

AldeiaNet API Dynamic Link Library

Definições

AldNetBC.DLL

Versão deste documento: 2.3 – 30/11/2001

© NOS Informática Consultoria, Treinamento e Representações Ltda

ÍNDICE

UTILIZANDO A DLL DE API DO ALDEIANET.....	4
DEFINIÇÕES DAS ROTINAS DA API.....	4
• INICIA SESSÃO COM A API ALDEIANET, PARA INÍCIO DAS OPERAÇÕES	4
<i>function F_ABRESESSAO</i>	4
• ENVIA ARQUIVO AO SISTEMA.....	4
<i>function F_ADICIONAARQUIVO</i>	4
• CONSULTA ARQUIVOS DISPONÍVEIS PARA RECEPÇÃO.....	5
<i>function F_CONSULTAARQUIVOS</i>	5
• RECEBE ARQUIVO DO SISTEMA	7
<i>function F_OBTEMARQUIVO</i>	7
• REMOVE ARQUIVO DO SISTEMA	8
<i>function F_REMOVEARQUIVO</i>	8
• LIBERA ARQUIVO SEM RECEBER (ALTERA STATUS PARA USUÁRIO LOGADO).....	8
<i>function F_LIBERAARQUIVO</i>	8
• CONSULTA RECIBOS RELACIONADOS A UM ARQUIVOS, DISPONÍVEIS PARA RECEPÇÃO	8
<i>function F_CONSULTARECIBOS</i>	8
• CARGA DE ARQUIVO MODELO PARA RECIBO (METAFILE)	9
<i>function F_METAFILELERMODELO</i>	9
• SUBSTITUIÇÃO DE MACRO NO ARQUIVO MODELO (METAFILE) CARREGADO PREVIAMENTE	10
<i>function F_METAFILESUBSTITUIMACRO</i>	10
• GRAVA O ARQUIVO DE RECIBO BASEADO NO ARQUIVO MODELO CARREGADO PREVIAMENTE, APÓS TEREM SIDO FEITAS AS SUBSTITUIÇÕES DE MACROS	10
<i>function F_METAFILESALVAARQUIVO</i>	10
• GERAÇÃO DE RECIBO APÓS VALIDAÇÃO ONLINE DE ARQUIVOS RECEBIDOS PELO SISTEMA	10
<i>function F_GERARECIBOONLINE</i>	10
• FINALIZAR SESSÃO COM A API ALDEIANET, CONCLUINDO OPERAÇÕES	12
<i>function F_FECHASESSAO : Longint; StdCall;</i>	12
• CARREGAR A CONFIGURAÇÃO A PARTIR DE UM ARQUIVO DE CONFIGURAÇÃO.....	12
<i>function F_CARREGACONFIGURACAO (v_NomeArquivoCfg : Pchar) : Integer; StdCall;</i>	12
ESTRUTURAS E CONSTANTES UTILIZADAS	12
<i>Constantes que podem ser utilizadas pelos aplicativos usuários da API AldeiaNet:</i>	12
FlagData para consulta de arquivos.....	12
Status de recebimento para consulta de arquivos	13
Categorias de recibos	13
Tamanhos de cadeias.....	13
Códigos de erros.....	13
Macros para geração de recibos.....	16
<i>Tipos e estruturas que podem ser utilizadas pelos aplicativos usuários da API AldeiaNet:</i>	16
Estruturas de cadeias para passagens de parâmetros	16
Estruturas passadas nas chamadas da API.....	17
CONVERSÕES DE BORLAND DELPHI PARA MICROSOFT VISUAL BASIC.....	19
• RELAÇÃO ENTRE TIPOS E CONSTANTES	19
Tipos simples	19
Constantes de tamanho de cadeias.....	19
Estruturas de cadeias	20
Estruturas do tipo <i>record</i> :.....	21
Parâmetros passados como Pchar (Ponteiro para Cadeia de Caracteres):.....	22
• DECLARAÇÕES DA API PARA VISUAL BASIC	22
• CONSTANTES, TIPOS E ESTRUTURAS UTILIZADAS PELA API, PARA VISUAL BASIC	23

<i>Constantes que podem ser utilizadas pelos aplicativos usuários da API AldeiaNet (Obs. Para constantes globais, substituir const por Global Const):</i>	23
FlagData para consulta de arquivos.....	23
Status de recebimento para consulta de arquivos	23
Categorias de recibos	23
Tamanhos de cadeias.....	24
Códigos de erros.....	24
Macros para geração de recibos.....	26
Estruturas passadas nas chamadas da API.....	27

Utilizando a DLL de API do AldeiaNet

Importante: ao utilizar esta biblioteca, somente uma sessão com o servidor AldeiaNet pode ser aberta por vez no escopo de UMA APLICAÇÃO, haja vista nesta versão a implementação da mesma possuir área pública comum a todas as chamadas.

Para o caso de ser necessário abrir mais de uma sessão com o servidor, recomenda-se a utilização da API disponível no formato de ActiveX (AldeiaNetActiveXControl.OCX), que permite várias instâncias do objeto na mesma aplicação. Consultar a documentação desse ActiveX para mais informações.

Definições das rotinas da API

Todos os tipos de dados descritos a seguir estão definidos na seção “*Constantes, tipos e estruturas utilizadas pela API*” presente neste documento. Todos os cabeçalhos de funções, definição de estruturas e dados utilizados aqui, são ilustrados na linguagem **Borland Delphi**.

Mais adiante, neste documento, há explicações sobre como fazer as conversões para *Microsoft Visual Basic*.

Obs.: Para não utilizar algum dos campos de alguma estrutura, inicialize-o com zeros binários (Nulos).

• Inicia sessão com a API AldeiaNet, para início das operações

Deve ser a primeira rotina a ser utilizada, antes de qualquer outra.

```
function F_ABRESESSAO  
(v_Contexto,v_Usuario,v_Senha:pchar) : Longint; StdCall;
```

Parâmetro:

Aqui devem ser passados:

- Contexto do sistema;
- Código do Usuário;
- Senha do Usuário.

Retorno: Se função executou com sucesso, o retorno deve ser zero. Caso contrário, retornará o código do erro ocorrido.

• Envia arquivo ao sistema

```
function F_ADICIONAARQUIVO  
(var v_RegIncluirArquivoExt:t_RegIncluirArquivoExt) : Longint; StdCall;
```

Parâmetro: v_RegIncluirArquivoExt:

```
t_RegIncluirArquivoExt = record  
    v_NomeArquivo      : t_CadeiaNomeArquivo;  
    v_Descricao        : t_CadeiaDescricao;  
    v_UsrOrigem        : t_CadeiaCodUsuario;  
    v_UsrDestino       : t_CadeiaCodUsuario;  
    v_DirDnload        : t_CadeiaPathArquivo;  
    v_ForcaDirDnload   : WordBool;
```

```

v_ConfirmaRecepcao: WordBool;
v_DataEntrada      : t_CadeiaDataHora; //saída
v_NumControle      : Longint; {saída}
v_IsnArquivo       : Longint; {saída}
v_Reservado        : array[0..255] of char;
end;

```

Aqui devem ser passados:

- Nome do arquivo a ser enviado ao sistema;
- Descrição ou assunto sobre o arquivo.;
- Usuário remetente (origem) do arquivo;
- Usuário destinatário do arquivo;
- Diretório sugerido para recepção pelo usuário destinatário;
- Opção de diretório de saída obrigatório no destinatário;
- Opção de confirmação de recepção pelo destinatário;

Nesta mesma estrutura serão retornados:

- Data e horário de entrada do arquivo no sistema;
- Número de controle do arquivo no sistema;
- Chave (ISN) do arquivo no sistema.

Retorno: Se função executou com sucesso, o retorno deve ser zero. Caso contrário, retornará o código do erro ocorrido.

• Consulta arquivos disponíveis para recepção

```

function F_CONSULTAARQUIVOS
(var v_RegConsultaArquivosExt:t_RegConsultaArquivosExt;
 var v_RegInfoArqExt:t_RegInfoArqExt) : Longint;stdcall;

```

Parâmetro de entrada: v_RegConsultaArquivosExt:

```

T_RegConsultaArquivosExt = record
  v_UsrDestino      : t_CadeiaCodUsuario;
  v_UsrOrigem       : t_CadeiaCodUsuario;
  v_RefNomeArquivo  : t_CadeiaNomeArquivo
  v_StatusDoDestino: Longint;
  v_StatusDoLogon   : Longint;
  v_FlagData        : Longint;
  v_DataEntrada     : t_CadeiaDataHora;
  v_RegInicio       : Longint;
  v_PontConsulta    : Longint;
  v_HabilitarConsultaHorario:WordBool;
  v_NumControle     : LongInt;
  v_ConsultaExpurgados:WordBool;
  v_TipoExclusao    : Longint;
  v_OrigemExclusao  : Array[0..c_TamCodusuario] of char;
  v_DataExclusao    : Array[0..c_TamData] of char;
  v_FlagDataExclusao:Longint;
  v_HabConsHorExclusao:WordBool;
  v_Reservado       : array[1..128] of byte;
end;

```

Aqui devem ser passados os dados para filtrar a consulta:

- Usuário destinatário;
- Usuário remetente;
- Referência a nome de arquivo;
- Situação do arquivo perante o destinatário original;
- Situação do arquivo perante o usuário que abriu a sessão;
- Opção de consulta de data;
- Data de entrada;
- se habilita o horário de entrada na consulta
- consultar por número de controle
- ConsultaExpurgados: se quiser consultar somente arquivo excluídos logicamente
- TipoExclusao: usado somente se consulta de expurgados
- OrigemExclusao: usado somente se consulta de expurgados
- DataExclusao: usado somente se consulta de expurgados
- FlagDataExclusao: usado somente se consulta de expurgados
- HabConsHorExclusao: usado somente se consulta de expurgados

As informações acima podem ser algumas ou todas omitidas, para assumir-se não filtragem na consulta. Nessa mesma estrutura serão passados ainda:

- Registro de reinício de consulta. Consulta inicial é igual a zeros. A partir da consulta subsequente, utilizar o campo retornado na consulta anterior;
- Identificados da consulta (ponteiro). Inicial é igual a zeros. A partir da consulta subsequente, utilizar o campo retornado na consulta anterior.

Parâmetro de saída: v_RegInfoArqExt:

```
T_RegInfoArqExt = packed record
    v_IsnArquivo      : Longint;
    v_NumControle     : Longint;
    v_UsuarioOrigem   : t_CadeiaCodUsuario;
    v_UsuarioDestino  : t_CadeiaCodUsuario;
    v_DataEntrada     : t_CadeiaDataHora;
    v_DataRecupDono    : t_CadeiaDataHora;
    v_DataRecupLogon   : t_CadeiaDataHora;
    v_StatusDono       : Longint;
    v_StatusLogon      : Longint;
    v_NomeArquivo     : t_CadeiaNomeArquivo;
    v_Tamanho         : Longint;
    v_DataArquivo     : t_CadeiaDataHora;
    v_Descricao       : t_CadeiaDescricao;
    v_TamCompactado   : Longint;
    v_TemProtocolo    : WordBool;
    v_TipoCompac      : Longint; //0=AldeiaNet
    v_TipoCript       : Longint; //0=Nenhuma;1=AldeiaNet;2=CEPESC
    v_RegProximo      : Longint;
    v_Continua        : WordBool;
    v_PontConsulta    : Longint;
    v_VersaoMaior     : Word;
    v_VersaoMenor     : Word;
    v_VersaoRelease   : Word;
    v_VersaoBuild     : Word;
    v_Operador        : array[0..c_TamCodUsuario] of char;
    v_Estacao         : array[0..15] of char;
    v_Reservado       : array[c_TamCodUsuario+1+16+9..255] of char;
end;
```

Aqui serão retornados informações sobre cada arquivo disponível no sistema que obdecer aos filtros da consulta:

- Chave do arquivo (ISN);
- Número de controle do arquivo;
- Usuário remetente;
- Usuário destinatário;
- Data de entrada do arquivo no sistema;
- Data da recuperação desse arquivo pelo seu destinatário, se já recebido anteriormente;
- Data da recuperação desse arquivo pelo usuário que abriu a sessão, se já recebido anteriormente;
- Situação do arquivo perante o destinatário original;
- Situação do arquivo perante o usuário que abriu a sessão;
- Nome original do arquivo;
- Tamanho do arquivo em bytes;
- Data da criação/modificação do arquivo;
- Descrição ou assunto do arquivo;
- Tamanho do arquivo após a sua compactação pelo AldeiaNet;
- Se há protocolo de envio associado ao arquivo;
- Tipo da compactação utilizada neste arquivo, pelo AldeiaNet;
- Tipo de criptografia utilizada neste arquivo, pelo AldeiaNet;
- Número do próximo registro da consulta. Esta informação deve ser utilizada na consulta subsequente, caso este não seja o último arquivo;
- Se há mais arquivos para esta consulta;
- Identificador da consulta (ponteiro). Inicial é igual a zeros. Nas consultas subseqüentes, utilizar o campo para obter as informações dos demais arquivos que obedeceram ao filtro informado.
- VersaoMaior: código de versão para arquivos executáveis ou bibliotecas;
- VersaoMenor: código de versão para arquivos executáveis ou bibliotecas;
- VersaoRelease: código de versão para arquivos executáveis ou bibliotecas;
- VersaoBuild: código de versão para arquivos executáveis ou bibliotecas;
- Operador: operador ou login alternativo que enviou o arquivo;
- Estacao: estação de onde o arquivo foi enviado;

Retorno: Se função executou com sucesso, o retorno deve ser zero. Caso contrário, retornará o código do erro ocorrido.

• Recebe arquivo do sistema

```
function F_OBTENARQUIVO  
(var v_RegObterArquivoExt:t_RegObterArquivoExt) : Longint; StdCall;
```

Parâmetros: Entrada: v_RegObterArquivoExt:

```
t_RegObterArquivoExt = record  
    v_IsnArquivo          : Longint;  
    v_DiretorioDestino     : t_CadeiaPathArquivo;  
    v_CaminhoCompletoSaida : t_CadeiaPathArquivo;//out  
    v_ChaveDecifragem      : t_CadeiaChaveDecifragem;  
end;
```

Aqui devem ser passados os dados para obtenção do arquivo desejado:

- Chave (ISN) do arquivo a ser recebido;
- Diretório de destino, aonde o arquivo será gravado, após a recepção;
- Chave para decifragem do arquivo (não utilizado por enquanto);

Será retornado nessa mesma estrutura:

- Caminho completo (path) de saída, após a recepção (concatenação do diretório de destino com o nome original do arquivo).

Retorno: Se função executou com sucesso, o retorno deve ser zero. Caso contrário, retornará o código do erro ocorrido.

- **Remove arquivo do sistema**

```
function F_REMOVEARQUIVO  
(var v_RegRemoverArquivoExt:t_RegRemoverArquivoExt) : Longint; StdCall;
```

Parâmetro: v_RegRemoverArquivoExt:

```
t_RegRemoverArquivoExt = packed record  
    v_IsnArquivo          : Longint;  
end;
```

Aqui deve ser passado a seguinte informação para localização do arquivo a ser removido:

- Chave (ISN) do arquivo;

Retorno: Se função executou com sucesso, o retorno deve ser zero. Caso contrário, retornará o código do erro ocorrido.

- **Libera arquivo sem receber (altera status para usuário logado)**

```
function F_LIBERAARQUIVO  
(var v_RegRemoverArquivoExt:t_RegRemoverArquivoExt) : Longint; StdCall;
```

Parâmetro: v_RegRemoverArquivoExt:

```
t_RegRemoverArquivoExt = packed record  
    v_IsnArquivo          : Longint;  
end;
```

Aqui deve ser passado a seguinte informação para localização do arquivo a ser liberado:

- Chave (ISN) do arquivo;

Retorno: Se função executou com sucesso, o retorno deve ser zero. Caso contrário, retornará o código do erro ocorrido.

- **Consulta recibos relacionados a um arquivos, disponíveis para recepção**

```
function F_CONSULTARECIBOS  
(var v_RegPedeRecibosExt:t_RegPedeRecibosExt;  
 var v_RegReciboRelacionadoExt:t_RegReciboRelacionadoExt;  
 var v_RegProximo:Longint; var v_Continua:WordBool;  
 var v_PtQuery:Longint):Longint;stdcall;
```

Parâmetro de entrada: v_RegPedeRecibosExt :

```
t_RegPedeRecibosExt = record
```

```

    v_IsnArquivo : Longint;
    v_RegInicio  : Longint;
    v_PontConsulta : Longint;
end;
```

Aqui devem ser passados os dados para filtrar a consulta:

- Isn Arquivo = Chave do arquivo do qual se deseja os recibos;

Nessa mesma estrutura serão passados ainda:

- Registro de reinício de consulta. Consulta inicial é igual a zeros. A partir da consulta subsequente, utilizar o campo retornado na consulta anterior;
- Identificados da consulta (ponteiro). Inicial é igual a zeros. A partir da consulta subsequente, utilizar o campo retornado na consulta anterior.

Parâmetro de saída: v_RegReciboRelacionado:

```

T_RegReciboRelacionadoExt = record
    v_IsnArquivo      : Longint;
    v_DataEntrada     : array[0..c_TamData] of char;
    v_NomeArquivo     : array[0..c_TamNomeArquivo] of char;
    v_Tamanho         : Longint;
    v_DataArquivo     : array[0..c_TamData] of char;
    v_CategoriaRecibo : Longint;
end;
```

Aqui serão retornados informações sobre cada recibo disponível no sistema relacionado ao arquivo informado:

- Chave do recibo (ISN);
- Data de entrada do recibo (geração) no sistema;
- Nome do arquivo de recibo;
- Tamanho do recibo em bytes;
- Data da criação/modificação do recibo;
- Categoria do Recibo: 1=Recibo de entrada/entrega; 2=Recibo de confirmação de recebimento.

Outros parâmetros retornados

- Número do próximo registro da consulta. Esta informação deve ser utilizada na consulta subsequente, caso este não seja o último arquivo;
- Se há mais arquivos para esta consulta;
- Identificador da consulta (ponteiro). Inicial é igual a zeros. Nas consultas subsequentes, utilizar o campo para obter as informações dos demais arquivos que obedeceram ao filtro informado.

Obs.: Desejando-se receber o recibo em questão, utilizar a função *F_OBTERRARQUIVO*, passando-se o *v_IsnArquivo* deste recibo

Retorno: Se função executou com sucesso, o retorno deve ser zero. Caso contrário, retornará o código do erro ocorrido.

• Carga de arquivo modelo para recibo (metafile)

```

function F_METAFILELERMODELO
(v_NomeModelo:pchar):Integer;
```

Parâmetro: v_NomeModelo.

- Caminho completo para o arquivo contendo o modelo para o recibo a ser gerado.

Esse arquivo será carregado pela rotinas para posteriores operações de substituição de macros e geração de um arquivo novo com o recibo desejado. O arquivo poderá ter qualquer formato, e será tratado pela rotina como um arquivo binário.

Retorno: Se função executou com sucesso, o retorno deve ser zero. Caso contrário, retornará o código do erro ocorrido.

- **Substituição de macro no arquivo modelo (metafile) carregado previamente**

```
function F_METAFILESUBSTITUIMACRO  
(v_Macro, v_Substituto:pchar) : Integer;
```

Parâmetros: v_Macro e v_Substituto.

A macro (cadeia de caracteres) será procurada no metafile modelo, e todas as ocorrências encontradas serão substituídas pelo parâmetro *v_substituto*.

Retorno: Se função executou com sucesso, o retorno deve ser zero. Caso contrário, retornará o código do erro ocorrido.

- **Grava o arquivo de recibo baseado no arquivo modelo carregado previamente, após terem sido feitas as substituições de macros**

```
function F_METAFILESALVAARQUIVO  
(v_NomeArquivo:pchar) : Integer;
```

Parâmetro: v_NomeArquivo.

- Caminho completo para o novo arquivo a ser gravado.

Este procedimento apenas gera o novo arquivo baseado no modelo utilizado. Para incluir o arquivo no sistema (enviar), se for o caso, deve-se, após esta gravação citada aqui, utilizar a rotina *F_ADICIONAARQUIVO*.

Retorno: Se função executou com sucesso, o retorno deve ser zero. Caso contrário, retornará o código do erro ocorrido.

- **Geração de recibo após validação online de arquivos recebidos pelo sistema**

Geração em de recibos após o recebimento de arquivos pelo servidor AldeiaNet. Esta rotina deverá ser utilizada por aplicativo do usuário (DLL ou Classe) extensivo do servidor AldeiaNet, pois a estrutura aqui passada como parâmetro (*T_ReciboExterno*) será obtida pela chamada "Call Back" do servidor a esse aplicativo.

```
function F_GERARECIBOONLINE  
(var v_ReciboExterno:T_ReciboExterno;  
 v_ArquivoModelo:pchar) : Longint; StdCall;
```

Parâmetro: v_ReciboExterno e v_ArquivoModelo:

```
t_ReciboExterno = packed record
  v_Titulo                : array[0..c_tamTitulo] of char;
  v_NumControle           : Longint;
  v_Contexto              : t_CadeiaContexto;
  v_DataEntrada           : t_CadeiaDataHora;
  v_CodUsuarioRemetente   : t_CadeiaCodUsuario;
  v_NomeUsuarioRemetente  : t_CadeiaUsuarioExtenso;
  v_CodUsuarioDestinatario : t_CadeiaCodUsuario;
  v_NomeUsuarioDestinatario : t_CadeiaUsuarioExtenso;
  v_NomeArquivo           : t_CadeiaNomeArquivo;
  v_Descricao             : t_CadeiaDescricao;
  v_TamanhoArquivo        : Longint;
  v_DataArquivo           : t_CadeiaDataHora;
  v_GeracaoRecibo         : Longint;
  v_CodUsrRecebedor       : t_CadeiaCodUsuario;
  v_NomeUsrRecebedor      : t_CadeiaUsuarioExtenso;
  v_DataRecepcao          : t_CadeiaDataHora;
  v_VersaoServidor        : t_CadeiaVersao;
  v_VersaoCliente         : t_CadeiaVersao;
  v_NomeServidor          : t_CadeiaNomeMaquina;
  v_NomeEstacao           : t_CadeiaNomeMaquina;
  v_IdConexao             : Longint;
  v_UsrSessaoValidacao    : array[0..c_TamCodUsuario] of char;
  v_SenhaUsrSessaoValidacao : array[0..c_TamSenha] of char;
  v_CodOperador           : array[0..c_TamCodUsuario] of char;
  //--
  v_Reservado             : array[63..128] of char;
end;
```

Descrição dos campos desta estrutura:

- Título configurado para este recibo;
- Número de controle do arquivo;
- Contexto do arquivo recebido;
- Data de entrada do arquivo no sistema;
- Código do usuário remetente;
- Nome extenso do usuário remetente;
- Código do usuário destinatário;
- Nome extenso do usuário destinatário;
- Nome original do arquivo recebido;
- Descrição ou assunto sobre o arquivo recebido;
- Tamanho do arquivo recebido;
- Data de criação do arquivo;
- Tipo de geração do recibo;
- Código do usuário recebedor;
- Nome extenso do usuário recebedor;
- Data e horário da recepção;
- Versão do módulo servidor que recebeu o arquivo;
- Versão do módulo cliente utilizado;
- Nome da máquina utilizando o módulo servidor que recebeu o arquivo;
- Nome da máquina utilizando o módulo cliente;
- Identificação da conexão.

Além da estrutura citada acima, também deverá ser passada para esta rotina o campo `v_ArquivoModelo`, que deverá apontar para o nome do arquivo modelo de recibo (metafile) a ser utilizado para a geração deste recibo em questão.

Retorno: Se função executou com sucesso, o retorno deve ser zero. Caso contrário, retornará o código do erro ocorrido.

- **Finalizar sessão com a API AldeiaNet, concluindo operações**

Deve ser a última rotina a ser utilizada, antes de encerrar o aplicativo do usuário.

```
function F_FECHASESSAO : Longint; StdCall;
```

Retorno: Se função executou com sucesso, o retorno deve ser zero. Caso contrário, retornará o código do erro ocorrido.

- **Carregar a configuração a partir de um arquivo de configuração**

Certos itens da configuração só terá efeito a partir da próxima abertura de sessão, como por exemplo, os parâmetros para a conexão.

```
function F_CARREGACONFIGURACAO (v_NomeArquivoCfg : Pchar ) : Integer;  
StdCall;
```

Retorno: Se função executou com sucesso, o retorno deve ser zero. Caso contrário, retornará o código do erro ocorrido.

Caso não seja utilizada esta função, a DLL utilizará a configuração do arquivo `AldNetCl.INI` encontrado no path do executável da aplicação. Caso não encontre, utilizará a configuração padrão.

Obs.: No caso de Visual Basic, caso se esteja utilizando no modo interpretado, o path da aplicação equivale-se ao path aonde o VB está instalado.

Estruturas e Constantes utilizadas

Constantes que podem ser utilizadas pelos aplicativos usuários da API AldeiaNet:

FlagData para consulta de arquivos

```
Igual          = 0;  
Menor          = 1;  
MenorIgual     = 2;  
MaiorIgual     = 3;  
Maior          = 4;  
Diferente      = 5;  
Qualquer       = 6;
```

Status de recebimento para consulta de arquivos

```
Todos          = 0;  
JaRecebidos    = 1;  
NaoRecebidos   = 2;
```

Categorias de recibos

```
crNenhum = 0;  
crReciboEntrada = 1;  
crReciboConfirmacao = 2;
```

Tamanhos de cadeias

```
c_TamTitulo          = 100;  
c_TamNumControle     = 10;  
c_TamContexto        = 15;  
c_TamData            = 17;  
c_TamCodUsuario      = 25;  
c_TamExtensoUsuario  = 80;  
c_TamNomeArquivo     = 120;  
c_TamPathArquivo     = 255;  
c_TamDescricao       = 200;  
c_TamLongint         = 10;  
c_TamSenha           = 10;  
c_TamVersao          = 10;  
c_TamNomeMaquina     = 15;  
c_TamSenhaCriptogr   = 21;  
c_TamEndereco        = 50;  
c_TamNumEndereco     = 5;  
c_TamComplEndereco   = 45;  
c_TamBairro          = 50;  
c_TamCep             = 10;  
c_TamTelefone        = 15;  
c_TamEMail           = 50;  
c_TamCidade          = 50;  
c_TamUf              = 2;  
c_TamChaveDecifragem = 70;
```

Códigos de erros

```
c_ErroNaoConectado      = 1;  
c_ErroNaoHaDados        = 2;  
c_ErroTransInvalida     = 3;  
c_ErroOpCancelado       = 4;  
c_ErroArqNaoExiste      = 5;  
c_ErroInsertRegistroUpload = 6;  
c_ErroUpdateRegistroUpload = 7;  
c_ErroLocateRegistroUpload = 8;  
c_ErroUpdateUsuarios    = 9;  
c_ErroUsuarioInexiste   = 10;  
c_ErroSenhaInvalida     = 11;  
c_ErroExceptionLocate   = 12;  
c_ErroDestinatarioInexiste = 13;  
c_ErroAcessoUsrGrp      = 14;  
c_ErroAcessoArquivos    = 15;  
c_ErroUserOrigemInexiste = 16;  
c_ErroSequenciaTransacao = 17;  
c_ErroAcessoUser        = 18;  
c_ErroSemRespostaCliente = 19;  
c_ErroConsultaSemDados  = 20;
```

c_ErroLocateRegistroDnload	= 21;
c_ErroUpdateAposDnload	= 22;
c_ErroExceptionTransacao	= 23;
c_ErroTbArquivosBloqueada	= 24;
c_ErroTbUsuariosBloqueada	= 25;
c_ErroCriacaoQuery	= 26;
c_ErroAcessoPermissoes	= 27;
c_ErroAcessoGrupos	= 28;
c_ErroInclusao	= 29;
c_ErroEdicao	= 30;
c_ErroExclusao	= 31;
c_ErroGrupoInexiste	= 32;
c_ErroAcessoConexoes	= 33;
c_ErroUsuarioJaExiste	= 34;
c_ErroGrupoJaExiste	= 35;
c_ErroUsuarioNaoCadastrado	= 36;
c_ErroUsrGrpJahExiste	= 37;
c_ErroUsrGrpNaoExiste	= 38;
c_ErroLoginJahExiste	= 39;
c_ErroLoginNaoCadastrado	= 40;
c_ErroLoginNaoExiste	= 41;
c_ErroAcessoLogins	= 42;
c_ErroUsrRelInexiste	= 43;
c_ErroRestricaoJaExiste	= 44;
c_ErroRestricaoNaoExiste	= 45;
c_ErroServiceApiNaoDisp	= 46;
c_ErroServiceNaoInstalado	= 47;
c_ErroServiceJahStartado	= 48;
c_ErroServiceNaoStartado	= 49;
c_ErroServiceInicioStart	= 50;
c_ErroServiceStartando	= 51;
c_ErroServiceInicioStop	= 52;
c_ErroServiceParando	= 53;
c_ErroTxRxArquivo	= 54;
c_ErroExecProg	= 55;
c_ErroOperacaoBloqueada	= 56;
c_ErroSemPermissaoParaFuncao	= 57;
c_ErroOperacaoIndevida	= 58;
c_ErroExisteLoginIgualUser	= 59;
c_ErroExisteUserIgualLogin	= 60;
c_ErroDeleteArquivo	= 61;
c_ErroRenameArquivo	= 62;
c_ErroExisteUserComEsteCodigo	= 63;
c_ErroExisteGrupoComEsteCodigo	= 64;
c_ErroComunicacao	= 65;
c_ErroPacoteInvalido	= 66;
c_ErroConexaoSemHandleArquivo	= 67;
c_ErroJahExisteAtribuicao	= 68;
c_ErroAtribuicaoNaoExiste	= 69;
c_ErroInclusaoAdministrador	= 70;
c_ErroContextoJaExiste	= 71;
c_ErroContextoInexiste	= 72;
c_ErroDependenciaContexto	= 73;
c_ErroAcessoContexto	= 74;
c_ErroAcessooxContexto	= 75;
c_ErroDiretorioRxInvalido	= 76;
c_ErroGabProtNaoEncontrado	= 77;

```
c_ErroDadosProtJaExiste           = 78;
c_ErroDadosProtNaoExiste          = 79;
c_ErroLeituraDadosProt            = 80;
c_ErroEnvioDadosProt              = 81;
c_ErroArqJahExiste                = 82;
c_ErroRecepcaoDadosProt           = 83;
c_ErroGeracaoReciboExterno        = 84;
c_ErroExecucaoTransacao           = 85;
c_ErroCompactacao                 = 86;
c_ErroDescompactacao              = 87;
c_ErroRecuperacaoArquivo          = 88;
c_ErroBancoDeDados                = 89;
c_ErroLeituraArquivoServidor      = 90;
c_ErroPosicionamentoArquivo      = 91;
c_ErroGravacaoArquivoServidor     = 92;
c_ErroSemConexaoSecundaria        = 93;
c_ErroNaoAtualizouConexaoSecundaria = 94;
c_ErroConexaoSecundariaAMesma     = 95;
c_ErroFinalizacaoArquivoServidor  = 96;
c_ErroAtualizacaoControleProtocolo = 97;
c_ErroExcecao                     = 98;
c_ErroNaChamada                   = 99;
c_ErroNaoExisteRelacionamento     =100;
c_ErroAindaExistemDadosDoProtocolo =101;
c_ErroGabProtJahExiste             =102;
c_ErroCriptografia                 =103;
c_ErroMaxConexoes                  =104;
c_ErroNenhumaRecepcao = 105;
c_ErroNenhumaTransmissao = 106;
c_ErroGravacaoServidorTamanho = 107;
c_ErroGravacaoServidorChecksum = 108;
c_ErroRegraExpurgoNaoExiste = 109;
c_ErroAcessoRegrasExpurgo = 110;
c_ErroNomeArquivo = 111;
c_ErroParametroInvalido = 112;
c_ErroProtocolo = 113;
c_ErroDirTmp = 114;
c_ErroTamanhoRecebido = 115;
c_ErroObtencaoChaveDecifra=116;
c_ErroDecifracao=117;
c_ErroBancoDeDadosBloqueado = 118;
c_ErroDesconexaoDaRede = 119;
c_ErroArquivoExpurgado = 120;
c_ErroAutorizacaoJaExiste=121;
c_ErroAutorizacaoNaoExiste=122;
c_ErroAcessoAdmExt=123;
c_ErroAdmPrincipal=124;
c_ErroAdmCriador=125;
c_ErroContextoBloqueado=126;
c_ErroContextoExpirado=127;
c_ErroNaoLogado=128;
c_ErroExisteArquivoAntigo=129;
c_ErroTransacaoAguardando=130;
c_ErroAreaDadosExpurgados=131;
c_ErroAreaDadosIndisponivel=132;
c_ErroAltSenha=133;
c_ErroVazao=134;
```

Macros para geração de recibos

c_TituloRecibo	= '[Titulo]';
c_NumeroControle	= '[NumeroControle]';
c_Contexto	= '[Contexto]';
c_DataEntrada	= '[DataEntrada]';
c_Remetente	= '[Remetente]';
c_Destinatario	= '[Destinatario]';
c_CodRemetente	= '[CodRemetente]';
c_CodDestinatario	= '[CodDestinatario]';
c_NomeArquivo	= '[NomeArquivo]';
c_DescricaoArquivo	= '[DescricaoArquivo]';
c_TamanhoArquivo	= '[TamanhoArquivo]';
c_DataArquivo	= '[DataArquivo]';
c_Recebedor	= '[Recebedor]';
c_DataRecepcao	= '[DataRecepcao]';
c_VersaoServidor	= '[VersaoServidor]';
c_VersaoCliente	= '[VersaoCliente]';
c_NomeServidor	= '[NomeServidor]';
c_NomeEstacao	= '[NomeEstacao]';
c_IdConexao	= '[IdConexao]';
c_DataAtual	= '[DataAtual]';
c_HoraAtual	= '[HoraAtual]';

Tipos e estruturas que podem ser utilizadas pelos aplicativos usuários da API AldeiaNet:

Estruturas de cadeias para passagens de parâmetros

t_CadeiaCodUsuario	= array[0..c_TamCodUsuario] of char;
t_CadeiaUsuarioExtenso	= array[0..c_TamUsuarioExtenso] of char;
t_CadeiaContexto	= array[0..c_TamContexto] of char;
t_CadeiaSenha	= array[0..c_TamSenha] of char;
t_CadeiaNomeArquivo	= array[0..c_TamNomeArquivo] of char;
t_CadeiaDescricao	= array[0..c_TamDescricao] of char;
t_CadeiaPathArquivo	= array[0..c_TamPathArquivo] of char;
t_CadeiaChaveDecifragem	= array[0..c_TamChaveDecifragem] of char;
t_CadeiaSenhaAldeianet	= array[0..c_TamSenhaCriptogr] of char;
t_CadeiaVersao	= array[0..c_TamVersao] of char;
t_CadeiaEndereco	= array[0..c_TamEndereco] of char;
t_CadeiaNumEndereco	= array[0..c_TamNumEndereco] of char;
t_CadeiaComplementoEndereco	= array[0..c_TamComplEndereco] of char;
t_CadeiaBairro	= array[0..c_TamBairro] of char;
t_CadeiaCep	= array[0..c_TamCep] of char;
t_CadeiaTelefone	= array[0..c_TamTelefone] of char;
t_CadeiaEMail	= array[0..c_TamEMail] of char;
t_CadeiaCidade	= array[0..c_TamCidade] of char;
t_CadeiaUF	= array[0..c_TamUf] of char;
t_CadeiaNomeMaquina	= array[0..c_TamNomeMaquina] of char;
t_CadeiaDataHora	= array[0..c_TamData] of char;

Observações:

Formato do *t_CadeiaDataHora* = AAAAMMDDHHMMSSNNN

Onde: AAAA = Ano com quatro posições; MM = Mês; DD = Dia;

HH = Hora; MM = Minutos; SS = Segundos; NNN = Milissegundos.

Todos as cadeia devem ser terminadas com “nulo”. Ou seja, seu último preenchimento lógico deve ser seguido por um caráter #0.

Estruturas passadas nas chamadas da API

```
t_RegSessaoExt = packed record
    v_Contexto          : t_CadeiaContexto;
    v_Usuario           : t_CadeiaCodUsuario;
    v_Senha             : t_CadeiaSenha;
end;

t_RegIncluirArquivoExt = packed record
    v_NomeArquivo       : t_CadeiaNomeArquivo;
    v_Descricao         : t_CadeiaDescricao;
    v_UsrOrigem         : t_CadeiaCodUsuario;
    v_UsrDestino        : t_CadeiaCodUsuario;
    v_DirDnload         : t_CadeiaPathArquivo;
    v_ForcaDirDnload    : WordBool;
    v_ConfirmaRecepcao  : WordBool;
    v_DataEntrada       : t_CadeiaDataHora; //saída
    v_NumControle       : Longint; {saída}
    v_IsnArquivo        : Longint; {saída}
    v_Reservado         : array[0..255] of char;
end;

T_RegConsultaArquivosExt = packed record
    v_UsrDestino        : t_CadeiaCodUsuario;
    v_UsrOrigem         : t_CadeiaCodUsuario;
    v_RefNomeArquivo    : t_CadeiaNomeArquivo
    v_StatusDoDestino   : Longint;
    v_StatusDoLogon     : Longint;
    v_FlagData          : Longint;
    v_DataEntrada       : t_CadeiaDataHora;
    v_RegInicio         : Longint;
    v_PontConsulta      : Longint;
end;

T_RegInfoArqExt = packed record
    v_IsnArquivo        : Longint;
    v_NumControle       : Longint;
    v_UsuarioOrigem     : t_CadeiaCodUsuario;
    v_UsuarioDestino    : t_CadeiaCodUsuario;
    v_DataEntrada       : t_CadeiaDataHora;
    v_DataRecupDono     : t_CadeiaDataHora;
    v_DataRecupLogon    : t_CadeiaDataHora;
    v_StatusDono        : Longint;
    v_StatusLogon       : Longint;
    v_NomeArquivo       : t_CadeiaNomeArquivo
    v_Tamanho           : Longint;
    v_DataArquivo       : t_CadeiaDataHora;
    v_Descricao         : t_CadeiaDescricao;
    v_TamCompactado     : Longint;
    v_TemProtocolo      : WordBool;
```

```
v_TipoCompac      : Longint; //0=AldeiaNet
v_TipoCript       : Longint; //0=Nenhuma;1=AldeiaNet;2=CEPESC
v_RegProximo      : Longint;
v_Continua        : WordBool;
v_PontConsulta    : Longint;
v_Reservado       : array[1..255] of char;
end;
```

```
t_RegObterArquivoExt = packed record
  v_IsnArquivo      : Longint;
  v_DiretorioDestino : t_CadeiaPathArquivo;
  v_CaminhoCompletoSaida : t_CadeiaPathArquivo; //saída
  v_ChaveDecifragem  : t_CadeiaChaveDecifragem;
  v_Reservado       : array[1..185] of char;
end;
```

```
t_RegRemoverArquivoExt = packed record
  v_IsnArquivo      : Longint;
end;
```

```
T_RegIncluirUsuarioExt = packed record
  v_CodUser         : t_CadeiaCodUsuario;
  v_Senha           : t_CadeiaSenhaAldeianet;
  v_NomeUser        : t_CadeiaUsuarioExtenso;
  v_Endereco        : t_CadeiaEndereco;
  v_Numero          : t_CadeiaNumEndereco;
  v_Complemento     : t_CadeiaComplementoEndereco;
  v_Bairro          : t_CadeiaBairro;
  v_Cep             : t_CadeiaCep;
  v_Telefone        : t_CadeiaTelefone;
  v_Fax             : t_CadeiaTelefone;
  v_Email           : t_CadeiaEmail;
  v_Cidade          : t_CadeiaCidade;
  v_Uf              : t_CadeiaUF;
  v_OrigemPadrao    : t_CadeiaCodUsuario;
  v_DestinoPadrao   : t_CadeiaCodUsuario;
  v_GrupoPadrao    : t_CadeiaCodUsuario;
  v_IsnUsuario      : Longint; {saída}
end;
```

```
t_ReciboExterno = packed record
  v_Titulo          : array[0..c_tamTitulo] of char;
  v_NumControle     : Longint;
  v_Contexto        : t_CadeiaContexto
  v_DataEntrada     : t_CadeiaDataHora;
  v_CodUsuarioRemetente : t_CadeiaCodUsuario;
  v_NomeUsuarioRemetente : t_CadeiaUsuarioExtenso;
  v_CodUsuarioDestinatario : t_CadeiaCodUsuario;
  v_NomeUsuarioDestinatario : t_CadeiaUsuarioExtenso;
  v_NomeArquivo     : t_CadeiaNomeArquivo
  v_Descricao       : t_CadeiaDescricao;
  v_TamanhoArquivo  : Longint;
  v_DataArquivo     : t_CadeiaDataHora;
```

```

v_GeracaoRecibo      : Longint;
//-- campos para recibo de confirmação de recebimento ---
v_CodUsrRecebedor    : t_CadeiaCodUsuario;
v_NomeUsrRecebedor   : t_CadeiaUsuarioExtenso;
v_DataRecepcao       : t_CadeiaDataHora;
//-- outros campos de controle ---
v_VersaoServidor     : t_CadeiaVersao;
v_VersaoCliente      : t_CadeiaVersao;
v_NomeServidor       : t_CadeiaNomeMaquina;
v_NomeEstacao        : t_CadeiaNomeMaquina;
v_IdConexao          : Longint;
//--
v_Reservado          : array[1..128] of char;
end;

//-- consultar recibos relacionados
t_RegPedeRecibosExt = record
  v_IsnArquivo : Longint;
  v_RegInicio  : Longint;
  v_PontConsulta : Longint;
end;

//--
T_RegReciboRelacionadoExt = record
  v_IsnArquivo      : Longint;
  v_DataEntrada     : array[0..c_TamData] of char;
  v_NomeArquivo     : array[0..c_TamNomeArquivo] of char;
  v_Tamanho         : Longint;
  v_DataArquivo     : array[0..c_TamData] of char;
  v_CategoriaRecibo : Longint;
end;

```

Conversões de Borland Delphi para Microsoft Visual Basic

• Relação entre Tipos e Constantes

Sugestões para conversões entre os tipos do Borland Delphi utilizados nesta documentação e tipos do Microsoft Visual Basic.

Todas essas definições em Visual Basic devem ficar dentro de módulos do sistema (.BAS).

Tipos simples

DELPHI 32 bits	VISUAL BASIC 32 bits
Integer	Long
Longint	Long
WordBool	Boolean
Char	Byte

Constantes de tamanho de cadeias

Algumas das cadeias utilizadas nesta API são do tipo `array[0..TamanhoMaximo]`, ou seja, tem tamanho fixo de `TamanhoMaximo+1`, pois começam de 0 (zero) e vai até o valor especificado pela constante utilizada. Isso porque leva-se em conta que utilizamos cadeias com terminação lógica em zero binário (`chr(0)`), e devemos sempre reservar um byte a mais em cada cadeia, para a sua finalização lógica, caso aconteça de atingirem o tamanho máximo permitido.

Quando as constantes que descrevem o tamanho máximo dessas cadeias forem ser declaradas em Visual Basic, lembrar de sempre definí-las com o valor incrementado de +1, pois como se usará strings (cadeias) de tamanho fixo no VB, e por definição, em VB não se define cadeias fixas como vetores, como no Delphi, então ter-se-á que especificar apenas o seu tamanho, como em `String*Tamanho`.

Assim, as definições das constantes de tamanho de cadeia em Visual Basic, para facilitar a visualização e documentação, ficariam:

```
const c_TamTitulo           = 100 + 1
const c_TamNumControle      = 10 + 1
const c_TamContexto        = 15 + 1
const c_TamData             = 17 + 1
const c_TamCodUsuario       = 25 + 1
const c_TamExtensoUsuario   = 80 + 1
const c_TamNomeArquivo      = 120 + 1
const c_TamPathArquivo      = 255 + 1
const c_TamDescricao        = 200 + 1
const c_TamLongint          = 10 + 1
const c_TamSenha            = 10 + 1
const c_TamVersao           = 10 + 1
const c_TamNomeMaquina      = 15 + 1
const c_TamSenhaCriptogr    = 21 + 1
const c_TamEndereco         = 50 + 1
const c_TamNumEndereco      = 5 + 1
const c_TamComplEndereco    = 45 + 1
const c_TamBairro           = 50 + 1
const c_TamCep              = 10 + 1
const c_TamTelefone         = 15 + 1
const c_TamEMail            = 50 + 1
const c_TamCidade           = 50 + 1
const c_TamUf               = 2 + 1
const c_TamChaveDecifragem  = 70 + 1
```

Estruturas de cadeias

Estruturas do tipo `T_Cadeia...` (Vetor com cadeia de caracteres):

Exemplo:

Em Delphi:

```
t_CadeiaCodUsuario = array[0..c_TamCodUsuario] of char;
```

Em Visual Basic ficaria:

```
Type t_CadeiaCodUsuario
    Valor as String * c_TamCodUsuario
End Type
```

Obs.: Considerando aqui que as constantes foram definidas de acordo com as sugestões citadas anteriormente.

No momento de declarar a variável, então o tipo definido seria utilizado. Exemplo:

```
v_CadeiaCodUsuario as t_CadeiaCodUsuario
```

Isso garante compatibilidade com a estrutura de vetor de caracteres em Delphi, além de operações de manipulação de cadeia do Visual Basic poderem ser utilizadas normalmente nesta estrutura. Por exemplo, para uma atribuição de literal à essa cadeia, seria feito assim:

```
V_CadeiaCodUsuario.Valor = "MARIA" + CHR(0)
```

Obs.: Sempre lembrar de acrescentar o CHR(0) (Nulo) para delimitar o tamanho lógico da cadeia.

Outra alternativa seria definir a variável diretamente em cima da definição de cadeia fixa do VB. Mas perderia a vantagem de se utilizar um tipo definido (User Type Defined), para efeito de reutilização nas definições. Exemplo:

```
V_CadeiaCodUsuario as String * c_TamCodUsuario
```

Nesse caso, uma atribuição de literal à essa cadeia, seria feito assim:

```
V_CadeiaCodUsuario = "MARIA" + CHR(0)
```

Outra opção (menos recomendável por questão de performance, quando da atribuição de valores pelo aplicativo VB, que precisaria ser em um loop), seria utilizar vetores (arrays) estáticos no VB, para representar as cadeia de caracteres do Delphi. Assim, em matéria de visualização ficariam muito parecidos. Assim:

Em Delphi:

```
Type t_CadeiaCodUsuario = array[0..c_TamCodUsuario] of char;
```

```
Var v_CadeiaCodUsuario : t_CadeiaCodUsuario;
```

Em Visual Basic:

```
Rem ** aqui a constante não precisa ser +1 **
```

```
Const c_TamCodUsuario = 25
```

```
Static v_CadeiaCodUsuario(0 to c_TamCodUsuario) as byte
```

E na hora da atribuição:

```
Valor = "MARIA"
```

```
...
```

```
for I = 0 to Len(Valor)
```

```
    v_CadeiaCodUsuario(I) = Valor(I)
```

```
Next I
```

Estruturas do tipo *record*:

Exemplo:

```
t_RegObterArquivoExt = record
    v_IsnArquivo      : Longint;
    v_DiretorioDestino : t_CadeiaPathArquivo;
    v_CaminhoCompletoSaida : t_CadeiaPathArquivo;
    v_ChaveDecifragem  : t_CadeiaChaveDecifragem;
end;
```

Em Visual Basic:

```
Type t_RegObterArquivoExt
    v_IsnArquivo      as Long
    v_DiretorioDestino as t_CadeiaPathArquivo
    v_CaminhoCompletoSaida as t_CadeiaPathArquivo
    v_ChaveDecifragem  as t_CadeiaChaveDecifragem
End Type
```

Ou então, pode-se também trocar os tipos das cadeias por strings fixas com referências diretas às constantes, ou literais, no caso de não se querer utilizar estruturas types das cadeias. Exemplo:

```
Type t_RegObterArquivoExt
    v_IsnArquivo      as Long
    v_DiretorioDestino as String * c_TamPathArquivo
    v_CaminhoCompletoSaida as String * c_TamPathArquivo
    v_ChaveDecifragem  as String * c_TamChaveDecifragem
End Type
```

Parâmetros passados como Pchar (Ponteiro para Cadeia de Caracteres):

Esse tipo de parâmetro, quando passado como referência (como são todos os casos aqui na API AldeiaNet), podem ser substituídos no Visual Basic simplesmente por Cadeias (String) passadas como referência. Exemplo: Na função **F_METAFILESALVAARQUIVO**, o parâmetro passado (v_NomeArquivo) é desse tipo. Então, em VB, a declaração seria:

```
Declare Function F_METAFILESALVAARQUIVO Lib "AldNetBc.dll" (v_NomeArquivo
As String) As Long
```

- **Declarações da API para Visual Basic**

A seguir, há o protótipo de definição de todas as rotinas constantes na API AldeiaNet. Estas declarações devem ficar dentro de um módulo do sistema (.BAS):

Obs.: No VB, o tipo de parâmetro default é *ByRef* (passagem por referência). Se não for especificado, este tipo de passagem de parâmetro é assumido.

```
Declare Function F_ABRESESSAO Lib "AldNetBc.dll" (ByVal v_Contexto as
String, ByVal v_Usuario as String, ByVal v_Senha as String) As Long
```

```
Declare Function F_ADICIONAARQUIVO Lib "AldNetBc.dll" (ByRef
v_RegIncluirArquivoExt As t_RegIncluirArquivoExt) As Long
```

```
Declare Function F_CONSULTAARQUIVOS Lib "AldNetBc.dll" (ByRef  
v_RegConsultaArquivosExt As t_RegConsultaArquivosExt, ByRef  
v_RegInfoArqExt As t_RegInfoArqExt) As Long  
  
Declare Function F_OBTEMARQUIVO Lib "AldNetBc.dll" (ByRef  
v_RegObterArquivoExt As t_RegObterArquivoExt) As Long  
  
Declare Function F_REMOVEARQUIVO Lib "AldNetBc.dll" (ByRef  
v_RegRemoverArquivoExt As t_RegRemoverArquivoExt) As Long  
  
Declare Function F_CONSULTARECIBOS Lib "AldNetBc.dll" (ByRef  
v_RegPedeRecibosExt As t_RegPedeRecibosExt, ByRef  
v_RegReciboRelacionadoExt As T_RegReciboRelacionadoExt, ByRef  
v_RegProximo As Long, ByRef v_Continua As Boolean, ByRef v_PtQuery As  
Long) As Long  
  
Declare Function F_FECHASESSAO Lib "AldNetBc.dll" () As Integer  
  
Declare Function F_GERARECIBOONLINE Lib "AldNetBc.dll" (ByRef  
v_ReciboExterno As T_ReciboExterno, ByVal v_ModeloArquivoRecibo As  
String) As Long  
  
Declare Function F_METAFILELERMODELO Lib "AldNetBc.dll" (ByVal  
v_NomeModelo As String) As Long  
  
Declare Function F_METAFILESUBSTITUIMACRO Lib "AldNetBc.dll" (ByVal  
v_Macro As String, ByVal v_Substituto As String) As Long  
  
Declare Function F_METAFILESALVAARQUIVO Lib "AldNetBc.dll" (ByVal  
v_NomeArquivo As String) As Long  
  
Declare Function F_CARREGACONFIGURACAO Lib "AldNetBc.dll" (ByVal  
v_NomeArquivoCfg As String) As Long
```

- **Constantes, tipos e estruturas utilizadas pela API, para Visual Basic**

Constantes que podem ser utilizadas pelos aplicativos usuários da API AldeiaNet (Obs. Para constantes globais, substituir *const* por *Global Const*):

FlagData para consulta de arquivos

```
const Igual      = 0  
const Menor      = 1  
const MenorIgual = 2  
const MaiorIgual = 3  
const Maior      = 4  
const Diferente  = 5  
const Qualquer   = 6
```

Status de recebimento para consulta de arquivos

```
const Todos      = 0  
const JaRecebidos = 1  
const NaoRecebidos = 2
```

Categorias de recibos

```
Const crNenhum = 0
```

```
Const crReciboEntrada = 1
Const crReciboConfirmacao = 2
```

Tamanhos de cadeias

```
const c_TamTitulo           = 100 + 1
const c_TamNumControle      = 10 + 1
const c_TamContexto        = 15 + 1
const c_TamData             = 17 + 1
const c_TamCodUsuario       = 25 + 1
const c_TamExtensoUsuario   = 80 + 1
const c_TamNomeArquivo      = 120 + 1
const c_TamPathArquivo      = 255 + 1
const c_TamDescricao        = 200 + 1
const c_TamLongint          = 10 + 1
const c_TamSenha            = 10 + 1
const c_TamVersao           = 10 + 1
const c_TamNomeMaquina      = 15 + 1
const c_TamSenhaCriptogr    = 21 + 1
const c_TamEndereco         = 50 + 1
const c_TamNumEndereco      = 5 + 1
const c_TamComplEndereco    = 45 + 1
const c_TamBairro           = 50 + 1
const c_TamCep              = 10 + 1
const c_TamTelefone         = 15 + 1
const c_TamEMail            = 50 + 1
const c_TamCidade           = 50 + 1
const c_TamUf               = 2 + 1
const c_TamChaveDecifragem  = 70 + 1
```

Códigos de erros

```
const c_ErroNaoConectado    = 1
const c_ErroNaoHaDados      = 2
const c_ErroTransInvalida   = 3
const c_ErroOpCancelado     = 4
const c_ErroArqNaoExiste    = 5
const c_ErroInsertRegistroUpload = 6
const c_ErroUpdateRegistroUpload = 7
const c_ErroLocateRegistroUpload = 8
const c_ErroUpdateUsuarios  = 9
const c_ErroUsuarioInexiste = 10
const c_ErroSenhaInvalida   = 11
const c_ErroExceptionLocate = 12
const c_ErroDestinatarioInexiste = 13
const c_ErroAcessoUsrGrp    = 14
const c_ErroAcessoArquivos  = 15
const c_ErroUserOrigemInexiste = 16
const c_ErroSequenciaTransacao = 17
const c_ErroAcessoUser      = 18
const c_ErroSemRespostaCliente = 19
const c_ErroConsultaSemDados = 20
const c_ErroLocateRegistroDnload = 21
const c_ErroUpdateAposDnload = 22
const c_ErroExceptionTransacao = 23
const c_ErroTbArquivosBloqueada = 24
const c_ErroTbUsuariosBloqueada = 25
const c_ErroCriacaoQuery     = 26
const c_ErroAcessoPermissoes = 27
```

const c_ErroAcessoGrupos	= 28
const c_ErroInclusao	= 29
const c_ErroEdicao	= 30
const c_ErroExclusao	= 31
const c_ErroGrupoInexiste	= 32
const c_ErroAcessoConexoes	= 33
const c_ErroUsuarioJaExiste	= 34
const c_ErroGrupoJaExiste	= 35
const c_ErroUsuarioNaoCadastrado	= 36
const c_ErroUsrGrpJahExiste	= 37
const c_ErroUsrGrpNaoExiste	= 38
const c_ErroLoginJahExiste	= 39
const c_ErroLoginNaoCadastrado	= 40
const c_ErroLoginNaoExiste	= 41
const c_ErroAcessoLogins	= 42
const c_ErroUsrRelInexiste	= 43
const c_ErroRestricaoJaExiste	= 44
const c_ErroRestricaoNaoExiste	= 45
const c_ErroServiceApiNaoDisp	= 46
const c_ErroServiceNaoInstalado	= 47
const c_ErroServiceJahStartado	= 48
const c_ErroServiceNaoStartado	= 49
const c_ErroServiceInicioStart	= 50
const c_ErroServiceStartando	= 51
const c_ErroServiceInicioStop	= 52
const c_ErroServiceParando	= 53
const c_ErroTxRxArquivo	= 54
const c_ErroExecProg	= 55
const c_ErroOperacaoBloqueada	= 56
const c_ErroSemPermissaoParaFuncao	= 57
const c_ErroOperacaoIndevida	= 58
const c_ErroExisteLoginIgualUser	= 59
const c_ErroExisteUserIgualLogin	= 60
const c_ErroDeleteArquivo	= 61
const c_ErroRenameArquivo	= 62
const c_ErroExisteUserComEsteCodigo	= 63
const c_ErroExisteGrupoComEsteCodigo	= 64
const c_ErroComunicacao	= 65
const c_ErroPacoteInvalido	= 66
const c_ErroConexaoSemHandleArquivo	= 67
const c_ErroJahExisteAtribuicao	= 68
const c_ErroAtribuicaoNaoExiste	= 69
const c_ErroInclusaoAdministrador	= 70
const c_ErroContextoJaExiste	= 71
const c_ErroContextoInexiste	= 72
const c_ErroDependenciaContexto	= 73
const c_ErroAcessoContexto	= 74
const c_ErroAcessoContexto	= 75
const c_ErroDiretorioRxInvalido	= 76
const c_ErroGabProtNaoEncontrado	= 77
const c_ErroDadosProtJaExiste	= 78
const c_ErroDadosProtNaoExiste	= 79
const c_ErroLeituraDadosProt	= 80
const c_ErroEnvioDadosProt	= 81
const c_ErroArqJahExiste	= 82
const c_ErroRecepcaoDadosProt	= 83
const c_ErroGeracaoReciboExterno	= 84

```

const c_ErroExecucaoTransacao          = 85
const c_ErroCompactacao                = 86
const c_ErroDescompactacao             = 87
const c_ErroRecuperacaoArquivo         = 88
const c_ErroBancoDeDados               = 89
const c_ErroLeituraArquivoServidor     = 90
const c_ErroPosicionamentoArquivo     = 91
const c_ErroGravacaoArquivoServidor    = 92
const c_ErroSemConexaoSecundaria       = 93
const c_ErroNaoAtualizouConexaoSecundaria=94
const c_ErroConexaoSecundariaAMesma    = 95
const c_ErroFinalizacaoArquivoServidor = 96
const c_ErroAtualizacaoControleProtocolo = 97
const c_ErroExcecao                    = 98
const c_ErroNaChamada                   = 99
const c_ErroNaoExisteRelacionamento    =100
const c_ErroAindaExistemDadosDoProtocolo =101
const c_ErroGabProtJahExiste           =102
const c_ErroCriptografia                =103
const c_ErroMaxConexoes                 =104
const c_ErroNenhumaRecepcao            = 105
const c_ErroNenhumaTransmissao         = 106
const c_ErroGravacaoServidorTamanho    = 107
const c_ErroGravacaoServidorChecksum   = 108
const c_ErroRegraExpurgoNaoExiste      = 109
const c_ErroAcessoRegrasExpurgo        = 110
const c_ErroNomeArquivo                = 111
const c_ErroParametroInvalido          = 112
const c_ErroProtocolo                  = 113
const c_ErroDirTmp                     = 114
const c_ErroTamanhoRecebido            = 115
const c_ErroObtencaoChaveDecifra       =116
const c_ErroDecifracao                 =117
const c_ErroBancoDeDadosBloqueado      = 118
const c_ErroDesconexaoDaRede           = 119
const c_ErroArquivoExpurgado           = 120
const c_ErroAutorizacaoJaExiste        =121
const c_ErroAutorizacaoNaoExiste       =122
const c_ErroAcessoAdmExt               =123
const c_ErroAdmPrincipal               =124
const c_ErroAdmCriador                 =125
const c_ErroContextoBloqueado           =126
const c_ErroContextoExpirado           =127
const c_ErroNaoLogado                  =128
const c_ErroExisteArquivoAntigo        =129
const c_ErroTransacaoAguardando        =130
const c_ErroAreaDadosExpurgados        =131
const c_ErroAreaDadosIndisponivel      =132
const c_ErroAltSenha                   =133
const c_ErroVazao                      =134

```

Macros para geração de recibos

```

const c_TituloRecibo                   = '[Titulo]'
const c_NumeroControle                 = '[NumeroControle]'
const c_Contexto                       = '[Contexto]'
const c_DataEntrada                    = '[DataEntrada]'
const c_Remetente                      = '[Remetente]'

```

```

const c_Destinatario           = '[Destinatario]'
const c_CodRemetente           = '[CodRemetente]'
const c_CodDestinatario       = '[CodDestinatario]'
const c_NomeArquivo            = '[NomeArquivo]'
const c_DescricaoArquivo       = '[DescricaoArquivo]'
const c_TamanhoArquivo         = '[TamanhoArquivo]'
const c_DataArquivo            = '[DataArquivo]'
const c_Recebedor              = '[Recebedor]'
const c_DataRecepcao           = '[DataRecepcao]'
const c_VersaoServidor         = '[VersaoServidor]'
const c_VersaoCliente          = '[VersaoCliente]'
const c_NomeServidor           = '[NomeServidor]'
const c_NomeEstacao            = '[NomeEstacao]'
const c_IdConexao              = '[IdConexao]'
const c_DataAtual              = '[DataAtual]'
const c_HoraAtual              = '[HoraAtual]'

```

Estruturas passadas nas chamadas da API

```

type t_RegIncluirArquivoExt
  v_NomeArquivo      as t_CadeiaNomeArquivo
  v_Descricao        as t_CadeiaDescricao
  v_UsrOrigem        as t_CadeiaCodUsuario
  v_UsrDestino       as t_CadeiaCodUsuario
  v_DirDnload        as t_CadeiaPathArquivo
  v_ForcaDirDnload   as Boolean
  v_ConfirmaRecepcao as Boolean
  v_DataEntrada       as t_CadeiaDataHora
  v_NumControle       as Long
  v_IsnArquivo        as Long
  v_Reservado         as String*256
end type

type T_RegConsultaArquivosExt
  v_UsrDestino       as t_CadeiaCodUsuario
  v_UsrOrigem        as t_CadeiaCodUsuario
  v_RefNomeArquivo   as t_CadeiaNomeArquivo
  v_StatusDoDestino  as Long
  v_StatusDoLogon    as Long
  v_FlagData         as Long
  v_DataEntrada       as t_CadeiaDataHora
  v_RegInicio        as Long
  v_PontConsulta     as Long
  v_HabilitarConsultaHorario as Boolean
  v_NumControle      as Long
  v_ConsultaExpurgados as Boolean
  v_TipoExclusao     as Long
  v_OrigemExclusao   as String * c_TamCodusuario
  v_DataExclusao     as String * c_TamData
  v_FlagDataExclusao as Long
  v_HabConsHorExclusao as Boolean
  v_Reservado        as String * 128
end type

type T_RegInfoArqExt

```

```
v_IsnArquivo      as Long
v_NumControle     as Long
v_UsuarioOrigem   as t_CadeiaCodUsuario
v_UsuarioDestino  as t_CadeiaCodUsuario
v_DataEntrada     as t_CadeiaDataHora
v_DataRecupDono   as t_CadeiaDataHora
v_DataRecupLogon  as t_CadeiaDataHora
v_StatusDono      as Long
v_StatusLogon     as Long
v_NomeArquivo     as t_CadeiaNomeArquivo
v_Tamanho         as Long
v_DataArquivo     as t_CadeiaDataHora
v_Descricao       as t_CadeiaDescricao
v_TamCompactado   as Long
v_TemProtocolo    as Boolean
v_TipoCompac      as Long 'AldeiaNet
v_TipoCript       as Long '0=Nenhuma;1=AldeiaNet;2=CEPESC
v_RegProximo      as Long
v_Continua        as Boolean
v_PontConsulta    as Long
v_VersaoMaior     as String*2
v_VersaoMenor     as String*2
v_VersaoRelease   as String*2
v_VersaoBuild     as String*2
v_Operador        as String * c_TamCodUsuario
v_Estacao         as String * 15 + 1
v_Reservado       as String * 204
end type
```

```
type t_RegObterArquivoExt
  v_IsnArquivo      as Long
  v_DiretorioDestino as t_CadeiaPathArquivo
  v_CaminhoCompletoSaida as t_CadeiaPathArquivo
  v_ChaveDecifragem as t_CadeiaChaveDecifragem
end type
```

```
type t_RegRemoverArquivoExt
  v_IsnArquivo      as Long
end type
```

```
type t_ReciboExterno
  v_Titulo          as String * c_tamTitulo
  v_NumControle     as Long
  v_Contexto        as t_CadeiaContexto
  v_DataEntrada     as t_CadeiaDataHora
  v_CodUsuarioRemetente as t_CadeiaCodUsuario
  v_NomeUsuarioRemetente as t_CadeiaUsuarioExtenso
  v_CodUsuarioDestinatario as t_CadeiaCodUsuario
  v_NomeUsuarioDestinatario as t_CadeiaUsuarioExtenso
  v_NomeArquivo     as t_CadeiaNomeArquivo
  v_Descricao       as t_CadeiaDescricao
  v_TamanhoArquivo  as Long
  v_DataArquivo     as t_CadeiaDataHora
  v_GeracaoRecibo   as Long
```

```
'/-- campos para recibo de confirmação de recebimento ---
v_CodUsrRecebedor      as t_CadeiaCodUsuario
v_NomeUsrRecebedor     as t_CadeiaUsuarioExtenso
v_DataRecepcao         as t_CadeiaDataHora
'/-- outros campos de controle ---
v_VersaoServidor       as t_CadeiaVersao
v_VersaoCliente        as t_CadeiaVersao
v_NomeServidor         as t_CadeiaNomeMaquina
v_NomeEstacao          as t_CadeiaNomeMaquina
v_IdConexao            as Long
v_UsrSessaoValidacao   as String * c_TamCodUsuario
v_SenhaUsrSessaoValidacao as String * c_TamSenha
v_CodOperador          as String * c_TamCodUsuario
v_Reservado            as String * 65
end type
```

```
Type t_RegPedeRecibosExt
  v_IsnArquivo As Long
  v_RegInicio  As Long
  v_PontConsulta As Long
End Type
```

```
Type T_RegReciboRelacionadoExt
  v_IsnArquivo      As Long
  v_DataEntrada     As String * c_TamData
  v_NomeArquivo     As String * c_TamNomeArquivo
  v_Tamanho         As Long
  v_DataArquivo     As String * c_TamData
  v_CategoriaRecibo As Long
End Type
```
