minsait

An Indra company

EDITRAN/TR

Windows/Unix
Manual de referencia

mayo de 2019

ecosystems
communications

1. INTRODUCCIÓN.

El módulo EDItran/TR permite a los usuarios y desarrolladores de aplicaciones acceder a los servicios de envío y recepción de preguntas y respuestas de la plataforma EDItran.

2. INTERFAZ DE PROGRAMACIÓN.

2.1. DESCRIPCION

El establecimiento de la sesión se realizará a petición del operador de EDItran/P.

Una vez establecida, la aplicación cliente enviará una pregunta indicando la sesión.

La respuesta la recibe el mismo proceso que envió la pregunta.

Si el proceso de pregunta/respuesta no tuviera éxito, la aplicación cliente enviará de nuevo la pregunta.

En el extremo servidor, se reciben las preguntas de todas las sesiones. La aplicación servidora recibirá las preguntas junto con un identificador y unos recursos que debe guardar para adjuntarlos a la respuesta de las mismas.

EDItran/TR no detecta la existencia de la conexión, es responsabilidad del usuario el establecimiento de dicha conexión.

2.2. FUNCIONES DE ENVIO Y RECEPCION

```
int      SendRequest (char *Sesion,
             char *Id,
             int *pResultado,
             int Lenguaje,
             char *pData,
             int DataLen);

int      ReceiveRequest (char *Sesion,
           char *Id,
           int *pResultado,
           char *Recursos,
           int Lenguaje,
           char *pData,
           int *pDataLen);

int      SendResponse (char *Sesion,
              char *Id,
              int *pResultado,
              char *Recursos,
              int Lenguaje,
              char *pData,
              int DataLen);

int      ReceiveResponse (char *Sesion,
            char *Id,
            int *pResultado,
            int Lenguaje,
            char *pData,
            int *pDataLen);
```

Descripción

Envía una petición EDItran/TR por la sesión Editran/P dada.

Parámetros

- Sesion — Sesión EDItran/P
- Id — Identificador asignado por la aplicación
- Recursos — Datos necesarios para entregar la respuesta al punto origen de la pregunta
- pResultado — Resultado de la petición

- Lenguaje Indica el juego de caracteres de pData, ALFABETO_BINARIO, ALFABETO_ASCII, ALFABETO_EBCDIC, ALFABETO_BCD
- pData Pregunta o respuesta
- pDataLen Longitud

Valores de retorno

```
T_NO_ERROR_RC        Resultado correcto
T_SNTX_ERROR_RC      Sintaxis incorrecta
T_INIT_ERROR_RC      Sistema EDItran/P no iniciado
T_SEND_ERROR_RC      Error al comunicar con EDItran/P (send)
T_RECV_ERROR_RC      Error al comunicar con EDItran/P (receive)
T_IN_ERROR_RC        Error al leer datos de entrada
T_LICENSE_ERROR_RC   Licencia incorrecta
T_PROTOCOL_ERROR_RC  Error de secuencia de señales
T_NO_PERFIL_ERROR_RC Error al leer el perfil de la sesión
```

2.3. FUNCION GetErrorText

```
char    *GetErrorText (int Error);
```

Descripción

Transforma los valores de retorno de las funciones en texto.

3. COMANDO

DESCRIPCION

```
     editrantr -c <id#> -s <sesion> [-l {aeb}] [-v]
     editrantr -r <serv> [-l {aeb}] [-v]

Siendo:
    -c <id#>     : cliente, identif. del mensaje
    -r <serv>    : servidor
    -s <sesion>  : identificador de la sesion
    -l aeb       : lenguaje ascii, ebcdic, binario
    -v           : modo verboso
```

Como cliente, para una sesión EDItran/P, lee la pregunta de su entrada standard, la envía y espera indefinidamente la respuesta y al recibirla la escribe en su salida standard, y así sucesivamente hasta que recibe la señal SIGINT.

Como servidor, para todas las sesiones EDItran/P, espera indefinidamente la recepción de una pregunta, la escribe en su salida standard, lee la respuesta de su entrada standard y la envía, y así sucesivamente hasta que recibe la señal SIGINT. La pregunta es procesada por **`/usr/bin/sh –c "serv"`** que la lee de su entrada standard y escribe en su salida standard la respuesta.

EJEMPLO

En la parte cliente:

```
editrantr –c 1 –s SESION1 –l a –v
```

En la parte servidora:

```
editrantr –r " " –v
```

4. EJEMPLO DE PROGRAMACIÓN

```
/*****************************************************************************
 *
 *      Departamento : Soluciones de comunicaciones
 *
 *      (C) INDRA, S.A   Noviembre, 2002.
 *
 *      <editrant.c>
 *
 *****************************************************************************/

/*---------------------------------------------------------------------------
   System includes
---------------------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <signal.h>
#include <string.h>
#include <ctype.h>

/*---------------------------------------------------------------------------
   Project includes
---------------------------------------------------------------------------*/
#include "alarm/alarm.h"
#include "license/des/license.h"
#include "share/copyrigh.h"
#include "share/cc_depen.h"
#include "editrant/api/t_api.h"
#include "editrant/convert/t_convert.h"

/*---------------------------------------------------------------------------
   Globals
---------------------------------------------------------------------------*/
static char     Version[] = "1.0";
char           *ProgName;

/*---------------------------------------------------------------------------
   Typedefs
---------------------------------------------------------------------------*/
/* Program options */
typedef struct {
    char            Sesion[100];  /* Sesion */
    char            Id[ID_SIZE];  /* Id */
    char            Server[100];  /* Server */
    char            Type; /* Cliente/servidor */
    int             bType;/* Opcion tipo */
    int             bSesion; /* Opcion sesion */
    int             Lenguaje;   /* Lenguaje */
    int             bVerbose;   /* Verboso */
} opts_t;

/*---------------------------------------------------------------------------
        quit
---------------------------------------------------------------------------*/
void
quit (int code)
{
    char            buffer[100];

    sprintf (buffer, "Received signal %d\n", code);
    SignalAlarm (buffer);
    exit (code);
}

/*---------------------------------------------------------------------------
   usage
```

```
------------------------------------------------------------------------------*/
void
usage (char *format,...)
{
    va_list         ap;

    va_start (ap, format);
    fprintf (stderr, "%s: ", ProgName);
    vfprintf (stderr, format, ap);
    fprintf (stderr,
        "\n\n%s. Comando de envio de mensajes de EDItran/TR\n",
        ProgName);
    Copyright (Version);

    fprintf (stderr,
        "Uso: %s -c <id#> -s <sesion> [-l {aeb}] [-v]   \n"
        "     %s -r <serv> [-l {aeb}] [-v]      \n"
        "Siendo:                                        \n"
        "    -c <id#>     : cliente, identif. del mensaje \n"
        "    -r <serv>    : servidor      \n"
        "    -s <sesion>  : identificador de la sesion     \n"
        "    -l aeb       : lenguaje ascii, ebcdic, binario\n"
        "    -v           : modo verboso        \n",
        ProgName, ProgName);
}

/*----------------------------------------------------------------------------
   GetResultText
------------------------------------------------------------------------------*/
void
GetResultText (int Resultado)
{
    fprintf (stderr, "Resultado - ");
    switch (Resultado) {
    case EMR_NO_ERROR:
   fprintf (stderr, "Correcto\n");
   break;
    case EMR_ERROR_VAR_VERSION:
   fprintf (stderr, "Error en variable version\n");
   break;
    case EMR_ERROR_VAR_RECURSOS:
   fprintf (stderr, "Error en variable recursos\n");
   break;
    case EMR_ERROR_VAR_ID:
   fprintf (stderr, "Error en variable identificador\n");
   break;
    case EMR_ERROR_VAR_DATOS:
   fprintf (stderr, "Error en variable datos\n");
   break;
    case EMR_ERROR_VAR_UNKNOWN:
   fprintf (stderr, "Error en variable desconocida\n");
   break;
    default:
   fprintf (stderr, "Error desconocido\n");
    }
}

/*----------------------------------------------------------------------------
   main
------------------------------------------------------------------------------*/
int
main (int argc, char *argv[])
{
    opts_t          opts;
    int             result = T_NO_ERROR_RC,
                    Resultado = EMR_NO_ERROR,
                    Longitud;
    char           *pstr,
                   *BaseDir,
                    Texto[DATA_SIZE],
```

```
                    Recursos[REC_SIZE],
                    Sesion[100],
                    Id[ID_SIZE];


  /*****************************************************
   Preambulo
  *****************************************************/
  if ((ProgName = (char *) strrchr (argv[0], SLASH)) == NULL)
ProgName = argv[0];
  else
++ProgName;

  BaseDir = (char *) strrchr (argv[0], '/');
  if (BaseDir != NULL) {
*BaseDir = '\0';
if (chdir (argv[0]) < 0)
    usage ("%s: directorio invalido", argv[0]);
  }
  if (!FeatureNumberIsOk ("editrantr", 1)) {
fprintf (stderr, "Licencia incorrecta.\n");
return (T_SNTX_ERROR_RC);
  }
  /*****************************************************
   Obtencion de opciones
  *****************************************************/
  argv++;
  argc++;

  memset (&opts, 0, sizeof (opts));

  while (*argv) {
pstr = *argv;

if (*pstr == '-') {  /* it's an option */
    pstr++;

    switch (tolower (*pstr)) {
    case 's':   /* sesion */
   argv++;
   if (*argv) {
       pstr = *argv;
       strcpy (opts.Sesion, pstr);
       opts.bSesion = TRUE;
   } else {
       usage ("Error - falta sesion");
       return (T_SNTX_ERROR_RC);
   }
   break;

    case 'c':   /* Cliente  */
   opts.Type = 'C';
   opts.bType = TRUE;
   argv++;
   if (*argv) {
       pstr = *argv;
       strcpy (opts.Id, pstr);
   }
   break;

    case 'r':   /* Servidor */
   opts.Type = 'S';
   opts.bType = TRUE;
   argv++;
   if (*argv) {
       pstr = *argv;
       strcpy (opts.Server, pstr);
   }
   break;
```

```
      case 'l':   /* Lenguaje */
    argv++;
    if (*argv) {
        pstr = *argv;
    } else {
        usage ("Error - falta lenguaje");
        return (T_SNTX_ERROR_RC);
    }
    switch (tolower (*pstr)) {
    case 'a':
        opts.Lenguaje = ALFABETO_ASCII;
        break;
    case 'e':
        opts.Lenguaje = ALFABETO_EBCDIC;
        break;
    case 'b':
        opts.Lenguaje = ALFABETO_BINARIO;
        break;
    default:
        usage ("Error - lenguaje incorrecto");
    }
    break;

      case 'v':   /* Verboso */
    opts.bVerbose = TRUE;
    break;

     default:
    usage ("Opcion ilegal -- %c", tolower (*pstr));
    return (T_SNTX_ERROR_RC);
      }
}
argv++;
  }

  /*****************************************************
   Validacion de opciones
  *****************************************************/
  if (!opts.bType) {
usage ("Error : falta cliente/servidor");
return (T_SNTX_ERROR_RC);
  }
  if (opts.Type == 'S') {
if (opts.bSesion) {
    usage ("Error : sobra sesion");
    return (T_SNTX_ERROR_RC);
}
  } else if (!opts.bSesion) {
usage ("Error : falta sesion");
return (T_SNTX_ERROR_RC);
  }
  /* Captura de señales */
  signal (SIGABRT, (SIG_PF) quit);
  signal (SIGFPE, (SIG_PF) quit);
  signal (SIGILL, (SIG_PF) quit);
  signal (SIGINT, (SIG_PF) quit);
  signal (SIGSEGV, (SIG_PF) quit);
  signal (SIGTERM, (SIG_PF) quit);

  /*****************************************************
   CLIENTE
  *****************************************************/
  if (opts.Type == 'C') {

if (opts.bVerbose)
    fprintf (stderr, "Modo de operacion cliente ...\n");

while (result == T_NO_ERROR_RC) {

    /*************************************************
```

```
    Lee y envia la peticion
     ***************************************************/
     gets (Texto);
     Longitud = strlen (Texto);

     result = SendRequest (opts.Sesion,
           opts.Id,
           &Resultado,
           opts.Lenguaje,
           Texto,
           Longitud);

     /***************************************************
    Recibe y escribe la respuesta
     ***************************************************/
     while (result == T_NO_ERROR_RC) {
    result = ReceiveResponse (Sesion,
              Id,
              &Resultado,
              opts.Lenguaje,
              Texto,
              &Longitud);

    if (memcmp (opts.Sesion,
           Sesion,
           min (strlen (opts.Sesion), strlen (Sesion))))
        fprintf (stderr,
       "Error : recibida sesion diferente a la enviada.\n");
    if (memcmp (opts.Id,
           Id,
           min (strlen (opts.Id), strlen (Id))))
        fprintf (stderr, "Error : recibido id diferente al enviado.\n");

    if (Longitud) {
        Texto[Longitud] = 0;
        puts (Texto);
    } else
        break;
     }
     fflush (stdout);

     if (opts.bVerbose && Resultado) {
    GetResultText (Resultado);
     }
}
  }
  /***************************************************
   SERVIDOR
  ***************************************************/
  if (opts.Type == 'S') {
char            Buffer[200];
FILE           *dpIn = NULL;

if (opts.bVerbose) {
    fprintf (stderr, "Modo de operacion servidor ...\n");
    fprintf (stderr, "Servidor : /usr/bin/sh -c %s\n", opts.Server);
}
while (result == T_NO_ERROR_RC) {

    /***************************************************
    Recibe y ejecuta la peticion
     ***************************************************/
     result = ReceiveRequest (Sesion,
              Id,
              &Resultado,
              Recursos,
              opts.Lenguaje,
              Texto,
              &Longitud);
```

```
      Texto[Longitud] = 0;
      sprintf (Buffer, "%s %s", opts.Server, Texto);
      dpIn = popen (Buffer, "r");
      if (dpIn == NULL) {
    fprintf (stderr, "Error al lanzar la peticion %s\n", Buffer);
    pclose (dpIn);
    break;
      }
      /*****************************************************
    Lee y envia la respuesta
      *****************************************************/
      while (result == T_NO_ERROR_RC) {
    if (fgets (Texto, DATA_SIZE, dpIn))
        Longitud = strlen (Texto) - 1;
    else
        Longitud = 0;

    result = SendResponse (Sesion,
                Id,
                &Resultado,
                Recursos,
                opts.Lenguaje,
                Texto,
                Longitud);
    if (!Longitud)
        break;
      }
      pclose (dpIn);
  }
    }
    /*************************************************************
            Finalizacion
    *************************************************************/
    if (opts.bVerbose) {
  fprintf (stderr, "Resultado = %d - %s.\n",
      result, GetErrorText (result));
    }
    return (result);
}
```

5. COMPONENTES DEL API.

- libedi_t.so Librería
- libmisc.so Librería
- libdes.so Librería
- alarm.o Objeto
- t_api.h. Cabecera para programas C
- editrantr.c Ejemplo de programación

minsait

An Indra company

Contacto
editran@indra.es
T +34 91 480 80 80

Avda. de Bruselas 35
28108 Alcobendas,
Madrid, España
T +34 91 480 50 00
F +34 91 480 50 80

www.minsait.com