**Manual de uso**

**Dll WatchComm**

**Relógio (Micro)**

Esse documento foi desenvolvido com o objetivo de auxiliar desenvolvedores a utilizar o assembly (Dll) WatchComm.dll que faz a comunicação com o relógio Micro. A Biblioteca WatchComm.dll é compatível com as plataformas. NET® Microsoft®.

Sumário

[Métodos 3](#_Toc342490189)

[SetMemoryFormat 3](#_Toc342490190)

[GetMemoryFormat 5](#_Toc342490191)

[ProgramTriggerType 5](#_Toc342490192)

[ProgramLotterySampleRate 5](#_Toc342490193)

[Activation 6](#_Toc342490194)

[EnableLogDeniedAccess 6](#_Toc342490195)

[SetDataTimeAndDST 7](#_Toc342490196)

[AlarmRing 7](#_Toc342490197)

[SetHoliday 8](#_Toc342490198)

[ClearProgramming 8](#_Toc342490199)

[CardEncryption 9](#_Toc342490200)

[ConfigureMessage 9](#_Toc342490201)

[ConfigureCard 12](#_Toc342490202)

[InquiryCardFormat 13](#_Toc342490203)

[SetCardList 13](#_Toc342490204)

[RemoveItemCardList 15](#_Toc342490205)

[SetAlternativeCode 15](#_Toc342490206)

[RemoveAlternativeCode 15](#_Toc342490207)

[Master 16](#_Toc342490208)

[UpdateShiftTable 16](#_Toc342490209)

[SetJourneyWorking 17](#_Toc342490210)

[GetStatus 20](#_Toc342490211)

[GetCurrentPunch 21](#_Toc342490212)

[RemoveCurrentPunch 21](#_Toc342490213)

[SaveBufferPoint 21](#_Toc342490214)

[StartBackupCollect 22](#_Toc342490215)

[CollectWithoutRemoving 22](#_Toc342490216)

[Coleta Backup 22](#_Toc342490217)

# Métodos

## SetMemoryFormat

Formatação da memória do relógio.

**Obs1:** Quanto menos espaço for reservado para “cadastros”, mais espaço ficará disponível para o armazenamento de registros.

**Obs2:** Esse comando somente será aceito pelo relógio, caso não existe nenhum registro pendente para coleta.

**Obs3:** Após o recebimento desse comando, o relógio perderá todos os cadastros (listas de cartões, faixas horárias, sinaleiros, etc).

Parâmetros do Método:

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Tipo** | **Descrição** |
| memoryFormat | MemoryFormat | Esse parâmetro aceita um objeto do tipo MemoryFormat.  Abaixo uma tabela para especificar as propriedades dessa classe. |

O objeto da classe MemoryFormat possui as seguintes propriedades:

|  |  |  |
| --- | --- | --- |
| **Propriedade** | **Tipo** | **Descrição** |
| MinDigitCard | Byte | Quantidade mínima de dígitos do cartão  Deve-se informar um valor compreendido entre 4 e 20 |
| MaxDigitCard | Byte | Quantidade náxima de dígitos do cartão  Deve-se informar um valor compreendido entre 4 e 20 |
| HasWork | Boolean | Utiliza jornada (true) ou não (false) |
| HasMessage | Boolean | Utiliza mensagem de usuários (true) ou não (false) |
| HasWay | Boolean | Utiliza via (true) ou não (false) |
| CounterAccess | Byte | Quantidade de contadores de acessos, baseado na quantidade de faixas horárias dentro de um turno.  Deve-se informar um valor compreendido entre 0 (não utiliza contadores de acesso) e 6 (número máximo de faixas horárias que um turno pode conter) |
| HasPassword | Boolean | Possui senha no cartão (true) ou não (false) |
| QuantityMaxCard | Int16 | Quantidade máxima de cartões  Deve-se informar um valor compreendido entre 1 e 5000 |
| QuantityMaxAlternativeID | Int16 | Quantidade máxima de códigos alternativos  Deve-se informar um valor compreendido entre 1 e 4000 |
| QuantityMaxWeeklyWork | Int16 | Quantidade máxima de jornadas semanais  Deve-se informar um valor compreendido entre 1 e 255 |
| QuantityMaxMonthlyWork | Int16 | Quantidade máxima de jornadas mensais  Deve-se informar um valor compreendido entre 1 e 255 |
| QuantityMaxPeriodicWork | Int16 | Quantidade máxima de jornadas periódicas  Deve-se informar um valor compreendido entre 1 e 255 |
| QuantityMaxAlarmRing | Byte | Quantidade máxima de sinaleiros  Deve-se informar um valor compreendido entre 1 e 99 |
| QuantityMaxShiftTable | Byte | Quantidade máxima de faixas horárias  Deve-se informar um valor compreendido entre 1 e 255 |
| QuantityMaxHoliday | Byte | Quantidade máxima de feriados  Deve-se informar um valor compreendido entre 1 e 99 |
| QuantityMaxFunction | Byte | Quantidade máxima de funções  Deve-se informar um valor compreendido entre 1 e 40 |
| MaxMessageUser | Byte | Quantidade máxima de mensagens de usuário  Deve-se informar um valor compreendido entre 1 e 7 |
| TypeCheck | TypeCheckCard | Tipo de checagem  Os valores possíveis para esse enumerador, são:  **NoCheck = 0** Não possui checagem  **Modulo11 = 1** Possui checagem do tipo modulo 11  **Modulo10 = 2** Possui checagem do tipo modulo 10 |

**Obs 1:** A soma das quantidades máximas para jornadas mensais, semanais e periódicas, não deve ultrapassar o valor de 255 jornadas.

**Obs 2:** Caso seja informado o mesmo valor para MinDigitCard e MaxDigitCard, somente serão lidos os cartões com a quantidade de dígitos especificada.

**Obs 3:** Contadores de acesso são definidos em relação a cada faixa horária de um turno (6 faixas horárias por turno). Quando definido o valor como sendo “0”, nenhuma faixa horária de um turno irá ser controlada por quantidade de acessos. Caso seja definido um valor entre “1” e “6”, somente as faixas horárias do turno correspondentes ao intervalo de “1” ao valor informado serão controladas por contador de acesso. Ex.: caso seja informado “2” somente a primeira e segunda faixas horárias poderão ser controladas por acesso.

## GetMemoryFormat

Obtém o formato de memória atualmente configurado no relógio. Retorna um objeto do tipo MemoryFormat, cujas propriedades já foram relacionadas acima, na descrição do método anterior.

## ProgramTriggerType

Configuração do tipo de acionamento.

Parâmetros do Método:

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Tipo** | **Descrição** |
| type | Byte | Tipo de acionamento.  Os valores possíveis para esse parâmetro, são:  **0** = Nenhum  **2** = Fechadura simples  **8** = Catraca de entrada  **9** = Catraca de saída  **10** = catraca bi-direcional dependente do leitor  **11** = Catraca de saída livre  **13** = Catraca bi-direcional independente do leitor  **14** = Catraca bi-direcional dependente do sentido da passagem  **17** = Fechadura dupla |
| time | Int32 | Tempo de acionamento.  No máximo 9,9 segundos.  A informação deve ser enviada em milisegundos.  Exemplo: para enviar 1,5 segundos. Deve-se enviar 150.  **Obs:** enviar essa informação apenas para acionamento do tipo “fechadura”. Do contrário, enviar 0 (zero). |

## ProgramLotterySampleRate

Configuração da taxa de amostragem.

Parâmetros do Método:

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Tipo** | **Descrição** |
| rate | Byte | Corresponde ao percentual para amostragem que o relógio deve selecionar.  Assim, enviar um número inteiro entre 0 e 100. |
| inout | Byte | Enviar fixo “0”.  Tal parâmetro é utilizado apenas para outro modelo de equipamento. |

## Activation

Estado de funcionamento do equipamento e configuração para controle de acesso ao acionamento configurado.

Parâmetros do Método:

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Tipo** | **Descrição** |
| Active | Boolean | Torna o equipamento operante ou inoperante. Geralmente utilizado para eventuais manutenções ou a critério do usuário.  Caso seja fornecido o valor “false”, o equipamento fica inoperante.  Caso seja fornecido o valor “true”, o equipamento fica operante. |
| Controlled | Boolean | Caso utilize algum tipo de acionamento (vide comando ProgramTriggerType), esse parâmetro passa a ser considerado. Do contrário, enviar sempre “false”.  Enviando “true” o acesso ao acionamento passa a ser controlado pelo equipamento.  Enviando “false” o acesso ao acionamento fica sempre livre. Exemplo: no caso de uma catraca, o braço ficará sempre livre. |

## EnableLogDeniedAccess

Configuração de armazenamento de registros bloqueados.

Parâmetros do Método:

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Tipo** | **Descrição** |
| enable | Boolean | Entende-se como registro bloqueado, um registro cujo cartão não esteja na lista de acesso do relógio, fora de turno ou contador de acesso excedido.  Se deseja que o registro seja armazenado, envie “true”, do contrário, envie “false”. |

## SetDataTimeAndDST

O método tem como objetivo enviar data, hora e horário de verão para o relógio.

Parâmetros do método

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Tipo** | **Descrição** |
| Date | DateTime | Data e hora a ser atualizada no relógio |
| dstBegin | DateTime | Início do horário de verão |
| dstEnd | DateTime | Fim do Horário de verão. |

**Obs:** Caso não deseje configurar o horário de verão, basta informar “null” para os parâmetros dstBegin e dstEnd. Desse modo, o horário de verão ficará desabilitado.

## AlarmRing

Envia uma coleção de sinaleiros para o relógio. O método espera um objeto do tipo AlarmRingsCollection. Tal objeto deve ser populado com objetos do tipo AlarmRings através da chamada do método Add do objeto da classe AlarmRingsCollection.

O objeto da classe AlarmRings possui as seguintes propriedades:

|  |  |  |
| --- | --- | --- |
| **Propriedade** | **Tipo** | **Descrição** |
| Id | Integer | Número de identificação do sinaleiro, compreendido entre 1 e 99. |
| TimeAlarm | String | Horário para execução do sinaleiro, no formato “hh:mm”. |
| Duration | Byte | Tempo de duração do sinaleiro em segundos. Informe um valor compreendido entre 1 e 15. |
| TypeRing | Enumerador | Parâmetro para indicar qual é o tipo de sinaleiro. Os valores possíveis para esse enumerador, são:  **Inside = 0** Interno  **Outside = 1** Externo |
| RingSunday | Enumerador | Valores possíveis:  **NoRing = 0** Não aciona  **Ring = 1** Aciona no domingo |
| RingMonday | Enumerador | **NoRing = 0** Não aciona  **Ring = 1** Aciona na segunda |
| RingTuesday | Enumerador | **NoRing = 0** Não aciona  **Ring = 1** Aciona na terça |
| RingWednesday | Enumerador | **NoRing = 0** Não aciona  **Ring = 1** Aciona na quarta |
| RingThursday | Enumerador | **NoRing = 0** Não aciona  **Ring = 1** Aciona na quinta |
| RingFriday | Enumerador | **NoRing = 0** Não aciona  **Ring = 1** Aciona na sexta |
| RingSaturday | Enumerador | **NoRing = 0** Não aciona  **Ring = 1** Aciona no sábado |
| RingHoliday | Enumerador | **NoRing = 0** Não aciona  **Ring = 1** Aciona no feriado |

**Obs:** O código informado (propriedade ID mencionada acima) não pode ser superior a quantidade máxima de sinaleiros previsto no formato de memória, vide comando “SetMemoryFormat”.

## SetHoliday

Envia uma coleção de feriados para o relógio. O método espera um objeto do tipo HolidayCollection. Tal objeto deve ser populado com objetos do tipo Holiday através da chamada do método Add do objeto da classe HolidayCollection.

O objeto da classe Holiday possui as seguintes propriedades:

|  |  |  |
| --- | --- | --- |
| **Propriedade** | **Tipo** | **Descrição** |
| Day | Byte | Dia correspondente ao feriado que deseja-se configurar. |
| Month | Byte | Mês correspondente ao feriado que deseja-se configurar. |

**Obs:** A quantidade de feriados populados no objeto da classe HolidayCollection não pode ser superior a quantidade máxima de feriados previsto no formato de memória, vide comando “SetMemoryFormat”.

## ClearProgramming

Limpeza de toda a tabela do relógio. É possível solicitar a limpeza de uma tabela especifica ou de todas em um só comando.

Parâmetros do Método:

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Tipo** | **Descrição** |
| clearProgramming | TypeClearProgramming | Refere-se a tabela que deseja-se limpar (apagar todos os registros).  Os valores possíveis para esse enumerador, são:  **All = 0** Todas as tabelas  **CardAndAlternativeCode = 1** Cartões e códigos alternativos  **AlternativeCode = 2** Códigos alternativos  **Holiday = 3** Feriados  **Turn = 4** Faixas horárias (Turno)  **Jornadas = 5** Jornadas  **MessageFunctions = 6** Mensagens de usuário, mensagens de sistema e mensagens de função  **RingBell = 7** Sinaleiros  **Master = 8** Supervisor |

## CardEncryption

Configuração da criptografia do cartão (caso seja necessário usar).

Parâmetros do Método:

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Tipo** | **Descrição** |
| encryptionType | EGenericEncryptionType | Refere-se a tabela que deseja-se limpar (apagar todos os registros).  Os valores possíveis para esse enumerador, são:  **Disable = 0** Desabilita a utilização da criptografia  **EightDigits = 1** Utiliza a criptografia de 8 dígitos  **TwelveDigits = 2** Utiliza a criptografia de 12 digitos |
| vector1 | Int64 | Vetor 1 |
| vector2 | Int64 | Vetor 2 |
| verify1 | Int32 | Verificador 1 |
| verify2 | Int32 | Verificador 2 |

**Obs:** Este documento não entrará em detalhes sobre o funcionamento desta criptografia, parâmetro restrito a utilização em conjunto com consultas de viabilidade efetuadas juntamente a DIMEP.

## ConfigureMessage

Configuração de mensagens de usuário, sistema e função.

Parâmetros do Método:

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Tipo** | **Descrição** |
| messageType | TypeMessage Configuration | Tipo de mensagem que deseja-se configurar.  Os valores possíveis para esse enumerador, são:  **User = 0** Para configurar uma mensagem de usuário  **System = 1** Para configurar uma mensagem de sistema  **Funcion = 2** Para configurar uma mensagem de função |
| messageCode | Byte | Refere-se ao código de mensagem de usuário, sistema ou função, conforme especificado no parâmetro anterior. |
| description | String | Nova descrição para a mensagem. |
| typeFunction | TypeActionFunction | Esse parâmetro deve ser utilizado sempre que estiver configurando uma mensagem de função (determinado pelo parametro “messageType”).  Esse parâmetro aceita um objeto do tipo TypeActionFunction.  Abaixo uma tabela para especificar as propriedades dessa classe. |

O objeto da classe TypeActionFunction possui as seguintes propriedades:

|  |  |  |
| --- | --- | --- |
| **Propriedade** | **Tipo** | **Descrição** |
| activeFunction | Boolean | Se a função está ativa (true) ou não (false) |
| checkList | Boolean | Se o relógio deve verificar o cartão na lista (true) ou não (false) |
| triggerOut | Boolean | Se deve acionar um dispositivo externo (true) ou não (false) |
| checkJourney | Boolean | Se deve verificar a jornada (true) ou não (false) |
| requestsMaster | Boolean | Se solicita a permissão do supervisor (true) ou não (false) |
| requesTypingKeyboard | Boolean | Se permite a digitação do cartão via teclado (true) ou não (false) |
| storesRecordBlocked | Boolean | Se armazena os registros bloqueados (true) ou não (false) |
| storesRecordFreed | Boolean | Se armazena os registros permitidos (true) ou não (false) |

**Obs:** Para preenchimento da propriedade messageCode, observe as tabelas abaixo.

Abaixo a tabela de mensagens de usuário:

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Mensagem Padrão (alterar através da propriedade description)** |
| 1 | Mensagem Usuário 01 | Mensagem Usuário 01 |
| 2 | Mensagem Usuário 02 | Mensagem Usuário 02 |
| 3 | Mensagem Usuário 03 | Mensagem Usuário 03 |
| 4 | Mensagem Usuário 04 | Mensagem Usuário 04 |
| 5 | Mensagem Usuário 05 | Mensagem Usuário 05 |
| 6 | Mensagem Usuário 06 | Mensagem Usuário 06 |
| 7 | Mensagem Usuário 07 | Mensagem Usuário 07 |

Abaixo a tabela de mensagens de sistema:

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Mensagem Padrão (alterar através da propriedade description)** |
| 0 | Header do Relógio | DIMEP PONTO & ACESSO |
| 3 | Bloqueado por Lista | Bloqueado por lista |
| 4 | Bloqueado por Horário | Bloqueado p/ horário |
| 7 | Excedeu Contador de Acesso | Excedeu cont acesso |
| 8 | Acesso Leitor 0 | Acesso Leitor 0 |
| 9 | Acesso Leitor 1 | Acesso Leitor 1 |
| 11 | Sorteado Para Revista | Sorteado |
| 12 | Senha Inválida | Senha Invalida |
| 14 | Acesso Permitido | Acesso Permitido |

Abaixo a tabela de mensagens de função:

|  |  |  |
| --- | --- | --- |
| **Código** | **Descrição** | **Mensagem Padrão (alterar através da propriedade description)** |
| 1 | Função 01 | Função 01 |
| 2 | Função 02 | Função 02 |
| 3 | Função 03 | Função 03 |
| 4 | Função 04 | Função 04 |
| 5 | Função 05 | Função 05 |
| 6 | Função 06 | Função 06 |
| 7 | Função 07 | Função 07 |
| 8 | Função 08 | Função 08 |
| 9 | Função 09 | Função 09 |
| 10 | Função 10 | Função 10 |
| 11 | Função 11 | Função 11 |
| 12 | Função 12 | Função 12 |
| 13 | Função 13 | Função 13 |
| 14 | Função 14 | Função 14 |
| 15 | Função 15 | Função 15 |
| 16 | Função 16 | Função 16 |
| 17 | Função 17 | Função 17 |
| 18 | Função 18 | Função 18 |
| 19 | Função 19 | Função 19 |
| 20 | Função 20 | Função 20 |
| 21 | Função 21 | Função 21 |
| 22 | Função 22 | Função 22 |
| 23 | Função 23 | Função 23 |
| 24 | Função 24 | Função 24 |
| 25 | Função 25 | Função 25 |
| 26 | Função 26 | Função 26 |
| 27 | Função 27 | Função 27 |
| 28 | Função 28 | Função 28 |
| 29 | Função 29 | Função 29 |
| 30 | Função 30 | Função 30 |
| 31 | Função 31 | Função 31 |
| 32 | Função 32 | Função 32 |
| 33 | Função 33 | Função 33 |
| 34 | Função 34 | Função 34 |
| 35 | Função 35 | Função 35 |
| 36 | Função 36 | Função 36 |
| 37 | Função 37 | Função 37 |
| 38 | Função 38 | Função 38 |
| 39 | Função 39 | Função 39 |
| 40 | Função 40 | Função 40 |

## ConfigureCard

Configuração do formato do cartão, para casos onde a configuração básica existente no formato da memória não atenda a necessidade do cliente.

Parâmetros do Método:

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Tipo** | **Descrição** |
| format | String | O formato do cartão serve para informar como o código de cartão será composto.  Cada dígito do cartão pode assumir os seguintes valores:  I – Dígito de Informação  O – Dígito opcional Opcional  X – Digito deve ser ingnorado  V – Dígito correspondente a via do cartão  C – Dígito correspondente a checagem do cartão  **Obs:** O formato do cartão não deve ultrapassar 16 caracteres. |

**Obs1:** A configuração do cartão enviada por esse comando irá sobrepor a configuração existente no formato de memória, implicando assim, na capacidade que o relógio terá para armazenar registros.

**Obs2:** Caso o comando SetMemoryFormat seja enviado após esse comando, prevalecerá a configuração enviado no formato de memória. Via de regra, vale a configuração enviada pelo último comando.

## InquiryCardFormat

Obtém o formato de cartão atualmente configurado no relógio.

O método retorna uma string que representa o formato do cartão. O entendimento dessa string dá-se da mesma forma para a formação da string enviada para configuração do formato, conforme mencionado no método acima (ConfigureCard).

## SetCardList

Inclusão de cartões da lista do relógio.

Parâmetros do Método:

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Tipo** | **Descrição** |
| cardCollection | CardCollection | O parâmetro espera um objeto do tipo CardCollection. Tal objeto deve ser populado com objetos do tipo Card através da chamada do método Add do objeto da classe CardCollection. |
| usesPrinter | Boolean | Relógio configurado para utilizar impressora (true) ou não (false).  Caso o relógio esteja configurado para utilizar impressora, o documento (propriedade da classe Card) também será enviado para o relógio.  Atualmente (11/10/2012), apenas o Chile utiliza esse modelo de relógio com impressora. |

O objeto da classe Card possui as seguintes propriedades:

|  |  |  |
| --- | --- | --- |
| **Propriedade** | **Tipo** | **Descrição** |
| Code | String | Código do cartão |
| Message | Byte | Código da mensagem de usuário (mensagem que será exibida quando o acesso for concedido) |
| Way | Byte | Número da via do cartão |
| Jornada | Int16 | Código correspondente a jornada que esse cartão obedecerá |
| CounterAccess | TypeCounter\_Access | Esse parâmetro aceita um objeto do tipo TypeCounter\_Access.  Abaixo uma tabela para especificar as propriedades dessa classe. |
| Password | Int32 | Senha para conceder acesso ao cartão |
| Document | String | Documento do funcionário que possui o cartão. Informar um valor com no máximo 10 digitos.  **Obs:** Atualmente (11/10/2012), essa informação é utilizada apenas no Chile |

O objeto da classe TypeCounter\_Access possui as seguintes propriedades:

|  |  |  |
| --- | --- | --- |
| **Propriedade** | **Tipo** | **Descrição** |
| Counter\_Access1 | Byte | Quantidade de acessos para a primeira faixa horária do turno. |
| Counter\_Access2 | Byte | Quantidade de acessos para a segunda faixa horária do turno. |
| Counter\_Access3 | Byte | Quantidade de acessos para a terceira faixa horária do turno. |
| Counter\_Access4 | Byte | Quantidade de acessos para a quarta faixa horária do turno. |
| Counter\_Access5 | Byte | Quantidade de acessos para a quinta faixa horária do turno. |
| Counter\_Access6 | Byte | Quantidade de acessos para a sexta faixa horária do turno. |

**Obs1:** Para todas as propriedades da classe acima, sempre informar um valor compreendido entre 0 e 9, entendendo que:

0 🡺 Significa não permitir o acesso do cartão dentro da faixa horária.

9 🡺 Significa permitir o acesso ilimitado do cartão dentro da faixa horária.

Diferente de 0 e 9 🡺 O número informado corresponderá a quantidade de acessos permitidos para o intervalo da faixa horária.

**Obs2:** Essa configuração possui relação com a configuração do formato da memória, no que diz respeito a propriedade CounterAccess do objeto MemoryFormat, enviado no método SetMemoryFormat. Exemplo, caso seja configurado para contar o acesso até a quarta faixa horária, de nada adianta configurar as propriedades Counter\_Access5 e Counter\_Access6, pois elas não serão levadas em consideração.

## RemoveItemCardList

Remove um cartão da lista de cartões do relógio.

Parâmetros do Método:

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Tipo** | **Descrição** |
| code | Int32 | Corresponde ao código do cartão que deseja-se remover da lista de cartões permitidos do relógio. |

## SetAlternativeCode

Insere um código alternativo na lista do relógio.

Parâmetros do Método:

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Tipo** | **Descrição** |
| alternativeCode | Int32 | Código alternativo para ser utilizado no lugar do cartão. |
| Code | String | Código do cartão que poderá ser acessado pelo código alternativo informado no parâmetro acima. |

## RemoveAlternativeCode

Remove um código alternativo da lista do relógio.

Parâmetros do Método:

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Tipo** | **Descrição** |
| alternativeCode | Int32 | Código alternativo que deseja-se remover da lista do relógio. |

## Master

Inclui um supervisor na lista do relógio.

Parâmetros do Método:

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Tipo** | **Descrição** |
| id | Int32 | Código de identificação do supervisor |
| password | Int32 | Senha para autenticação do supervisor |
| pendriveProgramming | Boolean | O supervisor possui permissão pra programação via pen drive (true) ou não (false). |
| dateTimeUpdated | Boolean | O supervisor possui permissão para atualização de data e hora (true) ou não (false) |
| programmingTechnical | Boolean | O supervisor possui permissão para efetuar programações tecnicas (true) ou não (false) |

## UpdateShiftTable

Envia uma coleção de faixas horárias para o relógio. O método espera um objeto do tipo ShiftTableCollection. Tal objeto deve ser populado com objetos do tipo ShiftTable através da chamada do método Add do objeto da classe ShiftTableCollection.

O objeto da classe ShiftTableCollection possui as seguintes propriedades:

|  |  |  |
| --- | --- | --- |
| **Propriedade** | **Tipo** | **Descrição** |
| Id | Int16 | Código de identificação da faixa horária.  **Obs:** O código informado não pode ser superior a quantidade máxima de faixas horárias previsto no formato de memória, vide comando “SetMemoryFormat”. |

O objeto da classe ShiftTable possui as seguintes propriedades:

|  |  |  |
| --- | --- | --- |
| **Propriedade** | **Tipo** | **Descrição** |
| Inicio | String | Horário para ínicio do período.  Preencher a propriedade com o horário formatado no padrão “HHmm”. |
| Fim | String | Horário para fim do período.  Preencher a propriedade com o horário formatado no padrão “HHmm”. |

**Obs1:** Popular o objeto da classe ShiftTableCollection com exatamente seis objetos da classe ShiftTable. Caso o usuário não tenha configurado todas as faixas. Preencha as propriedades inicio e fim com o valor “0000”.

**Obs2:** Não popular o objeto da classe ShiftTableCollection com um objeto da classe ShiftTable, cujas propriedades Inicio e Fim tenham sido carregadas com o conteúdo “0000” (sem faixa) sendo que após o objeto ShiftTableCollection será populado com um objeto ShiftTable que possui conteúdo diferente de “0000” para suas propriedades.

Exemplo do que NÃO deve ser feito:

Faixa 1: 08:00 até 13:00  
 Faixa 2: 00:00 até 00:00  
 Faixa 3: 14:00 até 18:00  
 Faixa 4: 00:00 até 00:00  
 Faixa 5: 00:00 até 00:00  
 Faixa 6: 00:00 até 00:00

**Obs3:** Não popular o objeto com faixas horárias (objetos da classe ShiftTable) com horários sobrepostos. Exemplo de faixas horárias sobrepostas:

Faixa 1: 09:00 até 15:00  
 Faixa 2: 14:00 até 18:00  
Faixa 3: 00:00 até 00:00  
Faixa 4: 00:00 até 00:00  
Faixa 5: 00:00 até 00:00  
Faixa 6: 00:00 até 00:00

Observe que no exemplo acima qualquer horário entre as 14:00hr e 15:00hr está previsto nas duas faixas horárias.

## SetJourneyWorking

Envia uma jornada para o relógio. Uma jornada pode ser do tipo semanal, mensal ou periódica. Assim, o método espera um objeto do tipo WeeklyJourneyWorking (jornada semanal), MonthlyJourneyWorking (jornada mensal) ou PeriodicJourneyWorking (jornada periódica).

O objeto da classe WeeklyJourneyWorking possui as seguintes propriedades:

|  |  |  |
| --- | --- | --- |
| **Propriedade** | **Tipo** | **Descrição** |
| Id | Int16 | Código de identificação da jornada. Deve-se informar um valor compreendido entre 1 e 255.  **Obs:** O código informado não pode ser superior a quantidade máxima de jornadas semanais previsto no formato de memória, vide comando “SetMemoryFormat”. |
| TypeWorking | TypeWorking | Tipo da jornada.  Os valores possíveis para esse enumerador, são:  **Weekly = 0** Jornada semanal  **Monthly = 1** Jornada mensal **Periodic = 2** Jornada periódica  **Obs:** informar o valor Weekly. |
| Sunday | Int16 | Código da faixa horária para o domingo.  Informar um valor compreendido entre 0 (não possui faixa horária para esse dia) e 255. |
| Monday | Int16 | Código da faixa horária para a segunda.  Informar um valor compreendido entre 0 (não possui faixa horária para esse dia) e 255. |
| Tuesday | Int16 | Código da faixa horária para a terça.  Informar um valor compreendido entre 0 (não possui faixa horária para esse dia) e 255. |
| Wednesday | Int16 | Código da faixa horária para a quarta.  Informar um valor compreendido entre 0 (não possui faixa horária para esse dia) e 255. |
| Thursday | Int16 | Código da faixa horária para a quinta.  Informar um valor compreendido entre 0 (não possui faixa horária para esse dia) e 255. |
| Friday | Int16 | Código da faixa horária para a sexta.  Informar um valor compreendido entre 0 (não possui faixa horária para esse dia) e 255. |
| Saturday | Int16 | Código da faixa horária para o sábado.  Informar um valor compreendido entre 0 (não possui faixa horária para esse dia) e 255. |
| Holiday | Int16 | Código da faixa horária para os feriados.  Informar um valor compreendido entre 0 (não possui faixa horária para feriado) e 255. |

O objeto da classe MonthlyJourneyWorking possui as seguintes propriedades:

|  |  |  |
| --- | --- | --- |
| **Propriedade** | **Tipo** | **Descrição** |
| Id | Int16 | Código de identificação da jornada. Deve-se informar um valor compreendido entre 1 e 255.  **Obs:** O código informado não pode ser superior a quantidade máxima de jornadas mensais previsto no formato de memória, vide comando “SetMemoryFormat”. |
| TypeWorking | TypeWorking | Tipo da jornada.  Os valores possíveis para esse enumerador, são:  **Weekly = 0** Jornada semanal  **Monthly = 1** Jornada mensal **Periodic = 2** Jornada periódica  **Obs:** informar o valor Monthly. |
| Turno | List<Int16> | Lista de códigos de faixas horárias.  Deve-se chamar o método “add” dessa propriedade para informar qual faixa horária deve ser utilizada para cada dia do mês.  Caso, não deseja-se utilizar faixa horária para algum dia do mês, informe zero, mas, obrigatóriamente deve-se preencher essa lista com 31 códigos.  A cada chamada do método “add” dessa propriedade, informar um valor compreendido entre 0 e 255. |

O objeto da classe PeriodicJourneyWorking possui as seguintes propriedades:

|  |  |  |
| --- | --- | --- |
| **Propriedade** | **Tipo** | **Descrição** |
| Id | Int16 | Código de identificação da jornada. Deve-se informar um valor compreendido entre 1 e 255.  **Obs:** O código informado não pode ser superior a quantidade máxima de jornadas peiódicas previsto no formato de memória, vide comando “SetMemoryFormat”. |
| TypeWorking | TypeWorking | Tipo da jornada.  Os valores possíveis para esse enumerador, são:  **Weekly = 0** Jornada semanal  **Monthly = 1** Jornada mensal **Periodic = 2** Jornada periódica  **Obs:** informar o valor Periodic. |
| DataInicio | DateTime | Data a partir da qual o relógio passará a considerar a configuração de jornada periódica. |
| Turno | List<Int16> | Lista de códigos de faixas horárias.  Deve-se chamar o método “add” dessa propriedade para informar qual faixa horária deve ser utilizada para cada dia do período.  Lembrando que ao fim de cada período, o relógio volta a considerar o primeiro dia, de maneira rotativa e continua, desde que a configuração de jornada não seja alterada.  Caso, não deseja-se utilizar faixa horária para algum dia do período, informe zero.  A cada chamada do método “add” dessa propriedade, informar um valor compreendido entre 0 e 255.  Preencher a lista com no mínimo 2 dias e no máximo 60 dias. |

## GetStatus

Solicita o status do relógio. O método não possui parâmetros e retorna um objeto do tipo AbstractStatusMessage.

O objeto da classe AbstractStatusMessage possui as seguintes propriedades:

|  |  |  |
| --- | --- | --- |
| **Propriedade** | **Tipo** | **Descrição** |
| CardListSize | UInt32 | Capacidade de armazenamento da lista de cartões |
| CardsInList | UInt32 | Quantidade de cartões contidos na lista |
| CheckType | Byte | Tipo de checagem  Os valores possíveis para essa propriedade, são:  **0 🡺** Não possui checagem  **1** 🡺 Possui checagem do tipo modulo 11  **2 🡺** Possui checagem do tipo modulo 10 |
| Date | DateTime | Data e hora atual |
| FirmwareVersion | String | Versão do firmware (software interno) sendo executado |
| RecordDeniedAccess | Boolean | Se configurado pra armazenar registros bloqueados (true) ou não (false) |
| RecordsCount | UInt32 | Quantidade de registros ainda não coletados |
| RecordsSize | UInt32 | Capacidade de armazenamento de registros |
| MinDigitCard | Byte | Quantidade mínima de dígitos do cartão |
| MaxDigitCard | Byte | Quantidade máxima de dígitos do cartão |
| LotterySampleRate | Byte | Taxa de amostragem |

## GetCurrentPunch

Solicita a coleta de um registro do relógio. O método não possui parâmetro e retorna um objeto do tipo AbstractMessage, caso exista registro a ser recolhido. Caso não exista registro a ser recolhido, o método retornará um ponteiro vazio (null / nothing). Após receber o registro, deve-se chamar o método RemoveCurrentPunch para confirmar ao relógio que o registro foi recebido com sucesso. O método RemoveCurrentPunch, será especificado nesse manual logo após este método.

Para esse modelo de relógio, deve-se converter (cast) o objeto da classe AbstractMessage para um objeto da classe AbstractPunchMessage.

O objeto da classe AbstractPunchMessage possui as seguintes propriedades:

|  |  |  |
| --- | --- | --- |
| **Propriedade** | **Tipo** | **Descrição** |
| Id | UInt64 | Código do cartão |
| Date | DateTime | Data e Hora a qual o registro foi gravado no relógio |
| WatchEvent | UInt32 | Código do evento que originou a gravação do registro |
| IsValid | Boolean | Indica se o registro coletado é válido ou não. Caso não seja válido, é interessante deixar isso registrado num log, por exemplo, e prosseguir com o processo de coleta. Do contrário, o registro inválido travará a coleta até que ele seja removido do equipamento. |
| DataArea | Byte[] | Array de bytes correspondentes a área de dados da mensagem TCP/IP trocada entre o equipamento e o software. Essa informação é importante para que o desenvolvedor do equipamento possa identificar com mais facilidade a causa de um registro inválido, por exemplo. |

## RemoveCurrentPunch

Confirma para o relógio o recebimento de um registro de coleta. Ao efetuar essa confirmação para o relógio, automaticamente o relógio enviará o próximo registro de coleta. Esse processo de confirmação deve ser repetido até que não existam mais registros a serem recolhidos. Esse ponto será identificado quando o método retornar um ponteiro vazio (null / nothing). Assim como o método GetCurrentPunch (acima mencionado), esse método retorna um objeto do tipo AbstractMessage, que deve ser convertido (cast) para um objeto do tipo AbstractPunchMessage, cujas propriedades já foram mencionadas na descrição do método GetCurrentPunch.

## SaveBufferPoint

Comando utilizado no processo de coleta backup (ler Sessão “Coleta Backup”). Numa visão macro, esse comando é responsável por salvar o ponteiro de coleta do relógio. No entanto, não é necessário entrar em detalhe quanto sua funcionalidade. Esse comando só possui utilidade se utilizado em conjunto a outros comandos, conforme descrito na Sessão “Coleta Backup”. Este comando não possui retorno.

## StartBackupCollect

Comando utilizado no processo de coleta backup (ler Sessão “Coleta Backup”). Numa visão macro, esse comando solicita ao relógio o inicio da coleta backup, ou seja, o relógio irá reposicionar seu ponteiro de registros para poder resgatá-los para coleta. Este comando não possui retorno.

Parâmetros do Método:

|  |  |  |
| --- | --- | --- |
| **Parâmetro** | **Tipo** | **Descrição** |
| startDate | DateTime | Data a partir da qual os registros devem ser resgatados para coleta.  Parâmetro opcional.  Caso não informado, todos os registros armazenados no relógio, serão recolhidos. |

## CollectWithoutRemoving

Comando utilizado no processo de coleta backup (ler Sessão “Coleta Backup”). Na prática, esse comando funciona de maneira semelhante ao comando GetCurrentPunch (já descrito acima). Ou seja, responsável por coletar um registro. Assim como o comando GetCurrentPunch, este também deve ser utilizado junto ao comando RemoveCurrentPunch (também já descrito acima). Este comando retorna um objeto do tipo AbstractMessage, ao qual precisa ser convertido (cast) num objeto do tipo AbstractPunchMessage.

# Coleta Backup

O relógio armazena na sua memória uma quantidade “x” de registros mesmo após os mesmos já terem sido coletados no processo convencional, já descrito nesse manual.

Desse modo, coleta backup trata-se de um mecanismo para coletar novamente esses registros armazenados.

Para realizar a coleta backup dos registros, é preciso executar uma série de comandos já descritos acima, de acordo com a seguinte ordem: SaveBufferPoint (salva o ponteiro de coleta), StartBackupCollect (Inicia a coleta backup), CollectWithoutRemoving (coleta do registro de backup) e RemoveCurrentPunch(confirma o recebimento do registro), devendo este ultimo, ser executado enquanto ainda houverem registros.