Registrador Eletrônico de Ponto (REP)
Família iPointLine

Manual de uso da *DLL*

Índice

1. Introdução

Este é o manual de referência do funcionamento da DLL “authotelcom.dll”, o qual realiza a comunicação direta com o REP, via cabo *Ethernet* e protocolo TCP/IP, realizando a criptografia necessária (AES-128) e disponibilizando uma interface de fácil utilização para o aplicativo final, o qual é instalado no computador e utilizado pelo usuário final.

2. Funções e Procedimentos

Segue a listagem de todas as funções (*functions)* e procedimentos (*procedures*) disponibilizadas pela *DLL* para a comunicação com o REP:

- configura.
- recebeMarcacoesTCP.
- enviaTrabalhadorTCP.
- enviaEmpregadorTCP.
- enviaDigitaisTCP.
- leEmpregadorTCP.
- leTrabalhadorTCP.
- leDigitaisTCP.
- fecharComunicacao.

2.1. Tipos de dados a serem passados para as funções

Os tipos de dados a seguir devem ser passados como referência para todas as funções, exceto para a “configura”:

```
TMarcacao = record
  nsr: LongWord;
  cont: LongWord;
  pis: string[12];
  dia: byte;
  mes: byte;
  ano: word;
  hora: byte;
  minuto: byte;
end;

TControle = record
  total: Word;
  atual: Word;
  start: boolean;
  erro: Shortint;
  porta: Word;
  s_tipo: byte;
  modelo: byte;
  endereco: string[15];
  backup: ShortString;
```

```
  baudrate: integer;
end;

TDados = record
  adcOUSubst: ShortString;
  pin: ShortString;
  pis: ShortString;
  identificador: ShortString;
  cei: ShortString;
  razaoSocial: ShortString;
  localPrestServ: ShortString;
  tipoId: ShortString;
  nome: ShortString;
  id_bio: ShortString;
  numCartao: ShortString;
  senha: ShortString;
  mestre: ShortString;
  verifica: ShortString;
end;

TDigitais = record
  pin: ShortString;
  dedo: AnsiString;
end;
```

2.2. Função “configura”

Esta função é utilizada para:

- Ajustar hora.
- Ler hora.
- Ajustar horário de verão.
- Ler horário de verão.
- Enviar quantidade de papel.
- Ler quantidade de papel.
- Enviar sinalização de pouco papel.
- Ler sinalização de pouco papel.

2.2.1. Declaração

```
function configura(tipo, ip, host, porta, end_dev, datahorai,
datahoraf, diasSem, diasSemF, info: PChar; com, total, atual,
duracaoToque, flag, config, qtde_rel, baud: integer): pchar;
stdcall; external 'authotelcom.dll';
```

2.2.2. Definição dos parâmetros

- ***tipo*** (*obrigatório*) – indica qual o tipo de dado a ser enviado para o REP. Os tipos são:

TIPO	DEFINIÇÃO
AR	Ajuste de hora
LR	Leitura da hora
AH	Ajuste de horário de verão
LH	Leitura do horário de verão
GP	Envio de quantidade de papel
LP	Leitura da quantidade de papel
GO	Envio de sinalização de pouco papel
LO	Leitura da sinalização de pouco papel

- ***end_dev*** (*obrigatório*) – deve permanecer em '**U,00'**.
- ***ip*** (obrigatório) – indica o IP do REP.
- ***porta*** (obrigatório) – indica a porta de comunicação utilizada. Ex.: 1001.
- ***host*** (opcional) – indica o endereço do computador que iniciou a comunicação.
- ***datahorai*** (obrigatório para AR e AH) – data/hora para ajuste de relógio e do horário de verão. Para as leituras, este parâmetro pode ficar vazio.
- ***datahoraf*** (obrigatório para o AH) – sinaliza a data/hora do final do horário de verão.
- ***diasSem*** – deve permanecer em vazio.
- ***diaSemF*** – deve permanecer vazio.
- ***info*** (obrigatório para o GO e GP) – informação a respeito da quantidade de papel.
- ***com*** – deve permanecer com o conteúdo **2**.
- ***total*** – deve permanecer com o conteúdo **0.**
- ***atual*** – deve permanecer com o conteúdo **0.**
- ***duracaoToque*** – deve permanecer com o conteúdo **0.**
- ***flag*** – deve permanecer com o conteúdo **0.**
- ***config*** – deve permanecer com o conteúdo **0.**
- ***qtde_rel*** – deve permanecer com o conteúdo **0.**
- ***baud*** – deve permanecer com o conteúdo **9600.**

Obs.: caso não seja possível estabelecer comunicação com o REP o retorno será '-1'.

2.2.3. Ajustar hora

O formato para o parâmetro *datahorai* é 'dd/mm/yyyy hh:mm:ss'. O retorno da função é o endereço do REP + 0 ou código de erro (3. Códigos dos erros). Obs.: o zero (0) indica que a função foi executada com sucesso.

Exemplo:

```
ip := '192.168.1.101'; // Colocar aqui o IP do REP
porta := '1001'; // Colocar aqui a porta
com := 2;
end_dev := '0';
end_con := end_dev;
end_dev := 'U,' + end_dev;
hora := formatdatetime('dd/mm/yyyy hh:MM:ss', now);

if (configura('AR', pchar(ip), '', pchar(porta), pchar(end_dev), pchar(hora), pchar('0'),
    pchar('0'), pchar('0'), pchar('01'), com, 1, 1, 0, 0, 1, 1, 9600)) = end_con + ' 0' then
begin
  ShowMessage('Relógio ajustado com sucesso!');
end
else
begin
  ShowMessage('Não foi possível ajustar o relógio!');
end;
```

2.2.4. Ler hora

O retorno da função é o endereço do REP + data/hora no formato 'dd/mm/yyyy hh:mm:ss' ou código de erro (3. Códigos dos erros).

Exemplo:

```
var
  ip, porta, end_dev, hora, end_con, resp: string;
  com: integer;
begin
  ip := '10.0.0.227';
  porta := '1001';
  com := 2;
  end_dev := '0';
  end_dev := 'U,' + end_dev;

  resp := '';
  resp := configura('LH', pchar(ip), pchar(''), pchar(porta), pchar(end_dev), pchar('0'),
      pchar('0'), pchar('0'), pchar('0'), pchar('01'), com, 1, 1, 0, 0, 1, 1, 9600);

  if (Length(resp) > 02) then
  begin
    ShowMessage(resp);
  end
  else
  begin
    ShowMessage('Não foi possível ler a hora!');
  end;
end;
```

2.2.5. Ajustar horário de verão

O formato para o parâmetro *datahorai* e *datahoraf* é 'dd/mm/yyyy hh:mm'. O retorno da função é o endereço do REP + 0 ou código de erro (3. Códigos dos erros).

Exemplo:

```
ip := '192.168.1.101'; // Colocar aqui o IP do REP
porta := '1001'; // Colocar aqui a porta
com := 2;
end_dev := '0';
end_con := end_dev;
end_dev := 'U,' + end_dev;
horai := formatdatetime('dd/mm/yyyy', DateTimePicker1.Date) + ' ' + ComboBox1.Text;
horaf := formatdatetime('dd/mm/yyyy', DateTimePicker2.Date) + ' ' + ComboBox2.Text;

if (configura('AH', pchar(ip), '', pchar(porta), pchar(end_dev), pchar(horai), pchar(horaf),
    pchar('0'), pchar('0'), pchar('01'), com, 1, 1, 0, 0, 1, 1, 9600)) = end_con + ' 0' then
begin
  ShowMessage('Horário de verão ajustado com sucesso!');
end
else
begin
  ShowMessage('Não foi possível ajustar o horário de verão!');
end;
```

2.2.6. Ler horário de verão

O retorno da função é o endereço do REP + data/hora de início + data/hora de fim no formato 'dd/mm/yyyy hh:mm dd/mm/yyyy hh:mm' ou código de erro (3. Códigos dos erros). Se o REP não possuir horário de verão cadastrado, o retorno será o endereço do REP + '00/00/0000 00:00 00/00/0000 00:00'.

Exemplo:

```
ip := '192.168.1.101'; // Colocar aqui o IP do REP
porta := '1001'; // Colocar aqui a porta
com := 2;
end_dev := '0';
end_dev := 'U,' + end_dev;

resp := '';
resp := configura('LH', pchar(ip), '', pchar(porta), pchar(end_dev), pchar('0'),
    pchar('0'), pchar('0'), pchar('0'), pchar('01'), com, 1, 1, 0, 0, 1, 1, 9600);

if length(resp) > 10 then
begin
  ShowMessage(resp);
end
else
begin
  ShowMessage('Não foi possível ler o horário de verão!');
end;
```

2.2.7. Enviar quantidade de papel

O formato para o parâmetro *info* deve ser em milímetros. Ex.: Bobina de 30m (30*1000), *info* = 30000. O retorno da função é o endereço do REP + 0 ou código de erro (3. Códigos dos erros).

Exemplo:

```
ip := '192.168.1.101'; // Colocar aqui o IP do REP
porta := '1001'; // Colocar aqui a porta
com := 2;
end_dev := '0';
end_con := end_dev;
end_dev := 'U,' + end_dev;
mm_papel := edt_m.Text;
mm_papel := FloatToStr(StrToFloat(mm_papel) * 1000);

ret := '';
ret := configura('GP', pchar(ip), '', pchar(porta), pchar(end_dev), pchar('0'),
    pchar('0'), pchar('0'), pchar('0'), pchar(mm_papel), com, 1, 1, 0, 0, 1, 1,  9600);

if (ret = end_con + ' 0') then
begin
  ShowMessage('Quantidade de papel ajustada com sucesso!');
end
else
begin
  ShowMessage('Não foi possível ajustar a quantidade de papel!');
end;
```

2.2.8. Ler quantidade de papel

O retorno da função é o endereço do REP + tamanho (em milímetros) ou código de erro (3. Códigos dos erros).

Exemplo:

```
ip := '192.168.1.101'; // Colocar aqui o IP do REP
porta := '1001'; // Colocar aqui a porta
com := 2;
end_dev := '0';
end_dev := 'U,' + end_dev;

resp := '';
resp := configura('LP', pchar(ip), '', pchar(porta), pchar(end_dev), pchar('0'),
    pchar('0'), pchar('0'), pchar('0'), pchar('01'), com, 1, 1, 0, 0, 1, 1, 9600);

if length(resp) > 2 then
begin
  ShowMessage(resp);
end
else
begin
  ShowMessage('Não foi possível ler a quantidade de papel!');
end;
```

2.2.9. Enviar sinalização de pouco papel

O formato para o parâmetro *info* deve ser em milímetros. Ex.: Avisar quando a bobina menor que 5m (5*1000), *info* = 5000. O retorno da função é o endereço do REP + 0 ou código de erro (3. Códigos dos erros).

Exemplo:

```
ip := '192.168.1.101'; // Colocar aqui o IP do REP
porta := '1001'; // Colocar aqui a porta
com := 2;
end_dev := '0';
end_con := end_dev;
end_dev := 'U,' + end_dev;
mm_papel := edt_m.Text;
mm_papel := FloatToStr(StrToFloat(mm_papel) * 1000);

if (configura('GO', pchar(ip), '', pchar(porta), pchar(end_dev), pchar('0'), pchar('0'),
    pchar('0'), pchar('0'), pchar(mm_papel), com, 1, 1, 0, 0, 1, 1,  9600)) = end_con + ' 0' then
begin
  ShowMessage('Sinalização de pouco papel enviada com sucesso!');
end
else
begin
  ShowMessage('Não foi possível enviar a sinalização de pouco papel!');
end;
```

2.2.10. Ler sinalização de pouco papel

O retorno da função é o endereço do REP + tamanho (em milímetros) ou código de erro (3. Códigos dos erros).

Exemplo:

```
ip := '192.168.1.101'; // Colocar aqui o IP do REP
porta := '1001'; // Colocar aqui a porta
com := 2;
end_dev := '0';
end_dev := 'U,' + end_dev;

resp := '';
resp := configura('LO', pchar(ip), '', pchar(porta), pchar(end_dev), pchar('0'),
    pchar('0'), pchar('0'), pchar('0'), pchar('01'), com, 1, 1, 0, 0, 1, 1, 9600);

if length(resp) > 2 then
begin
  ShowMessage(resp);
end
else
begin
  ShowMessage('Não foi possível ler a sinalização de pouco papel!');
end;
```

2.3. Função “recebeMarcacoesTCP”

Função utilizada para ler as marcações do REP. O retorno se encontra nos parâmetros que são passados por referência.

O valor a ser passado no parâmetro *evento* deve ser o próprio NF do REP. Obs.: o valor recebido em *marcacao.contador* pode ser descartado.

2.3.1. Declaração

```
function recebeMarcacoesTCP(var marcacao: array of TMarcacao; var controle: Tcontrole; evento: integer): boolean; stdcall; external 'authotelcom.dll';
```

2.3.2. Parâmetros e retorno

A primeira chamada da função deverá enviar a variável *controle.start* como *true.* Ao continuar recebendo as marcações, a variável passa a ser *false.*

O registro *controle.modelo* deve conter o valor **9**.

O *evento* deve conter o NS do REP.

As variáveis *controle.porta* (porta de comunicação de rede do REP. Ex.: 1001), *controle.s_tipo* (fixo em **2**), *controle.endereco* (endereço IP do REP. Ex.: 10.0.0.227) e *controle.baudrate* (fixo em **9600**) também deverão ser preenchidas, pois são campos obrigatórios.

Se o retorno da função for *false* e as variáveis do registro *controle* estiverem nulas, houve falha na comunicação. Se for *false* e a variável *controle.erro* for igual a 96, então não há registros de marcações no REP.

Se o retorno for *true*, as variáveis do registro *marcacao* terão os valores referentes à marcação atual e o procedimento de chamada da função deverá ser repetido até que o retorno da função seja *false* ou que a variável *controle.total* seja igual à variável *controle*.*atual*.

A função retorna até 10 marcações por vez, dependendo da configuração.

Caso ocorra algum erro no meio da comunicação, a variável *controle.erro* receberá o valor que indica qual o erro (3. Códigos dos erros) ocorrido.

2.3.3. Configuração da *DLL*

O diretório onde a *DLL* se encontrar deve possuir um arquivo de nome “config.rwt” contendo o valor de 1 a 10, o qual representa quantas marcações deverão ser recebidas por vez no máximo.

2.3.4. Exemplo

```
var
  controle: Tcontrole;
  marcacao: array [1..10] of Tmarcacao;
  recebe: boolean;
begin
  // Inicia as váriaveis
  controle.modelo   := 9; // Colocar aqui o modelo do REP usado
  controle.s_tipo   := 2;
  controle.endereco := '10.0.0.227'; // Colocar aqui o IP do REP
  controle.porta    := 1001; // Colocar aqui a porta
  controle.baudrate := 9600;

  // Aqui o sistema coleta corretamente as marcações
  controle.start := true;
  recebe := recebeMarcacoesTCP(marcacao, controle, 0); // 0 = todas as marcações
  controle.start := false;

  if (recebe) then
  begin
    while ((controle.total > controle.atual) and (recebeMarcacoesTCP(marcacao, controle, 0))) do
    begin
      { ************************************
        Tratar aqui as marcações recebidas...
      ************************************ }
    end;
  end
  else if (controle.erro = 96) then
  begin
    ShowMessage('Não há marcações a serem recebidas!');
  end
  else
  begin
    ShowMessage('Falha na comunicação!');
  end;

  // Necessário fechar a comunicação com o REP
  fecharComunicacao;
end;
```

2.4. Função “enviaTrabalhadorTCP”

Função utilizada para enviar dados dos funcionários para o REP. O retorno se encontra nos parâmetros que são passados por referência.

2.4.1. Declaração

```
function enviaTrabalhadorTCP(var dados: TDados; var controle:
Tcontrole): boolean; stdcall; external 'authotelcom.dll';
```

2.4.2. Parâmetros e retorno

A primeira chamada da função deverá enviar a variável *controle.start* como *true.* Ao continuar enviando os funcionários, a variável passa a ser *false.*

O registro *controle.modelo* deve conter o valor **9**.

As variáveis *controle.porta* (porta de comunicação de rede do REP. Ex.: 1001), *controle.s_tipo* (fixo em **2**), *controle.endereco* (endereço IP do REP. Ex.: 10.0.0.227) e *controle.baudrate* (fixo em **9600**) também deverão ser preenchidas, pois são campos obrigatórios.

A variável *controle.backup* dever conter 'N'.

As variáveis do registro *dados* deverão ser preenchidas com as informações referentes ao funcionário. Para cada funcionário enviado, o valor da variável *controle.atual* deverá ser incrementada de 1.

A variável *dados.adcOUSubst* terá importância na primeira chamada da função, essa variável indica se os funcionários enviados a seguir serão adicionados, substituídos, excluídos ou adicionados/substituídos:

- **Adicionar**: para incluir um funcionário no REP, o comando a ser enviado na variável *dados.adcOUSubst* será a letra **'A'**. Nesse caso, ele apenas inclui o funcionário na lista do REP, caso tente enviar um funcionário que possui um mesmo PIS já cadastrado no REP ou reenviar algum funcionário, haverá um código de erro (3. Códigos dos erros) como retorno.
- **Substituir**: para substituir um funcionário já cadastrado no REP, o comando a ser enviado na variável *dados.adcOUSubst* será a letra **'S'.** Nesse caso, ele apenas substitui os dados do funcionário cadastrado no REP pelos dados recebidos. O funcionário é localizado pelo PIS. Caso tente enviar dados de um funcionário cujo PIS inexiste no relógio, haverá um código de erro (3. Códigos dos erros) como retorno.
- **Excluir**: para excluir um funcionário do relógio, o comando a ser enviado na variável *dados.adcOUSubst* será a letra **'E'**. Nesse caso, ele exclui o funcionário da lista do REP. O funcionário a ser excluído é localizado pelo PIS. Caso tente excluir um funcionário cujo PIS inexiste no REP, haverá um código de erro (3. Códigos dos erros) como retorno.
- **Adicionar/Substituir**: para adicionar/substituir um funcionário no relógio, o comando a ser enviado na variável *dados.adcOUSubst* poderá ser qualquer

caractere diferente de **'A'**, **'S'** e **'E'**. Nesse caso, ele tenta incluir o funcionário na lista do REP, se o PIS já estiver cadastrado no relógio, o sistema tenta substituir os dados do funcionário. Se não for possível executar nenhuma das duas opções, haverá um código de erro (3. Códigos dos erros) como retorno.

Os campos a serem preenchidos no registro *dados* com as informações dos funcionários (todos os campos referentes às informações dos funcionários são obrigatórios, exceto o campo *dados.id_bio*):

- *dados.adcOUSubst* – indica qual o tipo de ação a ser executada (definida acima).
- *dados.pin* – número de identificação do funcionário (código, matrícula...).
- *dados.pis* – número do PIS do funcionário.
- *dados.identificador* – usado no envio de empregador.
- *dados.cei* – usado no envio de empregador.
- *dados.razaoSocial* – usado no envio de empregador.
- *dados.localPrestServ* – usado no envio de empregador.
- *dados.tipoId* – usado no envio de empregador.
- *dados.nome* – nome do funcionário (máximo de 52 caracteres).
- *dados.id_bio* – sempre 0 (controle do *firmware*).
- *dados.numCartao* – número do cartão de proximidade ou código de barras.
- *dados.senha* – senha numérica, máximo de 6 dígitos (campo opcional, porém, caso seja cadastrado, deve ter, no mínimo, 3 dígitos).
- *dados.mestre* – indica se o funcionário é mestre (0 – funcionário comum / 1 – funcionário mestre).
- *dados.verifica* – campo não utilizado nessa versão.

Se o retorno for *true*, os dados do próximo funcionário poderão ser enviados para o REP. Caso o retorno da função seja *false,* houve falha na comunicação.

Caso ocorra algum erro no meio da comunicação, a variável *controle.erro* receberá o valor que indica qual o erro ocorrido. Os valores podem ser visualizados em 3. Códigos dos erros.

2.4.3. Exemplo

```
var
  controle: Tcontrole;
  dados: Tdados;
begin
  Controle.modelo   := 9; // Colocar aqui o modelo do REP usado
  Controle.s_tipo   := 2;
  Controle.endereco := '10.0.0.227'; // IP do REP
  Controle.porta    := 1001;
  Controle.baudrate := 9600;
  Controle.erro     := 0;
  Controle.backup   := 'N';
  Controle.atual    := 1;
  Controle.total    := 1;
  Controle.start    := True;

  dados.nome        := 'José';
  dados.pin         := '91254';
  dados.pis         := '011111111116';
  dados.id_bio      := '0';
  dados.numCartao   := '7208973';
  dados.senha       := '1234';
  dados.mestre      := '0';
  dados.adcOUSubst  := 'W';

  if enviaTrabalhadorTCP(dados, controle) then
  begin
    controle.start := false;
    enviaTrabalhadorTCP(dados, controle);

    if controle.erro <> 0 then
      ShowMessage('Erro ao enviar funcionário!')
    else
      ShowMessage('Funcionário enviado com sucesso!');

    // Necessário fechar a comunicação com o REP
    fecharComunicacao;
  end
  else begin
    ShowMessage('Não foi possível o envio do funcionário!');
  end;
end;
```

2.5. Função “enviaDigitaisTCP”

Função utilizada para enviar digitais dos funcionários para o REP. O retorno se encontra nos parâmetros que são passados por referência.

2.5.1. Declaração

```
function enviaDigitaisTCP(var digitais: Tdigitais; var controle:
Tcontrole): boolean; stdcall; external 'authotelcom.dll';
```

2.5.2. Parâmetros e retorno

A primeira chamada da função deverá enviar a variável *controle.start* como *true.* Ao continuar enviando as digitais, a variável passa a ser *false.*

O registro *controle.modelo* deve conter o valor **9**.

As variáveis *controle.porta* (porta de comunicação de rede do REP. Ex.: 1001), *controle.s_tipo* (fixo em **2**), *controle.endereco* (endereço IP do REP. Ex.: 10.0.0.227) e *controle.baudrate* (fixo em **9600**) também deverão ser preenchidas, pois são campos obrigatórios.

A variável *controle.backup* dever conter 'N'.

As variáveis do registro *digitais* deverão ser preenchidas com as informações referentes ao funcionário. Para cada digital enviada, o valor da variável *controle.atual* deverá ser incrementada de 1, podendo ser enviada 10 digitais para cada funcionário.

Para cada funcionário com digital, a função é reiniciada.

O campo *digitais.pin* recebe o código do funcionário e o campo *digitais.dedo* recebe a primeira *template* do funcionário a ser enviada.

Para cadastrar a digital direto no PC, utilizamos o *hamster* da VIRDI.

2.5.3. Exemplo

```
var
  Controle: TControle;
  Digital: TDigitais;
begin
  Controle.modelo   := 9; // Colocar aqui o modelo do REP usado
  Controle.s_tipo   := 2;
  Controle.endereco := '10.0.0.225'; // IP do REP
  Controle.porta    := 1001;
  Controle.baudrate := 9600;
  Controle.erro     := 0;
  Controle.backup   := 'N';
  Controle.total    := 1; // quantidade de digitais a serem enviadas
  Controle.atual    := 1;
  Controle.start    := True;

  Digital.pin  := '91254'; // Código do funcionário no REP
  Digital.dedo := '';      // O primeiro envio deve ser vazio, pois somente sinaliza o início de uma comunicação

  if (enviaDigitaisTCP(Digital, Controle)) then
  begin
    Controle.start := False;

    while (Controle.atual <= Controle.total) do
    begin
      // A cada loop a Digital.dedo deve ser setada com a próxima digital do funcionário.
      Digital.dedo :=
        '554E494F4E055A00003CF40015519326917618 8E831A8539147D0F4C794049743014717B3D6501305A0B30581D3A541E1B4F'+
        '01284A931E4026683C3E683B40050D0E7E0C05480A0313080D5B078322015B7E0506F20627DC1665DD1A9FA22144DA2196DD'+
        '25A1852C97A34468DA4F46AB5112925923D55A00B36C39906C08946F60A900000000000000000000000000000000000000'+
        '000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000'+
        '000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000'+
        '000000000008080860302072468 6E535504022424685404 6F5703062E0D048004038003230503020304024700000000000'+
        '000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000'+
        '000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000'+
        '554E494F4E055A00003CE20014559320892D0F784D12776E3A773E476544145D044C5D3549553B1451853D4610313A173133'+
        'A01E2F0D28201B681C4B041933681201C0140E88170F931992A41C03DC1F4FFF2606B22814DC358AE03659DD4039DA6360DA'+
        '7052AB731D927B17D67B0CB3000000000000000000000000000000000000000000000000000000000000000000000000'+
        '000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000'+
        '000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000'+
        '000000000008005688005030208246A6F8083242724528470245606 2E2D230405800524800300000000000000000000000'+
        '000000000000000000000000000000000000000000000000000000000000000000000400400000334300023302020000 4004000'+
        '000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000';

      enviaDigitaisTCP(Digital, Controle);
      EscreveLog(Controle, Digital);
      Inc(Controle.atual); // Incrementa o Controle.atual até ser maior que Controle.total,
                           // que é a quantidade de digitais a serem enviadas
    end;
  end
  else
  begin
    ShowMessage('Não foi possível o envio de digitais!');
  end;

  // Necessário fechar a comunicação com o REP
  fecharComunicacao;
end;
```

2.5. Função “leDigitaisTCP”

Função utilizada para ler digitais dos funcionários do REP. O retorno se encontra nos parâmetros que são passados por referência.

2.5.1. Declaração

```
function leDigitaisTCP(var digitais: Tdigitais; var controle:
Tcontrole): boolean; stdcall; external 'authotelcom.dll';
```

2.5.2. Parâmetros e retorno

A primeira chamada da função deverá enviar a variável *controle.start* como *true* e *digitais.pin* deve conter o PIS do funcionário que se deseja ler a digital. Ao continuar lendo as digitais, a variável passa a ser *false.*

O registro *controle.modelo* deve conter o valor **9**.

As variáveis *controle.porta* (porta de comunicação de rede do REP. Ex.: 1001), *controle.s_tipo* (fixo em **2**), *controle.endereco* (endereço IP do REP. Ex.: 10.0.0.227) e *controle.baudrate* (fixo em **9600**) também deverão ser preenchidas, pois são campos obrigatórios.

A variável *controle.backup* dever conter **'N'**.

As variáveis do registro *digitais* deverão ser preenchidas com as informações referentes ao funcionário. Para cada digital enviada, o valor da variável *controle.atual* deverá ser incrementada de 1, podendo ser enviada 10 digitais para cada funcionário.

Para cada funcionários com digital, a função é reiniciada.

O campo *digitais.pin* recebe o PIS do funcionário e o campo *digitais.dedo* recebe a primeira *template* do funcionário lida.

2.5.3. Exemplo

```
var
  Controle: TControle;
  Digital: TDigitais;
begin
  Controle.modelo   := 9; // Colocar aqui o modelo do REP usado
  Controle.s_tipo   := 2;
  Controle.endereco := '10.0.0.225'; // IP do REP
  Controle.porta    := 1001;
  Controle.baudrate := 9600;
  Controle.erro     := 0;
  Controle.backup   := 'N';
  Controle.atual    := 1;
  Controle.start    := True;

  Digital.pin  := '011111111116'; // PIS do funcionário a ser lido (com zeros à esquerda)
  Digital.dedo := '';

  if (leDigitaisTCP(Digital, Controle)) then
  begin
    EscreveLog(Controle, Digital);
    Controle.start := False;

    while (leDigitaisTCP(Digital, Controle) and (Controle.total <> 0)) do
    begin
      if ((Controle.atual = Controle.total) and (Digital.dedo = '')) then
      begin
        { Finaliza a leitura)
        exit;
      end;
      { ************************************
        Tratar aqui as digitais recebidas...
      ************************************ }
    end;
  end
  else begin
    ShowMessage('Não foi possível a leitura de digitais!');
  end;

  // Necessário fechar a comunicação com o REP
  fecharComunicacao;
end;
```

2.6. Função “enviaEmpregadorTCP”

Função utilizada para enviar dados do empregador para o REP. O retorno se encontra nos parâmetros que são passados por referência.

2.6.1. Declaração

```
function enviaEmpregadorTCP(var dados: TDados; var controle:
Tcontrole): boolean; stdcall; external 'authotelcom.dll';
```

2.6.2. Parâmetros e retorno

O REP poderá receber apenas um empregador, portanto essa função será chamada apenas duas vezes: uma chamada, com o *controle.start = true*, indicando que será feito o envio do empregador e uma segunda chamada, com o *controle.start = false*, enviando os dados do empregador.

O registro *controle.modelo* deve conter o valor **9**.

As variáveis *controle.porta* (porta de comunicação de rede do REP. Ex.: 1001), *controle.s_tipo* (fixo em **2**), *controle.endereco* (endereço IP do REP. Ex.: 10.0.0.227) e *controle.baudrate* (fixo em **9600**) também deverão ser preenchidas, pois são campos obrigatórios.

As variáveis do registro *dados* deverão ser preenchidas com as informações referentes ao empregador.

A variável *dados.adcOUSubst* terá importância na primeira chamada da função. Essa variável indica se o empregador enviado a seguir será adicionado, substituído ou adicionado/substituído:

- **Adicionar**: para incluir o empregador no REP, o comando a ser enviado na variável *dados.adcOUSubst* será a letra **'A'**. Nesse caso, ele apenas inclui o empregador no REP. Caso já exista uma empresa cadastrada no REP, haverá um código de erro (3. Códigos dos erros) como retorno.
- **Substituir**: para substituir o empregador já cadastrado no REP, o comando a ser enviado na variável *dados.adcOUSubst* será a letra **'S'.** Nesse caso, ele apenas substitui os dados do empregador, cadastrado no REP, pelos dados recebidos. A empresa é localizada pelo CNPJ/CPF. Caso tente enviar dados onde o CNPJ/CPF não existe no REP, haverá um código de erro (3. Códigos dos erros) como retorno.
- **Adicionar/Substituir**: para adicionar/substituir o empregador no REP, o comando a ser enviado na variável *dados.adcOUSubst* poderá ser qualquer caractere diferente de **'A'** e **'S'**. Nesse caso, ele tenta incluir a empresa no REP, se o CNPJ/CPF já estiver cadastrado no REP, o sistema tenta substituir os dados do empregador. Se não for possível executar nenhuma das duas opções, haverá um código de erro (3. Códigos dos erros) como retorno.

Os campos a serem preenchidos no registro *dados* com as informações da empresa (todos os campos referentes às informações do empregador são obrigatórios, exceto o campo *dados.cei*) são:

- *dados.adcOUSubst* – indica qual o tipo de ação a ser executada (definida acima).
- *dados.pin* – usado no envio de funcionários.
- *dados.pis* – usado no envio de funcionários.
- *dados.identificador* – número do CNPJ ou do CPF.
- *dados.cei* – número do CEI. Caso não haja, enviar **'0'**.
- *dados.razaoSocial* – razão social da empresa (máximo de 150 caracteres).
- *dados.localPrestServ* – local da prestação de serviço (máximo de 100 caracteres).
- *dados.tipoId* – informa qual o tipo de identificador a ser enviado: 1 para CNPJ ou 2 para CPF.
- *dados.nome* – usado no envio de funcionários.
- *dados.id_bio* – usado no envio de funcionários.
- *dados.numCartao* – usado no envio de funcionários.
- *dados.senha* – usado no envio de funcionários.
- *dados.mestre* – usado no envio de funcionários.
- *dados.verifica* – usado no envio de funcionários.

Caso ocorra algum erro no meio da comunicação, a variável *controle.erro* receberá o valor que indica qual o erro ocorrido. Os valores podem ser visualizados em 3. Códigos dos erros.

Obs.: após o envio do empregador, o mesmo não poderá ser apagado do REP, podendo apenas ser substituído.

2.6.3. Exemplo

```
var
  Controle: TControle;
  Dados: TDados;
begin
  Controle.modelo   := 9; // Colocar aqui o modelo do REP usado
  Controle.s_tipo   := 2;
  Controle.endereco := '10.0.0.227'; // IP do REP
  Controle.porta    := 1001;
  Controle.baudrate := 9600;
  Controle.erro     := 0;
  Controle.backup   := 'N';
  Controle.atual    := 1;
  Controle.total    := 1;
  Controle.start    := True;

  dados.tipoId          := '1';
  dados.identificador   := '35503945000100';
  dados.cei             := '0';
  dados.razaoSocial     := 'EMPRESA LTDA - ME';
  dados.localPrestServ  := 'R UM, 10, CENTRO';
  dados.adcOUSubst      := 'W';

  if enviaEmpregadorTCP(dados, controle) then
  begin
    Controle.start := False;
    enviaEmpregadorTCP(dados, controle);

    if controle.erro <> 0 then
      ShowMessage('Erro ao enviar empregador!')
    else
      ShowMessage('Empregador enviado com sucesso!');

    // Necessário fechar a comunicação com o REP
    fecharComunicacao;
  end
  else
  begin
    ShowMessage('Não foi possível enviar empregador!');
  end;
end;
```

2.7. Função “leEmpregadorTCP”

Função utilizada para ler dados do empregador do REP. O retorno se encontra nos parâmetros que são passados por referência.

2.7.1. Declaração

```
function leEmpregadorTCP(var dados: TDados; var controle:
Tcontrole): boolean; stdcall; external 'authotelcom.dll';
```

2.7.2. Parâmetros e retorno

O registro *controle.modelo* deve conter o valor **9**.

As variáveis *controle.porta* (porta de comunicação de rede do REP. Ex.: 1001), *controle.s_tipo* (fixo em **2**), *controle.endereco* (endereço IP do REP. Ex.: 10.0.0.227) e *controle.baudrate* (fixo em **9600**) também deverão ser preenchidas, pois são campos obrigatórios.

Os dados recebidos do REP se encontrarão nas variáveis: *dados.identificador, dados.cei, dados.razaoSocial, dados.localPrestServ* e *dados.tipoId*.

Caso ocorra algum erro no meio da comunicação, a variável *controle.erro* receberá o valor que indica qual o erro ocorrido. Os valores podem ser visualizados em 3. Códigos dos erros.

2.7.3. Exemplo

```
var
  Controle: TControle;
  Dados: TDados;
begin
  Controle.modelo   := 9; // Colocar aqui o modelo do REP usado
  Controle.s_tipo   := 2;
  Controle.endereco := '10.0.0.227'; // IP do REP
  Controle.porta    := 1001;
  Controle.baudrate := 9600;
  Controle.erro     := 0;
  Controle.backup   := 'N';
  Controle.atual    := 1;
  Controle.start    := True;

  if (leEmpregadorTCP(dados, controle)) then
  begin
    controle.start := False;
    leEmpregadorTCP(dados, controle);

    if controle.erro <> 0 then
      ShowMessage('Erro ao ler empregador!')
    else
      EscreveLog(dados);

    // Necessário fechar a comunicação com o REP
    fecharComunicacao;
  end
  else
  begin
    ShowMessage('Não foi possível o ler empregador!');
  end;
end;
```

2.8. Função “leTrabalhadorTCP”

Função utilizada para ler dados dos funcionários cadastrados no REP. O retorno se encontra nos parâmetros que são passados por referência.

2.8.1. Declaração

```
function leTrabalhadorTCP(var dados: TDados; var controle:
Tcontrole): boolean; stdcall; external 'authotelcom.dll';
```

2.8.2. Parâmetros e retorno

O registro *controle.modelo* deve conter o valor **9**.

As variáveis *controle.porta* (porta de comunicação de rede do REP. Ex.: 1001), *controle.s_tipo* (fixo em **2**), *controle.endereco* (endereço IP do REP. Ex.: 10.0.0.227) e *controle.baudrate* (fixo em **9600**) também deverão ser preenchidas, pois são campos obrigatórios.

Os dados recebidos do REP se encontrarão nas variáveis: *dados.pin, dados.pis, dados.nome, dados.id_bio*, *dados.numCartao*, *dados.senha* e *dados.mestre*.

O campo *dados.senha* retornará o valor '000000' quando não houver senha cadastrada.

A senha é numérica, porém, caso haja zeros à esquerda, esses zeros à esquerda serão substituídos pela letra 'A'. A senha possui no máximo 6 dígitos. Exemplo: a senha enviada para o REP ou digitada no mesmo é “00123”. Nesse caso, se for feita a leitura do trabalhador, o retorno no campo senha será: 0AA123, os zeros a esquerda devem ser desconsiderados, pois são apenas complementos para o campo de 6 bytes. Se digitado “123”, o retorno será '000123'; se digitado “1020”, o retorno será '001020'; se digitado “012305”, o retorno A12305.

Os campos *controle.atual* e *controle.total* serão atualizados pela *DLL*. Para cada informação recebida, a função deve ser chamada novamente para que seja feita a confirmação dos dados recebidos. Caso haja mais dados, em cada confirmação os campos do registro *dados* estarão com informações do próximo funcionário, até que *controle.atual* seja igual a *controle.total*, encerrando, assim, o ciclo com uma confirmação.

O campo *dados.pis* pode ser preenchido e enviado para o REP. Nesse caso, o REP envia os dados somente do funcionário com o PIS enviado. Caso contrário, o *dados.pis* deve ser igual a 0 (zero); assim todos os funcionários cadastrados no REP serão lidos.

Caso ocorra algum erro no meio da comunicação, a variável *controle.erro* receberá o valor que indica qual o erro ocorrido. Os valores podem ser visualizados em 3. Códigos dos erros.

2.8.3. Exemplo

```
var
  Controle: TControle;
  Dados: TDados;
begin
  Controle.modelo   := 9; // Colocar aqui o modelo do REP usado
  Controle.s_tipo   := 2;
  Controle.endereco := '10.0.0.227'; // IP do REP
  Controle.porta    := 1001;
  Controle.baudrate := 9600;
  Controle.erro     := 0;
  Controle.backup   := 'N';
  Controle.start    := True;

  Dados.pis := '0';

  if (leTrabalhadorTCP(dados, controle)) then
  begin
    // Tratamento quando o REP não possui funcionários cadastrados
    if ((controle.erro = 0) and (controle.total = 0) and (controle.atual = 0)) then
    begin
      // Necessário fechar a comunicação com o REP
      fecharComunicacao;
      ShowMessage('Não existe funcionários no REP');
      Exit;
    end;

    EscreveLog(Dados);
    controle.start := False;

    while (leTrabalhadorTCP(dados, controle) and (controle.total <> 0)) do
    begin
      if controle.erro <> 0 then
        ShowMessage('Erro ao ler funcionário!')
      else
        EscreveLog(dados);
    end;

    // Necessário fechar a comunicação com o REP
    fecharComunicacao;
    ShowMessage('Todos os funcionários lidos com sucesso!');
  end
  else
  begin
    ShowMessage('Não foi possível o ler funcionários!');
  end;
end;
```

2.9. Função “fecharComunicação”

Procedure utilizada para interromper/finalizar a comunicação com o REP.

2.9.1. Declaração

```
procedure fecharComunicacao; stdcall; external 'authotelcom.dll';
```

3. Códigos dos erros

CÓDIGO	DESCRIÇÃO
01	O BCC recebido não confere com BCC calculado.
02	A hora desejada (enviada pelo PC) não constitui uma hora válida.
03	Parâmetro e/ou Tamanho e/ou Flag/Error não suportados.
04	O Comando enviado não é suportado ou é desconhecido.
05	Erro não especificado.
08	Não foi encontrado um funcionário com o PIS solicitado.
10	Identificador (CPF/CNPJ/PIS) inconsistente.
11	Identificador (CPF/CNPJ/PIS) recusado.
12	Código recusado.
13	Espaço insuficiente.
96	O *frame* recebido contém erro.