

Brew

Brew

TETRA Homebrew Protocol

Protocol is based on WebSocket version 13 ([RFC 6455](#)) without extensions. It's highly recommended to use socket options [TCP_NODELAY](#) and [TCP_QUICKACK](#).

Endpoint, authentication

To establish a new connection client should make a HTTP GET call to get web socket's endpoint and pass authentication procedure. Authentication is based on *HTTP Digest Access Authentication* ([RFC 2831](#)). As a result of successful authentication server returns HTTP 200 and a URI to an endpoint to be used for WebSocket connection.

```
GET /brew/ HTTP/1.1
User-Agent: TETRAHS/012345

HTTP/1.1 200 OK

/brew/722e2b04-07ad-4976-ac55-75e845ae4d8a
```

Server supports following response codes: 101 (Switching protocols), 200 (OK), 301 (Moved), 401 (Unauthorized), 403 (Forbidden), 404 (Not found), 426 (Upgrade required), 429 (Too many requests), 500 (Internal server error).

HTTP header 'User-Agent' is mandatory!

WebSocket transport layer

Client should support following frame opcodes:

- **Close, ping, pong.** Ping and pong messages might contain a payload to measure delays.
- **Single-frame masked and unmasked binary messages.** Server always send unmasked messages, client may send masked messages.

Binary messages

Every message contains two-byte prefix:

1. Message class
2. Message type

All following data has non-aligned values in Little-Endian order

Subscriber control (mobility and affiliation, message class 0xf0)

```
#define BREW_SUBSCRIBER_DEREGISTER 0
#define BREW_SUBSCRIBER_REGISTER 1
#define BREW_SUBSCRIBER_REREGISTER 2
#define BREW_SUBSCRIBER_AFFILIATE 8
#define BREW_SUBSCRIBER_DEAFFILIATE 9

struct BrewSubscriberData
{
    uint8_t kind; // 0xf0
    uint8_t type; // BREW_SUBSCRIBER_*
    uint32_t number; // ISSI
    uint64_t time; // UNIX timestamp
    uint32_t fraction; // Nanoseconds
    uint32_t groups[0]; // GSSIs (BREW_SUBSCRIBER_AFFILIATE / BREW_SUBSCRIBER_DEAFFILIATE)
} __attribute__((packed));
```

TBD

Call control (message class 0xf1)

```
#define CALL_STATE_GROUP_TX 2 // | Simplified
#define CALL_STATE_GROUP_IDLE 3 // | Group Call
```

```

#define CALL_STATE_SETUP_REQUEST  4  // |           | Origin -> Receiver
#define CALL_STATE_SETUP_ACCEPT   5  // |           | Origin <- Receiver
#define CALL_STATE_SETUP_REJECT   6  // | General   | Origin <- Receiver
#define CALL_STATE_CALL_ALERT     7  // | Circuit Call | Origin <- Receiver
#define CALL_STATE_CONNECT_REQUEST 8  // |           | Origin <- Receiver
#define CALL_STATE_CONNECT_CONFIRM 9  // |           | Origin -> Receiver
#define CALL_STATE_CALL_RELEASE  10  // |           |

```

```

#define CALL_STATE_SHORT_TRANSFER 11 //

```

```

#define CALL_STATE_SIMPLEX_GRANTED 12 // | Simplex Call
#define CALL_STATE_SIMPLEX_IDLE    13 // | (on top of Circuit Call)

```

```

#define CALL_STATE_PDP_REQUEST    14 // |
#define CALL_STATE_PDP_ACCEPT     15 // | Packed Data
#define CALL_STATE_PDP_REJECT     16 // |
#define CALL_STATE_PDP_RELEASE    17 // |

```

```

#define PDP_FLAG_IPV4             (1 << 0)
#define PDP_FLAG_IPV6             (1 << 1)

```

```

struct BrewCircularCall
{
    uint32_t source;
    uint32_t destination;
    char number[32]; // External number (ASCII)
    uint8_t priority; // Call priority
    uint8_t service; // Table 14.79: Speech service
    uint8_t mode; // Table 14.52: Circuit mode type
    uint8_t duplex; // Duplex flag
    uint8_t method; // Table 14.62: Hook method
    uint8_t communication; // Table 14.54: Communication type
    uint8_t grant; // Table 14.80: Transmission grant
    uint8_t permission; // Table 14.81: Transmission request permission
    uint8_t timeout; // Table 14.50: Call time-out
    uint8_t ownership; // Table 14.38: Call ownership
    uint8_t queued; // Table 14.48: Call queued
} __attribute__((packed));

```

```

struct BrewCircularGrant

```

```

{
    uint8_t grant;        // Table 14.80: Transmission grant
    uint8_t permission;   // Table 14.81: Transmission request permission
} __attribute__((packed));

struct BrewShortData
{
    uint32_t source;
    uint32_t destination;
    char number[32];      // External number (ASCII)
} __attribute__((packed));

struct BrewPacketRequest
{
    uint32_t number;      // ISSI
    uint8_t flags;        // PDP_FLAG_*
    uint32_t profile;     // GSM 04.08 QoS Profile
    uint8_t authenticator[0]; // CHAP Challenge + CHAP Response
} __attribute__((packed));

struct BrewPacketContext
{
    uint8_t flags;        // PDP_FLAG_*
    in_addr_t v4;         // IPv4
    struct in6_addr v6;   // IPv6
    uint32_t profile;     // GSM 04.08 QoS Profile
} __attribute__((packed));

struct BrewCallControlData
{
    uint8_t kind;         // 0xf1
    uint8_t type;         // CALL_STATE_*
    uuid_t identifier;    // Call session UUID
    union
    {
        uint8_t cause;    // Table 14.55: Disconnect cause / GTP cause (ETSI TS 29.060 v3.9.0 7.7)
        struct BrewShortData data;
        struct BrewCircularGrant grant;
        struct BrewCircularCall circular;
        struct BrewGroupTransmission group;
    }
}

```

```
struct BrewPacketRequest request;
struct BrewPacketContext context;
};
} __attribute__((packed));
```

CALL_STATE_GROUP_TX

Length and contents: `offsetof(struct BrewCallControlData, data) + sizeof(struct BrewGroupTransmission)`

CALL_STATE_GROUP_IDLE, CALL_STATE_SETUP_REJECT, CALL_STATE_CALL_RELEASE

Length and contents: `offsetof(struct BrewCallControlData, data) + sizeof(uint8_t)`

CALL_STATE_SETUP_ACCEPT, CALL_STATE_CALL_ALERT

Length and contents: `offsetof(struct BrewCallControlData, data)`

CALL_STATE_SETUP_REQUEST, CALL_STATE_CONNECT_REQUEST

Length and contents: `offsetof(struct BrewCallControlData, data) + sizeof(struct BrewCircularCall)`

CALL_STATE_CONNECT_CONFIRM, CALL_STATE_SIMPLEX_GRANTED, CALL_STATE_SIMPLEX_IDLE

Length and contents: `offsetof(struct BrewCallControlData, data) + sizeof(struct BrewCircularGrant)`

CALL_STATE_SHORT_TRANSFER

Length and contents: `offsetof(struct BrewCallControlData, data) + sizeof(struct BrewShortData)`

CALL_STATE_PDP_REQUEST

Length and contents: `offsetof(struct BrewCallControlData, data) + sizeof(struct BrewPacketRequest) + len(CHAP_Challenge) + len(CHAP_Response)`

`BrewPacketRequest.authenticator[]` contains two CHAP packets: challenge and response ([RFC 1994](#)).

CALL_STATE_PDP_ACCEPT

Length and contents: `offsetof(struct BrewCallControlData, data) + sizeof(struct BrewPacketContext)`

CALL_STATE_PDP_REJECT, CALL_STATE_PDP_RELEASE

Length and contents: `offsetof(struct BrewCallControlData, data) + sizeof(uint8_t)`

Voice and data frames (message class 0xf2)

```
#define FRAME_TYPE_TRAFFIC_CHANNEL 0
#define FRAME_TYPE_SDS_TRANSFER 1
#define FRAME_TYPE_SDS_REPORT 2
#define FRAME_TYPE_DTMF_DATA 3
#define FRAME_TYPE_PACKET_DATA 4

struct BrewFrameData
{
    uint8_t kind; // 0xf2
    uint8_t type; // FRAME_TYPE_*
    uuid_t identifier; // Call session UUID
    uint16_t length; // Length of following data in bits
    uint8_t data[0];
} __attribute__((packed));
```

FRAME_TYPE_TRAFFIC_CHANNEL

The audio frame contains 60 ms of audio in format, based on STE defined at [ETSI TS 100 392-3-6 V1.1.1 \(2003-12\)](#)

- Octet 0: bit 7 = 1 (0x80), bits 6-2 = 4 STE control bits C1-C5, bits 1-0 = 0
- Octet 1+: 137 bits of ACELP subframe 1
- Octet 18+: 137 bits of ACELP subframe 2 (shifted by 1 bit right)
- Octet 35: bits 7-6 = the rest of subframe 2, bits 5-0 = unused, 0x3f

FRAME_TYPE_SDS_TRANSFER

Data field contains full SDS Type 4 PDU. So the first octet of data should contain protocol identifier, defined at [ETSI TS 100 392-2 V3.9.1 \(2019-01\)](#), Table 29.21

FRAME_TYPE_SDS_REPORT

This frame indicates SDS delivery and **NOT** SDS-TL delivery report. Data contains single-byte status code:

- 0 = Success

FRAME_TYPE_DTMF_DATA

Data field contains single DTMF code encoded in ASCII: '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '*', '#', 'A', 'B', 'C', 'D'

FRAME_TYPE_PACKET_DATA

Data field contains an IPv4 or IPv6 packet

Errors (message class 0xf3)

```
#define BREW_TYPE_MALFORMED      0
#define BREW_TYPE_RESTRICTED    1

struct BrewErrorData
{
    uint8_t kind;    // 0xf3
    uint8_t type;    // BREW_TYPE_*
    uint8_t data[0];
} __attribute__((packed));
```

BREW_TYPE_MALFORMED, BREW_TYPE_RESTRICTED

Data field contains malformed message content

Service messages (message class 0xf4)

```
struct BrewServiceData
{
    uint8_t kind;    // 0xf4
    uint8_t type;    //
    char data[0];    // JSON + NULL
} __attribute__((packed));
```

NULL-terminator at the end of JSON is mandatory!

Query subscribers (message type 1, client -> server)

```
[
    1234567 // List of ISSIs to query
]
```

On success server will respond with subscriber profiles list (see message type 2)

Subscriber profiles (message type 2, server -> client)

```
{
  "1234567" : // ISSI
  {
    "call" : "NOCALL",          // Callsign (optional)
    "active" : true,            // When blocked will be false (optional)
    "status" : 1,              // Registration status (when registered)
    "date" : "2025-03-10T00:00:00.000000Z" // Last update date and time (when registered)
  }
}
```

Revision #46

Created 16 February 2025 18:00:55 by R3ABM

Updated 16 March 2025 09:21:25 by R3ABM