Show

Request:

```
{
  "cmd": "SHOW_SAVE",
  "id": 124,
  "args": {
    "name": "My Show"
  }
}
```

SHOW_LOAD - Load a Show

Request:

```
{
  "cmd": "SHOW_LOAD",
  "id": 125,
```

```
  "args": {
    "name": "My Show"
  }
}
```

Response: Returns complete show data (similar to INIT)

SHOW_DELETE - Delete a Show

Request:

```
{
  "cmd": "SHOW_DELETE",
  "id": 126,
  "args": {
    "name": "My Show"
  }
}
```

SHOW_RENAME - Rename a Show

Request:

```
{
  "cmd": "SHOW_RENAME",
  "id": 127,
  "args": {
    "name": "Old Name",
    "newName": "New Name"
  }
}
```

SNAP_LIST - List Snapshots in Show

Request:

```
{
  "cmd": "SNAP_LIST",
  "id": 128,
  "args": {
    "show": "My Show"
  }
}
```

Response:

```
{
  "cmd": "SNAP_LIST",
  "id": 128,
  "show": "My Show",
  "args": {"show": "My Show"},
  "data": ["Snapshot 1", "Snapshot 2", "Intro"]
}
```

SNAP_SAVE - Save Snapshot

Request:

```
{
  "cmd": "SNAP_SAVE",
  "id": 129,
  "args": {
    "show": "My Show",
    "name": "Snapshot Name"
  }
}
```

SNAP_LOAD - Load Snapshot

Request:

```
{
  "cmd": "SNAP_LOAD",
  "id": 130,
  "args": {
    "show": "My Show",
    "name": "Snapshot Name"
  }
}
```

SNAP_DELETE - Delete Snapshot

Request:

```
{
  "cmd": "SNAP_DELETE",
  "id": 131,
  "args": {
    "show": "My Show",
    "name": "Snapshot Name"
  }
}
```

SNAP_RENAME - Rename Snapshot

Request:

```
{
  "cmd": "SNAP_RENAME",
  "id": 132,
  "args": {
    "show": "My Show",
    "name": "Old Name",
    "newName": "New Name"
  }
}
```

Preset Management

PRESET_LIST - List Presets of Type

Request:

```
{
  "cmd": "PRESET_LIST",
  "id": 200,
  "args": {
    "type": "eq"
  }
}
```

Response:

```
{
  "cmd": "PRESET_LIST",
  "id": 200,
  "type": "eq",
  "args": {"type": "eq"},
  "data": ["Vocal EQ", "Bass Heavy", "Acoustic"]
}
```

Preset Types:

- `"eq"` - Equalizer presets
- `"comp"` - Compressor presets
- `"gate"` - Gate presets
- `"geq"` - Graphic EQ presets

- `"ds"` - De-esser presets
- `"fx"` - Effects presets
- `"mcomp"` - Multiband compressor presets

PRESET_READ - Load Preset

Request:

```
{
  "cmd": "PRESET_READ",
  "id": 201,
  "args": {
    "type": "eq",
    "name": "Vocal EQ"
  }
}
```

Response:

```
{
  "cmd": "PRESET_READ",
  "id": 201,
  "data": {
    "eq.b1.freq": 100,
    "eq.b1.gain": 2.5,
    "eq.b1.q": 1.88,
    ... // Preset parameters
  }
}
```

PRESET_WRITE - Save Preset

Request:

```
{
  "cmd": "PRESET_WRITE",
  "id": 202,
  "args": {
    "type": "eq",
    "name": "My EQ",
    "data": {
      "eq.b1.freq": 100,
      "eq.b1.gain": 3.0,
      ... // Parameters to save
    }
  }
}
```

PRESET_DELETE - Delete Preset

Request:

```
{
  "cmd": "PRESET_DELETE",
  "id": 203,
  "args": {
    "type": "eq",
    "name": "Unused Preset"
  }
}
```

PRESET_RENAME - Rename Preset

Request:

```
{
  "cmd": "PRESET_RENAME",
  "id": 204,
  "args": {
    "type": "eq",
    "name": "Old Name",
    "newName": "New Name"
  }
}
```

Network Configuration

NETCFG - Get Network Configuration

Request:

```
{
  "cmd": "NETCFG",
  "id": 300,
  "args": {}
}
```

Response:

```
{
  "cmd": "NETCFG",
  "id": 300,
```

```
  "data": {
    "hs_enable": 1,
    "hs_ssid": "MIXER",
    "hs_pass": "12345678",
    "hs_band5ghz": 0,
    "hs_country": "US",
    "hs_channel": 9,
    "wifi_join": 0,
    "wifi_ssid": "HOME WIFI",
    "wifi_pass": "password",
    "wifi_not_dhcp": 0,
    "wifi_ip": "192.168.1.50",
    "lan_enable": 1,
    "lan_not_dhcp": 0,
    "lan_ip": "10.10.2.1",
    ... // More network settings
  }
}
```

SETNETCFG - Update Network Configuration

Request:

```
{
  "cmd": "SETNETCFG",
  "id": 301,
  "args": {
    "lan_ip": "10.10.3.1",
    "hs_ssid": "MY_MIXER",
    ... // Fields to update
  }
}
```

GUI Control Commands

SELECT_CHANNEL - Select a Channel (Broadcast)

Request:

```
{
  "cmd": "SELECT_CHANNEL",
  "args": {
    "channel": "i.5"
  }
}
```

Behavior: Broadcasted to all clients to sync channel selection.

SCROLL_TO_CHANNEL - Scroll to Channel (Broadcast)

Request:

```
{
  "cmd": "SCROLL_TO_CHANNEL",
  "args": {
    "channel": "a.2"
  }
}
```

SWITCH_PAGE - Switch GUI Page (Broadcast)

Request:

```
{
  "cmd": "SWITCH_PAGE",
  "args": {
    "page": 3
  }
}
```

SYNC_ID - Sync Selected Channel Across Clients

Request:

```
{
  "cmd": "SYNC_ID",
  "id": 0,
  "args": {
    "id": 1,
    "strip": "i.7",
    "vca": 0
  }
}
```

State Updates (Parameter Changes)

Setting Individual Parameters

Client → Server:

```
{
  "i.0.mix": 0.85,
```

```
  "i.0.pan": 0.3,
  "a.2.mix": 0.65
}
```

Server → All Clients (Broadcast):

```
{
  "i.0.mix": 0.85,
  "i.0.pan": 0.3,
  "a.2.mix": 0.65
}
```

Fast Path Optimization

For single numeric value updates, the server supports a fast path:

```
{"i.0.mix": 0.75}
```

The server detects this pattern and broadcasts without full JSON parsing.

Parameter Naming Convention

Parameters use a hierarchical dot notation:

Structure

```
<prefix>.<index>.<category>.<parameter>
```

Channel Prefixes

Prefix	Type	Description
`i`	Input	Input channels (64 channels)
`a`	Aux	Aux buses (24 buses)
`f`	FX	FX buses (8 buses)
`s`	Sub	Subgroup buses (8 buses)
`r`	Return	FX returns (8 channels)
`v`	VCA	VCA/DCA groups (16 groups)
`x`	Matrix	Matrix outputs (8 outputs)

Prefix	Type	Description
`X`	Matrix Send	Matrix send channels (12 channels)
`m`	Master	Master LR bus
`p`	Player	Audio player
`c`	Click	Click/Metronome
`d`	DAW	DAW/Tape channels
`t`	Multitrack	Multitrack returns
`h`	Hardware	Hardware input gain
`osc`	Oscillator	Test oscillator

Parameter Categories

Mix Parameters

- `.mix` - Fader level (0.0 to 1.0, maps to -∞ to +6dB)
- `.pan` - Pan position (0.0=full left, 0.5=center, 1.0=full right)
- `.solo` - Solo state (0 or 1)
- `.mute` - Mute state (0 or 1)
- `.unmute` - Temporary unmute flag
- `.mgmask` - Mute group assignment (bitmask)
- `.submask` - Sub group routing (bitmask)
- `.vcamask` - VCA assignment (bitmask)

Input Processing

- `.src` - Input source assignment (e.g., `"h.0"` for hardware input 1)
- `.trim` - Input trim (-24 to +24 dB, normalized 0-1)
- `.gain` - Hardware preamp gain (-10 to +40 dB, normalized 0-1)
- `.phantom` - Phantom power (0=off, 1=+48V, 2=Hi-Z)
- `.invert` - Phase invert (0 or 1)
- `.delay` - Channel delay in milliseconds (0-250)
- `.delayOn` - Delay enable (0 or 1)

Equalizer (`.eq.`)

- `.eq.bypass` - EQ bypass (0 or 1)
- `.eq.b1.type` - Band 1 type (`"bell"`, `"hpf"`, `"lpf"`, `"hshelf"`, `"lshelf"`, `"notch"`)
- `.eq.b1.freq` - Band 1 frequency (20-20000 Hz, normalized 0-1)
- `.eq.b1.gain` - Band 1 gain (-15 to +15 dB, normalized 0-1)
- `.eq.b1.q` - Band 1 Q factor (0.1-15, normalized 0-1)
- `.eq.b1.bypass` - Band 1 bypass
- `.eq.b2.*`, `.eq.b3.*`, `.eq.b4.*` - Bands 2, 3, 4
- `.eq.hp.freq` - High-pass filter frequency

- `.eq.hp.bypass` - High-pass bypass
- `.eq.hp.slope` - Filter slope (6, 12, or 24 dB/octave)
- `.eq.lp.freq` - Low-pass filter frequency
- `.eq.lp.bypass` - Low-pass bypass

Dynamic EQ (`.eq.b1.` for dynamic bands)

- `.eq.b1.dynamic` - Enable dynamic EQ for this band
- `.eq.b1.thresh` - Threshold (-80 to 0 dB)
- `.eq.b1.ratio` - Compression ratio (1:1 to 20:1)
- `.eq.b1.attack` - Attack time (0.5-500 ms)
- `.eq.b1.release` - Release time (10-2000 ms)
- `.eq.b1.sc` - Sidechain source (empty or channel prefix)
- `.eq.b1.slot` - Resource slot allocation
- `.eq.b1.filters` - Sidechain filters enable
- `.eq.b1.hpfFreq` - Sidechain high-pass frequency
- `.eq.b1.lpfFreq` - Sidechain low-pass frequency

Graphic EQ (`.geq.`)

- `.geq.bypass` - GEQ bypass
- `.geq.peak.0` to `.geq.peak.30` - 31 band levels (-15 to +15 dB)
- `.geq.hp.freq` - High-pass filter
- `.geq.hp.bypass` - High-pass bypass
- `.geq.lp.freq` - Low-pass filter
- `.geq.lp.bypass` - Low-pass bypass
- `.geq.slot` - Resource slot allocation

Compressor (`.comp.`)

- `.comp.bypass` - Compressor bypass
- `.comp.thresh` - Threshold (-80 to 0 dB, normalized 0-1)
- `.comp.ratio` - Ratio (1:1 to 20:1, normalized 0-1)
- `.comp.gain` - Makeup gain (0 to +24 dB, normalized 0-1)
- `.comp.attack` - Attack time (0.5-500 ms, normalized 0-1)
- `.comp.release` - Release time (10-2000 ms, normalized 0-1)
- `.comp.hold` - Hold time (0-500 ms, normalized 0-1)
- `.comp.knee` - Knee (0=hard, 1=soft, or 0-60 dB)
- `.comp.sc` - Sidechain source
- `.comp.scListen` - Sidechain monitor
- `.comp.filters` - Sidechain filters enable
- `.comp.hp.freq` - Sidechain HPF
- `.comp.lp.freq` - Sidechain LPF
- `.comp.type` - Compressor type (`"clean"`, `"vca"`, `"fet"`, `"opto"`)
- `.comp.model` - Compressor model (0=main, 22=DNA22, 34=DNA34)
- `.comp.limiter` - Limiter mode

Gate (`.gate.`)

- `.gate.bypass` - Gate bypass
- `.gate.thresh` - Threshold (-80 to 0 dB)
- `.gate.depth` - Range/Depth (-100 to 0 dB)
- `.gate.attack` - Attack time (0.5-500 ms)
- `.gate.hold` - Hold time (0-500 ms)
- `.gate.release` - Release time (10-2000 ms)
- `.gate.knee` - Knee (0=hard, 1=soft)
- `.gate.sc` - Sidechain source
- `.gate.scListen` - Sidechain monitor
- `.gate.filters` - Sidechain filters enable
- `.gate.hp.freq` - Sidechain HPF
- `.gate.lp.freq` - Sidechain LPF

De-esser (`.ds.`)

- `.ds.bypass` - De-esser bypass
- `.ds.thresh` - Threshold (-96 to 0 dB)
- `.ds.freq` - Center frequency (20-20000 Hz)
- `.ds.q` - Q factor (0.1-15)
- `.ds.range` - Reduction range (0 to -15 dB)

Multiband Compressor (`.mcomp.`)

- `.mcomp.bypass` - Bypass
- `.mcomp.gain` - Overall makeup gain
- `.mcomp.knee` - Knee
- `.mcomp.slot` - Resource slot (0=off, 1-12)
- `.mcomp.0.freq` to `.mcomp.3.freq` - Crossover frequencies (4 bands)
- `.mcomp.0.bypass` to `.mcomp.3.bypass` - Band bypasses
- `.mcomp.0.solo` to `.mcomp.3.solo` - Band solo
- `.mcomp.0.thresh` to `.mcomp.3.thresh` - Band thresholds
- `.mcomp.0.ratio` to `.mcomp.3.ratio` - Band ratios
- `.mcomp.0.gain` to `.mcomp.3.gain` - Band makeup gains
- `.mcomp.0.attack` to `.mcomp.3.attack` - Band attack times
- `.mcomp.0.release` to `.mcomp.3.release` - Band release times
- `.mcomp.0.range` to `.mcomp.3.range` - Band ranges

Aux/FX Sends

- `.aux.0.value` to `.aux.23.value` - Send levels (24 aux sends)
- `.aux.0.pan` to `.aux.23.pan` - Send panning
- `.aux.0.post` - Pre/post fader (0=pre, 1=post, 2=pre-EQ)
- `.aux.0.mute` - Send mute

- `.aux.0.followpan` - Follow main pan
- `.fx.0.value` to `.fx.7.value` - FX send levels (8 FX sends)
- `.fx.0.post` - Pre/post fader
- `.fx.0.mute` - Send mute

FX Parameters (`.f.X.`)

- `.f.0.fxtype` - Effect type (0=Reverb, 1=Room, 2=Delay, 3=Chorus, 4=DX480)
- `.f.0.fxbypass` - Effect bypass
- `.f.0.fxdata` - Effect parameters (JSON object)

```
{
  "type": 0,
  "v0": 0.0,    // Param 1 (varies by effect)
  "v1": 0.5,    // Param 2
  "v2": 0.7,    // Param 3
  "v3": 0.3,    // Param 4
  "v4": 0.8,    // Param 5
  "v5": 0.2,    // Param 6
  "v6": 0       // Param 7
}
```

Reverb/Room Parameters (type 0/1):

- `v0` - Pre-delay (0-50 ms)
- `v1` - Time (300-8000 ms)
- `v2` - Damping/HF (0-100%)
- `v3` - LF (0-100%)
- `v4` - LPF (400-20000 Hz)
- `v5` - HPF (20-5000 Hz)

Delay Parameters (type 2):

- `v0` - Time (100-2000 ms)
- `v1` - Subdivision (1/32 to 2/1)
- `v2` - Feedback (0-100%)
- `v3` - LPF (20-20000 Hz)

Chorus Parameters (type 3):

- `v0` - Detune (-100 to +100 cents)
- `v1` - Density (0-100%)
- `v2` - LPF (400-20000 Hz)

DX480 Reverb Parameters (type 4):

- `v0` - Dry/Wet (0-100%)

- `v1` - Pre-delay (0-250 ms)
- `v2` - Decay (300-8000 ms)
- `v3` - Sus/Atk Mix (0-100%)
- `v4` - Attack Dump (0-100%)
- `v5` - Sustain Dump (0-100%)
- `v6` - Density (0 or 1)

Insert Effects

- `.insert.enable` - Insert enable
- `.insert.send` - Insert send routing
- `.insert.return` - Insert return routing

Processing Flow Order

- `.flow` - Processing order string (e.g., `"gcqdmei"`)
  - `g` = Gate
  - `c` = Compressor
  - `q` = EQ
  - `d` = De-esser
  - `m` = Multiband Compressor
  - `e` = Graphic EQ
  - `i` = Insert

Channel Metadata

- `.name` - Channel name (string)
- `.color` - Color index (0-N)
- `.linked` - Link to channel (channel prefix or empty)
- `.linkedLR` - Stereo link position (0=none, 1=left, 2=right)
- `.rec` - Recording enable for multitrack
- `.cue` - Monitor cue assignment (0=CUE1, 1=CUE2)

Routing

- `.src` - Input source (e.g., `"h.0"`, `"madi.5"`, `"net.10"`)
- `.scsrc` - Soundcheck source
- `out.hw.0` to `out.hw.23` - Analog output routing
- `out.aes.0` to `out.aes.3` - AES output routing
- `out.hp.0` to `out.hp.3` - Headphone output routing
- `out.madi.0` to `out.madi.63` - MADI output routing
- `out.exp.0` to `out.exp.63` - Expansion output routing

Global Settings

Player Settings

- `playState` - Player state (0=stop, 1=play, 2=pause)
- `recState` - Recording state
- `playPosition` - Current position in milliseconds
- `trackLength` - Track length in milliseconds
- `trackCurrent` - Current track name
- `trackNext` - Next track name
- `trackPrev` - Previous track name
- `bpm` - Global BPM (60-600)

Click/Metronome

- `c.0.play` - Click playback (0 or 1)
- `c.0.bpm` - Click BPM
- `c.0.bpmGlobal` - Use global BPM
- `c.0.level` - Click level (-36 to 0 dB)
- `c.0.sound` - Click sound (`"click"`, `"hihat"`, `"sticks"`, `"wood"`)
- `c.0.signature` - Time signature (`"1/4"`, `"2/4"`, `"3/4"`, `"4/4"`, `"5/4"`, `"6/4"`, `"7/4"`, `"6/8"`, `"9/8"`, `"12/8"`)
- `c.0.accent` - Accent first beat

Monitoring/Solo Settings

- `settings.solo.routing` - Solo routing (0=HP, 1=MAIN+HP)
- `settings.solo.type` - Solo type (0=PFL, 1=AFL)
- `settings.solo.mode` - Solo mode (0=MIX, 1=LAST)
- `settings.solo.followselect` - Auto-solo on selection
- `settings.solo.inplace` - Solo-in-place mode
- `settings.pfldim` - PFL dim amount (-40 to 0 dB)
- `settings.afldim` - AFL dim amount (-40 to 0 dB)
- `settings.hpvolume` - Headphone volume (-∞ to +6 dB)
- `settings.solo.delay` - Solo delay (0-500 ms)
- `settings.solo.delayOn` - Solo delay enable

View/Mute Groups

- `mgmask` - Global mute groups active state (bitmask)
  - Bit 0-5: Mute groups 1-6
  - Bit 23: Mute FX
  - Bit 24: Mute All
- `mgName.0` to `mgName.5` - Mute group names
- `vgName.0` to `vgName.5` - View group names
- `vg.0` to `vg.5` - View group members (array of channel IDs)

Snapshot Isolation (Scene Filtering)

- `iso.enable` - Isolation filtering enable
- `iso.src` - Load source routing (0=no, 1=all, 2=selected)
- `iso.in` - Load input processing

- `iso.gate` - Load gate
- `iso.comp` - Load compressor
- `iso.eqds` - Load EQ and de-esser
- `iso.mixpan` - Load mix/pan/solo/mute
- `iso.sublr` - Load sub routing
- `iso.auxsend` - Load aux sends
- `iso.fxsend` - Load FX sends
- `iso.vca` - Load VCA assignments
- `iso.mute` - Load mute states
- `iso.chan` - Load channel metadata
- `iso.fxr0` to `iso.fxr3` - Load FX rack slots
- `iso.channels` - Selected channels for isolation (comma-separated)

Resource Allocation

- `res.dyneq` - Dynamic EQ instances used (max 64)
- `res.peq` - EQ bands allocated (max 64)
- `res.geq` - Graphic EQs used (max 12)
- `res.mcomp` - Multiband compressors used (max 12)

System Info

- `version.fw` - Firmware version
- `version.fpga` - FPGA version
- `eth_ip` - Ethernet IP address
- `wifi_ip` - WiFi IP address

Remote Device Management (Networked Mixers)

- `slot.0` to `slot.5` - Connected remote device info (JSON)

```
{
  "model": "Dmix128",
  "name": "mixer-foh",
  "network_role": "shared",
  "gain_role": "follower"
}
```

- `slot.0.role` to `slot.5.role` - Device role (JSON)
- `slot.0.hostname` to `slot.5.hostname` - Device hostname
- `slot.0.madiMode` - MADI mode (`"madi32"`, `"madi56"`, `"madi64"`)
- `slot.0.dspsrate` - Sample rate (`"48k"`, `"96k"`)
- `slot.0.clksrc` - Clock source (JSON)
- `slot.0.ipmode` - IP mode (JSON)
- `slot.0.h.src` - Slot analog input routing
- `slot.0.h.gain` - Slot analog gain
- `slot.0.h.phantom` - Slot phantom power

Special Client Features

No VU/RTA Data Flag

Client can opt out of receiving VU and RTA data:

```
"novurta"
```

Sent as a bare message (Socket.IO wrapped):

```
42["message","novurta"]
```

TCP Direct Connection (Port 7712)

An alternative TCP connection for local/internal processes (e.g., VU meter sender):

Connection: TCP to `127.0.0.1:7712`

Protocol: JSON lines (newline-delimited)

Messages: Same JSON format as WebSocket, but without Socket.IO wrapping

Example:

```
{"cmd":"INIT","id":0,"args":{}}\n
{"cmd":"VU","id":0,"data":{...}}\n
```

Fast Init Command:

```
CMD:INIT\n
```

Data Types and Value Ranges

Normalized Values (0.0 - 1.0)

Most continuous parameters are normalized to 0.0-1.0 range and converted by client/server:

Parameter	Range	Normalized	Conversion
Mix/Fader	-∞ to +6 dB	0.0 - 1.0	Logarithmic
Pan	-100 to +100 cents	0.0 - 1.0	Linear

Parameter	Range	Normalized	Conversion
Gain	-10 to +40 dB	0.0 - 1.0	Linear
Trim	-24 to +24 dB	0.0 - 1.0	Linear
EQ Frequency	20 to 20000 Hz	0.0 - 1.0	Logarithmic
EQ Gain	-15 to +15 dB	0.0 - 1.0	Linear
EQ Q	0.1 to 15	0.0 - 1.0	Logarithmic
Comp Threshold	-80 to 0 dB	0.0 - 1.0	Linear
Comp Ratio	1:1 to 20:1	0.0 - 1.0	Logarithmic
Comp Gain	0 to +24 dB	0.0 - 1.0	Linear
Comp Attack	0.5 to 500 ms	0.0 - 1.0	Logarithmic
Comp Release	10 to 2000 ms	0.0 - 1.0	Logarithmic

Note: The "0 dB" fader position (unity gain) is at value `0.7647058823531552`

Error Handling

Command Errors

When a command fails, the server may:

1. Not send a response (timeout on client)
2. Send an error response (implementation-specific)
3. Log error to server console

Connection Loss

Client implements automatic reconnection with:

- Initial delay: 500ms
- Maximum delay: 2500ms
- Randomization factor: 0.05

On reconnection, client sends `INIT` command to resync state.

Performance Optimizations

Message Broadcasting

- State updates are broadcasted to all clients except the sender
- VU/RTA messages use a separate fast path
- Command messages `SELECT_CHANNEL`, `SCROLL_TO_CHANNEL`, `SWITCH_PAGE` are directly broadcasted without processing

Fast Path Detection

Server uses pattern matching to detect and optimize:

- Single key-value updates
- VU meter updates
- RTA updates
- INIT requests
- BMSG broadcasts

VU Meter Optimization

- VU data sent as Base64-encoded binary (not JSON)
- Clients can opt out with `"novurta"` message
- Separate TCP port for high-frequency VU updates from DSP

Security Considerations

- No authentication by default
- Network isolation recommended for production
- JWT support available but optional (`$CFG.crypto`)
- Mixer state auto-saved to disk every 5 seconds

Example Client Implementation (JavaScript)

```
// Connect
const socket = io('ws://192.168.1.100', {
  timeout: 4000,
  reconnectionDelay: 500,
  transports: ['websocket']
});

// Handle connection
socket.on('connect', () => {
  console.log('Connected');

  // Request initial state
  socket.send({
    cmd: 'INIT',
    id: Math.floor(Math.random() * 2147483647),
    args: {}
  });
});

// Handle messages
socket.on('message', (data) => {
  if (data.cmd === 'INIT') {
    console.log('Got initial state:', data.data);
  } else if (data.cmd === 'VU') {
```

```
    // Process VU meters
    updateMeters(data.data);
  } else {
    // Handle parameter updates
    for (let key in data) {
      updateParameter(key, data[key]);
    }
  }
});

// Set a parameter
socket.send({
  'i.0.mix': 0.85
});

// Send a command
socket.send({
  cmd: 'SNAP_LOAD',
  id: Math.floor(Math.random() * 2147483647),
  args: {
    show: 'My Show',
    name: 'Scene 1'
  }
});
```

Example Python Client

```
import socketio
import base64

sio = socketio.Client()

@sio.on('connect')
def on_connect():
    print('Connected')
    # Request initial state
    sio.send({
        'cmd': 'INIT',
        'id': 12345,
        'args': {}
    })

@sio.on('message')
def on_message(data):
    if isinstance(data, dict):
        if 'cmd' in data:
            if data['cmd'] == 'INIT':
                print(f"Got {len(data['data'])} parameters")
            elif data['cmd'] == 'VU':
                # Decode VU data
```

```
                vu_data = base64.b64decode(data['data']['in.post'])
                levels = list(vu_data)
                print(f"Channel 1 level: {levels[0]}")
        else:
            # Parameter update
            for key, value in data.items():
                print(f"{key} = {value}")

# Connect
sio.connect('http://192.168.1.100:80')

# Set input 1 fader to -10dB (approx 0.65)
sio.send({'i.0.mix': 0.65})

# Load a snapshot
sio.send({
    'cmd': 'SNAP_LOAD',
    'id': 54321,
    'args': {
        'show': 'Default',
        'name': 'Intro'
    }
})

sio.wait()
```

Appendix: Complete Channel Type Reference

Type	Prefix	Count	Description
Input	`i`	128	Main input channels (mic/line)
Aux	`a`	24	Aux bus masters
FX	`f`	8	Effects bus masters (stereo)
Sub	`s`	8	Subgroup bus masters (stereo)
FX Return	`r`	8	Effects return channels (stereo)
VCA	`v`	16	VCA/DCA groups
Matrix	`x`	8	Matrix outputs
Matrix Send	`X`	12	Matrix send channels
Master	`m`	1	Master LR bus (stereo)
Player	`p`	1	Audio player (stereo)
Click	`c`	1	Click/Metronome (stereo)
DAW	`d`	Variable	DAW/Tape channels

Type	Prefix	Count	Description
Multitrack	`t`	Variable	Multitrack returns
Oscillator	`osc`	1	Test oscillator

Appendix: Source Type Reference

Type	Prefix	Description
Hardware Input	`h`	Analog/mic inputs (32 channels)
AES	`aes`	AES3 digital inputs (4 stereo pairs)
MADI	`madi`	MADI inputs (32/56/64 channels)
Expansion	`exp`	Expansion card inputs (64 channels)
Network	`net`	Network audio (64 channels)
DAW	`daw`	DAW/USB inputs
Player	`pl`	Audio player
Multitrack	`mlt`	Multitrack returns
Master	`m`	Master LR
Aux	`aux`	Aux bus
FX	`fx`	FX bus
Sub	`sub`	Subgroup
Monitor	`hp`	Monitor/headphone
Matrix	`mtx`	Matrix

Version History

Protocol Version: 1 (JSON-based)
Legacy Version: 0 (String-based with `^` separator - deprecated)

This documentation describes Protocol Version 1, which is the current production version.

Support & Contact

For technical support or questions about the DMIX128 protocol: Contact us through the website www.dmix128.com

Document generated from source code analysis - October 2025