Specficiation

The Protocol is designed to operate in OSI Model Level 5 and 6 [osi model specification]. In a hypothetical larger, commerical stage of the project, a definitive protocol would have to be solidified. The device implemented in (II) will be done over Wi-Fi, for example, but a more realistic situation would likely use bluetooth due to the limited range of connections, wifi connection not needing to be configured manually, etc.

Each connection is split into 3 states:

- **Phase I: Authentication**: The device connects to a nearby device and exchanges initial information used for encrpyting and identifying future messages
- **Phase II: Configuration**: Period of initial communication where details such as sampling rates, feature ability and parameter options enumerated and agreed upon.
- **Phase III: Steady State Operation**: Normal mode of operaton where delivery commands are processed by the system.

Phase I Authentication

Phase II Configuration

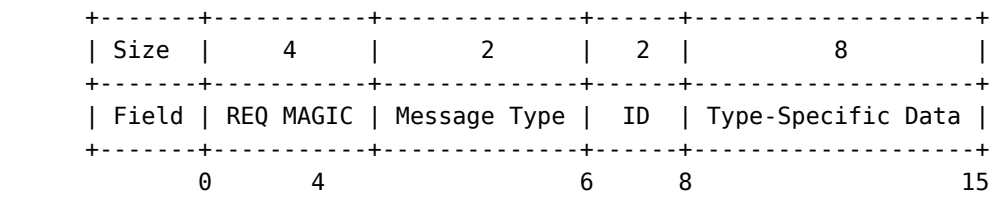
Phase III Steady-State Operation

Once Authentication and configuration is finished, the devices communicate using packets of the following two forms:

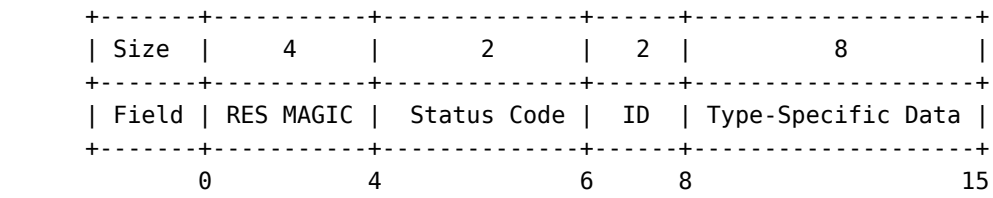
- Requests: sent by devices to initiate a stream of communication
- Responses: sent in responses to requests

Once this exchange is made, a more domain-specific protocol could be designed on top of this, using the basic authentication and configuration operations to probe support for different communication models.

The Request has a 16-byte header:



and the Response:



The fields are as follows:

REQ/S MAGIC: A 4-byte recognizable signature to identify proper starting packets, along with their type (request/response).

Message Type: One of the following:

- Reconfiguration Request - asks the device to reenter the configuration stage, either to update new hardware capabilities, or as part of an error recovery process.

- Set Parameter Value - requests a parameter change on a device implementing the delivery behaviors.
- Read Parameter Value - requests to read back a parameter on a device implementing the delivery behaviors.
- Get Device Status - requests a small informational block describing the error state of the device, for debugging.
- Notify Device - sends a regular generic data packet.
- Request Sensor Hook - requests a device implementing the Sensor to implement simple logic-based asynchronous I/O

Example implementation in a C-like syntax

```
struct Request {
  u8   magic[4];

  enum MessageType {
    RECONFIGURE,
    SET_PARAM,
    READ_PARAM,
    GET_STATUS,
    NOTIFY,
    RequestSensorHook,
  } message_type :32;

  union {
    struct Reconfigure {
      u8 must_reauthenticate;
      u56 resv;
    } reconfigure;
    struct SetParam {
      u32 param_selector;
      u32 param_value;
    };
    struct ReadParam {
      u32 param_selector;
      u32 resv;
    };
    struct GetStatus, Notify {
      u64 resv;
    };
    struct RequestSensorHook {
      enum HookType {
        SENSOR_UPDATE,
        SENSOR_LEVEL_UPPER,
        SENSOR_LEVEL_LOWER,
      } event_type :u32;
      u32 sensor_id;
    };
    struct Notify {
      u32 raw_data;
    }
  } content;
};
```

medical devices: sensors, delivery devices, and control devices

the network: wearable medical devices connected to one or more other devices

Devices can choose to mark data as protected (default) or public. In the case of public data, other devices will be able to query values reported by sesnors. This could be useful in the case of devices that want to have a histograph display.