IOT Server/Client communication

The IOT server running on the central node (IOT hub) has a TCP server running on port (5070), additionally, there is a TCP server on each node, running on port (1234), so that the server can send requests to the nodes at any time.

Server setup:

- IOT Server:
  Initiated with address 192.168.137.1, port (5070).
- Node server:
  Initiated on port IP, port (1234).

Message format:

The message is sent as a string, ending with a newline character `\r\n`.

- The 1st character determines the message type, if it's a request message or a response message.
- The 2nd and 3rd characters determine the message operation.
- The rest is the message data. The message data is key, value pairs formatted as the following:
  Key_1=value_1&key_2=value_2&key_3=value_3……&key_n=value_n
  If the message is a request message, it must contain the node ID parameter. If it's a response message, it will contain the response reason as a string (e.g. 210reason=Success) for clarity and ease of debugging.

Message types:

1. Request:
   - Initiated by a client to request a service from the server. Like introducing a new node into the server.
   - Code: 1
2. Response:
   - Reply from the server to the client, usually returns a status whether the request was valid or not. If the request is valid, it will be processed, the reply will be a success response. If the request was invalid, the reply will be a response message with the reason why the request failed.
   - Code: 2

Request messages:

1. Registration request:
   - Initiated by IOT nodes to add new node to the server. Requires node ID, the node ID is unique per node. A node should only add itself. If a node is already registered, this request has no effect, but the server will reply with a success response. Must be followed by a data update request.
   - Only sent by nodes to the hub.
   - Code: 10
2. Unregister request:
   - Removes a node from the server. Requires node ID. When successful, the node associated data are removed from the server permanently. A node can't be removed before registration first.
   - Only sent by nodes to the hub.
   - Code: 20
3. Data update request:
   - Update node data on the IOT server, or on the IOT node. Requires node ID and the IOT data.
   - Can be sent by the nodes to the hub (when the device's state change), or from the hub (server) to the node (when the user changes node state through the web interface).
   - Code: 30

Response messages:

1- Success:
   - Indicates that the request is valid, and successfully processed.
   - Code: 10

2- Failed:
   - Indicates that the request is invalid, and was not processed. The reason is not clear.
   - Code: 20

3- Invalid Message:
   - Indicates that the message format is wrong.
   - Code: 30

4- Invalid Message Format:
   - Indicates that the received message format is invalid (e.g. message length is insufficient to hold required information like message type and operation).
   - Code: 31

5- Invalid Message Object:
   - Indicates an error while forming message object in a server side script (internal error).
   - Code: 40

6- Unknown Message Type:
   - The received message type isn't either a request nor a response.
   - Code: 41

7- Unknown Request:
   - Request message operation is not known by the server.
   - Code: 42

8- Unknown Response:
   - Response message is not known by the server (internal error, as responses don't call for a reply message, when this response occur, it means something is misbehaving on the other side).
   - Code: 43

9- Unsupported Module:
   - Indicates that the module (node) trying to register is not supported by the server.
   - Code: 44

10- Missing Module HTML Template:
   - Indicates that the server can't find the modules HTML template file, even though the module is supported by the server (internal error). Make sure the templates files exist.
   - Code: 45

11- Missing Module Data template:
    - Indicates that the server can't locate module's data template file (internal error).
    - Cade: 46

12- Missing Module HTML File:
    - Indicates that the server can't locate module's HTML file, even though the module is supported and registered. This may occur due to an error during module registration process, that led to terminating the process before creating the module's files.
    - Code: 47

13- Missing Module Data File:
    - Same as the previous response, except this one is for the data file. If this error occurs, the problem might be related to template data file parsing as a JSON onject.
    - Code: 48

14- Unregistered module:
    - Indicates that the request can't be processed because the module (node) is not registered.
    - Code: 49

15- Missing Module ID:
    - Indicates that the request can't be processed as the message is missing the module ID parameter.
    - Code: 50

16- Missing Module IP:
    - Indicates that the module data is missing IP parameter. If this response occurs, the problem may be with the server request handler, as it's responsible for getting client's address from the sent request.
    - Code: 51

17- Invalid Data Filed Value:
    - Indicates that the received message's data had an invalid value for a data parameter.
    - Code: 52

18- Missing Data Field Value:
    - Indicates that the received message's data is missing a key-value pair.
    - Code: 53

Example:

1- Registration sequence:
   - Node: send a registration request: [110id=ilm_12345]
   - Server: process the request and send a reply [210reson=Success]

   Steps:

   - Start servers (web server, and IOT server).
   - Start IOT node, connect to the same network as the server.
   - Connect to the server's TCP server on address: 192.168.137.1:5070
   - Send request: 110id= ilm_12345
   - The server will reply with: 210reason=Success

2- Data update request:
   - Node: send data update request to the server: [130id= ilm_12345&state=on]
   - Server: process the request and reply [210reason=Success]
   - Module state is updated on the web interface.

3- Data update through web interface:
   - Open the web interface through the browser.
   - Change the status of the module, the click update.
   - The server will send a request to the node [130id=ilm_12345&state=off]

4- Unregister module:
   - Node: send unregister request to the server: [120id=ilm_12345]
   - Server: process the request, removes the module data from the server, and send a reply [210reason=Success]