Alarm Protocol Specification

Patrik Guthenberg, Alexander Eklund, Johan Mölder, Astbrq Jamil

February 2022

1 Introduction

The alarm protocol is a simple JSON and TCP based protocol designed to be used for an alarm system being developed as a project for the D0020E class at LTU. See `https://github.com/PatrikG-96/D0020E_Grupp11/tree/main`. The purpose of the protocol is to facilitate communication between 4 different entities, the monitor, the alarm server, the alarm API and the clients, all described in the Github.

2 Types

2.1 Alarm types

- fall_detected
- fall_confirmation

3 Error codes

Current error codes listen in a JSON format.

```
{
"InvalidJson" : 101,
"InvalidEncoding" : 102,
"NoSuchAlarm" : 201,
"InvalidToken" : 301,
"InvalidClient" : 302,
"InvalidResponse" : 303,
"InvalidType" : 304,
"InvalidMessageFormat" : 305
}
```

4 Messages

4.1 Generic message

The generic message format is extremely simple. All messages extend from this Message type. As of now, it only consists of a single "type" tag. This tag is exactly what it sounds like, the type of the message. More fields are likely to be added here given that there is time to do so. For example, protocol version, encoding etc.

```
{
"type" : "Message"
}
```

4.2 RequireToken

This message is also extremely simple. It has one single purpose, to indicate to the connected client that a secret token is required to prove that the client is authorized to connect to the server. This is done by simply changing the "type" tag.

```
{
"type" : "RequireToken"
}
```

Receiving this message indicates to the client that the server expects a response, which is described below.

4.3 TokenResponse

This is the client response to receiving a "RequireToken" message. It adds 2 new fields, "client_id" and "token". The "client_id" field is a unique identifier of the client. This should be stored in the client after adding it as an authorized client, and will be verified by the server. The second fields is "token". This is the secret token that the client must get from the REST API before attempting to connected to the server. More type constraints may be added later.

```
{
"type" : "TokenResponse",
"client_id" : Integer,
"token" : String
}
```

4.4 TokenAuthResult

A simple response to the TokenReponse. It adds a single tag to the generic message, "success". If verifying the token was successful, "success" is True, otherwise False. If success is False, the server should immediately terminate the connection after sending the message.

```
{
"type" : "TokenAuthResult",
"success" : Bool
}
```

4.5 AlarmNotification

This message is used to notify connected clients that a new alarm has been generated in real time. It adds a number of new fields, as seen in the image below.

```
{
"type" : "AlarmNotification"
"timestamp" : String,
"alarm_id: Int,
"monitor_id" : Int,
"alarm_type" : String,
"sensor_id" : String,
"sensor_info" : JSON,
"info" : JSON
}
```

Timestamp is a string that follows the format "YEAR-MONTH-DAY HOUR:MINUTE:SECONDS", for example "2022-02-23 17:05:43". Alarm id is a unique identifier for this specific alarm. Monitor id is a unique identifier for the Monitor. Alarm type is a string from the alarm types described above. Sensor id is a unique identifier for the sensor that generated the alarm. Sensor info is information about the sensor in JSON format. This can include fields like the name of the sensor and so on. Info is additional information about the alarm, currently not really used.

4.6 AlarmResponse

This message is a client initiated response to a given alarm.

```
{
"type" : "AlarmReponse"
"timestamp" : String,
```

```
"alarm_id": Int,
"response_type" : String,
"client_id" : Int,
"info" : JSON
}
```

The only new field here is "response_type". This field is either "READ" or "SOLVED". If response_type is "READ" it indicates that a client has received and noted the alarm. If it is "SOLVED" it indicates that a client has acted on the alarm and according to the clients judgements resolved the situation.

4.7 AlarmResponseConfirmation

A simple response to an Alarm reponse, indicating that the server has received the response. It works the same as the TokenAuthResult message, except that it obviously doesn't result in a termination of the connection if it fails for whatever reason.

```
{
"type" : "AlarmResponseConfirmation",
"success" : Bool
}
```

4.8 MonitorAlert

The MonitorAlert message is used when a monitor receives data from a sensor that indicates it should alert the system. This can result in one of two things. Either the server simply logs the event to the database, or if severe enough it also sends out an AlarmNotification message to all connected clients.

```
{
"type" : "MonitorAlert",
"sensor_id" : Int,
"sensor_name" : String,
"alarm_type" : String.
"timestamp" : String,
"params" : JSON
}
```

The only new field is params. It can contain information specific for the type of alarm that is generated. For example, if the alarm is for a fall event, it could contain the coordinates for this fall. If no params are needed, it can be an empty JSON, .

4.9 MonitorAlertResponse

A simple server response to a monitor alert, indicating whether or not it was successfully processed. Works like the other server responses.

```
{
"type" : "MonitorAlertResponse",
"success" : Bool
}
```

4.10 ErrorMessage

A simple message containing an error code that is described above.

```
{
"type" : "ErrorMessage",
"error_code" : Int
}
```