

## 1 Introdução

---

A Hiro.dll é uma biblioteca que provê todos os recursos necessários para a comunicação do integrador de bombas de combustível Hiro com softwares de automação comercial. Para o correto funcionamento e utilização da Hiro.dll é necessário que o computador possua o Microsoft.NET framework 4.0 ou superior e a Operacoes.dll.

Este manual descreve de forma detalhada todos os recursos implementados na DLL:

- Detalha todos os elementos que compõem a DLL (classes, enumerações, estruturas);
- Para cada classe detalha seus atributos e métodos;
- Detalha a arquitetura da comunicação.

Visando atender as mais diversos sistemas existentes no mercado, nossa DLL apresenta diversos métodos que visam atender ao mesmo objetivo, tendo como diferença os parâmetros de entrada e retorno.

### 1.1 Versão do Manual

---

- Versão = 4.0

#### 1.1.1 Histórico de Alterações

---

Data	Responsável	Descrição
13/03/2023	Ronaldo Pereira	Inserção de novo método: "LiberaAbastecelD".
01/03/2023	Ronaldo Pereira	Adição da propriedade "PortaTCPFixa".
10/12/2019	Ronaldo Pereira	Inserção de novos métodos: GetPrecoDoBicoNivel e SetPrecoBicoNivel.
06/11/2017	Ronaldo Pereira	Ajustes neste manual.
11/10/2017	Ronaldo Pereira	Inserção de novos métodos: LeStatusTodosBicos, GetLeituraDeAbastecimentoPAFECE, GetLeituraDeAbastecimentoPT, GetStatusMedidor, LiberacaoAbastece.  Inserção de novas enumerações: EAcaoBico. Atualização das interfaces.  Inserção de novas estruturas: SMedidor.
17/02/2015	Daniel Martins	Adição dos métodos referentes ao envio e gravação de firmware no concentrador; Complementação da descrição das propriedades da interface IConcentrador.

## 2 Enumerações

---

As enumerações descrevem um conjunto de valores possíveis para uma variável.

### 2.1 *EParametrosComunicacao (Obsoleto)*

**Utilize o HIRO CONFIG para definir os parâmetros de conexão com o equipamento.**

Descreve os valores de configuração utilizados pela Hiro.dll para efetuar a comunicação com o concentrador. O valor selecionado indica o parâmetro que está sendo utilizado.

Valores:

- Concentrador: Indica o ID do concentrador;
- Tipo: Indica o tipo de comunicação (1-Serial, 2-TCP/IP, 3-USB);
- PortaCOM: Porta com a qual o concentrador está comunicando;
- PID: Identificador do produto (Product ID);
- VID: Identificador do fabricante (Vendor ID);
- EnderecoIP: Endereço do concentrador;
- PortaTCP: Porta utilizada para a comunicação com o concentrador;
- MascaraSubRede: Mascara da rede na qual o concentrador está conectado;
- DNSPadrao: DNS padrão utilizado pela rede na qual o concentrador está conectado;
- Gateway: Gateway padrão utilizado pela rede na qual o concentrador está conectado;
- DHCP: Indica se deve ou não utilizar DHCP.
- ProtocoloSerial: 'Hiro' ou 'CT'.

### 2.2 *EStatusAbastecimento*

Indica o estado em que um abastecimento se encontra na memória do concentrador.

Valores:

- ValidoNaoApagado (?) : Indica que o abastecimento ainda não foi lido e apagado da memória do concentrador;
- ValidoLidoNaoApagado (=): Indica que o abastecimento foi lido mas não foi apagado do concentrador;
- ValidoLidoApagado (<): Indica que o abastecimento foi lido e apagado da memória do concentrador.

### 2.3 *EStatusBico*

Indica o estado em que o bico se encontra.

Valores:

- Livre (L): Bico livre para abastecimento;
- Ocupado (O): Bico com abastecimento em andamento;
- SemComunicação (D): Bico não comunica com o concentrador;
- LivreBloqueada (E): Bico livre aguardando liberação;
- OcupadaBloqueada (P): Bico ocupado aguardando liberação;
- SolicitandoLiberacaoSemCartao (S): Aguardando liberação sem cartão;
- SolicitandoLiberacaoComCartao (C): Aguardando liberação com cartão;
- Inexistente (p): Bico não configurado no concentrador;
- SemConfiguração: (f): Bico não configurado no concentrador.

## 2.4 EStatusComando

Indica o resultado de um comando enviado ao concentrador.

Valores:

- ComandoProcessadoComSucesso (s): Comando processado sem falhas;
- ErroNoProcessamento (r): Falha ao processar o comando;
- ErroDeParametro (p): Parâmetro inválido.

## 2.5 EStatusConfirmacaoLeitura

Indica o resultado da confirmação da leitura de abastecimento.

Valores:

- Incrementado (s): Indica que o ponteiro de abastecimentos foi incrementado;
- NaoIncrementado (r): Indica que o ponteiro de abastecimentos não foi incrementado;
- IDInexistente (p): Indica que o ID informado não existe;
- Pendente (P): Indica que o abastecimento está pendente de confirmação.

## 2.6 EStatusDaComunicacao

Indica o estado da comunicação com o concentrador (estado da troca de mensagens entre o sistema e o concentrador).

Valores:

- SemComunicacao: Nenhuma comunicação no momento;
- Iniciada: Comunicação entre sistema e concentrador iniciada;
- VerificandoDirecao: Verificação se a direção da comunicação é a esperada;
- VerificandoConcentrador: Verificando se o concentrador que respondeu é o esperado;
- VerificandoComando: Verifica se o comando retornado é o comando esperado;
- RecebendoDados: Recebendo o retorno da comunicação com o concentrador;
- AguardandoFim: Aguardando parâmetro de fim de mensagem;
- Encerrada: Comunicação encerrada.
- Cancelada: Comunicação abortada.

## 2.7 EStatusDaComunicacaoConcentrador

Indica o estado da comunicação com o concentrador (indica se o sistema consegue comunicar com o concentrador).

Valores:

- FalhaNaPorta: Impossível comunicar com a porta COM informada;
- ConcentradorNaoResponde: Sistema não recebe nenhum retorno do concentrador (timeout);
- OK : Comunicação estabelecida;
- FalhaNaComunicacao: Erro ao tentar comunicar.

## 2.8 ETipoCartao

Indica o tipo de cartão utilizado.

Valores:

- Funcionario (f): Cartão de funcionário;
- Cliente (c): Cartão de cliente;
- Motorista (m): Cartão de motorista;
- Veiculo (v): Cartão do veículo.

## 2.9 ETipoComunicacao

Indica o tipo de comunicação.

Valores:

- Serial: Indica que o sistema está se comunicando com o concentrador via porta serial;
- USB: Indica que o sistema está se comunicando com o concentrador via porta USB;
- TCP/IP: Indica que o sistema está se comunicando com o concentrador via LAN;
- Homologacao (não implementada): Modo de comunicação utilizado para comunicar com o sistema emulador de concentrador de bomba de gasolina.

## 2.10 EStatusGravacaoFirmware

Indica o estado em que a gravação do firmware se encontra.

Valores:

- ComandoProcessadoComSucesso (S): Indica que o comando foi enviado e recebido com sucesso;
- TamanhoMaiorQueOSuportadoPelaMemoria (T): Indica que o arquivo de firmware enviado é maior do que o suportado pelo concentrador;
- PonteiroDeMemoriaNaoInicializado (N): Indica que o comando de inicio de gravação não foi enviado;
- MemoriaCheia (M): Indica que a memória do concentrador está cheia;
- PerdaDePacote (P): Indica que um pacote foi perdido e deve ser reenviado;
- ErroDeGravacaoDeMemoria (R): Indica que ocorreu um erro e o processo de gravação deve ser reiniciado;
- NaoExisteAplicativoNovoNaMemoria (W): Indica que não existe firmware novo no concentrador para a gravação;
- CalculandoCheck (C): Indica que o concentrador está calculando o check.

## 2.11 StOptions

Indica o status de um bico.

Valores:

- Livre;
- Pronta;
- Falha;
- Concluiu;
- Abastecendo;
- Bloqueada;
- SolicitaLib.

## 2.12 Error

Indica a compatibilidade de protocolo

Valores:

- ErroString;
- None;
- ErroCodBico;
- ErroCaracterModo;
- ErroTimeOut;
- ErroResposta.

### ***2.13 EAcaoBico***

Tipo de ação para o método de liberação de abastecimento

Valores:

- Bloquear\_Cartao\_NOK
- Liberar\_Cartao\_Ok
- Liberar\_SemCartao

## 3 Interfaces

---

Interfaces descrevem comportamentos e características padrões que devem ser implementados pelas classes do sistema.

### 3.1 IComunicacao

Descreve as propriedades e métodos das classes que realizam a comunicação com o concentrador.

#### 3.1.1 *Propriedades*

As propriedades descrevem as características das classes que implementam a interface.

##### 3.1.1.1 Concentrador

**Nome:** Concentrador

**Tipo:** Int

**Descrição:** Busca ou seta o ID do concentrador.

##### 3.1.1.2 Parametros

**Nome:** Parametros

**Tipo:** List<string>

**Descrição:** Busca ou seta a lista de parâmetros de comunicação.

##### 3.1.1.3 RetornoComunicacao

**Nome:** RetornoComunicacao

**Tipo:** string

**Descrição:** Busca ou seta o retorno da comunicação.

##### 3.1.1.4 BytesRecebidos

**Nome:** Bytesrecebidos

**Tipo:** List<byte>

**Descrição:** Busca ou seta a lista de bytes recebidos do concentrador.

##### 3.1.1.5 ByteArray

**Nome:** ByteArray

**Tipo:** Byte[]

**Descrição:** Busca ou seta o array de bytes que será enviado ao concentrador.

##### 3.1.1.6 TX

**Nome:** TX

**Tipo:** bool

**Descrição:** Busca ou seta o status da comunicação entre o sistema e o concentrador.

##### 3.1.1.7 RX

**Nome:** RX

**Tipo:** bool

**Descrição:** Busca ou seta o status da comunicação entre o concentrador e o sistema.

### 3.1.1.8 PortaTCPFixa

**Nome:** PortaTCPFixa

**Tipo:** Int

**Descrição:** Quando conectar via rede TCP/IP, define uma porta TCP fixa para esta conexão. É possível especificar as portas 3001, 1771 ou 2001.

### 3.1.2 Métodos

Os métodos descrevem os comportamentos esperados das classes que implementam a interface.

#### 3.1.2.1 bool ComunicacaoConcentrador(string strComando)

Efetua a comunicação com o concentrador, é responsável pelo envio e recebimento de dados, pela validação dos dados recebidos e por setar o status da comunicação.

**Parâmetros de entrada:**

1 - strComando: string com o comando a ser enviado para o concentrador. O comando deve estar conforme o protocolo de comunicação<sup>1</sup>;

2 - timeout: Multiplicador pelo qual o timeout padrão deve ser multiplicado. Default: 1;

3 - abrirConexao: Indica se a conexão deve ser aberta antes do envio do comando. Default: True;

4 - fecharConexao: Indica se a conexão deve ser fechada ao final do envio do comando. Default: True.

**Retorno:** Valor booleano indicando se a comunicação foi efetuada com sucesso. “True” em caso de sucesso e “False” em caso de falha.

#### 3.1.2.2 EStatusDaComunicacaoConcentrador TestarComunicacao()

Testa a comunicação com o concentrador de acordo com os parâmetros definidos.

**Parâmetros de entrada:** Nenhum

**Retorno:** Status da comunicação conforme o item 2.7.

#### 3.1.2.3 void AbrirComunicacaoPorta()

Abre a comunicação com a porta (independente do tipo de comunicação).

**Parâmetros de entrada:** Nenhum.

**Retorno:** Nenhum.

#### 3.1.2.4 EStatusDaComunicacaoConcentrador TestarComunicacao()

Fecha a comunicação (independente do tipo de comunicação).

**Parâmetros de entrada:** Nenhum.

**Retorno:** Nenhum.

## 3.2 IComunicacaoHiro

---

<sup>1</sup> Ver documento Protocolo de Comunicação. O envio do comando adequado é transparente, sendo de responsabilidade da DLL. Nenhuma ação do integrador é necessária.

Descreve as propriedades e métodos da classe que realiza a troca de dados entre a DLL e o concentrador, independentemente do tipo de comunicação utilizada (serial, USB ou LAN).

### 3.2.1 Propriedades

As propriedades descrevem as características das classes que implementam a interface.

### 3.2.2 Métodos

Os métodos descrevem os comportamentos esperados das classes que implementam a interface.

#### 3.2.2.1 bool SetConfiguracoesNoRegistro(ETipoComunicacao tipoComunicacao, params string [] args)

Salva os parâmetros de configuração no registro do Windows.

##### Parâmetros de entrada:

- 1 - tipoComunicacao: Indica o tipo de comunicação (ver item 2.9);
- 2 - args: lista de strings com os dados para registro de log.

**Retorno:** Valor booleano indicando se a operação foi efetuada com sucesso. “True” em caso de sucesso e “False” em caso de falha.

#### 3.2.2.2 EStatusComando SetNumeroSerieConcentrador(string numeroSerie)

Seta o número de série do concentrador.

##### Parâmetros de entrada:

1. numeroSerie: string de 4 caracteres com o número de série do concentrador.

**Retorno:** EStatusComando (ver item 2.4).

#### 3.2.2.3 EStatusComando SetIDConcentrador(int idConcentrador)

Seta o número identificador do concentrador (inteiro de 0 a 9, enviar 0 se não tiver ID).

##### Parâmetros de entrada:

1. idConcentrador: Número identificador do concentrador.

**Retorno:** EStatusComando (ver item 2.4).

#### 3.2.2.4 Concentrador GetVersaoConcentrador()

Busca as informações de versão do concentrador.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Objeto do tipo Concentrador.

#### 3.2.2.5 EStatusComando SetRelogioConcentrador(string dataEHora)

Seta o relógio interno do concentrador.

##### Parâmetros de entrada:

1. dataEHora: String com a data e a hora a ser setada. Deve estar no formato AAMMDDHHmmSS sendo:  
  
AA = Ano.



MM= Mês.  
DD = Dia.  
HH = Hora.  
mm = Minutos.  
SS = segundos.

**Retorno:** EStatusComando (ver ítem 2.4).

### 3.2.2.6 DateTime GetRelogioConcentrador()

Busca a data e a hora interna do concentrador.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Data completa do concentrador. Tipo: DateTime.

### 3.2.2.7 EStatusComando SetConfiguracaoBico(Bico bico)

Seta as configurações de um bico no concentrador.

**Parâmetros de entrada:**

1. bico: Objeto do tipo Bico contendo as informações de configuração do bico.

**Retorno:** EStatusComando (ver ítem 2.4).

### 3.2.2.8 EStatusComando SetConfiguracaoBicoComLeitor(Bico bico)

Seta as configurações de um bico no concentrador, permite a configuração do leitor de cartão, para isso os dados da CPU (ID e porta) devem ser informados no objeto passado por parâmetro.

**Parâmetros de entrada:**

1. bico: Objeto do tipo Bico contendo as informações de configuração do bico.

**Retorno:** EStatusComando (ver ítem 2.4).

### 3.2.2.9 Bico GetConfiguracaoBico(int numeroBico)

Busca as configurações do bico no concentrador.

**Parâmetros de entrada:**

1. numeroBico: Número do bico.

**Retorno:** Objeto do tipo Bico com as configurações do bico. Caso não exista configuração retorna um objeto do tipo bico com o status = SemConfiguracao e as demais propriedades vazias.

### 3.2.2.10 Bico GetConfiguracaoBicoComLeitor(int numeroBico)

Busca as configurações do bico no concentrador. Retorna as informações de configuração de leitor de cartões.

**Parâmetros de entrada:**

1. numeroBico: Número do bico.

**Retorno:** Objeto do tipo Bico com as configurações do bico. Caso não exista configuração retorna um objeto do tipo bico com o status = SemConfiguracao e as demais propriedades vazias.

### 3.2.2.11 bool ApagarConfiguracao()

Apaga todas as configurações de bicos existentes no concentrador.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Valor booleano indicando o resultado da operação. *True* em caso de sucesso e *False* caso ocorra algum erro.

### 3.2.2.12 Bico GetStatusBico(int numeroBico)

Busca as informações de status do bico.

**Parâmetros de entrada:**

1. numeroBico: Número do bico.

**Retorno:** Objeto do tipo bico com as informações de status. Se o bico não existir retorna nulo.

### 3.2.2.13 Abastecimento GetLeituraDeAbastecimento()

Busca um abastecimento não lido no concentrador.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Objeto do tipo abastecimento. Caso não haja nenhum abastecimento não lido retorna nulo.

### 3.2.2.14 Abastecimento SetConfirmacaoLeituraAbastecimento(Abastecimento abastecimento)

Confirma a leitura de um abastecimento. Este método deve ser chamado sempre que a leitura de um abastecimento for feita com o método GetLeituraDeAbastecimento, para que o ponteiro do concentrador seja incrementado.

**Parâmetros de entrada:**

1. Objeto do tipo abastecimento.

**Retorno:** Objeto do tipo abastecimento.

Obs. Quando estes métodos são utilizados o abastecimento não poderá ser lido novamente.

### 3.2.2.15 Abastecimento GetLeituraDeAbastecimentoSemApagar()

Busca um abastecimento não lido no concentrador.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Objeto do tipo abastecimento. Caso não haja nenhum abastecimento não lido retorna nulo.

### 3.2.2.16 Abastecimento SetConfirmacaoLeituraAbastecimentoSemApagar(Abastecimento abastecimento)

Confirma a leitura de um abastecimento. Este método deve ser chamado sempre que a leitura de um abastecimento for feita com o método GetLeituraDeAbastecimentoSemApagar, para que o ponteiro do concentrador seja incrementado.

**Parâmetros de entrada:**

1. Objeto do tipo abastecimento.

**Retorno:** Objeto do tipo abastecimento.

---

*Obs. Quando estes métodos forem utilizados o abastecimento poderá ser lido novamente utilizando o método `GetLeituraDeAbastecimento`.*

---

### 3.2.2.17 bool ApagarAbastecimentos()

Apaga todos os abastecimentos existentes na memória do concentrador.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Valor booleano indicando se a operação foi realizada com sucesso. *True* em caso de sucesso e *False* em caso de falha.

### 3.2.2.18 string GetPrimeiroAbastecimentoNaoApagadoParaEnvio()

Retorna o endereço do primeiro abastecimento não enviado pelo concentrador.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Endereço do primeiro abastecimento ou string vazia caso não exista abastecimento.

### 3.2.2.19 Abastecimento GetMemoriaDeAbastecimentos(string enderecoAbastecimento)

Busca os dados de um abastecimento armazenado em um determinado endereço de memória.

**Parâmetros de entrada:**

1. enderecoAbastecimento: Endereço do abastecimento.

**Retorno:** Objeto do tipo Abastecimento com os dados do abastecimento.

### 3.2.2.20 Bico GetEncerrantesBico(int numeroBico)

Busca o valor dos encerrantes do bico no concentrador. De acordo com o modelo de bomba poderá retornar o encerrante de valor, o encerrante de litros ou ambos.

**Parâmetros de entrada:**

2. numeroBico: Número do bico.

**Retorno:** Objeto do tipo bico com os valores de encerrante. Caso nenhum bico seja encontrado irá retornar uma exceção.

### 3.2.2.21 Bico GetPrecoDoBico(int numeroBico)

Busca o preço do bico.

**Parâmetros de entrada:**

1. numeroBico: Número do bico.

**Retorno:** Objeto do tipo bico com o preço setado. Caso nenhum bico seja encontrado irá retornar uma exceção.

### 3.2.2.22 EStatusComando SetPrecoBico(int numeroBico, string valorBico)

Seta o preço a ser cobrado no bico.

**Parâmetros de entrada:**

1. numeroBico: Número do bico.
2. valorBico: Valor a ser cobrado no bico, deve ser informado completo sem a virgula Ex.: para R\$5,00 informar 500.

**Retorno:** EStatusComando.

### 3.2.2.23 EStatusComando DesbloquearBico(int numeroBico)

Libera um bico para que um abastecimento possa ser efetuado.

**Parâmetros de entrada:**

1. numeroBico: Número do bico.

**Retorno:** EStatusComando.

### 3.2.2.24 EStatusComando BloquearBico(int numeroBico)

Bloqueia o bico

**Parâmetros de entrada:**

1. numeroBico: Número do bico.

**Retorno:** EStatusComando.

**Observação:** Este comando não tem efeito no Simulador Hiro.

### 3.2.2.25 EStatusComando SetValorAbastecimento(int numeroBico, string valorAbastecimento)

Libera um abastecimento com o valor informado.

**Parâmetros de entrada:**

1. numeroBico: Número do bico.
2. valorBico: Valor a ser cobrado no bico, deve ser informado completo sem a virgula Ex.: para R\$5,00 informar 500.

**Retorno:** EStatusComando.

**Observação:** Este comando não tem efeito no Simulador Hiro.

### 3.2.2.26 EStatusComando SetLitrosAbastecimento(int numeroBico, string litrosAbastecimento)

Libera um abastecimento com o total de litros informado.

**Parâmetros de entrada:**

1. numeroBico: Número do bico.
2. litrosAbastecimento: Total de litros a ser abastecido.

**Retorno:** EStatusBico.

**Observação:** Este comando não tem efeito no Simulador Hiro.

### 3.2.2.27 EStatusComando SetCadastroDeCartao(ETipoCartao tipo, string codigo)

Efetua o cadastro de um cartão no concentrador.

**Parâmetros de entrada:**

1. tipo: Tipo do cartão (ver ítem 2.8).
2. codigo: Código do cartão (4 caracteres).

**Retorno:** EStatusComando.

**3.2.2.28 Cartao GetCadastroDeCartao(ETipoCartao tipo, string endereco)**

Pesquisa os dados de um cartão através de seu tipo e endereço.

**Parâmetros de entrada:**

1. tipo: TipoCartao (ver ítem 2.8), endereço: Endereço do cartão.
2. Endereço: Funcionarios = 0 a 999  
Clientes = 0 a 3999  
Motoristas = 0 a 3999  
Veiculos = 0 a 3999

**Retorno:** Objeto do tipo Cartao com os dados do cartão.

**3.2.2.29 Cartao GetCadastroDeCartao(string codigo, ETipoCartao tipo)**

Pesquisa os dados de um cartão através de seu tipo e código.

**Parâmetros de entrada:**

1. codigo: Código do cartão.
2. tipo: TipoCartao (ver ítem 2.8).

**Retorno:** Objeto do tipo Cartao com os dados do cartão.

**3.2.2.30 List<Cartao> GetArquivoCartoes(ETipoCartao tipo)**

Pesquisa todos os cartões cadastrados de um determinado tipo.

**Parâmetros de entrada:**

1. tipo: TipoCartao (ver ítem 2.8).

**Retorno:** Objeto do tipo Cartao com os dados do cartão.

**3.2.2.31 EStatusComando DeletaRegistroDeCartao(ETipoCartao tipo)**

Deleta todo o registro de cartões de acordo com o tipo informado.

**Parâmetros de entrada:**

1. tipo: Tipo do cartão (ver ítem 2.8)

**Retorno:** EStatusComando.

**3.2.2.32 EStatusComando DeletaRegistroDeCartao(string código, ETipoCartao tipo)**

Deleta um cartão cadastrado pelo código e tipo.

**Parâmetros de entrada:**

1. codigo: Código do cartão.
2. tipo: Tipo do cartão (ver ítem 2.8)

**Retorno:** EStatusComando.

### 3.2.2.33 EStatusComando DeletaRegistroDeCartao(ETipoCartao tipo, string endereco)

Deleta um cartão pelo tipo e endereço de memória.

**Parâmetros de entrada:**

1. tipo: Tipo do cartão (ver item 2.8);
2. endereco: Endereço de memória do cartão:  
Clientes = 0 a 3999  
Motoristas = 0 a 3999  
Veiculos = 0 a 3999

**Retorno:** EStatusComando.

### 3.2.2.34 EStatusComando SetConfiguracaoLan(SConfiguracaoTCPIP configuracao)

Seta as configurações da comunicação TCP/IP no concentrador.

**Parâmetros de entrada:**

1. Configuração: Estrutura do tipo SConfiguracaoTCPIP (ver descrição neste documento) com os dados da configuração a serem informados ao concentrador.

**Retorno:** EStatusComando.

### 3.2.2.35 SConfiguracaoTCPIP GetConfiguracaoLan()

Busca as configurações da comunicação TCP/IP salvas no concentrador.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Estrutura de dados do tipo SConfiguracaoTCPIP com os dados da configuração salva no concentrador.

### 3.2.2.36 EStatusComando SetConfiguracaoServidorSMTP(SConfiguracaoEmail configuracao)

Seta as configurações do servidor de envio de e-mail no concentrador.

**Parâmetros de entrada:**

1. Configuração: Estrutura do tipo SConfiguracaoEmail (ver descrição neste documento) com os dados da configuração a serem informados ao concentrador.

**Retorno:** EStatusComando.

### 3.2.2.37 void GetConfiguracaoServidorSMTP(ref SConfiguracaoEmail configuracao)

Busca as configurações do servidor de envio de e-mail salvas no concentrador.

**Parâmetros de entrada:**

1. Configuração: Estrutura do tipo SConfiguracaoEmail (ver descrição neste documento) que irá receber os dados da configuração (por referência).

**Retorno:** Estrutura de dados do tipo SConfiguracaoEmail com os dados da configuração salva no concentrador (por referência).

### 3.2.2.38 SConfiguracaoEmail GetConfiguracaoServidorSMTP2()

Busca as configurações do servidor de envio de e-mail salvas no concentrador.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Estrutura de dados do tipo SConfiguracaoEmail com os dados da configuração salva no concentrador.

### 3.2.2.39 EStatusComando SetConfiguracaoUsuarioSMTP(SConfiguracaoEmail configuracao)

Seta as configurações do usuário de envio de e-mail no concentrador.

**Parâmetros de entrada:**

1. Configuração: Estrutura do tipo SConfiguracaoEmail (ver descrição neste documento) com os dados da configuração a serem informados ao concentrador.

**Retorno:** EStatusComando.

### 3.2.2.40 void GetConfiguracaoUsuarioSMTP (ref SConfiguracaoEmail configuracao)

Busca as configurações do usuário de envio de e-mail salvas no concentrador.

**Parâmetros de entrada:**

1. Configuração: Estrutura do tipo SConfiguracaoEmail (ver descrição neste documento) que irá receber os dados da configuração (por referência).

**Retorno:** Estrutura de dados do tipo SConfiguracaoEmail com os dados da configuração salva no concentrador (por referência).

### 3.2.2.39 SConfiguracaoEmail GetConfiguracaoUsuarioSMTP2 ()

Busca as configurações do usuário de envio de e-mail salvas no concentrador.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Estrutura de dados do tipo SConfiguracaoEmail com os dados da configuração salva no concentrador.

### 3.2.2.40 EStatusComando SetDestinatarioSMTP(SDestinatarioSMTP destinatario)

Seta um destinatário e-mail no concentrador.

**Parâmetros de entrada:**

1. Destinatario: Estrutura do tipo SetDestinatarioSMTP (ver descrição neste documento) com os dados da configuração a serem informados ao concentrador.

**Retorno:** EStatusComando.

### 3.2.2.41 SDestinatarioSMTP GetDestinatarioSMTP(int Codigo, char Tipo)

Busca as configurações de um destinatário de e-mail salvo no concentrador.

**Parâmetros de entrada:**

1. Codigo: Código do destinatário (valor de 1 a 4);
2. Tipo: Tipo de destinatário ("T" Técnica, "P" posto).

**Retorno:** Estrutura de dados do tipo SDestinatarioSMTP com os dados da configuração salva no concentrador (por referência).

### 3.2.2.42 EStatusComando SetTipoCombustivel(SCombustivel combustivel)

Seta um tipo de combustível no concentrador.

**Parâmetros de entrada:**

1. Combustível: Estrutura de dados do tipo SCombustivel ( ver descrição no final do documento) com os dados do combustível a ser salvo.

**Retorno:** EStatusComando.

**3.2.2.43 SCombustivel GetTipoCombustivel(int Codigo)**

Busca as configurações de um tipo de combustível salvo no concentrador.

**Parâmetros de entrada:**

1. Codigo: Código do combustível (valor de 1 a 15)

**Retorno:** Estrutura de dados do tipo SCombustivel com os dados da configuração salva no concentrador (por referência).

**3.2.2.44 EStatusComando SetHorarioEnvioTotais(SHorarioEnvioTotais horario)**

Seta um horário para envio de e-mails no concentrador.

**Parâmetros de entrada:**

1. Horario: Estrutura de dados do tipo SHorarioEnvioTotais ( ver descrição no final do documento) com os dados do horário a ser salvo.

**Retorno:** EStatusComando.

**3.2.2.45 SHorarioEnvioTotais GetHorarioEnvioTotais(int Codigo)**

Busca as configurações de um tipo de combustível salvo no concentrador.

**Parâmetros de entrada:**

1. Codigo: Código do horário (valor de 1 a 4)

**Retorno:** Estrutura de dados do tipo SHorarioEnvioTotais com os dados da configuração salva no concentrador (por referência).

**3.2.2.46 EStatusComando StartConfiguracaoDeLeitorDeCartoes()**

Entra no modo de configuração de leitor de cartão.

**Parâmetros de entrada:**

- 1 - slot: Slot com o qual o leitor irá se comunicar;
- 2 - canal: Canal com o qual o leitor irá se comunicar.

**Retorno:** EStatusComando.

**Observação:** Este comando não tem efeito no Simulador Hiro.

**3.2.2.45 EStatusComando EndConfiguracaoDeLeitorDeCartoes ()**

Sai do modo de configuração de leitor de cartões.

**Parâmetros de entrada:**

- 1 - Codigo: Código do horário (valor de 1 a 4)

**Retorno:** EStatusComando.



### 3.2.2.46 EStatusGravacaoFirmware StartGravacaoFirmware(long tamanhoArquivo)

Inicia o envio e gravação de firmware no concentrador.

**Parâmetros de entrada:**

1 - tamanhoArquivo: Tamanho do arquivo de firmware que será enviado para o concentrador.

**Retorno:** EStatusGravacaoFirmware.

### 3.2.2.47 EStatusGravacaoFirmware SendPacoteFirmware(byte[] buffer, int indexPacote)

Envia um pacote do firmware para o concentrador.

**Parâmetros de entrada:**

1 - buffer: Pacote de dados (0 a 200 bytes hexa);

2 - indexPacote: identificador do pacote, deve ser sequencial. Retornar para 0 quando ultrapassar 255

**Retorno:** EStatusGravacaoFirmware.

Se retorno = PerdaDePacote então o pacote deve ser enviado novamente (máximo 3 tentativas depois reiniciar o envio);

Se retorno = ComandoProcessadoComSucesso então envia o próximo pacote;

Qualquer outro valor de retorno o processo deve ser reiniciado.

### 3.2.2.48 EStatusGravacaoFirmware StopGravacaoFirmware()

Encerra o envio de firmware para o concentrador.

**Parâmetros de entrada:** Nenhum.

**Retorno:** EStatusGravacaoFirmware.

### 3.2.2.49 EStatusGravacaoFirmware ExecuteFirmware()

Indica que o Hiro (concentrador) deve inicializar o novo firmware enviado.

**Parâmetros de entrada:** Nenhum.

**Retorno:** EStatusGravacaoFirmware.

### 3.2.2.50 string LeStatusTodosBicos()

Efetua uma única leitura para obter o status de todos os bicos.

**Parâmetros de entrada:** Nenhum.

**Retorno:** string com 100 posições, sendo cada posição com um caracter indicando o status do bico correspondente. Posição 1 = bico 1, Posição 2, bico 2, etc.

Status:

L = bico livre

O = bico abastecendo

D = sem comunicação

E = Livre bloqueada

S = Solicitando liberação

C = Solicitando autorização para um cartão: (neste caso deve ser solicitado qual o cartão pelo comando LB.)

N = Bico Não configurado  
B = Bico chaveado (bloqueado pelo sistema)

### 3.2.2.51 Abastecimento GetLeituraDeAbastecimentoPAFECF ()

Busca um abastecimento não lido no concentrador.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Objeto do tipo abastecimento. Caso não haja nenhum abastecimento não lido retorna nulo. Este método retorna algumas propriedades a mais (p.e. encerrante inicial).

### 3.2.2.52 Abastecimento GetLeituraDeAbastecimentoPT ()

Busca um abastecimento não lido no concentrador.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Objeto do tipo abastecimento. Caso não haja nenhum abastecimento não lido retorna nulo. Este método retorna algumas propriedades a mais (p.e. tempo de abastecimento).

### 3.2.2.53 SMedidor GetStatusMedidor (int NumeroMedidor)

Efetua a leitura do medidor de tanque de combustíveis, caso exista integração.

**Parâmetros de entrada:** Número (int) do medidor/tanque de combustíveis.

**Retorno:** Objeto do tipo SMedidor.

### 3.2.2.54 EStatusComando LiberacaoAbastece(int numeroBico, Hiro.Enumeracoes.EAcaoBico Acao, string LimiteValor2D, string LimiteLitros)

Efetua uma liberação de abastecimento no bico especificado.

**Parâmetros de entrada:**

1. Número do bico.
2. Ação (ver 2.13 EAcaoBico).
3. Limite do valor a ser abastecido, informado completo sem a virgula Ex.: para R\$5,00 informar 500.
4. Limite de litros a ser abastecido.

**Retorno:** Objeto do tipo EStatusComando.

**Observação:** Este comando não tem efeito no Simulador Hiro.

## 3.2.3 Métodos de Compatibilidade

Esta seção descreve uma série de métodos com assinaturas compatíveis com as mais diversas linguagens, como VB e Cobol.

### 3.2.3.1 Error CobAlteraPreco(string dados)

Seta o preço a ser cobrado no bico.

**Parâmetros de entrada:**

1. dados: Número do bico e novo preço a ser cobrado. Formato: BBPPPP onde: BB = Número do bico e PPPP = Novo valor.

**Retorno:** Resultado da execução da operação. "None" indica que não ocorreu nenhum erro. Ver estrutura de dados Error.

### 3.2.3.2 void CobLeEnc(ref Enc dados)

Faz a leitura do encerrante do bico de acordo com os dados informados como parâmetro.

**Parâmetros de entrada:**

1. dados: Estrutura com as informações do bico (número e tipo de encerrante desejado). O valor do encerrante solicitado será adicionado à estrutura. Caso o bico não seja encontrado o valor do encerrante é "0".

**Retorno:** O valor do encerrante solicitado é retornado no campo "valor" do parâmetro "dados" passado por referência.

### 3.2.3.3 void CobLePPL(ref string dados)

Faz a leitura do encerrante do bico de acordo com os dados informados como parâmetro.

**Parâmetros de entrada:**

1. dados: string de duas posições com o número do bico.

**Retorno:** O valor cobrado no bico é informado na variável passada por referência.

### 3.2.3.4 void CobLeStructIDSt(ref Abast3 dados)

Faz a leitura de um abastecimento e incrementa o ponteiro. Adiciona a informação de ID do abastecimento.

**Parâmetros de entrada:**

1. dados: Estrutura de dados do tipo Abast3 que irá receber os dados do abastecimento lido.

**Retorno:** Estrutura de dados abast com os dados do abastecimento lido (por referência). Caso não tenha encontrado nenhum abastecimento o campo VALUE = "FALSE".

### 3.2.3.5 void CobLeStructSt(ref Abast2 dados)

Faz a leitura de um abastecimento e incrementa o ponteiro. Adiciona a informação de ID do abastecimento.

**Parâmetros de entrada:**

1. dados: Estrutura de dados do tipo Abast2 que irá receber os dados do abastecimento lido.

**Retorno:** Estrutura de dados abast com os dados do abastecimento lido (por referência). Caso não tenha encontrado nenhum abastecimento o campo VALUE = "FALSE".

### 3.2.3.6 void CobLeVis(ref visualizacao dados)

Faz a leitura do status de todos os abastecimentos em andamento.

**Parâmetros de entrada:**

1. dados: Estrutura de dados que irá armazenar as informações dos abastecimentos em andamento.

**Retorno:** Campo stfull preenchido com os dados dos abastecimentos no formato "BBBBLLLLBBBBLLLLBBBBLLLL..." onde "BB" corresponde ao número do bico e "LLLLLL" a quantidade de litros abastecidos no momento da leitura.

### 3.2.3.7 Error CobPreset(ref string dados)

Faz a liberação de um abastecimento com a quantidade de litros informada.

**Parâmetros de entrada:**

1. dados: String (por referência) no formato "BBLLLLLL" onde "BB" é o número do bico e "LLLLLL" a quantidade de litros.

**Retorno:** Estrutura do tipo Error com o resultado da operação, onde "None" indica que a operação foi executada com sucesso.

### 3.2.3.8 bool CobSetClock(ref string dados)

Seta o relógio do concentrador.

**Parâmetros de entrada:**

1. dados: "AUTO" para ajuste automático utilizando horário atual do computador "ddhhmm" para ajuste manual, representando dia, hora e minuto.

**Retorno:** "True" se a operação foi executada com sucesso. "False" em caso de falha.

### 3.2.3.9 Error AlteraPreco(string bico, double preco, byte decimais)

Seta o preço a ser cobrado no bico.

**Parâmetros de entrada:**

1. bico: Número do bico.
2. preco: Valor a ser cobrado no bico, decimais: quantidade de casas decimais utilizadas no bico.

**Retorno:** Resultado da execução da operação. "None" indica que não ocorreu nenhum erro. Ver estrutura de dados Error.

### 3.2.3.10 Error AutoLibera(string bico)

Libera o bico para que abastecimentos possam ser efetuados.

**Parâmetros de entrada:**

1. bico: Número do bico.

**Retorno:** Resultado da execução da operação. "None" indica que não ocorreu nenhum erro. Ver estrutura de dados Error.

### 3.2.3.11 Error BloqueiaBico(string bico)

Bloqueia o bico para que abastecimentos não possam ser efetuados.

**Parâmetros de entrada:**

1. bico: Número do bico.

**Retorno:** Resultado da execução da operação. "None" indica que não ocorreu nenhum erro. Ver estrutura de dados Error.

### 3.2.3.12 Encerrante ConsultaEncerrante(char modo, string bico)

Busca o encerrante do bico de acordo com o tipo solicitado.

**Parâmetros de entrada:**

1. modo: Tipo de encerrante desejado. '\$' Encerrante valor, 'L' Encerrante Litros.
2. bico: Número do bico.

**Retorno:** Estrutura do tipo Encerrante com o número do bico e o valor do encerrante solicitado.

### 3.2.3.13 abast LeAbastecimento()

Faz a leitura de um abastecimento e incrementa o ponteiro.

**Parâmetros de entrada:**

1. Nenhum.

**Retorno:** Estrutura do tipo abast com os dados do abastecimento. Caso não tenha encontrado nenhum abastecimento o total de litros e de valor estarão com valor 0.

### 3.2.3.14 abast LeAbFix()

Faz a leitura de um abastecimento e não incrementa o ponteiro.

**Parâmetros de entrada:**

1. Nenhum.

**Retorno:** Estrutura do tipo abast com os dados do abastecimento. Caso não tenha encontrado nenhum abastecimento o total de litros e de valor estarão com valor 0.

### 3.2.3.15 Abastecimento LeRegistro(int NumReg)

Busca os dados de um abastecimento armazenado em um determinado endereço de memória.

**Parâmetros de entrada:**

1. NumReg: Endereço do abastecimento.

**Retorno:** Objeto do tipo Abastecimento com os dados do abastecimento.

### 3.2.3.16 string LePart(char option)

Faz a leitura de um abastecimento (não incrementa o ponteiro) e retorna a informação solicitada de acordo com o parâmetro informado.

**Parâmetros de entrada:**

1. option: char que indica a informação desejada.
  - ☐ L: Litros abastecidos.
  - ☐ T: Valor abastecido.
  - ☐ P: Valor unitário.
  - ☐ C: Data e hora.
  - ☐ E: Encerrantes.

**Retorno:** Número do bico (dois caracteres) + Informação solicitada para:

L: Número do Bico + Total de litros abastecidos.

T: Número do Bico + Valor abastecido.

P: Número do bico + Valor unitário.

C: Número do bico + Data e hora.

E: Número do bico + Encerrante litros (10 caracteres) + Encerrante valor (10 caracteres).

### 3.2.3.17 double LePPL(string bico)

Busca o valor unitário praticado no bico.

**Parâmetros de entrada:**

1. bico: Número do bico.

**Retorno:** Valor unitário praticado no bico.

### 3.2.3.18 MultiStatus LeStatus()

Faz a leitura do status de todos os abastecimentos em andamento.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Array de 100 posições com o status de cada bico.

### 3.2.3.19 StStatus2 LeStatusVB()

Faz a leitura do status de todos os abastecimentos em andamento.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Array de string de 100 posições com o status de cada bico.

### 3.2.3.20 string LeSTEncerrante(string modo, string bico)

Busca o encerrante do bico de acordo com o tipo solicitado.

**Parâmetros de entrada:**

1. modo: Tipo de encerrante desejado. '\$' Encerrante valor, 'L' Encerrante Litros.
2. Bico: Valor de 1 – 100, número do bico.

**Retorno:** Valor do encerrante solicitado.

### 3.2.3.21 string LeStReduzida()

Faz a leitura do abastecimento na memória.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Abastecimento no formato:

"TTTTTLLLLLPPPPVVCCCCBBDDHHMMKK"

Ou "0" se nenhum abastecimento na memória.

Detalhe:

TTTTTT: Total a Pagar; (bombas mecânicas retornam "000000");

LLLLLL: Volume abastecido (Litros);

PPPP: Preço unitário;

VV: Código de vírgula;

CCCC: Tempo de abastecimento (Hexadecimal);

BB: Código de bico;

DD: Dia;

HH: Hora;

MM: Minuto;

KK: Checksum.

### 3.2.3.22 string LeStRegistro(int endereco)

Busca o encerrante do bico de acordo com o tipo solicitado.

**Parâmetros de entrada:**

1. endereço: Endereço de memória do abastecimento.

**Retorno:** Abastecimento no formato:

"TTTTTLLLLLLPPPPVVCCCCBDDHHMMKK"

Ou "0" se nenhum abastecimento na memória.

Detalhe:

TTTTTT: Total a Pagar; (bombas mecânicas retornam "000000");

LLLLLL: Volume abastecido (Litros);

PPPP: Preço unitário;

VV: Código de vírgula;

CCCC: Tempo de abastecimento (Hexadecimal);

BB: Código de bico;

DD: Dia;

HH: Hora;

MM: Minuto;

KK: Checksum.

### 3.2.3.23 string LeStringAb(ref string resposta)

Busca o encerrante do bico de acordo com o tipo solicitado.

**Parâmetros de entrada:**

1. resposta: string que receberá a resposta por referência.

**Retorno:** Abastecimento no formato:

"TTTTTLLLLLLPPPPVVCCCCBDDHHMMKK"

Ou "0" se nenhum abastecimento na memória.

Detalhe:

TTTTTT: Total a Pagar; (bombas mecânicas retornam "000000");

LLLLLL: Volume abastecido (Litros);

PPPP: Preço unitário;

VV: Código de vírgula;

CCCC: Tempo de abastecimento (Hexadecimal);

BB: Código de bico;

DD: Dia;

HH: Hora;

MM: Minuto;

KK: Checksum.

### 3.2.3.24 string LeStringAbVB ()

Faz a leitura do abastecimento na memória.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Abastecimento no formato:

"TTTTTLLLLLLPPPPVVCCCCBDDHHMMKK"

Ou "0" se nenhum abastecimento na memória.

Detalhe:

TTTTTT: Total a Pagar; (bombas mecânicas retornam "000000");

LLLLLL: Volume abastecido (Litros);

PPPP: Preço unitário;  
VV: Código de vírgula;  
CCCC: Tempo de abastecimento (Hexadecimal);  
BB: Código de bico;  
DD: Dia;  
HH: Hora;  
MM: Minuto;  
KK: Checksum.

### 3.2.3.25 void LeStringX(ref Retorno2 retorno)

Faz a leitura do abastecimento na memória.

#### Parâmetros de entrada:

1. retorno: Estrutura do tipo retorno que receberá os dados do abastecimento.

**Retorno:** Campo value da estrutura passada por referência com os dados do abastecimento no formato:

"TTTTTLLLLLLPPPPVVBBDDHHMMKK"

Ou "0" se nenhum abastecimento na memória.

Detalhe:

TTTTTT: Total a Pagar; (bombas mecânicas retornam "000000");

LLLLLL: Volume abastecido (Litros);

PPPP: Preço unitário;

VV: Código de vírgula;

BB: Código de bico;

DD: Dia;

HH: Hora;

MM: Minuto;

KK: Checksum.

### 3.2.3.26 stEncerrante LeStructEncerrante(string modo, string bico)

Busca o encerrante do bico de acordo com o tipo solicitado.

#### Parâmetros de entrada:

1. modo: Tipo de encerrante desejado. '\$' Encerrante valor, 'L' Encerrante Litros.
2. Bico: Valor de 1 – 100, número do bico.

**Retorno:** Estrutura do tipo stEncerrante com os dados do encerrante solicitado.

### 3.2.3.27 stPPI LeStructPPL(string bico)

Busca o valor unitário cobrado no bico.

#### Parâmetros de entrada:

1. Bico: Valor de 1 – 100, número do bico.

**Retorno:** Estrutura do tipo stPPI com os dados solicitados.

### 3.2.3.28 void LeStructSt(ref Abast2 dados)

Faz a leitura de um abastecimento e incrementa o ponteiro. Adiciona a informação de ID do abastecimento.



**Parâmetros de entrada:**

1. dados: Estrutura de dados do tipo Abast2 que irá receber os dados do abastecimento lido.

**Retorno:** Estrutura de dados abast com os dados do abastecimento lido (por referência). Caso não tenha encontrado nenhum abastecimento o campo VALUE = "FALSE".

**3.2.3.29 StStatus LeStStatus()**

Faz a leitura do status de todos os bicos. (Le as 100 posições possíveis).

**Parâmetros de entrada:** Nenhum.

**Retorno:** Estrutura do tipo StStatus com o status de cada bico.

Tabela de status:

- L Bomba encontra-se livre para abastecer.
- B Bomba bloqueada para realizar abastecimentos.
- A Bomba está em processo de abastecimento.
- E Bomba está aguardando liberação da automação para iniciar o processo de abastecimento.
- F Bomba não presente ou em falha.

**3.2.3.30 string LeStStatus2()**

Faz a leitura do status de todos os bicos. (Le as 100 posições possíveis).

**Parâmetros de entrada:** Nenhum.

**Retorno:** String de 101 posições, sendo a primeira "S" para indicar que é status e as demais com o status de cada bico.

Tabela de status:

- L Bomba encontra-se livre para abastecer.
- B Bomba bloqueada para realizar abastecimentos.
- A Bomba está em processo de abastecimento.
- E Bomba está aguardando liberação da automação para iniciar o processo de abastecimento.
- F Bomba não presente ou em falha.

**3.2.3.31 OnLine LeVisualizacao()**

Faz a leitura do status de todos os bicos. (Le as 100 posições possíveis).

**Parâmetros de entrada:** Nenhum.

**Retorno:** Estrutura do tipo OnLine com os dados de todos os bicos. As posições que não possuem configuração será retornado o valor "00" no número do bico.

**3.2.3.32 Error Preset(string bico, double valor)**

Libera um abastecimento conforme o valor informado.

**Parâmetros de entrada:**

1. Bico: Número do bico que irá efetuar o abastecimento.
2. valor: Valor a ser liberado.

**Retorno:** Enum do tipo Error com o resultado da operação.

**3.2.3.33 void RefAltPreco(string bico, double preco, byte decimais, ref Error status)**

Seta o preço que será cobrado no bico.

**Parâmetros de entrada:**

1. Bico: Número do bico que irá efetuar o abastecimento.
2. preco: Novo preço a ser cobrado no bico.
3. Decimais: Número de casas decimais depois da vírgula.
4. Status (por referência): Enum do tipo error com o resultado da operação.

**Retorno:** Enum do tipo Error com o resultado da operação (por referência)..

### 3.2.3.34 RefAutoLibera(string bico, ref Error status)

Libera o bico para que abastecimentos possam ser efetuados.

**Parâmetros de entrada:**

1. Bico: Número do bico.
2. Status (por referência): Enum do tipo error com o resultado da operação.

**Retorno:** Enum do tipo Error com o resultado da operação (por referência).

### 3.2.3.35 RefBloqueiaBico(string bico, ref Error status)

Bloqueia o bico para que abastecimentos não possam ser efetuados.

**Parâmetros de entrada:**

1. Bico: Número do bico.
2. Status (por referência): Enum do tipo error com o resultado da operação.

**Retorno:** Enum do tipo Error com o resultado da operação (por referência).

### 3.2.3.36 bool SetClock(string dados)

Seta o relógio do concentrador.

**Parâmetros de entrada:**

1. dados: "AUTO" para ajuste automático utilizando horário atual do computador ou "ddhhmm" para ajuste manual, representando dia, hora e minuto.

**Retorno:** "True" se a operação foi executada com sucesso. "False" em caso de falha.

### 3.2.3.37 bool SetIntClock(byte dia, byte hora, byte minuto)

Seta o relógio do concentrador.

**Parâmetros de entrada:**

1. dia: Dia.
2. Hora: Hora.
3. Minuto: Minuto.

**Retorno:** "True" se a operação foi executada com sucesso. "False" em caso de falha.

### 3.2.3.38 bool SetPreset(string dados)

Libera um abastecimento cm o valor informado.

**Parâmetros de entrada:**

1. Dados: Número do bico e valor a ser liberado no formato "BBVVVVVV" onde "BB" é o número do bico e "VVVVVV" é o valor do abastecimento, deve incluir o valor completo (inteiro e decimais) sem a vírgula.

**Retorno:** "True" se a operação foi executada com sucesso. "False" em caso de falha.

### 3.2.3.39 string STVisualizacao(ref string dados)

Faz a leitura de todos os abastecimentos em andamento.

**Parâmetros de entrada:**

1. Dados: String passada por referência para receber todos os abastecimentos em andamento.

**Retorno:** String com todos os abastecimentos em andamento, possui o mesmo valor que a string passada por referência. Formato: "BBVVVVVVBBVVVVVV..." onde "BB" é o número do bico e "VVVVVV" é o valor na bomba na hora da leitura.

### 3.2.3.40 AbastVB VBLeAbastecimento()

Faz a leitura de um abastecimento e incrementa o ponteiro.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Estrutura de dados AbastVB com os dados do abastecimento lido. Caso não tenha encontrado nenhum abastecimento o campo VALUE = 0.

### 3.2.3.41 void VBLePPL(ref string dados)

Ler o preço unitário praticado por determinado bico.

**Parâmetros de entrada:**

1. Dados: "BB", representando o bico que desejamos consultar.

**Retorno:** String "PPPP" (por referência), representando o preço unitário praticado no bico consultado.

### 3.2.3.42 VBOnline VBLeVisualizacao()

Ler o volume que todos os bicos estão abastecendo na hora da consulta. Faz a leitura das 100 posições).

**Parâmetros de entrada:** Nenhum.

**Retorno:** Estrutura do tipo VBOnLine com os dados de todos os bicos. As posições que não possuírem configuração será retornado o valor "00" no número do bico.

### 3.2.3.43 void VBSetAutoLibera(ref string bico)

Libera o bico para que abastecimentos possam ser efetuados.

**Parâmetros de entrada:**

1. Dados: "BB", representando o bico que desejamos liberar.

**Retorno:** A string passada por referência recebe "00" em caso de falha, do contrário retorna o número do bico.

### 3.2.3.44 void VBSetBloqueiaBico(ref string bico)

Bloqueia o bico para que abastecimentos possam ser efetuados.

**Parâmetros de entrada:**

1. Dados: "BB", representando o bico que desejamos bloquear.

**Retorno:** A string passada por referência recebe "00" em caso de falha, do contrário retorna o número do bico.

**3.2.3.45 void VBSetPPL(ref string dados)**

Seta o preço a ser cobrado no bico.

**Parâmetros de entrada:**

1. Dados: Número do bico e novo preço a ser cobrado. Formato: "BBPPP" onde: "BB" = Número do bico e "PPP" = Novo valor.

**Retorno:** Por referência:

- "?b": Em caso de parâmetro inválido ou bico inexistente;
- "?t": Em caso de erro no processamento;
- "Bb": Em caso de sucesso.

**3.2.3.46 bool Inicializaserial(byte np)**

Inicializa a comunicação serial.

**Parâmetros de entrada:**

1. np: Número da porta.

**Retorno:** True: em caso de sucesso. False em caso de falha.

**3.2.3.47 long FechaSerial();**

Encerra a comunicação serial.

**Retorno:** 0: em caso de sucesso. 1 em caso de falha.

**3.2.3.48 Error AutorizaAbast(string bico)**

Libera o bico para que abastecimentos possam ser efetuados.

**Parâmetros de entrada:**

1. bico: Número do bico.

**Retorno:** Resultado da execução da operação. "None" indica que não ocorreu nenhum erro.

**3.2.3.49 void RefPreset(string bico, double valor, ref Error status)**

Libera um abastecimento no valor definido.

**Parâmetros de entrada:**

1. Bico: Número do bico;
2. Valor: Valor do abastecimento que será liberado;
3. Status: Variável do tipo Error que receberá o retorno da operação.

**Retorno:** Por referência: Resultado da execução da operação. "None" indica que não ocorreu nenhum erro.

### 3.2.3.50 EStatusComando SetPrecoBicoNivel (int numeroBico, int Nivel, string valorBico)

Seta o preço a ser cobrado no bico.

**Parâmetros de entrada:**

1. numeroBico: Número do bico.
2. Nivel: Nível do cartão.
3. valorBico: Valor a ser cobrado no bico, deve ser informado completo sem a virgula Ex.: para R\$5,00 informar 500.

**Retorno:** EStatusComando.

### 3.2.3.21 Bico GetPrecoDoBicoNivel(int numeroBico, int Nivel)

Busca o preço do bico.

**Parâmetros de entrada:**

1. numeroBico: Número do bico.
2. Nivel: Nível do cartão.

**Retorno:** Objeto do tipo bico com o preço setado. Caso nenhum bico seja encontrado irá retornar uma exceção.

### 3.2.3.22 EStatusComando LiberaAbastecelD(int numeroBico, UInt16 ID, Char Campo\_ID, Char Liberacao, , string LimiteValor2D, string LimiteLitros)

Busca o preço do bico.

**Parâmetros de entrada:**

1. numeroBico: Número do bico.
2. ID: Identificação do Abastecimento ou da Venda ou Documento etc.
3. Campo\_ID: 'F' = Campo Frentista ou 'C' = Campo Cliente.
4. Liberacao: 'S' = Libera a bomba sem necessidade de identificação do frentista  
– ou 'N' = Libera a bomba somente com a identificação do frentista.
5. LimiteValor2D: Limite do valor a ser abastecido, informado completo sem a virgula Ex.: para R\$50,00 informar 5000. Informar 00 para não impor limite de valor.
6. LimiteLitros: Limite de litros a ser abastecido. Enviar 00 para não limitar a quantidade de litros no abastecimento.

**Retorno:** EStatusComando.

### 3.3 IComunicacaoDelphi

Interface criada para permitir e facilitar a utilização da Hiro.dll em sistema Delphi. Descreve as mesmas características e comportamentos das interfaces Icomunicacao e IcomunicacaoHiro juntas.

### 3.4 IConcentrador

Representa a entidade física concentrador, descrevendo suas características e comportamentos.

#### 3.4.1 *Propriedades*

As propriedades descrevem as características das classes que implementam a interface.

##### 3.4.1.1 NumeroSerie

**Nome:** NumeroSerie

**Tipo:** string

**Descrição:** Busca ou seta o número de série do concentrador.

##### 3.4.1.2 ID

**Nome:** ID

**Tipo:** int

**Descrição:** Busca ou seta o identificador do concentrador.

##### 3.4.1.3 VersaoHardware

**Nome:** VersaoHardware

**Tipo:** string

**Descrição:** Busca ou seta a versão do hardware do concentrador.

##### 3.4.1.4 FirmWareBIOS

**Nome:** FirmWareBIOS

**Tipo:** string

**Descrição:** Busca ou seta a versão do firmware da BIOS do concentrador.

##### 3.4.1.5 FirmWareAplicativo

**Nome:** FirmWareAplicativo

**Tipo:** string

**Descrição:** Busca ou seta a versão do firmware do aplicativo do concentrador.

##### 3.4.1.6 DataHora

**Nome:** DataHora

**Tipo:** DateTime

**Descrição:** Busca ou seta a data e o horário internos do concentrador.

##### 3.4.1.7 HorarioEnvioTotais

**Nome:** HorarioEnvioTotais

**Tipo:** List<SHorarioEnvioTotais>

**Descrição:** Busca ou seta a lista de horários de envio de totais.

##### 3.4.1.8 DestinatariosTecnica

**Nome:** DestinatariosTecnica

**Tipo:** List<SDestinatarioSMTP>

**Descrição:** Busca ou seta a lista de destinatários técnicos.

#### 3.4.1.9 DestinatariosPosto

**Nome:** DataHora

**Tipo:** List<SDestinatarioSMTP>

**Descrição:** Busca ou seta a lista de destinatários posto.

#### 3.4.1.10 ConfiguracaoEmail

**Nome:** ConfiguracaoEmail

**Tipo:** SConfiguracaoEmail

**Descrição:** Busca ou seta as configurações de envio de email.

#### 3.4.1.11 Combustiveis

**Nome:** Combustiveis

**Tipo:** List<SCombustivel>

**Descrição:** Busca ou seta a tabela de combustíveis.

#### 3.4.1.12 Protocolo

**Nome:** Protocolo

**Tipo:** string

**Descrição:** Busca ou seta a versão do protocolo do concentrador.

### 3.4.2 Métodos

Os métodos descrevem os comportamentos das classes que implementam a interface.

#### 3.4.2.1 string ToXml()

Retorna a representação Xml da classe.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Representação Xml da classe (string).

## 3.5 ICartao

Representa a entidade física cartão, podendo ser definido como cartão de funcionário, cartão de cliente, cartão de veículo ou cartão de motorista.

### 3.5.1 Propriedades

As propriedades descrevem as características das classes que implementam a interface.

#### 3.5.1.1 Tipo

**Nome:** Tipo

**Tipo:** ETipoCartao (ver item 2.8).

**Descrição:** Busca ou seta o tipo de cartão.

#### 3.5.1.2 Codigo

**Nome:** Codigo

**Tipo:** string

**Descrição:** Busca ou seta o código do cartão.

#### 3.5.1.2 Endereco

**Nome:** Codigo

**Tipo:** string

**Descrição:** Busca ou seta o endereço do cartão.

## 3.6 IBico

Representa um bico de abastecimento.

### 3.6.1 *Propriedades*

As propriedades descrevem as características das classes que implementam a interface.

#### 3.6.1.1 Numero

**Nome:** Numero

**Tipo:** int

**Descrição:** Busca ou seta o número do bico (valor de 1 a 100).

#### 3.6.1.2 PosicaoFisica

**Nome:** PosicaoFisica

**Tipo:** int

**Descrição:** Busca ou seta a posição física do bico na bomba (valor de 1 a 8).

#### 3.6.1.3 Canal

**Nome:** Canal.

**Tipo:** int

**Descrição:** Busca ou seta o canal de comunicação (valor de 0 a 3).

#### 3.6.1.4 Slot

**Nome:** Slot.

**Tipo:** int

**Descrição:** Busca ou seta o slot de comunicação (valor de 0 a 3).

#### 3.6.1.5 Modelo

**Nome:** Modelo

**Tipo:** char

**Descrição:** Modelo da bomba. (Ver documento Protocolo de Comunicação).

#### 3.6.1.6 Modo

**Nome:** Modo

**Tipo:** char

**Descrição:** Busca ou seta o modo de operação do bico.

- A: Automático;
- P: Por autorização;
- L: Autorização pelo PC e liberação por lado;
- B: Autorização pelo PC e liberação por bico.

#### 3.6.1.7 EnderecoLogico

**Nome:** EnderecoLogico

**Tipo:** int

**Descrição:** Busca ou seta o endereço lógico do bico.

#### 3.6.1.8 PosicaoLogica

**Nome:** PosicaoLogica

**Tipo:** int

**Descrição:** Busca ou seta a posição lógica do bico.



### 3.6.1.9 VirgulaLitro

**Nome:** VirgulaLitro

**Tipo:** int

**Descrição:** Busca ou seta a quantidade de casas decimais depois da vírgula (sempre 3).

### 3.6.1.10 VirgulaPU

**Nome:** VirgulaPU

**Tipo:** int

**Descrição:** Busca ou seta a quantidade de casas decimais depois da vírgula.

### 3.6.1.11 ValorNaBomba

**Nome:** ValorNaBomba

**Tipo:** decimal

**Descrição:** Busca ou seta o valor mostrado no display da bomba.

### 3.6.1.12 LitrosNaBomba

**Nome:** LitrosNaBomba

**Tipo:** decimal

**Descrição:** Busca ou seta a quantidade de litros mostrada no display da bomba.

### 3.6.1.13 ValorPorLitro

**Nome:** ValorPorLitro

**Tipo:** int

**Descrição:** Busca ou seta o valor cobrado por litro no bico.

### 3.6.1.14 Status

**Nome:** Status

**Tipo:** EStatusBico (Ver item 2.3).

**Descrição:** Busca ou seta o status do bico.

### 3.6.1.15 Cartao

**Nome:** Cartao

**Tipo:** Cartao

**Descrição:** Busca ou seta o cartão utilizado no abastecimento.

### 3.6.1.16 EncerranteLitro

**Nome:** EncerranteLitro

**Tipo:** string

**Descrição:** Busca ou seta o valor do totalizador por litros.

### 3.6.1.17 EncerranteValor

**Nome:** EncerranteValor

**Tipo:** string

**Descrição:** Busca ou seta o valor do totalizador monetário.

### 3.6.1.18 Template

**Nome:** Template

**Tipo:** string

**Descrição:** Busca ou seta o template utilizado no bico.

### 3.6.1.19 Combustivel

**Nome:** Combustivel

**Tipo:** SCombustivel (ver descrição).

**Descrição:** Busca ou seta o tipo de combustível utilizado no bico.

### 3.6.1.20 LeitorDeCartao

**Nome:** LeitorDeCartao

**Tipo:** LeitorDeCartao (ver descrição).

**Descrição:** Busca ou seta os dados do leitor de cartão utilizado no bico.

### 3.6.1.21 EncerranteLitroInicial

**Nome:** EncerranteLitroInicial

**Tipo:** string

**Descrição:** Busca ou seta o valor do totalizador anterior por litros, caso o método disponha do recurso.

## 3.6.2 Métodos

Os métodos descrevem o comportamento das classes que implementam a interface.

### 3.6.2.1 public void SetStatus(char status)

Seta apenas o status do bico de acordo com o parâmetro recebido.

**Parâmetros de entrada:**

1. status: Char representando o status do bico (ver ítem 2.3).\

**Retorno:** Nenhum.

### 3.6.2.2 public bool SetStatus(Comunicacao comunicacao)

Busca o status do bico no concentrador e seta todos os dados do mesmo (valor na bomba, litros na bomba, valor por litro e status).

**Parâmetros de entrada:**

1. Objeto do tipo comunicação (Ver ítem 4.5).

**Retorno:** True se a verificação do status tiver sucesso e False caso ocorra algum erro.

### 3.6.2.3 public bool SetEncerrantes(Comunicacao comunicacao)

Busca os encerrantes do bico no concentrador e seta os valores.

**Parâmetros de entrada:**

1. Objeto do tipo comunicação (Ver ítem 4.5).

**Retorno:** True se a operação for executada com sucesso e False caso ocorra algum erro.

### 3.6.2.4 public string ToXML()

Cria a representação do bico em XML.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Representação do bico em string.

## 3.7 IAbastecimento

Representa a entidade abstrata abastecimento.

### 3.7.1 Propriedades

As propriedades descrevem as características das classes que implementam a interface.

#### 3.7.1.1 Endereco

**Nome:** Endereco

**Tipo:** string

**Descrição:** Busca ou seta o endereço do abastecimento.

#### 3.7.1.2 EnderecoBytes

**Nome:** EnderecoBytes

**Tipo:** bytes[]

**Descrição:** Busca ou seta o endereço do abastecimento em um array de bytes.

#### 3.7.1.3 Status

**Nome:** Status

**Tipo:** EStatusAbastecimento (Ver item 2.2).

**Descrição:** Busca ou seta o status do abastecimento.

#### 3.7.1.4 ID

**Nome:** ID

**Tipo:** string

**Descrição:** Busca ou seta o identificador do abastecimento.

#### 3.7.1.5 Concentrador

**Nome:** Concentrador

**Tipo:** Concentrador (Ver item 4.1).

**Descrição:** Busca ou seta os dados do concentrador.

#### 3.7.1.6 Bico

**Nome:** Bico

**Tipo:** Bico (Ver item 4.3).

**Descrição:** Busca ou seta os dados do bico onde ocorreu o abastecimento.

#### 3.7.1.7 DataHora

**Nome:** DataHora

**Tipo:** DateTime

**Descrição:** Busca ou seta a data e o horário em que o abastecimento foi efetuado.

#### 3.7.1.8 ValorAbastecido

**Nome:** ValorAbastecido

**Tipo:** decimal

**Descrição:** Busca ou seta o valor abastecido.

#### 3.7.1.9 LitrosAbastecidos

**Nome:** LitrosAbastecidos

**Tipo:** decimal

**Descrição:** Busca ou seta o total de litros abastecidos.

#### 3.7.1.10 ValorPorLitro

**Nome:** ValorPorLitro

**Tipo:** decimal

**Descrição:** Busca ou seta o valor cobrado por litro.

#### 3.7.1.10 CasaDecimalLitro

**Nome:** CasaDecimalLitro

**Tipo:** int

**Descrição:** Busca a quantidade de casas decimais no total de litros.

#### 3.7.1.11 CasaDecimalValorPorLitro

**Nome:** CasaDecimalValorPorLitro

**Tipo:** int

**Descrição:** Busca a quantidade de casas decimais no valor por litro.

#### 3.7.1.12 CartaoFuncionario

**Nome:** CartaoFuncionario

**Tipo:** Cartao (Ver item 4.2).

**Descrição:** Busca ou seta os dados do cartão do funcionário.

#### 3.7.1. 13 CartaoCliente

**Nome:** CartaoFuncionario

**Tipo:** Cartao (Ver item 4.2).

**Descrição:** Busca ou seta os dados do cartão do cliente.

#### 3.7.1.14 CartaoVeiculo

**Nome:** CartaoFuncionario

**Tipo:** Cartao (Ver item 4.2).

**Descrição:** Busca ou seta os dados do cartão do veículo.

#### 3.7.1. 15 CartaoMotorista

**Nome:** CartaoFuncionario

**Tipo:** Cartao (Ver item 4.2).

**Descrição:** Busca ou seta os dados do cartão do motorista.

#### 3.7.1.16 StatusConfirmacaoLeitura

**Nome:** StatusConfirmacaoLeitura

**Tipo:** EStatusConfirmacaoLeitura

**Descrição:** Busca ou seta o status do envio e confirmação de leitura do abastecimento (Ver item 2.5 EStatusConfirmacaoLeitura).

#### 3.7.1.17 TempoDeAbastecimento

**Nome:** TempoDeAbastecimento

**Tipo:** int

**Descrição:** Se o método dispor do recurso, irá indicar o tempo em segundos que o abastecimento durou. Caso contrário, null.

### 3.8 ILeitorDeCartao

Representa a entidade abstrata leitor de cartão.

#### 3.8.1 Propriedades

As propriedades descrevem as características das classes que implementam a interface.

##### 3.8.1.1 IDCPU

**Nome:** IDCPU

**Tipo:** int

**Descrição:** Busca ou seta o identificador da CPU.

### 3.8.1.2 PortaCPU

**Nome:** PortaCPU

**Tipo:** int

**Descrição:** Busca ou seta a porta da CPU.

## 4 Classes

---

A Hiro.dll possui uma série de classes utilizadas para a comunicação e tratamento dos dados que são enviados e recebidos do concentrador.

### 4.1 Concentrador

Implementa a interface IConcentrador (Ver item 3.4).

#### 4.1.1 Métodos

Os métodos descrevem os comportamentos da classe (Ver item 3.4.2).

##### 4.1.1.1 public Concentrador()

Método construtor da classe.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Nenhum.

### 4.2 Cartao

Implementa a interface ICartao (Ver item 3.5).

#### 4.2.1 Métodos

Os métodos descrevem os comportamentos da classe (Ver item 3.5.2).

##### 4.2.1.1 public Cartao()

Método construtor da classe.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Nenhum.

##### 4.2.1.2 public Cartao(ETipoCartao tipo, string strCodigo)

Método construtor da classe.

**Parâmetros de entrada:**

1. tipo: Tipo do cartão.
2. strCodigo: Código do cartão.

**Retorno:** Nenhum.

### 4.3 Bico

Implementa a interface IBico (Ver item 3.6).

### 4.3.1 Métodos

Os métodos descrevem o comportamento da classe (Ver item 3.6.2).

#### 4.3.1.1 public Bico()

Método construtor da classe.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Nenhum.

## 4.4 Abastecimento

Implementa a interface IAbastecimento (Ver item 3.7).

### 4.4.1 Métodos

Os métodos descrevem o comportamento da classe (Ver item 3.7.2).

#### 4.4.1.1 public Abastecimento()

Método construtor da classe.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Nenhum.

## 4.5 LeitorDeCartao

Implementa a interface ILeitorDeCartao (Ver item 3.8).

### 4.5.1 Métodos

Os métodos descrevem o comportamento da classe (Ver item 3.8.2).

#### 4.5.1.1 public LeitorDeCartao()

Método construtor da classe.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Nenhum.

## 4.6 Comunicacao

Representa a entidade abstrata comunicação. Esta classe concentra todos os métodos necessários para a troca de mensagens com o concentrador. Implementa a interface IComunicacao e a interface IComunicacaoHiro. Esta classe é a única classe de comunicação que deve ser utilizada pela integração. De acordo com a configuração efetuada (Serial, USB ou LAN) ela utiliza o padrão de projeto Strategy para fazer com que a comunicação com o concentrador seja transparente, de modo que o software integrador não precise se preocupar com o tipo de comunicação. Em resumo, a aplicação integradora deve apenas se preocupar em configurar o tipo de comunicação utilizando o HiroConfig. Se a comunicação for via rede TCP/IP agora é possível definir uma porta TCP fixa, evitando um conflito de comunicação com outras aplicações.

### 4.6.1 Métodos

Os métodos descrevem o comportamento da classe.

#### 4.6.1.1 public Comunicacao()

Método construtor da classe.

**Parâmetros de entrada:** Nenhum.

**Retorno:** Nenhum.

## 4.7 ComunicacaoSerial

Implementa a interface IComunicacao. É utilizada quando o tipo de comunicação utilizada for serial.

## 4.8 ComunicacaoUSB

Implementa a interface IComunicacao. É utilizada quando o tipo de comunicação utilizada for USB.

## 4.9 ComunicacaoTCPIP

Implementa a interface IComunicacao. É utilizada quando o tipo de comunicação utilizada for TCP/IP.

## 4.10 ComunicacaoDelphi

Implementa a interface IComunicacaoDelphi. É utilizada como modelo para integração em Delphi. As aplicações que desejarem podem utilizar a classe ComunicacaoDelphi ao invés da Comunicação.

## 5 Estruturas de Dados

Estruturas de dados utilizadas na comunicação com o concentrador.

### 5.1 *abast*

Estrutura de dados para armazenamento dos dados de um abastecimento.

Nome do Campo	Tipo	Descrição
value	bool	Indica se o abastecimento foi lido.
total_dinheiro	decimal	Valor do abastecimento.
total_litros	double	Litros abastecidos.
PU	decimal	Preço unitário.
tempo	string	Duração do abastecimento.
canal	string	Canal do bico.
data	string	Data do abastecimento.
hora	string	Horário do abastecimento.
st_full	string	
registro	int	
encerrante	double	Encerrante litros.
integridade	bool	
checksum	bool	

### 5.2 *Abast2*

Estrutura de dados para armazenamento dos dados de um abastecimento.

Nome do Campo	Tipo	Descrição
value	string	Indica se o abastecimento foi lido.
total_dinheiro	string	Valor do abastecimento.
total_litros	string	Litros abastecidos.
PU	string	Preço unitário.
tempo	string	Duração do abastecimento.
canal	string	Canal do bico.
data	string	Data do abastecimento.
hora	string	Horário do abastecimento.
st_full	string	
registro	string	
encerrante	string	Encerrante litros.
integridade	string	
checksum	string	

### 5.3 *Abast3*

Estrutura de dados para armazenamento dos dados de um abastecimento.

Nome do Campo	Tipo	Descrição
value	string	Indica se o abastecimento foi lido.
total_dinheiro	string	Valor do abastecimento.
total_litros	string	Litros abastecidos.
PU	string	Preço unitário.
tempo	string	Duração do abastecimento.
canal	string	Canal do bico.
data	string	Data do abastecimento.



hora	string	Horário do abastecimento.
st_full	string	
registro	string	
encerrante	string	Encerrante litros.
id	string	
integridade	string	
checksum	string	

## 5.4 AbastFid

Estrutura de dados para armazenamento dos dados de um abastecimento.

Nome do Campo	Tipo	Descrição
value	bool	Indica se o abastecimento foi lido.
total_dinheiro	decimal	Valor do abastecimento.
total_litros	double	Litros abastecidos.
PU	decimal	Preço unitário.
tempo	string	Duração do abastecimento.
canal	string	Canal do bico.
data	string	Data do abastecimento.
hora	string	Horário do abastecimento.
st_full	string	
registro	int	
encerrante	double	Encerrante litros.
integridade	bool	
checksum	bool	
tag	string	

## 5.5 AbastVB

Estrutura de dados para armazenamento dos dados de um abastecimento.

Nome do Campo	Tipo	Descrição
value	bool	Indica se o abastecimento foi lido.
total_dinheiro	decimal	Valor do abastecimento.
total_litros	double	Litros abastecidos.
PU	decimal	Preço unitário.
tempo	string	Duração do abastecimento.
canal	string	Canal do bico.
data	string	Data do abastecimento.
hora	string	Horário do abastecimento.
st_full	string	
registro	int	
encerrante	double	Encerrante litros.
Integridade	bool	
checksum	bool	

## 5.6 canal

Estrutura de dados que armazena dados de todos os canais.

Nome do Campo	Tipo	Descrição
canal	byte[]	Array de bytes (lista de canais)
PuAux	Double[]	

## 5.7 Enc

Estrutura para registro de dados de encerrantes.

Nome do Campo	Tipo	Descrição
Bico	string	Número do bico.
tipo	string	"\$" Encerrante valor. "L" Encerrante litros.
valor	string	Valor do encerrante.

## 5.8 Encerrante

Estrutura para registro de dados de encerrantes.

Nome do Campo	Tipo	Descrição
Bico	string	Número do bico.
valor	double	Valor do encerrante litros.

## 5.9 MultiStatus

Estrutura para registro do status de todos os bicos.

Nome do Campo	Tipo	Descrição
Status	StOptions[]	Array indicando o status de todos os bicos.

## 5.10 OnLine

Estrutura para registro dos abastecimentos em andamento.

Nome do Campo	Tipo	Descrição
Litragem	Decimal[]	Array com o valor na bomba.
Bico	String[]	Array com o número dos bicos.

## 5.11 SCombustivel

Estrutura de dados que descreve um tipo de combustível de acordo com o mapa.

Nome do Campo	Tipo	Descrição
Codigo	int	Código do combustível.
Descricao	string	Descrição do combustível.

## 5.12 Retorno

Estrutura para registro de abastecimento.

Formato: "TTTTTLLLLLLPPPPVCCCCBBDDHMMNNRRRREEEEEEEEEESSKK"

Nome do Campo	Tipo	Descrição
value	string	String com o abastecimento.

## 5.13 Retorno2

Estrutura para registro de abastecimento.

Formato: "TTTTTLLLLLLPPPPVCCCCBBDDHMMKK"

Nome do Campo	Tipo	Descrição
value	string	String com o abastecimento.

### 5.14 SConfiguracaoEmail

Estrutura que armazena os dados de envio e recebimento de emails.

Nome do Campo	Tipo	Descrição
Endereco	string	Endereço do servidor de email.
Porta	long	Porta de envio de email.
Usuario	string	Email de usuário.
Senha	string	Senha.
Autenticacao	bool	“True” se utiliza. “False” se não.
IdentificacaoPosto	string	Identificação do posto.

### 5.15 SConfiguracaoTCPIP

Estrutura que armazena os dados da configuração de comunicação com o concentrador via LAN.

Nome do Campo	Tipo	Descrição
EnderecoIP	string	Endereço do concentrador.
Porta	string	Porta.
Gateway	string	Gateway padrão.
DNS	string	Endereço de DNS.
Mascara	string	Mascara de sub rede.
DHCP	bool	“True” se permite. “False” se não.

### 5.16 SDestinatarioSMTP

Estrutura que armazena os dados dos destinatários de email.

Nome do Campo	Tipo	Descrição
Codigo	int	Código do destinatário (1 a 4).
Tipo	cahr	‘P’ Posto. ‘T’ Técnica.
Email	string	Email.

### 5.17 SHorarioEnvioTotais

Armazena os dados de um horário de envio de totais.

Nome do Campo	Tipo	Descrição
Codigo	int	Código do destinatário (1 a 4).
Hora	char	Hora do envio.
Minuto	string	Minuto do envio.

### 5.18 SEncerrante

Estrutura que descreve um encerrante.

Nome do Campo	Tipo	Descrição
Bico	string	Código do bico.
Encerrante	string	Valor do encerrante.

### 5.19 stPPL

Estrutura que armazena o preço por litro de um bico.

Nome do Campo	Tipo	Descrição
Bico	string	Código do bico.
PPL	string	Preço por litro.

### 5.20 StStatus

Estrutura com o status de todos os bicos.

Tabela de status:

‘L’ - Bico encontra-se livre para abastecer;

‘B’ - Bico bloqueado para realizar abastecimentos;

‘A’ - Bico está em processo de abastecimento;

‘E’ - Bico está aguardando liberação da automação para iniciar o processo de abastecimento;

‘F’ - Bico não presente ou em falha.

Nome do Campo	Tipo	Descrição
Value	string	String com o status de todos os bicos.

### 5.21 StStatus2

Estrutura com o status de todos os bicos. O Status é descrito conforme a estrutura de dados StOptions.

Nome do Campo	Tipo	Descrição
Value	String[]	Array de strings com o status dos bicos.

### 5.22 VBOnline

Estrutura que registra o valor mostrado na bomba em todos os bicos no momento da consulta.

Nome do Campo	Tipo	Descrição
Bico	String[]	Array com o código dos bicos.
Volume	String[]	Array com o valor abastecido no momento da consulta.

### 5.23 visualizacao

Estrutura que registra o valor que todos os bicos estão abastecendo no momento da consulta.

Formato: "BBVVVVVVBBVVVVVV..." onde "BB" é o número do bico e "VVVVVV" é o valor na bomba na hora da leitura.

Nome do Campo	Tipo	Descrição
stFull	string	String com todos os abastecimentos.

### 5.24 SMedidor

Estrutura de dados para armazenamento dos dados de um medidor de tanque de combustíveis.

Nome do Campo	Tipo	Descrição
NumeroTanque	Int	Indica o número do tanque/medidor.
TotalLitros	Int	Tamanho do tanque em litros
CombustivelLitros	Int	Volume de combustível no tanque em litros
CombustivelMilimetros	Int	Volume de combustível no tanque em milímetros
AguaLitros	Int	Volume de água no tanque em litros
AguaMilimetros	Int	Altura da água no tanque em milímetros
EspacoLivre	Int	
CombustivelTemperatura	double	Temperatura do combustível em graus celsius (1 casa decimal)
ErroLeitura	bool	Indica que não foi possível ler este tanque

## 6 Arquitetura da Comunicação

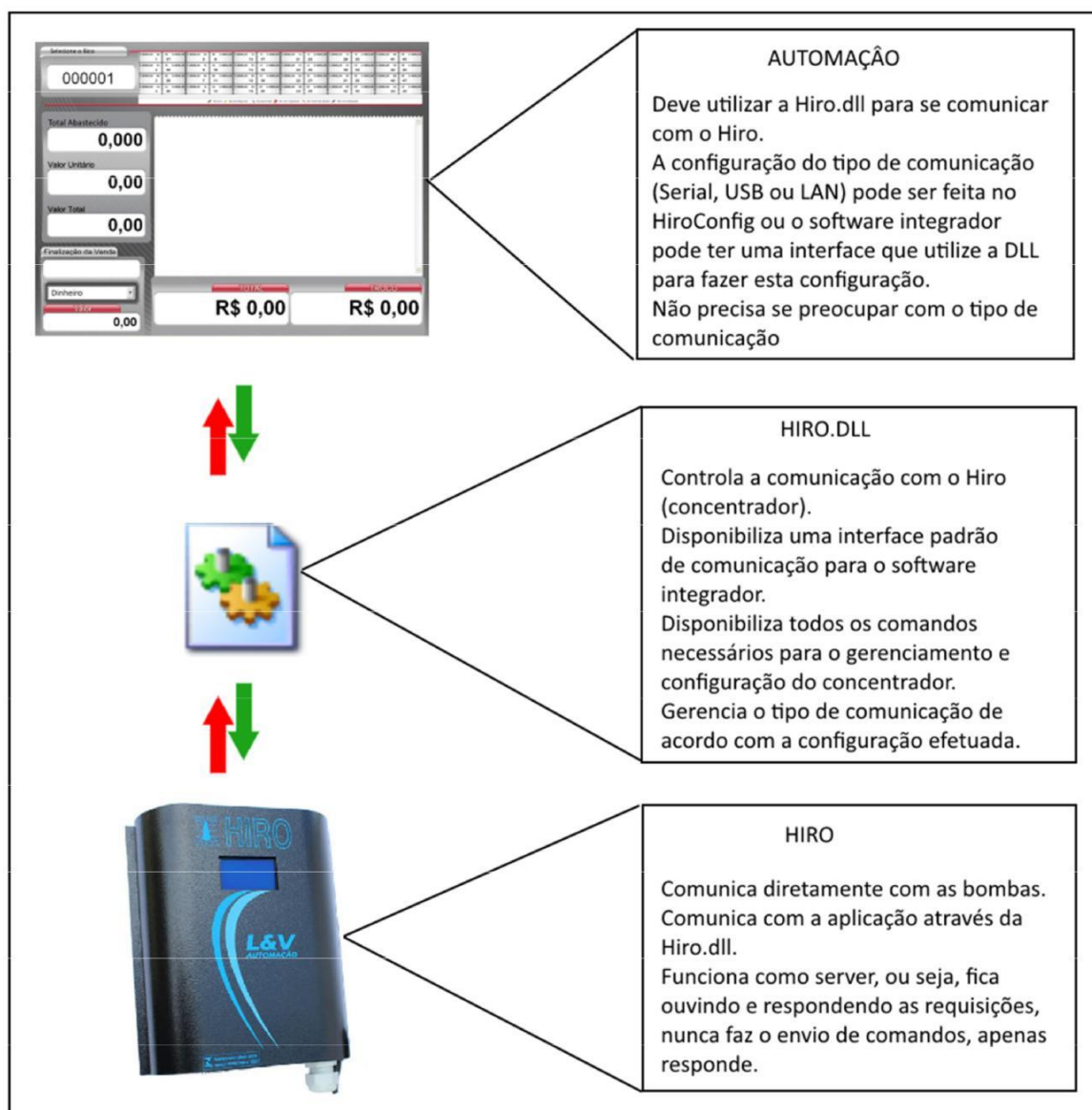


Figura 1 Estrutura da Comunicação

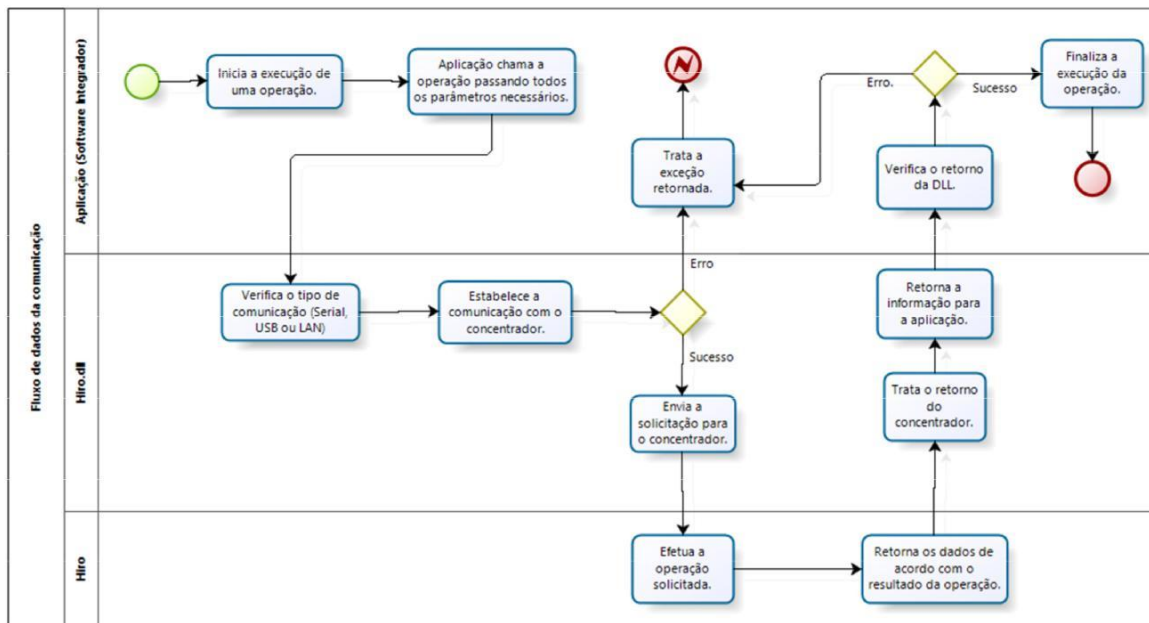


Figura 2: Fluxo de Dados da Comunicação.