Sistema control de consolas FlulpyCrea®

NOTAS:

El Servidor de **FlulpyCrea®** puede ser interconectado a **cualquier hardware de control de consola** fabricado por terceros que cuente con el **driver apropiado** de control.

La **interfaz del driver** es de conocimiento público a través de este documento, su existencia es la consecuencia de una de las políticas de **FlulpyCrea®** de no fabricar hardware de control propio, o almenos, si lo fabricáramos en el futuro, la de mantener esta interfaz compatible en un 100% siempre. Es decir, se delega a cualquier fabricante y/o entusiasta del hardware de control de consolas, la tarea de crear un simple driver en formato "**DLL ActiveX**" para interfazar.

Permitir que su hardware de control sea compatible con nuestro sistema, tiene la ventaja de ahorrarle el trabajo de crear un software de gestión dedicado para su hardware, lo cual es una tarea de años si desea realizar un software comercialmente competitivo.

Por último, les recuerdo a los fabricantes de hardware de control, que la suit de gestión de **FlulpyCrea®** se puede usar en lo que respecta al **CyberControl y todo lo relacionado en forma 100% freeware** y puede ser empaquetada junto al producto de hardware fabricado, siempre que el software no sea modificado ni bloqueado en ningún aspecto, es decir, debe poder el sistema ser auto-actualizado online por sí mismo, y el cliente debe poder comprarle una licencia a FlulpyCrea®, en el caso de que quiera utilizar las características adicionales que requieren tal licencia.

EL DRIVER

TIPO / UBICACIÓN / INSTALACIÓN

El driver debe ser un archivo de tipo "**DLL ActiveX**", el cual debe estar instalado/registrado en el sistema operativo Windows de la CPU servidora.

Al estar instalado/registrado, no importa la ubicación del mismo, ya que el sistema operativo Windows sabrá donde se encuentra.

La instalación/registración se hace como con cualquier otra DLL, proceso que realiza el instalador del driver (ver ejemplo).

```
RegSvr32 <my_driver.dll>
```

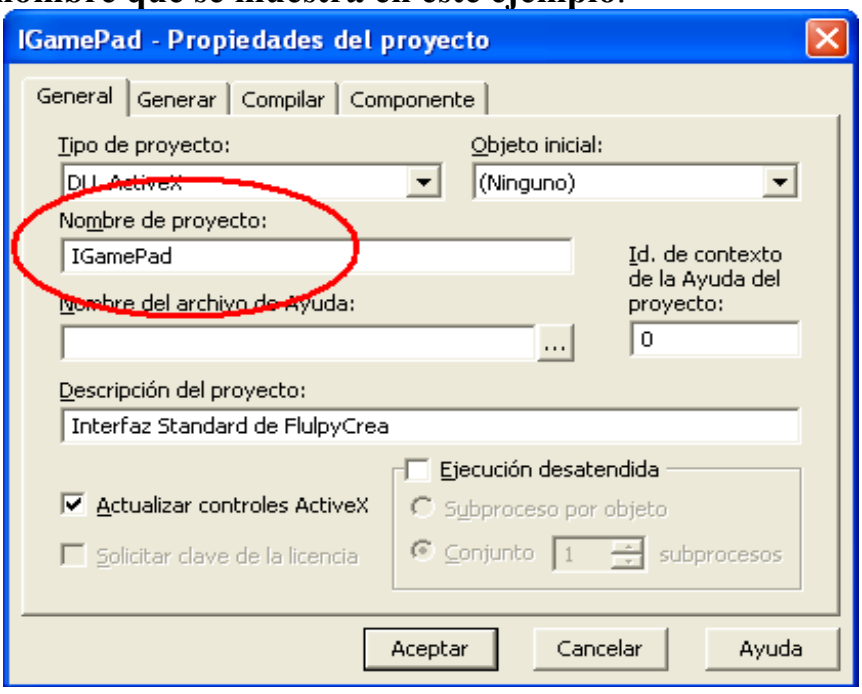
TIPO CLASE

La **DLL ActiveX** deberá tener el nombre de clase (coclass) "**ClassInterfazGamePad**", este nombre identifica a la DLL instalada/registrada, como perteneciente al tipo de driver que el servidor FlulpyCrea® aceptará.

Todos las DLL que se creen bajo esta clase, deberán tener la misma interfaz preestablecida que ya es conocida por el servidor.

NOMBRE LIBRARY

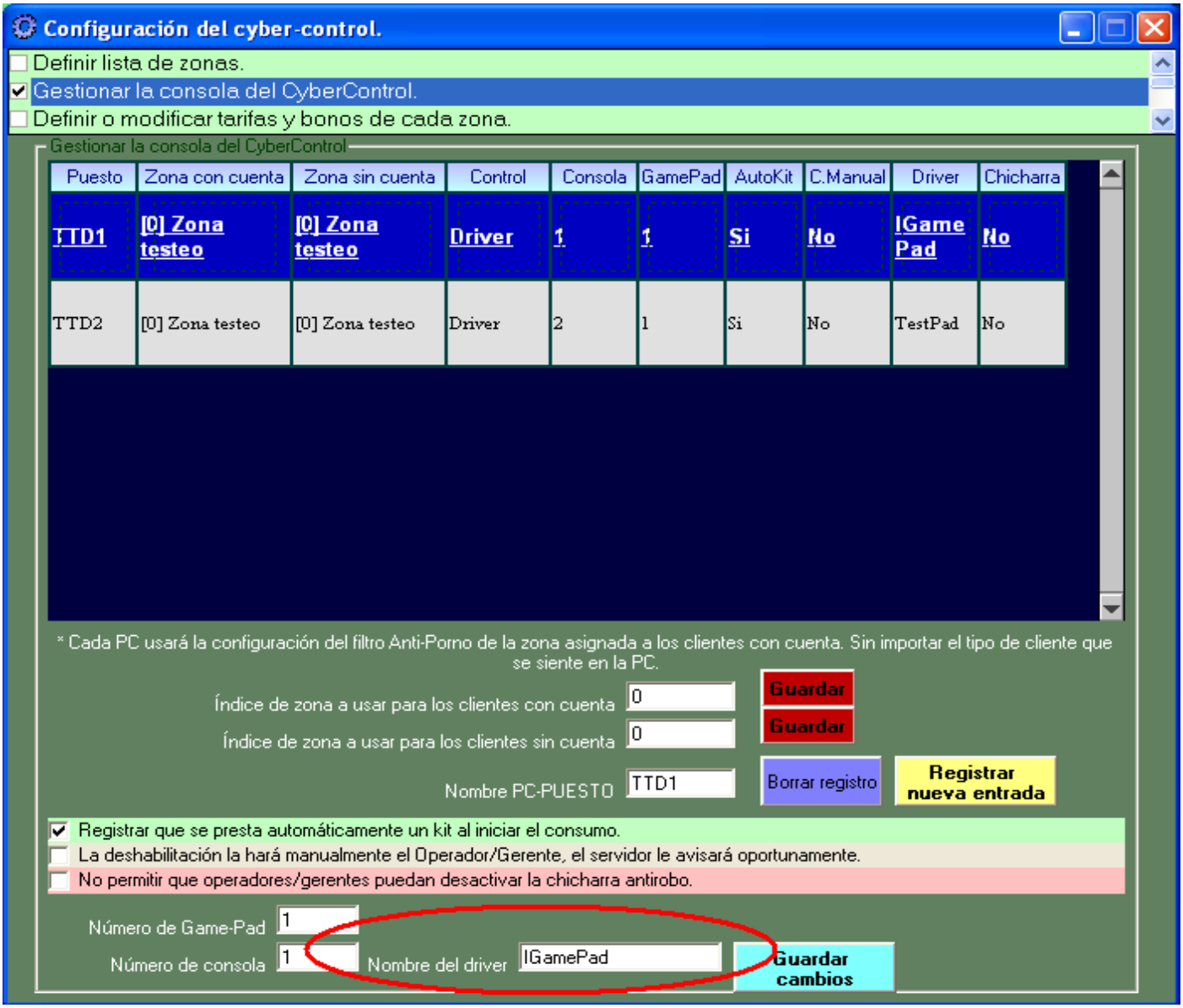
El nombre de librería (en el caso de usar VB5/6, es el nombre del proyecto) y es el nombre que se usará para idividualizar al driver en el caso de un sistema con múltiples drivers, es mas, **no uses el nombre que se muestra en este ejemplo**.



Se puede crear el driver en cualquier lenguaje de programación siempre que el resultado sea un objeto **ActiveX DLL**.

El nombre del archivo DLL generado no tiene importancia y puedes usar cualquier nombre de archivo.

En la configuración del servidor para que el mismo haga uso del driver es indicando el nombre de librería ya que el servidor carga al driver de forma dinámica con enlazado en tiempo de ejecución **CreateObject(Nombre_de_library + ".ClassInterfazGamePad")**.



En el sitio web se puede descargar el **código fuente** plantilla para crear el driver.

Notas no indicadas en el código fuente

El servidor de FlulpyCrea® está preparado para trabajar en conjunto con hardware de control de consolas tan sensillos como por ejemplo; un simple switch bloqueador de VIDEO controlado por puerto LPT, un poco mas avanzado podría ser un switch bloqueador para cada GamePad. Pero estos simples switches bloqueadores, hacen que recaiga todo el trabajo de habilitar los tiempos a consumir en los GamePads exclusivamente en el cajero, porque los usuarios no podrían abrir sus cuentas para habilitar los GamePads.

Ya mas avanzado se podría comprar un hardware de control que cuente con una unidad con todos los switches, un micro-controlador, pantalla LCD, teclado alfanumérico, sensores de desconectividad del cableado (sistema de alarma), y hasta un monedero inteligente para recibirle el dinero al cliente.

Es por eso que la interfaz del driver, tiene por un lado funciones simples que comunican cuestiones básicas como DES/HABILITADO, DES/ACTIVAR ALARMA, etc, y por el otro lado hay 2 funciones que el servidor ejecuta periodicamente que son un canal de comunicación para interactuar con el microcontrolador "**GetAction**" y "**SendOkError**".

Funciones simple:

ClassInterfazGamePad.**GetOnline**
- El servidor pregunta si el GamePad está conectado y listo fisicamente.

ClassInterfazGamePad.**SetDetectionAlarm**
- El servidor ordena activar o desactivar los sensores de la alarma de un GamePad dado.

ClassInterfazGamePad.**GetAlarm**
- El servidor pregunta si la alarma de un GamePad dado está sonando (indicando un posible robo).

ClassInterfazGamePad.**SendStringDisplay**
- El servidor envía al hardware de control, información útil textual para ser mostrada en la pantalla LCD o lo que sea que tenga el hardware de control como display.

ClassInterfazGamePad.**SetStatus**
- El servidor ordena habilitar o deshabilitar un GamePad dado o refresca dicho status.

Funciones complejas:

ClassInterfazGamePad.**GetAction**
- El servidor pregunta al hardware de control sobre eventos (acciones del usuario) que deben ser procesadas.

ClassInterfazGamePad.**SendOkError**
- El servidor devuelve el resultado de los eventos recibidos mediante **GetAction** u otros datos enviados asincrónicamente.

Las enumeradas, son las 7 funciones que conforman la clase ClassInterfazGamePad.

Notas:

- Si el hardware de control cuenta con terminal de control que permite a los clientes controlar las habilitaciones, y se debe mostrar la tabla tarifaria a los efectos de que el usuario pueda elegir el tiempo y costo a consumir, dicha tabla deberá ser refrescada almenos una vez por minuto, también cuando se abra o cierre la cuenta y cuando se habilite/deshabilite el GamePad (ya incluido en el protocolo). Esto es requerido para mostrar todo el tiempo la tabla tarifaria correcta, según el horario/día, y tipo de cliente. Dicho refrescado no debe molestar al usuario final con la selección así como la parte de la tabla visualizada (scroll vertical), por lo que sugerimos solo hacer los cambios visibles en pantalla cuando se haya recibido toda la tabla en memoria.
- Cuando el servidor envía tablas o listas, el último parámetro (Param4 o Param5) es "P","F","U", esto significa que el primer elemento recibido de la tabla o lista tendrá un valor "P" (de primero), y el último elemento tendrá "F" (de fin), los elementos del medio no tendrán valor, en el caso de una tabla con un único elemento tendrá "U".
- La tabla tarifaria transmitida podrá tener un máximo de registros como para listar en el peor de los casos, un registro cada 5 minutos para cubrir el máximo de 12 horas, es decir; 144 registros máximos (pero se puede exceder, en tal caso, descartar los ultimos elementos excedentes). Cada registro, en el campo TIME almacena la cantidad de minutos cuyo rango puede ir desde 1 hasta 720 (12 horas), mientras que el campo costo tendrá un valor numérico que es transmitido textualmente en decimal con fracción (Ej; centavos) que en el peor de los caso será "###############0.0000" (19 caracteres).
- La tabla de bonos deberá ser refrescada cuando se abra y cierre la cuenta para que todo el tiempo muestre los bonos correctos según el tipo de cliente (ya incluido en el protocolo).
- El mensaje "MT" enviado por el servidor que informa los saldos de la cuenta o MoneyBox, puede ocurrir asincrónicamente y sin que el driver le haya solicitado. Por ejemplo; cuando el operador/gerente deposita dinero en una cuenta, y la misma está abierta en una Consola/GamePad, el mensaje "MT" le será transmitidos para informarle el cambio del saldo.
- Parámetros que refieren a periodos de tiempo, están expresados de forma textual decimal entero y corresponde su valor a cantidad de minutos, por ejemplo: 120 minutos == 2 horas. La longitud máxima de la expresión es de 5 caracteres prudentemente hablando (en realidad lo normal es que se usen 3 caracteres, 4 es raro, y 5 muy raro pero posible).
- Parámetros que refieren a costos o valores monetarios, están expresados de forma textual decimal con o sin fracción (según corresponda) y el punto decimal es un ".", la longitud máxima de la expresión es de 19 caracteres.
- El campo "Nombre de bono" puede tener una longitud máxima de 25 caracteres.
- El texto usado por el servidor FlulpyCrea@ es según UNICODE.
- Las acciones que esperan un OK o Error como respuesta, son acciones bloqueantes (es decir; sincrónicas) y bloquean la generación de otras acciones que también sean bloqueantes, mientras que otras informaciones pueden pasar de manera asincrónica e incluso de forma no-solicitada.
- Si el microcontrolador, está esperando la respuesta de una acción bloqueante (sincrónica) y no la recibe durante 1 minuto, debe asumir la caida del servidor, y a nivel microcontrolador debe desloguear unilateralmente todas las cuentas logueadas porque el servidor iniciará de ese modo.
- Aclaramos que una cosa es la cuenta identificada para controlar, y otra cosa es la cuenta de la cual se consume el tiempo habilitado en curso. El servidor al reiniciar inicia sin cuentas logueadas, pero los tiempos que se consumen desde sus respectivas cuentas siguen su curso. Cabe mencionar, que si se consume tiempo desde una cuenta, el servidor impide que se identifique para controlar con otra cuenta.
- Haya o no, cuenta identificada para controlar, que se pueda usar la función **AbortTime** (deshabilitar el tiempo), el servidor guardará el tiempo sobrante no consumido (si corresponde) en la cuenta que habilitó el tiempo.
- Como lo mas probable es que haya una sola terminal de control, así como monedero para todos los GamePads controlados (o almenos uno por consola de juegos), es obvio q la interfaz del usuario en pantalla primero antes que hacer nada debe permitir indicar a cual GamePad se hará referencia, no solo para esta función sino para el manejo del monedero.
- Cuando se informa sobre la acción (evento) **MoneyIn**, se debe indicar el **valor acumulado de las monedas que el usuario ha insertado** en el monedero asociado al GamePad dado **y que aun nunca**

haya sido informado al servidor. Inmediatamente después, el servidor acusará que ha aceptado el ingreso del dinero y de cuanto dinero aceptado se trató. En la instancia en la cual el dinero ingresado se informa al servidor, se deberá ser descontado de la variable "**dinero nunca informado**" y recordado en una segunda variable "**dinero informado no acusado**", luego cuando el servidor acusa que ha aceptado el dinero y cual cantidad, se descontará lo aceptado de la variable "**dinero informado no acusado**". Habiendo un saldo restante en la variable "**dinero informado no acusado**" durante mas de 1 minuto y que no ha habido un acuse por parte del servidor de dinero recibido, el hardware de control transferirá ese saldo a la variable "**dinero nunca informado**" de modo que cuando se pregunte por una acción se ejecute MoneyIn informando nuevamente sobre ese dinero y se cumpla el ciclo. Esto es así porque si se pierde conectividad entre el servidor y el hardware controlador (como la caida del servidor), el hardware controlador enviará el evento MoneyIn pero nunca recibirá una respuesta de acuse, además el comando MoneyIn es un comando asincrónico es decir, no bloqueante y en cualquier momento se puede producir y múltiples veces y de no haber conectividad, el dinero tiene que poder quedarse en standby hasta que el servidor pueda recibir la información y procesarla. **Un minuto** es un tiempo adecuado de **PING_TIME_OUT** ya que el servidor reinicia como muy rápido en 1 minuto y 20 segundos, para controlar que en la variable "**dinero informado no acusado"** tenga todo su dinero con almenos un minuto de antigüedad de no acusado, se usará un contador de tiempo que se reiniciará cada vez que la variable cambie su valor, o cuando su saldo sea cero.

- Cuando ingresa dinero, y se le informa al servidor, el mismo tomará alguna de las siguientes acciones.
  1. Si hay cuenta identificada en el GamePad, se deposita el dinero en la cuenta.
  2. Si no hay cuenta asociada, el servidor acumulará el dinero a la espera de que el cliente indique que hacer con el dinero (comprar un bono habilitador, comprar tiempo, etc).

INFORME DE OBJECT VIEWER

```
// Generated .IDL file (by the OLE/COM Object Viewer)
//
// typelib filename: IGamePad.dll

[
  uuid(32120F74-8878-4923-BF52-4709B8EF8A98),
  version(9.0),
  helpstring("Interfaz Standard de FlulpyCrea")
]
library IGamePad
{
    // TLib :     // TLib : OLE Automation : {00020430-0000-0000-C000-000000000046}
    importlib("stdole2.tlb");

    // Forward declare all types defined in this typelib
    interface _ClassInterfazGamePad;

    [
      odl,
      uuid(69702550-8DCC-49CC-913B-ECA42064F161),
      version(1.0),
      hidden,
      dual,
      nonextensible,
      oleautomation
    ]
    interface _ClassInterfazGamePad : IDispatch {
        [id(0x60030000)]
        HRESULT GetOnline(
                        [in] unsigned char Console,
                        [in] unsigned char GamePad,
                        [out, retval] VARIANT_BOOL* );
        [id(0x60030001)]
        HRESULT SetDetectionAlarm(
                        [in] unsigned char Console,
                        [in] unsigned char GamePad,
                        [in] VARIANT_BOOL Active);
        [id(0x60030002)]
        HRESULT GetAlarm(
                        [in] unsigned char Console,
                        [in] unsigned char GamePad,
                        [out, retval] VARIANT_BOOL* );
        [id(0x60030003)]
        HRESULT SendStringDisplay(
                        [in] unsigned char Console,
                        [in] unsigned char GamePad,
                        [in] BSTR Mensaje);
        [id(0x60030004)]
        HRESULT SetStatus(
                        [in] unsigned char Console,
                        [in] unsigned char GamePad,
                        [in] long Status);
        [id(0x60030005)]
        HRESULT GetAction(
                        [in] unsigned char Console,
                        [in] unsigned char GamePad,
                        [in, out] BSTR* Param1,
                        [in, out] BSTR* Param2,
                        [out, retval] unsigned char* );
        [id(0x60030006)]
        HRESULT SendOkError(
                        [in] unsigned char Console,
                        [in] unsigned char GamePad,
                        [in] unsigned char Error,
                        [in] BSTR Param1,
                        [in] BSTR Param2,
                        [in] BSTR Param3,
                        [in] BSTR Param4,
                        [in] BSTR Param5);
    };

    [
```

```
      uuid(F8400B8F-61B8-4877-AD32-A7390875EF6B),
      version(1.0)
    ]
    coclass ClassInterfazGamePad {
        [default] interface _ClassInterfazGamePad;
    };
};
```