**Monitor and Control Garage Protocol (MCGP)**

Group 2

Kenneth Balogh

Arudra Venkat

Xiaofeng Zhou

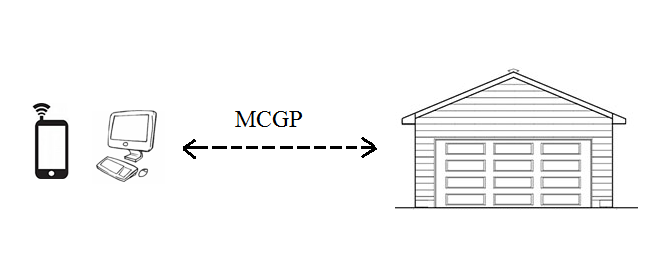
Daniel Speicher

1. Service Description

The Monitor and Control Garage Protocol (MCGP) provides a mechanism for users to manage, monitor and control a garage through the services defined by the protocol. MCGP provides a communication specification for applications, and is a protocol defined at the application layer utilizing TCP/IP. MCGP uses TCP as its transport layer protocol in order to have a reliable, secure and ordered connection. It adopts client-server mode and provide safety mechanism, the mechanism is based by user authentication which is realized by message signing via public-private key and embedding hash-signature in PDU.

After the client passes the server’s authentication, users can have a remote connection with their garage. Applications that follows MCGP will provide two kinds of services for users. First is the monitor service, server may connect to many digital devices in the garage such as thermometer, barometer and hygrometer. This allows a user to watch various environment parameters through the client. Second is the control service, in which a user can remotely control the door of the garage or the light inside it through client. Figure 1 is the schematic diagram of the MCGP.

Figure 1.



2 Message Definition – PDU

2.1 Addressing

The protocol is designed to operate over any reliably ordered transport layers. It is recommended that TCP/IP be used for the implementation. The client will establish a connection with a server using its IP address and a designated port number. In order to allow multiple connections to the server port numbers will used to allow different devices to be connected at any given time. An unused port of 9003 was selected because it does not require registration as is currently not being used by any other services at this time.

2.2 Flow Control

By utilizing TCP/IP flow control is handled at the TCP/IP Layer. This includes features such as QOS and Flow control.

2.3 PDU Definition

Communication is divided as follows:

1. Handshake – Sends the protocol version and authentication
2. Initialization – Server Sends a Check Garage Status Message to the client
3. Client/Server Communication –
   1. Client sends messages at any time and the server responds based on the message type. Table 1 below shows the initial message types.
   2. Server Responds according to the message type and action.

All messages are an ordered byte stream with a fixed size. The following convention will be used for this document C designates communication from the client and S is communication from the Server. [MessageByte] will be used to designate the message/ An example is C:[0x00] which is the client sending the request connection.

Message Byte Code Message Type Sent By

0x00 Request Connection Client

0x01 Error Client/Server

0x02 Connection Ready Server

0x03 Read Device List/Status Server

0x04 Control Device Client

0x05 Done Client

0x06 Confirm Server

0x07 Version Server/Client

0x08 Authenticate Server

Table 1 Message Codes

Error Byte Error Reason

0x00 Connection Refused

0x01 Version Error

0x02 Authentication Error

Table 2: Error Bytes

2.3.1 Handshake

The handshake phase is used to accomplish two tasks:

1. Version
2. Authentication

A client initiates this by sending a request connection command.

C:[0x00]

The server will then respond with the version byte code followed by the version it supports. This implementation will version 1 so 0x01 will be sent.

S:[0x07][0x01]

The client will respond with the version it supports. A server can except the client by initiating authentication.

S:[0x08][authtication challenge]

PGP will be used for authtication and decryption that will be discussed more in section 5.

If the authentication is successful, the server responds with a connection ready.

S:[0x02]

If there was an error in the client version or authentication, then the server responds with an error message and an error code.

S:[0x01][ErrorByteCode]

2.3.2 Discover Phase/Check Garage Status

Once the Handshake Phase has been completed the client will send a check garage status command. The server will respond with the list of devices and their current status.

All device actions are predefined in this version of the protocol.

S:[0x03][Number of Devices to Be Sent]

S:[D0][DeviceCode][CurrentState]

..

S:[DX][DeviceCode][CurrentState]

Until All Devices a Sent(DX represents last device example if 5 Device then D4 is last device sent]

2.3.3 Client to Server/ Server to Client Messages

The client can send messages to control the device or read a device. C:[0x04][DeviceID][Action]

An example of this if the client wants to turn a light D1 on. A client will send

C:[0x04][D1][Action]

Future versions could support additional parameters or custom controls.

Table 3 below shows the initial list of devices that are supported.

Initial Device Types

Code Device Type States Action

0x00 Garage Door open(0x00)/closed(0x01) turn on(0x01) turn off(0x00)

0x01 Light on(0x01)/off(0x00) open(1) close(0x00)

0x02 Temp Sensor on(0x01)/off(0x00) read temp(0x02)

0x03 BaroPres Sensor on(0x01)/off (0x00) read pressure(0x03)

At any time if an error is received the error message and error byte will be sent by either the client or server.

2.4 Quality of Service/ Error Control

The are several different types of error control as mentioned above. When an error occurs the error message and error code are sent. The connection is then terminated and everyone goes back to the IDLE state. The client must then reinitiate the connection with a Request Connection as described above.

The protocol is simplistic to use but still allows for authentication with the server. It provides a simple method to control and read the status of various devices. This is an advantage when most devices will be small resource constrained devices. Having a protocol that is simple to implement but still provides control and security is important in an ever connected world.

1. DFA
   1. Will Add on 5/6 after final updates
2. Extensibility
3. Security