Manual do Desenvolvedor

Terminal de Consulta 506 Mídia

DLL SC504 V 2.6.3

R00.22

Sumário

Inicialização e Finalização

tc_startserver

int __stdcall tc_startserver(HWND mywhnd, int conecmsg, int commumsg);
Primeira função a ser chamada para iniciar a comunicação, onde os terminais já se conectarão ao servidor.
Retorna: 1 para sucesso, 0 para erro.

- mywhnd: Handle para a janela principal do software, onde a DLL irá mandar as mensagens para troca de dados. Caso não necessite, seu valor deve ser NULL.
- conecmsg: Valor da mensagem que a DLL enviará quando um terminal conectar/desconectar.
- commumsg: Valor da mensagem que a DLL enviará quando terminal enviar dados.

tc_setWorkDir

int __stdcall tc_setWorkDir(char* path);
Deve ser chamada após a função tc_startserver, para indicar o diretório onde a dll poderá escrever arquivos temporários, caso não seja o diretório de trabalho, será usado o diretório atual do executável da aplicação.

tc_finishserver

void __stdcall tc_finishserver(void);
Após chamar esta função, a DLL libera a memória armazenada, desconecta todos os terminais e para de aceitar novas conexões.

Funções Diversas

tc_version

char __stdcall tc_version(void);
Retorna: versão da DLL.
Ex.: 0x20 corresponde à versão 2.0.

tc_gethostip

char * __stdcall tc_gethostip(char * cpHostIP);
Retorna: IP da máquina local em ASCII formatada por pontos.
cpHostIp: Array de bytes onde será escrito os dados.

tc_inet_ntoa

char * __stdcall tc_inet_ntoa(DWORD dwIP);
Retorna: IP do terminal em ASCII formatado por pontos.
dwIP: IP no formato de rede (DWORD).

tc_inet_addr

DWORD __stdcall tc_inet_addr(char *cpStrIP);
Retorna: IP do terminal no formato de rede (DWORD).
cpStrIP: Array de bytes em ASCII formatado por pontos.

tc_ipfromid

char * __stdcall tc_ipfromid(int ID);
Retorna: IP formatado por pontos de um terminal.

- ID: ID do terminal.

Comandos para o terminal

tc_sendlive

int __stdcall tc_sendlive(int ID);
Envia o comando de “vivo” para o terminal (IDvLive).
Retorna: 1 para sucesso, 0 para erro.

tc_sendalwayslive

int __stdcall tc_sendalwayslive(int ID);
Envia o comando de “sempre vivo” para o terminal (IDvAlwaysLive).
Retorna: 1 para sucesso, 0 para erro.

tc_reqconfig

int __stdcall tc_reqconfig(int ID);
Requisita configuração do terminal (IDvGetSetupTCP). Para receber a configuração utilize a função tc_getconfig (pág. 14).
Retorna: >0 para sucesso, <1 para erro.

tc_sendconfig

int __stdcall tc_sendconfig(int ID,ARG_SETUP_TCP * psConfig);
Envia configuração para o terminal (IDvSetSetupTCP).
Retorna: >0 para sucesso, <1 para erro.

```
typedef struct
{
DWORD dwMY_IP_ADD; //Endereço IP do terminal
DWORD dwServer_IP; //Endereço IP do servidor
DWORD dwNetMask; //Máscara de rede
DWORD dwGateway; //Endereço do Gateway
DWORD dwNameServer; //Endereço do DNS
char TCName[32]; //Nome do terminal (string terminada em caracter nulo
DWORD wPortsv; //Porta de comunicação com o servidor
char FTPs[100]; //String com o endereço do servidor de tualização
//("http://..." ou "ftp://....")
char FTPu[30]; //Nome do usuário para o servidor de FTP atualização)
char FTPp[30]; //Senha do usuário para o servidor de FTP
DWORD dwDHCP; //Se o terminal usar IP Dinâmico, será 1.senão 0.
DWORD dwSearchServer; // Se a busca do servidor for automática, será 1 senão 0.
}ARG_SETUP_TCP;
```

tc_sendconfigserial

int __stdcall tc_sendconfigserial(int ID, int sercom, ARG_SERIAL_CFG *config);
Envia configuração da serial para o terminal.
Retorna: >0 para sucesso, <1 para erro.

- sercom = 0 para COM 1, 1 para COM 2;

```
typedef struct
{
DWORD dwOpen; // 0 = abrir, 1 = fechar
DWORD dwBaud; // Velocidade da porta serial
BYTE bParity; // Paridade
BYTE bDataBits; // Databits
WORD wTimeOut; // reservado
}ARG_SERIAL_CFG;
```

tc_writeserial

int __stdcall tc_writeserial(int ID, int sercom, int tambuf, char *sbuf);
Escreve dados na serial do terminal.
Retorna: >0 para sucesso, <1 para erro.

- sercom: 0 para COM 1, 1 para COM 2;
- tambuf: número de bytes a serem escritos.
- sbuf: array de bytes contendo bytes a serem escritos.

tc_sersetstatus

int __stdcall tc_sersetstatus(int ID, int sercom, unsigned char sts);
Envia o estado do RTS e DTR da serial (IDvSetStatus).
Retorno: bit 0 = RTS, bit 1 = DTR;

- sercom: 0 para COM 1, 1 para COM 2;

```
typedef struct
{
BYTE aserial; // 0 = COM 1, 1 = COM 2
BYTE status; // Estado de controle
}ARG_SERIAL_STS;
```

tc_serreqstatus

int __stdcall tc_serreqstatus(int ID, int sercom);
Requisita estado do CDC, DSR e CTS da serial (IDvGetStatus). Para receber o estado da serial utilize a função tc_sergetstatus(pág. 13).

- sercom: 0 para COM 1, 1 para COM 2.

tc_updatesoft

int __stdcall tc_updatesoft(int ID);
Faz com que o terminal seja atualizado no endereço pré-configurado (IDUpdateSoft).
Retorna: >0 para sucesso, <1 para erro.

tc_restart

int __stdcall tc_restart(int ID);
Reinicia o terminal (IDRestart).
Retorna: >0 para sucesso, <1 para erro.

tc_stopadv

int __stdcall tc_stopadv(int ID);
Interrompe loop de imagens no terminal (IDStopAdv).
Retorna: >0 para sucesso, <1 para erro.

tc_reloadadv

int __stdcall tc_reloadadv(int ID, unsigned long doreload);
Recarregua loop de imagens no terminal (IDReloadAdv).
Retorna: >0 para sucesso, <1 para erro.

- doreload: deve ser sempre 1.

tc_deleteadv

int __stdcall tc_deleteadv(int ID);
Apaga os arquivos do loop de imagens (IDvDeleteAdv).
Retorna: >0 para sucesso, <1 para erro.

tc_reloadpreimg

int __stdcall tc_reloadpreimg(int ID, unsigned long doreload);
Faz com que o terminal recarregue suas imagens predefinidas (IDReloadPreImg). Se o número de imagens for maior do que o anterior, é recarregado também, o loop de imagens.
Retorna: >0 para sucesso, <1 para erro.

- doreload: deve ser sempre 1.

tc_deletepreimg

int __stdcall tc_deletepreimg(int ID);
Apaga os arquivos de imagens predefinidas (IDvDeletePreImg).
Retorna: >0 para sucesso, <1 para erro.

tc_sendimage

int __stdcall tc_sendimage(int ID, char * cpPaleta, char * cpImage);
Envia uma imagem na tela do terminal (IDvShowIMG).
Retorna: >0 para sucesso, <1 para erro.

- cpPaleta: Array de bytes contendo a paleta de cores (256 x 3 = 768 bytes).
- cpImage: Array de bytes contendo a imagem (320 x 240 = 768000 bytes para TC504 e 480 x 272 = 130560 bytes para TC506 mídia).

tc_sendimgfromfile

int __stdcall tc_sendimgfromfile(int ID, char *cpfilename);
Envia uma imagem na tela do terminal (IDvShowIMG).
Retorna: >0 para sucesso, <1 para erro.

- cpfilename: caminho da imagem (.BMP ou .JPG) que será enviada para o terminal.

tc_sendimageblock

int __stdcall tc_sendimageblock(int ID, int x, int y, int width, int height, char *cpImagem);
Envia dimensão e posição de imagem na tela do terminal (IdvShowImageBlock).
Retorna: >0 para sucesso, <1 para erro.

- cpImagem: Array de byte contendo a imagem a ser mostrada (width x height bytes).

- x,y,width,height: Posição, largura e altura de onde será desenhada a imagem.

tc_enviapaleta

int __stdcall tc_enviapaleta(int ID, char * cpPaleta);
Envia paleta de cores para o terminal (IDvSendPalette).
Retorna: >0 para sucesso, <1 para erro.

- cpPaleta: Array de bytes contendo a paleta de cores (256 x 3 = 768 bytes).

tc_sendtext

int __stdcall tc_sendtext(int ID, ARG_DISPALY_TEXT * psText);
Envia texto pro display.
Retorna: >0 para sucesso, <1 para erro.
typedef struct
{
WORD wPosX; // Posição X
WORD wPosY; // Posição Y
char sText[128]; // Texto a ser escrito
char sFont[32]; // Caminho da fonte utilizada (ex: \fonts\lucida12.bmp
WORD wSize; // reservado
WORD wColor; // Cor do texto
WORD wBGColor; // Cor do fundo
}ARG_DISPLAY_TEXT;

tc_fillscreen

int __stdcall tc_fillscreen(int ID, short color);
Escreve texto na tela do terminal (IDvShowText).
Retorna: >0 para sucesso, <1 para erro.

- color: cor de preenchimento.

tc_loadimage

int __stdcall tc_loadimage(int ID, int frame);
Mostra uma imagem pre-carregada na tela do terminal (IDShowFrame).
Retorna: >0 para sucesso, <1 para erro.

- frame: número do frame que será mostrado no terminal (o primeiro frame é o 0).

tc_reqfile

int __stdcall tc_reqfile(int ID, char *pcFilename);
Requisita um arquivo do terminal. Após receber o evento “RIDvRecvFile”, deve-se chamar a função tc_getfile, para receber o arquivo.
Retorna: >0 para sucesso, <1 para erro.

- pcFilename: caminho do arquivo do terminal que se deseja receber.

tc_sendfileptr

int __stdcall tc_sendfileptr(int ID, char * cpFilename, char * cpFiledata, unsigned long filesize);
Envia um arquivo para o terminal, o evento “RIDvSendFile” será gerado indicando se arquivo foi gravado com sucesso ou não.
Retorna: >0 para sucesso, <1 para erro.

- cpFilename: Caminho do arquivo que será escrito no terminal.
- cpFiledata: Array de bytes com conteúdo do arquivo.

- filesize: Numero de bytes que será escritos no arquivo.

tc_sendfile

int __stdcall tc_sendfile(int ID, char *cpSrcFilename, char *cpDestFilename);
Envia um arquivo para o terminal, o evento “RIDvSendFile” será gerado indicando se arquivo foi gravado com sucesso ou não.
Retorna: >0 para sucesso, <1 para erro.

- cpSrcFilename: Caminho de origem do arquivo (PC).
- cpDestFilename: Caminho de destino do arquivo (Terminal).

tc_requid

int __stdcall tc_requid(int ID);
Requisita o MAC Address e o nome do terminal, o evento “RIDvGetUID” será gerado indicando sucesso ou não da operação, em caso de sucesso a função tc_getuid deve ser chamada para receber os dados.
Retorna: >0 para sucesso, <1 para erro.

tc_reqsc

int __stdcall tc_reqsc(int ID);
Requisita a identificação de segurança do terminal, o evento “RIDSecretCode” será gerado indicando sucesso ou não da operação, em caso de sucesso a função tc_getsc deve ser chamada para receber os dados.
Retorna: >0 para sucesso, <1 para erro.

tc_reqimageupdate

int __stdcall tc_reqimageupdate(int ID);
Requisita configuração de atualização do terminal (IDGetConfigAdvServer). Para receber os dados, utilize a função tc_getimageupdateconfig.

tc_sendimageupdate

int __stdcall tc_sendimageupdate(int ID, ARG_IMGUPD *imgupd);
Envia configuração de atualização de imagens para o terminal (IDSetConfigAdvServer).
Retorna: >0 para sucesso, <1 para erro.

```
typedef struct
{
long imgupdenable; // 0 = desabilita, 1 = habilita
char imgupdserver[100]; // endereço do servidor de atualização de imagens
long imgupddeftime; // tempo padrão de atualização
long imgupdcurtime; // tempo atual de atualiação
} ARG_IMGUPD;
```

tc_sendupdateimages

int __stdcall tc_sendupdateimages(int ID);
Terminal realiza a atualização das imagens no servidor configurado.
Retorna: >0 para sucesso, <1 para erro.

SendConfigWifi

int __stdcall SendConfigWifi(int ID, DWORD dwHabWifi, DWORD dwModo, char* pcSSID, DWORD dwCanal, DWORD dwCripto, char* pcKey);

Envia configuração do WiFi para o terminal.

Retorna: >0 para sucesso, <1 para erro.

- ID : indentificação do terminal
- HabWifi : 0 desabilita Wifi, 1 habilita wifi.
- Modo : 0 paramodo infraestrutura, 1 para modo ad-hoc.
- SSID : SSID da rede sem fio.
- Canal : Canal da rede sem fio (somente para ad-hoc)
- Cripto : 0 sem segurança, 1 WEP, 2 WPA, 3 WPA2.
- Key : Chave da rede sem fio se tiver menos do que 64 caracteres, deve ser finalizada com

tc_ sendfileStatus

int __stdcall tc_sendfileStatus(int ID, char *pcSrcFilename, char *pcDestFilename, int* piStatus);

Envia arquivo para o terminal e atualiza a variável piStatus conforme o arquivo é enviado, o evento “RIDvSendFile” será gerado indicando se arquivo foi gravado com sucesso ou não.

Retorna: >0 para sucesso, <1 para erro.

- ID : Identificação do terminal
- pcSrcfilename : Caminho do arquivo a ser enviado
- pcDestFilename : Caminho onde o arquivo vai ser armazenado. ex1: INT_MEM\imagem.bmp ou SDCARD0\imagem.bmp
- piStatus : Ponteiro que será atualizado de acordo com o envio do arquivo, vale de -1 á 100.

tc_ sendpresencesensorconf

int __stdcall tc_sendpresencesensorconf(int ID, char *pcFilename);

Envia a lista de configuração do playlist do sensor de presença, ou seja, se o sensor de presença estiver habilitado e um indivíduo passar em frente ao terminal este playlist será exibido. Pode conter imagens, vídeos, e áudios. O evento “RIDvSendFile” será gerado indicando se arquivo foi gravado com sucesso ou não.

Obs. As mídias devem ser enviadas previamente para o terminal via funçãos; tc_sendfile ou tc_sendfileptr ou tc_sendfileStatus.

Retorna: >0 para sucesso, <1 para erro.

- ID : Identificação do terminal.
- pcFilename : Caminho completo do arquivo.

tc_ sendmediaconf

int __stdcall tc_sendmediaconf(int ID, char *pcFilename);

Envia a lista de configuração do playlist de propaganda, esse playlist será exibido continuamente. Pode conter imagens, vídeos e áudios. O evento “RIDvSendFile” será gerado indicando se arquivo foi gravado com sucesso ou não.

Obs. As mídias devem ser enviadas previamente para o terminal via funçãos; tc_sendfile ou tc_sendfileptr ou tc_sendfileStatus.

Retorna: >0 para sucesso, <1 para erro.

- ID : identificação do terminal.
- pcFilename : caminho completo do arquivo.

tc_ sendfontfile

int __stdcall tc_sendfontfile(int ID, char *pcFilename, char *pcFontname)
Envia arquivo de fonte para o terminal. O evento “RIDvSendFile” será gerado indicando se arquivo foi gravado com sucesso ou não.
Retorna: >0 para sucesso, <1 para erro.

- ID : identificação do terminal.
- pcFilename : caminho completo do arquivo.
- pcFontname : nome e extensão da fonte.

tc_ pedeallmedias

int __stdcall tc_pedeallmedias(int ID);
Pede para o terminal especificado pelo ID o arquivo "allMedias.conf", que contem a lista de todas a mídias armazenadas no terminal. O evento “RIDvRecvFile” será gerado indicando o sucesso da operação ou não. Após o recebimento o arquivo será gravado no diretório "Temp" que será criado dentro diretório indicado pela função tc_setWorkDir ou no diretório atual do executável da aplicação, caso não seja indicado o diretório de trabalho. Para obter a lista de mídias a aplicação deverá pedir a lista de mídias através a função tc_getallmedias.
Retorna: >0 para sucesso, <1 para erro.

tc_ pedesensorstatus

int __stdcall tc_pedesensorstatus(int ID);
Pede para o terminal especificado o status do sensor de presença. O evento “RIDGetSensorStatus” será gerado indicando o sucesso da operação ou não. Em caso de sucesso a função tc_getsensorstatus deve ser chamada para ler o status.
Retorna: >0 para sucesso, <1 para erro.

tc_ setsensor

int __stdcall tc_setsensor(int ID, DWORD dwStatus);
Envia configuração de status do sensor de presença. O evento “RIDSetSensor” será gerado indicando o sucesso da operação ou não.
Retorna: >0 para sucesso, <1 para erro.

- dwStatus: 0x01 (1) – Habilita sensor, 0x00 (0) – Desabilita sensor.

tc_ ShowLocalMedia

int __stdcall tc_ShowLocalMedia(int ID, char * cpFileName);
Executa mídia armazenada no terminal. O evento “IDShowLocalMedia” será gerado indicando o sucesso da operação ou não.
Retorna: >0 para sucesso, <1 para erro.
Exemplos: INT_MEM\imagem.bmp ou SDCARD1\imagem.bmp

- cpFileName: Caminho do arquivo a ser exibido (Terminal).

tc_ DeleteLocalMedia

int __stdcall tc_DeleteLocalMedia(int ID, char * cpFilename);
Deleta mídia armazenada no terminal. O evento “RIDDeleteLocalMedia” será gerado indicando o sucesso da operação ou não.
Retorna: >0 para sucesso, <1 para erro.
Exemplos: INT_MEM\imagem.bmp ou SDCARD0\imagem.bmp

- cpFilename: Caminho do arquivo a ser excluido (Terminal).

tc_ CleanInternalMem

int __stdcall tc_CleanInternalMem(int ID);
Apaga todas as mídias armazenadas na memória interna do terminal. O evento “RIDCleanInternalMem” será gerado indicando o sucesso da operação ou não.
Retorna: >0 para sucesso, <1 para erro.

tc_ CleanExternalMem

int __stdcall tc_CleanExternalMem(int ID);
Apaga todas as mídias armazenadas na memória externa do terminal. O evento “RIDCleanExternalMem” será gerado indicando o sucesso da operação ou não.
Retorna: >0 para sucesso, <1 para erro.

tc_ setaudio

int __stdcall tc_setaudio(int ID, DWORD dwEnable);
Habilita ou desabilita o áudio do Terminal. O evento “RIDSetAudio” será gerado indicando o sucesso da operação ou não.
Retorna: >0 para sucesso, <1 para erro.

- arg: 0x01 – Habilita áudio, 0x00 – Desabilita áudio.

tc_ pedeaudiostatus

int __stdcall tc_pedeaudiostatus(int ID);
Pede para o terminal o status do áudio. O evento “RIDGetAudioStatus” será gerado indicando o sucesso da operação ou não. Em caso de sucesso a função tc_getaudiostatus deve ser chamada para ler o status.
Retorna: >0 para sucesso, <1 para erro.

tc_ setvolume

int __stdcall tc_setvolume(int ID, DWORD dwVolume);
Ajusta o valor do volume dos alto-falantes do terminal de consulta. O evento “RIDSetVolume” será gerado indicando o sucesso da operação ou não.
Retorna: >0 para sucesso, <1 para erro.

- dwVolume: Valor do volume na faixa de 0 a 100

tc_ pedevolume

int __stdcall tc_pedevolume(int ID);
Pede para o terminal o volume do áudio. O evento “RIDGetVolume” será gerado indicando o sucesso da operação ou não. Em caso de sucesso a função tc_getvolume deve ser chamada para ler o volume.
Retorna: >0 para sucesso, <1 para erro.

tc_ setbrightness

int __stdcall tc_setbrightness(int ID, DWORD dwBrightness);
Ajusta o valor do brilho do display do terminal de consulta.
Retorna: >0 para sucesso, <1 para erro.

- dwBrightness: Valor do brilho na faixa de 0 a 100.

tc_pedebrightness

int __stdcall tc_pedebrightness(int ID);
Pede para o terminal o brilho. O evento “RIDGetBrightness” será gerado indicando o sucesso da operação ou não. Em caso de sucesso a função tc_getbrightness deve ser chamada para ler o brilho.
Retorna: >0 para sucesso, <1 para erro.

tc_savemedias

int __stdcall tc_savemedias(char acMedias[][256], int quant char* pcDestDir);
Cria o arquivo de configuração de mídias no formato apropriado e salva em diretório especificado.
Retorna: >0 para sucesso, <1 para erro.

- acMedias: lista de mídias na ordem de exibição no seguinte formato:

| nome da imagem | tempo | qtd de repetições da imagem|.

- quant: quantidade de mídias.
- pcDestDir: pasta de destino para salvar as mídias no terminal

Recebendo dados dos terminais

tc_getserial

int __stdcall tc_getserial(int ID, int *sercom, char *buf);
Recebe dados da serial enviados pelo terminal (IdvReadSerialA/B).
Retorna: número de bytes lidos.

- sercom: 0 para COM 1, 1 para COM 2.
- buf: Array de bytes contendo dados lidos da serial.

tc_sergetstatus

int __stdcall tc_sergetstatus(int ID, int sercom, ARG_SERIAL_STS *serialsts);
Recebe o estado da porta serial do terminal (IDvGetStatus).
Status: Bit0: indefinido, Bit1: DCD; Bit2: DSR; Bit3: CTS.

```
typedef struct
{
BYTE aserial; // 0 = COM 1, 1 = COM 2
BYTE status; // Estado de controle
}ARG_SERIAL_STS
```

tc_getmag

int __stdcall tc_getmag(int ID, char *buf1, char *buf2);
Recebe dados do leitor de cartão magnético enviados pelo terminal (IDbReadBuffLEC).
Retorna: 0 se buffer vazio, 1 se conseguiu pegar dados.

- buf1,buf2: array de bytes contendo dados ligos das trilhas 1,2.

tc_getconfig

int __stdcall tc_getconfig(int ID, ARG_SETUP_TCP *config);
Recebe configuração do terminal requisitada pela função tc_reqconfig.
Retorna: 0 se buffer vazio, 1 se conseguiu pegar dados.

- config: estrutura contendo configuração do terminal.

tc_getfile

int __stdcall tc_getfile (int ID, char *pcFilename, int *pcFilesize, char *pcFileData);
Recebe arquivo requisitado pela função tc_reqfile.
Retorna: -1 se ID inválido, 0 se arquivo não existir, >1 se conseguiu pegar dados.

- pcFilename: Será preenchido como o nome do arquivo requisitado.
- pcFilesize: Será preenchido como o tamanho do arquivo requisitado.
- pcFileData: Array de bytes que será preenchido como o conteúdo o arquivo.

tc_getuid

int __stdcall tc_getuid(int ID, char *macaddr, char *tcname);
Recebe o MAC Address e o nome do terminal requisitado pela função tc_requid.
Retorna: 0 se buffer vazio, 1 se conseguiu pegar dados.

- macaddr: array de bytes contendo o mac address (6 bytes).
- tcname: array de bytes contendo o nome do terminal.

tc_getsc

int __stdcall tc_getsc(int ID, char *sc);
Recebe a identificação de segurança requisitada pela função tc_reqsc.
Retorna: 0 se buffer vazio, 1 se conseguiu pegar dados.

- sc: array de bytes contendo as informações de segurança do terminal.

tc_getident

unsigned long __stdcall tc_getident(int ID);
Recebe identificação do terminal.
Retorna: 4 bytes com a identificação do terminal.

tc_ getimageupdateconfig

int __stdcall tc_getimageupdateconfig(int ID,ARG_IMGUPD *imgupd);
Recebe a configuração de atualização de imagens requisitada pela função tc_reqimageupdate.
Retorna: 0 se buffer vazio, 1 se conseguiu pegar dados.

tc_gettermconectados

TTABTERM* __stdcall tc_gettermconectados(TTABTERM *pTabTerm);
Retorna tabela com todos os terminais conectados.
Retorna ponteiro para a estrutura TTABTERM.

- pTabTerm : Ponteiro para a tabela que receberá a lista de terminais conectados.

```
typedef struct
{
char TabName[256][34]; // Tabela de nomes
int TabSock[256]; // Tabela de identificação dos terminais
DWORD TabIP[256]; // Tabela de IP dos terminais
int Tipo[256]; // Tabela de tipos dos terminais (506 ou 504)
int NumSockConec; // Número de terminais conectados
}TTABTERM;
```

tc_ TemResposta

bool __stdcall tc_TemResposta(int* ID, byte* comando, DWORD* Resp);
Verifica se existe evento gerado pelo terminal.
Retorna: false se não tem evento, true se tem evento

- ID : identificação do terminal que enviou o evento.
- comando : identificação do evento, os eventos estão listados abaixo
- Resp : resposta do comando.
-

Segue a tabela de eventos:

RIDShowLocalMedia	0xA7
RIDSetSensor	0xA9
RIDGetSensorStatus	0xAB
RIDSetAudio	0xAD
RIDGetAudioStatus	0xAF
RIDSetVolume	0xB1
RIDGetVolume	0xB3
RIDSetBrightness	0xB5
RIDGetBrightness	0xB7
RIDvRecvFile	0x62
RIDDeleteLocalMedia	0xB9
RIDCleanInternalMem	0xBB
RIDCleanExternalMem	0xBD
RIDvGetTimeExhib	0x2A

tc_ GetResposta

DWORD __stdcall tc_GetResposta(int ID);
Retorna a ultima resposta do ultimo evento do terminal especificado.

tc_ getallmedias

void __stdcall tc_getallmedias(int ID, char acMediasList [ ][256], int quant);
Recebe lista de mídias requisitado pela função tc_pedeallmedias. Retorna um array de strings contendo todas as mídias armazenadas no terminal.

- acMediasList: Array de strings que receberá a lista de mídias.
- quant: Quantidade de posições disponíveis no Array.

tc_ getsensorstatus

DWORD __stdcall tc_getsensorstatus(int ID);
Recebe status do sensor de presença requisitado pela função tc_pedesensorstatus.
ID : identificação do terminal.
Retorna: status do sensor de presença do terminal especificado
MSB: Sensor desabilitado: 0x00
Sensor habilitado: 0x01
LSB: Sensor não atuado: 0x00
Sensor atuado: 0x01
Retorno: Como retorno, o terminal responde no LSB o estado do sensor (atuado/ não atuado) e no byte seguinte, à esquerda, a indicação de que o mesmo está ou não habilitado.

tc_ getaudiostatus

DWORD __stdcall tc_getaudiostatus(int ID);
Recebe status do audio requisitado pela função tc_pedeaudiostatus.
Retorno : 0x00 - Áudio desabilitado;
0x01 - Áudio habilitado.
Retorna: status do áudio

tc_ getvolume

DWORD __stdcall tc_getvolume(int ID);
Recebe o volume do audio requisitado pela função tc_pedevolume.
Retorna o volume do áudio na faixa de 0 a 100.
Retorna: <1 houve erro, 0 a 100 se comando realizado com sucesso.

tc_ getbrightness

DWORD __stdcall tc_getbrightness(int ID);
Recebe o brilho da tela requisitado pela função tc_pedebrightness.
Retorna o valor do brilho na faixa de 0 a 100.
Retorna: <1 houve erro, 0 a 100 se comando realizado com sucesso.

Compatibilidade com a versão 1.0

Para manter compatibilidade com a DLL 1.0, foram criadas funções que fazer a interface com a DLL 2.0. Porém, é preciso observar que as chamadas para tais funções sofreram pequenas alterações, como a diretiva "__stdcall", que foi implementada para haver maior compatibilidade da DLL entre diversas linguagens de programação e algumas funções que tinham como parâmetro, array de bytes, foram substituídos por ponteiro de bytes, portanto, para utilizar a DLL 2.0 nos servidores que utilizam a 1.0 é preciso recompilar o programa, reajustando as chamadas de funções. Segue abaixo, protótipo das funções.

Funções em C

- void __stdcall GetIPFromHost(char *ret);
- void __stdcall TCinet_ntoa(DWORD nIP, char *buf);
- DWORD __stdcall TCinet_addr(char *buf);
- TTABSOCK __stdcall GetTabConectados(void);
- void __stdcall EnviaVivo(int ID);
- void __stdcall EnviaSempreVivo(int ID);
- void __stdcall SendConfig(int ID, char *ClienteAdd, char *ServerAdd, char *NetMaskAdd, char *GatewayAdd, char *NameserverAdd, char *TCNameAdd, WORD PortsvAdd, char *FTPServerAdd, char *FTPUserAdd, char *FTPPAssAdd, DWORD bDHCP, DWORD AutoFind);
- void __stdcall PedeConfig(int ID);
- void __stdcall EnviaTexto(int ID, WORD Posx, WORD Posy, char *Text2Show, char *Fonte, WORD TextColor, WORD BgColor);
- void __stdcall ClearDisplay(int ID, int Color);
- void __stdcall EnviaImagem(int ID, char *APaleta, char *tAImagem);
- void __stdcall EnviaImagemDoArquivo(int ID, char *filename);
- void __stdcall SetaTempoExib(int ID, WORD TempoEx);
- void __stdcall PedeTempoExib(int ID);
- void __stdcall UpdateSoft(int ID);
- void __stdcall StopAdv(int ID);
- void __stdcall DeleteAdv(int ID);

- void __stdcall HabilitaTeclado(int ID, DWORD HabSim);
- void __stdcall PedeHabilitaTeclado(int ID);
- void __stdcall AtualizaAdv(int ID, DWORD DoReload);
- void __stdcall HabilitaLEC(int ID, DWORD HabSim);
- void __stdcall PedeHabilitaLEC(int ID);
- void __stdcall AbreSerialCOM(int ID, BYTE SerCOM, BYTE SimAbre, int Baud, BYTE parity, BYTE databits);
- void __stdcall EscreveSerial(int ID, BYTE SerCOM, int TamBuf, char *sbuf);
- void __stdcall EnviaArquivo(int ID, char *localfilename, char *destfilename);
- bool __stdcall GetSerial(int *ID, int *Porta, char *buffer, int *Nbr);
- bool __stdcall GetLEC(int *ID, char *ptrilha, int *errocode);
- bool __stdcall Get_KBD_char(int *ID, char *caracter);
- bool __stdcall GetConfig(int *pID, char *pmyip, char *pserverip, char *pnetmask, char *pgateway, char *pnameserver, char *ptcname, WORD *pport, char *pupdserv, char *pupduser, char *pupdpass,char *pdynamicip, char *pfindserverchar);
- void __stdcall StartServerTC504(void);
- void __stdcall CloseTC504(void);
- void __stdcall InitTC504(void);

Funções em Pascal

- procedure GetIPFromHost(var res : byte); far; stdcall; external 'sc504.dll';
- procedure TCinet_ntoa(nIP : DWORD; var buf: byte); far; stdcall; external 'sc504.dll';
- function TCinet_addr(var buf: byte): DWORD; far; stdcall; external 'sc504.dll';
- procedure EnviaVivo(ID: Integer); far; stdcall; external 'sc504.dll';
- function GetConfig(var pID: integer; var pmyip: byte; var pserverip: byte; var pnetmask: byte; var pgateway: byte; var pnameserver: byte; var ptcname: byte; var pport : WORD; var pupdserv: byte; var pupduser: byte; var pupdpass: byte; var pdynamicip: byte; var pfindserver: byte): boolean; far; stdcall; external 'sc504.dll';
- procedure SendConfig(ID: Integer; var ClienteAdd: byte; var ServerAdd : byte; var NetMaskAdd: byte; var GatewayAdd: byte; var NameserverAdd: byte; var TCNameAdd: byte; PortsvAdd: WORD; var FTPServerAdd: byte; var FTPUserAdd: byte; var FTPPAssAdd: byte; bDHCP: DWORD; AutoFind: DWORD); far; stdcall; external 'sc504.dll';
- procedure PedeConfig(ID: Integer); far; stdcall; external 'sc504.dll'; procedure EnviaTexto(ID: Integer; Posx: Word; Posy: Word; var Text2Show: byte; var Fonte: byte; TextColor: Word; BgColor: Word); far; stdcall; external 'sc504.dll';
- procedure ClearDisplay(ID: Integer; Color : Integer); far; stdcall; external 'sc504.dll';
- procedure EnviaImagem(ID: Integer; var APaleta: byte; var AImagem: byte); stdcall; external 'sc504.dll';
- procedure SetaTempoExib(ID: Integer; TempoEx: word); far; stdcall; external 'sc504.dll';
- procedure PedeTempoExib(ID: Integer); far; stdcall; external 'sc504.dll';
- procedure UpdateSoft(ID: Integer); far; stdcall; external 'sc504.dll';
- procedure StopAdv(ID: Integer); far; stdcall; external 'sc504.dll';
- procedure DeleteAdv(ID: Integer); far; stdcall; external 'sc504.dll';
- procedure HabilitaTeclado(ID: Integer; HabSim: DWORD); far; stdcall; external 'sc504.dll';
- procedure PedeHabilitaTeclado(ID: Integer); far; stdcall; external 'sc504.dll';
- procedure AtualizaAdv(ID: Integer; DoReload: DWORD); far; stdcall; external 'sc504.dll';
- procedure HabilitaLEC(ID: Integer; HabSim: DWORD); far; stdcall; external 'sc504.dll';
- procedure PedeHabilitaLEC(ID: Integer); far; stdcall; external 'sc504.dll';
- procedure AbreSerialCOM(ID: Integer; SerCOM: byte; SimAbre: byte; Baud: integer; parity: byte; databits: byte); far; stdcall; external 'sc504.dll';
- procedure EscreveSerial(ID: Integer; SerCOM : byte; TamBuf: integer; var sbuf: byte); far; stdcall; external 'sc504.dll';
- procedure EnviaArquivo(ID: Integer; var localfilename: byte; var destfilename: byte); far; stdcall; external 'sc504.dll';
- procedure StartServerTC504; far; stdcall; external 'sc504.dll';

- procedure InitTC504; far; stdcall; external 'sc504.dll';
- function GetTabConectados(nada: integer): TTABSOCK; far; stdcall; external 'sc504.dll';
- function GetSerial(var ID: integer; var Porta: integer;var buffer: byte; var Nbr: integer): boolean; far; stdcall; external 'sc504.dll';
- function GetLEC(var ID: integer; var ptrilha: byte; var errcode: integer): boolean; far; stdcall; external 'sc504.dll';

function Get_KBD_char(var ID: integer; var caracter: byte): boolean; far; stdcall; external 'sc504.dll';

A troca de mensagens do programa principal com a DLL

Para aumentar a agilidade de troca de informação da DLL com a aplicação que usa a DLL, e evitar processamento desnecessário, foi implementada a troca de mensagens. A DLL envia mensagens para a aplicação sempre que ocorrer em evento entre o terminal e a DLL. Para facilitar o entendimento, utilizaremos o C++ Builder® como exemplo.

Para receber estas mensagens, o servidor deve chamar a função da DLL tc_startserver da seguinte forma:

```
#define COMUNICATION_MSG WM_USER + 1
#define CONNECT_MSG WM_USER + 2
tc_startserver(Form1->Handle,CONNECT_MSG,COMUNICATION_MSG);
```

Onde Form1 corresponde ao formulário (janela) principal, CONNECT_MSG corresponde à mensagem que o servidor enviará quando um terminal conectar/desconectar e COMUNICATION_MSG corresponde à mensagem que o servidor enviará quando um terminal enviar dados.

Devemos “redefinir” a função WndProc do formulário para prodermos receber as mensagem. Para isso devemos seguir os seguintes passos:

1) No arquivo de header (geralmente unit1.h) devemos adicionar na classe do formulário, a chamada para a função:

```
...
private: // User declarations
virtual void __fastcall WndProc(Messages::TMessage &Message);
...
```

2) Devemos então, “reescrever” esta função (unit1.cpp):

```
void __fastcall TForm1::WndProc(Messages::TMessage &Message)
{
if (Message.Msg == COMUNICATION_MSG)
{
//recebe mensagens enviadas pelo terminal
return;
}
else if (Message.Msg == CONNECT_MSG)
{
//recebe mensagens quando um terminal conectou/desconectou
return;
}
TForm::WndProc(Message); //chama WndProc antiga
}
```

Para saber como tratar as mensagens recebidas, fornecemos o código fonte do servidor para ser tomado como exemplo.

Funções incompatíveis com o TC 504

As seguintes funções da DLL SC504 da tabela abaixo apenas são compatíveis com o terminal de consulta TC 506 Mídia.

Segue a tabela:

tc_sendfileStatus
tc_sendpresencesensorconf
tc_sendmediaconf
tc_sendfontfile
tc_pedeallmedias
tc_pedesensorstatus
tc_setsensor
tc_ShowLocalMedia
tc_DeleteLocalMedia
tc_CleanInternalMem
tc_CleanExternalMem
tc_setaudio
tc_pedeaudiostatus
tc_setvolume
tc_pedevolume
tc_setbrightness
tc_pedebrightness
tc_savemedias
tc_TemResposta
tc_GetResposta
tc_getallmedias
tc_getsensorstatus
tc_getaudiostatus
tc_getvolume
tc_getbrightness