**ActiveX Library Kernel7x.dll**

**Interface kernel**
**Interface alternativo - for visual basic 6, visual basic .net, c#**
Activex Middleware for Device 7x Multi-threads
Support Serial 232-485/Tcpip/Usb/Modem/GPRS
Biometric Algorithm Matching

**Interface hamster**
Activex for Hamster Management

---

# Programmer Advanced Guide

SDK revision 7.2.0.29

# Sumário

# Introdução

O ActiveX COM+ .NET kernel7x oferece suporte total a comunicação e configuração de equipamentos da família 7x I, II, III, V, VI e Família IP. Suporta conexão Serial 232-485/ TcpIP / USB / Modem / GPRS utilizando tipos de dados e métodos para simples integração.

Além do gerenciamento dos equipamentos Henry, este ActiveX também oferece suporte a captura facilitada de digitais a partir de scanner biométrico.

Este Activex possui várias interfaces, são elas:

Kernel – Interface para controle de equipamentos da geração 7x

Alternativo – Interface semelhante ao Kernel, porém desenvolvida para integradores .NET, utiliza apenas variáveis de tipo primitivo nas suas funções .

Hamster – Interface especializada em captura de digitais via leitores biométricos.

Ao receber este pacote SDK você terá acesso a exemplos de utilização da dll. Antes de iniciar a integração entenda qual a sua necessidade e direcione o desenvolvimento, isso tornará a integração mais rápida e fácil.

Esta documentação possui explicações sobre os métodos das interfaces. No caso da interface kernel/alternativo estão descritos o método original e, se existir, a referência no alternativo.

Para dúvidas ou sugestões acesse o espaço 7x Interativo na internet.

http://www.henry.com.br/pt/contato.htm

# Alterações

### Versão 7.2.0.29

- Implementada validação do número seri al do equipamento.
- Adicionado suporte para configuração do timeout de processamento online do bilhete.

### Versão 7.2.0.28

- Otimizações para coleta de eventos.

### Versão 7.2.0.27

- Implementado suporte para equipamentos Orion 6. Funções de gerenciamento de empregador, usuários do equipamento e coleta dos registros foram implementadas.

### Versão 7.2.0.26

- Implementado suporte para status de porta forçada.
- Revisado envio de broadcast para equipamentos.
- Correção na adição de equipamento GPRS para o Kernel Alternati vo.
- Implementado tempo configurável para reinicialização da porta GPRS através da função “setConnResetTimeout”.
- Propriedade “setConectado” configurada como padrão = verdadeiro.
- Otimização no evento “OnRegistro”.
- Implementado suporte para configurações de concentradora no Kernel Alternativo.

### Versão 7.2.0.25

- Correção no envio de particionamento para cartucho USB .
- Implementada rotina para receber versão de firmware do equipamento.

### Versão 7.2.0.24

- Otimização na velocidade da comunicação.
- Implementado suporte ao recurso de resposta por status.
- Implementado propriedade para verificar qualidade da digital cadastrada.
- Implementada função para verificação de particionamento da memória do equipamento com cartucho com mais de 1 memória.

- Implementada proteção contra envio e recepção de digitais corrompidas.

### Versão 7.2.0.23

- Reestruturação interna no envio e recebimento de informações.
- Reestruturação interna nas rotinas de biometria.
- Incrementada segurança durante comunicação OnOff.
- Incrementada a estabilidade da comunicação.
- Suporte ao modo de comunicação via GPRS
- Inserida proteção e validações no envio de configurações USB
- Suporte ao recurso Faixas de Acesso
- Implementada funcionalidade de captura de imagem

### Versão 7.2.0.22

- Reestruturação da coleta de registros via USB
- Correções e implementações de rotinas no Kernel Alternativo ( ver anexos)

### Versão 7.2.0.21

- Otimização da recuperação de registros via USB
- Otimização na comunicação TCP/IP e Serial
- Implementação interna para limpeza da lista de acesso antes de novo envio

### Versão 7.2.0.18

- Correção interna na recuperação de registros via USB

### Versão 7.2.0.17

- Correção interna na comunicação Serial 485
- Correção na utilização do novo driver USB
- Otimização do gerenciamento de sincronização de catraca

# Funções utilitárias

Em laranja métodos alternativos.
Quando em modo alternativo nos métodos com retorno direto de valor utilize ThreadLastError para consultar sucesso na operação.

- **ListaPortasSeriais: WideString;**

  Esta propriedade retorna uma string com as portas COM encontradas na máquina, por exemplo para COM1 e COM2 retorna: "COM1-COM2".

  **ListaPortasSeriais : WideString ;**

- **USB_Remove: WordBool;**

  Este método desativa o cartucho USB permitindo remoção segura do computador.

  **USB_Remove : WordBool;**

- **EnviaBeep(pThreadIndex: Integer; pBeep: SBeep): WordBool;**

  Este método dispara um alerta sonoro no equipamento, os valor possível de SBeep são cbLiberado, cbNegado.

  **EnviaBeep(pThreadIndex: Integer; pBeep: SB eep): WordBool;**

- **Versao: WideString;**

  Este método retorna a versão da biblioteca ActiveX em uso.

  **Versao: WideString;**

- **DetectarVelocidade(pThreadIndex: Integer; out pVelocidade: SVelocidade): WordBool;**

  Este método determina e retorna a velocidade da comunic ação serial.

  **DetectarVelocidade(pThreadIndex: Integer ): SVelocidade ;**

- **OnProgresso(pThreadIndex: Integer; pByte: Integer; pByteMax: Integer; pBuffer: Integer; pBufferMax: Integer);**

  Este evento permite monitoramento das transferências de envio e recepção de dados. Útil para apresentar uma interface ao usuário.
  Comunicações USB ainda não suportam este recurso.

**OnProgresso(pThreadIndex: Integer; pByte: Integer; pByteMax: Integer; pBuffer: Integer; pBufferMax: Integer);**

- **NumDigitosValidos(pConfiguracao: SConfiguracao): Byte;**

Executa os cálculos necessários baseados em diversos parâmetros da configuração para retornar a quantidade de dígitos válidos da matrícula em uso pelo equipamento.

**NumDigitosValidos: Byte;**

Na interface alternativa este método não requer parâmetros porém utiliza a última configuração carrega como referencia, veja envia e rece be configurações para maiores detalhes.

- **TamanhoRegistro(pConfiguracao: SConfiguracao): Byte;**

Executa os cálculos necessários baseados em diversos parâmetros da config uração para retornar a o tamanho de cada registro na memória do equipamento.

**TamanhoRegistro: Byte;**

Na interface alternativa este método não requer parâmetros porém utiliza a última configuração carrega como referencia, veja envia e rece be configurações para maiores detalhes.

- **NumDigitosPadraoT(pConfiguracao: SConfiguracao): Byte;**

Retorna número de dígitos de acordo com arquivo de configuração HenryT.dat que deve ser indicado dentro da configuração.

**NumDigitosPadraoT: Byte;**

- **SetorPercentual(pParticionamento: SParticionamento; pSetor: SParticao): Double;**

Este método retorna o percentual ocupado por um determinado setor na memória. Vide EnviaParticionamento.

**SetorPercentual(pFuncoes, pFeriados, pAcionamentos, pListaAcesso, pPeriodos, pHorarios, pMsgEspecif ica, pRegistros : Word; pSetor: SParticao): Double;**

- **SetorPercentualEx(pParticionamento: SParticionamento; pSetor: SParticao; pExpansao: SExpansao): Double;**

Este método retorna o percentual ocupado por um determinado setor na memória. O parâmetro “pExpansão“ indica quantas memórias existem no cartucho. Configure o valor 0 (emb2) para cartucho com 2 memórias e o valor 1 (emb8) para cartucho com 8 memórias. Vide EnviaParticionamento.

**SetorPercentual(pFuncoes, pFeriados, pAcionamentos, pListaAcesso, pPeriodos, pHorarios, pMsgEspecifica, pRegistros : Word ; pSetor: SParticao; pExpansao: SExpansao): Double;**

- **BeginLargeTranfer(pThreadIndex : Integer);**

Informa a thread que será iniciada uma operação offline com diversas rajadas de comunicação e a conexão pode ser mantida aberta neste período. Este método oferece maior performance para tais operações. Exemplos de uso: Coleta por pacotes, manutenção de digitais.

**BeginLargeTranfer(pThreadIndex : Integer);**

- **EndLargeTranfer(pThreadIndex : Integer);**

Finaliza sessão de comunicação do BeginLargeTranfer.

**EndLargeTranfer(pThreadIndex : Integer);**

- **DigitosRange(pPlaca :SPlacaCard; pMinimo : WordBool);**

Retorna mínimo ou máximo de dígitos suportados pelo modelo.

**DigitosRange(pPlaca : SPlacaCard; pMinimo : WordBool );**

- **MostRecentFirmware(pConfig: SConfiguracao): WideString; safecall;**

Este método retorna a versão mais atual do firmware existente para o equipamento em uso.

**MostRecentFirmware(pPlacaCard: SPlacaCard; pOrion: WordBool) : WideString;**

# Utilização básica

- **AdicionaCard(pConfig: SComConfig; out pThreadIndex: Integer): WordBool;**

Este método adiciona uma thread de comunicação ao Kernel.
A Estrutura SComConfig define parâmetros desta thread.
O retorno >= 0 representa sucesso ao instanciar a thread de comunicação.

## Estrutura SComConfig

| Atributo | Tipo | Descrição |
|---|---|---|
| Tcp | SComTcp | Estrutura de parâmetros para comunicação TcpIp |
| Serial | SComSerial | Estrutura de parâmetros para comunicação Serial |
| Modem | SComModem | Estrutura de parâmetros para comunicação Modem |
| ModoComunicacao | SModoComunicacao | Configuração do modo de comunicação com o equipamento, dentro da enumeração:<br>**cmcOffline :** Equipamento toma decisão de ponto e acesso sem necessidade do computador e armazena marcações de acordo com a configuração.<br>**cmcOnline :** Equipamento solicita decisão exclusivamente ao computador, marcações são armazenadas no computador em tempo real.<br>**cmcOnOff:** Modo online quando computador disponível, e offline livre para todos quando computador indisponível.<br>**cmcOnOffCtrl:** Trabalha como onoff porém controla acesso em offline segundo lista. |
| TipoComunicacao | STipoComunicacao | Configuração do canal de comunicação dentro da enumeração:<br>**ctcSerial:** Comunicação Serial 232<br>**ctcTcpIp:** Comunicação TcpIp<br>**ctcUsb:** Comunicação Usb<br>**ctcModem:** Comunicação Modem<br>**ctcSerial485:** Comunicação Serial 485<br>**ctcGPRS:** Comunicação GPRS |
| IsCatraca | Boolean | Determina se equipamento é catraca, está configuração interfere em diversos pontos de gerenciamento do equipamento no kernel |

## Estrutura SComTcp

| Atributo | Tipo | Descrição |
|---|---|---|
| Ip | String | Endereço IP v4 do equipamento ou roteador. |
| Porta | Inteiro | Porta de conexão no equipamento ou roteador. Padrão 3000. |
| Mac | String | Opcional, o preenchimento indica comunicação DHCP e o gerenciamento será realizado pelo kernel, caso contrário enviar vazio |

## Estrutura SComSerial

| Atributo | Tipo | Descrição |
|---|---|---|
| NumeroRelogio | String | Número do equipamento, usado para comunicação, suporta valores de 1 até 85, outros valores são obtidos através emulação do seu software. |
| Porta | String | Porta COM para conexão. Ex: “COM1” |
| Velocidade | String | Velocidade de comunicação serial dentro da enumeração:<br>**cv9600:** 9600bps<br>**cv19200:** 19200bps<br>**cv57600:** 57600bps<br>**cv115200:** 115200bps |

## Estrutura SComModem

| Atributo | Tipo | Descrição |
|---|---|---|
| Fone | String | Número do fone para conexão |
| Porta | String | Porta COM onde esta instalado o modem |

## Estrutura SComGPRS

| Atributo | Tipo | Descrição |
|---|---|---|
| Porta | Inteiro | Porta que será utilizada na comunicação via GPRS. |

**AdicionaCardSerial(pNumero : Byte; pPorta : WideString; pVelocidade : SVelocidade; pCatraca : WordBool ; pModoComunicacao : SModoComunicacao ) : Integer;**

**AdicionaCardTcpip(pIp, pMac : WideString; pPorta : Integer; pCatraca : WordBool; pModoComunicacao : SModoComunicacao) : Integer;**

**AdicionaCardUsb(pCatraca : WordBool) : Integer;**

**AdicionaCardModem(pPorta, pFone : Wi deString pCatraca : WordBool) : Integer;**

**AdicionaCardSerial485(pNumero : Byte; pPorta : WideString; pVelocidade : SVelocidade; pCatraca : WordBool; pModoComunicacao : SModoComunicacao) : Integer;**

**AdicionaCard GPRS(pCatraca: WordBool; pPorta: Integer; pModoComunicacao: SModoComunicacao): Integer ;**

Os métodos alternativos de adição de equipamento dividem a estrutura SComConfig, facilitando o entendimento dos métodos necessários para cada comunicação.

- **RemoveCard(pThreadIndex: Integer): WordBool;**

Este método exclui a thread indicada.

**RemoveCard(pThreadIndex: Integer): WordBool;**

- **Set485OffNumber(pThreadIndex: Integer; pNumero: Byte): WordBool;**

A comunicação 485 está inversamente relacionada ao conceito multi -threads do Kernel. O gerenciamento online é feito de forma “automática”, porém para execução da comunicação com um determinado equipamento é necessário indicar qual já que todos estão na mesma thread, esta indicação acontece através do número do equipamento .

**Set485OffNumber (pThreadIndex: Integer ; pNumero : Byte): WordBool;**

- **SetSincronizar(pThreadIndex: Integer; pSincronizar: WordBool);**

Este método define se o equipamento indicado deve verificar e atualizar se necessário a data e hora a cada batida on-line.

**SetSincronizar(pThreadIndex: Integer; pSincronizar : WordBool): WordBool;**

- **ThreadLastError(pThreadIndex: Integer): Integer;**

Esta propriedade retorna o último erro ocorrido na thread. As constantes de erro podem ser encontradas no fim da documentação.

**ThreadLastError(pThreadIndex: Integer): Intege r;**

- **ErrorDescription (pErrorCode: Integer): WideString;**

Retorna a descrição do erro de acordo com o código.

**ErrorDescription(pErrorCode: Integer): WideString;**

- **KernelLastError: Integer;**

Esta propriedade retorna o último erro ocorrido no kernel. As const antes de erro podem ser encontradas no fim da documentação.

**KernelLastError: Integer;**

- **RaiseExceptions : WordBool;**

Este método define a exibição de mensagem dos erros. O Padrão é desativado.

**RaiseExceptions : WordBool;**

- **SetConectado(pThreadIndex: Integer; pConectado: WordBool);**

Este método permite conectar e desconectar um equipamento. Usado para atualizações de firmware ou compartilhar um canal de comunicação.

**SetConectado(pThreadIndex: Integer; pConectado: WordBool);**

- **SetICMPProtocol(pThreadIndex: Integer; pEnabled: WordBool);**

Este método permite conectar e desconectar um equipamento. Usado para atualizações de firmware ou compartilhar um canal de comunicação.

**SetICMPProtocol (pThreadIndex: Integer; pEnabled: WordBool);**

- **SetConcentrador (pThreadIndex: Integer; pIsConcentrador: WordBool);**

  Este método determina que a thread esta gerenciando um concentrador. Essa configuração interfere na interpretação de registros entre outros dados e operações.

  **SetConcentrador(pThreadIndex : Integer; pIsConcentrador : WordBool);**

- **ThreadPrioridade(pThreadIndex: Integer; pPrioridade: SPrioridade);**

  Este método define a prioridade da comunicação com o equipamento indicado.

  **ThreadPrioridade(pThreadIndex: Integer; pPrioridade: SPrioridade);**

- **EnviaAcionaCtrl (pThreadIndex: Integer; pId : Byte; pAcionaCtrl : SAcionaCtrl);**

  Este método envia acionamento de reles ao controlador sem realização de marcação.

  **EnviaAcionaCtrl(pThreadIndex: Integer; pId : Byte; pAcionaCtrl : SAcionaCtrl);**

- **setConnResetTimeout (pThreadIndex: Integer; pTimeout: SResetCon);**

  Este método define o intervalo de reinicialização de porta para equipamentos GPRS.

  **setGPRSResetTimeout (pThreadIndex: Integer; pTimeout: SResetCon);**

- **getConnResetTimeout(pThreadIndex: Integer; out pTimeout: SResetCon): WordBool;**

  Este método retorna o intervalo de reinicialização de porta definido para equipamentos GPRS.

  **getGPRSResetTimeout(pThreadIndex: Integer; out pTimeout: SResetCon);**

# Configurações básicas

- **EnviaConfiguracao(pThreadIndex: Integer; pConfig: SConfiguracao): WordBool;**

  Este método envia configuração para o equipamento indicado. Para a interface kernel7x.alternativo será necessário armazenar a configuração em cachê antes de realizar as operações de envio e outras como operações SR (Size Required).

**cfg_setReles(ReleIndex : Byte; Status : SReleStatus; TipoNANF : SReleNANF; Tempo : Byte);**

Neste método ReleIndex informa o rele de 0 a 4 e os demais parâmetros a configuração do mesmo.

**cfg_setLeitores (Leitor1, Leitor2, Leitor3 : SLeitor);**

**cfg_setCodigoBarras(PadraoD, Letras, PadraoLivre, DigitosAutomatico, PadraoT, OcultarDigitos : WordBool);**

**cfg_setRevista(Tipo : STipoRevista; Percentual : Byte);**

**cfg_setEmpresas(Empresa1, Empresa2, Empresa3, Empresa4, Empresa5 : WideString);**

**cfg_setCtrlAcessoEx(Gravacao, AtivarMaster, Sinaliza50Percent, SenhaPadraoHenry, Catraca, CatracaInvertida, CatracaDupla : WordBool) ;**

**cfg_setControles(BloqueiaPeriodo, BiometriaOnline, FuncoesEspecificas, CatBioLiberaAmbos, Visitantes, Touch, AutoOn : WordBool);**

**cfg_setAntiPassBack(Habilitado, EntradaSaida : WordBool; Tempo : Byte);**

**cfg_setSensores(SensorIndex : Byte; Habilitado, Porta, Botao, Rele1, Rele2, Rele3, Rele4, Rele5 : WordBool);**

Neste método SensorIndex informa o sensor de 0 a 1 e os demais parâmetros a configuração do mesmo.

**cfg_setConfig(ModoComunicacao : SModoComunicacao; Teclado : STeclado; NumDigitos, NivelAcesso, ToquesAtender : Byte; DigitosSel : SDigitos; SenhaMenu : WordBool; Senha : WideString; CtrlAcesso : SCtrlAcesso);**

**cfg_setControleLeitoras(pLeitoraVerificaDigital, pBiometria11: WordBool);**

**cfg_getControleTempos(out pTimeoutProcessamentoOnline, pLatenciaOffline: Integer);**

**cfg_setControleTempos(pTimeoutProcessamentoOnline, pLatenciaOffline: Integer);**

**EnviaConfiguracao(pThreadIndex : Integer ) : WordBool;**

Envia a configuração armazenada com os métodos alternativos apresentados acima.

- **RecebeConfiguracao(pThreadIndex: Integer; out pConfig: SConfiguracao): WordBool;**

  Este método recebe configuração do equipamento indicado.

  **RecebeConfiguracao(pThreadIndex : Integer ) : WordBool;**

  Recebe a configuração do equipamento e armazena para que os dados sejam obtidos com os métodos alternativos apresentados abaixo.

  **cfg_getInfo(out Versao : WideString; out Placa : SPlacaCard; out Expansao : WordBool; out Orion : WordBool; out Biom etria : SBiometria);**

  **cfg_getReles(ReleIndex : Byte; Status : SReleStatus; TipoNANF : SReleNANF; Tempo : Byte);**

  Neste método ReleIndex informa o rele de 0 a 4 e os demais parâmetros a configuração do mesmo.

  **cfg_getLeitores (out Leitor1, Leitor2, Leitor3 : SLeitor);**

  **cfg_getCodigoBarras(out PadraoD, Letras, PadraoLivre, DigitosAutomatico, PadraoT, OcultarDigitos : WordBool);**

  **cfg_getRevista(out Tipo : STipoRevista; out Percentual : Byte);**

  **cfg_getEmpresas(out Empresa1, Empresa2, Empresa3, Empresa4, Empresa5 : WideString);**

  **cfg_getCtrlAcessoEx(out Gravacao, AtivarMaster, Sinaliza50Percent, SenhaPadraoHenry, Catraca, CatracaInvertida, CatracaDupla : WordBool);**

  **cfg_getControles(out BloqueiaPeriodo, BiometriaOnline, FuncoesEspecificas, CatBioLiberaAmbos, Visitantes, Touch, AutoOn : WordBool);**

  **cfg_getAntiPassBack(out Habilitado, EntradaSaida : WordBool; Tempo : Byte);**

  **cfg_getSensores(SensorIndex : Byte; out Habilitado, Porta, Botao, Rele1, Rele2, Rele3, Rele4, Rele5 : WordBool);**

Neste método SensorIndex informa o sensor de 0 a 1 e os demais parâmetros a configuração do mesmo.

**cfg_getConfig(out ModoComunicacao : SModoComunicacao; out Teclado : STeclado; out NumDigitos, NivelAcesso, ToquesAtender : Byte; out DigitosSel : SDigitos; out SenhaMenu : WordBool; out Senha : WideString; out CtrlAcesso : SCtrlAcesso);**

**cfg_getControleLeitoras(out pLeitoraVerificaDigital, pBiometria11: WordBool); safecall;**

## Estrutura SConfiguracao

| Atributo | Tipo | Descrição |
|---|---|---|
| Versao | String | Versão do Firmware |
| PlacaCard | SPlacaCard | Modelo do Equipamento dentro da enumeração: **cpcCardI, cpcCardII, cpcCardIII, cpcCardIV, cpcCardV, cpcTikko, cpcConcentrador, cpcProxIP** |
| Orion | Boolean | Informa se modelo mecânico é Orion |
| Expansao | Boolean | Informa se cartucho de memória possui memória com 4 chips |
| ModoComunicacao | SModoComunicacao | Determina modo de trabalho do equipamento dentro da enumeração:<br>• **cmcOffline** = Equipamento armazena registros para coleta posterior<br>• **cmcOnline =** Equipamento realiza marcações com armazenamento e tomada de decisão no servidor em tempo real.<br>• **cmcOnOff** = Trabalha Offline ou Online automaticamente dependendo da disponibilidade do servidor.<br>• **cmcOnOffCtrl** =OnOff porém consultando lista de acesso em offline |
| Teclado | STeclado | Determina modo de operação do teclado, dentro da enumeração: c**tDesativado , ctPerguntaES, ctLeitor1, ctLeitor2, ctLeitor3Bio, ctAtalhoFuncoes** |
| Reles | SReles | Configuração dos reles de 1 a 5 |
| Leitores | SLeitores | Configuração dos leitores de 1 a 3 |
| NivelAcesso | Byte | Reservado |
| CodigoBarras | SConfigBarCode | Configurações de Código de Barras |
| NumDigitos | Byte | Quantidade de dígitos do cartão |
| Revista | SConfigRevista | Opções de revista offline |
| SenhaMenu | Boolean | Menu esta protegido por senha |

Somente leitura

| | | |
|---|---|---|
| Senha | String | 5 caracteres numéricos com a senha do menu |
| DigitosSel | SDigitos | Digitos selecionados, a string atende a concatenação padrão do kernel, dígito formatado em 3 casas e concatenado. Ex: “001003005” |
| Empresas | SEmpresas | Reservado |
| Biometria | SBiometria | Modelo da biometria do equipamento |
| CtrlAcesso | SCtrlAcesso | Configurações de Acesso |
| CtrlAcessoEx | SCtrlAcessoEx | Configurações extra de acesso |
| ToquesAtender | Byte | Toques para equipamento atender quando modem |
| Controles | SControles | Configurações de controles adicionais |
| AntiPassBack | SAntiPassBack | Configurações de antipassback offline |
| Sensores | SSensores | Configurações dos sensores |
| AutoOff | SAutoOff | Configurações de desligamento automático para economia de energia |
| Controladores | String | Retorna controladores online com o concentrador |
| FaixaAcesso | Boolean | Indica se equipamento usa faixa de acesso. |
| ControleTempos | SControleTempos | Configura timeout de processamento de bilhetes online e latência offline. |

- **EnviaDadosEmpregador(pThreadIndex: Integer; pEmpregador: SEmpregador): WordBool;**

  Este método envia as informações do empregador para o equipamento. Disponível apenas para Orion 6.

**Get_EnviaDadosEmpregador(pThreadIndex: Integer; const pRazaoSocial, pLocal, pDocumento, pCEI: WideString; pIdEmpregador: SIdEmpregador): WordBool;**

- **RecebeDadosEmpregador(pThreadIndex: Integer; out pEmpregador: SEmpregador): WordBool; safecall;**

  Este método Recebe as informações do empregador cadastrado no equipamento. Disponível apenas para Orion 6.

**Get_ RecebeDadosEmpregador(pThreadIndex: Integer; out pRazaoSocial, pLocal, pDocumento, pCEI: WideString; out pIdEmpregador: SIdEmpregador; out pStatus: WordBool); safecall;**

## Estrutura SEmpregador

| Atributo | Tipo | Descrição |
|---|---|---|
| IdEmpregador | SIdEmpregador | Tipo de documento: **cieCPF; cieCNPJ;** |
| Documento | String | Número do documento do empregador |
| CEI | String | CEI (caso existir) |
| RazaoSocial | String | Razão social do empregador |
| Local | String | Local |

- **EnviaUsuarioEquipamento(pThreadIndex: Integer; pUsuarioEquipamento: SUsuarioEquipamento): WordBool;**

  Este método envia um usuário ao equipamento. Disponível apenas para Orion 6.

**Get_EnviaUsuarioEquipamento(pThreadIndex: Integer; const pMatricula, pPIS, pNome: WideString; pVerificaDigital: WordBool; pTipoOperacao: SOperacaoUsuarioEquipamento): WordBool;**

- **RecebeListaUsuarioEquipamento(pThreadIndex: Integer): WordBool;**

  Recebe a lista de usuários no equipamento. Ao receber, armazena os dados numa lista que pode ser acessada através da função “Rec_UsuarioEquipamento”. Disponível apenas para Orion 6.

**Get_ RecebeListaUsuarioEquipamento(pThreadIndex: Integer): WordBool;**

- **Rec_UsuarioEquipamento(pThreadIndex: Integer; out pUsuarioEquipamento: SUsuarioEquipamento): WordBool;**

  Retorna um usuário da lista de usuários. Enquanto o retorno da função for verdadeiro, existirão usuários na lista. Disponível apenas para Orion 6.

**Get_ Rec_UsuarioEquipamento(pThreadIndex: Integer; out pMatriculas, pPIS, pNome: WideString; out pVerificaDigital: WordBool;**

## Estrutura SUsuarioEquipamento

| Atributo | Tipo | Descrição |
|---|---|---|
| Matriculas | String | Matrículas do usuário. As matrículas devem ser separadas pelo caractere ';'. Todas as matrículas serão utilizadas para realizar a identificação do usuário. |
| PIS | String | PIS do usuário |
| Nome | String | Nome do usuário (52 caracteres) |
| VerificaDigital | Boolean | Flag de verificação de digital |
| TipoOperacao | SOperacaoUsuario Equipamento | Tipo de operação: **couAdicao, couAlteracao, couExclusao** |

- **EnviaCfgControlador (pThreadIndex : Integer; pId : Byte; pConfig : SConfigCtrl): WordBool;**

Este método envia as configurações para o controlador de uma concentradora.

**Get_EnviaConfiguracaoControlador(pThr eadIndex, pIdControlador: Integer): WordBool;**

**cfg_setCfgControlador(pCatraca, pWorkOff: WordBool; pTempoRele1, pTempoRele2, pTempoRele3: Byte);**

**cfg_setLeitoresCtrl(pIndexLeitora: Integer; pRele1, pRele2, pRele3: Byte);**

**cfg_setSensoresCtrl(pIndex Sensor: Integer; pHabilitado: WordBool; pTipoSensor: STipoSensor; pRele1, pRele2, pRele3: WordBool);**

- **RecebeCfgControlador (pThreadIndex : Integer; pId : Byte; out pConfig : SConfigCtrl): WordBool;**

Este método recebe as configurações do controlador de uma concentradora.

**Get_RecebeConfiguracaoControlador(pThreadIndex, pIdControlador: Integer): WordBool;**

**cfg_getCfgControlador(out pCatraca, pWorkOff: WordBool; out pTempoRele1, pTempoRele2, pTempoRele3: Byte);**

**cfg_getLeitoresCtrl(pIndexLeitor a: Integer; out pRele1, pRele2, pRele3: Byte);**

**cfg_getSensoresCtrl(pIndexSensor: Integer; out pHabilitado: WordBool; out pTipoSensor: STipoSensor; out pRele1, pRele2, pRele3: WordBool);**

- **AlterarVelocidade(pThreadIndex: Integer; pNovaVelocidade: SVelocidade): WordBool;**

Este método altera a velocidade de comunicação do equipamento indicado (A velocidade da thread é alterada automaticamente em caso de sucesso).

**AlterarVelocidade(pThreadIndex: Integer; pNovaVelocidade: SVelocidade): WordBool;**

- **EnviaDataHora(pThreadIndex: Integer; pDataHora: TDateTime): WordBool;**

Este método realiza o envio de Data e Hora para o equipamento.

**EnviaDataHora(pThreadIndex: Integer; pDataHora: TDate Time): WordBool;**

- **RecebeDataHora(pThreadIndex: Integer; out pDataHora: TDateTime): WordBool;**

Este método recebe a data e hora do equipamento.

**RecebeDataHora(pThreadIndex: Integer; pDataHora: TDateTime): WordBool;**

- **EnviaDataHoraEx(pThreadIndex: Integer; pDataHoraEx: SDataHoraCompleta): WordBool;**

Este método executa a mesma operação do método EnviaDataHora, porém também configura o horário de verão.

**EnviaDataHoraEx(pThreadIndex: Integer; pDataHora: TDateTime; pUsarHorarioVerao : WordBool; pHorarioVersaoInicio, pHorarioVeraoTermino : TDate): WordBool;**

Caso pUsarHorarioVersao seja falso o inicio e término não serão necessários.

- **RecebeDataHoraEx(pThreadIndex: Integer; out pDataHoraEx: SDataHoraCompleta): WordBool;**

Este método executa a mesma operação do método RecebeDaraHora, porém também recebe a configuração de horário de verão.

```
RecebeDataHoraEx(pThreadIndex: Integer; out pDataHora: TDateTime; out pUsarHorarioVerao : WordBool; out pHorarioVeraoInicio, pHorarioVeraoTermino : TDate );
```

- **EnviaTipoCatraca(pThreadIndex: Integer; pOperacao: SOperacaoCatraca): WordBool;**

Este método envia configuração de operação para catracas. A operação da catraca é normalmente bloqueada, porém pode ser liberada em algum sentido ou ambos, por um determinado tempo ou até que receba outra configuração.

```
EnviaTipoCatraca(pThreadIndex: Integer; out pStatusGiro : SStatusGiro; out pTempoLiberacao : Byte );
```

- **RecebeTipoCatraca(pThreadIndex: Integer; out pOperacao: SOperacaoCatraca): WordBool;**

Este método recebe a configuração de operação de catracas.

```
RecebeTipoCatraca(pThreadIndex: Integer; out pStatusGiro : SStatusGiro; out pTempoLiberacao : Byte);
```

- **EnviaMsgPadrao(pThreadIndex: Integer; pMsgPadrao: SMsgPadrao): WordBool;**

Este método envia as mensagens padrão para o equipamento, o parâmetro pMsgPadrao é divido em 3 atributos do tipo SMensagem, sendo cada atributo responsável pela exibição em determinados eventos do equipamento de forma independente dos outros.

## Estrutura SMsgPadrao

| Atributo | Tipo | Descrição |
|---|---|---|
| Padrão | SMensagem | Exibido quando equipamento em standby |
| Entrada | SMensagem | Exibido quando é realizada marcação de entrada |
| Saída | SMensagem | Exibido quando é realizada marcação de saída |

## Estrutura SMensagem

| Atributo | Descrição |
|---|---|
| Estilo | Estilo da mensagem que deve aparecer. Alguns estilos desabilitam a exibição de linhas:<br>• cmsDesativada (Display em branco)<br>• cmeMatricula(A matrícula aparece em linha1, válido em Entrada e Saida)<br>• cmeDataHora(A linha1 exibe Data e Hora)<br>• cmePersonalizada (As duas linhas são customizáveis)<br>• cmeSaudacao (A linha1 exibe “Bom Dia” por exemplo) |
| Linha1 | 16 Caracteres ASCII que devem aparecer na linha superior |
| Linha2 | 16 Caracteres ASCII que devem aparecer na linha inferior |
| Tempo | Tempo em segundos de exibição no display 1..255 (Desconsiderado na mensagem Padrão) |

**EnviaMsgPadrao(pThreadIndex: Integer; pEstiloPadrao, pEstiloEntrada, pEstiloSaida: SMsgEstilo; pMsgPadraoLinha1, pMsgPadraoLinha2, pMsgEntradaLinha1, pMsgEntradaLinha2, pMsgSaidaLinha1, pMsgSaidaLinha2 : WideString; pTempoPadrao, pTempoEntrada, pTempoSaida : Byte): WordBool;**

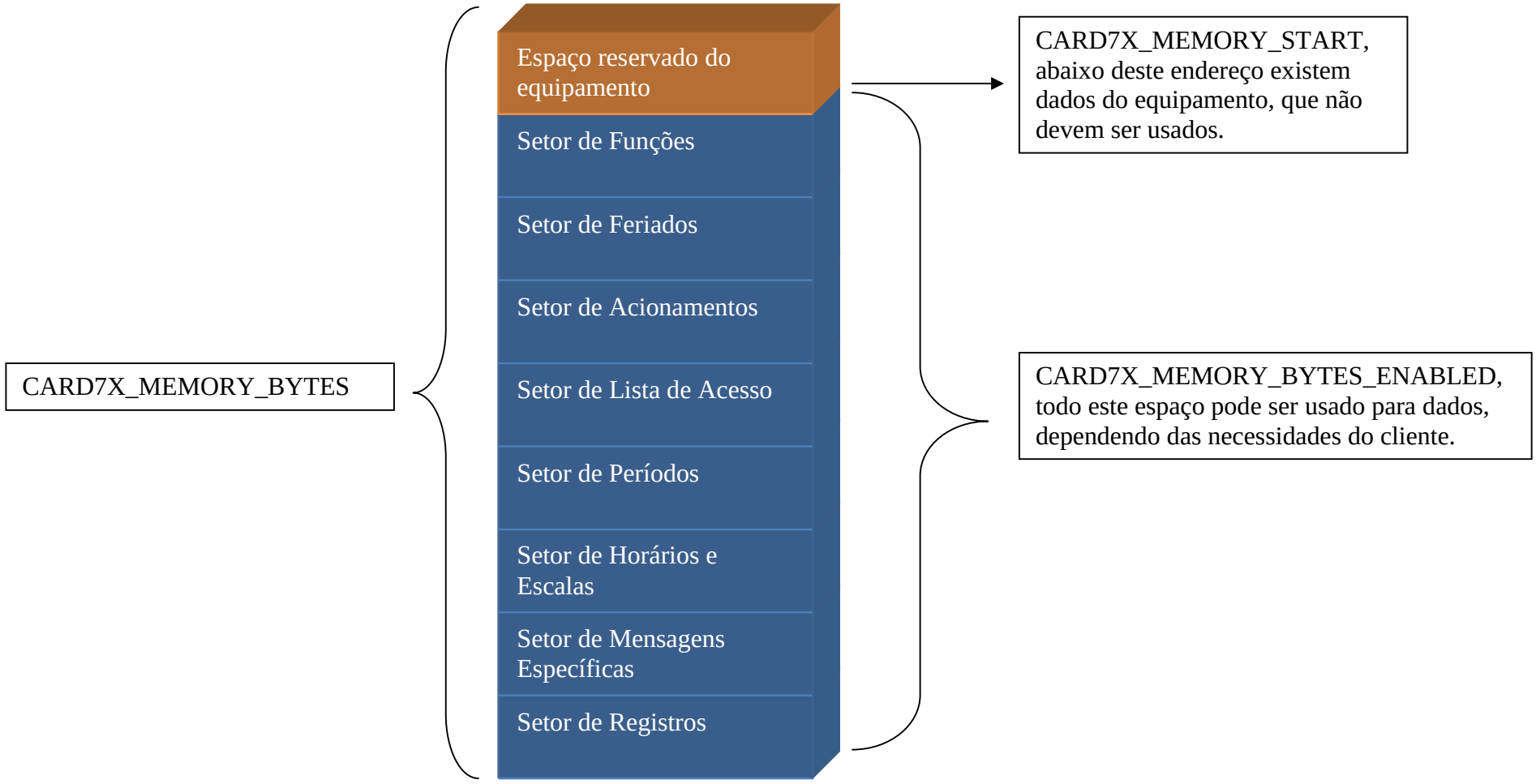
- **RecebeMsgPadrao(pThreadIndex: Integer; out pMsgPadrao: SMsgPadrao): WordBool;**

Este método recebe as mensagens padrão do equipamento.

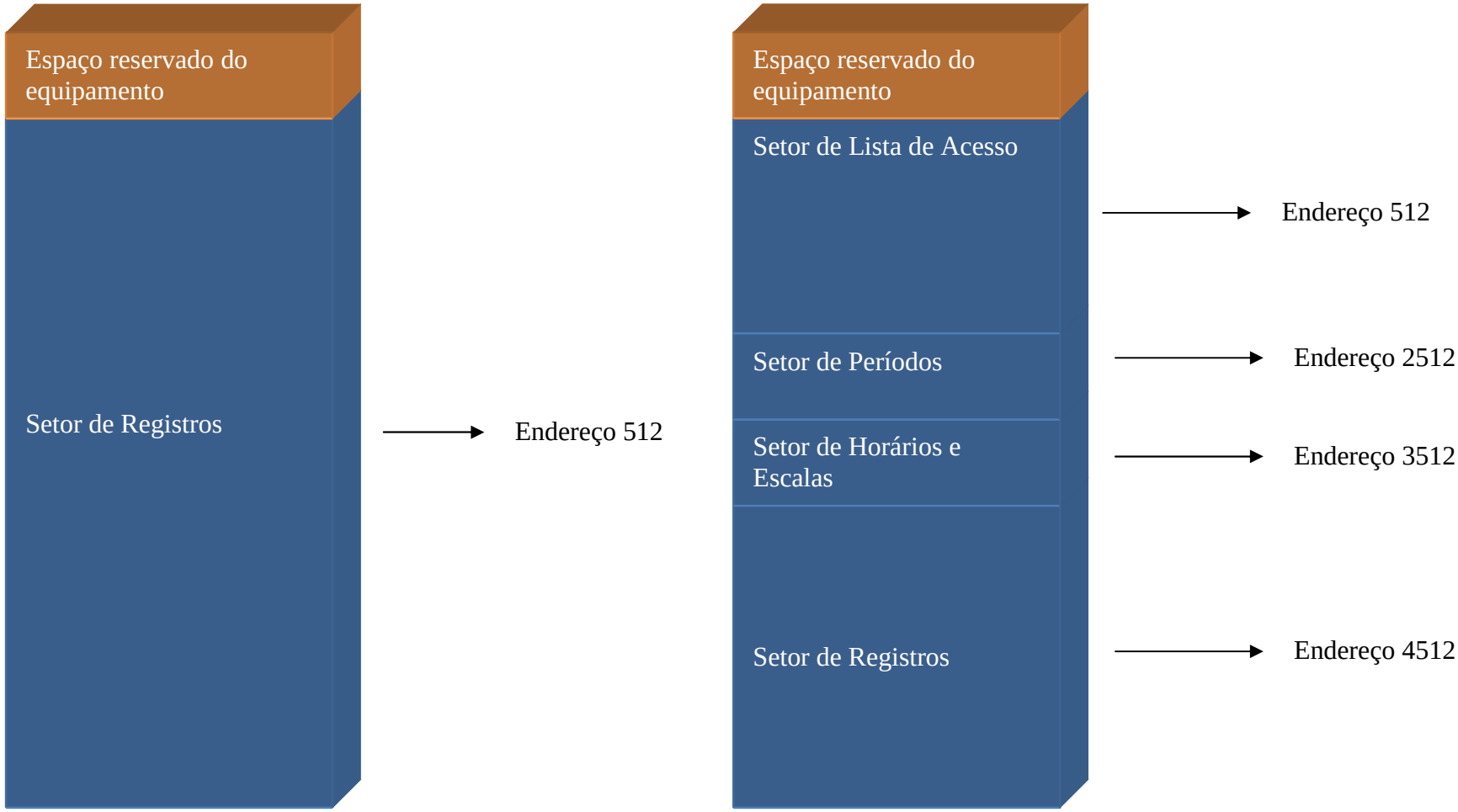
```
RecebeMsgPadrao(pThreadIndex: Integer;
out pEstiloPadrao, pEstiloEntrada, pEstiloSaida: SMsgEstilo;
out pMsgPadraoLinha1, pMsgPadraoLinha2, pMsgEntradaLinha1,
pMsgEntradaLinha2, pMsgSaidaLinha1, pMsgSaidaLinha2 :
WideString;
out pTempoMsgPadrao, pTempoMsgEntrada, pTempoMsgSaida :
Byte
out pStatus: WordBool);
```

# Formatação ou Particionamento

Na formatação devem ser passados os endereços dos setores na memória.
O inicio dos endereços esta em CARD7X_MEMORY_START, em seguida os setores recebem a soma do tamanho do antecessor, quando desativado o endereço do setor é zero. Existem ainda outras constantes de controle CARD7X_MEMORY_BYTES e CARD7X_MEMORY_BYTES_ENABLED, que definem respectivamente o tamanho da memória e total de bytes formatáveis.
O desenho abaixo exemplifica a disposição da memória do equipamento utilizando todos os setores.



Os desenhos abaixo demonstram a formatação padrão e exemplo comum de uso.
Para saber o valor dos endereços utiliza os métodos SR descritos abaixo.



Os método SR (Size Required) retornam o tamanho necessário para o setor. Além de trabalharem offline com base na configuração passada. No caso da interface kernel7x.alternativo a configuração usada será a ultima armazenada em cachê.

- **SRFuncoes (pConfig : SConfiguracao; pQtFuncoes : Integer) : Integer;**

Este método retorna a quantidade de bytes, solicitando a quantidade de funções.

**SRFuncoes (pQtFuncoes : Integer) : Integer;**

- **SRFuncoesEspecificas (pConfig : SConfiguracao; pQtFuncoes, pQtFuncoesPorMatr : Integer) : Integer;**

Este método retorna a quantidade de bytes, solicitando a quantidade de funções e quantidade de funções por matrícula.

**SRFuncoesEspecificas (pQtFuncoes, pQtFu ncoesPorMatr : Integer) : Integer;**

- **SRFeriados(pConfig : SConfiguracao; pQtFeriados: Integer) : Integer;**

Este método retorna a quantidade de bytes, solicitando a quantidade de feriados.

**SRFeriados(pQtFeriados: Integer) : Integer;**

- **SRAcionamentos(pConfig : SConfiguracao; pQtAcionamentos: Integer) : Integer;**

Este método retorna a quantidade de bytes, solicitando a quantidade de acionamentos.

**SRAcionamentos(pQtAcionamentos: Integer) : Integer;**

- **SRListaAcesso(pConfig : SConfiguracao; pQtItems: Integer) : Integer;**

Este método retorna a quantidade de bytes, solicitando a quantidade de itens de acesso.

**SRListaAcesso(pQtItems: Integer) : Integer;**

- **SRPeriodos(pConfig : SConfiguracao; pQtPeriodos: Integer) : Integer;**

Este método retorna a quantidade de bytes, so licitando a quantidade de períodos.

**SRPeriodos(pQtPeriodos: Integer) : Integer;**

- **SRHorariosEscalas(pConfig : SConfiguracao; pQtHorarios, pQtPeriodosPorHorario, pQtEscalas, pQtHorariosPorEscala: Integer) : Integer;**

Este método retorna a quantidade de byte s, solicitando a quantidade de horários, máximo de períodos que compõem os horários, quantidade de escalas, máximo de horários que compõem cada escala..

**SRHorariosEscalas(pQtHorarios, pQtPeriodosPorHorario, pQtEscalas, pQtHorariosPorEscala: Integer) : In teger;**

- **SRMsgEspecifica(pConfig : SConfiguracao; pQtMatriculas, pQtMsgPorMatr: Integer) : Integer;**

Este método retorna a quantidade de bytes, solicitando a quantidade de matrículas e mensagens por matrícula.

**SRMsgEspecifica(pQtMatriculas, pQtMsgPorMatr: Integer) : Integer;**

- **EnviaParticionamento(pThreadIndex: Integer; pParticionamento: SParticionamento): WordBool;**

Este método define uma nova configuração de particionamento da memória do equipamento.
No caso de concentrador o setor de funções é obrigatóri o e deve ter tamanho de 288 bytes, os setores de acionamentos e mensagens especificas nunca devem ser usados.

**EnviaParticionamento(pThreadIndex: Integer; p Funcoes, pFeriados, pAcionamentos, pListaAcesso, pPeriodos, pHorarios, pMsgEspecifica, pRegistros : Integer): WordBool;**

- **RecebeParticionamento(pThreadIndex: Integer; out pParticionamento: SParticionamento): WordBool;**

Este método recebe a configuração de particionamento do equipamento indicado.

**RecebeParticionamento(pThreadIndex: Integer; out pFuncoes, pFeriados, pAcionamentos, pListaAcesso, pPeriodos, pHorarios, pMsgEspecifica, pRegistros : Integer);**

# Inserindo Dados nos setores

- **Add_FncEsp_Matricula(pThreadIndex: Integer; pMatricula : WideString);**

  Este método adiciona uma matrícula * ao cachê de funções especificas, esta será usada como referência para o método Add_FncEsp_Funcao.

  ```
  Add_FncEsp_Matricula(pThreadIndex: Integer; pMatricula :
  WideString);
  ```

- **Add_FncEsp_Funcao (pThreadIndex: Integer; pMatricula : WideString; pFuncao : SFuncaoEx);**

  Este método adiciona uma função * ao cachê de funções especificas para o método EnviaFuncoes que esvazia o cachê ao ser executado.

  ```
  Add_FncEsp_Funcao (pThreadIndex: Integer;  pMatricula,
  pMensagem : WideString; pTempo, pNumero : Byte);
  ```

- **Add_Funcao(pThreadIndex: Integer; pFuncao: SFuncao);**

  Este método adiciona uma função ao cachê de funções para o método EnviaFuncoes que esvazia o cachê ao ser executado.

  ```
  Add_ Funcao (pThreadIndex: Integer; pAtiva , pLiberaAcesso :
  WorldBool; pMensagem : WideString; pTempo : Byte);
  ```

- **EnviaFuncoes(pThreadIndex: Integer): WordBool;**

  Este método executa o envio das funções anteriormente carregados com o método Add_Funcao, considerando o tamanho do setor.

  ```
  EnviaFuncoes (pThreadIndex: Integer) : WordBool;
  ```

- **RecebeFuncoes(pThreadIndex: Integer): WordBool;**

  Este método executa a recepção de funções e disponibiliza no método Rec_Funcao.

  ```
  RecebeFuncoes (pThreadIndex: Integer) : WordBool;
  ```

- **Rec_Funcao(pThreadIndex: Integer; out pFuncao: SFuncao): WordBool;**

Este método retorna uma função do cachê de funções carregado pelo método RecebeFuncoes, ao receber uma função ela é automaticamente excluída do cachê.

```
Rec_Funcao (pThreadIndex: Integer; out pAtiva, pLiberaAcesso :
WordBool; out pMensagem : WideString; out pTempo : Byte);
```

- **Rec_FncEsp_Matricula(pThreadIndex: Integer; out pMatricula : WideString): WordBool;**

Este método retorna uma matrícula com funções relacionadas em cachê, estas podem ser recebidas com o método Rec_FncEsp_Funcao. Para utilizar funções especificas habilite em SConfiguracao.Controles, isto desativa automaticamente funções normais.

```
Rec_FncEsp_Matricula(pThreadIndex: Integer; out pMatricula :
WideString; out pStatus: WordBool);
```

- **Rec_FncEsp_Funcao(pThreadIndex: Integer; pMatricula : WideString; out pFuncao : SFuncaoEx): WordBool;**

Este método retorna uma função relacionada a matrícula indicada, após receber todas a funções a matrícula é excluída do cachê.

```
Rec_FncEsp_Funcao(pThreadIndex: Integer; pMatricula ,
pMensagem : WideString; out pTempo, pNumero : Byte; out
pStatus: WordBool);
```

- **Add_Feriado(pThreadIndex: Integer; pDia: TDateTime);**

Este método adiciona um feriado ao cachê de funções para o método EnviaFeriados que esvazia o cachê ao ser executado.

```
Add_Feriado(pThreadIndex: Integer; pDia: TDateTime);
```

- **EnviaFeriados(pThreadIndex: Integer): WordBool;**

Este método executa o envio dos feriados anteriormente carregados com o método Add_Feriado, considerando o tamanho do setor.

**EnviaFeriados(pThreadIndex: Integer): WordBool;**

- **RecebeFeriados(pThreadIndex: Integer): WordBool;**

Este método executa a recepção de funções e disponibiliza no método Rec_Feriado.

**RecebeFeriados(pThreadIndex: Integer): WordBool;**

- **Rec_Feriado(pThreadIndex: Integer; out pDia: TDateTime): WordBool;**

Este método retorna um feriado do cachê de feriados carregado pelo método RecebeFeriados, ao receber um feriado ele é automaticamente excluído do cachê.

**Rec_Feriado(pThreadIndex: Integer; out pDia: TDateTime);**

- **Add_Acionamento(pThreadIndex: Integer; pAcionamento: SAcionamento);**

Este método executa o envio dos acionamentos anterio rmente carregados com o método Add_Acionamento, considerando o tamanho do setor.

**Add_Acionamento(pThreadIndex: Integer; out pHorario: TDateTime; pTempo : Byte; pDomingo, pSegunda, pTerca, pQuarta, pQuinta, pSexta, pSabado, pFeriado : WordBool );**

- **EnviaAcionamentos(pThreadIndex: Integer): WordBool;**

Este método executa o envio dos acionamentos anteriormente carrega com o método Add_Acionamento, considerando o tamanho do setor.

**EnviaAcionamentos(pThreadIndex: Integer): WordBool;**

- **RecebeAcionamentos(pThreadIndex: Integer): WordBool;**

Este método executa a recepção de acionamentos e disponibiliza no método Rec_Acionamento.

**RecebeAcionamentos(pThreadIndex: Integer): WordBool;**

- **Rec_Acionamento(pThreadIndex: Integer; out pAcionamento: SAcionamento): WordBool;**

Este método retorna um acionamento do cachê de acionamentos carregado pelo método RecebeAcionamentos, ao receber um acionamento ele é automaticamente excluído do cachê.

```
Rec_Acionamento(pThreadIndex: Integer; out p Horario: TDateTime;
out pTempo : Byte; out pD omingo, pSegunda, pTerca, pQuarta,
pQuinta, pSexta, pSabado, pDomingo : WordBool );
```

- **Add_Periodo(pThreadIndex: Integer; pPeriodo: SPeriodo);**

Este método adiciona um período ao cachê de períodos para o método EnviaPeriodos que esvazia o cachê ao ser executad o.

```
Add_Periodo(pThreadIndex: Integer; out p Horario: TDateTime;
pTolerancia : Byte; pDomingo, pSegunda, pTerca, pQuarta, pQuinta,
pSexta, pSabado, pFeriado : WordBool ): WordBool;
```

- **EnviaPeriodos(pThreadIndex: Integer): WordBool;**

Este método executa o envio dos períodos anteriormente carregados com o método Add_Periodo, considerando o tamanho do setor.

```
EnviaPeriodos(pThreadIndex: Integer): WordBool;
```

- **RecebePeriodos(pThreadIndex: Integer): WordBool;**

Este método executa a recepção dos períodos e disponibiliza no método Rec_Periodo.

```
RecebePeriodos(pThreadIndex: Integer): WordBool;
```

- **Rec_Periodo(pThreadIndex: Integer; out pPeriodo: SPeriodo): WordBool;**

Este método retorna um período do cachê de períodos carregado pelo método RecebePeriodos, ao receber um período ele é automaticamente excluído do cachê.

**Rec_Periodo(pThreadIndex: Integer; out pHorario: TDateTime; pTolerancia : Byte; pSegunda, pTerca, pQuarta, pQuinta, pSexta, pSabado, pFeriado : WordBool): WordBool;**

- **Add_Horario(pThreadIndex: Integer; pPeriodos: WideString; pHorarioIndex : Byte) : WordBool;**

Este método adiciona um Horário ao cachê de Horários para o método EnviaHorários que esvazia o cachê ao ser executado e retorna o índex para uso em escalas ou lista de acesso. O Formato da string são os index formatados em 3 casas e concatenados. **

** O Horário é um conjunto de períodos.
** Ex: '001002003' contém 3 períodos
** O primeiro período possui index = 1

**Add_Horario(pThreadIndex: Integer; out pHorario: TDateTime; pTolerancia : Byte; pSegunda, pTerca, p Quarta, pQuinta, pSexta, pSabado, pFeriado, pStatus : WordBool);**

- **Add_Escala (pThreadIndex: Integer; pEscala: SEscala; pIndexEscala : Byte) : WordBool;**

Este método adiciona uma Escala ao cachê de Escalas para o método EnviaHorários que esvazia o cachê ao ser executado. ***

> A Escala é um conjunto de Horários.
> Ex: '000001002' contém 3 horários
> O primeiro horário possui index = 0
> Quando um dia não possue horário utilize "###"

**Add_Escala (pThreadIndex: Integer; p DataInicio : TDateTime; pHorarios : WideString);**

- **EnviaHorarios(pThreadIndex: Integer): WordBool;**

Este método executa o envio dos horários anteriormente carregados com o método Add_Horario, considerando o tamanho do setor.

```
EnviaHorarios(pThreadIndex: Integer);
```

- **RecebeHorarios(pThreadIndex: Integer): WordBool;**

Este método executa a recepção dos períodos e disponibiliza no método Rec_Horario.

*** *A escala é um conjunto de horário tendo um dia inicial e um horário para cada dia do conjunto, consultado de forma cíclica a partir da data inicial. Para utiliza Escalas habilite em SConfiguracao.Controles*

```
RecebeHorarios(pThreadIndex: Integer);
```

- **Rec_Horario(pThreadIndex: Integer; out pPeriodos: WideString): WordBool;**

Este método retorna um Horário do cachê de Horários carregado pelo método RecebeHorários, ao receber um Horário ele é automaticamente excluído do cachê.

```
Rec_Horario(pThreadIndex: Integer; out pPeriodos : WideString);
```

- **Rec_Escala(pThreadIndex: Integer; out pEscala: SEscala): WordBool;**

Este método retorna uma Escala do cachê de Escala carre gado pelo método RecebeHorários, ao receber uma Escala ela é automaticamente excluída do cachê.

```
Rec_Escala(pThreadIndex: Integer; out pDataInicio : TDateTime;
pHorarios : WideString; out pStatus : WordBool);
```

- **Add_ItemAcesso(pThreadIndex: Integer; pItemAcesso: SItemAcesso);**

Este método adiciona um Item de Acesso ao cachê da Lista de Acesso para o método EnviaListaAcesso que esvazia o cachê ao ser executado.

## Estrutura SItemAcesso

| Atributo | Tipo | Descrição |
|---|---|---|
| Matricula | String | Matrícula de realizou a marcação formatada em 20 digitos |
| IndexHorario | Inteiro | Indicador do horário ou escala que o colaborador deve respeitar |
| Acesso | SAcessoOffline | Definição da forma de acesso do usuário, dentro da enumeração:<br>• **cafLiberado**<br>• **cafNegado**<br>• **cafHorario**<br>• **cafEscala**<br>• **cafFaixa** |
| VerificarDigital | Boleano | Determina se usuário deve verificar digital para realizar a marcação |
| PeriodoBloqueio | SPeriodoBloqueio | Considerações para período de bloqueio segundo a estrutura SPeriodoBloqueio |
| Reles | SReles | Reles que devem ser acionados no acesso do usuário |
| Master | Boleano | Determina se usuário é master |
| Visitante | Boleano | Determina se usuário é visitante |
| Controladores | SControladores | Determina quais controladores o usuário pode utilizar.<br>Dentro da estrutura com valores<br>**Control1, Control2,... Control8** todos boolean |

## Estrutura SPeriodoBloqueio

| Atributo | Tipo | Descrição |
|---|---|---|
| Habilitado | Boleano | Determina se período de bloqueio será usado |
| Inicio | Date | Indicador do horário ou escala que o colaborador deve respeitar |
| Final | Date | Determina se usuário deve verificar digital para realizar a marcação |

**Add_ItemAcesso(pThreadIndex: Integer; p Matricula : WideString; pIndexHorario : Byte; pAcesso : SAcessoOffline; pPerBloqInicio, pPerBloqFinal : TDateTime; pPerBloqHabilitado, pAcionaRele1, pAcionaRele2, pAcionaRele3, pVerificarDigital, pMaster, pVisitante : WordBool);**

Para informações de como configurar e utilizar **Faixas de Acesso** verifique o anexo ao final do arquivo.

- **EnviaListaAcesso(pThreadIndex: Integer): WordBool;**

Este método executa o envio da Lista de Acesso anteri ormente carregada com o método Add_ItemAcesso, considerando o tamanho do setor.

```
EnviaListaAcesso(pThreadIndex: Integer): WordBool;
```

- **RecebeListaAcesso(pThreadIndex: Integer): WordBool;**

Este método executa a recepção da Lista de Acesso e disponibiliza no mé todo Rec_ItemAcesso.

```
RecebeListaAcesso(pThreadIndex: Integer): WordBool;
```

- **Rec_ItemAcesso(pThreadIndex: Integer; out pItemAcesso: SItemAcesso): WordBool;**

Este método retorna um Item de Acesso do cachê da Lista de Acesso carregado pelo método RecebeListaAcesso, ao receber um Item de Acesso ele é automaticamente excluído do cachê.

```
Rec_ItemAcesso(pThreadIndex: Integer; out pMatricula :
WideString; out pIndexHorario : Byte; out pAcesso :
SAcessoOffline; out pPerBloqInicio, pPerBloqFinal : TDateTime; out
pPerBloqHabilitado, pAcionaRele1, pAcionaRele2, pAcionaRele3,
pVerificarDigital, pMaster, pVisitante , pStatus : WordBool) :
WordBool;
```

- **Add_MsgEspec(pThreadIndex: Integer; pMsgEspec: SMsgEspecifica);**

Este método adiciona uma Mensagem Especifica ao cachê de Mensag ens Especificas para o método EnviaMsgEspec que esvazia o cachê ao ser executado.

```
Add_MsgEspec(pThreadIndex: Integer; pMsgEstilo : SMsgEstilo;
pMsgTempo : Byte; pMsg Linha1, pMsgLinha2, pMatriculas :
WideString; pData : TDateTime; pTodosDias : WordBool);
```

- **EnviaMsgsEspecificas(pThreadIndex: Integer): WordBool;**

Este método executa o envio das Mensagens Especificas anteriormente carregada com o método Add_MsgEspec, considerando o tamanho do setor.

**EnviaMsgsEspecificas(pThreadIndex: Integer): WordBool;**

- **RecebeMsgsEspecificas(pThreadIndex: Integer): WordBool;**

Este método executa a recepção das Mensagens Especificas e disponibiliza no método Rec_MsgEspec.

**RecebeMsgsEspecificas(pThreadIndex: Integer): WordBool;**

- **Rec_MsgEspec(pThreadIndex: Integer; out pMsgEspec: SMsgEspecifica): WordBool;**

Este método retorna uma Mensagem Especifica do cachê de Mensagens Especificas carregadas pelo método RecebeMsgEspec, ao receber uma Mensagem Especifica ela é automaticamente excluído do cachê.

- **Rec_MsgEspec(pThreadIndex: Integ er; out pMsgStatus : SMsgStatus; out pMsgTempo : Byte; out pMsg Linha1, pMsgLinha2 , pMatriculas : WideString; out pData : TDateTime; out pTodosDias , pStatus : WordBool);**

# Trabalhando offline

- **ExistemRegistros(pThreadIndex: Integer; out pExistem: WordBool): WordBool;**

  Este método retorna se existem registros no equipamento. Melhor usado na comunicação USB que opera com o método RecebeQtRegistros limitado.

  **ExistemRegistros(pThreadIndex: Integer; out pExistem: WordBool): WordBool;**

- **RecebeQtRegistros(pThreadIndex: Integer; out pQtRegs : Integer): WordBool;**

  Este método retorna a quantidade de registros de batidas no equipamento.
  No caso de USB retorna somente 0 ou 1 quando existirem registros.

  **RecebeQtRegistros(pThreadIndex: Integer; out pQtRegs: Integer): WordBool;**

  Alternativo retorno -1 significa falha na operação.

- **RecebePacote (pThreadIndex: Integer): WordBool;**

  Permite maior controle e estabilidade para coleta. Este método executa a coleta do próximo pacote com até 100 registros do equipamento. Não realiza a deleção do pacote automaticamente.

  **RecebePacote (pThreadIndex: Integer): WordBool;**

- **QuantRegsColetados(pThreadIndex: Integer): WordBool;**

  Este método retorna a quantidade de registros coletados, apagados do equipamento *, e aguardando recuperação da aplicação para gravação definitiva.

  **QuantRegsColetados(pThreadIndex: Integer): WordBool;**

- **RegistroOff(pThreadIndex: Integer; out pRegistro: SRegistro): WordBool;**

Este método retorna os registros coletados pelo método RecebeRegistros e armazenados no cachê do ActiveX, cada registros retornado é automaticamente excluído, ao retornar Falso indica que acabaram os registros.

Para saber mais sobre a estrutura SRegistro, veja a tabela na página "Trabalhando on - line".

```
RegistroOff(pThreadIndex: Integer; out pNumeroRelogio, pFuncao :
Byte; out pMatricula : WideString; out pDataHora : TDateTime; out
pFlag : SFlagRegistro; out pSaida, pMasterLiberou, pFuncaoLiberou,
pAcessoNegado : WordBool; out pFonteEntrada : SFonteEntrada;
pTipoNegado : STipoNegado);
```

Em caso de interface alternativo o RegistroOff não posse retorno o que torna necessário o uso do método QuantRegsColetados para saber quantos registros estão em cache.

- **ApagaUltimoPacote (pThreadIndex: Integer): WordBool;**

Este método exclui o último pacote de registros recebidos. Sem apagar o ultimo pacote não é possível receber o próximo pacote.

```
ApagaUltimoPacote (pThreadIndex: Integer): WordBool;
```

- **RecuperaRegistros(pThreadIndex: Integer): WordBool;**

Este método retorna ativos os registros anteriormente coletados com o método RecebeRegistros e ainda disponíveis no equipamento.

```
RecuperaRegistros(pThreadIndex: Integer): WordBool;
```

* Os registros apagados do equipamentos podem ser recuperados com o método RecuperaRegistros.
* Os métodos da sessão Trabalhando Offline estão na ordem de uso correto, não sendo obrigatório o uso de todos os métodos.

- **ColetaEventos(pThreadIndex: Integer; const pPathAFD: WideString): WordBool;**

Este método coleta os eventos do equipamento e salva o AFD (Arquivo Fonte de Dados) no caminho especificado. Ao finalizar a coleta será disparado o evento OnColetaEventos com o resultado da operação, quantidade de eventos salvos e caminho do AFD. Disponível apenas para Orion 6.

```
ColetaEventos(pThreadIndex: Integer; const pPathAFD: WideString): WordBool;
```

- **ColetaEventosEx(pThreadIndex: Integer; const pPathAFD: WideString; pData: TDateTime; pEmpregador: SEmpregador): WordBool;**

Este método coleta os eventos do equipamento e salva o AFD (Arquivo Fonte de Dados) no caminho especificado. Através do parâmetro "pData" o integrador poderá estabelecer um filtro de data e hora para a coleta. Serão retornados os eventos com data igual ou maior à data informada. Ao finalizar a coleta será disparado o evento OnColetaEventos com o resultado da operação, quantidade de eventos salvos e caminho do AFD. Disponível apenas para Orion 6.

```
ColetaEventosEx(pThreadIndex: Integer; const pPathAFD: WideString;
    pData: TDateTime; const pRazaoSocial, pLocal, pDocumento,
    pCEI: WideString; pIdEmpregador: SIdEmpregador): WordB ool;
```

- **PararColetaEventos(pThreadIndex: Integer): WordBool;**

Este método finaliza a coleta de eventos. Nenhum arquivo de coleta será gerado. Ao finalizar a coleta será disparado o evento OnColetaEventos informando que a coleta foi cancelada. Disponível apenas para Orion 6.

```
PararColetaEventos(pThreadIndex: Integer): WordBool;
```

# Trabalhando on-line

- **OnStatus (pThreadIndex, pDeviceId, pStatus: Integer);**

Este evento é disparado quando ocorre alteração de status em algum equipamento. O Parâmetro DeviceID pode ser o número do equipamento comunicando em 485 ou controlador que gerou o evento no caso de concetrador.

## Enumeração SDeviceStatus

| Range de valores |
|---|
| • cdsOnline → Informa que equipamento voltou a permitir conexão<br>• cdsOffline → Informa que equipamento está ind isponível para conexão<br>• cdsPortaAberta →Informa que porta está aberta<br>• cdsBotao → Informa que botão foi pressionado |

**OnStatus (pThreadIndex, pDeviceId , pStatus: Integer);**

- **OnRegistro(pThreadIndex: Integer);**

Este evento é disparado quando um equipamento tr abalhando em modo online recebe um pedido do equipamento.

**OnRegistro(pThreadIndex: Integer);**

- **RegistroOn(pThreadIndex: Integer; out pRegistro : SRegistro);**

Este método retorna o registro on-line durante o evento OnRegistro.

**RegistroOn(pThreadIndex: Integ er; out pNumeroRelogio, pFuncao : Byte; out pMatricula : WideString; out pDataHora : TDateTime; out pFlag : SFlagRegistro; out pSaida, pMasterLiberou, pFuncaoLiberou, pAcessoNegado : WordBool; out pFonteEntrada : SFonteEntrada; out pTipoNegado : STipoNegad o);**

## Estrutura SRegistro

| Atributo | Tipo | Descrição |
|---|---|---|
| Matricula | String | Matrícula que realizou a marcação |
| DataHora | DateTime | Data e Hora da marcação (no equipamento) |
| Funcao | Byte | Função utilizada (255 significa sem função) |
| Flag | SFlagRegistro | Enumeração com valores possíveis:<br>**sfrGirou :** Ocorreu giro do braço da catraca<br>**sfrNaoGirou :** Não houve giro |
| NumeroRelogio | Byte | Numero do equipamento onde ocorreu o registro (tcpip sempre 1) |
| Tipo | STipoBilhete | Diversas informações adicionais sobre a marcação |
| IDControlador | Byte | Usado para Controlador:<br>Identifica o controlador que originou o registro. |
| IDSensor | Byte | Usado para Controlador.<br>Identifica o sensor que originou o registro. |

## Estrutura STipoBilhete

| Atributo | Tipo | Descrição |
|---|---|---|
| Saida | Boolean | Determina sentido do giro, valor falso significa entrada. |
| MasterLiberou | Boolean | Informa de registro foi liberado por master |
| FonteEntrada | Byte | Indica o periférico que gerou a marcação, estando dentro da enumeração:<br>**cfeTeclado, cfeCracha, cfeDigital11, cfeDigital1N** |
| FuncaoLiberou | Boolean | Informa de marcação foi através de liberação por função |
| AcessoNegado | Boolean | Informa se marcação teve acesso negado |
| TipoNegado | STipoNegado | Informa motivo do acesso negado off-line, dentro da enumeração:<br>**ctnSenha, ctnNivel, ctnHorario, ctnEmpresa, ctnNenhum** |

- **RespostaOn(pThreadIndex: Integer; pResposta : SResposta);**

Este método envia a resposta on-line durante o evento OnRegistro.

**RespostaOn(pThreadIndex: Integer; pAcesso : SAcessoO nline; pMensagem : WideString; pTempo : Byte);**

**RespostaOnB(pThread Index: Integer; pAcesso Liberado : WordBool; pIdControlador, pTempoRele1, pTempoRele2, pTempoRele3 : Byte);**

## Estrutura SResposta

| Atributo | Tipo | Descrição | |
|---|---|---|---|
| Acesso | SAcessoOnline | Determina resposta física ao usuário, atendendo a enumeração:<br>• **canNegado:** Acesso negado<br>• **canLibEntrada:** Acesso liberado no sentido de entrada<br>• **canLibSaida:** Acesso liberado no sentido de saída<br>• **canRevista:** Acesso liberado no sentido de saída informando revista<br>• **canAmbosLados:** Acesso liberado para ambos os lados<br>• **canMensagem:** Idem a canNegado porém sem alerta sonoro | |
| Mensagem | String | Mensagem para exibição no display com 32 caracteres | Exclusivo Relógio e Catraca |
| Tempo | Byte | Tempo em segundos de exibição da mensagem e acionamento de catraca | |
| IDControlador | Byte | Index do controlador, pode ser obtido na estrutura de registro | Exclusivo Concentrador |
| TempoRele1 | Byte | Tempo em segundos de acionamento do Rele 1 | |
| TempoRele2 | Byte | Tempo em segundos de acionamento do Rele 2 | |
| TempoRele3 | Byte | Tempo em segundos de acionamento do Rele 3 | |

- **RespostaStatus(pThreadIndex: Integer; pIndexMensagem : Integer);**

Este método ativa uma função específica, informada no parâmetro pIndexMensagem, do equipamento como resposta. A mensagem específica deve ser previamente enviada.

**RespostaStatus(pThreadIndex: Integer; pIndexMensagem : Integer);**

- **OnExistOff(pThreadIndex: Integer; pQtRegs: Integer);**

Este evento é disparado quando um equipamento estiver com Modo de Comunicação em OnOff ou OnOffCtrl e possuir registros em offline não coletados. O alerta será chamado com intervalo de 5 minutos para não sobrecarregar a aplicação.

**OnExistOff(pThreadIndex: Integer; pQtRegs: Integer);**

- **OnImagemDSP(pThreadIndex: Integer; const pImagem: WideString);**

Este evento é disparado quando um equipamento estiver com Modo de Biometria com Imagem ativado. Esta opção pode ser ativada através da variável "BiometriaImagem" na estrutura de configuração do equipamento. Esta opção não pode ser utilizada com a opção "AutoOn" ativada.

**OnImagemDSP(pThreadIndex: Integer; pImagem: WideString);**

- **OnColetaEventos(pThreadIndex : Integer; pResultado : Boolean; pEventos : integer; pPathAFD : WideString);**

Este evento é disparado no final da coleta de eventos. Disponível para Orion 6.

**OnImagemDSP(pThreadIndex: Integer; pResultado : Boolean; pEventos : integer; pPathAFD : WideString);**

# Cadastro de Digitais Modelo 1 e 2

- **Bio_CriaDigitalM1M2(const pMatricula: WideString; pFinger: Byte; pMaster: WordBool; out pTemplateF: STemplate7x; out pTemplateFl: STemplate7x; out pTemplateFH: STemplate7x): WordBool;**

Este método retorna digitais para tipos de biometria F, FL, FH a partir de um “Hamster” modelo 1 ou 2.

# Cadastro de Digitais Modelo 3

Para operação com biometria S a partir de um “Hamster” você deverá usar o objeto Hamster (outra interface do Kernel). Abaixo os métodos do objeto Hamster.

- **Capture(out pTemplate: WideString) : WordBool;**

Este método retorna a digital para tipo de biometria S a partir de um “Hamster” modelo 3.

**CaptureNet(out pTemplate: WideString; out pStatus: WordBool);**

- **CaptureContinuous(out pTemplate: WideString) : WordBool;**

Este método retorna a digital para tipo de biometria S a partir de um “Hamster” modelo 3, porém sem tempo limite para colocação do dedo.
Este método deve ser executado em processo separado por trabalhar de forma modal.

**CaptureContinuousNet(out pTemplate: WideString; out pStatus: WordBool);**

- **CaptureImage(out pImgTemplate: WideString; out pStatus: WordBool);**

Este método retorna uma WideString que contém as informações de uma imagem .bmp com o desenho da digital capturada. Para ser utilizada a template deve ter sido capturada anteriormente através dos métodos de captura de template.

- **AbortCapturing;**

Este método aborta a captura de digital, sendo recomendado para utilização com o método CaptureContinuous.

- **Verify (pTemplate: WideString) : WordBool;**

Este método captura outra template e executa a comparação com a template informada.

- **Timeout : Integer;**

Esta propriedade define o timeout de captura. O Padrão é 0 ou seja sem timeout;

- **Brightness : Integer;**

Configuração de brilho, aceita valores entre 0 e 255;

- **Sensitivity: Integer;**

Configuração de sensibilidade, aceita valores entre 0 e 7;

- **Quality: Integer;**

Qualidade da imagem obtida ao capturar template. Valores entre 0 e 100;

- **SetImagem(pHandle, pTop, pLeft, pHeight, pWidth : Integer);**

  Configuração de objeto (em Delphi recomenda-se panel) para exibir imagem durante a captura. Deve ser utilizada antes de iniciar a captura.

- **SetUser(pTemplate, pMatricula : WideString; pMaster : Boolean) : WideString;**

  Determina usuário para template. Necessário para envio ao equipamento.

# Manutenção de Digitais

- **Bio_RecListaUsuarios(pThreadIndex: Integer): WordBool;**

Este método retorna a lista dos usuários cadastrados no equipamento.

**Bio_RecListaUsuarios(pThreadIndex: Integer): WordBool;**

- **Bio_GetUsuario(pThreadIndex: Integer; out pUsuario: SUsuarioBioEx): WordBool;**

Este método retorna um item da lista de usuários recebida com o método RecListaUsuarios, ao quando um usuário é recebido automaticamente é excluído do cachê.

Ex:
Se Bio_RecListaUsuario() Então
  Enquanto Bio_GetUsuario(Usuário) Faça
        Impressão do Usuário

**Bio_RecListaUsuarios(pThreadIndex: Integer ; pMatricula, pId : WideString; pDedo : Byte; pMaster : WordBool ): WordBool;**

- **Bio_UsuariosQuant(pThreadIndex: Integer; out pQuantidade: Integer): WordBool;**

Este método retorna a quantidade de digitais armazenadas no equipamento.

**Bio_UsuariosQuant(pThreadIndex: Integer; out pQuantidade: Integer): WordBool;**

- **Bio_UsuarioExiste(pThreadIndex: Integer; const pUsuarioID: WideString; out pExiste: WordBool): WordBool;**

Este método retorna se usuários esta cadastrado no equipamento.

**Bio_UsuarioExiste(pThreadIndex: Integer; const pUsuarioID: WideString; out pExiste: WordBool): WordBool;**

- **Bio_RecTemplate(pThreadIndex: Integer; const pUsuarioID: WideString; out pTemplate: WideString): WordBool;**

Este método retorna a digital de um usuário cadastrado.

**Bio_RecTemplate(pThreadIndex: Integer; const pUsuarioID: WideString; out pTemplate: WideString): WordBool;**

- **Bio_RecUsuario(pThreadIndex: Integer; pPrimeiro: WordBool; out pUsuario: SUsuarioBioEx): WordBool;**

Este método retorna a digital e informações de um usuário do equipamento. O Parâmetro pPrimeiro é usado para indicar o inicio da operação de recepção dos usuários, Este método é usado quando o tipo de biometria não suporta listar todos os usuários. Os limites podem ser consultados com o metódo Bio_GetMaxQuantLista.

Ex:
Variável PrimeiroRegistro ← Verdadeiro
Enquanto Bio_RecUsuario(Usuário) Faça
Inicio
  PrimeiroRegistro ← Falso;
  Imprime Usuário;
Fim

**Bio_RecUsuario(pThreadIndex: Integer; pPrimeiro: WordBo ol; out pMatricula, pTemplate, pID: WideString; out pMaster : WordBool; out pDedo : Byte): WordBool;**

- **Bio_GetMaxQuantLista(pThreadIndex : Integer; out pQt : Integer): WordBool;**

Este método retorna a quantidade máxima de usuários para listagem de acordo com a biometria, 0 Indica que o limite é a capacidade de armazenamento.

- **Bio_EnvTemplate(pThreadIndex: Integer; const pTemplate: WideString): WordBool;**

Este método cadastra uma digital no equipamento;
Obs: Todas as digitais do usuário devem ser excluídas a ntes de envia-las;

* Observação válida para biometria cb_S;

**Bio_EnvTemplate(pThreadIndex: Integer; const pTemplate: WideString): WordBool;**

- **Bio_DelTemplate(pThreadIndex: Integer; const pUsuarioID: WideString; pFingerOnly : WordBool): WordBool;**

Este método exclui uma digital do equipamento
Obs: O parâmetro pFingerOnly define se deve exclui somente uma digital ou todas as do usuário*.

* Observação válida para biometria cb_S;

**Bio_DelTemplate(pThreadIndex: Integer; const pUsuarioID: WideString; pFingerOnly : WordBool): WordBool;**

- **Bio_DelTemplateTodas(pThreadIndex: Integer): WordBool;**

Este método exclui todas as digitais do equipamento.

**Bio_DelTemplateTodas(pThreadIndex: Integer): WordBool;**

- **Bio_GeraUserID(pBiometria: SBiometria; const pMatricula: WideString; pDedo: Byte; pMaster: WordBool): WideString;**

Este método gera um ID de usuário de acordo com os parâmetros informados . O ID do usuário é necessário em diversos métodos de manutenção biométrica.

**Bio_GeraUserID(pBiometria: SBiometria; const pMatricula: WideString; pDedo: Byte; pMaster: WordBool): WideString;**

- **Bio_UsuariosQuantLivre(pThreadIndex: Integer; out pQuantidade: Integer): WordBool;**

Este método retorna o espaço disponível para novos cadastrados no equipamento.
**Suportado somente em biometria cb_S.**

**Bio_UsuariosQuantLivre(pThreadIndex: Integer; out pQuantidade: Integer): WordBool;**

# Manutenção de Digitais Avançada

- **Bio_RecConfiguracaoF_FL(pThreadIndex: Integer; out pConfig: SDspcfg_F_FL): WordBool;**

  Este método permite receber a configuração do módulo biométrico modelos F ou Fl.

- **Bio_EnvConfiguracaoF_FL(pThreadIndex: Integer; pConfig: SDspcfg_F_FL): WordBool;**

  Este método permite enviar configurações para os módulos F ou FL.

- **Bio_RecConfiguracaoS(pThreadIndex: Integer; out pConfig: SDspcfg_S): WordBool;**

  Este método permite receber configurações do módulo S.

  **RecebeConfigDSP (pThreadIndex: Integer; out pNivelSeguranca, pVelocidade, pSensibilidade, pQualidadeImagem : Integer; out pCondicaoIluminacao, pStatus: WordBool);**

- **Bio_EnvConfiguracaoS(pThreadIndex: Integer; pConfig: SDspcfg_S): WordBool;**

  Este método permite enviar configurações para o módulo S.

  **EnviaConfigDSP (pThreadIndex: Integer; pNivelSeguranca, pVelocidade, pSensibilidade, pQualidadeImagem : Integer; pCondicaoIluminacao: WordBool) : WordBool;**

- **Bio_CfgDefaultF_FL(pTipo: SCfgDspPadrao): SDspcfg_F_FL;**

  Este método carrega estruturas de configuração padrão para o módulo F ou FL. Será necessário o método Bio_EnvConfiguracaoF_FL para efetivar a operação.

# Biometria Online

Este recurso estará disponível mediante licença, não licenciado será possível carregar 5 digitais.

- **Bio_CarregaTemplate (pTemplateS : WideString): WordBool;**

  Este método carrega a template armazenada para matching online no kernel, a matrícula encontrada será passada de forma transparent e no evento OnRegistro. A template passada como parâmetro deve ter utilizado o método SetUser da interface Hamster.

  **Bio_CarregaTemplate (pTemplateS : WideString): WordBool;**

- **Bio_DropTemplates;**

  Este método excluir todas as templates carrega das da memória.

  **Bio_DropTemplates;**

- **Bio_ProcuraTemplate (pTemplate, pMatricula : WideString): WideString;**

  Este método realiza busca nas templates adicionadas com o método CarregaTemplate retornando a matrícula caso localize a digital, modo 1 -N. Também é possível realizar busca 1-1 informando no parâmetro matrícula a digital desejada.

  **Bio_ProcuraTemplate (pTemplate, pMatricula : WideString): WideString;**

- **SetSecurityLevel(pLevel : Byte);**

  Este método determina o nível de segurança da verificação online, o pad rão é 5 podendo variar de 1 a 7, quanto maior mais segura e lenta será a busca.

  **SetSecurityLevel(pLevel : Byte);**

- **SetSearchTimeout (pSegundos : Integer);**

  Este método determina o tempo limite de busca de template no banco, o valor 0 deixa busca sem limite de tempo.

  **SetSearchTimeout (pSegundos : Integer);**

# Exportação e Importação de Dados

- **ExportConfiguracao(const pCaminho: WideString; pConfiguracao: SConfiguracao): WordBool;**

  Este método salva uma configuração de equipamento em arquivo .

- **ImportConfiguracao(const pCaminho: WideString; out pConfiguracao: SConfiguracao): WordBool;**

  Este método carrega uma configuração de equipamento de um arquivo.

- **SaveAsTemplate7x(const pCaminho: WideString; pTemplate: STemplate7x): WordBool;**

  Este método salva uma template 7x em arquivo. A template 7x é uma estrutura genérica para armazenamento de qualquer formato de digital suportado pelos equipamentos Henry.

- **OpenTemplate7x(const pCaminho: WideString; out pTemplate: STemplate7x): WordBool;**

  Este método carrega uma template 7x de um arquivo.

- **USB_RecebeCartucho(pThreadIndex: Integer; const pFileName: WideString): WordBool;**

  Este método copia um cartucho USB Henry para um arquivo de memória .

- **USB_EnviaCartucho(pThreadIndex: Integer; const pFileName: WideString): WordBool;**

  Este método grava um arquivo de memória em cartucho USB Henry.

# Redistribuição

Para distribuir o kernel7x e seus componentes juntamente com sua aplicação, garantindo compatibilidade total de métodos e funcionamento, basta adicionar ao seu instalador o pacote **Redist**, este pacote instala e registra as bibliotecas no sistema.

Além de drivers para operação com hamster e USB.

A execução do redist suporta personalização, a mais importante para funcionamento da sua aplicação é personalizar o diretório de instalação, segue exemplo de execução que pode ser escrita em um arquivo bat caso você não possua um instalador.

**Kernel7x_v7.2.0.22_Redist.exe /DIR="C:\Arquivos de programas\MinhaEmpresa\MeuPrograma"**

Para outras customizações procure na internet por “inno setup command line”.

Tenha seu aplicativo compilado com a versão do kernel que será usado.

Para obter este pacote acesse www.henry.com.br, entre como integrador e baixe a versão mais recente.

É recomendável manter sua integração com o Kernel sempre atualizada com a última versão disponível.

# Constantes de erro

```
CARD7X_ERRO_NONE                    = 0000; //Operação realizada com sucesso
CARD7X_ERRO_TIMEOUT                  = 0001; //Ultrapassado tempo máximo de espera
CARD7X_ERRO_FIRMWARE                = 0002; //Firmware não suporta comando
CARD7X_ERRO_CHECKSUN_HEADER          = 0003; //Cabeçalho do pacote inválido
CARD7X_ERRO_CHECKSUN_DATA           = 0004; //Dados do pacote inválido
CARD7X_ERRO_ONLINEPRIORITY           = 0005; //Operação abortada por solicitação online
CARD7X_ERRO_CONEXAO                 = 0006; //Não foi possível conectar com o equipamento
CARD7X_ERRO_QUEBRADADOS             = 0007; //Não foi possível receber todos os dados esperados
CARD7X_ERRO_CARDHEADER              = 0008; //Cabeçalho do pacote é inválido
CARD7X_ERRO_PARTITION_SMALL         = 0009; //Particionamento não suporta quantidade de dados
CARD7X_ERRO_COMTYPE                 = 0011; //Tipo de comunicação não suporta comando
CARD7X_ERRO_REGS_EMPTY              = 0012; //Sem registros
CARD7X_ERRO_ITEMS_FULL              = 0013; //Quantidade de itens limite atingida
CARD7X_ERRO_KERNEL_EXCEPTION        = 0014; //Número/ID do equipamento inválido
CARD7X_ERRO_ITEMS_NOTFOUND          = 0015; //Item solicitado não existe
CARD7X_ERRO_ITEMS_DUPLICATED        = 0016; //Existem itens duplicados na lista
CARD7X_ERRO_ICMP_FAIL               = 0017; //Ping não foi realizado com sucesso
CARD7X_ERRO_ICMP_TIMEOUT            = 0018; //Detectada lentidão na conexão podendo causar perda de dados
CARD7X_ERRO_NEXISTE_LISTA           = 0019; //Registro não existe na lista
CARD7X_ERRO_EXISTE_LISTA            = 0020; //Registro existe na lista
CARD7X_ERRO_LISTA_CHEIA             = 0021; //Lista de registros está cheia
CARD7X_ERRO_EVENTO_NAO_EXISTE       = 0022; //Evento não existe
CARD7X_ERRO_PARAMETROS_INVALIDOS    = 0023; //Parâmetros inválidos
CARD7X_ERRO_CANCELAR_COLETA         = 0024; //Coleta cancelada

CARD7X_ERRO_IO_ENVIO                = 0101; //Não foi possível enviar os dados corretamente
CARD7X_ERRO_IO_RECEPCAO             = 0102; //Não foi possível receber os dados corretamente

CARD7X_ERRO_DSP_EXCEPTION           = 0201; //Módulo biométrico não aceitou o comando
CARD7X_ERRO_DSP_EMPTY               = 0202; //Módulo biométrico sem usuários
CARD7X_ERRO_DSP_OVERBUFFERSIZE      = 0203; //Capacidade do buffer para módulo biométrico ultrapassada

CARD7X_ERRO_INVALID_SERIALPORT      = 0301; //Porta serial inválida
CARD7X_ERRO_INVALID_METHOD          = 0302; //Comando inválido para a configuração atual
CARD7X_ERRO_INVALID_PARTITION        = 0303; //Particionamento não é válido
```

```
CARD7X_ERRO_INVALID_THREAD                = 0304; //Thread não foi encontrada
CARD7X_ERRO_INVALID_TEMPLATE              = 0305; //Erro de template inválida
CARD7X_ERRO_INVALID_FACILITY               = 0306; //Facility code inválido
CARD7X_ERRO_INVALID_CONTROLADORID         = 0307; //ID do controlador inválido
CARD7X_ERRO_INVALID_COUNTPERIODOS         = 0308; //Quantidade de períodos no horário inválida
CARD7X_ERRO_INVALID_GPRSPORT              = 0309; //A porta definida já está em uso por outro equipamento
CARD7X_ERRO_INVALID_SERIALINUSE           = 0310; //A porta de comunicação está sendo utilizada por outro processo

CARD7X_ERRO_USB_INVALID_IDNUM             = 0401; //Número/ID do equipamento inválido
CARD7X_ERRO_USB_INVALID_IXREG             = 0402; //Dados do Cartucho USB inconsistentes
CARD7X_ERRO_USB_INVALID_IXCHIP            = 0403; //Dados do Cartucho USB inconsistentes
CARD7X_ERRO_USB_INVALID_NUMDIGITOS        = 0404; //Dados do Cartucho USB inconsistentes
CARD7X_ERRO_USB_INVALID_FILESIZE          = 0405; //Arquivo USB inválido para capacidade do cartucho
CARD7X_ERRO_USB_INVALID_COUNTCHIPS        = 0406; //Número de chips inválido
CARD7X_ERRO_USB_PROTOCOL                  = 0407; //Falha na montagem do protocolo USB

 CARD7X_ERRO_HEADER_RESP                  = 0500; //Resposta inválida recebida
CARD7X_ERRO_HEADER_CMD                     = 0501; //Comando inválido recebido pelo relógio
CARD7X_ERRO_HEADER_VERSAO                  = 0502; //Versão incompatível com o Relógio
CARD7X_ERRO_HEADER_STATUS                  = 0503; //Status errado foi recebido pelo relógio
CARD7X_ERRO_HEADER_TAMREG                 = 0504; //Tamanho dos Reg. diferentes foi recebido pelo relógio
CARD7X_ERRO_HEADER_NUMREG                  = 0505; //Número dos Reg. inválido foi recebido pelo relógio
CARD7X_ERRO_HEADER_CHECKSUN                = 0506; //Relógio recebeu cabeçalho com erro de check sum

CARD7X_ERRO_BIO_KEY_OVERLIMIT             = 0601; //Chave de biometria excedeu o limite
CARD7X_ERRO_BIO_INVALID_TEMPLATE          = 0602; //Formato da template invalido
CARD7X_ERRO_IMAGE_NO_LICENSE              = 0603; //Kernel7x não está licenciado para este serviço

CARD7X_ERROR_CONFIG_DIGITOS_REGS          = 0700; //Colete registros antes de executar essa operação
```

# Anexos

- **Alterações kernel7x Alternativo 7.2.0.22**

No intuito de facilitar a integração e prover agilidade no desenvolvimento f oram alterados os parâmetros e implementadas novas funções para o Kernel7x Alternativo. Abaixo segue a relação com as funções e as modificações feitas.

## Novas rotinas

Foram criadas quatro novas rotinas.

### Método RecebeConfigDSP

Método criado para receber configurações do módulo biométrico do equipamento.
pStatus retorna verdadeiro se a rotina foi executada com sucesso.

**procedure** TAlternativo.RecebeConfigDSP(pThreadIndex: Integer;
**out** pNivelSeguranca, pVelocidade, pSensibilidade, pQualidadeImagem: Integer;
**out** pCondicaoIluminacao, pStatus: WordBool);

| | | | |
|---|---|---|---|
| TAlternativo.RecebeConfigDSP | pThreadIndex | Integer | Índice da thread a ser acessada |
| | pNivelSeguranca | Integer | Nível de segurança da verificação |
| | pVelocidade | Integer | Velocidade da verificação |
| | pSensibilidade | Integer | Sensibilidade do leitor |
| | pQualidadeImagem | Integer | Qualidade da imagem |
| | pCondicaoIluminacao | Boolean | Condição de iluminação |
| | pStatus | Boolean | Situação da execução da proce dure. True = Executou com sucesso |

---

### Método EnviaConfigDSP

Método criado para enviar configurações ao módulo biométrico do equipamento.

**function** TAlternativo.Get_EnviaConfigDSP(pThreadIndex, pNivelSeguranca,
pVelocidade, pSensibilidade, pQualidadeImage m: Integer;
pCondicaoIluminacao: WordBool): WordBool;

| | | | |
|---|---|---|---|
| TAlternativo.Get_EnviaConfigDSP | pThreadIndex | Integer | Índice da thread a ser acessada |
| | pNivelSeguranca | Integer | Nível de segurança da verificação |
| **Retorno : Boolean** | pVelocidade | Integer | Velocidade da verificação |
| | pSensibilidade | Integer | Sensibilidade do leitor |
| | pQualidadeImagem | Integer | Qualidade da imagem |
| | pCondicaoIluminacao | Boolean | Condição de iluminação |

---

## Método CaptureNet

Método criado para capturar a digital do Hamster com timeout definido. Retorna a digital e uma variável que indica se a operação foi realizada com sucesso.

**procedure** THamster.CaptureNet(out pTemplate: WideString; out pStatus: WordBool);

| | | | |
|---|---|---|---|
| TAlternativo.CaptureNet | pTemplate | WideString | Retorna a digital em forma de widestring |
| | pStatus | Boolean | Situação da execução da procedure. True = Executou com sucesso |

## Método CaptureContinuousNet

Método criado para capturar a digital do Hamster sem timeout. Retorna a digital e uma variável que indica se a operação foi realizada com sucess o.

**procedure** THamster.CaptureContinuousNet(out pTemplate: WideString;
out pStatus: WordBool);

| | | | |
|---|---|---|---|
| TAlternativo.CaptureContinuousNet | pTemplate | WideString | Retorna a digital em forma de widestring |
| | pStatus | Boolean | Situação da execução da procedure. True = Executou com sucesso |

# Rotinas alteradas

## Método RecebeMsgPadrao

Alterado método “RecebeMsgPadrao”. Foi alterado o tipo de rotina (de function para procedure) e inserido um parâmetro chamado “pStatus”, que será agora o retorno da procedure. Se “pStatus” retornar “Verdadeiro” então a rotina executou corretamente.

**procedure** TAlternativo.RecebeMsgPadrao(pThreadIndex: Integer;
**out** pEstiloMsgPadrao, pEstiloMsgEntrada, pEstiloMsgSaida: SMsgEstilo;
**out** pMsgPadraoLinha1, pMsgPadraoLinha2, pMsgEntradaL inha1, pMsgEntradaLinha2, pMsgSaidaLinha1, pMsgSaidaLinha2: WideString;
**out** pTempoMsgPadrao, pTempoMsgEntrada, pTempoMsgSaida: Shortint;
**out** pStatus: WordBool);

| Procedimento | Parâmetros | Tipo | Descrição |
|---|---|---|---|
| TAlternativo.RecebeMsgPadrao | pThreadIndex | Integer | Índice da thread a ser acessada |
| | pEstiloMsgPadrao | SMsgEstilo | Estilo da mensagem a ser enviada |
| | pEstiloMsgEntrada | SMsgEstilo | Estilo da mensagem a ser enviada |
| | pEstiloMsgSaida | SMsgEstilo | Estilo da mensagem a ser enviada |
| | pMsgPadraoLinha1 | String | Mensagem padrão na linha 1 |
| | pMsgPadraoLinha2 | String | Mensagem padrão na linha 2 |
| | pMsgEntradaLinha1 | String | Mensagem de Entrada na linha 1 |
| | pMsgEntradaLinha2 | String | Mensagem de Entrada na linha 2 |
| | pMsgSaidaLinha1 | String | Mensagem de Saída na linha 1 |
| | pMsgSaidaLinha2 | String | Mensagem de Saída na linha 2 |
| | pTempoMsgPadrao | Byte | Tempo que aparece a mensagem no display |
| | pTempoMsgEntrada | Byte | Tempo que aparece a mensagem no display |
| | pTempoMsgSaida | Byte | Tempo que aparece a mensagem no display |
| | pStatus | Boolean | Situação da execução da procedure. True = Executou com sucesso |

---

## Método Rec_Escala

Modificado de function para procedure.

**procedure** TAlternativo.Rec_Escala(pThreadIndex: Integer;
**out** pDataInicio: TDateTime;
**out** pHorarios: WideString;
**out** pStatus: WordBool);

| | | | |
|---|---|---|---|
| TAlternativo.Rec_Escala | pThreadIndex | Integer | Índice da thread a ser acessada |
| | pDataInicio | DateTime | Data início da escala |
| | pHorarios | WideString | Horários da escala |
| | pStatus | Boolean | Situação da execução da procedure. True = Executou c om sucesso |

## Método Rec_FncEsp_Matricula

Alterado para procedure e foi implementado um parâmetro que indica a situação da execução da procedure: pStatus.

**procedure** TAlternativo.Rec_FncEsp_Matricula(pThreadIndex: Integer;
  **out** pMatricula: WideString;
  **out** pStatus: WordBool);

| | | | |
|---|---|---|---|
| TAlternativo.Rec_FncEsp_Matricula | pThreadIndex | Integer | Índice da thread a ser acessada |
| | pMatricula | WideString | Matrícula |
| | pStatus | Boolean | Situação da execução da procedure. True = Executou com sucesso |

## Método Rec_FncEsp_Funcao

Alterado para procedure e foi implementado um parâmetro que indica a situação da execução da procedure: pStatus.

**procedure** TAlternativo.Rec_FncEsp_Funcao(pThreadIndex: Integer;
  **out** pMatricula,  pMensagem: WideString;
  **out** pTempo, pNumero: Byte;
  **out** pStatus: WordBool);

| | | | |
|---|---|---|---|
| TAlternativo.Rec_FncEsp_Funcao | pThreadIndex | Integer | Índice da thread a ser acessada |
| | pMatricula | WideString | Matricula |
| | pMensagem | WideString | Mensagem |
| | pTempo | Byte | Tempo da função |
| | pNumero | Byte | Número da função |
| | pStatus | Boolean | Situação da execução da procedure. True = Executou com sucesso |

## Método Rec_Periodo

Alterado para procedure e implementado parâmetro que indica a situação da execução da procedure: pStatus.

**procedure** TAlternativo. Rec_Periodo(pThreadIndex: Integer;
**out** pHorario: TDateTime;
**out** pTolerancia: Byte;
**out** pDomingo, pSegunda, pTerca, pQuarta, pQuinta, pSexta, pSabado, pFeriado, pStatus : WordBool);

| | | | |
|---|---|---|---|
| TAlternativo. Rec_Periodo | pThreadIndex | Integer | Índice da thread a ser acessada |
| | pHorario | TDateTime | Horário início do período |
| | pTolerancia | Byte | Tempo do período |
| | pSegunda | Boolean | Habilitado no dia |
| | pTerca | Boolean | Habilitado no dia |
| | pQuarta | Boolean | Habilitado no dia |
| | pQuinta | Boolean | Habilitado no dia |
| | pSexta | Boolean | Habilitado no dia |
| | pSabado | Boolean | Habilitado no dia |
| | pFeriado | Boolean | Habilitado no dia |
| | pStatus | Boolean | Situação da execução da procedure. True = Executou com sucesso |

## Rec_MsgEspec

Alterado para procedure e implementado par âmetro que indica a situação da execução da procedure: pStatus.

**procedure** TAlternativo.Rec_MsgEspec(pThreadIndex: Integer;
**out** pMsgStatus: SMsgEstilo;
**out** pMsgTempo: Byte;
**out** pMsgLinha1, pMsgLinha2, pMatriculas: WideString;
**out** pData: TDateTime;
**out** pTodosDias, pStatus: WordBool);

| | | | |
|---|---|---|---|
| TAlternativo.Rec_MsgEspec | pThreadIndex | Integer | Índice da thread a ser acessada |
| | pMsgStatus | SMsgEstilo | Estilo da mensagem |
| | pMsgTempo | Byte | Tempo da mensagem |
| | pMsgLinha1 | WideString | Linha 1 do display |
| | pMsgLinha2 | WideString | Linha 2 do display |
| | pMatriculas | WideString | Matriculas |
| | pData | TDateTime | Data |

| | | | |
|---|---|---|---|
| | pTodosDias | Boolean | Marcador para executar todos os dias |
| | pStatus | Boolean | Situação da execução da procedure. True = Executou com sucesso |

## Rec_ItemAcesso

A função foi alterada para procedure e foi implementado parâmetro que indica a situação da execução da procedure: pStatus.

**procedure** TAlternativo.Rec_ItemAcesso(pThreadIndex: Integer;
**out** pMatricula: WideString;
**out** pHorario: Byte;
**out** pAcesso: SAcessoOffline;
**out** pPerBloqIni, pPerBloqFim: TDateTime;
**out** pPerBloqHab, pAcionaRele1, pAcionaRele2, pAcionaRele3, pVerificarDigital, pMaster, pVisitante, pStatus: WordBool);

| | | | |
|---|---|---|---|
| TAlternativo.Rec_ItemAcesso | pThreadIndex | Integer | Índice da thread a ser acessada |
| | pMatricula | WideString | Matricula |
| | pHorario | Byte | Índice do horário |
| | pAcesso | SAcessoOffline | Tipo de acesso |
| | pPerBloqIni | TDateTime | Início do período de bloqueio |
| | pPerBloqFim | TDateTime | Fim do período de bloqueio |
| | pPerBloqHab | Boolean | Período de bloqueio habilitado |
| | pAcionaRele1 | Boolean | Aciona Relé 1 |
| | pAcionaRele2 | Boolean | Aciona Relé 2 |
| | pAcionaRele3 | Boolean | Aciona Relé3 |
| | pVerificarDigital | Boolean | Verifica digital |
| | pMaster | Boolean | Verifica se é master |
| | pVisitante | Boolean | Verifica se é visitante |
| | pStatus | Boolean | Situação da execução da procedure. True = Executou com sucesso |

### ➢ Utilizando Faixas de Acesso

Para utilizar as faixas de acesso são necessários no mínimo dois Itens de Acesso. O primeiro deve conter a matrícula inicial da faixa e na propriedade acesso a configuração "**cafFaixa**". As demais configurações deste Item de Acesso serão ignoradas pois ele está configurado como início de uma faixa. O primeiro Item de Acesso enviado após o Item configurado como "**cafFaixa**" conterá a matrícula final da faixa e todos os privilégios de acesso.

**Exemplo:**

Adiciono o primeiro item com a matrícula "00000000000000000001" e o acesso configurado como "**cafFaixa**". Adiciono o segundo item com a matrícula "00000000000000000020" com o acesso configurado como " **cafNegado**". Todas as matrículas entre 1 e 20 estarão configuradas com exatamente as mesmas configurações do segundo item enviado, nesse caso, o Item de Acesso com a matrícula 20. Portanto todas as matrícula neste intervalo estarão com o acesso negado.

O mesmo ocorre com as demais configurações (Verificação de digital, Relés, Máster, Visitante, Período de Bloqueio e Controladores).

A configuração de Faixa de Acesso é configurada por padrão no equipamento como **ativada**. Para que ela funcione em equipamentos offline basta que o Firmware do equipamento seja superior ou igual a 7100 para equipamentos Card e Orion.