Registrador Eletrônico de Ponto (REP)

Famílias iPointLine e PointLine



**Manual de uso das *DLLs***

Índice

[1. Introdução 3](#__RefHeading__74_968010720)

[2. Funções e Procedimentos 3](#__RefHeading__4120_467821017)

[2.1. Tipos de dados a serem passados para as funções 3](#__RefHeading__4122_467821017)

[2.2. Parâmetros “cpf” e “hash” 5](#__RefHeading__9176_1128431964)

[2.3. Função “configura” 5](#__RefHeading__4124_467821017)

[2.4. Função “recebeMarcacoesTCP” 11](#__RefHeading__4126_467821017)

[2.5. Função “enviaTrabalhadorTCP” 13](#__RefHeading__4128_467821017)

[2.6. Função “enviaDigitaisTCP” 15](#__RefHeading__4130_467821017)

[2.7. Função “leDigitaisTCP” 17](#__RefHeading__4132_467821017)

[2.8. Função “enviaEmpregadorTCP” 18](#__RefHeading__4140_467821017)

[2.9. Função “leEmpregadorTCP” 21](#__RefHeading__4148_467821017)

[2.10. Função “leTrabalhadorTCP” 22](#__RefHeading__4156_467821017)

[2.11. Função “leChaveTcp” 24](#__RefHeading__3391_230252977)

[2.12. Função “fecharComunicação” 26](#__RefHeading__4164_467821017)

[2.13. Função “AbrirComunicacaoComRep” (DLL “rwtrep”) 26](#__RefHeading__9178_1128431964)

[2.14. Função “FecharComunicacaoComRep” (DLL “rwtrep”) 26](#__RefHeading__5973_640939653)

[2.15. Função “ComunicacaoAbertaComRep” (DLL “rwtrep”) 26](#__RefHeading__5975_640939653)

[2.16. Função “EnviarDigital” (DLL “rwtrep”) 27](#__RefHeading__5977_640939653)

[3. Códigos dos erros 28](#__RefHeading__3904_467821017)

# 1. Introdução

Este é o manual de referência do funcionamento das DLLs “authotelcom.dll” e “rwtrep”, as quais realizam 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 pelas *DLLs* para a comunicação com o REP:

DLL “authotelcom.dll”:

* configura.
* recebeMarcacoesTCP.
* enviaTrabalhadorTCP.
* enviaEmpregadorTCP.
* enviaDigitaisTCP (**utilizada apenas para a linha PointLine**).
* leEmpregadorTCP.
* leTrabalhadorTCP.
* LeDigitaisTCP.
* leChaveTcp
* FecharComunicacao.

DLL “rwtrep.dll” (**utilizada apenas para a linha iPointLine**):

* AbrirComunicacaoComRep.
* FecharComunicaoComRep.
* ComunicacaoAbertaComRep.
* EnviarDigital.

## 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 da DLL authotelcom.dll, exceto para a função “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;

cpf: ShortString;

cmp: ShortString;

chave: ShortString;

end;

TDigitais = record

pin: ShortString;

dedo: AnsiString;

end;

## 2.2. Parâmetros “cpf” e “hash”

Sempre quando os campos “CPF” e “HASH” forem citados, deve-se ter as seguintes considerações:

**Linha PointLine:**

* cpf: deve-se passar o valor “01”
* hash: deve-se passar o valor “0”

**Linha iPointLine:**

* cpf: deve-se passar o CPF do responsável pela operação, sempre com 12 caracteres, completando com zero (0) à esquerda caso o CPF tenha menos de 12 caracteres
* hash: deve-se passar o hash do REP (64 caracteres)

## 2.3. 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.
* Ler papel enroscado.

### 2.3.1. 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; hash: String): pchar; stdcall; external 'authotelcom.dll';

### 2.3.2. 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 |
| LN | Leitura de papel enroscado |

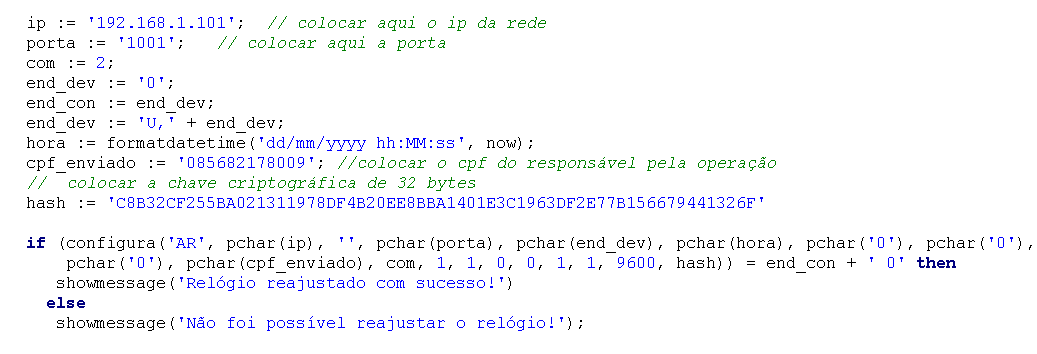
* ***end*\_*dev*** (*obrigatório*) – deve permanecer em '**U,00'**.
* ***ip*** (obrigatório) – indica o IP do REP.
* ***porta*** (obrigatório) – indica a porta de comunicação utilizada. Ex.: 1001.
* ***host*** (opcional) – indica o endereço do computador que iniciou a comunicação.
* ***datahorai*** (obrigatório 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*** – Para AR, LR, AH, LH, LP, LO e LN deve permanecer vazio. Para o GO e GP preencher com CPF do responsável pela operação.
* ***info*** (obrigatório) – Para AR, LR, AH, LH, LP, LO e LN preencher com CPF do responsável pela operação. Para o GO e GP preencher com 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.**
* **hash** – (*obrigatório*) chave criptográfica.

**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 (Erro: Origem da referência não encontrada). Obs.: o zero (0) indica que a função foi executada com sucesso.

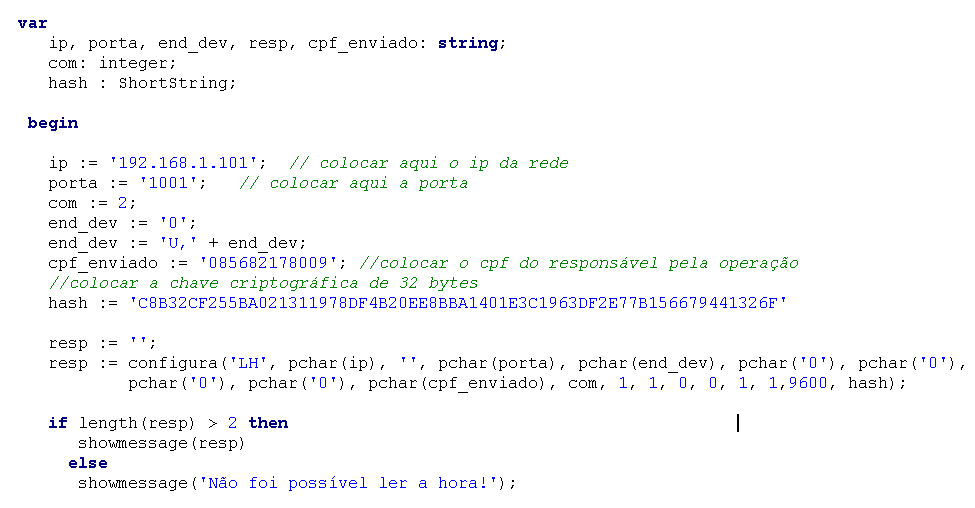
Exemplo:



### 2.3.4. 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 (Erro: Origem da referência não encontrada).

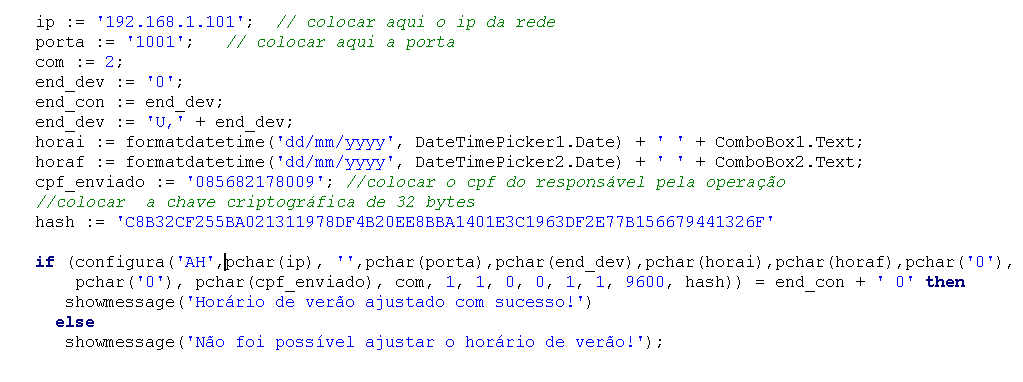
Exemplo:



### 2.3.5. 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 (Erro: Origem da referência não encontrada).

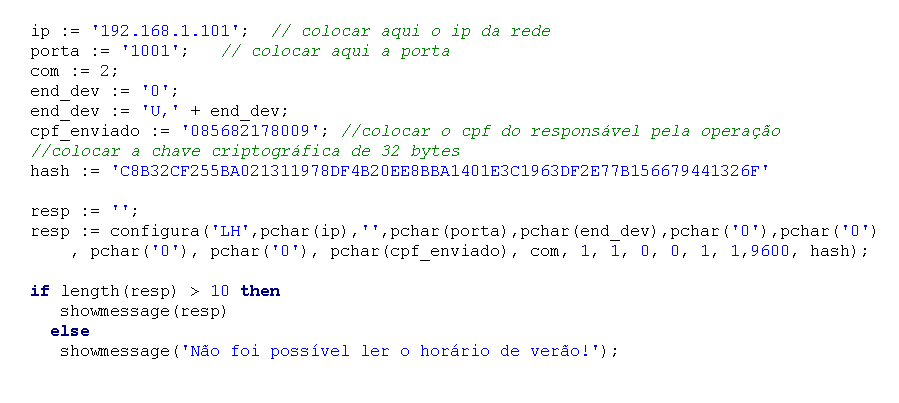
Exemplo:



### 2.3.6. 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 (Erro: Origem da referência não encontrada). 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'.

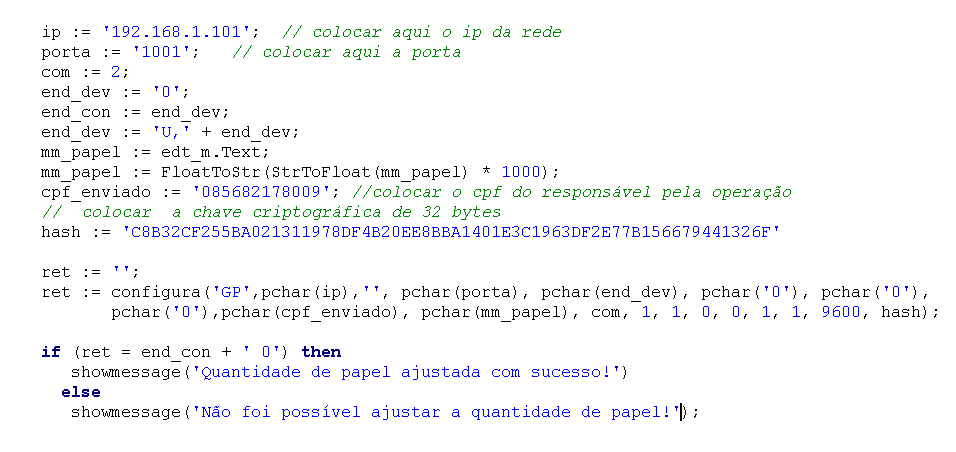
Exemplo:



### 2.3.7. 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 (Erro: Origem da referência não encontrada).

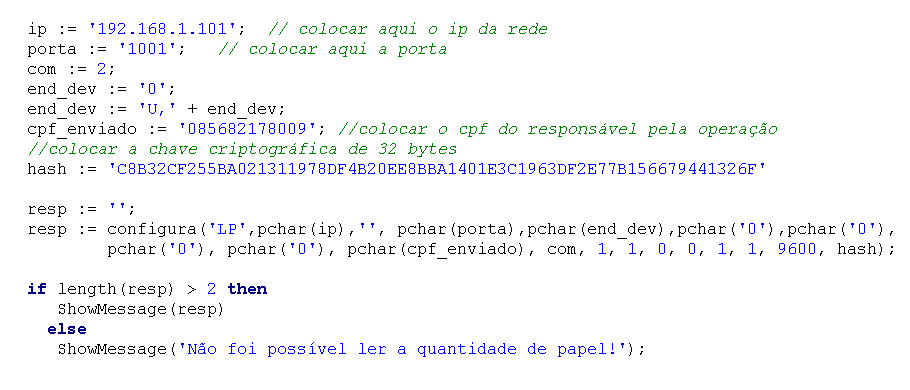
Exemplo:



### 2.3.8. Ler quantidade de papel

O retorno da função é o endereço do REP + tamanho (em milímetros) ou código de erro (Erro: Origem da referência não encontrada).

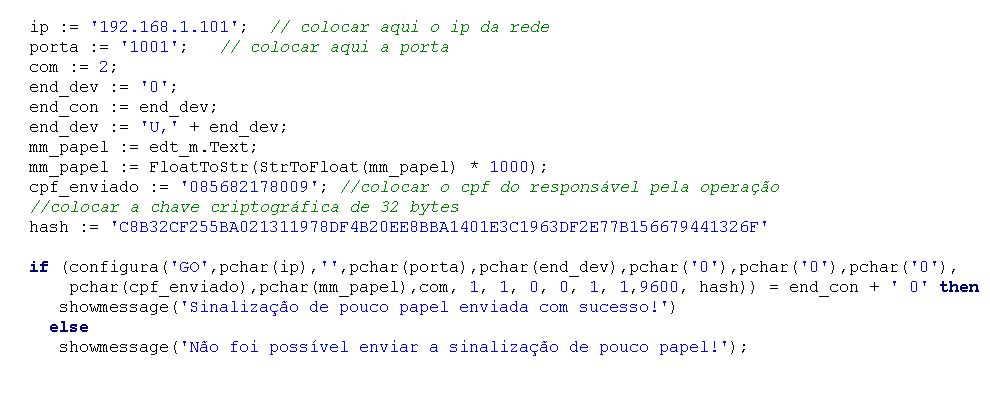
Exemplo:



### 2.3.9. 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 (Erro: Origem da referência não encontrada).

Exemplo:



### 2.3.10. Ler sinalização de pouco papel

O retorno da função é o endereço do REP + tamanho (em milímetros) ou código de erro (Erro: Origem da referência não encontrada).

Exemplo:

### 

### 2.3.11. Ler papel enroscado

O retorno da função é o endereço do REP + a informação do papel enroscado, ou a informação do papel não enroscado ou ou código de erro (Erro: Origem da referência não encontrada).

Exemplo:

## 

## 2.4. 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.

### 2.4.1. Declaração

function recebeMarcacoesTCP(var marcacao: array of TMarcacao; var controle: Tcontrole; evento: integer): 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 recebendo as marcações, a variável passa a ser *false.*

O registro *controle.modelo* deve conter o valor **11**.

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), *controle.baudrate* (fixo em **9600**) e controle.hash (chave criptográfica) 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 (Erro: Origem da referência não encontrada) ocorrido.

### 2.4.3. 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.

* + 1. **Exemplo**

**var**

controle: TControle;

marcacao: **array** [1..50] **of** TMarcacao;

recebe: boolean;

hash : PChar;

**begin**

*// Inicia as variáveis*

controle.modelo := 11; *// Colocar aqui o modelo do REP*

controle.s\_tipo := 2;

controle.endereco := '10.0.0.227'; // Colocar aqui o ip do REP

controle.porta := 1001; *// Colocar aqui a porta do REP*

controle.baudrate := 9600;

controle.start := true;

*// No campo 'hash' deve-se colocar a chave criptográfica do REP*

hash:='D6C0AA3A4AE114BBE7B6E6E4278CF20F9D5ABB197A09CD9CBD510ED826D03CA4';

recebe:=recebeMarcacoesTCP(marcacao, controle, 0, hash); *// 0 = todas*

controle.start := false;

**if** (recebe) **then**

**begin**

**while**((controle.total>controle.atual)

**and**

(recebeMarcacoesTCP(marcacao, controle, 0, hash))) **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*

fecharComunicacao();

**end**;

## 2.5. Função “enviaTrabalhadorTCP”

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

### 2.5.1. Declaração

function enviaTrabalhadorTCP(var dados: TDados; 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 os funcionários, a variável passa a ser *false.*

O registro *controle.modelo* deve conter o valor **11**.

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), *controle.baudrate* (fixo em **9600**) e controle.hash (chave criptográfica) também deverão ser preenchidas, pois são campos obrigatórios.

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

A variável dados.cpf do registro dados deverá ser preenchida com o CPF do responsável pela operação.

As demais 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 (Erro: Origem da referência não encontrada) 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 (Erro: Origem da referência não encontrada) como retorno.

**Observação**: sempre quando for adicionar um funcionário, antes é necessário fazer a exclusão do mesmo.

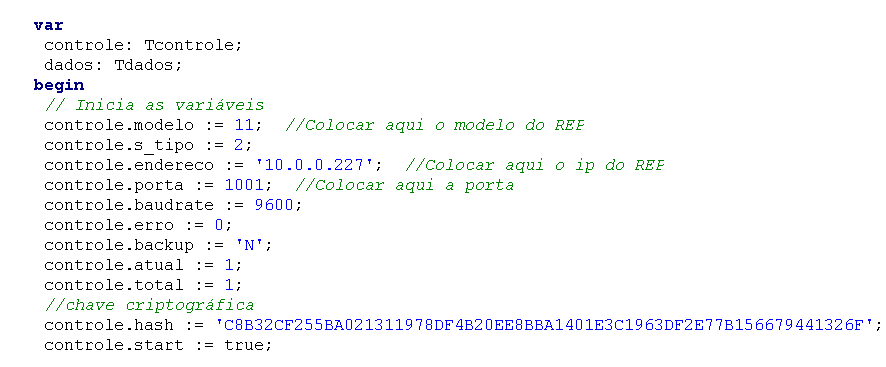
Os campos a serem preenchidos no registro *dados* com as informações dos funcionários (todos os campos referentes às informações dos funcionários são obrigatórios, exceto o campo *dados.id\_bio*):

* *dados.adcOUSubst* – indica qual o tipo de ação a ser executada (definida acima).
* *dados.pin* – número de identificação do funcionário (código).
* *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.
* dados.cpf – deverá ser preenchida com o CPF do responsável pela operação.

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 Erro: Origem da referência não encontrada.

### 2.5.3. Exemplo



## 2.6. Função “enviaDigitaisTCP”

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

### 2.6.1. Declaração

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

### 2.6.2. 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 **11**.

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**) e controle.hash (chave criptográfica) também deverão ser preenchidas, pois são campos obrigatórios.

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

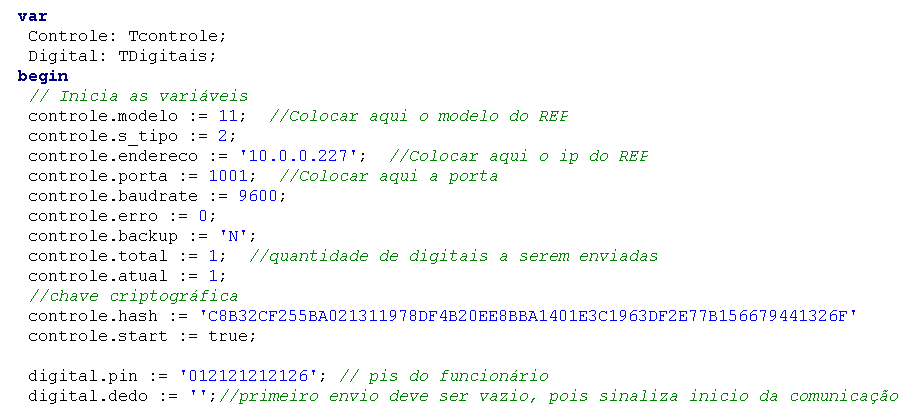
As variáveis do registro *digitais* deverão ser preenchidas com as informações referentes ao funcionário. Para cada digital enviada, o valor da variável *controle.atual* deverá ser incrementada de 1, podendo ser enviada 10 digitais para cada funcionário.

Para cada funcionário com digital, a função é reiniciada.

O campo *digitais.pin* recebe o pis 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



## 2.7. Função “leDigitaisTCP”

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

### 2.7.1. Declaração

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

### 2.7.2. 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 **11**.

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), *controle.baudrate* (fixo em **9600**) e controle.hash (chave criptográfica) também deverão ser preenchidas, pois são campos obrigatórios.

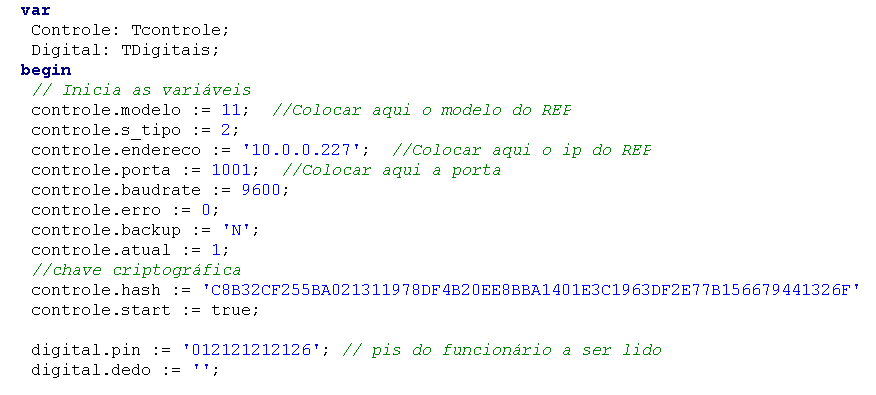
A variável *controle.backup* dever conter **'N'**.

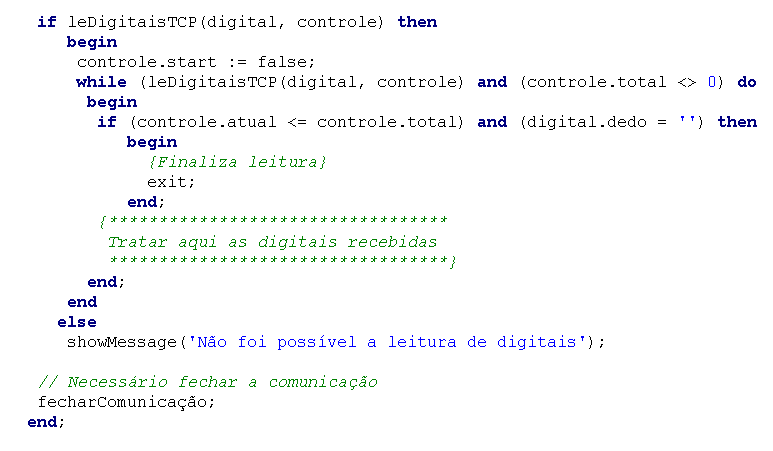
As variáveis do registro *digitais* deverão ser preenchidas com as informações referentes ao funcionário. Para cada digital enviada, o valor da variável *controle.atual* deverá ser incrementada de 1, podendo ser enviada 10 digitais para cada funcionário.

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.

### 2.7.3. Exemplo



****

## 2.8. Função “enviaEmpregadorTCP”

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

### 2.8.1. Declaração

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

### 2.8.2. 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 **11**.

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), *controle.baudrate* (fixo em **9600**) e controle.hash (chave criptográfica) 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:

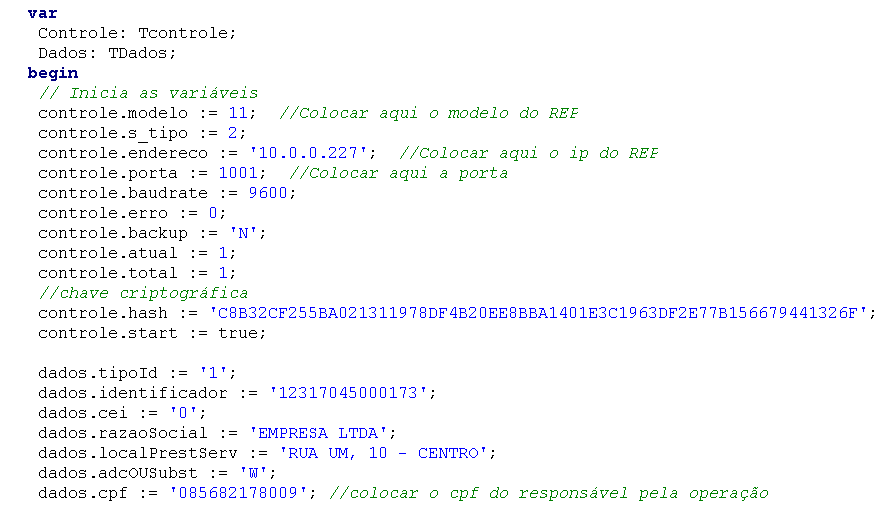
* **Adicionar**: para incluir o empregador no REP, o comando a ser enviado na variável *dados.adcOUSubst* será a letra **'A'**. Nesse caso, ele apenas inclui o empregador no REP. Caso já exista uma empresa cadastrada no REP, haverá um código de erro (Erro: Origem da referência não encontrada) como retorno.
* **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 (Erro: Origem da referência não encontrada) 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 (Erro: Origem da referência não encontrada) como retorno.

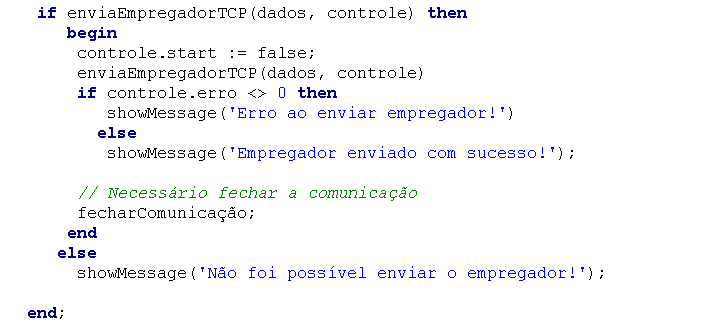
Os campos a serem preenchidos no registro *dados* com as informações da empresa (todos os campos referentes às informações do empregador são obrigatórios, exceto o campo *dados.cei*) são:

* *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.
* dados.cpf – deverá ser preenchida com o CPF do responsável pela operação.

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 Erro: Origem da referência não encontrada.

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

**2.8.3. Exemplo**

****

## 2.9. Função “leEmpregadorTCP”

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

### 2.9.1. Declaração

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

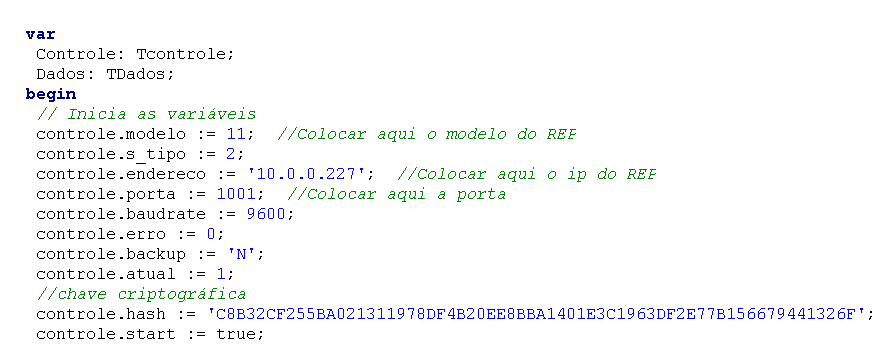
### 2.9.2. Parâmetros e retorno

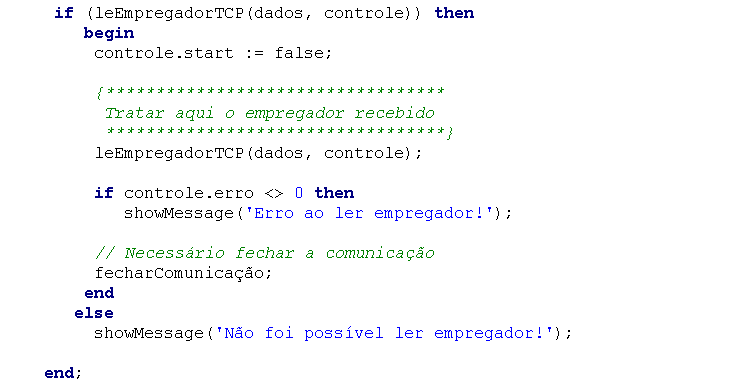
O registro *controle.modelo* deve conter o valor **11**.

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), *controle.baudrate* (fixo em **9600**) e controle.hash (chave criptográfica) 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 Erro: Origem da referência não encontrada.

****

**2.9.3. Exemplo**

## 2.10. Função “leTrabalhadorTCP”

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

### 2.10.1. Declaração

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

### 2.10.2. Parâmetros e retorno

O registro *controle.modelo* deve conter o valor **11**.

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), *controle.baudrate* (fixo em **9600**) e controle.hash (chave criptográfica) 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.

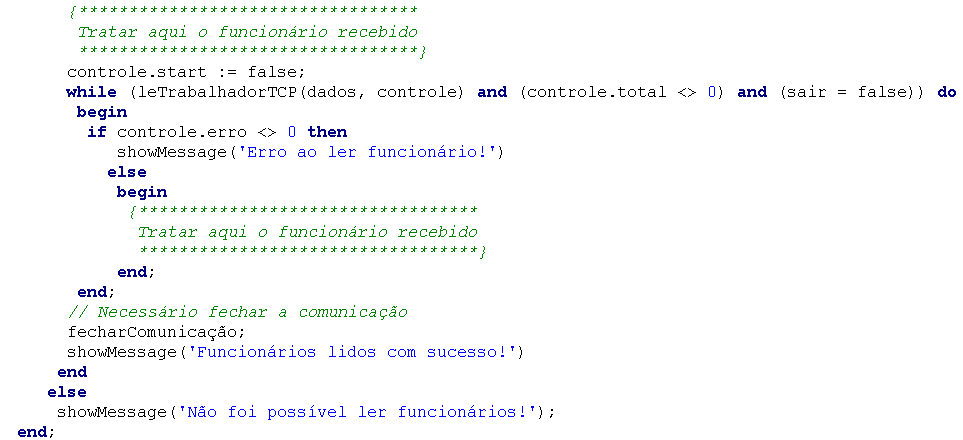
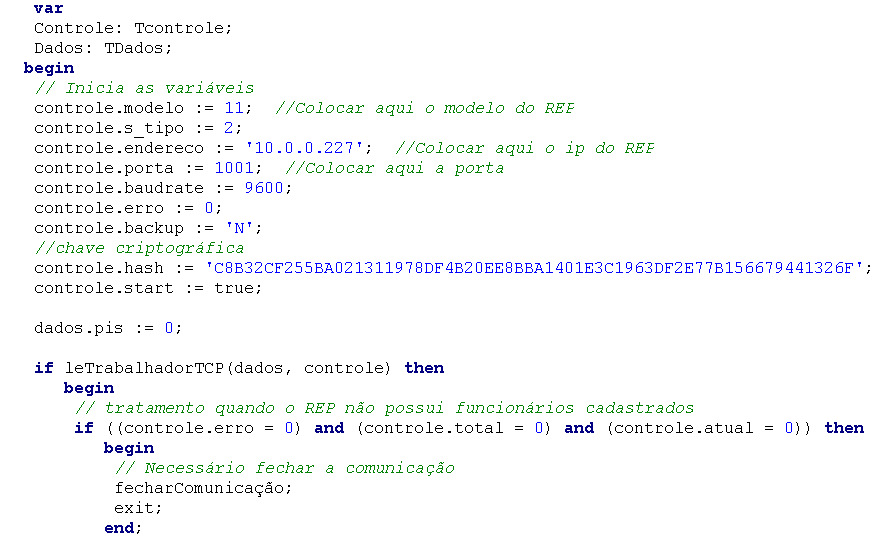
A senha é numérica, porém, caso haja zeros à esquerda, esses zeros à esquerda serão substituídos pela letra 'A'. A senha possui no máximo 6 dígitos. Exemplo: a senha enviada para o REP ou digitada no mesmo é “00123”. Nesse caso, se for feita a leitura do trabalhador, o retorno no campo senha será: 0AA123, 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 Erro: Origem da referência não encontrada.

### 2.10.3. Exemplo

****

## 2.11. Função “leChaveTcp”

*Procedure* utilizada para ler a chave criptográfica que será gerada pelo REP, portanto deverá ser utilizada somente na primeira comunicação.

### 2.11.1. Declaração

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

### 2.11.2. Parâmetros e retorno

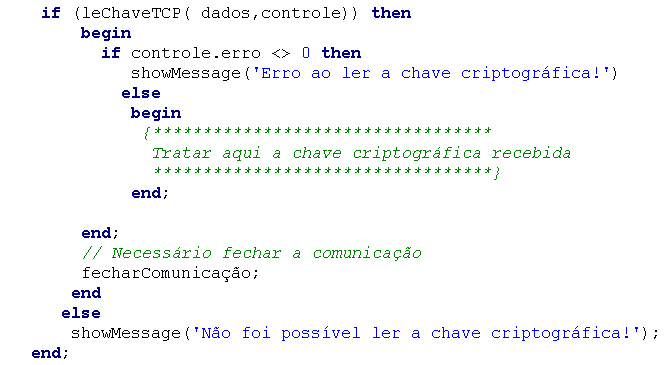
O registro *controle.modelo* deve conter o valor **11**.

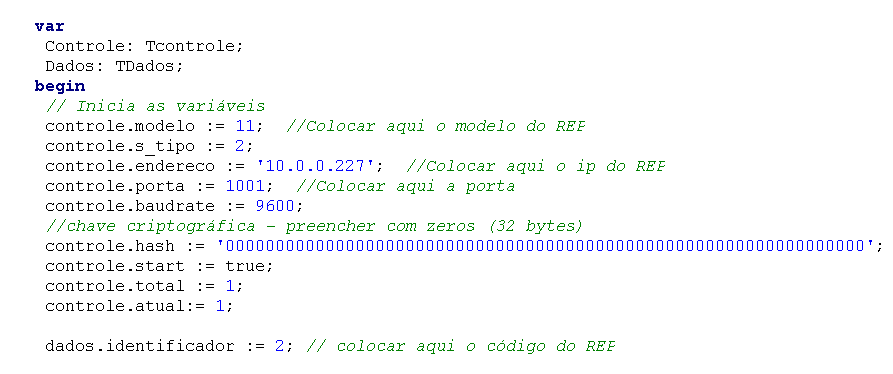
As variáveis *controle.porta* (porta de comunicação de rede do REP. Ex.: 1001), *controle.s\_tipo* (fixo em **2**), *controle.endereco* (endereço IP do REP. Ex.: 10.0.0.227), *controle.atual* (fixo em **1**), *controle.total* (fixo em **1**), *controle.baudrate* (fixo em **9600**) e controle.hash (preencher com zeros – 32 bytes: '00000000000000000000000000000000' ) também deverão ser preenchidas, pois são campos obrigatórios. A variável dados.identificador deve ser preenchida com o código do REP.

A chave criptográfica gerada pelo REP retornará na variável controle.hash.

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 Erro: Origem da referência não encontrada.

**2.11.3. Exemplo**

****

****

## 2.12. Função “fecharComunicação”

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

### 2.12.1. Declaração

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

## 2.13. Função “AbrirComunicacaoComRep” (DLL “rwtrep”)

Função utilizada para iniciar a comunicação com o REP.

### 2.13.1. Declaração

fucntion AbrirComunicacaoComRep(ip: Pchar; porta: integer; timeoutMs: integer; hash: Pchar): integer; stdcall; external 'rwtrep.dll';

### 2.13.2. Parâmetros

* **ip** – (*obrigatório*) indica o IP do REP.
* **porta** – (*obrigatório*) – indica a porta de comunicação utilizada. Ex.: 1001.
* **timeoutMs** – (*obrigatório*) *timeout*, em milissegundos**.**
* **hash** – (*obrigatório – 64 caracteres*) chave criptográfica**.**

## 2.14. Função “FecharComunicacaoComRep” (DLL “rwtrep”)

Função utilizada para interromper/fechar a comunicação com o REP.

### 2.14.1. Declaração

function FecharComunicacaoComRep(): integer; stdcall; external 'rwtrep.dll';

## 2.15. Função “ComunicacaoAbertaComRep” (DLL “rwtrep”)

Função utilizada para verificar se a comunicação com o REP está aberta/ativa.

### 2.15.1. Declaração

function ComunicacaoAbertaComRep(): integer; stdcall; external 'rwtrep.dll';

## 2.16. Função “EnviarDigital” (DLL “rwtrep”)

Função utilizada para enviar digitais dos funcionários para o REP.

### 2.16.1. Declaração

function EnviarDigital(pis: Pchar; atual: byte; total: byte; digital: Pchar): integer; stdcall; external 'rwtrep.dll';

### 2.16.2. Parâmetros

* **pis** – (*obrigatório – 12 caracteres*) indica o PIS do funcionário.
* **atual** – (*obrigatório*) – indica a sequência da digital a ser enviada. *Exemplo*: Existem 3 digitais do mesmo funcionário; a 1ª digital deve ser enviada com o valor 1 no parâmetro “atual”, e a 2ª com valor 2, e assim sucessivamente.
* **total** – (*obrigatório*) total de digitais do mesmo funcionário a serem enviadas. No exemplo acima, o total seria 3.
* **digital** – (*obrigatório*) digital do funcionário. A captura da digital é feita somente via REP. Por isso, após realizar o cadastro das digitais no REP, recomenda-se realizar a leitura de digitais, para que as mesmas fiquem gravadas na aplicação cliente também e, posteriormente, seja possível enviá-las da aplicação cliente para o REP.

### 2.16.3. Exemplo

var

pis: pChar;

atual: byte;

total: byte;

digital: pChar; // 1600 caracteres (800 pra cada *template*; 2 *templates* cada digital)

begin

if (AbrirComunicacaoComRep('10.0.0.100', 1001, 10000, 'C8B32CF255BA021311978DF4B20EE8BBA1401E3C1963DF2E77B1579441326F10') = COD\_ERRO\_OK) then

begin

pis := '010101010106';

atual := 1;

total := 1; //quantidade de digitais a serem enviadas

digital :=

'';

while (atual <= total) do

begin

EnviarDigital(pis, atual, total, digital);

atual := atual + 1;

end;

end

else

showMessage('Não foi possível abrir comunicação com o REP!');

FecharComunicacaoComRep();

end;

# 3. Códigos dos erros

|  |  |
| --- | --- |
| **CÓDIGO** | **DESCRIÇÃO** |
| 01 | A verificação de integridade identificou inconsistência dos dados. |
| 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. |
| 06 | Algum erro ocorreu durante a execução do comando. |
| 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. |
| 300 | Timeout atingido. |
| 301 | Inconsistência nos dados. |
| 302 | Comunicação inativa/fechada. |
| 303 | Erro na resposta do REP. |
| 304 | Erro na descriptografia. |
| 305 | Erro na montagem dos dados. |
| 306 | Erro na conexão. |
| 307 | Erro no envio. |
| 308 | Erro no formato dos dados. |
| 320 | Erro inesperado. |