ZPM Indústria e Comércio Ltda.

Manual DLL

Registrador Eletrônico de Ponto

RepZPM
Família “R”

www.zpm.com.br

Sumário

Revisões

1.0 –12/09/2014

- *Revisão Inicial.*

1.1 –29/10/2014

- *Alterados os itens 2, 4 e 7, inclusão das funções “Modo Mifare”.*

Revisões da DLL

1.0.0 –08/02/2011

- *Versão Inicial*

1.0.1 – 24/03/2011

- *Corrigido retorno para Data Inválida na Função DLLREP_BuscaPonto(). Ao invés de -2, retorna código de erro -21*

1.1.0 – 31/03/2011

- *Habilitado: múltiplas operações em DLLREP_Funcionario_Prepara() também para exclusão e alteração*
- *Adicionados métodos DLLREP_AjustaHorarioVerao (), DLLREP_BuscaHorarioVerao ()*

1.2.0 – 06/04/2011

- *Inclusão na documentação das funções de leitura AFD DLLREP_BuscaLeituraMRP () e DLLREP_RetornoLeituraMRP()*
- *Incluído comando para Exclusão de AFD e Exclusão de Ponto (somente disponíveis para Controles de Acesso)*
- *Incluído o comando para Busca Seletiva de Funcionários Cadastrados DLLREP_BuscaFuncionario_Prepara()*
- *Correção de exceção gerada após tentar realizar conexão com outro REP depois de Conexão.*
- *Correção de funções de retorno. Ao buscar retorno em linhas com erro, ocorria eventualmente violação de memória. Passa a ser retornado código -6 (Buscar mais detalhe através das Funções de Tratamento de Erro).*

1.2.1 – 14/4/2011

- *Função DefineModoIP passa a aceitar endereço do Host como alternativa ao Endereço IP*

1.2.2 – 29/6/2011

- *Parametro TemplateBiometrico nas funções DLLREP_FuncionarioPrepara() e DLLREP_RetornoFuncionario() passa a permitir uma lista de até 10 templates separados por ponto-e-vírgula;*

1.2.3 – 15/7/2011

- *Inclusão das funções DLLREP_ConfiguraCodigoBarra() e DLLREP_LeCodigoBarra()*

1.2.5 – 17/10/2011

- *Parâmetro "NumeFunc" da função DLLREP_FuncionarioPrepara() passa a ser obrigatório quando a Operação for igual a 2 (Alteração)*
- *Correção da passagem de parâmetro com valores nulos que geravam exceções em várias funções*

1.2.6 – 28/11/2011

- *Alteração do layout do manual*
- *Inclusão no manual das declarações das funções em Delphi, C# e VB.NET.*

1.2.7 – 30/05/2012

- *Inclusão das funções DLLREP_IniciaLog() e DLLREP_EncerraLog()*

1.3.0 – 28/08/2012

- *Inclusão das funções DLLREP_BuscaStatusCompleto(), DLLREP_RetornoStatusCompleto(), DLLREP_BuscaLeituraMRP_NSR(), DLLREP_ReinicializaSenhas() e DLLREP_RetornoStatusSenhas().*
- *Exclusão das funções DLLREP_ExcluiPonto() e DLLREP_ExcluiAFD() utilizadas exclusivamente no controle de acesso, modelo CA200.*

1.3.1 – 14/02/2013

- *Inclusão das funções DLLREP_ConfiguraMifareInvertehexa() e DLLREP_LeMifareInverteHexa().*

2.0.0 – 13/08/2013

- *Inclusão das funções DLLREP_Localiza() e DLLREP_RetornoLocaliza() para localização de REPs (somente R300)*
- *Inclusão da função de criptografia DLLREP_DefineModoIPCriptografado(). A função DLLREP_LeModo passa a retornar valor específico para o modo IPCriptografado. (somente R300).*
- *Suporte a operação do REP em modo cliente através da função DLLREP_IniciaLeituraEventos(), DLLREP_AguardaRetornoREP e DLLREP_EncerraLeituraEventos (somente R300).*
- *Suporte a Eventos através da função DLLREP_BuscaEvento() e DLLREP_RetornoUltimoEvento() (somente R300).*
- *Extensão das funções Funcionario_Prepara() e DLLREP_Empregador(), permitindo a Operação 4 (Inclusão ou Alteração) (somente R300).*

2.0.4 – 25/06/2014

- *Inclusão das funções DLLREP_ConfigMascaraBarras(), DLLREP_ConfigModoTAG(), DLLREP_ConfigCriptoBarras(), DLLREP_BuscaMascaraBarras(), DLLREP_BuscaUltimosRegistros(), DLLREP_BuscaModoTAG(), DLLREP_BuscaCriptoBarras(), DLLREP_RetornoMascaraBarras(), DLLREP_RetornoUltimosRegistros(), DLLREP_RetornoModoTAG() e DLLREP_RetornoCriptoBarras(). (Somente R300).*

2.0.5 – 28/10/2014

- *Inclusão das funções:*
  - DLLREP_ConfigModoMifare()
  - DLLREP_BuscaModoMifare()
  - DLLREP_RetornoModoMifare()

Introdução

A DLLREP.DLL é uma biblioteca de vínculo dinâmico que encapsula o protocolo de comunicação do REP-ZPM. Funciona em ambiente Windows® com as principais linguagens de programação, como Visual Basic, VB.NET, C#, C++, Delphi, etc. Ela proporciona um acesso simples para as funções de configuração, cadastramento de funcionários e empregador e de busca de marcações.

Destina-se ao desenvolvimento de aplicações que necessitam estabelecer um canal de comunicação entre microcomputadores com ambiente Windows® e o REP-ZPM. A DLL oferece os recursos abaixo:

- Uso simples nas principais linguagens e ambientes;
- Comunicação com múltiplos REPs;
- Envio de múltiplos comandos relativos a funcionários em uma única operação;
- Possibilidade de comunicação com o REP por troca de arquivos (via pendrive) ou pela rede.

Modos de Operação

O REP-ZPM suporta a comunicação através de sockets ou pendrive (quando não há rede disponível na localização do REP). As funções *DLLREP_DefineModoIP()* e *DLLREP_DefineModoArquivo()* definem, respectivamente, o modo de comunicação do REP.

No Modo IP, as chamadas são síncronas e a resposta on-line. Para a comunicação simultânea com vários REPs, sugere-se a utilização de Threads.

No Modo Arquivo, as chamadas são naturalmente assíncronas e off-line. A chamada de execução de comandos, neste caso, provoca a gravação de arquivos de comando na unidade de pendrive. É possível gerar, nesses casos, múltiplos comandos destinados a múltiplos REPs. No entanto, a ordem de execução não será garantida. Após o processamento dos arquivos neste pendrive, pode-se recuperar o resultado através da função *DLLREP_VerificaRetornoPenDrive()*.

1. Funções de Inicialização

➢ DLLREP_IniciaDriver()

Inicializa uma instância de comunicação com determinado REP. O Número de Fabricação do REP deve ser informado como parâmetro.

Declaração:

int DLLREP_IniciaDriver(string(17) **numFabricacao**)

Parâmetros:

numFabricacao – Número de Fabricação do REP

Retorno:

Handle
-1 Erro Inicialização

- Exemplo em VB.NET:

```
Declare Function DLLREP_IniciaDriver Lib "dllrep.dll" (ByVal numFabricacao As String) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_IniciaDriver(System.String NumFabricacao);
```

- Exemplo em Delphi:

```
function DLLREP_IniciaDriver (NumeroSerie: String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_EncerraDriver()

Encerra a comunicação com o REP, limpa todos os comandos eventualmente na fila.

Declaração:

int DLLREP_EncerraDriver(int **Handle**)

Parâmetros:

Handle – Handle do REP

Retorno:

1 OK
-1 Erro handle inválido

- Exemplo em VB.NET:

```
Declare Function DLLREP_EncerraDriver Lib "dllrep.dll" (ByVal Handle As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_EncerraDriver(int Handle);
```

- Exemplo em Delphi:

```
function DLLREP_EncerraDriver (iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

DLLREP_Localiza()

Envia comando UDP para localizar REPs na(s) subredes(s). Os REPs localizados deverão ser recuperados pelas funções *DLLREP_TotalRetornos()* e *DLL_RetornoLocaliza()*.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 02.01.00 do SB.

Declaração:

```
int DLLREP_Localiza(string(15) EnderecoIP,
                    int Porta,
                    int Timeout)
```

Parâmetros:

EnderecoIP – Endereço IP para o qual será enviado o comando de localização. Permite endereços de broadcast, como por exemplo, 192.168.0.255.

Porta – Porta do REP para a qual será enviado o comando de localização. Padrão do REP: 5001.

Timeout – Tempo em milissegundos em que será aguardado retorno dos REPs.

Retorno:

Handle do comando de localização
-1 Erro Inicialização

- Exemplo em VB.NET:

```
Declare Function DLLREP_Localiza Lib "dllrep.dll" (ByVal EnderecoIP As String, ByVal Porta As Integer, ByVal Timeout As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_Localiza(System.String EnderecoIP, int Porta, int Timeout);
```

- Exemplo em Delphi:

```
function DLLREP_Localiza (EnderecoIP: String; Porta: Integer; Timeout: Integer): Integer; Stdcall; External 'DLLREP.DLL';
```

DLLREP_RetornoLocaliza()

Recupera informações dos REPs localizados pela função *DLLREP_Localiza()*.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 02.01.00 do SB.

Declaração:

```
int DLLREP_RetornoLocaliza(int Handle,
                           int IndiceLinha,
                           string(4) Modelo,
                           string(17) NumFabricacao,
                           string(8)  VersaoREP,
                           string(15) EnderecoIP,
                           string(5) Porta,
                           string(14) MAC,
                           string(1) Criptografia)
```

Parâmetros de Entrada:

Handle – Handle do comando de localização.

IndiceLinha – índice da linha à recuperar.

Parâmetros de Saída:

Modelo – Modelo do Equipamento

NumFabricacao – Número de Série do Equipamento

VersaoREP – Versão do firmware do software básico

EnderecoIP – Endereço IP do Cliente

Porta – Porta TCP configurada para o modo cliente

MAC – Endereço MAC do Cliente

Criptografia – (habilitado = 1) e (desabilitado = 0)

- Exemplo em VB.NET:

```
Declare Function DLLREP_RetornoLocaliza Lib "dllrep.dll" (ByVal Handle_Localizacao As Integer, ByVal IndiceLinha As Integer, ByVal Modelo As String, ByVal NumFabricacao As String, ByVal VersaoREP As String, ByVal EnderecoIP As String, ByVal Porta As String, ByVal MACAddress As String, ByVal Criptografia As String) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_RetornoLocaliza (int Handle_Localizacao, int IndiceLinha, System.String Modelo, System.String NumFabricacao, System.String VersaoREP, System.String EnderecoIP, System.String Porta, System.String MACAddress, System.String Criptografia);
```

- Exemplo em Delphi:

```
function DLLREP_RetornoLocaliza (Handle_Localizacao: Integer; IndiceLinha: Integer; Modelo: String; NumFabricacao: String; VersaoREP: String; EnderecoIP: String; Porta: String; MACAddress: String; Criptografia: String): Integer; StdCall; External 'DLLREP.DLL';
```

2. Funções de Configuração

➢ DLLREP_Versao()

Retorna a versão da DLL.

Declaração:

void DLLREP_Versao(string(10) **Versao**)

Parâmetros de Saída:

Versao – Versão da DLL

- Exemplo em VB.NET:

```
Declare Sub DLLREP_Versao Lib "dllrep.dll" (ByVal Versao As String)
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern void DLLREP_Versao(StringBuilder Versao);
```

- Exemplo em Delphi:

```
procedure DLLREP_Versao (Versao: String) StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_DefineModoIP()

Define que a comunicação com o REP será pela rede.

Declaração:

```
int DLLREP_DefineModoIP(int Handle,
                        string EnderecoIP,
                        int Porta)
```

Parâmetros:

Handle – Handle do REP
EnderecoIP – Endereço IP ou Host do REP
Porta – Porta do REP para comunicação

Nota: Para as famílias R100 e R200, a porta informada deve ser 5000.

Retorno:

1 OK
-1 Erro Handle Inválido
-7 Erro Parâmetro Inválido/Ausente

- Exemplo em VB.NET:

```
Declare Function DLLREP_DefineModoIP Lib "dllrep.dll" (ByVal Handle As Integer, ByVal EnderecoIP As String, ByVal
Porta As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_DefineModoIP(int Handle, System.String EnderecoIP, int Porta);
```

- Exemplo em Delphi:

function DLLREP_DefineModoIP (iHandle: Integer; EnderecoIP: String; Porta: Integer): Integer; StdCall; External 'DLLREP.DLL';

➢ DLLREP_DefineModoIPCriptografado()

Define que a comunicação com o REP será no modo IP com criptografia. O REP deverá estar configurado para operar no modo criptografado.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 02.01.00 do SB.

Declaração:

int DLLREP_DefineModoIPCriptografado(int **Handle**,
string(15) **EnderecoIP**,
int **Porta**)

Parâmetros:

Handle – Handle do REP.

EnderecoIP – Endereço IP ou Host do REP.

Porta – Porta do REP para comunicação

Retorno:

1 OK
-1 Erro Handle Inválido
-7 Erro Parâmetro Inválido/Ausente

- Exemplo em VB.NET:

```
Declare Function DLLREP_DefineModoIPCriptografado Lib "dllrep.dll" (ByVal Handle As Integer, ByVal EnderecoIP As String, ByVal Porta As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_DefineModoIPCriptografado (int Handle, System.String EnderecoIP, int Porta); /*R300*/
```

- Exemplo em Delphi:

function DLLREP_DefineModoIPCriptografado (iHandle: Integer; EnderecoIP: String; Porta: Integer): Integer; StdCall; External 'DLLREP.DLL';

➢ DLLREP_DefineModoArquivo()

Define que a comunicação com o REP será por troca de arquivos (via pendrive).

Declaração:

int DLLREP_DefineModoArquivo(int **Handle**,
string **Unidade**)

Parâmetros:

Handle – Handle do REP

Unidade – Caminho para a unidade de disco do pendrive. Ex: “F:”

Retorno:

1 OK
-1 Erro Handle Inválido
-7 Erro Parâmetro Inválido/Ausente

- Exemplo em VB.NET:

```
Declare Function DLLREP_DefineModoArquivo Lib "dllrep.dll" (ByVal Handle As Integer, ByVal Unidade As String) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_DefineModoArquivo(int Handle, System.String Unidade);
```

- Exemplo em Delphi:

```
function DLLREP_DefineModoArquivo (iHandle: Integer; Unidade: String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_LeModo()

Retorna ao modo atualmente configurado no REP.

Declaração:

Int DLLREP_LeModo(int **Handle**)

Parâmetros:

Handle – Handle do REP

Retorno:

1: IP
2: Arquivo
3: IP Criptografado (somente família R300)
0: Não configurado

- Exemplo em VB.NET:

```
Declare Function DLLREP_LeModo Lib "dllrep.dll" (ByVal Handle As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_LeModo(int Handle);
```

- Exemplo em Delphi:

```
function DLLREP_LeModo (iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_DefineTimeout()

Define o timeout de comunicação com o REP-ZPM em milissegundos. Este timeout é utilizado como tempo limite para transmissão e recepção do comando. O valor default do Timeout é 10000 (10 segundos).

Declaração:

int DLLREP_DefineTimeout(int **Handle**,
int **Timeout**)

Parâmetros:

Handle – Handle do REP
Timeout - Tempo em milissegundos

Retorno:

1: Sucesso
-1:Erro

- Exemplo em VB.NET:

```
Declare Function DLLREP_DefineTimeout Lib "dllrep.dll" (ByVal Handle As Integer, ByVal Timeout As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_DefineTimeout(int Handle, int Timeout);
```

- Exemplo em Delphi:

```
function DLLREP_DefineTimeout (iHandle: Integer; Timeout: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_LeTimeout()

Verifica o timeout de comunicação definido para o REP.

Declaração:

int DLLREP_LeTimeout (int **Handle**)

Parâmetros:

Handle – Handle do REP

Retorno:

Timeout em milissegundos.
-1: Erro

- Exemplo em VB.NET:

```
Declare Function DLLREP_LeTimeout Lib "dllrep.dll" (ByVal Handle As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_LeTimeout(int Handle);
```

- Exemplo em Delphi:

```
function DLLREP_LeTimeout (iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_ConfiguraCodigoBarra()

Define um tamanho fixo de dígitos para os códigos de barras (CODIGO_B) utilizados nas funções *DLLREP_Funcionario_Prepara()* e *DLLREP_RetornoFuncionario()*. Sempre que for necessário, o parâmetro será sempre precedido pelo caractere '0', a fim de completar o tamanho definido. O valor '0' (default) permite um tamanho variável.

Declaração:

void DLLREP_ConfiguraCodigoBarra (int **Tamanho**)

Parâmetros:

Tamanho - Tamanho da mascara de entradas para o parâmetro CODIGO_B. Valor deve ser entre '2' e '20'. '0' (zero) define um tamanho variável para o campo.

Retorno:

1: Sucesso
-1:Erro

- Exemplo em VB.NET:

```
Declare Sub DLLREP_ConfiguraCodigoBarra Lib "dllrep.dll" (ByVal Tamanho As Integer)
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern void DLLREP_ConfiguraCodigoBarra(int Tamanho);
```

- Exemplo em Delphi:

```
procedure DLLREP_ConfiguraCodigoBarra(Tamanho: Integer) StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_LeCodigoBarra()

Retorna o valor configurado por *DLLREP_ConfiguraCodigoBarra()*.

Declaração:

int DLLREP_LeCodigoBarra(int **Handle**)

Parâmetros:

Handle – Handle do REP

Retorno:

Tamanho da máscara de entradas para o parâmetro CODIGO_B

- Exemplo em VB.NET:

```
Declare Function DLLREP_LeCodigoBarra Lib "dllrep.dll" (ByVal Handle As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_LeCodigoBarra(int Handle);
```

- Exemplo em Delphi:

```
function DLLREP_LeCodigoBarra(iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

DLLREP_ConfiguraMifareInverteHexa()

Define as Funções de Envio e Busca de Funcionário da DLL devem realizar a inversão do campo Código Mifare (Codigo_M). A inversão de bytes é realizada sobre a representação em 32 bits do Código Mifare. A configuração é aplicada a todas as chamadas para *DLLREP_Funcionario_Prepara()* e *DLLREP_BuscaFuncionario_Prepara().*

Declaração:

void DLLREP_ConfiguraMifareInverteHexa(int **Valor**)

Parâmetros:

Valor – '1' para configurar a inversão.
'0' para não realizar a inversão.

Retorno:

1: Sucesso
-1: Erro

- Exemplo em VB.NET:

```
Declare Sub DLLREP_ConfiguraMifareInverteHexa Lib "dllrep.dll" (ByVal Valor As Integer)
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_ConfiguraMifareInverteHexa(int Valor);
```

- Exemplo em Delphi:

```
function DLLREP_ConfiguraMifareInverteHexa(Valor: Integer) StdCall; External 'DLLREP.DLL';
```

DLLREP_LeMifareInverteHexa()

Verifica se a inversão do código Mifare está configurada.

Declaração:

int DLLREP_LeMifareInverteHexa(int **Handle**)

Parâmetros:

Handle – Handle do REP

Retorno:

1: Configurada
0: Não Configurada

- Exemplo em VB.NET:

```
Declare Function DLLREP_LeMifareInverteHexa Lib "dllrep.dll" (ByVal handle As Integer)
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_LeMifareInverteHexa(int handle);
```

- Exemplo em Delphi:

```
function DLLREP_LeMifareInverteHexa(iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_IniciaLeituraEventos()

Passa a ouvir / tratar eventos recebidos de REPs Clientes. O último evento recebido de cada REP pode ser recuperado pelo comando *DLLREP_Retorno_UltimoEvento()*.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 02.01.00 do SB.

Declaração:

Int DLLREP_IniciaLeituraEventos (int **Handle**,
int **Porta**,
int **EnviarComandosNoEvento**)

Parâmetros de Entrada:

Handle – Handle do REP

Porta – Porta de entrada na qual serão aguardados os eventos enviados pelo REP. (Default: 5002)

EnviarComandosNoEvento (1 ativado, 0 desativado) - Informa se os comandos deverão ser enviados para o REP no modo Cliente.

Ativado – todo comando solicitado, será enviado no modo Cliente REP, ou seja, serão colocados em fila pela DLL e executado somente no momento da recepção do próximo evento enviado pelo REP. Através do comando *DLLREP_VerificaRetornoComandoEvento()* poderá ser consultada a disponibilidade do resultado do comando. De forma análoga ao comando *VerificaRetornoPenDrive()*.

Desativado – os comandos continuam sendo enviados pela porta configurada em *IniciaDriver()* para o REP.

Retorno:

1 – OK
-1 – Handle inválido
-62 – Modo IP não configurado

- Exemplo em VB.NET:

```
Declare Function DLLREP_IniciaLeituraEventos Lib "dllrep.dll" (ByVal Handle As Integer, ByVal Porta As Integer, ByVal EnviarComandosNoEvento As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_IniciaLeituraEventos (int Handle, int Porta, int EnviarComandosNoEvento);
```

- Exemplo em Delphi:

```
function DLLREP_IniciaLeituraEventos (iHandle: Integer; Porta: Integer; EnviarComandosNoEvento: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_EncerraLeituraEventos()

Para de ouvir / tratar eventos recebidos de REPs Clientes.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 02.01.00 do SB.

Declaração:

Int DLLREP_EncerraLeituraEventos (int **Handle**)

Parâmetros de Entrada:

Handle – Handle do REP

Retorno:

1 – OK
-1 – Handle inválido
-62 – Modo IP não configurado

- Exemplo em VB.NET:

```
Declare Function DLLREP_EncerraLeituraEventos Lib "dllrep.dll" (ByVal Handle As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_EncerraLeituraEventos (int Handle);
```

- Exemplo em Delphi:

```
function DLLREP_EncerraLeituraEventos (iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_IniciaLog()

Inicia o registro das atividades de comunicação com determinado REP. No arquivo de Log constam os comandos enviados, respostas e mensagens de erro de comunicação.

Se o arquivo definido já existir, não sobrescreve.

Declaração:

```
int DLLREP_IniciaLog(int Handle,
                     string(250) nomeArquivo)
```

Parâmetros:

Handle – Handle do REP
nomeArquivo – nome do arquivo de log a ser gerado. Se não for definido, será gerado um arquivo de log com o nome "REP<Número de Fabricação REP>_<AAAA>_<MM>_<DD>.log".

Retorno:

handle
-1: Erro inicialização
-10: Erro de criação / gravação no log

- Exemplo em VB.NET:

```
Declare Function DLLREP_IniciaLog Lib "dllrep.dll" (ByVal Handle As Integer, Byval nomeArquivo As String) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_IniciaLog(int Handle, StringBuilder nomeArquivo);
```

- Exemplo em Delphi:

```
function DLLREP_IniciaLog(iHandle: Integer; nomeArquivo:String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_EncerraLog()

Encerra o registro de Log para determinado REP.

Declaração:

int DLLREP_EncerraLog(int **Handle**)

Parâmetros:

Handle – Handle do REP

Retorno:

1: OK
-1: Erro Handle inválido

- Exemplo em VB.NET:

```
Declare Function DLLREP_EncerraLog Lib "dllrep.dll" (ByVal Handle As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_EncerraLog(int Handle);
```

- Exemplo em Delphi:

```
function DLLREP_EncerraLog(iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_ConfigMascaraBarras()

Este comando possibilita a configuração da máscara do código de barras lido pelo equipamento, assim podendo remover ou alterar caracteres do código de barras. Sempre alinhado a direita.
Este comando esta funcional apenas por comunicação via rede.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 04.00.08 do SB.

Declaração:

```
int DLLREP_ConfigMascaraBarras (int Handle,
                                char *Digito01,
                                char *Digito02,
                                char *Digito03,
                                char *Digito04,
                                char *Digito05,
                                char *Digito06,
                                char *Digito07,
                                char *Digito08,
                                char *Digito09,
                                char *Digito10,
                                char *Digito11,
                                char *Digito12,
                                char *Digito13,
                                char *Digito14,
                                char *Digito15,
                                char *Digito16,
                                char *Digito17,
                                char *Digito18,
                                char *Digito19,
                                char *Digito20);
```

Parâmetros:

Handle – Handle do REP
Digito01 – Primeiro caractere do código
Digito02 – Segundo caractere do código
Digito03 – Terceiro caractere do código
Digito04 – Quarto caractere do código
Digito05 – Quinto caractere do código
Digito06 – Sexto caractere do código
Digito07 – Sétimo caractere do código
Digito08 – Oitavo caractere do código
Digito09 – Nono caractere do código
Digito10 – Décimo caractere do código
Digito11 – Décimo primeiro caractere do código
Digito12 – Décimo segundo caractere do código
Digito13 – Décimo terceiro caractere do código
Digito14 – Décimo quarto caractere do código
Digito15 – Décimo quinto caractere do código
Digito16 – Décimo sexto caractere do código
Digito17 – Décimo sétimo caractere do código
Digito18 – Décimo oitavo caractere do código
Digito19 – Décimo nono caractere do código
Digito20) – vigésimo caractere do código

Retorno:

>=0 Linha do Comando
-1 Erro Handle Inválido
-2 Erro de conexão/Comunicação (apenas modo IP)
-4 Erro de Timeout (apenas modo IP)
-5 Erro de Protocolo (apenas modo IP)
-10 Erro de gravação (apenas modo arquivo)

Exemplos:

Código de barras: 12345678901234567890 (20 caracteres).

Enviando:
DLLREP_ConfigMascaraBarras**Handle**,1”,”1”,”1”,”1”,”1”,”1”,”1”,”1”,”1”,”1”,”20”,”20”,”20”,”20”,”20”,”20”,”20”,”20”;”20”,”20”,”20”);
Teremos o código lido: 11111111110000000000

DLLREP_ConfigMascaraBarras**Handle**,”9”,”9”,”10”,”10”,”11”,”11”,”12”,”12”,”13”,”13”,”14”,”14”,”15”,”15”,”16”,”16”,”17”;”17”,”18”,”18”);
Teremos o código lido: 99001122334455667788

➢ DLLREP_ConfigModoTAG()

Altera o dado lido pelo leitor de proximidade TAG/RFID (125kHz).
Este comando esta funcional apenas por comunicação via rede.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 04.00.08 do SB.

Declaração:

```
int DLLREP_ConfigModoTAG (int Handle,
                          char *Modo);
```

Parâmetros:

Handle – Handle do REP
Modo –
1: Decimal
2: Hexadecimal

Retorno:

>=0 Linha do Comando
-1 Erro Handle Inválido
-2 Erro de conexão/Comunicação (apenas modo IP)
-4 Erro de Timeout (apenas modo IP)
-5 Erro de Protocolo (apenas modo IP)
-10 Erro de gravação (apenas modo arquivo)

➢ DLLREP_ConfigCriptoBarras()

Este comando possibilita a utilização de criptografia na leitura do código de barras.

Para que este comando seja executado com sucesso, será necessári um arquivo específico contendo as informações criptográficas do código de barras.
Este comando esta funcional apenas por comunicação via rede.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 04.00.08 do SB.

Declaração:

```
int DLLREP_ConfigCriptoBarras (int Handle,
                               char *Arq,
                               char *Cripto);
```

Parâmetros:

Handle – Handle do REP
Arq – Caminha completo de onde esta o arquivo de criptografia (máximo 1024 caracteres)
Cripto –
1: Habilita a criptografia
0: Desabilita a criptografia

Retorno:

>=0 Linha do Comando
-1 Erro Handle Inválido
-2 Erro de conexão/Comunicação (apenas modo IP)
-4 Erro de Timeout (apenas modo IP)
-5 Erro de Protocolo (apenas modo IP)
-10 Erro de gravação (apenas modo arquivo)
-80 Erro na abertura do arquivo

➢ DLLREP_ConfigModoMIFARE()

Altera o dado lido pelo leitor de proximidade MIFARE.

Este dado pode ser representado no formato little-endian ou big-endian.

Por exemplo, um cartão com ID = 2864434397 pode ser representado em Hexa por AABBCCDD.

-Configurado como little-endian o ID em hexa será AABBCCDD e em decimal será 2864434397.

-Configurado como big-endian o ID em hexa será DDCCBBAA e em decimal será 3721182122.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 04.00.18 do SB.

Declaração:

int DLLREP_ConfigModoMifare (int **Handle**,
char ***Modo**);

Parâmetros:

Handle – Handle do REP
Modo –
1: Little-endian
2: Big-endian

Retorno:

>=0 Linha do Comando
-1 Erro Handle Inválido
-2 Erro de conexão/Comunicação (apenas modo IP)
-4 Erro de Timeout (apenas modo IP)
-5 Erro de Protocolo (apenas modo IP)
-10 Erro de gravação (apenas modo arquivo)

3. Funções de Atualizações do REP

Essas funções preparam comandos que provocarão modificações no cadastro de empregador ou funcionários do REP. As funções de cadastro de empregador e ajuste de relógio (*DLLREP_Empregador()* e *DLLREP_AtualizaDataHora()*) são atômicas e enviadas imediatamente após sua execução. Já para o cadastro de funcionários, é necessário preparar primeiramente os comandos através da função *DLLREP_Funcionario_Prepara()* e depois executar a lista de comandos com *DLLREP_Funcionario_Envia()*.

Se a operação for executada com sucesso, será retornado um identificador de comando válido. No modo arquivo, ele será necessário para recuperar o resultado através da função *DLLREP_VerificaRetornoPenDrive ()*. No modo IP, pode-se imediatamente obter os resultados e verificar os erros através das Funções de Tratamento de Erro.

➢ DLLREP_Empregador()

Executa operação de Inclusão ou Atualização do Empregador. Pode haver somente um Empregador cadastrado em um REP.

Declaração:

int DLLREP_Empregador(int **Handle**,
int **Operacao**,
string(1) **Tipo**,
string(14) **Identificacao**,
string(12) **CEI**,
string(150) **RazaoSocial**,
string(100) **LocalTrabalho**)

Parâmetros:

Handle – Handle do REP

Operação –

1 – Inclusão
2 – Alteração
4 – Inclusão / Alteração (somente família R300)

Tipo* – "J": Pessoa Jurídica (CNPJ)
"F" Pessoa Física (CPF)

Identificacao* – CNPJ ou CPF

CEI* – CEI

RazaoSocial* – Razão Social do Empregador

LocalTrabalho* – Local de Trabalho

** Para os comandos de Alteração (operacao=2), valores de parâmetro NULL ou string em branco (""), não alteram o conteúdo do campo. Para apagar o conteúdo de um campo deve-se passar como parâmetro uma string composta do caractere "#".*

Retorno:

>=0 Identificador do Comando

-1 Erro Handle Inválido

-2 Erro de conexão/Comunicação (apenas modo ip)

-4 Erro de Timeout (apenas modo IP)

-5 Erro de Protocolo (apenas modo IP)

-10 Erro de gravação (apenas modo arquivo)

-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos ()* para liberar o buffer.

- Exemplo em VB.NET:

```
Declare Function DLLREP_Empregador Lib "dllrep.dll" (ByVal Handle As Integer, ByVal Operacao As Integer, ByVal Tipo As String, ByVal Identificacao As String, ByVal CEI As String, ByVal RazaoSocial As String, ByVal LocalTrabalho As String) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_Empregador(int Handle,int Operacao, System.String Tipo, System.String Identificacao, System.String CEI, System.String RazaoSocial, System.String LocalTrabalho);
```

- Exemplo em Delphi:

```
function DLLREP_Empregador (iHandle: Integer; Operacao: Integer; Tipo: String; Identificacao: String; CEI: String; RazaoSocial: String; LocalTrabalho: String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_Funcionario_Prepara()

Prepara comando de Cadastro de Funcionários. Ao contrário dos outros comandos, este comando não é executado imediatamente, mas colocado numa fila. Para efetivamente enviar as operações sobre funcionários colocados na fila, utilize o comando *DLLREP_Funcionario_Envia()*. É possível enviar múltiplos comandos para Inclusão, Exclusão ou Alteração, desde que todos os comandos seja do mesmo tipo.

Declaração:

```
int DLLREP_Funcionario_Prepara (int Handle,
                                int Operacao,
                                string(11) PIS,
                                string(20) Matricula,
                                string(52) NomeFunc,
                                string (20000) TemplateBiometrico,
                                string(1) PIS_Teclado,
                                string(16) Codigo_K,
                                string(20) Codigo_B,
                                string(20) Codigo_M,
                                string(20) Codigo_T)
```

Parâmetros:

Handle – Handle do REP

Operação –

0 – Exclusão

1 – Inclusão

2 – Alteração

4 – Inclusão / Alteração (somente família R300)

PIS – Pis do Funcionário

Matricula* – Matrícula do Funcionário

Nome* – Nome do Empregado (obrigatório para operações 1 e 2)

TemplateBiometrico* – Lista de Templates Biométricos separados por ponto-e-vírgula (";"). Limite de 1 Template para Linha R200, e de 10 Templates para demais modelos.

PIS_Teclado* – "S": Permite digitar PIS no teclado.
"N": Não permite.

Codigo_K* – código para ser usado no teclado (keyboard)

Codigo_B* – código a ser usado no cartão de barras. Configurável por *DLLREP_ConfiguraCodigoBarra()*

Codigo_M* – código a ser utilizado no MIFARE

Codigo_T* – código a ser utilizado com o TAG

**Para os comandos de Alteração (operacao=2), valores de parâmetro NULL ou string em branco (""), não alteram o conteúdo do campo. Para apagar o conteúdo de um campo deve-se passar como parâmetro uma string composta do caractere "#".*

***Para o comando de Exclusão (operação=3) , é necessário passar somente o PIS como parâmetro. Os parâmetros restantes serão ignorados*

Retorno:

>=0 Linha do Comando

-1: Erro Handle inválido

-11: Erro. Existem comandos incompatíveis com este no buffer de envio. Utilize *DLLREP_LimpaComandos()* para liberar o buffer.

-20: Erro em algum parâmetro.

-22: Tamanho do Codigo de Barras Invalido

- Exemplo em VB.NET:

```
Declare Function DLLREP_Funcionario_Prepara Lib "dllrep.dll" (ByVal Handle As Integer, ByVal Operacao As Integer, ByVal PIS As String, ByVal Matricula As String, ByVal NomeFuncionario As String, ByVal TemplateBiometrico As String, ByVal PIS_Teclado As String, ByVal CodigoTeclado As String, ByVal Codigobarra As String, ByVal CodigoMifare As String, ByVal CodigoTAG As String) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_Funcionario_Prepara(int Handle, int Operacao, System.String PIS, System.String Matricula, System.String NomeFuncionario, System.String TemplateBiometrico, System.String PIS_Teclado, System.String CodigoTeclado, System.String CodigoBarra, System.String CodigoMifare, System.String CodigoTAG);
```

- Exemplo em Delphi:

```
function DLLREP_Funcionario_Prepara (iHandle: Integer; Operacao: Integer; PIS: String; Matricula: String; NomeFuncionario: String; Biometrico: String; Habilita_Teclado: String; Cod_Teclado: String; Cod_Barras: String; Cod_MIFARE: String; Cod_TAG: String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_Funcionario_Envia()

Envia os comandos de funcionário enfileirados com *DLLREP_Funcionario_Prepara()* ou *DLLREP_BuscaFuncionario_Prepara()* para o REP.

Declaração:

int DLLREP_Funcionario_Envia (int **Handle**)

Parâmetros:

Handle – Handle do REP

Retorno:

>=0 Linha do Comando

-1 Erro Handle Inválido

-2 Erro de conexão/Comunicação (apenas modo IP)

-4 Erro de Timeout (apenas modo IP)

-5 Erro de Protocolo (apenas modo IP)

-10 Erro de gravação (apenas modo arquivo)

- Exemplo em VB.NET:

```
Declare Function DLLREP_Funcionario_Envia Lib "dllrep.dll" (ByVal Handle As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_Funcionario_Envia(int Handle);
```

- Exemplo em Delphi:

```
function DLLREP_Funcionario_Envia (iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_AtualizaDataHora()

Atualiza a data/ hora do REP para determinada data. Se não for inserido parâmetro, atualiza para a data corrente do computador.

Declaração:

int DLLREP_AtualizaDataHora(int **Handle**,
string(19) **DataHora**)

Parâmetros:

Handle – Handle do REP

DataHora (opcional) – Data/hora nos formatos DD/MM/YYYY HH:mm:SS

Retorno:

>=0 Identificador do Comando

-1 Erro Handle Inválido

-2 Erro de conexão/Comunicação (apenas modo ip)

-4 Erro de Timeout (apenas modo IP)

-5 Erro de Protocolo (apenas modo IP)

-10 Erro de gravação (apenas modo arquivo)

-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos ()* para liberar o buffer.

- Exemplo em VB.NET:

```
Declare Function DLLREP_AtualizaDataHora Lib "dllrep.dll" (ByVal Handle As Integer, ByVal DataHora As String) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_AtualizaDataHora(int Handle, System.String DataHora);
```

- Exemplo em Delphi:

```
function DLLREP_AtualizaDataHora (iHandle: Integer; DataHora: String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_AjustaHorarioVerao ()

Ajusta ou exclui o período de vigência do horário de verão.

Declaração:

int DLLREP_AjustaHorarioVerao (int **Handle**,
string(10) **DataInicio**,
string(10) **DataFim**)

Parâmetros:

Handle – Handle do REP

DataInicio – Início da vigência do horário de verão(dd/mm/yyyy) ou # para excluir a Data de Início atual

DataFim – Fim da vigência do horário de verão(dd/mm/yyyy) ou # para excluir o Fim do Período

Retorno:

>=0 Identificador do Comando

-1 Erro Handle Inválido

-2 Erro de conexão/Comunicação (apenas modo IP)

-4 Erro de Timeout (apenas modo IP)

-5 Erro de Protocolo (apenas modo IP)

-10 Erro de gravação (apenas modo arquivo)

-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos()* para liberar o buffer.

- Exemplo em VB.NET:

```
Declare Function DLLREP_AjustaHorarioVerao Lib "dllrep.dll" (ByVal Handle As Integer, ByVal DataInicio As String, ByVal DataFim As String) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_AjustaHorarioVerao(int Handle, System.String DataInicio, System.String DataFim);
```

- Exemplo em Delphi:

```
function DLLREP_AjustaHorarioVerao(iHandle: Integer; DataInicio: String; DataFim: String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_ReinicializaSenhas()

Este comando faz com que as senhas de administrador, supervisor do equipamento volte à configuração de fábrica.

OBS.: Este comando está disponível nos equipamentos:

- Família R100 a partir da versão 01.40.31 do SB;
- Família R200 a partir da versão 03.00.19 do SB;
- Família R300 a partir da versão 02.01.00 do SB.

Declaração:

int DLLREP_ReinicializaSenhas(int **Handle**)

Parâmetros:

Handle – Handle do REP

Retorno:

>=0 Identificador do Comando

-1 Erro Handle Inválido

-2 Erro de conexão/Comunicação (apenas modo IP)

-4 Erro de Timeout (apenas modo IP)

-5 Erro de Protocolo (apenas modo IP)

- Exemplo em VB.NET:

```
Declare Function DLLREP_ReinicializaSenhas Lib "dllrep.dll" (ByVal Handle As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_ReinicializaSenhas(int Handle);
```

- Exemplo em Delphi:

```
function DLLREP_ReinicializaSenhas(iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

4. Funções de Leitura do REP

São funções que recuperam informações do REP-ZPM.

➢ DLLREP_BuscaEmpregador()

Busca dados do Empregador. A função *DLLREP_RetornoEmpregador()* recupera os dados lidos.

Declaração:

int DLLREP_BuscaEmpregador (int **Handle**)

Parâmetros:

Handle – Handle do REP

Retorno:

>=0 Identificador do Comando
-1 Erro Handle Inválido
-2 Erro de conexão/Comunicação (apenas modo IP)
-4 Erro de Timeout (apenas modo IP)
-5 Erro de Protocolo (apenas modo IP)
-10 Erro de gravação (apenas modo arquivo)
-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos()* para liberar o buffer.

- Exemplo em VB.NET:

```
Declare Function DLLREP_BuscaEmpregador Lib "dllrep.dll" (ByVal Handle As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_BuscaEmpregador(int Handle);
```

- Exemplo em Delphi:

```
function DLLREP_BuscaEmpregador (iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_BuscaTodosFuncionarios()

Prepara o comando para busca informações de todos os Funcionário cadastrados no REP. A função *DLLREP_TotalRetornos()* retorna o número de funcionários lidos. Para recuperar as informações de cada funcionário utilize *DLLREP_RetornoFuncionario().*

Declaração:

int DLLREP_BuscaTodosFuncionarios (int **Handle**)

Parâmetros:

Handle – Handle do REP

Retorno:

>=0 Identificador do Comando

-1 Erro Handle Inválido

-2 Erro de conexão/Comunicação (apenas modo IP)

-4 Erro de Timeout (apenas modo IP)

-5 Erro de Protocolo (apenas modo IP)

-10 Erro de gravação (apenas modo arquivo)

-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos()* para liberar o buffer.

- Exemplo em VB.NET:

```
Declare Function DLLREP_BuscaTodosFuncionarios Lib "dllrep.dll" (ByVal Handle As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_BuscaTodosFuncionarios(int Handle);
```

- Exemplo em Delphi:

```
function DLLREP_BuscaTodosFuncionarios (iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_BuscaFuncionario_Prepara()

Prepara o comando para busca de informações de determinados Funcionários. Este comando não é executado imediatamente, mas colocado numa fila. Para efetivamente enviar o comando ao REP, utilize o comando *DLLREP_Funcionario_Envia()*. Caso seja enviado com sucesso, os resultados devem ser obtidos através da função *DLLREP_RetornoFuncionario()*.

Declaração:

```
int DLLREP_BuscaFuncionario_Prepara(int Handle,
                                    string(11) PIS)
```

Parâmetros:

Handle – Handle do REP

PIS – PIS do Funcionário

Retorno:

>=0 Linha do Comando

-1: Erro: Handle inválido

-11 Erro. Existem comandos incompatíveis com este no buffer de envio. Utilize *DLLREP_LimpaComandos()* para liberar o buffer.

- Exemplo em VB.NET:

```
Declare Function DLLREP_BuscaFuncionario_Prepara Lib "dllrep.dll" (ByVal Handle As Integer, ByVal PIS As String) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_BuscaFuncionario_Prepara(int Handle, System.String PIS);
```

- Exemplo em Delphi:

```
function DLLREP_BuscaFuncionario_Prepara (iHandle: Integer; PIS: String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_BuscaPonto()

Efetua a busca da marcação de pontos num determinado intervalo de data. A função *DLLREP_TotalRetornos()* retorna o número de marcações lidas. Para recuperar as informações de cada marcação, utilize *DLLREP_RetornoPonto()*.

Declaração:

```
int DLLREP_BuscaPonto(int Handle,
                      string(19) DataInicio,
                      string(19) DataFim)
```

Parâmetros:

Handle – Handle do REP

DataInicio – Data Inicial para Busca de Pontos (formato DD/MM/YYYY)

DataInicio – Data Inicial para Busca de Pontos (formato DD/MM/YYYY HH:MM:SS) (Somente para a família R300)

DataFim – Data Final para Busca de Pontos (formato DD/MM/YYYY)

DataFim – Data Final para Busca de Pontos (formato DD/MM/YYYY HH:MM:SS) (Somente para a família R300)

Retorno:

>=0 Identificador do Comando

-1 Erro Handle Inválido

-2 Erro de conexão/Comunicação (apenas modo IP)

-4 Erro de Timeout (apenas modo IP)

-5 Erro de Protocolo (apenas modo IP)

-10 Erro de gravação (apenas modo arquivo)

-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos ()* para liberar o buffer.

-21 Formato de Data Inválido

Exemplo em VB.NET:

```
Declare Function DLLREP_BuscaPonto Lib "dllrep.dll" (ByVal Handle As Integer, ByVal DataInicio As String, ByVal DataFim As String) As Integer
```

Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_BuscaPonto(int Handle, System.String DataInicio, System.String DataFim);
```

Exemplo em Delphi:

```
function DLLREP_BuscaPonto (iHandle: Integer; DataInicio: String; DataFim: String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_BuscaPonto_NSR()

Efetua a busca da marcação de pontos num determinado intervalo de NSR. A função *DLLREP_TotalRetornos()* retorna o número de marcações lidas. Para recuperar as informações de cada marcação, utilize *DLLREP_RetornoPonto()*.

Declaração:

```
int DLLREP_BuscaPonto_NSR(int Handle,
                          string(10) NSRInicio,
                          string(10) NSRFim)
```

Parâmetros:

Handle – Handle do REP

NSRInicio – NSR Inicial para Busca de Pontos

NSRFim – NSR Final para Busca de Pontos

Retorno:

>=0 Identificador do Comando

-1 Erro Handle Inválido

-2 Erro de conexão/Comunicação (apenas modo IP)

-4 Erro de Timeout (apenas modo IP)

-5 Erro de Protocolo (apenas modo IP)

-10 Erro de gravação (apenas modo arquivo)

-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos ()* para liberar o buffer.

-21 Formato de Data Inválido

- Exemplo em VB.NET:

```
Declare Function DLLREP_BuscaPonto_NSR Lib "dllrep.dll" (ByVal Handle As Integer, ByVal NSRInicio As String, ByVal NSRFim As String) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_BuscaPonto_NSR(int Handle, System.String NSRInicio, System.String NSRFim);
```

- Exemplo em Delphi:

```
function DLLREP_BuscaPonto_NSR (iHandle: Integer; NSRInicio: String; NSRFim: String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_BuscaStatus()

Comando para buscar informações de Status do REP. Utilize *DLLREP_RetornoStatus()* para recuperar as informações.

Declaração:

int DLLREP_BuscaStatus(int **Handle**)

Parâmetros:

Handle – Handle do REP

Retorno:

>=0 Identificador do Comando

-1 Erro Handle Inválido

-2 Erro de conexão/Comunicação (apenas modo IP)

-4 Erro de Timeout (apenas modo IP)

-5 Erro de Protocolo (apenas modo IP)

-10 Erro de gravação (apenas modo arquivo)

-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos ()* para liberar o buffer.

- Exemplo em VB.NET:

```
Declare Function DLLREP_BuscaStatus Lib "dllrep.dll" (ByVal Handle As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_BuscaStatus(int Handle);
```

- Exemplo em Delphi:

```
function DLLREP_BuscaStatus (iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_BuscaStatusCompleto()

Comando para buscar informações adicionais de Status do REP. Utilize *DLLREP_RetornoStatus()* para recuperar as informações.

OBS.: Este comando está disponível nos equipamentos:

- Família R100 a partir da versão 01.40.28 do SB;
- Família R200 a partir da versão 03.00.19 do SB;
- Família R300 a partir da versão 02.01.00 do SB.

Declaração:

int DLLREP_BuscaStatusCompleto(int **Handle**)

Parâmetros:

Handle – Handle do REP

Retorno:

>=0 Identificador do Comando
-1 Erro Handle Inválido
-2 Erro de conexão/Comunicação (apenas modo IP)
-4 Erro de Timeout (apenas modo IP)
-5 Erro de Protocolo (apenas modo IP)
-10 Erro de gravação (apenas modo arquivo)
-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos ()* para liberar o buffer.

- Exemplo em VB.NET:

```
Declare Function DLLREP_BuscaStatusCompleto Lib "dllrep.dll" (ByVal Handle As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_BuscaStatusCompleto(int Handle);
```

- Exemplo em Delphi:

```
function DLLREP_BuscaStatusCompleto (iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_BuscaLeituraMRP ()

Efetua a busca do conteúdo da MRP em um determinado intervalo de data no formato AFD. A função *DLLREP_TotalRetornos()* retorna o número de marcações lidas. Para recuperar os registros AFD, utilize *DLLREP_RetornoLeituraMRP().*

Declaração:

int DLLREP_BuscaLeituraMRP(int **Handle**,
string(19) **DataInicio**,
string(19) **DataFim**)

Parâmetros:

Handle – Handle do REP

DataInicio – Data Inicial para Leitura da MRP (formato DD/MM/YYYY)

DataInicio – Data Inicial para Busca de Pontos (formato DD/MM/YYYY HH:MM:SS) (Somente p/ a família R300)

DataFim – Data Final para Leitura da MRP (formato DD/MM/YYYY)

DataFinal – Data Final para Busca de Pontos (formato DD/MM/YYYY HH:MM:SS) (Somente p/ a família R300)

Retorno:

>=0 Identificador do Comando

-1 Erro Handle Inválido

-2 Erro de conexão/Comunicação (apenas modo IP)

-4 Erro de Timeout (apenas modo IP)

-5 Erro de Protocolo (apenas modo IP)

-10 Erro de gravação (apenas modo arquivo)

-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos ()* para liberar o buffer.

-21 Formato de Data Inválido

Exemplo em VB.NET:

```
Declare Function DLLREP_BuscaLeituraMRP Lib "dllrep.dll" (ByVal Handle As Integer, ByVal DataInicio As String, ByVal DataFim As String) As Integer
```

Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_BuscaLeituraMRP(int Handle, System.String DataInicio, System.String DataFim);
```

Exemplo em Delphi:

```
function DLLREP_BuscaLeituraMRP (iHandle: Integer; DataInicio: String; DataFim: String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_BuscaLeituraMRP_NSR ()

Efetua a busca do conteúdo da MRP em um determinado intervalo de NSR no formato AFD. A função *DLLREP_TotalRetornos()* retorna o número de marcações lidas. Para recuperar os registros AFD, utilize *DLLREP_RetornoLeituraMRP().*

OBS.: Este comando está disponível nos equipamentos:

- Família R100 a partir da versão 01.40.25 do SB;
- Família R200 a partir da versão 03.00.19 do SB.

Declaração:

int DLLREP_BuscaLeituraMRP_NSR(int **Handle**,
string(10) **NSRInicio**,
string(10) **NSRFim**)

Parâmetros:

Handle – Handle do REP

NSRInicio – NSR Inicial para Leitura da MRP

NSRFim – NSR Final para Leitura da MRP

Retorno:

>=0 Identificador do Comando

-1 Erro Handle Inválido

-2 Erro de conexão/Comunicação (apenas modo IP)

-4 Erro de Timeout (apenas modo IP)

-5 Erro de Protocolo (apenas modo IP)

-10 Erro de gravação (apenas modo arquivo)

-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos ()* para liberar o buffer.

- Exemplo em VB.NET:

```
Declare Function DLLREP_BuscaLeituraMRP_NSR Lib "dllrep.dll" (ByVal Handle As Integer, ByVal NSRInicio As String, ByVal NSRFim As String) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_BuscaLeituraMRP_NSR(int Handle, System.String NSRInicio, System.String NSRFim);
```

- Exemplo em Delphi:

```
function DLLREP_BuscaLeituraMRP_NSR (iHandle: Integer; NSRInicio: String; NSRFim: String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_BuscaHorarioVerao ()

Comando para buscar as definições vigentes para início e fim do horário de verão. Utilize *DLLREP_RetornoHorarioVerao()* para recuperar as informações.

Declaração:

int DLLREP_BuscaHorarioVerao (int **Handle**)

Parâmetros:

Handle – Handle do REP

Retorno:

>=0 Identificador do Comando
-1 Erro Handle Inválido
-2 Erro de conexão/Comunicação (apenas modo IP)
-4 Erro de Timeout (apenas modo IP)
-5 Erro de Protocolo (apenas modo IP)
-10 Erro de gravação (apenas modo arquivo)
-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos ()* para liberar o buffer.

- Exemplo em VB.NET:

```
Declare Function DLLREP_BuscaHorarioVerao Lib "dllrep.dll" (ByVal Handle As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_BuscaHorarioVerao(int Handle);
```

- Exemplo em Delphi:

```
function DLLREP_BuscaHorarioVerao (iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_BuscaEvento ()

Solicita explicitamente o status de eventos do equipamento. Caso haja sucesso, pode ser recuperado pelo DLLREP_RetornoUltimoEvento().

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 02.01.00 do SB.

Declaração:

int DLLREP_BuscaEvento (int **Handle**)

Parâmetros:

Handle – Handle do REP

Retorno:

>=0 Identificador do Comando
-1 Erro Handle Inválido
-2 Erro de conexão/Comunicação (apenas modo IP)
-4 Erro de Timeout (apenas modo IP)
-5 Erro de Protocolo (apenas modo IP)
-10 Erro de gravação (apenas modo arquivo)
-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos ()* para liberar o buffer.

- Exemplo em VB.NET:

```
Declare Function DLLREP_BuscaEvento Lib "dllrep.dll" (ByVal Handle As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_BuscaEvento (int Handle);
```

- Exemplo em Delphi:

```
function DLLREP_BuscaEvento (iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_BuscaMascaraBarras()

Busca a configuração de máscara para o código de barras configurado no equipamento.
Este comando esta funcional apenas por comunicação via rede.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 04.00.08 do SB.

Declaração:

int DLLREP_BuscaMascaraBarras (int **Handle**);

Parâmetros:

Handle – Handle do REP

Retorno:

>=0 Identificador do Comando
-1 Erro Handle Inválido
-2 Erro de conexão/Comunicação (apenas modo IP)
-4 Erro de Timeout (apenas modo IP)
-5 Erro de Protocolo (apenas modo IP)
-10 Erro de gravação (apenas modo arquivo)
-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos ()* para liberar o buffer.

➢ DLLREP_BuscaUltimosRegistros()

Retorna os 10 últimos registros da MRP.
Este comando esta funcional apenas por comunicação via rede.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 04.00.08 do SB.

Declaração:

int DLLREP_BuscaUltimosRegistros (int **Handle**);

Parâmetros:

Handle – Handle do REP

Retorno:

>=0 Identificador do Comando
-1 Erro Handle Inválido
-2 Erro de conexão/Comunicação (apenas modo IP)
-4 Erro de Timeout (apenas modo IP)
-5 Erro de Protocolo (apenas modo IP)
-10 Erro de gravação (apenas modo arquivo)
-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos ()* para liberar o buffer.

➢ DLLREP_BuscaModoTAG()

Busca a configuração do TAG no equipamento.
Este comando esta funcional apenas por comunicação via rede.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 04.00.08 do SB.

Declaração:

int DLLREP_BuscaModoTAG (int **Handle**);

Parâmetros:

Handle – Handle do REP

Retorno:

>=0 Identificador do Comando
-1 Erro Handle Inválido
-2 Erro de conexão/Comunicação (apenas modo IP)
-4 Erro de Timeout (apenas modo IP)
-5 Erro de Protocolo (apenas modo IP)
-10 Erro de gravação (apenas modo arquivo)
-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos ()* para liberar o buffer.

➢ DLLREP_BuscaCriptoBarras()

Busca as informações criptográficas do código de barras do equipamento.
Este comando esta funcional apenas por comunicação via rede.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 04.00.08 do SB.

Declaração:

int DLLREP_BuscaCriptoBarras (int **Handle**);

Parâmetros:

Handle – Handle do REP

Retorno:

>=0 Identificador do Comando
-1 Erro Handle Inválido
-2 Erro de conexão/Comunicação (apenas modo IP)
-4 Erro de Timeout (apenas modo IP)
-5 Erro de Protocolo (apenas modo IP)
-10 Erro de gravação (apenas modo arquivo)
-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos ()* para liberar o buffer.

➢ DLLREP_BuscaModoMifare()

Busca a configuração do Mifare no equipamento.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 04.00.18 do SB.

Declaração:

int DLLREP_BuscaModoMifare (int **Handle**);

Parâmetros:

Handle – Handle do REP

Retorno:

>=0 Identificador do Comando
-1 Erro Handle Inválido
-2 Erro de conexão/Comunicação (apenas modo IP)
-4 Erro de Timeout (apenas modo IP)
-5 Erro de Protocolo (apenas modo IP)
-10 Erro de gravação (apenas modo arquivo)
-11 Erro. Comando no buffer de envio. Utilize *DLLREP_LimpaComandos ()* para liberar o buffer.

5. Funções de Transmissão de Comandos

➢ DLLREP_LimpaComandos ()

Remove todos os comandos na fila do REP, enviados ou não.

Declaração:

int DLLREP_LimpaComandos(int **Handle**)

Parâmetros:

Handle – Handle do REP

Retorno:

1: Sucesso

-1:Erro

- Exemplo em VB.NET:

```
Declare Function DLLREP_LimpaComandos Lib "dllrep.dll" (ByVal Handle As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_LimpaComandos(int Handle);
```

- Exemplo em Delphi:

```
function DLLREP_LimpaComandos (iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_VerificaRetornoPenDrive ()

Somente para o modo Arquivo. Verifica se o arquivo de retorno para o comando já está disponível na unidade. Se estiver, carrega os respectivos resultados, que podem ser acessados através das funções de tratamento de erro e de retorno.

Declaração:

int DLLREP_VerificaRetornoPenDrive(int **Handle**,
int **idComando**)

Parâmetros:

Handle – Handle do REP

idComando – Identificador do Comando retornado pelo comando

Retorno:

1: OK, Arquivo Processado. Retorno Disponível

-40 Erro. Arquivo de comandos continua na Unidade. (Não foi processado ainda pelo REP)

-41 Erro. Arquivo de retorno não encontrado

-42 Erro. Não foi possível acessar a unidade

- Exemplo em VB.NET:

```
Declare Function DLLREP_VerificaRetornoPenDrive Lib "dllrep.dll" (ByVal Handle As Integer, ByVal idComando As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_VerificaRetornoPenDrive(int Handle, int ID_Comando);
```

- Exemplo em Delphi:

```
function DLLREP_VerificaRetornoPenDrive (iHandle: Integer; ID_Comando: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_VerificaRetornoComandoEvento ()

Somente disponível na comunicação por Rede através do modo Cliente (*DLLREP_IniciaLeituraEventos()* com opção EnviarComandosNoEvento habilitada) Verifica se o resultado do comando colocado na fila para execução durante já está disponível. Se estiver, carrega os respectivos resultados, que podem ser acessados através das funções de tratamento de erro e de retorno.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 02.01.00 do SB.

Declaração:

```
int DLLREP_VerificaRetornoComandoEvento (int Handle,
                                         int IDComando)
```

Parâmetros:

Handle – Handle do REP

IDComando – Identificador do comando retornado pelo comando

Retorno:

1 – OK, arquivo processado. Retorno disponível

-50 – Aguardando evento do REP para enviar comando

-51 - Comando está sendo processado pelo REP.

- Exemplo em VB.NET:

```
Declare Function DLLREP_VerificaRetornoComandoEvento Lib "dllrep.dll" (ByVal Handle As Integer, ByVal IDComando As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_VerificaRetornoComandoEvento (int Handle, int IDComando);
```

- Exemplo em Delphi:

```
function DLLREP_VerificaRetornoComandoEvento (iHandle: Integer; IDComando: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

6. Funções de Tratamento de Erro

➢ DLLREP_ObtemCodigoErro()

Exibe o código de erro retornado pelo REP.

Declaração:

int DLLREP_ObtemCodigoErro(int **Handle**,
int **idLinha**)

Parâmetros de Entrada:

Handle – Handle do REP

idLinha – Se foi executado um comando simples, deve ser igual a 1. No caso de operações múltiplas sobre funcionários (com *DLLREP_Funcionario_Prepara()* ou *DLLREP_BuscaFuncionario_Prepara()*), deve corresponder ao n-ésimo comando.

Retorno:

0 = OK, sem erro
1 = Registro mal formado
2 = Valor inválido
3 = Falha durante a operação do comando
4 = Comando inválido
5 = Registro não encontrado (comando de busca)
6 = Troca de comandos via pendrive desativada
7 = Arquivo de comando excede tamanho permitido
8 = Espaço da Memória de Trabalho esgotado
9 = Espaço da MRP esgotado
-1 = Erro da função. Resultado não disponível

Exemplo em VB.NET:

```
Declare Function DLLREP_ObtemCodigoErro Lib "dllrep.dll" (ByVal Handle As Integer, ByVal idLinha As Integer) As Integer
```

Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_ObtemCodigoErro(int Handle, int ID_Linha);
```

Exemplo em Delphi:

```
function DLLREP_ObtemCodigoErro (iHandle: Integer; ID_Linha: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_ObtemMensagemErro()

Exibe o código e a mensagem de erro retornado pelo REP-ZPM.

Declaração:

int DLLREP_ObtemMensagemErro(int **Handle**,
string(256) **MensagemErro**,
int **idLinha**)

Parâmetros de Entrada:

Handle – Handle do REP

idLinha – Se foi executado um comando simples, deve ser igual a 1. No caso de operações múltiplas sobre funcionários (com *DLLREP_Funcionario_Prepara()* ou *DLLREP_BuscaFuncionario_Prepara()*), deve corresponder ao n-ésimo comando.

Parâmetros de Saída:

MensagemErro – Retorna a Mensagem de Erro.

Retorno:

0 = OK, sem erro
1 = Registro mal formado
2 = Valor inválido
3 = Falha durante a operação do comando
4 = Comando inválido
5 = Registro não encontrado (comando de busca)
6 = Troca de comandos via pendrive desativada
7 = Arquivo de comando excede tamanho permitido
8 = Espaço da Memória de Trabalho esgotado
9 = Espaço da MRP esgotado
-1 = Erro da função. Resultado não disponível

Exemplo em VB.NET:

```
Declare Function DLLREP_ObtemMensagemErro Lib "dllrep.dll" (ByVal Handle As Integer, ByVal MensagemErro As String, ByVal idLinha As Integer) As Integer
```

Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_ObtemMensagemErro(int Handle, StringBuilder sMensagemErro, int ID_Linha);
```

Exemplo em Delphi:

```
function DLLREP_ObtemMensagemErro (iHandle: Integer; MensagemErro: String; ID_Linha: Integer): Integer; StdCall;
External 'DLLREP.DLL';
```

7. Funções de Obtenção de Retorno

Estas funções apresentarão resultados para seus respectivos comandos de busca.

➢ DLLREP_TotalRetornos()

Número de linhas retornadas para uma função de busca.

Declaração:

int DLLREP_TotalRetornos(int **Handle**)

Parâmetros de Entrada:

Handle – Handle do REP

Retorno:

Número de linhas disponíveis no Retorno

- Exemplo em VB.NET:

```
Declare Function DLLREP_TotalRetornos Lib "dllrep.dll" (ByVal Handle As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_TotalRetornos(int Handle);
```

- Exemplo em Delphi:

```
function DLLREP_TotalRetornos (iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_RetornoEmpregador()

Recupera informações do empregador retornadas pelo comando *DLLREP_BuscaEmpregador()*.

Declaração:

int DLLREP_RetornoEmpregador(int **Handle**,
string(1) **Tipo**,
string(14) **Identificacao**,
string(12) **CEI**,
string(150) **RazaoSocial**,
string(100) **LocalTrabalho**)

Parâmetros de Entrada:

Handle – Handle do REP

Parâmetros de Saída:

Tipo –

"J": Pessoa Jurídica (CNPJ)

"F": Pessoa Física (CPF)

Identificacao – CNPJ ou CPF

CEI – CEI

RazaoSocial – Razão Social do Empregador

LocalTrabalho – Local de Trabalho

Retorno:

1: OK

-1: Erro

-6: REP retornou Mensagem Erro: Utilize Função *DLLREP_ObtemCodigoErro()* ou *DLLREP_ObtemMensagemErro*, para maiores detalhes.

- Exemplo em VB.NET:

```
Declare Function DLLREP_RetornoEmpregador Lib "dllrep.dll" (ByVal Handle As Integer, ByVal Tipo As String, ByVal Identificacao As String, ByVal CEI As String, ByVal RazaoSocial As String, ByVal LocalTrabalho As String) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_RetornoEmpregador(int Handle, StringBuilder Tipo, StringBuilder Identificacao, StringBuilder CEI, StringBuilder RazaoSocial, StringBuilder LocalTrabalho);
```

- Exemplo em Delphi:

```
function DLLREP_RetornoEmpregador (iHandle: Integer; Tipo: String; Identificacao: String; CEI: String; RazaoSocial: String; LocalTrabalho: String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_RetornoFuncionario()

Recupera informações de um funcionário retornadas pelo comando *DLLREP_BuscaFuncionario()* ou *DLLREP_BuscaTodosFuncionarios().*

Declaração:

```
int DLLREP_RetornoFuncionario(int Handle,
                              int indiceLinha,
                              string(11) PIS,
                              string(20) Matricula,
                              string(52) NomeFunc,
                              string(20000) TemplateBiometrico,
                              string(1) PIS_Teclado,
                              string(16) Codigo_K,
                              string(20) Codigo_B,
                              string(20 Codigo_M),
                              string(20) Codigo_T)
```

Parâmetros de Entrada:

Handle – Handle do REP

indiceLinha – Linha a Recuperar (entre 1 e *DLLREP_TotalRetornos()*)

Parâmetros de Saída:

PIS (string[11]) – Pis do Funcionário

Matricula – Matrícula do Funcionário

Nome – Nome do Empregado

TemplateBiometrico – Lista de Templates Biométricos separados por ponto-e-vírgula (“;”). Limite de 1 Template para Linha R200, e de 10 Templates para demais modelos.

PIS_Teclado – “S”: Permite digitar PIS no teclado.
“N”: Não permite.

Codigo_K – código para ser usado no teclado (keyboard)

Codigo_B – código a ser usado no cartão de barras

Codigo_M – codigo a ser utilizado no MIFARE

Codigo_T – codigo a ser utilizado com o TAG

Retorno:

1: OK

-1: Erro

-6: REP retornou Mensagem Erro: Utilize Função *DLLREP_ObtemCodigoErro()* ou *DLLREP_ObtemMensagemErro*, para maiores detalhes.

-31: Não existe Retorno para o indiceLinha especificado. Este deve ser maior ou igual a *DLLREP_TotalRetornos()*.

- Exemplo em VB.NET:

```
Declare Function DLLREP_RetornoFuncionario Lib "dllrep.dll" (ByVal Handle As Integer, ByVal IndiceLinha As Integer, ByVal PIS As String, ByVal Matricula As String, ByVal NomeFuncionario As String, ByVal TemplateBiometrico As String, ByVal PIS_Teclado As String, ByVal CodigoTeclado As String, ByVal CodigoBarras As String, ByVal CodigoMifare As String, ByVal CodigoTAG As String) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_RetornoFuncionario(int Handle, int IndiceLinha, StringBuilder PIS, StringBuilder Matricula, StringBuilder NomeFuncionario, StringBuilder TemplateBiometrico, StringBuilder PIS_Teclado, StringBuilder CodigoTeclado, StringBuilder CodigoBarras, StringBuilder CodigoMifare, StringBuilder CodigoTAG);
```

- Exemplo em Delphi:

```
function DLLREP_RetornoFuncionario (iHandle: Integer; IndiceLinha: Integer; PIS: String; Matricula: String; NomeFuncionario: String; Biometrico: String; Habilita_Teclado: String; Cod_Teclado: String; Cod_Barras: String; Cod_MIFARE: String; Cod_TAG: String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_RetornoPonto()

Verifica as informações de um ponto retornadas pelo comando de Leitura de Ponto.

Declaração:

```
int DLLREP_RetornoPonto(int Handle,
                        int indiceLinha,
                        string(11) PIS,
                        string(19) DataHora,
                        string(20) NSR)
```

Parâmetros de Entrada:

Handle – Handle do REP

idComando – Identificador único do Comando

indiceLinha – Linha a Recuperar (entre 1 e *DLLREP_TotalRetornos()*)

Parâmetros de Saída:

PIS – Pis do Funcionário

DataHora – Data no formato “DD/MM/YYYY HH:mm:SS”

NSR – Número Seqüencial de Registro

Retorno:

1: OK

-1: Erro

-6: REP retornou Mensagem Erro: Utilize Função *DLLREP_ObtemCodigoErro()* ou *DLLREP_ObtemMensagemErro*, para maiores detalhes.

-31: Não existe Retorno para o indiceLinha especificado. Este deve ser menor ou igual a *DLLREP_TotalRetornos()*.

- Exemplo em VB.NET:

```
Declare Function DLLREP_RetornoPonto Lib "dllrep.dll" (ByVal Handle As Integer, ByVal IndiceLinha As Integer, ByVal PIS As String, ByVal DataHora As String, ByVal NSR As String) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_RetornoPonto(int Handle, int Indicelinha, StringBuilder PIS, StringBuilder DataHora, StringBuilder NSR);
```

- Exemplo em Delphi:

```
function DLLREP_RetornoPonto (iHandle: Integer; ID_Linha: Integer; PIS: String; DataHora: String; NSR: String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_RetornoStatus()

Verifica as informações de status retornadas pelo comando de Leitura de Ponto.

Declaração:

int DLLREP_RetornoStatus(int **Handle**,
string(17) **numFabricacao**,
string(12) **ultimaMarcacaoPIS**,
string(19) **ultimaMarcacaoDataHora**,
string(1) **statusPapel**,
string(20) **DataHora**,
string(20) **memTotalMRP**,
string(20) **memUsoMRP**)

Parâmetros de Entrada:

Handle – Handle do REP

Parâmetros de Saída:

numFabricacao – Número de Fabricação do REP

ultimaMarcacaoPIS – PIS do funcionário que realizou última marcação

ultimaMarcacaoDataHora – Data e Hora da última marcação (DD/MM/YYYY HH:mm:SS)

statusPapel – “0”: Sem papel “1”: Presença de papel “2”: Pouco papel

DataHora – Data e Hora atual do REP

memTotalMRP – memória da MRP em bytes

memUsoMRP – memória usada pelo MRP em bytes

Retorno:

1: OK

-1: Erro

-6: REP retornou Mensagem Erro. Utilize Função *DLLREP_ObtemCodigoErro()* ou, para maiores detalhes

- Exemplo em VB.NET:

```
Declare Function DLLREP_RetornoStatus Lib "dllrep.dll" (ByVal Handle As Integer, ByVal NumFabricacao As String, ByVal UltimaMarcacaoPIS As String, ByVal UltimaMarcacaoDataHora As String, ByVal StatusPapel As String, ByVal DataHora As String, ByVal MemoriaTotalMRP As String, ByVal MemoriaUsoMRP As String) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_RetornoStatus(int Handle, StringBuilder NumFabricacao, StringBuilder UltimaMarcacaoPIS, StringBuilder UltimaMarcacaodataHora, StringBuilder StatusPapel, StringBuilder DataHora, StringBuilder MemoriaTotalMRP, StringBuilder MemoriaUsoMRP);
```

- Exemplo em Delphi:

```
function DLLREP_RetornoStatus (iHandle: Integer; NumeroFabricacao: String; UltimaMarcacaoPIS: String; UltimaMarcacaoDataHora: String; StatusPapel: String; DataHora: String; MemTotalMRP: String; MemUsoMRP: String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_RetornoStatusCompleto()

Verifica as informações de status retornadas pelo comando de *BuscaStatusCompleto()*.

OBS.: Este comando está disponível nos equipamentos:

- Família R100 a partir da versão 01.40.28 do SB;
- Família R200 a partir da versão 03.00.19 do SB;
- Família R300 a partir da versão 02.01.00 do SB.

Declaração:

```
int DLLREP_RetornoStatusCompleto(int Handle,
                                 string(17) numFabricacao,
                                 string(12) ultimaMarcacaoPIS,
                                 string(19) ultimaMarcacaoDataHora,
                                 string(1) StatusPapel,
                                 string(20) DataHora,
                                 string(20) memTotalMRP,
                                 string(20) memUsoMRP,
                                 string(20) repVersao,
                                 string(2) Cutter,
                                 string(20) mtVersao,
                                 string(10) empCad,
                                 string(10) empCadMax,
                                 string(10) NSRAtual,
                                 string(10) NSRMax,
                                 string(2) bioTipo,
                                 string(10) bioVersao,
                                 string(3) bioSeg,
                                 string(10) bioCad,
                                 string(10) bioMax,
                                 string(20) mrpVersao,
                                 string(1) redeDHCP,
                                 string(16) redeIP,
                                 string(16) redeGateway,
                                 string(16) redeMask,
```

string(10) **redePort**,
string(24) **redeMAC**,
string(1) **HorarioVerao**,
string (10) **HorarioVeraoInicio**,
string(10) **HorarioVeraoFim**.

Parâmetros de Entrada:

Handle – Handle do REP

Parâmetros de Saída:

numFabricacao – Número de Fabricação do REP

ultimaMarcacaoPIS – PIS do funcionário que realizou última marcação

ultimaMarcacaoDataHora – Data e Hora da última marcação (DD/MM/YYYY HH:mm:SS)

statusPapel –

0 **–** Sem papel
1 **–** Presença de papel
2 **–** Pouco papel

DataHora – Data e Hora atual do REP

memTotaMRP – Memória da MRP em bytes

memUsoMRP – Memória usada pelo MRP em bytes

repVersao – Versão do firmware do equipamento

Cutter – Nível que o cutter está programado, apenas para o REP.

mtVersao – Versão da memória de trabalho.

empCad – Quantidade de empregados cadastrados na memória de trabalho.

empCadMax – Quantidade máxima de empregados que podem ser cadastrados na MT.

NSRAtual – Número sequencial atual no REP.

NSRMax – Número máximo de NSR.

bioTipo – Tipo de leitor biométrico do equipamento:

0 – Nenhum;
1 – Fingerprint;
2 – Suprema;
3 – Nitgen;
4 – CAMA;
5 – Sagem.

bioVersao – Versão do firmware do leitor biométrico

bioSeg – Nível de segurança do leitor biométrico

bioCad – Quantidade de digitais cadastradas no leitor biométrico

bioMax – Quantidade máxima de digitais que o equipamento suporta

mrpVersao – Versao do firmware da MRP

redeDHCP – Habilitação do DHCP.

0 – Desabilitado;
1 – Habilitado.

redeIP – Endereço IP do equipamento.

redeGateway – Configuração do gateway do equipamento.

redeMask – Configuração da máscara de rede do equipamento.

redePort – Porta de comunicação do equipamento.

redeMAC – Endereço MAC do equipamento.

HorarioVerao – Habilitação do horário de verão.

0 – Desabilitado;
1 – Habilitado.

HorarioVeraoInicio – Início do horário de verão.
0 – Não configurado. DD/MM/AAAA
Data programada para entrar no horário de verão.

HorarioVeraoFim – Final do horário de verão.
0 – Não configurado. DD/MM/AAAA
Data programada para sair do horário de verão.

Retorno:

1: OK

-1: Erro

-6: REP retornou Mensagem Erro: Utilize Função *DLLREP_ObtemCodigoErro()* ou *DLLREP_ObtemMensagemErro*, para maiores detalhes.

Exemplo em VB.NET:

```
Declare Function DLLREP_RetornoStatusCompleto Lib "dllrep.dll" (ByVal Handle As Integer, ByVal NumFabricacao As String, ByVal UltimaMarcacaoPIS As String, ByVal UltimaMarcacaoDataHora As String, ByVal StatusPapel As String, ByVal DataHora As String, ByVal MemoriaTotalMRP As String, ByVal MemoriaUsoMRP As String, ByVal REPVersao As String, ByVal Cutter As String, ByVal MTVersao As String, ByVal EmpCad As String, ByVal EmpCadMax As String, , ByVal NSRAtual As String, ByVal NSRMax As String, ByVal BioTipo As String, ByVal BioVersao As String, ByVal BioSeg As String, ByVal BioCad As String, ByVal BioMax As String, ByVal MRPVersao As String, ByVal RedeDHCP As String, ByVal RedeIP As String, ByVal RedeGateway As String, ByVal RedeMask As String, ByVal RedePort As String, ByVal RedeMAC As String, ByVal HorarioVerao As String, ByVal HorarioVeraoInicio As String, ByVal HorarioVeraoFim As String) As Integer
```

Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_RetornoStatus(int Handle, StringBuilder NumFabricacao, StringBuilder UltimaMarcacaoPIS, StringBuilder UltimaMarcacaodataHora, StringBuilder StatusPapel, StringBuilder DataHora, StringBuilder MemoriaTotalMRP, StringBuilder MemoriaUsoMRP, StringBuilder REPVersao, StringBuilder Cutter, StringBuilder MTVersao, StringBuilder EmpCad, StringBuilder EmpCadMax, StringBuilder NSRAtual, StringBuilder NSRMax, StringBuilder BioTipo, StringBuilder BioVersao, StringBuilder BioSeg, StringBuilder BioCad, StringBuilder BioMax, StringBuilder MRPVersao, StringBuilder RedeDHCP, StringBuilder RedeIP, StringBuilder RedeGateway, StringBuilder RedeMask, StringBuilder RedePort, StringBuilder RedeMAC, StringBuilder HorarioVerao, StringBuilder HorarioVeraoInicio, StringBuilder HorarioVeraoFim);
```

Exemplo em Delphi:

```
function DLLREP_RetornoStatus (iHandle: Integer; NumeroFabricacao: String; UltimaMarcacaoPIS: String; UltimaMarcacaoDataHora: String; StatusPapel: String; DataHora: String; MemTotalMRP: String; MemUsoMRP: String; REPVersao: String; Cutter: String; MTVersao: String; EmpCad: String; EmpCadMax: String; NSRAtual: String; NSRMax: String; BioTipo: String; BioVersao: String; BioSeg: String; BioCad: String; BioMax: String; MRPVersao: String; RedeDHCP: String; RedeIP: String; RedeGateway: String; RedeMask: String; RedePort: String; RedeMAC: String; HorarioVerao: String; HorarioVeraoInicio: String; HorarioVeraoFim: String): Integer; StdCall; External 'DLLREP.DLL';
```

DLLREP_RetornoLeituraMRP()

Verifica as informações de um ponto retornadas pelo comando de Leitura de MRP. Por ser uma leitura “RAW” de registros, este comando não permite as funções.

Declaração:

```
int DLLREP_RetornoLeituraMRP(int Handle,
                             int indiceLinha,
                             string(300) RegistroAFD)
```

Parâmetros de Entrada:

Handle – Handle do REP

indiceLinha – Linha a Recuperar (entre 1 e *DLLREP_TotalRetornos()*)

Parâmetros de Saída:

RegistroAFD – Registro AFD

Retorno:

1: OK

-1: Erro

-6: REP retornou Mensagem Erro. Utilize, para maiores detalhes

-31: Não existe Retorno para o indiceLinha especificado. Este deve ser menor ou igual a *DLLREP_TotalRetornos().*

Exemplo em VB.NET:

```
Declare Function DLLREP_RetornoLeituraMRP Lib "dllrep.dll" (ByVal Handle As Integer, ByVal IndiceLinha As Integer, ByVal RegistroAFD As String) As Integer
```

Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_RetornoLeituraMRP(int Handle, int IndiceLinha, StringBuilder RegistroAFD);
```

Exemplo em Delphi:

```
function DLLREP_RetornoLeituraMRP (iHandle: Integer; indiceLinha: Integer; RegistroAFD: String): Integer; StdCall; External 'DLLREP.DLL';
```

DLLREP_RetornoHorarioVerao()

Retorna a configuração do Horário de Verão solicitada por *DLLREP_BuscaHorarioVerao* ()*).*

Declaração:

```
int DLLREP_RetornoHorarioVerao(int handle,
                               string(10) DataInicio,
                               string(10) DataFim)
```

Parâmetros de Entrada:

Handle – Handle do REP

Parâmetros de Saída:

DataInicio – Data de Início da vigência do Horário de Verão (DD/MM/YYYY) ou 0 caso esteja indefinido

DataInicio – Data do Início da vigência do Horário de Verão (DD/MM/YYYY), ou 0 caso esteja indefinido

Retorno:

1: OK

-1: Erro

-6: REP retornou Mensagem Erro: Utilize Função *DLLREP_ObtemCodigoErro()* ou *DLLREP_ObtemMensagemErro*, para maiores detalhes.

- Exemplo em VB.NET:

```
Declare Function DLLREP_RetornoHorarioVerao Lib "dllrep.dll" (ByVal Handle As Integer, ByVal DataInicio As String, ByVal DataFim As String) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_RetornoHorarioVerao(int Handle, StringBuilder DataInicio, StringBuilder DataFim);
```

- Exemplo em Delphi:

```
function DLLREP_RetornoHorarioVerao (iHandle: Integer; DatInicio: String; DataFim: String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_RetornoReinicializaSenhas()

Verifica se as senhas foram reinicializadas com sucesso por *DLLREP_BuscaHorarioVerao ()*.

OBS.: Este comando está disponível nos equipamentos:

- Família R100 a partir da versão 01.40.31 do SB;
- Família R200 a partir da versão 03.00.19 do SB;
- Família R300 a partir da versão 02.01.00 do SB.

Declaração:

int DLLREP_RetornoReinicializaSenhas(int **Handle**)

Parâmetros de Entrada:

Handle – Handle do REP

Retorno:

1: OK

-1: Erro

-6: REP retornou Mensagem Erro: Utilize Função *DLLREP_ObtemCodigoErro()* ou *DLLREP_ObtemMensagemErro*, para maiores detalhes.

- Exemplo em VB.NET:

```
Declare Function DLLREP_RetornoReinicializaSenhas Lib "dllrep.dll" (ByVal Handle As Integer) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_RetornoReinicializaSenhas(int Handle);
```

- Exemplo em Delphi:

```
function DLLREP_RetornoReinicializaSenhas (iHandle: Integer): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_RetornoUltimoEvento()

Recupera o último evento enviado pelo REP.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 02.01.00 do SB.

Declaração:

int DLLREP_RetornoUltimoEvento(int **Handle**,
string (20) **datahora**,
string(17) **numFabricacao**,
string(4) **modelo**,
string(2) **imp1Ativada**,
string(2) **imp1SemPapel**,
string(2) **imp1CabecaQuente**,
string(2) **imp1Tampa**,
string(10) **imp1Nivel**,
string(10) **imp1TicketsRestantes**,
string(2) **imp1NivelPoucoPapel**,
string(2) **imp2Ativada**,
string(2) **imp2SemPapel**,
string(2) **imp2CabecaQuente**,
string(2) **imp2Tampa**,
string(10) **imp2Nivel**,
string(10) **imp2TicketsRestantes**,
string(2) **imp2NivelPoucoPapel**,
string(2) **batBaixa**,
string(3) **batNivel**,
string(2) **energiaFalta**,
string(2) **energiaRetorno**,
string(2) **pendriveFisco**,
string(2) **pendrive_ComCmdUsuario**,
string(2) **pendrive_SemCmdUsuario**,
string(2)**intrusao**,
string(2) **batTampaAberta**,
string(2) **extraindoRIM**,
string(2) **menuAdmin**,
string(2)**menuSup**)

Parâmetros de Entrada:

Handle – Handle do REP

Parâmetros de Saída:

datahora – Momento em que foi recebido o evento pela DLL
numFabricacao – Número de Série do Equipamento
modelo - Modelo do Equipamento
imp1Ativada – Impressora 1 ativada
imp1SemPapel – Impressora 1 Sem Papel
imp1CabecaQuente – Impressora 1 com Cabeça Quente
imp1Tampa – Impressora 1 com Tampa Aberta
imp1Nivel – Nível do papel da Impressora 1 (em metros)
imp1TicketsRestantes – Quantidade aproximada de Tickets restantes na impressora 1
imp1NivelPoucoPapel – Nível de pouco papel configurado para a impressora 1(em metros).
imp2Ativada – Impressora 2 ativada
imp2SemPapel – Impressora 2 Sem Papel
imp2CabecaQuente - Impressora 2 com Cabeça Quente
imp2Tampa – Impressora 2 com Tampa Aberta

imp2Nivel – Nível do papel da Impressora 2 (em metros)
imp2TicketsRestantes – Quantidade aproximada de Tickets restantes na impressora 2
imp2NivelPoucoPapel – Nível de pouco papel configurado para a impressora 2(em metros).
batBaixa – Nível Crítico da Bateria
batNivel – Nível da Bateria: 0 (mínimo) a 5 (máximo). 100 – Carregando
energiaFalta – Falta de energia elétrica
energiaRetorno – Retorno de Energia Elétrica
pendriveFisco – Pendrive inserido na porta do Fisco
pendrive_ComCmdUsuario – Pendrive inserido na porta do usuário – com Comando
pendrive_SemCmdUsuario – Pendrive inserido na porta do usuário - sem comando
intrusão – Intrusão detectada
batTampaAberta – Tampa da Bateria Aberta
extraindoRIM – Extraindo RIM
menuAdmin – REP no Menu Administrador
menuSup – REP no Menu Supervisor

Retorno:

>0 – ID sequencial do último evento recebido
-1 – Não há evento para ser retornado

Observação: O valor "-1" para qualquer parâmetro de saída sinaliza que o valor não está disponível ou não se aplica.

- Exemplo em VB.NET:

```
Declare Function DLLREP_RetornoUltimoEvento Lib "dllrep.dll" (ByVal Handle As Integer, ByVal DataHora As String, ByVal NumFabricacao As String, ByVal Modelo As String, ByVal Imp1Ativada As String, ByVal Imp1SemPapel As String, ByVal Imp1CabecaQuente As String, ByVal Imp1Tampa As String, ByVal Imp1Nivel As String, ByVal Imp1TicketsRestantes As String, ByVal Imp1PoucoPapel As String, ByVal Imp2Ativada As String, ByVal Imp2SemPapel As String, ByVal Imp2CabecaQuente As String, ByVal Imp2Tampa As String, ByVal Imp2Nivel As String, ByVal Imp2TicketsRestantes As String, ByVal Imp2NivelPoucoPapel As String, ByVal BatBaixa As String, ByVal BatNivel As String, ByVal EnergiaFalta As String, ByVal EnergiaRetorno As String, ByVal PendriveFisco As String, ByVal PendriveComCmdUsuario As String, ByVal PendriveSemCmdUsuario As String, ByVal Intrusao As String, ByVal BatTampaAberta As String, ByVal ExtraindoRIM As String, ByVal MenuAdmin As String, ByVal MenuSup As String) As Integer
```

- Exemplo em C#:

```
[DllImport("dllrep.dll")] public static extern int DLLREP_RetornoUltimoEvento (int Handle, StringBuilder DataHora, StringBuilder NumFabricacao, StringBuilder Modelo, StringBuilder Imp1Ativada, StringBuilder Imp1SemPapel, StringBuilder Imp1CabecaQuente, StringBuilder Imp1Tampa, StringBuilder Imp1Nivel, StringBuilder Imp1TicketsRestantes, StringBuilder Imp1PoucoPapel, StringBuilder Imp2Ativada, StringBuilder Imp2SemPapel, StringBuilder Imp2CabecaQuente, StringBuilder Imp2Tampa, StringBuilder Imp2Nivel, StringBuilder Imp2TicketsRestantes, StringBuilder Imp2NivelPoucoPapel, StringBuilder BatBaixa, StringBuilder BatNivel, StringBuilder EnergiaFalta, StringBuilder EnergiaRetorno, StringBuilder PendriveFisco, StringBuilder PendriveComCmdUsuario, StringBuilder PendriveSemCmdUsuario, StringBuilder Intrusao, StringBuilder BatTampaAberta, StringBuilder ExtraindoRIM, StringBuilder MenuAdmin, StringBuilder MenuSup);
```

- Exemplo em Delphi:

```
function DLLREP_RetornoUltimoEvento (iHandle: Integer; DataHora: String; NumFabricacao: String; Modelo: String; Imp1Ativada: String; Imp1SemPapel: String; Imp1CabecaQuente: String; Imp1Tampa: String; Imp1Nivel: String; Imp1TicketsRestantes: String; Imp1PoucoPapel: String; Imp2Ativada: String; Imp2SemPapel: String; Imp2CabecaQuente: String; Imp2Tampa: String; Imp2Nivel: String; Imp2TicketsRestantes: String; Imp2NivelPoucoPapel: String; BatBaixa: String; BatNivel: String; EnergiaFalta: String; EnergiaRetorno: String; PendriveFisco: String; PendriveComCmdUsuario: String; PendriveSemCmdUsuario: String; Intrusao: String; BatTampaAberta: String; ExtraindoRIM: String; MenuAdmin: String; MenuSup: String): Integer; StdCall; External 'DLLREP.DLL';
```

➢ DLLREP_RetornoMascaraBarras()

Recupera as informações da máscara do código de barras do equipamento.
Este comando esta funcional apenas por comunicação via rede.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 04.00.08 do SB.

Declaração:

```
int DLLREP_RetornoMascaraBarras (int Handle,
                                 char *Digito01,
                                 char *Digito02,
                                 char *Digito03,
                                 char *Digito04,
                                 char *Digito05,
                                 char *Digito06,
                                 char *Digito07,
                                 char *Digito08,
                                 char *Digito09,
                                 char *Digito10,
                                 char *Digito11,
                                 char *Digito12,
                                 char *Digito13,
                                 char *Digito14,
                                 char *Digito15,
                                 char *Digito16,
                                 char *Digito17,
                                 char *Digito18,
                                 char *Digito19,
                                 char *Digito20);
```

Parâmetros:

Handle – Handle do REP
Digito01 – Primeiro caractere do código de barras lido pelo equipamento.
Digito02 – Segundo caractere do código de barras lido pelo equipamento.
Digito03 – Terceiro caractere do código de barras lido pelo equipamento.
Digito04 – Quarto caractere do código de barras lido pelo equipamento.
Digito05 – Quinto caractere do código de barras lido pelo equipamento.
Digito06 – Sexto caractere do código de barras lido pelo equipamento.
Digito07 – Sétimo caractere do código de barras lido pelo equipamento.
Digito08 – Oitavo caractere do código de barras lido pelo equipamento.
Digito09 – Nono caractere do código de barras lido pelo equipamento.
Digito10 – Décimo caractere do código de barras lido pelo equipamento.
Digito11 – Décimo primeiro caractere do código de barras lido pelo equipamento.
Digito12 – Décimo segundo caractere do código de barras lido pelo equipamento.
Digito13 – Décimo terceiro caractere do código de barras lido pelo equipamento.
Digito14 – Décimo quarto caractere do código de barras lido pelo equipamento.
Digito15 – Décimo quinto caractere do código de barras lido pelo equipamento.
Digito16 – Décimo sexto caractere do código de barras lido pelo equipamento.
Digito17 – Décimo sétimo caractere do código de barras lido pelo equipamento.
Digito18 – Décimo oitavo caractere do código de barras lido pelo equipamento.
Digito19 – Décimo nono caractere do código de barras lido pelo equipamento.
Digito20 – Vigésimo caractere do código de barras lido pelo equipamento.

Retorno:

1: Sucesso
-1: Handle inválido
-6: REP retornou Mensagem Erro: Utilize Função *DLLREP_ObtemCodigoErro()* ou *DLLREP_ObtemMensagemErro*, para maiores detalhes.
-30: Erro no comando.

➢ DLLREP_RetornoUltimosRegistros()

Recupera a informação dos últimos registros da MRP.
Este comando esta funcional apenas por comunicação via rede.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 04.00.08 do SB.

Declaração:

int DLLREP_RetornoUltimosRegistros (int **Handle**,
char ***identificadorReg1**,
char ***identificadorReg2**,
char ***identificadorReg3**,
char ***identificadorReg4**,
char ***identificadorReg5**,
char ***identificadorReg6**,
char ***identificadorReg7**,
char ***identificadorReg8**,
char ***identificadorReg9**,
char ***identificadorReg10**,
char ***nsrReg1**,
char ***nsrReg2**,
char ***nsrReg3**,
char ***nsrReg4**,
char ***nsrReg5**,
char ***nsrReg6**,
char ***nsrReg7**,
char ***nsrReg8**,
char ***nsrReg9**,
char ***nsrReg10**,
char ***tipoReg1**,
char ***tipoReg2**,
char ***tipoReg3**,
char ***tipoReg4**,
char ***tipoReg5**,
char ***tipoReg6**,
char ***tipoReg7**,
char ***tipoReg8**,
char ***tipoReg9**,
char ***tipoReg10**,
char ***operacaoReg1**,
char ***operacaoReg2**,
char ***operacaoReg3**,
char ***operacaoReg4**,
char ***operacaoReg5**,
char ***operacaoReg6**,
char ***operacaoReg7**,
char ***operacaoReg8**,
char ***operacaoReg9**,
char ***operacaoReg10**,
char ***dataHoraReg1**,
char ***dataHoraReg2**,
char ***dataHoraReg3**,
char ***dataHoraReg4**,

char *dataHoraReg5,
char *dataHoraReg6,
char *dataHoraReg7,
char *dataHoraReg8,
char *dataHoraReg9,
char *dataHoraReg10);

Parâmetros de Entrada:

Handle – Handle do REP

identificadorReg – Identificador do registro (PIS, PASEP, CPF ou CNPJ) valor numérico.
nsrReg – NSR do registro

tipoReg – Tipo do registro

50: Empregador

51: Ponto

52: Ajuste de relógio

53: Manipulação de empregado

operacaoReg – Operação do registro

48: Sem descrição

65: Alteração

69: Exclusão

73: Inclusão

dataHoraReg – Data e hora do registro (DD/MM/AAAA hh:mm:ss)

Retorno:

1: Sucesso
-1: Handle inválido
-6: REP retornou Mensagem Erro: Utilize Função *DLLREP_ObtemCodigoErro()* ou *DLLREP_ObtemMensagemErro*, para maiores detalhes.
-30: Erro no comando.

➢ DLLREP_RetornoModoTAG()

Recupera a informação da configuração do TAG no equipamento.
Este comando esta funcional apenas por comunicação via rede.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 04.00.08 do SB.

Declaração:

int DLLREP_RetornoModoTAG (int **Handle**,
char *__Modo__);

Parâmetros de Entrada:

Handle – Handle do REP

Modo – Informação do modo de leitura TAG configurado

1: Decimal

2: Hexadecimal

Retorno:

1: Sucesso
-1: Handle inválido
-6: REP retornou Mensagem Erro: Utilize Função *DLLREP_ObtemCodigoErro()* ou *DLLREP_ObtemMensagemErro*, para maiores detalhes.
-30: Erro no comando.

➢ DLLREP_RetornoCriptoBarras()

Recupera informações da criptografia configurada no equipamento.
Este comando esta funcional apenas por comunicação via rede.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 04.00.08 do SB.

Declaração:

int DLLREP_RetornoCriptoBarras (int **Handle**,
char ***Cripto**);

Parâmetros de Entrada:

Handle – Handle do REP

Cripto – Status da utilização de criptografia do código de barras pelo equipamento
0: Sem criptografia
1: Com criptografia

Retorno:

1: Sucesso
-1: Handle inválido
-6: REP retornou Mensagem Erro: Utilize Função *DLLREP_ObtemCodigoErro()* ou *DLLREP_ObtemMensagemErro*, para maiores detalhes.
-30: Erro no comando.

➢ DLLREP_RetornoModoMifare()

Recupera a informação da configuração do Mifare no equipamento.

OBS.: Este comando está disponível apenas nos equipamentos da família R300 a partir da versão 04.00.18 do SB.

Declaração:

int DLLREP_RetornoModoMifare (int **Handle**,
char ***Modo**);

Parâmetros de Entrada:

Handle – Handle do REP

Modo – Informação do modo de leitura Mifare configurado
1: Little-endian
2: Big-endian

Retorno:

1: Sucesso
-1: Handle inválido
-6: REP retornou Mensagem Erro: Utilize Função *DLLREP_ObtemCodigoErro()* ou *DLLREP_ObtemMensagemErro*, para maiores detalhes.
-30: Erro no comando.

Exemplos de Uso

➢ Seqüência de chamadas para Leitura de Ponto em um dado período.

Modo IP (Rede)

```
IniciaDriver
DefineModoIP
BuscaPonto(DataInicio,DataFim)
ObtemCodigoErro>0? Erro
para i=1 até TotalRetornos
  BuscaPontoRetorno(i)
EncerraDriver
```

Modo Arquivo (PenDrive)

```
IniciaDriver
DefineModoArquivo
BuscaPonto(DataInicio,DataFim)
enquanto (VerificaRetornoPenDrive<0)  {}
ObtemCodigoErro>0? Erro
for i=1 to TotalRetornos
  BuscaPontoRetorno(i)
EncerraDriver
```

➢ Seqüência de chamadas para cadastrar Empregador no REP.

Modo IP (Rede)

```
IniciaDriver
DefineModoIP
Empregador
ObtemCodigoErro>0
  então erro no cadastro
EncerraDriver
```

Modo Arquivo (PenDrive)

```
IniciaDriver
DefineModoArquivo
Empregador
enquanto VerificaRetornoPenDrive<0  {}
ObtemCodigoErro>0
  então erro no cadastro
EncerraDriver
```

➢ Seqüência de chamadas para inserir 2 funcionários no REP.

Modo IP (Rede)

```
IniciaDriver
DefineModoIP
i1=Funcionario_Prepara
i2=Funcionario_Prepara
Funcionario_Envia
ObtemCodigoErro(i1)>0
  então erro na primeira inserção
ObtemCodigoErro(i2)>0
  então erro na segunda inserção
EncerraDriver
```

Modo Arquivo (PenDrive)

```
IniciaDriver
DefineModoArquivo
i1=Funcionario_Prepara
i2=Funcionario_Prepara
Funcionario_Envia
enquanto (VerificaRetornoPenDrive<0)  {}
ObtemCodigoErro(i1)>0
  então erro na primeira inserção
ObtemCodigoErro(i2)>0
  então erro na segunda inserção
EncerraDriver
```

➢ Seqüência de chamadas para ler eventos enviados pelo REP

Somente Modo IP (Rede)

```
IniciaDriver()
DefineModoIP() ou DefineModoIPCriptografado()
IniciaLeituraEventos()
eventoId=-1
enquanto (não encerraLeitura)
  {
  eventoId_Old=eventoId
  eventoId=RetornaUltimoEvento()
  if (eventoId>eventoId_old)
     print("Novo Evento "+eventoId)
  }
EncerraLeituraEventos()
FinalizaDriver()
```