Busca Preço G2 S

Manual do Desenvolvedor

Versão 1.3.0

1) Apresentação

O Terminal de Consulta Busca Preço G2 S é um verificador de preços que foi desenvolvido com o objetivo de oferecer um excelente desempenho e facilidade de instalação.

O equipamento utiliza protocolo de comunicação aberto, baseado no protocolo TCP/IP, que permite fácil adaptação em qualquer sistema operacional que dê suporte à rede Ethernet 10BaseT e/ou 100BaseTx usada pelos servidores.

Possui uma chave de liga e desliga e uma entrada USB para gravar ou ler as configurações do terminal via pendrive (arquivo de configuração), configurar o terminal via teclado, bem como atualização do firmware via pendrive. Possui também um botão de reset para restaurar as configurações do equipamento.

Suporta propagandas e consultas com imagens (Formato GIF, ou seja, para mandar uma imagem é necessário mandar um GIF estático) e GIF animados. Para modelos com áudio é possível tocar áudio no formato WAV - 8Khz - Mono (1 canal) - 8 bits - PCM_U8, de tempo mínimo de 2 segundos e no máximo 7 segundos, com a capacidade mínima do arquivo de áudio de 16KB e máxima 68KB. Para manter compatibilidade com sistemas legados o terminal tem a capacidade de funcionar como Busca Preço G1. Possui um Web Server integrado para configuração remota.

O Busca Preço G2 S lê o código de barras do produto através de um scanner CCD unidirecional de alta performance de 300 varreduras por segundo, e envia essa informação através de uma rede local Ethernet usando o protocolo TCP/IP a um programa servidor. O servidor consulta o banco de dados e retorna ao cliente, pelo terminal de consulta, o nome e o preço do produto consultado ou a informação de produto não encontrado. O Busca Preço G2 S apresenta essas informações num display TFT colorido de 320 x 240 pontos (256K cores) de fácil leitura. O Protocolo do Terminal de Consultas Gertec da camada aplicação é aberto e é fácil de ser usado por desenvolvedores de programas. Os desenvolvedores contam também com kit de desenvolvimento de software (SDK) para aplicações personalizadas.

O terminal, quando não está sendo usado para consulta de preços, exibe quatro mensagens pré-configuradas em seu display quando conectado a um servidor. Essas frases podem ser configuradas pela rede.

O terminal possui interface de comunicação cabeada via Ethernet e WIFI seguindo os padrões 802.11b e 802.11g. A interface a ser utilizada deve ser escolhida pelo menu de configurações. Os terminais WIFI são compatíveis com os servidores utilizados com as versões anteriores utilizando um protocolo de comunicação binário proprietário da Gertec. Esta aplicação servidora (Ex: TC Server) pode fornecer resultados de consultas de preço, realizar configurações nos terminais e transferir arquivos.

O aplicativo principal do Busca Preço G2 S é o verificador de preços. Ele é executado automaticamente sem a necessidade de pressionar tecla. Não é necessário conectar teclado no conector USB para que esse aplicativo seja executado. Ao iniciar o programa verificador de preços do terminal, o aplicativo configura a porta serial do scanner, inicializa o controlador de rede e o display. Durante esse processo, o terminal mostra no display informações sobre a configuração de rede (seu endereço IP, endereço MAC, máscara de rede etc.).

Após esse processo, o terminal tenta se conectar ao servidor no endereço IP configurado. Se não for possível conectar-se ao servidor (IP do servidor errado, cabo desconectado, aplicativo servidor desativado, conflito de IP, servidor DHCP fora do ar etc.), o terminal fica indefinidamente tentado conectar-se até que a conexão acabe se estabelecendo ou o terminal seja desligado. Após a conexão com servidor, o terminal já está pronto para realizar uma consulta de preços, ou seja, quando algum código de barras for passado no scanner, este é enviado para o servidor que deve retornar o nome e preço do produto, ou uma mensagem de produto não cadastrado. Além disso, o servidor pode alterar os diversos parâmetros de configuração do terminal, reiniciá-lo e/ou enviar uma mensagem para seu display.

2) Utilização

2.1) Protocolo de Comunicação (TCP-IP)

O protocolo de comunicação do Busca Preço G2 S é executado na camada de Aplicação do TCP/IP, utilizando um socket binário na porta 6500. Quando o terminal se conecta ao servidor, cria-se um link para trocar mensagens (do servidor com o terminal). Essas mensagens obedecem a uma regra chamada de Protocolo do TC406 Gertec. A Gertec disponibiliza programas servidores para exemplificar o uso deste protocolo, com código fonte completamente aberto e DLL para auxiliar a criação de novos servidores.
No processo de resposta a mensagem de leitura do código de barras do terminal, o servidor necessita enviar um gif, uma mensagem ou um áudio. Caso não haja resposta dos comandos citados, o terminal responderá o servidor com a seguinte mensagem: **#queryprocessfailure**.

2.1.1) Comandos

2.1.1.1) #ok

Comando	Resposta	Origem	Ação realizada
#ok	#protocolo|versãofw	Servidor	Estabelece comunicação com o terminal

#ok: Este é o primeiro que deve ser enviado para o terminal, a fim de estabelecer comunicação com o terminal, e poder enviar e receber comandos.
Exemplo de resposta do comando para modelo Busca Preço G1: #tc502|4,0
Exemplo de resposta do comando para modelo Busca Preço G2: #tc406|1.1.0
Exemplo de resposta do comando para modelo Busca Preço G2 S: #tc406|3.3.1 S

Exemplo em Java:

Envio de comando:

```
private BufferedInputStream in;
private BufferedOutputStream out;

ServerSocket server = new ServerSocket(6500); //Abre um socket na porta 6500
Socket sock = server.accept(); //Aguarda estabelecer conexão


out = new BufferedOutputStream(sock.getOutputStream());
in = new BufferedInputStream(sock.getInputStream());


out.write("#ok".getBytes()); //Envia o comando #ok para o equipamento
out.flush();
```

Resposta do equipamento:

```
serverStatus = sock.isConnected(); //Verifica se há conexão no socket
        if (serverStatus == true) {

           Thread.sleep(500); //Tempo para garantir a resposta do equipamento

            String comando;
            byte vetor[] = new byte[255];
```

```
int qtdLida;
if (in.available() > 0) {
   qtdLida = in.read(vetor);
   comando = new String(vetor); //Resposta do BP
   String reciveCommand = comando.substring(0, qtdLida); //Resposta tratada
}
```

reciveCommand = #tc406|4.0

Exemplo em C#:

Envio de comando:

```
private Socket server;
private IPEndPoint IPServer;
private Socket cliente;

server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, Proto-
colType.Tcp);//Cria e configura o socket servidor
IPServer = new IPEndPoint(IPAddress.Any, 6500); //Configura a porta e o IP do servi-
dor

server.Bind(IPServer); //Configura a porta do servidor
server.Listen(5); //Abre a porta para conexões

cliente = server.Accept(); //Aceita a conexão do terminal e retorna o socket para
comunicação

byte[] comando = Encoding.ASCII.GetBytes("#ok"); //Transforma a string em bytes

cliente.Send(comando); //Envia o comando para o equipamento
```

Resposta do equipamento:

```
System.Threading.Thread.Sleep(500); //Tempo para garantir resposta do equipamento

String resposta = null; //Variável para guardar a resposta
byte[] dados = new byte[255]; //Cria um vetor de 255 bytes

cliente.Receive(dados);//Faz a leitura da resposta do Busca Preço

resposta = new System.Text.ASCIIEncoding().GetString(dados); //Converte os bytes para
texto
```

resposta = #tc406|4.0

2.1.1.2) #live?

Comando	Resposta	Origem	Ação realizada
#live?	#live	Servidor	Verifica se o terminal está vivo

#live?: Este comando serve apenas para verificar se o terminal está vivo. Esse comando precisa ser mandado esporadicamente (a cada 15 segundos, por exemplo) para o equipamento não perder a conexão com o servidor.

Exemplo em Java:

Envio do comando:

```
byte[] command = "#live?".getBytes(); //Transforma a string em bytes
out.write(command); //Envia o comando para o equipamento
out.flush();
```

Resposta do equipamento:

```
String comando;
 byte vetor[] = new byte[255];
 int qtdLida;
 if (in.available() > 0) {
    qtdLida = in.read(vetor);
    comando = new String(vetor); //Resposta do BP
    String reciveCommand = comando.substring(0, qtdLida); //Resposta tratada
 }
```

recieveComand = #live

Exemplo em C#:

Envio de comando:

```
byte[] comando = Encoding.ASCII.GetBytes("#live?"); //Transforma a string em bytes

cliente.Send(comando); //Envia o comando para o equipamento
```

Resposta do equipamento:

```
System.Threading.Thread.Sleep(500); //Tempo para garantir resposta do equipamento

String resposta = null; //Variável para guardar a resposta
byte[] dados = new byte[255]; //Cria um vetor de 255 bytes

cliente.Receive(dados);//Faz a leitura da resposta do Busca Preço

resposta = new System.Text.ASCIIEncoding().GetString(dados); //Converte os bytes para
texto
```

resposta = #live

2.1.1.3) #alwayslive

Comando	Resposta	Origem	Ação realizada
#alwayslive	#alwayslive_ok	Servidor	Mantém o terminal sempre conectado

#alwayslive: Mantém o terminal conectado na rede, não necessitando enviar de tempos em tempos o comando **#live?** (compatibilidade).

Exemplo em Java:

Envio do comando:

```
byte[] command = "#alwayslive".getBytes(); //Transforma a string em bytes
out.write(command); //Envia o comando para o equipamento
out.flush();
```

Resposta do equipamento:

```
String comando;
 byte vetor[] = new byte[255];
 int qtdLida;
 if (in.available() > 0) {
    qtdLida = in.read(vetor);
    comando = new String(vetor); //Resposta do BP
    String reciveCommand = comando.substring(0, qtdLida); //Resposta tratada
 }
```

recieveComand = #alwayslive_ok

Exemplo em C#:

Envio de comando:

```
byte[] comando = Encoding.ASCII.GetBytes("#alwayslive"); //Transforma a string em bytes

cliente.Send(comando); //Envia o comando para o equipamento
```

Resposta do equipamento:

```
System.Threading.Thread.Sleep(500); //Tempo para garantir resposta do equipamento

String resposta = null; //Variável para guardar a resposta
byte[] dados = new byte[255]; //Cria um vetor de 255 bytes

cliente.Receive(dados);//Faz a leitura da resposta do Busca Preço

resposta = new System.Text.ASCIIEncoding().GetString(dados); //Converte os bytes para texto
```

resposta = #alwayslive_ok

2.1.1.4) #checklive

Comando	Resposta	Origem	Ação realizada
#checklive	#checklive_ok	Servidor	Não mantém o terminal conectado

#checklive: Não mantém o terminal conectado na rede, necessitando enviar de tempos em tempos o comando **#live?** (compatibilidade).

Exemplo em Java:

Envio do comando:

```
byte[] command = "#checklive".getBytes(); //Transforma a string em bytes
out.write(command); //Envia o comando para o equipamento
out.flush();
```

Resposta do equipamento:

```
String comando;
 byte vetor[] = new byte[255];
 int qtdLida;
 if (in.available() > 0) {
```

```
            qtdLida = in.read(vetor);
            comando = new String(vetor); //Resposta do BP
            String reciveCommand = comando.substring(0, qtdLida); //Resposta tratada
        }
```

recieveComand = #checklive_ok

Exemplo em C#:

Envio de comando:

```
        byte[] comando = Encoding.ASCII.GetBytes("#checklive"); //Transforma a string em
bytes

        cliente.Send(comando); //Envia o comando para o equipamento
```

Resposta do equipamento:

```
        System.Threading.Thread.Sleep(500); //Tempo para garantir resposta do equipamento

        String resposta = null; //Variável para guardar a resposta
        byte[] dados = new byte[255]; //Cria um vetor de 255 bytes

        cliente.Receive(dados);//Faz a leitura da resposta do Busca Preço

        resposta = new System.Text.ASCIIEncoding().GetString(dados); //Converte os bytes para
texto
```

resposta = #checklive_ok

2.1.1.5) #restartsoft

Comando	Resposta	Origem	Ação realizada
#restartsoft + senha	#restartsoft_ok	Servidor	Reinicializa Terminal

#restartsoft + senha: Enviando este comando, o terminal é reiniciado.

A senha é um longword (4 bytes) que devem ser enviados para que o terminal realize este comando (0xA5CC5A33).

Exemplo em Java:

Envio do comando:

```
        //Monta o comando para ser enviado ao equipamento
        byte command[] = new byte[16];
        command[0]  = '#';
        command[1]  = 'r';
        command[2]  = 'e';
        command[3]  = 's';
        command[4]  = 't';
        command[5]  = 'a';
        command[6]  = 'r';
        command[7]  = 't';
        command[8]  = 's';
        command[9]  = 'o';
        command[10] = 'f';
```

```
command[11] = 't';
command[12] = (byte)0xA5;
command[13] = (byte)0xCC;
command[14] = (byte)0x5A;
command[15] = (byte)0x33;

out.write(command); //Envia o comando para o equipamento
out.flush();
```

Resposta do equipamento:

```
String comando;
 byte vetor[] = new byte[255];
 int qtdLida;
 if (in.available() > 0) {
    qtdLida = in.read(vetor);
    comando = new String(vetor); //Resposta do BP
    String reciveCommand = comando.substring(0, qtdLida); //Resposta tratada
 }
```

recieveComand = #restartsoft_ok

Exemplo em C#:

Envio de comando:

```
byte[] senhaBytes = {0xa5, 0xcc, 0x5a, 0x33}; //Bytes da senha
string senha =   new ASCIIEncoding().GetString(senhaBytes); //Transforma a senha em
string
string restart = "#restartsoft" + senha; //Monta comando

byte[] comando = Encoding.ASCII.GetBytes(restart); //Transforma a string em bytes

cliente.Send(comando); //Envia comando para o equipamento
```

Resposta do equipamento:

```
System.Threading.Thread.Sleep(500); //Tempo para garantir resposta do equipamento

String resposta = null; //Variável para guardar a resposta
byte[] dados = new byte[255]; //Cria um vetor de 255 bytes

cliente.Receive(dados);//Faz a leitura da resposta do Busca Preço

resposta = new System.Text.ASCIIEncoding().GetString(dados); //Converte os bytes para
texto
```

resposta = #restartsoft_ok

2.1.1.6) #config?

Comando	Resposta	Origem	Ação realizada
#config?	#config + dados	Servidor	Responde com a Configuração

#config?: Este comando solicita a configuração atual do terminal. O terminal responde com os seguintes dados:

1 byte: tamanho da string do IP do servidor.

1 string: IP do servidor.
1 byte: tamanho da string do IP do terminal.
1 string: IP do terminal.
1 byte: tamanho da string da máscara de rede.
1 string: máscara de rede.
1 byte: tamanho da string do texto a linha 1.
1 string: texto da linha 1.
1 byte: tamanho da string do texto da linha 2.
1 string: texto da linha 2.
1 byte: tempo de exibição.

OBS: Para saber o tamanho real de cada string ou do tempo de exibição, devemos subtrair 48 (decimal) do valor de cada byte.

Exemplo de configurações de um terminal:

IP do servidor: 172.18.4.81
Tamanho do IP do servidor: 11 caracteres
IP do terminal: 172.18.4.38
Tamanho do IP do terminal: 11 caracteres
Máscara: 255.255.255.0
Tamanho da Máscara: 13 caracteres
Texto da linha 1: PASSE DUAS
Tamanho da linha 1: 10 caracteres
Texto da linha 2: COISAS
Tamanho da linha 2: 6 caracteres
Tempo de exibição: 5 segundos

Para se calcular o resultado dos bytes referente ao tamanho dos dados, deve-se adicionar 48 no tamanho e transformar esse valor em ASCII. Ou seja, com um IP de 172.18.4.81, o tamanho da string é de 11 caracteres. Ao adicionar 48, resulta em 59, ou seja, o caractere especial “;” na tabela ASCII.
O mesmo deve ser feito com o tempo de exibição: somar 48 no valor do tempo, e transformar esse número em um caractere ASCII.

Tamanho do IP do servidor = 11 + 48 = 59 (; em ASCII)
Tamanho do IP do terminal = 11 + 48 = 59 (; em ASCII)
Tamanho da Máscara = 13 + 48 = 61 (= em ASCII)
Tamanho da linha 1: 10 + 48 = 58 (: em ASCII)
Tamanho da linha 2: 6 + 48 = 54 (6 em ASCII)
Tempo de exibição: 5 + 48 = 53 (5 em ASCII)

São esses valores convertidos que serão respondidos pelo Busca Preço G2 S nos parâmetros de byte. Caso só se tenha a resposta do equipamento, e se deseje obter o tamanho original do texto, deve-se converter o caractere de resposta do equipamento no seu valor decimal correspondente e subtrair 48.

Exemplo em Java:

Envio do comando:

```
byte[] command = "#config?".getBytes(); //Transforma a string em bytes
out.write(command); //Envia o comando para o equipamento
out.flush();
```

Resposta do equipamento:

```
String comando;
 byte vetor[] = new byte[255];
 int qtdLida;
 if (in.available() > 0) {
    qtdLida = in.read(vetor);
    comando = new String(vetor); //Resposta do BP
    String reciveCommand = comando.substring(0, qtdLida); //Resposta tratada
 }
```

No exemplo, o valor de recieveComand é #config;172.18.4.81;172.18.4.38=255.255.255.0:PASSE DUAS6COISAS5

Exemplo em C#:

Envio de comando:

```
byte[] comando = Encoding.ASCII.GetBytes("#config?"); //Transforma a string em bytes

cliente.Send(comando); //Envia o comando para o equipamento
```

Resposta do equipamento:

```
System.Threading.Thread.Sleep(500); //Tempo para garantir resposta do equipamento

String resposta = null; //Variável para guardar a resposta
byte[] dados = new byte[255]; //Cria um vetor de 255 bytes

cliente.Receive(dados);//Faz a leitura da resposta do Busca Preço

resposta = new System.Text.ASCIIEncoding().GetString(dados); //Converte os bytes para texto
```

No exemplo, o valor de resposta é #config;172.18.4.81;172.18.4.38=255.255.255.0:PASSE DUAS6COISAS5

2.1.1.7) #config02?

Comando	Resposta	Origem	Ação realizada
#config02?	#config02 + dados	Servidor	Responde com a Configuração

#config02?: Este comando solicita a configuração atual do terminal. O terminal responde com os seguintes dados:

1 byte: tamanho da string do IP do servidor.
1 string: IP do servidor.
1 byte: tamanho da string do IP do terminal.
1 string: IP do terminal.
1 byte: tamanho da string da máscara de rede.
1 string: máscara de rede.
1 byte: tamanho da string do texto a linha 1.
1 string: texto da linha 1.
1 byte: tamanho da string do texto da linha 2.
1 string: texto da linha 2.
1 byte: tamanho da string do texto a linha 3.
1 string: texto da linha 3.
1 byte: tamanho da string do texto da linha 4.
1 string: texto da linha 4.
1 byte: tempo de exibição.

OBS: Para saber o tamanho real de cada string ou do tempo de exibição, devemos subtrair 48 (decimal) do valor de cada byte.

Exemplo de configurações de um terminal:

IP do servidor: 172.18.4.81
Tamanho do IP do servidor: 11 caracteres
IP do terminal: 172.18.4.38
Tamanho do IP do terminal: 11 caracteres
Máscara: 255.255.255.0
Tamanho da Máscara: 13 caracteres
Texto da linha 1: PASSE DUAS
Tamanho da linha 1: 10 caracteres
Texto da linha 2: COISAS
Tamanho da linha 2: 6 caracteres
Texto da linha 3: PASSE UM
Tamanho da linha 3: 8 caracteres
Texto da linha 4: PRODUTO
Tamanho da linha 4: 7 caracteres
Tempo de exibição: 5 segundos

Para se calcular o resultado dos bytes referente ao tamanho dos dados, deve-se adicionar 48 no tamanho e transformar esse valor em ASCII. Ou seja, com um IP de 172.18.4.81, o tamanho da string é de 11 caracteres. Ao adicionar 48, resulta em 59, ou seja, o caractere especial “;” na tabela ASCII.
O mesmo deve ser feito com o tempo de exibição: somar 48 no valor do tempo, e transformar esse número em um caractere ASCII.

Tamanho do IP do servidor = 11 + 48 = 59 (; em ASCII)
Tamanho do IP do terminal = 11 + 48 = 59 (; em ASCII)
Tamanho da Máscara = 13 + 48 = 61 (= em ASCII)
Tamanho da linha 1: 10 + 48 = 58 (: em ASCII)
Tamanho da linha 2: 6 + 48 = 54 (6 em ASCII)
Tamanho da linha 3: 8 + 48 = 56 (8 em ASCII)
Tamanho da linha 4: 7 + 48 = 55 (7 em ASCII)
Tempo de exibição: 5 + 48 = 53 (5 em ASCII)

São esses valores convertidos que serão respondidos pelo Busca Preço G2 S nos parâmetros de byte. Caso só se tenha a resposta do equipamento, e se deseje obter o tamanho original do texto, deve-se converter o caractere de resposta do equipamento no seu valor decimal correspondente e subtrair 48.

Exemplo em Java:

Envio do comando:

```
byte[] command = "#config02?".getBytes(); //Transforma a string em bytes
out.write(command); //Envia o comando para o equipamento
out.flush();
```

Resposta do equipamento:

```
String comando;
 byte vetor[] = new byte[255];
 int qtdLida;
 if (in.available() > 0) {
```

```
            qtdLida = in.read(vetor);
            comando = new String(vetor); //Resposta do BP
            String reciveCommand = comando.substring(0, qtdLida); //Resposta tratada
        }
```

No exemplo, o valor de recieveComand é #config02;172.18.4.81;172.18.4.38=255.255.255.0:PASSE DUAS6COISAS8PASSE UM7PRODUTO5

Exemplo em C#:

Envio de comando:

```
        byte[] comando = Encoding.ASCII.GetBytes("#config02?"); //Transforma a string em bytes

        cliente.Send(comando); //Envia o comando para o equipamento
```

Resposta do equipamento:

```
        System.Threading.Thread.Sleep(500); //Tempo para garantir resposta do equipamento

        String resposta = null; //Variável para guardar a resposta
        byte[] dados = new byte[255]; //Cria um vetor de 255 bytes

        cliente.Receive(dados);//Faz a leitura da resposta do Busca Preço

        resposta = new System.Text.ASCIIEncoding().GetString(dados); //Converte os bytes para texto
```

No exemplo, o valor de resposta é #config02;172.18.4.81;172.18.4.38=255.255.255.0:PASSE DUAS6COISAS8PASSE UM7PRODUTO5

2.1.1.8) #extconfig?

Comando	Resposta	Origem	Ação realizada
#extconfig?	#extconfig + dados	Servidor	Responde com a Configuração

#extconfig?: Comando semelhante aos anteriores, porém com mais dados de configuração, são eles:

1 byte: tamanho da string do IP do servidor.
1 string: IP do servidor.
1 byte: tamanho da string do IP do terminal.
1 string: IP do terminal.
1 byte: tamanho da string da máscara de rede.
1 string: máscara de rede.
1 byte: tamanho da string do Gateway.
1 string: Gateway.
1 byte: 59 (decimal).
1 string: “Sem Suporte” (sem as aspas).
1 byte: tamanho da string do nome do terminal.
1 string: Nome do terminal.
1 byte: tamanho da string do texto a linha 1.
1 string: texto da linha 1.
1 byte: tamanho da string do texto da linha 2.
1 string: texto da linha 2.
1 byte: 59 (decimal).
1 string: “Sem Suporte” (sem as aspas).
1 byte: 59 (decimal).

1 string: “Sem Suporte” (sem as aspas).
1 byte: 59 (decimal).
1 string: “Sem Suporte” (sem as aspas).
1 byte: Tempo de Exibição.
1 byte: IP dinâmico/fixo. (48 = fixo, 49 = dinâmico)
1 byte: 48 (decimal).

OBS: Para saber o tamanho real de cada string ou do tempo de exibição, devemos subtrair 48 (decimal) do valor de cada byte.

Exemplo de configurações de um terminal:

IP do servidor: 172.18.4.81
Tamanho do IP do servidor: 11 caracteres
IP do terminal: 172.18.4.38
Tamanho do IP do terminal: 11 caracteres
Máscara: 255.255.255.0
Tamanho da Máscara: 13 caracteres
Gateway: 192.168.0.1
Tamanho do Gateway: 11 caracteres
Nome do terminal: Mauricio
Tamanho do nome do terminal: 8
Texto da linha 1: PASSE DUAS
Tamanho da linha 1: 10 caracteres
Texto da linha 2: COISAS
Tamanho da linha 2: 6 caracteres
Tempo de exibição: 5 segundos
IP fixo

Para se calcular o resultado dos bytes referente ao tamanho dos dados, deve-se adicionar 48 no tamanho e transformar esse valor em ASCII. Ou seja, com um IP de 172.18.4.81, o tamanho da string é de 11 caracteres. Ao adicionar 48, resulta em 59, ou seja, o caractere especial “;” na tabela ASCII.
O mesmo deve ser feito com o tempo de exibição: somar 48 no valor do tempo, e transformar esse número em um caractere ASCII.

Tamanho do IP do servidor = 11 + 48 = 59 (; em ASCII)
Tamanho do IP do terminal = 11 + 48 = 59 (; em ASCII)
Tamanho da Máscara = 13 + 48 = 61 (= em ASCII)
Tamanho do Gateway = 11 + 48 = 59 (; em ASCII)
Tamanho do nome do terminal = 8 + 48 = 56 (8 em ASCII)
Tamanho da linha 1: 10 + 48 = 58 (: em ASCII)
Tamanho da linha 2: 6 + 48 = 54 (6 em ASCII)
Tempo de exibição: 5 + 48 = 53 (5 em ASCII)
IP fixo = 48 = 0 em ASCII

São esses valores convertidos que serão respondidos pelo Busca Preço G2 S nos parâmetros de byte. Caso só se tenha a resposta do equipamento, e se deseje obter o tamanho original do texto, deve-se converter o caractere de resposta do equipamento no seu valor decimal correspondente e subtrair 48.

Exemplo em Java:

Envio do comando:

```
byte[] command = "#config02?".getBytes(); //Transforma a string em bytes
out.write(command); //Envia o comando para o equipamento
```

```
out.flush();
```

Resposta do equipamento:

```
String comando;
 byte vetor[] = new byte[255];
 int qtdLida;
 if (in.available() > 0) {
    qtdLida = in.read(vetor);
    comando = new String(vetor); //Resposta do BP
    String reciveCommand = comando.substring(0, qtdLida); //Resposta tratada
 }
```

No exemplo, o valor de recieveComand é:
#extconfig;172.18.4.81;172.18.4.38=255.255.255.0;192.168.0.1;Sem Suporte8Mauricio:PASSE DUAS6COISAS;Sem Suporte;Sem Suporte;Sem Suporte500

Exemplo em C#:

Envio de comando:

```
byte[] comando = Encoding.ASCII.GetBytes("#extconfig?"); //Transforma a string em bytes

cliente.Send(comando); //Envia o comando para o equipamento
```

Resposta do equipamento:

```
System.Threading.Thread.Sleep(500); //Tempo para garantir resposta do equipamento

String resposta = null; //Variável para guardar a resposta
byte[] dados = new byte[255]; //Cria um vetor de 255 bytes

cliente.Receive(dados);//Faz a leitura da resposta do Busca Preço

resposta = new System.Text.ASCIIEncoding().GetString(dados); //Converte os bytes para texto
```

No exemplo, o valor de resposta é:
#extconfig;172.18.4.81;172.18.4.38=255.255.255.0;192.168.0.1;Sem Suporte8Mauricio:PASSE DUAS6COISAS;Sem Suporte;Sem Suporte;Sem Suporte500

2.1.1.9) #paramconfig?

Comando	Resposta	Origem	Ação realizada
#paramconfig?	#paramconfig + dados	Servidor	Responde com a Configuração

#paramconfig?: Este comando solicita os parâmetros extras da configuração da rede. O terminal responde com os seguintes dados:

1 byte: valor do IP dinâmico.
1 byte: valor da busca do servidor.

Se o equipamento estiver configurado como IP dinâmico, o primeiro campo recebe 1, caso contrário, recebe 0.

O valor da busca do servidor sempre será 0 neste equipamento, pois o Busca Preço G2 S não utiliza essa função.

Exemplo em Java:

Envio do comando:

```
byte[] command = "#paramconfig?".getBytes(); //Transforma a string em bytes
out.write(command); //Envia o comando para o equipamento
out.flush();
```

Resposta do equipamento:

```
String comando;
 byte vetor[] = new byte[255];
 int qtdLida;
 if (in.available() > 0) {
    qtdLida = in.read(vetor);
    comando = new String(vetor); //Resposta do BP
    String reciveCommand = comando.substring(0, qtdLida); //Resposta tratada
 }
```

Se o equipamento estiver com IP fixo, reciveCommand = #paramconfig00
Se o equipamento estiver com IP dinâmico, reciveCommand = #paramconfig10

Exemplo em C#:

Envio de comando:

```
byte[] comando = Encoding.ASCII.GetBytes("#paramconfig?"); //Transforma a string em bytes

cliente.Send(comando); //Envia o comando para o equipamento
```

Resposta do equipamento:

```
System.Threading.Thread.Sleep(500); //Tempo para garantir resposta do equipamento

String resposta = null; //Variável para guardar a resposta
byte[] dados = new byte[255]; //Cria um vetor de 255 bytes

cliente.Receive(dados);//Faz a leitura da resposta do Busca Preço

resposta = new System.Text.ASCIIEncoding().GetString(dados); //Converte os bytes para texto
```

Se o equipamento estiver com IP fixo, resposta = #paramconfig00
Se o equipamento estiver com IP dinâmico, resposta = #paramconfig10

2.1.1.10) #updconfig?

Comando	Resposta	Origem	Ação realizada
#updconfig?	#updconfig + dados	Servidor	Responde com a Configuração

#updconfig?: Comando que pede a configuração de atualização do terminal:

1 byte: tamanho da string do Gateway.
1 string: Gateway.

1 byte: 59 (decimal).
1 string: “Sem Suporte” (sem as aspas).
1 byte: tamanho da string do nome do terminal.
1 string: Nome do terminal.
1 byte: 59 (decimal).
1 string: “Sem Suporte” (sem as aspas).
1 byte: 59 (decimal).
1 string: “Sem Suporte” (sem as aspas).
1 byte: 59 (decimal).
1 string “Sem Suporte” (sem as aspas).

OBS: Para saber o tamanho real de cada string ou do tempo de exibição, devemos subtrair 48 (decimal) do valor de cada byte.

Exemplo de configurações de um terminal:

Gateway: 192.168.0.1
Tamanho do Gateway: 11 caracteres
Nome do terminal: Mauricio
Tamanho do nome do terminal: 8

Para se calcular o resultado dos bytes referente ao tamanho dos dados, deve-se adicionar 48 no tamanho e transformar esse valor em ASCII. Ou seja, com um Gateway de 192.168.0.1, o tamanho da string é de 11 caracteres. Ao adicionar 48, resulta em 59, ou seja, o caractere especial “;” na tabela ASCII.

Tamanho do Gateway = 11 + 48 = 59 (; em ASCII)
Tamanho do nome do terminal = 8 + 48 = 56 (8 em ASCII)

São esses valores convertidos que serão respondidos pelo Busca Preço G2 S nos parâmetros de byte. Caso só se tenha a resposta do equipamento, e se deseje obter o tamanho original do texto, deve-se converter o caractere de resposta do equipamento no seu valor decimal correspondente e subtrair 48.

Exemplo em Java:

Envio do comando:

```
byte[] command = " #updconfig?".getBytes(); //Transforma a string em bytes
out.write(command); //Envia o comando para o equipamento
out.flush();
```

Resposta do equipamento:

```
String comando;
 byte vetor[] = new byte[255];
 int qtdLida;
 if (in.available() > 0) {
    qtdLida = in.read(vetor);
    comando = new String(vetor); //Resposta do BP
    String reciveCommand = comando.substring(0, qtdLida); //Resposta tratada
 }
```

No exemplo, o valor de resposta é #updconfig;192.168.0.1;Sem Suporte8Mauricio;Sem Suporte;Sem Suporte;Sem Suporte

Exemplo em C#:

Envio de comando:

```
        byte[] comando = Encoding.ASCII.GetBytes("#updconfig?"); //Transforma a string em
bytes

        cliente.Send(comando); //Envia o comando para o equipamento
```

Resposta do equipamento:

```
        System.Threading.Thread.Sleep(500); //Tempo para garantir resposta do equipamento

        String resposta = null; //Variável para guardar a resposta
        byte[] dados = new byte[255]; //Cria um vetor de 255 bytes

        cliente.Receive(dados);//Faz a leitura da resposta do Busca Preço

        resposta = new System.Text.ASCIIEncoding().GetString(dados); //Converte os bytes para
texto
```

No exemplo, o valor de resposta é #updconfig;192.168.0.1;Sem Suporte8Mauricio;Sem Suporte;Sem Suporte;Sem Suporte

2.1.1.11) #rconf

Comando	Resposta	Origem	Ação realizada
#rconf + dados	Nenhuma	Servidor	Altera Configurações do Terminal

#rconf + dados: Configura o terminal. Os dados de configuração que precisam ser passados são:

1 byte: tamanho da string do IP do servidor.
1 string: IP do servidor.
1 byte: tamanho da string do IP do terminal.
1 string: IP do terminal.
1 byte: tamanho da string da máscara de rede.
1 string: máscara de rede.
1 byte: tamanho da string do texto a linha 1.
1 string: texto da linha 1.
1 byte: tamanho da string do texto da linha 2.
1 string: texto da linha 2.
1 byte: tempo de exibição.

OBS: O valor do byte com o tamanho de cada string ou do tempo de exibição deve ser somado com 48 (decimal).

Exemplo em Java:

Envio do comando:

```
        //Define os parâmetros que serão enviados na função
        String IpServidor     = "172.18.4.81";
        String IpTerminal   = "172.18.4.200";
        String Mascara     = "255.255.255.0";
        String Linha1     = "Teste do comando ";
        String Linha2     = "rconf";
        String tempo        = "4";
```

```
//Começo da string, e variável que guardará todas as informações que serão enviadas
String comandoCompleto = "#rconf";

//Cálculo dos tamanhos dos parâmetros
int Tamanho1, Tamanho2,TamanhoIPserv, TamanhoIPterm, TamanhoMascara, TempoDisplay;

//Para definir os tamanhos que precisam ser passados para o protocolo, deve-se adicionar 48
(ou 0x30 em hexa) nos tamanhos das strings
Tamanho1      = Linha1.length() + 0x30;
Tamanho2      = Linha2.length() + 0x30;
TamanhoIPserv  = IpServidor.length() + 0x30;
TamanhoIPterm  = IpTerminal.length() + 0x30;
TamanhoMascara    = Mascara.length() + 0x30;
TempoDisplay = Integer.valueOf(tempo) + 0x30;

comandoCompleto += (char)TamanhoIPserv;
comandoCompleto += IpServidor;
comandoCompleto += (char)TamanhoIPterm;
comandoCompleto += IpTerminal;
comandoCompleto += (char)TamanhoMascara;
comandoCompleto += Mascara;
comandoCompleto += (char)Tamanho1;
comandoCompleto += Linha1;
comandoCompleto += (char)Tamanho2;
comandoCompleto += Linha2;
comandoCompleto += (char)TempoDisplay;

byte[] command = comandoCompleto.getBytes();
out.write(command); //Envia o comando para o equipamento
out.flush();
```

Neste caso, não há valor de resposta do equipamento. Se o comando for bem executado, o Busca Preço G2 S irá reiniciar, e quando se conectar novamente ao servidor já estará com a nova configuração.

Exemplo em C#:

Envio de comando:

```
//Define os parâmetros que serão enviados na função
string IpServidor = "172.18.4.81";
string IpTerminal = "172.18.4.200";
string Mascara = "255.255.255.0";
string Linha1 = "Teste do comando ";
string Linha2 = "rconf";
int tempo = 4;

//Para definir os tamanhos que precisam ser passados para o protocolo, deve-se adici-
onar 48 (ou 0x30 em hexa) nos tamanhos das strings
char tamIPServer = (char)(IpServidor.Length + 48);
char tamIPCliente = (char)(IpTerminal.Length + 48);
char tamMascara = (char)(Mascara.Length + 48);
char tamTLinha1 = (char)(Linha1.Length + 48);
char tamTLinha2 = (char)(Linha2.Length + 48);

//Variável que guardará todas as informações que serão enviadas
string str = "#rconf" +
      tamIPServer + IpServidor +
      tamIPCliente + IpTerminal +
      tamMascara + Mascara +
```

```
            tamTLinha1 + Linha1 +
            tamTLinha2 + Linha2 +
            (char)(tempo + 48);

        byte[] comando = Encoding.ASCII.GetBytes(str); //Transforma a string em bytes

        cliente.Send(comando); //Envia o comando para o equipamento
```

Neste caso, não há valor de resposta do equipamento. Se o comando for bem executado, o Busca Preço G2 S irá reiniciar, e quando se conectar novamente ao servidor já estará com a nova configuração.

2.1.1.12) #reconf02

Comando	Resposta	Origem	Ação realizada
#reconf02 + dados	Nenhuma	Servidor	Altera Configurações do Terminal

#reconf02 + dados: Configura o terminal. Os dados de configuração que precisam ser passados são:

1 byte: tamanho da string do IP do servidor.
1 string: IP do servidor.
1 byte: tamanho da string do IP do terminal.
1 string: IP do terminal.
1 byte: tamanho da string da máscara de rede.
1 string: máscara de rede.
1 byte: tamanho da string do texto a linha 1.
1 string: texto da linha 1.
1 byte: tamanho da string do texto da linha 2.
1 string: texto da linha 2.
1 byte: tamanho da string do texto a linha 3.
1 string: texto da linha 3.
1 byte: tamanho da string do texto da linha 4.
1 string: texto da linha 4.
1 byte: tempo de exibição.

OBS: O valor do byte com o tamanho de cada string ou do tempo de exibição deve ser somado com 48 (decimal).

Exemplo em Java:

Envio do comando:

```
//Define os parâmetros que serão enviados na função
String IpServidor     = "172.18.4.81";
String IpTerminal   = "172.18.4.200";
String Mascara     = "255.255.255.0";
String Linha1     = "Teste de ";
String Linha2     = "Funcionalidade";
String Linha3     = "Do equipamento";
String Linha4     = "BPG2";
String tempo        = "4";

//Começo da string, e variável que guardará todas as informações que serão enviadas
String comandoCompleto = "#reconf02";

//Cálculo dos tamanhos dos parâmetros
int Tamanho1, Tamanho2, Tamanho3, Tamanho4, TamanhoIPserv, TamanhoIPterm,
```

```
TamanhoMascara, TempoDisplay;

        //Para definir os tamanhos que precisam ser passados para o protocolo, deve-se adicionar 48 (ou 0x30 em hexa) nos tamanhos das strings
        Tamanho1      = Linha1.length() + 0x30;
        Tamanho2      = Linha2.length() + 0x30;
        Tamanho3      = Linha3.length() + 0x30;
        Tamanho4      = Linha4.length() + 0x30;
        TamanhoIPserv  = IpServidor.length() + 0x30;
        TamanhoIPterm  = IpTerminal.length() + 0x30;
        TamanhoMascara    = Mascara.length() + 0x30;
        TempoDisplay = Integer.valueOf(tempo) + 0x30;

        //Monta o comando
        comandoCompleto += (char)TamanhoIPserv;
        comandoCompleto += IpServidor;
        comandoCompleto += (char)TamanhoIPterm;
        comandoCompleto += IpTerminal;
        comandoCompleto += (char)TamanhoMascara;
        comandoCompleto += Mascara;
        comandoCompleto += (char)Tamanho1;
        comandoCompleto += Linha1;
        comandoCompleto += (char)Tamanho2;
        comandoCompleto += Linha2;
        comandoCompleto += (char)Tamanho3;
        comandoCompleto += Linha3;
        comandoCompleto += (char)Tamanho4;
        comandoCompleto += Linha4;
        comandoCompleto += (char)TempoDisplay;

        byte[] command = comandoCompleto.getBytes();//Transforma a string em bytes
        out.write(command); //Envia o comando para o equipamento
        out.flush();
```

Neste caso, não há valor de resposta do equipamento. Se o comando for bem executado, o Busca Preço G2 S irá reiniciar, e quando se conectar novamente ao servidor já estará com a nova configuração.

Exemplo em C#:

Envio de comando:

```
        //Define os parâmetros que serão enviados na função
        string IpServidor = "172.18.4.81";
        string IpTerminal = "172.18.4.200";
        string Mascara = "255.255.255.0";
        string Linha1 = "Teste de ";
        string Linha2 = "Funcionalidade";
        string Linha3 = "Do equipamento";
        string Linha4 = "BPG2";
        int tempo = 4;

        //Para definir os tamanhos que precisam ser passados para o protocolo, deve-se adicionar 48 (ou 0x30 em hexa) nos tamanhos das strings
        char tamIPServer = (char)(IpServidor.Length + 48);
        char tamIPCliente = (char)(IpTerminal.Length + 48);
        char tamMascara = (char)(Mascara.Length + 48);
        char tamTLinha1 = (char)(Linha1.Length + 48);
        char tamTLinha2 = (char)(Linha2.Length + 48);
```

```
char tamTLinha3 = (char)(Linha3.Length + 48);
char tamTLinha4 = (char)(Linha4.Length + 48);

//Variável que guardará todas as informações que serão enviadas
string str = "#reconf02" +
       tamIPServer + IpServidor +
       tamIPCliente + IpTerminal +
       tamMascara + Mascara +
       tamTLinha1 + Linha1 +
       tamTLinha2 + Linha2 +
       tamTLinha3 + Linha3 +
       tamTLinha4 + Linha4 +
       (char)(tempo + 48);

byte[] comando = Encoding.ASCII.GetBytes(str); //Transforma a string em bytes

cliente.Send(comando); //Envia o comando para o equipamento
```

Neste caso, não há valor de resposta do equipamento. Se o comando for bem executado, o Busca Preço G2 S irá reiniciar, e quando se conectar novamente ao servidor já estará com a nova configuração.

2.1.1.13) #rextconf

Comando	Resposta	Origem	Ação realizada
#rextconf + dados	#rextconf_ok	Servidor	Altera Configurações do Terminal

#rextconf + dados: Semelhante ao comando anterior, este configura o terminal, porém com mais parâmetros. Os dados de configuração que precisam ser passados são:

1 byte: tamanho da string do IP do servidor.
1 string: IP do servidor.
1 byte: tamanho da string do IP do terminal.
1 string: IP do terminal.
1 byte: tamanho da string da máscara de rede.
1 string: máscara de rede.
1 byte: tamanho da string do Gateway.
1 string: Gateway.
1 byte: tamanho da string do Servidor de Nomes.
1 string: Servidor de nomes (Este campo será ignorado – Enviar uma string vazia).
1 byte: tamanho da string do Nome do Terminal.
1 string: Nome do terminal.
1 byte: tamanho da string do texto a linha 1.
1 string: texto da linha 1.
1 byte: tamanho da string do texto da linha 2.
1 string: texto da linha 2.
1 byte: tamanho da string Endereço do Servidor de Atualização.
1 string: Endereço do Servidor de Atualização. (Este campo será ignorado – Enviar uma string vazia).
1 byte: tamanho da string do Nome do Usuário.
1 string: Nome do Usuário. (Este campo será ignorado – Enviar uma string vazia).
1 byte: tamanho da string da Senha do Usuário.
1 string: Senha do Usuário. (Este campo será ignorado – Enviar uma string vazia).
1 byte: Tempo de Exibição.
1 byte: IP dinâmico/fixo. (48 = fixo, 49 = dinâmico)
1 byte: Procura Servidor. (48 = não procura pelo servidor, 49 = Procura) (Este campo será ignorado).

OBS1: O valor do byte com o tamanho de cada string ou do tempo de exibição deve ser somado com 48 (decimal).

OBS2: No caso das strings ignoradas pelo terminal, elas precisam ser vazias "". Dessa forma, deverá ser enviados os tamanhos das respectivas strings com o valor 48 (decimal). Estes campos foram mantidos para permitir a compatibilidade com as versões anteriores.

Exemplo em Java:

Envio do comando:

```
//Define os parâmetros que serão enviados na função
String IpServidor     = "172.18.4.81";
String IpTerminal   = "172.18.4.200";
String Mascara     = "255.255.255.0";
String Gateway     = "192.168.0.1";
String ServidorNomes = ""; //Parâmetro ignorado
String NomeTerminal = "Meu BPG2";
String Linha1     = "Teste de ";
String Linha2     = "Funcionalidade";
String EndereçoServidor = ""; //Parâmetro ignorado
String NomeUsuário = ""; //Parâmetro ignorado
String SenhaUsuário = ""; //Parâmetro ignorado
String tempo        = "5";
int TipoIP = 48; //IP fixo
int ProcuraServidor = 48; //Parâmetro ignorado

//Começo da string, e variável que guardará todas as informações que serão enviadas
String comandoCompleto = "#rextconf";

//Cálculo dos tamanhos dos parâmetros
int Tamanho1, Tamanho2,TamanhoIPserv, TamanhoIPterm,TamanhoMascara,
   TamanhoGateway, TamanhoServidor, TamanhoNome, TamanhoEndereço,
   TamanhoNomeUsuario, TamanhoSenha, TempoDisplay;

//Para definir os tamanhos que precisam ser passados para o protocolo, deve-se adicionar 48
(ou 0x30 em hexa) nos tamanhos das strings
Tamanho1       = Linha1.length() + 0x30;
Tamanho2       = Linha2.length() + 0x30;
TamanhoIPserv  = IpServidor.length() + 0x30;
TamanhoIPterm  = IpTerminal.length() + 0x30;
TamanhoMascara    = Mascara.length() + 0x30;
TamanhoGateway = Gateway.length() + 0x30;
TamanhoServidor = ServidorNomes.length() + 0x30;
TamanhoNome = NomeTerminal.length() + 0x30;
TamanhoEndereço = EndereçoServidor.length() + 0x30;
TamanhoNomeUsuario = NomeUsuário.length() + 0x30;
TamanhoSenha = SenhaUsuário.length() + 0x30;
TempoDisplay = Integer.valueOf(tempo) + 0x30;

//Monta o comando
comandoCompleto += (char)TamanhoIPserv;
comandoCompleto += IpServidor;
comandoCompleto += (char)TamanhoIPterm;
comandoCompleto += IpTerminal;
comandoCompleto += (char)TamanhoMascara;
comandoCompleto += Mascara;
comandoCompleto += (char)TamanhoGateway;
comandoCompleto += Gateway;
comandoCompleto += (char)TamanhoServidor;
comandoCompleto += ServidorNomes;
```

```
comandoCompleto += (char)TamanhoNome;
comandoCompleto += NomeTerminal;
comandoCompleto += (char)Tamanho1;
comandoCompleto += Linha1;
comandoCompleto += (char)Tamanho2;
comandoCompleto += Linha2;
comandoCompleto += (char)TamanhoEndereço;
comandoCompleto += EndereçoServidor;
comandoCompleto += (char)TamanhoNomeUsuario;
comandoCompleto += NomeUsuário;
comandoCompleto += (char)TamanhoSenha;
comandoCompleto += SenhaUsuário;
comandoCompleto += (char)TempoDisplay;
comandoCompleto += (char)TipoIP;
comandoCompleto += (char)ProcuraServidor;

byte[] command = comandoCompleto.getBytes();//Transforma a string em bytes
out.write(command); //Envia o comando para o equipamento
out.flush();
```

Neste caso, não há valor de resposta do equipamento. Se o comando for bem executado, o Busca Preço G2 S irá reiniciar, e quando se conectar novamente ao servidor já estará com a nova configuração.

Exemplo em C#:

Envio de comando:

```
//Define os parâmetros que serão enviados na função
string IpServidor = "172.18.4.81";
string IpTerminal = "172.18.4.200";
string Mascara = "255.255.255.0";
string Gateway = "192.168.0.1";
string ServidorNomes = ""; //Parâmetro ignorado
string NomeTerminal = "Meu BPG2";
string Linha1 = "Teste de ";
string Linha2 = "Funcionalidade";
string EndereçoServidor = ""; //Parâmetro ignorado
string NomeUsuário = ""; //Parâmetro ignorado
string SenhaUsuário = ""; //Parâmetro ignorado
int tempo = 5;
int TipoIP = 0; //IP fixo
int ProcuraServidor = 0; //Parâmetro ignorado


//Para definir os tamanhos que precisam ser passados para o protocolo, deve-se adici-
onar 48 (ou 0x30 em hexa) nos tamanhos das strings
char tamIPServer = (char)(IpServidor.Length + 48);
char tamIPCliente = (char)(IpTerminal.Length + 48);
char tamMascara = (char)(Mascara.Length + 48);
char tamGateway = (char)(Gateway.Length + 48);
char tamServidor = (char)(ServidorNomes.Length + 48);
char tamNome = (char)(NomeTerminal.Length + 48);
char tamEndereço = (char)(EndereçoServidor.Length + 48);
char tamNomeUsuario = (char)(NomeUsuário.Length + 48);
char tamSenha = (char)(SenhaUsuário.Length + 48);
char tamTLinha1 = (char)(Linha1.Length + 48);
char tamTLinha2 = (char)(Linha2.Length + 48);

//Variável que guardará todas as informações que serão enviadas
```

```
string str = "#rextconf" +
       tamIPServer + IpServidor +
       tamIPCliente + IpTerminal +
        tamMascara + Mascara +
        tamGateway + Gateway +
        tamServidor + ServidorNomes +
        tamNome + NomeTerminal +
        tamTLinha1 + Linha1 +
        tamTLinha2 + Linha2 +
        tamEndereço + EndereçoServidor +
        tamNomeUsuario + NomeUsuário +
        tamSenha + SenhaUsuário +
        (char)(tempo + 48) +
        (char)(TipoIP + 48) +
        (char)(ProcuraServidor + 48);

byte[] comando = Encoding.ASCII.GetBytes(str); //Transforma a string em bytes

cliente.Send(comando); //Envia o comando para o equipamento
```

Neste caso, não há valor de resposta do equipamento. Se o comando for bem executado, o Busca Preço G2 S irá reiniciar, e quando se conectar novamente ao servidor já estará com a nova configuração.

2.1.1.14) #rparamconfig

Comando	Resposta	Origem	Ação realizada
#rparamconfig + dados	#rparamconfig _ok	Servidor	Realiza configuração extra dos parâmetros de rede

#rparamconfig + dados: Comando que realiza as configurações extras dos parâmetros de rede.
Segue os dados de configuração:
1 byte: valor do IP dinâmico (48 = desativado, 49 = ativado).
1 byte: 48 (decimal).

Exemplo em Java:

Envio do comando:

```
//Define os parâmetros que serão enviados na função
int TipoIP = 49; //IP dinâmico
int extra = 48;

//Começo da string, e variável que guardará todas as informações que serão enviadas
String comandoCompleto = "#rparamconfig";

//Monta o comando
comandoCompleto += (char)TipoIP;
comandoCompleto += (char)extra;

byte[] command = comandoCompleto.getBytes();//Transforma a string em bytes
out.write(command); //Envia o comando para o equipamento
out.flush();
```

Resposta do equipamento:

```
String comando;
 byte vetor[] = new byte[255];
```

```
int qtdLida;
if (in.available() > 0) {
   qtdLida = in.read(vetor);
   comando = new String(vetor); //Resposta do BP
   String reciveCommand = comando.substring(0, qtdLida); //Resposta tratada
}
```

recieveComand = #rparamconfig_ok

Exemplo em C#:

Envio de comando:

```
//Define os parâmetros que serão enviados na função
int TipoIP = 1; //IP dinâmico
int ProcuraServidor = 0; //Parâmetro ignorado

//Variável que guardará todas as informações que serão enviadas
string str = "#rparamconfig" +
       (char)(TipoIP + 48) +
       (char)(ProcuraServidor + 48);

byte[] comando = Encoding.ASCII.GetBytes(str); //Transforma a string em bytes

cliente.Send(comando); //Envia o comando para o equipamento
```

Resposta do equipamento:

```
System.Threading.Thread.Sleep(500); //Tempo para garantir resposta do equipamento

String resposta = null; //Variável para guardar a resposta
byte[] dados = new byte[255]; //Cria um vetor de 255 bytes

cliente.Receive(dados);//Faz a leitura da resposta do Busca Preço

resposta = new System.Text.ASCIIEncoding().GetString(dados); //Converte os bytes para
texto
```

resposta = #rparamconfig_ok

2.1.1.15) #rupdconfig

Comando	Resposta	Origem	Ação realizada
#rupdconfig + dados	#rupdconfig_ok	Servidor	Realiza configuração de atualização do terminal

#rupdconfig + dados: Comando que realiza a configuração dos parâmetros de atualização do terminal. Os dados de configuração que precisam ser passados são:
1 byte: tamanho da string do Gateway.
1 string: Gateway.
1 byte: tamanho da string do Servidor de Nomes
1 string: Servidor de Nomes. (Este campo será ignorado – Enviar uma string vazia).
1 byte: tamanho da string do nome do terminal
1 string: Nome do terminal
1 byte: 61 (decimal).
1 string: “Não suportado” (sem as aspas).
1 byte: 61 (decimal).
1 string: “Não suportado” (sem as aspas).
1 byte: 61 (decimal).
1 string:. “Não suportado” (sem as aspas).

OBS1: O valor do byte com o tamanho de cada string deve ser somado com 48 (decimal).

OBS2: No caso das strings ignoradas pelo terminal, elas precisam ser vazias ””. Dessa forma, deverá ser enviados os tamanhos das respectivas strings com o valor 48 (decimal). Estes campos foram mantidos para permitir a compatibilidade com as versões anteriores.

Exemplo em Java:

Envio do comando:

```
//Define os parâmetros que serão enviados na função
String Gateway     = "192.168.0.1";
String ServidorNomes = ""; //Parâmetro ignorado
String NomeTerminal = "Meu BPG2";
String NS = "Não suportado";
int extra = 61;

//Começo da string, e variável que guardará todas as informações que serão enviadas
String comandoCompleto = "#rupdconfig";

//Cálculo dos tamanhos dos parâmetros
int TamanhoGateway, TamanhoServidor, TamanhoNome;

//Para definir os tamanhos que precisam ser passados para o protocolo, deve-se adicionar 48 (ou 0x30 em hexa) nos tamanhos das strings

TamanhoGateway = Gateway.length() + 0x30;
TamanhoServidor = ServidorNomes.length() + 0x30;
TamanhoNome = NomeTerminal.length() + 0x30;

//Monta o comando
comandoCompleto += (char)TamanhoGateway;
comandoCompleto += Gateway;
comandoCompleto += (char)TamanhoServidor;
comandoCompleto += ServidorNomes;
comandoCompleto += (char)TamanhoNome;
comandoCompleto += NomeTerminal;
comandoCompleto += (char)extra;
comandoCompleto += NS;
comandoCompleto += (char)extra;
comandoCompleto += NS;
comandoCompleto += (char)extra;
comandoCompleto += NS;

byte[] command = comandoCompleto.getBytes();//Transforma a string em bytes

out.write(command); //Envia o comando para o equipamento
out.flush();
```

Resposta do equipamento:

```
String comando;
 byte vetor[] = new byte[255];
 int qtdLida;
 if (in.available() > 0) {
    qtdLida = in.read(vetor);
    comando = new String(vetor); //Resposta do BP
```

```
            String reciveCommand = comando.substring(0, qtdLida); //Resposta tratada
        }
```

recieveComand = #rupdconfig_ok

Exemplo em C#:

Envio de comando:

```
        //Define os parâmetros que serão enviados na função
        string Gateway = "192.168.0.1";
        string ServidorNomes = ""; //Parâmetro ignorado
        string NomeTerminal = "Meu BPG2";
        string NS = "Não suportado";
        int extra = 61;

        //Para definir os tamanhos que precisam ser passados para o protocolo, deve-se adici-
onar 48 (ou 0x30 em hexa) nos tamanhos das strings
        char tamGateway = (char)(Gateway.Length + 48);
        char tamServidor = (char)(ServidorNomes.Length + 48);
        char tamNome = (char)(NomeTerminal.Length + 48);

        //Variável que guardará todas as informações que serão enviadas
        string str = "#rupdconfig" +
             tamGateway + Gateway +
             tamServidor + ServidorNomes +
             tamNome + NomeTerminal +
             (char)(extra) + NS +
             (char)(extra) + NS +
             (char)(extra) + NS;

        byte[] comando = Encoding.ASCII.GetBytes(str); //Transforma a string em bytes

        cliente.Send(comando); //Envia o comando para o equipamento
```

Resposta do equipamento:

```
        System.Threading.Thread.Sleep(500); //Tempo para garantir resposta do equipamento

        String resposta = null; //Variável para guardar a resposta
        byte[] dados = new byte[255]; //Cria um vetor de 255 bytes

        cliente.Receive(dados);//Faz a leitura da resposta do Busca Preço

        resposta = new System.Text.ASCIIEncoding().GetString(dados); //Converte os bytes para
texto
```

resposta = #rupdconfig_ok

2.1.1.16) #mesg

Comando	Resposta	Origem	Ação realizada
#mesg + dados	Nenhuma	Servidor	Mostra Mensagem no Display

#mesg + dados: Mostra no Display do terminal uma mensagem em um tempo determinado. As mensagens passadas não ficam salvas no terminal, ou seja, não é uma função de configuração de equipamento. Após a mensagem ser mostrada, as linhas do Busca Preço G2 S voltarão a mostrar os textos salvos nele. Os dados são formatados da seguinte forma:

1 byte: tamanho da string da mensagem da primeira linha.
1 string: mensagem da primeira linha.

1 byte: tamanho da string da mensagem da segunda linha.
1 string: mensagem da segunda linha.
1 byte: tempo de exibição.
1 byte: reservado, deve ser = 48.

OBS: O valor do byte com o tamanho de cada string ou do tempo de exibição deve ser somado com 48 (decimal).

Exemplo em Java:

Envio do comando:

```
//Define os parâmetros que serão enviados na função
String Linha1     = "Nova Linha 1";
String Linha2     = "Nova Linha 2";
String tempo        = "7"; //7 segundos para exibição da mensagem
int reservado = 48; //byte reservado

//Começo da string, e variável que guardará todas as informações que serão enviadas
String comandoCompleto = "#mesg";

//Cálculo dos tamanhos dos parâmetros
int Tamanho1, Tamanho2, TempoDisplay;

//Para definir os tamanhos que precisam ser passados para o protocolo, deve-se adicionar 48 (ou 0x30 em hexa) nos tamanhos das strings
Tamanho1      = Linha1.length() + 0x30;
Tamanho2      = Linha2.length() + 0x30;
TempoDisplay = Integer.valueOf(tempo) + 0x30;

//Monta o comando
comandoCompleto += (char)Tamanho1;
comandoCompleto += Linha1;
comandoCompleto += (char)Tamanho2;
comandoCompleto += Linha2;
comandoCompleto += (char)TempoDisplay;
comandoCompleto += (char)reservado;

byte[] command = comandoCompleto.getBytes();//Transforma a string em bytes
out.write(command); //Envia o comando para o equipamento
out.flush();
```

Neste caso, não há valor de resposta do equipamento. Se o comando for bem executado, o Busca Preço G2 S mostrará a mensagem passada nos parâmetros da função pelo tempo definido.

Exemplo em C#:

Envio de comando:

```
//Define os parâmetros que serão enviados na função
string Linha1 = "Nova Linha 1";
string Linha2 = "Nova Linha 2";
int tempo = 5;
int reservado = 48; //byte reservado

//Para definir os tamanhos que precisam ser passados para o protocolo, deve-se adicionar 48 (ou 0x30 em hexa) nos tamanhos das strings
char tamTLinha1 = (char)(Linha1.Length + 48);
```

```
char tamTLinha2 = (char)(Linha2.Length + 48);

 //Variável que guardará todas as informações que serão enviadas
 string str = "#mesg" +
       tamTLinha1 + Linha1 +
       tamTLinha2 + Linha2 +
       (char)(tempo + 48) +
       (char)(reservado + 48);

 byte[] comando = Encoding.ASCII.GetBytes(str); //Transforma a string em bytes

 cliente.Send(comando); //Envia o comando para o equipamento
```

Neste caso, não há valor de resposta do equipamento. Se o comando for bem executado, o Busca Preço G2 S mostrará a mensagem passada nos parâmetros da função pelo tempo definido.

2.1.1.17) #gif

Comando	Resposta	Origem	Ação realizada
#gif + dados	#gif_ok+índice ou #img_error	Servidor	Envia imagem a ser exibida na tela do terminal

#gif + dados: Comando enviado do servidor para o terminal para enviar imagens que serão exibidas na tela do terminal enquanto não são feitas consultas de preços ou exibida imediatamente. O terminal pode responder com ok seguido do índice da mensagem ou erro. Os dados são formados da seguinte maneira:

1 byte: índice da imagem, em hexadecimal codificado em ASCII:
- 00: imagem exibida imediatamente;
- 01 a FE: imagem do loop de imagens;
- FF: Reset do loop de imagens.

1 byte: número de loops, em hexadecimal codificado em ASCII:
- 00 a FF: número de vezes que o gif animado será repetido

1 byte: tempo de espera, em hexadecimal codificado em ASCII:
- 00 a FF: tempo em segundos em que a imagem será exibida no terminal.

6 bytes: tamanho, em hexadecimal, de cada quadro da imagem que será enviada.

1 word: checksum, em hexadecimal. Operação de “OU – Exclusivo” entre todos os bytes da imagem que seguem após o <ETB>. O equipamento não valida esse campo, porém é necessário seu envio. Recomenda-se enviar 0000.

<ETB>; separador entre cabeçalho e dados das imagens, 0x17 (hexadecimal).

OBS: Para modelo do Busca Preço G2 sem áudio o tamanho máximo da imagem/gif a ser mandado não pode exceder 192KB. Já para o modelo com áudio (Busca Preço G2 S) o tamanho máximo da imagem/gif a ser mandado não pode exceder 124KB, pois para esse modelo a memória do display é compartilhada com o áudio. É possível também enviar ao terminal mais de uma imagem em loop, desde que a soma deles não ultrapasse esse limite. Caso os requisitos sejam atendidos, o segundo gif começará logo após o término das repetições do 1° gif enviado.

OBS2: O tempo de exibição define o tempo que o gif enviado permanecerá na tela do Busca Preço G2 S. Para cálculo exato do valor, é necessário enviar um valor em hexadecimal que será convertido em ASCII. O equipamento então lerá o valor em hexa e converterá para decimal, e esse número é o valor real de tempo de exibição.

Exemplo1: manda-se 0x30 0x39 (09 em ASCII). Esse valor é lido pelo terminal em hexadecimal, e este irá interpretar como 9 segundos de exibição.
Exemplo2: manda-se 0x31 0x30 (10 em ASCII). Esse valor é lido como hexadecimal pelo equipamento, e será convertido para decimal. Ou seja, 10 em hexa equivale a 16 em decimal, e o tempo de exibição será 16 segundos.
Exemplo3: manda-se 0x35 0x31 (51 em ASCII). Esse valor será lido em hexadecimal e convertido para decimal. 0x51 = 81 decimal. A imagem ficará 81 segundos na tela do equipamento.
Exemplo4: deseja-se enviar 10 segundos de tempo de exibição. 10 decimal equivale a 0A em hexadecimal. Dessa forma, deve-se enviar o equivalente em ASCII destes dois caracteres, ou seja 0x30 (0 em ASCII) 0x41 (A em ASCII).

Exemplo em Java:

Envio do comando:

```
        //Diretório da imagem
        String diretory = "C:\\Users\\lkubo\\Downloads\\Backup\\SC501Protocol\\ampulheta.gif";

        byte[] gif = "#gif".getBytes(); //Começo do comando
        byte[] IndiceImagem = {0x30, 0x30}; //Indice = 00 -> Exibição imediata da imagem
        byte[] numeroLoops = {0x30, 0x35}; //Número de Loops = 05
        byte[] tempoGif = {0x30, 0x35}; //Tempo de Exibição = 05

        byte[] checksum = {0x30, 0x30, 0x30, 0x30}; //Checksum -> Não é implementado, mas deve ser enviado. Recomenda-se mandar 0000

        byte ETB = 0x17; //Byte para finalizar o envio

        long tamImg;
        File image = new File(diretory);
        tamImg = image.length();

        byte[] imageSize = String.format("%06X", tamImg).getBytes(); //Para obter o tamanho do gif

        //Monta envio com os parâmetros definidos acima
        byte[] command = {gif[0], gif[1], gif[2], gif[3], IndiceImagem[0], IndiceImagem[1], numeroLoops[0], numeroLoops[1], tempoGif[0], tempoGif[1], imageSize [0],
            imageSize [1], imageSize[2], imageSize[3], imageSize[4], imageSize[5], checksum[0], checksum[1], checksum[2], checksum[3], ETB};

        byte[] arrayGif = Files.readAllBytes(Paths.get(diretory)); //Obtem-se os bytes do gif
        out.write(command); //Envia o comando para o equipamento
        out.write(arrayGif); //Envia o gif para ser mostrado
        out.flush();
```

Resposta do equipamento:

```
        String comando;
         byte vetor[] = new byte[255];
         int qtdLida;
         if (in.available() > 0) {
            qtdLida = in.read(vetor);
            comando = new String(vetor); //Resposta do BP
            String reciveCommand = comando.substring(0, qtdLida); //Resposta tratada
         }
```

recieveComand = #gif_ok00

Exemplo em C#:

Envio de comando:

```
        //Diretório da imagem
        string diretory =
        "C:\\Users\\lkubo\\Downloads\\Backup\\SC501Protocol\\ampulheta.gif";
        FileInfo fi = new FileInfo(diretory);

        byte[] gif = Encoding.ASCII.GetBytes("#gif"); //Começo do comando
        byte[] IndiceImagem = { 0x30, 0x30 }; //Indice = 00 -> Exibição imediata da imagem
        byte[] numeroLoops = { 0x30, 0x35 }; //Número de Loops = 05
        byte[] tempoGif = { 0x30, 0x35 }; //Tempo de Exibição = 05

        byte[] checksum = { 0x30, 0x30, 0x30, 0x30 }; //Checksum -> Não é implementado, mas
deve ser enviado. Recomenda-se mandar 0000

        byte ETB = 0x17; //Byte para finalizar o envio

        long tamImg = fi.Length; //Tamanho do gif

        byte[] imageSize = Encoding.ASCII.GetBytes(tamImg.ToString("X6")); //Transforma em
bytes

        //Monta envio com os parâmetros definidos acima
        byte[] command = {gif[0], gif[1], gif[2], gif[3], IndiceImagem[0], IndiceImagem[1],
numeroLoops[0], numeroLoops[1], tempoGif[0], tempoGif[1], imageSize[0],
        imageSize[1], imageSize[2], imageSize[3], imageSize[4], imageSize[5], checksum[0],
checksum[1], checksum[2], checksum[3], ETB};

        byte[] arrayGif = File.ReadAllBytes(diretory); //Obtem-se os bytes do gif

        cliente.Send(command); //Envia o comando para o equipamento
        cliente.Send(arrayGif); //Envia o comando para o equipamento
```

Resposta do equipamento:

```
        System.Threading.Thread.Sleep(500); //Tempo para garantir resposta do equipamento

        String resposta = null; //Variável para guardar a resposta
        byte[] dados = new byte[255]; //Cria um vetor de 255 bytes

        cliente.Receive(dados);//Faz a leitura da resposta do Busca Preço

        resposta = new System.Text.ASCIIEncoding().GetString(dados); //Converte os bytes para
texto
```

resposta = #gif_ok00

2.1.1.18) #playaudiowithmessage

Comando	Resposta	Origem	Ação realizada
#playaudiowithmessage + dados	#playaudiowithmessage_ok ou #playaudiowithmessage_error	Servidor	Envia um Áudio com ou sem nome do Preço e Produto no Display

#playaudiowithmessage + dados: Comando enviado do servidor para o terminal para enviar áudio com mensagens da descrição do produto e preço que serão exibidas na tela do terminal. O terminal pode responder com ok seguido ou erro. Os dados são formados da seguinte maneira:

6 bytes: tamanho, em hexadecimal, do áudio que será enviado.

1 byte: duração do áudio, em hexadecimal codificado em ASCII:
2 - 7: tempo de duração do áudio (segundos);

1 byte: volume do áudio, em hexadecimal codificado em ASCII:
0: baixo;
1: médio;
2: alto;
3: muito alto.

Os próximos bytes que compõe os dados do comando áudio refere-se o envio da descrição do produto e preço ou a mensagem de “produto não cadastrado” no display do terminal:

Para exibir a mensagem de “produto não cadastrado” utilizar os seguintes bytes:

6 bytes: utilizar a string nfound;

Para exibir a descrição do produto e preço utilizar os seguintes bytes:

2 bytes: tamanho do campo de descrição do produto;

Variável: descrição do produto;

2 bytes: tamanho do campo de descrição do preço;

Variável: descrição do preço;

OBS: O tamanho máximo do audio/wav a ser mandado não pode exceder 68KB e tem que ter no mínimo 16KB. O tempo do áudio está limitado entre 2 a 7 segundos de execução. Para converter o áudio a ser enviado para o terminal é necessário utilizar o conversor ffpmeg que disponibilizamos. O formato do arquivo é o seguinte: WAV - 8Khz - Mono (1 canal) - 8 bits - PCM_U8.

2.1.1.19)

Comando	Resposta	Origem	Ação realizada
# + dados	#nfound ou #nome|preço	Terminal	Mostra Nome e Preço do Produto no Display

 # + dados: Caso algum código de barras seja passado no *scanner* do terminal, este envia os dados com código de barras para o servidor.

Exemplo: ao passar o código de barras: 123456 no *scanner* do terminal o mesmo é enviado para o servidor a seguinte string: #123456.

O servidor pode responder com o nome e preço do produto, ou envia mensagem de produto não cadastrado. (#nfound).
Os dados com o nome e preço do produto são formatados da seguinte forma:

+ string com nome do produto (4 linhas x 20 colunas = 80 bytes)+ | + string com o preço do produto (1 linha x 20 colunas = 20 bytes)

OBS: Não é permitido o caractere # (octothorpe) na string com o preço do produto.

2.1.1.20) #macaddr?

Comando	Resposta	Origem	Ação realizada

#macaddr?	# macaddr + dados	Servidor	Solicita o MAC Address do terminal

#macaddr?: Este comando solicita o MAC Address do terminal. O terminal responde com os seguintes dados:

1 byte: Atual interface de rede selecionada: 0 (Ethernet) e 1 (WIFI)
1 byte: tamanho da string do MAC Address
1 string: MAC Address: XX:XX:XX:XX:XX:XX (o valor vai variar de acordo com cada terminal e interface selecionada)

OBS: Para saber o tamanho real de cada string, devemos subtrair 48 (decimal) do valor de cada byte.

Exemplo de configurações de um terminal:

Rede Ethernet
MAC: 00:1D:5B:00:65:A8

Para se calcular o resultado dos bytes referente ao tamanho dos dados, deve-se adicionar 48 no tamanho e transformar esse valor em ASCII. Ou seja, com um MAC Addres de 00:1D:5B:00:65:A8, o tamanho da string é de 17 caracteres. Ao adicionar 48, resulta em 64, ou seja, o caractere A na tabela ASCII.

Tamanho do MAC = 17 + 48 = 64 (A em ASCII)

São esses valores convertidos que serão respondidos pelo Busca Preço G2 S nos parâmetros de byte. Caso só se tenha a resposta do equipamento, e se deseje obter o tamanho original do texto, deve-se converter o caractere de resposta do equipamento no seu valor decimal correspondente e subtrair 48.

Exemplo em Java:

Envio do comando:

```
byte[] command = "#macaddr?".getBytes(); //Transforma a string em bytes
out.write(command); //Envia o comando para o equipamento
out.flush();
```

Resposta do equipamento:

```
String comando;
 byte vetor[] = new byte[255];
 int qtdLida;
 if (in.available() > 0) {
    qtdLida = in.read(vetor);
    comando = new String(vetor); //Resposta do BP
    String reciveCommand = comando.substring(0, qtdLida); //Resposta tratada
 }
```

recieveComand = #macaddr0A00:1D:5B:00:65:A8

Exemplo em C#:

Envio de comando:

```
byte[] comando = Encoding.ASCII.GetBytes("#macaddr?"); //Transforma a string em bytes
```

```
cliente.Send(comando); //Envia o comando para o equipamento
```

Resposta do equipamento:

```
System.Threading.Thread.Sleep(500); //Tempo para garantir resposta do equipamento

String resposta = null; //Variável para guardar a resposta
byte[] dados = new byte[255]; //Cria um vetor de 255 bytes

cliente.Receive(dados);//Faz a leitura da resposta do Busca Preço

resposta = new System.Text.ASCIIEncoding().GetString(dados); //Converte os bytes para
texto
```

resposta = #macaddr0A00:1D:5B:00:65:A8

2.1.1.21) #fullmacaddr?

Comando	Resposta	Origem	Ação realizada
#fullmacaddr?	# fullmacaddr + dados	Servidor	Solicita o MAC Address da Ethernet e Wifi do terminal

#fullmacaddr?: Este comando solicita o MAC Address Ethernet e Wifi do terminal. O terminal responde com os seguintes dados:

1 byte: estado da interface: 0 (desabilitado) e 1 (habilitado)
1 byte: tamanho da string do MAC Address Ethernet
1 string: MAC Address Ethernet: XX:XX:XX:XX:XX:XX (o valor vai variar de acordo com cada terminal)
1 byte: estado da inteface: 0 (desabilitado) e 1 (habilitado)
1 byte: tamanho da string do MAC Address Wifi
1 string: MAC Address Wifi: XX:XX:XX:XX:XX:XX (o valor vai variar de acordo com cada terminal)

OBS: Para saber o tamanho real de cada string, devemos subtrair 48 (decimal) do valor de cada byte.

Exemplo de configurações de um terminal:

MAC Ethernet: 00:1D:5B:00:65:A8
MAC Wifi: 89:88:5A:3A:2C:79

Para se calcular o resultado dos bytes referente ao tamanho dos dados, deve-se adicionar 48 no tamanho e transformar esse valor em ASCII. Ou seja, com um MAC Addres de 00:1D:5B:00:65:A8, o tamanho da string é de 17 caracteres. Ao adicionar 48, resulta em 64, ou seja, o caractere A na tabela ASCII.

Tamanho do MAC = 17 + 48 = 64 (A em ASCII)

São esses valores convertidos que serão respondidos pelo Busca Preço G2 S nos parâmetros de byte. Caso só se tenha a resposta do equipamento, e se deseje obter o tamanho original do texto, deve-se converter o caractere de resposta do equipamento no seu valor decimal correspondente e subtrair 48.

OBS: Este comando é compatível apenas com o Busca Preço G2 S (firmware de referência: 3.X.X).

2.1.1.22) #audioconfig?

Comando	Resposta	Origem	Ação realizada

#audioconfig?	# audioconfig + dados	Servidor	Solicita o estado do áudio do terminal

#audioconfig?: Este comando solicita o estado do áudio do terminal. O terminal responde com os seguintes dados:

1 byte: estado do áudio: 0 (desabilitado) e 1 (habilitado)

OBS: Caso o estado do áudio esteja desativado, o equipamento não irá executar o áudio de "Consultando o produto aguarde", "Ops. Tente novamente" (para falha na resposta da consulta) e também não aceitará o comando #playaudiowithmessage, e o mesmo retornará um erro para o protocolo.

OBS: Este comando é compatível apenas com o Busca Preço G2 S (firmware de referência: 3.X.X).

2.1.1.22) #raudioconfig

Comando	Resposta	Origem	Ação realizada
#raudioconfig + dados	#raudioconfig_ok	Servidor	Configura o estado do áudio do terminal

#raudioconfig: Este comando configura o estado do áudio do terminal. O terminal responde com os seguintes dados:

1 byte: estado do áudio: 0 (desabilitado) e 1 (habilitado).

OBS: Este comando é compatível apenas com o Busca Preço G2 S (firmware de referência: 3.X.X).

Protocolo de Comunicação (HTTP)

A porta de comunicação do protocolo HTTP (Web Socket) obedecerá ao valor definido no campo (Porta) do menu do terminal.
Exemplo de requisição HTTP caso o campo (HTTP GET) no menu do terminal esteja preenchido da seguinte forma:
http://192.168.0.80:8080/priceserver/consulta?codEan=

O servidor HTTP receberá a String acima concatenado com o código de barras lido no terminal:
http://192.168.0.80:8080/priceserver/consulta?codEan=012345678901

Logo após receber a requisição se faz necessário responder o comando para o terminal no seguinte padrão:
<body>|linha1 da descrição do produto|linha2 da descrição do produto||||preço 1|preço 2|</body>