CashKeeper®

Closed and secure cash handling

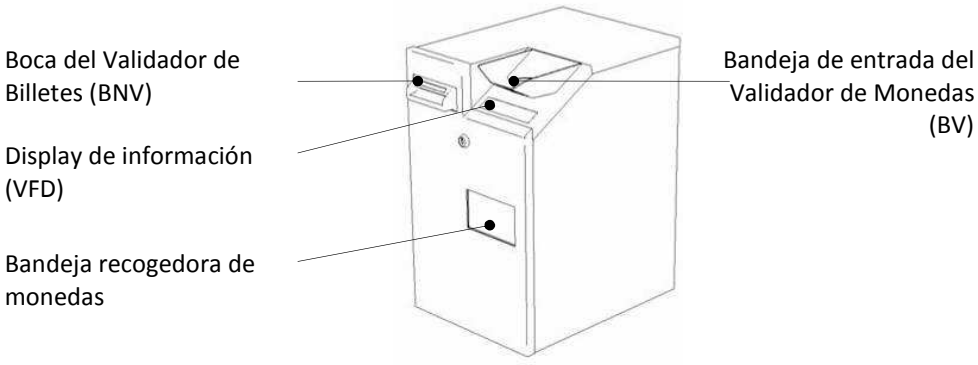
MANUAL DE INTEGRACIÓN

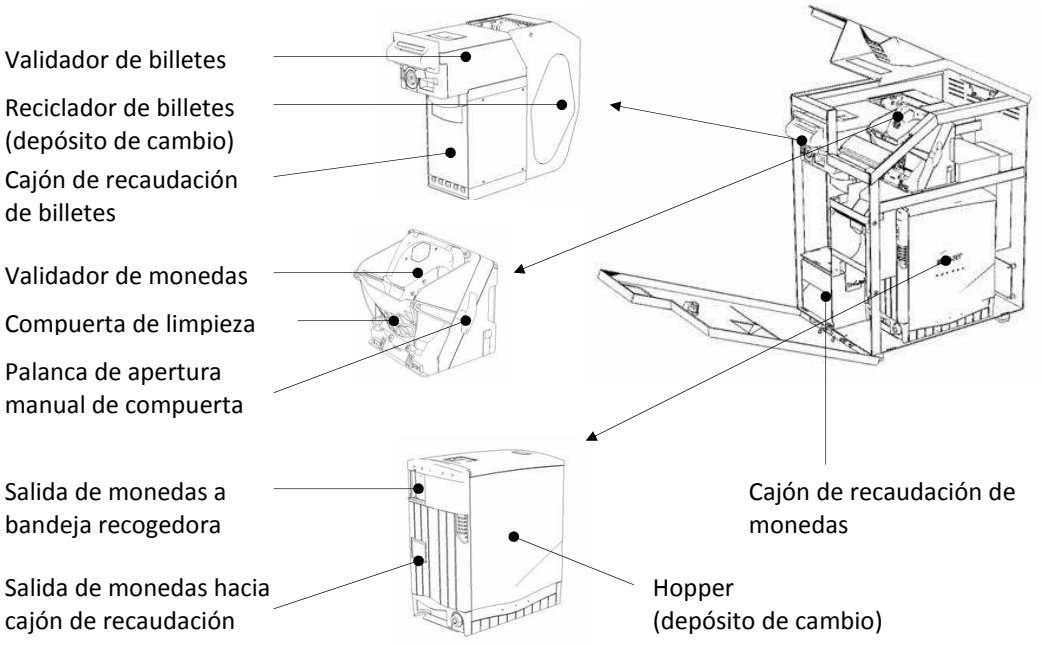
v. *preliminar*

Fecha	Modificación	Resp	Nivel Doc	Versión DLL	Versión CSSI	Compatible	Protocolo
18/03/12	Versión preliminar	MIC	-	-			
11/03/15	Añadido CriticalBehaviour	MIC					
5/10/15	Actualizada la tabla de errores. Añadida función GetLastLevels. Añadida función SetDateTime. Añadida función GetNetworkParams. Añadida función SetNetworkParams. Añadida función EmptyCashBoxEx. Añadida función GetCCDetails. Actualizada la función GetAllProperties.	MIC					
19/12/16	Actualizada función GetCurrentLevels	MIC					
25/05/17	Actualizada la función GetLevels Añadida sección Propiedades especiales	MIC					

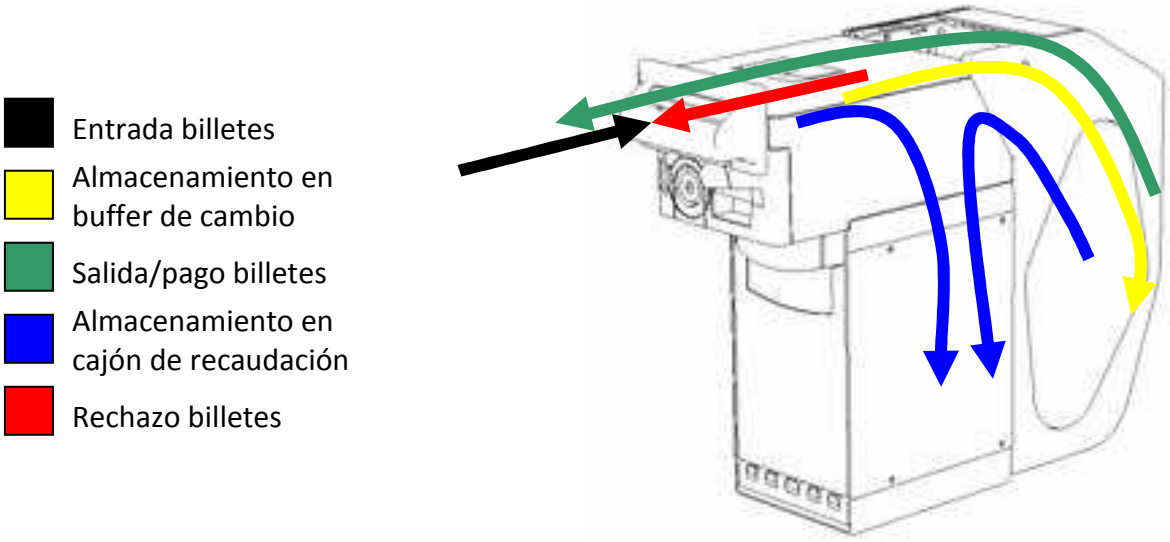
1. DESCRIPCIÓN DE *CashKeeper* ®

- **Descripción de los elementos**





- **Rutas de billetes y monedas en CashKeeper ®**



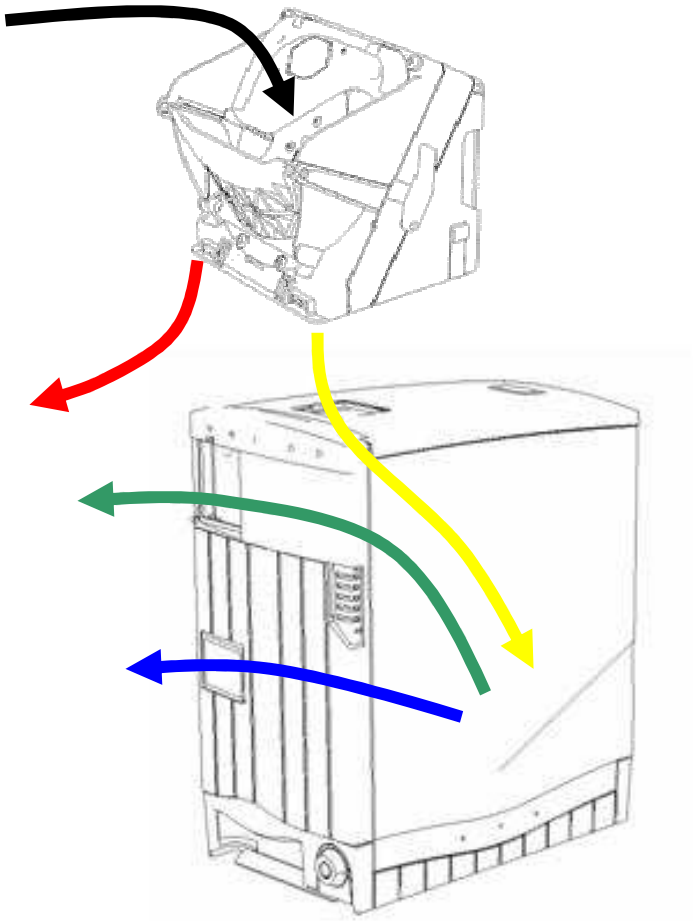
Entrada monedas

Almacenamiento en buffer de cambio

Salida/pago monedas

Almacenamiento en cajón de recaudación

Rechazo monedas



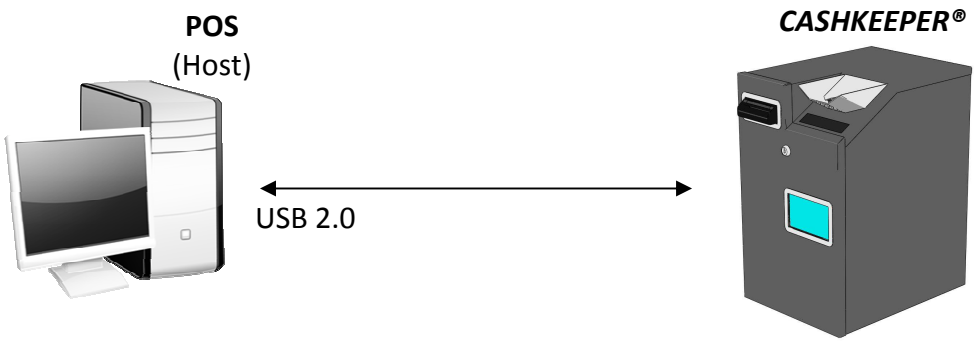
2. CONCEPTO DE INTEGRACIÓN DE *CashKeeper ®*

Debido a la propia naturaleza de los procesos de pago y cobro, ***CashKeeper®*** es un dispositivo que funciona de forma **asíncrona**. Existen diversos procedimientos que pueden ser realizados de forma síncrona, pero los que implican funcionamiento físico del dispositivo (entiéndase cobros, pagos, envíos de efectivo a cajones de recaudación, etc…), funcionan de manera totalmente asíncrona.

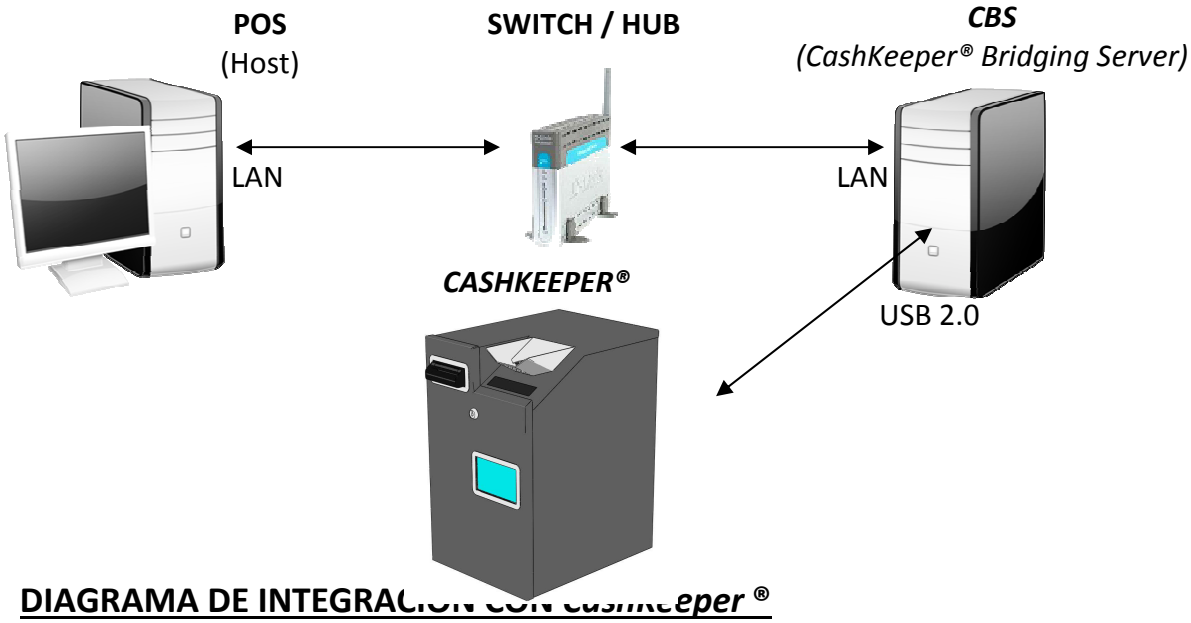
La integración de un dispositivo ***CashKeeper®*** se debe realizar a través de la comunicación con una interfaz llamada ***'CashKeeper® Secure Server Interface' (CSSI)***, basada en protocolo TCP/IP. Para establecer esta comunicación con la ***CSSI***, se podrá realizar a través de la herramienta ***Ckeeper.DLL*** (CashKeeper Control Tool), o a través del '***método directo***', atacando a la CSSI directamente vía sockets (TCP/IP).

La interfaz ***CSSI*** podrá estar residente en el propio TPV o se podrá utilizar un medio 'puente' entre el TPV y el dispositivo ***CashKeeper®***, el ***'CashKeeper® Bridging Server' (CBS)***. El ***CBS*** es un servidor que dedicará toda su capacidad y sus recursos a controlar la comunicación entre el TPV y el dispositivo ***CashKeeper®***, a través de la ***CSSI***.

- **INTEGRACIÓN CON *CSSI* LOCAL**

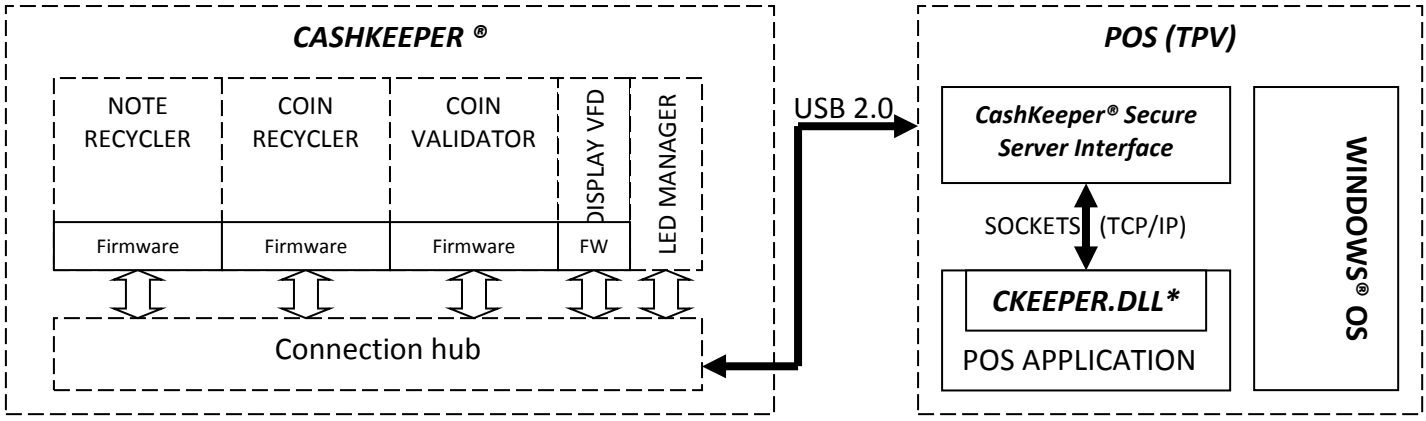


- **INTEGRACIÓN CON *CSSI* REMOTA (con *CashKeeper® Bridging Server*)**



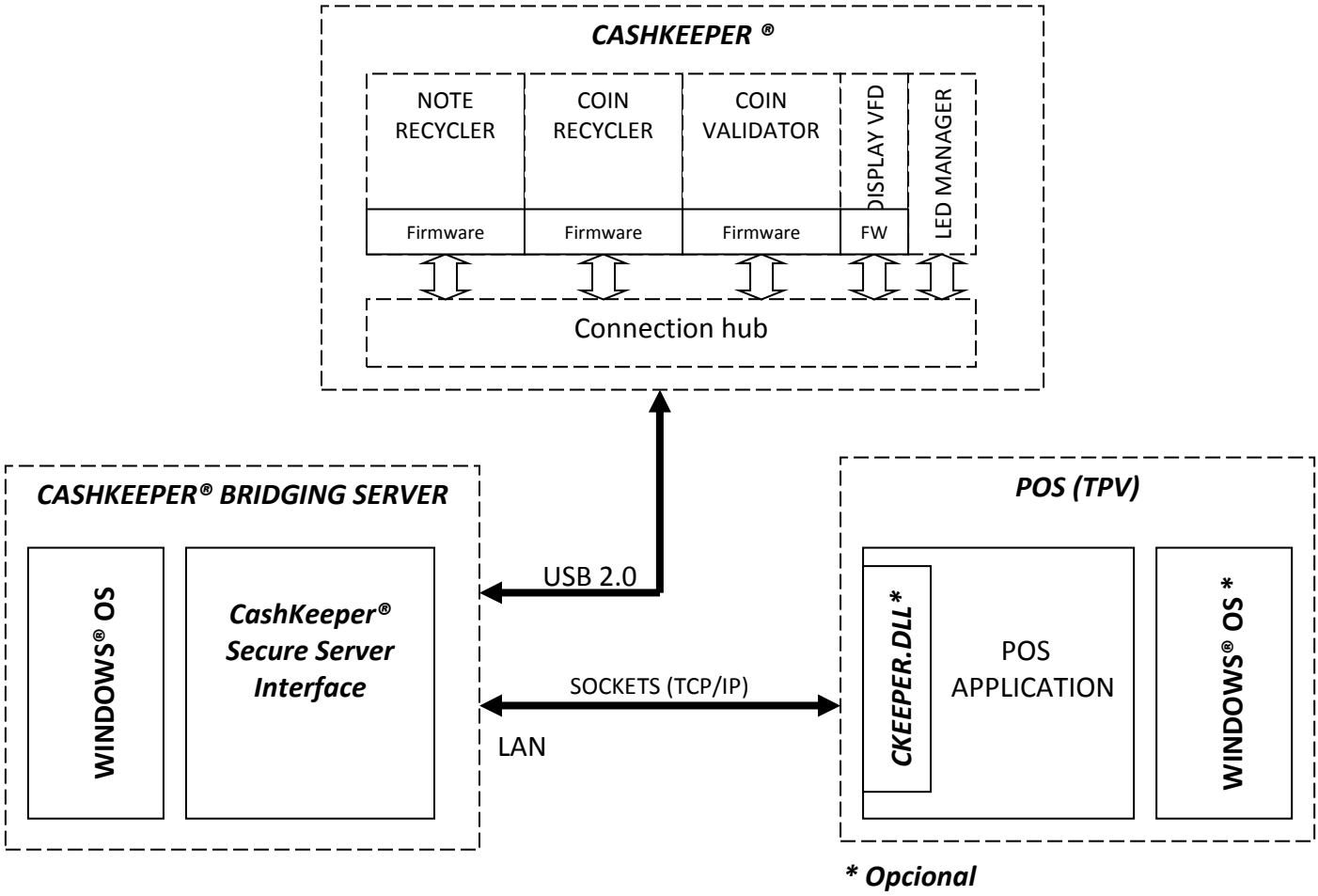
3. DIAGRAMA DE INTEGRACIÓN CON *CashKeeper ®*

- **INTEGRACIÓN CON *CSSI* LOCAL**



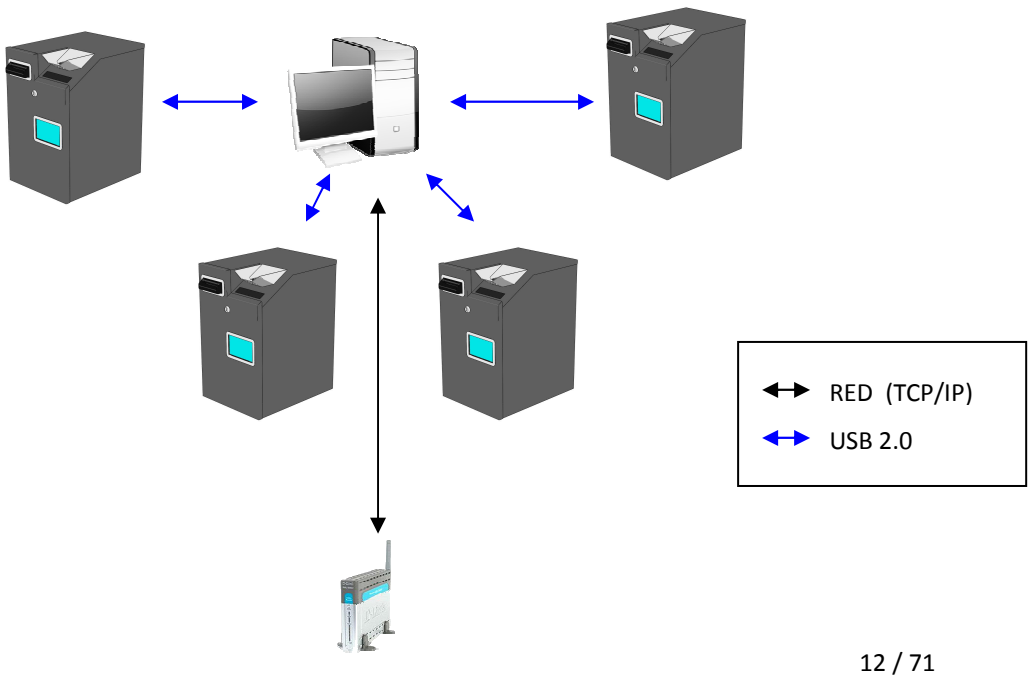
**** Opcional***

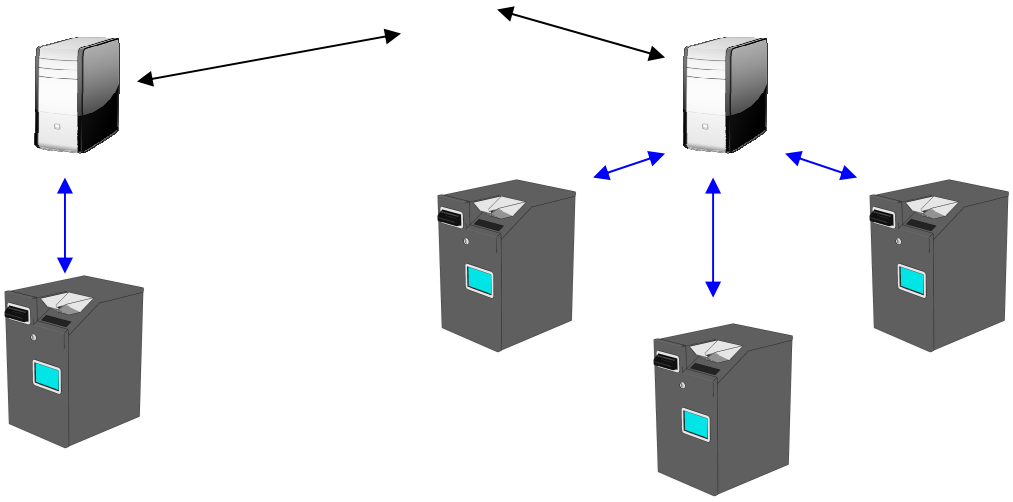
- **INTEGRACIÓN CON *CSSI* REMOTO (con *CashKeeper® Bridging Server*)**



3.1. Integración de múltiples CashKeeper

(protocolo 4 o posterior)





4. Metodos de integración con CashKeeper

4.1. CKEasy

Se trata del sistema más fácil y rápido de integración, ya que incorpora su propia interficie grafica. Todas las llamadas son síncronas, lo que facilita el uso de las distintas funcionalidades. Dichas funcionalidades, son de muy alto nivel, del tipo cobra, paga, hacer caja, etc. Así mismo, incorpora algunas llamadas sin soporte gráfico para permitir a la aplicación la obtención de datos de caja.

4.2. CKeeper

Se trata de un sistema orientado a eventos que permite controlar CashKeeper en toda su plenitud, no incorpora interficie gráfica, por lo que permite una completa integración con su aplicación. Ofrece la misma funcionalidad que el método sockets, pero al tratarse de un Objeto ActiveX, facilita la programación.

4.3. Sockets

Es el sistema de integración de mas bajo nivel y el que requiere una mayor complejidad, aunque le permite ser transparente al SO utilizado, el único requerimiento es que el SO en el que se ejecuta su aplicación disponga de sockets. Hoy en día todos.

5. Utilizar CKEasy

5.1. Utilización

Para utilizar CKEasy, cree un objeto ActiveX EasyCashKeeper Ya que el proceso de conexión requiere unos segundos. Se recomienda conectar al iniciar la aplicación (aunque el integrador puede elegir cuando conectar y en todo caso, se conecta automáticamente al llamar a cualquier función) y desconectar al salir. Una vez conectado, utilice las llamadas a las funciones según se desee. Atención, esta librería no es 'thread safe', por lo que no se pueden realizar llamadas durante la ejecución de otra llamada.

5.2. Conexión

Si se conecta al CSSI ubicado en el propio ordenador, solo necesita llamar el método **Connect**.

Si se conecta al CSSI ubicado en otro ordenador hace falta informar la propiedad IP con la dirección ip o el nombre de

red del ordenador que tenga conectado físicamente el CashKeeper. A continuación llamar el método **Connect**.

5.3. Desconexión

Solo necesita llamar la función **Disconnect**.

5.4. Cobro

Para realizar un cobro la función **Charge** realizará el cobro, devolviendo una vez finalizado el importe introducido, el cambio devuelto y si se ha producido algún problema.

5.5. Pago

Para realizar un pago la función a llamar es **Pay**, devolverá el resultado del pago (correcto/ incorrecto).

5.6. Pago específico

Para realizar pagos indicando en que denominaciones se desea, hay que llamar la función **PaySpecific**. Devolverá el resultado (correcto/incorrecto) así como el importe pagado.

5.7. Añadir cambio

La función **AddChange** es específica para añadir cambio, mostrando en la pantalla la carencia/insuficiencia según la configuración de las distintas denominaciones y permitiendo la introducción de monedas y billetes. Devuelve el importe.

5.8. Dar cambio

La función **Change** abre una pantalla que permite introducir dinero y elegir en que forma queremos que nos lo devuelva. Hay que recordar que en una instalación con CashKeeper no hay dinero en mano para dar cambio para la maquina de tabaco, cochecitos, etc. Devuelve el importe introducido y el valor pagado.

5.9. Parámetros

La función **Configuration** permite abrir una pantalla donde se pueden definir algunos parámetros que permiten adaptar el funcionamiento de CashKeeper al funcionamiento del cliente.

5.10. Caja

La función **CashBoxControl** permite a la aplicación controlar los procesos de fin de día, configurar la aceptación/rechazo de las distintas denominaciones, vaciar cajones de recaudación, vaciar totalmente el cambio disponible a cajón de recaudación, establecer las cantidades iniciales, etc. Incluye varias opciones sin interficie gráfica, para el caso de que la aplicación ya disponga de una pantalla propia.

5.11. Funciones sin interficie gráfica

Las funciones **EmptyCashBox**, **GetAmounts** y **GetLevels** permiten vaciar cajones, obtener el importe y obtener el detalle de las distintas denominaciones sin ninguna interferencia gráfica.

5.12. Propiedades especiales

Estas propiedades son de solo lectura y solo son válidas cuando se ha establecido la conexión. Si no hay conexión, devolverán valores por defecto. Ambas propiedades permiten a los programas adaptarse a las distintas divisas.

ShowDecimals Esta propiedad contiene el número de decimals a mostrar en los importes.

AmountFactor Esta propiedad contiene el factor de conversion entre la unidad de la divisa y los importes a enviar y recibir con CashKeeper.

Ej. En EUR el factor es 100.

Si quiero cobrar o pagar 1,45€, tengo que mandar 145 (1,45 * 100 = 145)

Si añado cambio, CashKeeper me indicará 145, Aplicando el Factor obtendré 1,45€ (145 / 100 = 1,45)

5.13. Otros parámetros

BackColor y **InverseColor** son dos propiedades que permiten elegir al integrador el color de los formularios para "mejorar" la integración de CashKeeper con su aplicación.

ErrorCode y **ErrorDescription** son dos variables que contienen el código y la descripción en caso de error.

IP es una propiedad que permite seleccionar la ip del cashkeeper, por defecto contiene "localhost".

OnErrorDiscard es una propiedad que permite indicar que en caso de error, se eliminen los créditos pendientes.

5.14. Referencia de las funciones.

AddChange(PaidInValue As Int32)

Abre una pantalla que permite añadir cambio a CashKeeper.

Parámetros: de salida

PaidInValue contendrá el importe añadido a cambio

Devuelve:

Verdadero si ha terminado correctamente

Falso si se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

CashBoxControl(Mode As CBC_Modes, CoinsValueInCB As Int32, NotesValueInCB As Int32, RemainingChange As Int32)

Parámetros de entrada:

Mode valores posibles:

0-Normal: Abre un formulario para ver los niveles de Cashkeeper y permite cambiar la configuración de valores iniciales, mínimos y máximos así como hacer caja.

1- CBC_Disable_Config: Abre un formulario para ver los niveles de CashKeeper y permite hacer caja.

4- CBC_Disable_Actions: Abre un formulario para ver los niveles de CashKeeper y permite cambiar la configuración de valores iniciales, mínimos y máximos.

7- CBC_Blind: Muestra una pantalla de proceso (sin datos) y hace el proceso de fin de día (enviar a cajón el sobrante del cambio inicial y vaciar los cajones de recaudación).

15- CBC_Blind_By_Value: Muestra una pantalla de proceso (sin datos) y hace el proceso de fin de día (enviar a cajón el sobrante del cambio inicial y vaciar los cajones de recaudación). Debe informarse el parámetro RemainingChange con el importe que se desea mantener en cambio.

5: Es la combinación del modo 1 y 4, Lo que muestra una pantalla con los datos de CashKeeper, pero no deja realizar ninguna acción.

Parámetros: de salida

CoinsValueInCB: contendrá el importe de las monedas en cajón de recaudación en caso de haber vaciado los cajones de recaudación.

NotesValueInCB: contendrá el importe de los billetes en cajón de recaudación en caso de haber vaciado los cajones de recaudación.

RemainingChange: contendrá el importe del cambio que tiene CashKeeper.

Devuelve:

Verdadero si ha terminado correctamente

Falso si se ha producido algún error, consulte las propiedades ErrorCode y ErrorDescription para más información.

Configuration()

Abre una pantalla que permite configurar distintos parámetros de CashKeeper.

Devuelve:

Verdadero si ha terminado correctamente

Falso si se ha producido algún error, consulte las propiedades ErrorCode y ErrorDescription para más información.

Connect ()

Inicia la conexión con el CSSI.

Devuelve:

Verdadero si ha terminado correctamente

Falso si se ha producido algún error, consulte las propiedades ErrorCode y ErrorDescription para más información.

Change(PaidInValue As Int32, PaidOutValue As Int32)

Abre una pantalla que permite realizar cambios de denominaciones. Por ejemplo 1 billete de 5€ por 5 monedas de 1€.

Parámetros: de salida:

PaidInValue: Importe introducido

PaidOutValue: Importe devuelto

Devuelve:

Verdadero si ha terminado correctamente

Falso si se ha producido algún error, consulte las propiedades ErrorCode y ErrorDescription para más información.

Charge(ValueInCents As Int32, PaidInValue As Int32, PaidOutValue As Int32)

Abre una pantalla para realizar el cobro de un importe en céntimos.

Parámetros de entrada:

ValueInCents : Importe a cobrar

Parámetros: de salida:

PaidInValue: Importe introducido

PaidOutValue: Importe devuelto

Devuelve:

Verdadero si ha terminado correctamente

Falso si se ha producido algún error, consulte las propiedades ErrorCode y ErrorDescription para más información.

Es importante verificar los valores devueltos haya terminado correctamente o no. SI ha terminado correctamente, verificando los valores, permite detectar el "broken cent", En caso de finalizar incorrectamente, permite verificar si falta algo por devolver o que hay un importe parcial.

Disconnect ()

Termina la conexión con el CSSI. Este método, nunca puede fallar, por lo que no devuelve valores.

EmptyCashBox(Device As CKD_Devices, Value As Long)

Función sin pantalla para vaciar los cajones de recaudación.

Parámetros de entrada:

Device: Cajón a vaciar valores posibles:

CKD_Coins = 0

CKD_Notes = 1

Parámetros: de salida:

Value: Importe que contenía el/los cajón/es vaciado/s antes de vaciarlo/s.

Devuelve:

Verdadero si ha terminado correctamente

Falso si se ha producido algún error, consulte las propiedades ErrorCode y ErrorDescription para más información.

GetAmounts(Denoms As String, Quantities As String, Value As Int32)

Función sin pantalla para obtener el detalle del contenido en cambio de CashKeeper.

Parámetros: de salida:

Denoms: Lista de denominaciones separada por comas.

Quantities: Lista de las cantidades de las denominaciones separadas por coma y respectiva a la lista de denominaciones.
Value: Importe del contenido en cambio.
Devuelve:
Verdadero si ha terminado correctamente
Falso si se ha producido algún error, consulte las propiedades ErrorCode y ErrorDescription para más información.

GetLevels(LevelType As CKL_Levels, Denoms As String, Levels As String)

Función sin pantalla para obtener el detalle del contenido en de CashKeeper. En función del parámetro LevelType devolverá los iniciales, máximos, cambio o cajón de recaudación.
Atención: En caso de que algún dispositivo esté en error, el contenido de dicho dispositivo puede no mostrarse.
Parámetros de entrada:
LevelType: valores posibles:
CKL_Initial = 0
CKL_Max = 1
CKL_Current = 2
CKL_CashBox = 3
CKL_LevelStatus = 4
Parámetros: de salida:
Denoms: Lista de denominaciones separada por comas.
Levels: Lista de las cantidades de las denominaciones separadas por coma y respectiva a la lista de denominaciones. En el caso de CKL_LevelStatus, Levels contendra los codigos de los semáforos (0 = verde, 1 = naranja, 2 = rojo).
Devuelve:
Verdadero si ha terminado correctamente
Falso si se ha producido algún error, consulte las propiedades ErrorCode y ErrorDescription para más información.

Pay(ValueInCents As Int32, PaidOutValue As Int32)

Abre una pantalla que informa sobre el proceso de pago de un importe en céntimos.

Parámetros de entrada:

ValueInCents: importe en céntimos a pagar.

Parámetros: de salida:

PaidOutValue: Importe realmente pagado.

Devuelve:

Verdadero si ha terminado correctamente

Falso si se ha producido algún error, consulte las propiedades ErrorCode y ErrorDescription para más información.

PaySpecific(ValueInCents As Int32, PaidOutValue As Int32)

Abre una pantalla que permite indicar en que denominaciones se va a realizar el pago, una vez indicado muestra el proceso de pago de un importe en céntimos.

Parámetros de entrada:

ValueInCents: importe en céntimos a pagar.

Parámetros: de salida:

PaidOutValue: Importe realmente pagado.

Devuelve:

Verdadero si ha terminado correctamente

Falso si se ha producido algún error, consulte las propiedades ErrorCode y ErrorDescription para más información.

5.15. Referencia de las propiedades

BackColor

Indica el color de fondo de los formularios

InverseColor

Indica el color de las letras, se calcula automaticamente al cambiar la propiedad BackColor

IP

Dirección IP donde esta ubicado el CSSI. Se utiliza en el momento de conexión por lo que hay que informarlo antes de conectar.

OnErrorDiscard

Cuando se produce un error en una operación de cobro, puede suceder que quede saldo “pendiente”. Si esta propiedad esta a falso, al realizar el siguiente cobro, tendrá en cuenta el saldo “pendiente” de la operación anterior. En caso de ser cierto, el saldo “pendiente” no se tendrá en cuenta.

6. UTILIZAR *CKEEPER*

A continuación se detallan los procedimientos a realizar para operar con ***CashKeeper®***:

6.1. Estados de *CashKeeper®*

Para entender el funcionamiento de los procesos de CashKeeper, antes debemos explicar sus estados:

- ALL_CLOSED (0x00): Indica que los puertos de comunicación no están abiertos y no hay comunicación activa con el dispositivo. Es el estado inicial. Atención, ya que se trata de un estado de CashKeeper, sin estar conectado es imposible de saberlo. Por lo tanto el integrador debe controlar cuando esta o no esta conectado.

- IDLE (0X01): El sistema se encuentra en estado de reposo. La comunicación con el dispositivo está activa y es posible realizar cualquier acción. A excepción de algunos comandos concretos (utilizados para que CashKeeper cambie de estado), la mayoría de las funciones deben ejecutarse cuando CashKeeper está en este estado.

- ENABLED (0X02): El sistema se encuentra en estado de recepción de efectivo.

- PAYING (0X03): El sistema está realizando un pago (en billetes y/o monedas). Durante este estado, el sistema desencadenará el evento **ValueOUT** informando del total del importe pagado.

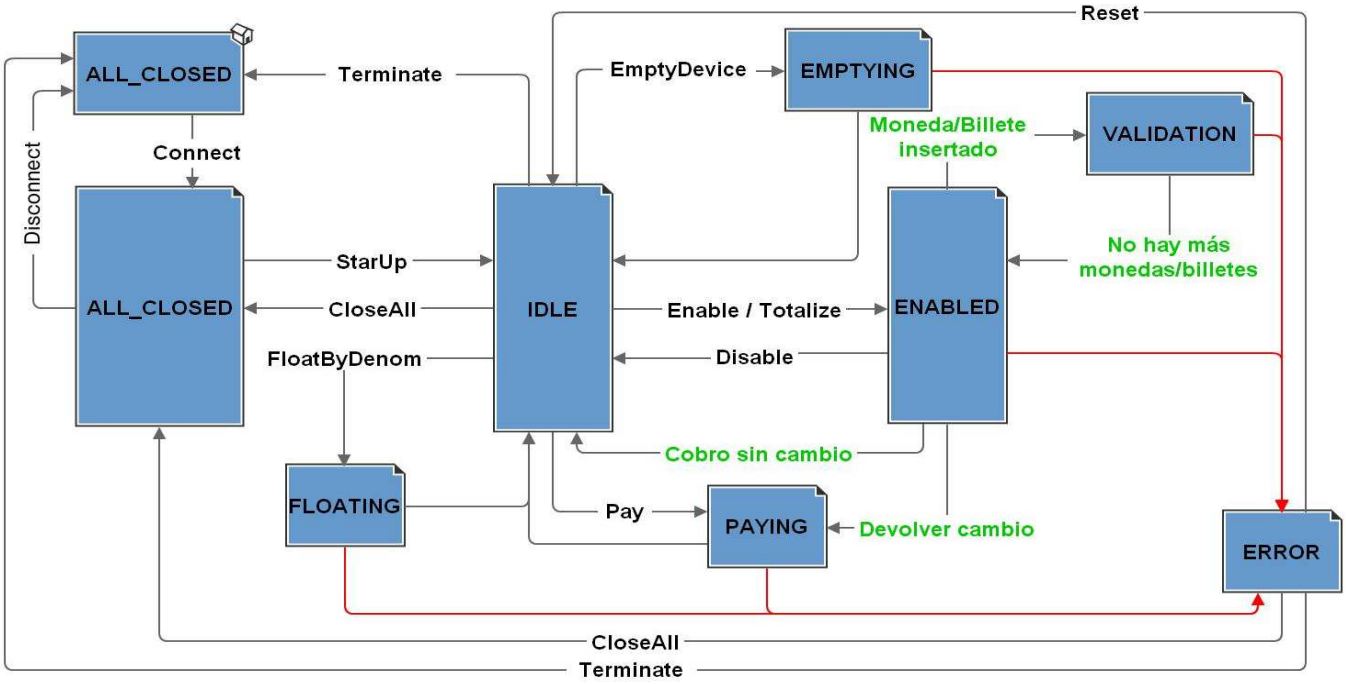
- FLOATING (0x04): El sistema está enviando monedas y/o billetes hacia los cajones de recaudación. Durante este estado, el sistema desencadenará el evento **ValueToCashBox** informando del total del importe enviado a cajón/es.

- EMPTYING (0X05): El sistema está vaciando los recipientes de cambio de monedas y/o billetes hacia los cajones de recaudación. Durante este estado, el sistema desencadenará el evento **ValueToCashBox** informando del total del importe enviado a cajón/es.

- ERROR (0x06): El sistema se encuentra en un estado de error y no es posible operar con él. El sistema adquiere este estado cuando se produce un error no recuperable (billete o moneda atascada, etc.). La única excepción seria la función **Reset**, que durante su ejecución, adquiere este estado para evitar la ejecución de otros comandos.

- VALIDATION(0x07): El sistema se encuentra verificando las monedas/billetes introducidos, esto significa que hay movimiento mecánico en marcha. Es en este estado cuando se producen los eventos **ValueIN** correspondientes al efectivo validado.

6.2. Diagrama de estados y su transición



Leyenda
Cajitas: Estados
Flechas en negro: funciones CashKeeper
Flechas en verde: acciones externas/decisiones
Flechas en rojo: Situaciones de error.

6.3. Consideraciones previas

6.3.1. Denominaciones aceptadas

CashKeeper es capaz de aceptar y validar distintas denominaciones, de distintos países. A modo general, se establece una divisa (EUR, GBP, USD, etc.) como principal (será la única que se paga), y puede haber otras divisas que solo se acepten. Por ejemplo, en zona fronteriza/turística puede interesar aceptar además de la moneda local, dólares USA.

6.3.2. Información de datos de valor

Todos los datos de valor que se devuelvan o se deban informar al sistema, se harán en modo de céntimos y sin decimales.
(p.e. 4,5 € → 450, 3.5£ → 350).

6.3.3. Respuesta de funciones

Todas las funciones de CashKeeper devuelven un valor tipo 'boolean' como resultado de la misma. Si la función se puede llevar a cabo de forma satisfactoria ésta devolverá verdadero. En caso contrario, devolverá falso y se deberá consultar a las propiedades 'ErrorCode' y 'ErrorDescription'.

6.3.4. Pagos en CashKeeper

CashKeeper está programado para realizar los pagos en el menor número de monedas y/o billetes posible. De todas formas, en algún caso, podrían no utilizarse las denominaciones óptimas teóricas para reducir los tiempos de pago (p.e. si se debe realizar un pago de 10 céntimos de Euro habiendo monedas de 10 céntimos disponibles, es posible que CashKeeper utilice 2 monedas de 5 céntimos si el proceso de búsqueda de la moneda de 10 céntimos demora demasiado).

6.3.5. Evento DISABLED

Debido a la naturaleza de los procesos de CashKeeper, las funciones que implican actividad mecánica, son procesos asíncronos. Para ayudar al integrador a conocer en qué momento se da por finalizado un proceso y, por tanto, se puede proceder al envío de nuevos comandos, se desencadena el evento **'Disabled'** para indicar el final del mismo.

6.3.6. Gestión de Eventos

La librería activa los eventos uno a uno, hasta que no finaliza un evento no envia el siguiente, por lo que hay que procurar que el proceso de los eventos sea lo mas rápido posible.

6.3.7. Características

Esta librería no es 'thread safe' , por lo que no esta permitido, realizar llamadas durante la ejecución de otras llamadas.

6.4. Puesta en marcha *(proceso síncrono)*

[Funciones implicadas: Connect, GetDevices , StartUp]

Para la puesta en marcha del sistema, antes se debe abrir una conexión con la CSSI activa, mediante el método **Connect(***HostIP as string, HostPort as byte, OfficePort as byte, ConnectionType as byte, SecuritySeed as Int32***).** Para ello se deberán passar los siguientes parámetros:

- **HostIP**: nombre de red o dirección IP donde localizar a la CSSI.
- **HostPort**: puerto de escucha de la CSSI para las conexiones HOST.
- **OfficePort**: puerto de escucha de la CSSI para las conexiones BACKOFFICE.
- **ConnectionType**: tipo de conexión que se está realizando (Host o BackOffice)
- **SecuritySeed**: Código de seguridad para securizar las conexiones con la CSSI (de valor mayor a 100000 e inferior a 999999).

6.4.1. CSSI Local

Si CashKeeper se encuentra conectado directamente a la máquina HOST, la CSSI se iniciará automáticamente al invocar el método CONNECT.

6.4.2. CSSI Remoto

En el caso de que CashKeeper se encuentre conectado a un PC remoto (CBS u otro), la CSSI deberá estar activa previamente antes de invocar la función Connect.

Si el resultado de la función Connect es satisfactorio, la función devolverá verdadero. En caso contrario, devolverá falso y se deberá consultar las propiedades **ErrorCode y ErrorDescription** para conocer la causa del fallo.

Una vez se haya podido establecer la comunicación con la CSSI, podemos consultar el identificador de cada CashKeeper conectado al ordenador en que se ejecuta el CSSI. En el caso de que ya se haya configurado previamente y sepamos la configuración, podemos obviar este paso.

El siguiente paso seria invocar la función **StartUp(***ConfigID as byte, Optional Device as Int32***)** para abrir los puertos de comunicación con CashKeeper. Los parámetros de són:

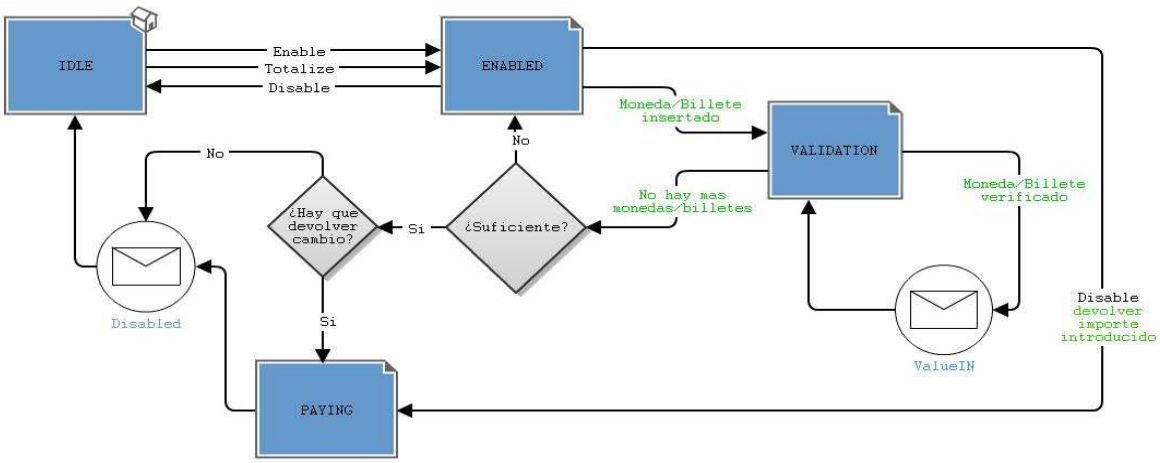
- **ConfigID** (Byte): Indica la configuración a iniciar. En caso de no existir, se crea automáticamente. Se pueden tener tantas configuraciones como se desee (256 máximo) con el mismo dispositivo. En el caso que un mismo PC tenga conectados más de un CashKeeper, la configuración indica con cual de ellos se va a conectar.
- **Device** (Long): Identificador de dispositivo. En el caso de la primera conexión de un CashKeeper y que el ordenador que ejecuta el CSSI tenga mas de un CashKeeper conectado, este parámetro pasa a ser obligatorio y necesitamos indicar con cual de ellos queremos conectar.

Una vez se haya ejecutado de forma satisfactoria la función **StartUp** , el estado de CashKeeper pasará del estado ALL_CLOSED(0x00) a IDLE (0x01).

6.5. Cobros

[Funciones implicadas: Enable, Totalize, Disable, AlternateOperation, DiscardOperation]

El principal uso del dispositivo es el cobro de importes, Cashkeeper puede tener hasta 10 operaciones de cobro abiertas simultáneamente, aunque solo una puede estar activa. De forma predeterminada, se trabaja en la operación cero, Si en medio de un cobro (cuando este en estado **ENABLED** (0x02)), queremos cambiar a otra operación, podemos hacerlo mediante la función **AlternateOperation**. La transición de estados para realizar un cobro, seria la siguiente:



Podemos distinguir dos tipos de cobro:

Conocemos el importe de antemano: A través de la función **Totalize**, informamos a CashKeeper que deseamos cobrar un determinado importe y que cierre automáticamente el cobro.

Queremos habilitar el dispositivo, pero aún no conocemos el importe: A través de la función **Enable** ponemos CashKeeper en estado ENABLED para que acepte dinero. Una vez conocemos el importe, utilizamos la función **Totalize** para informar a CashKeeper el importe a cobrar.

Si estando en estado ENABLED queremos abortar el cobro, el comando Disable nos permite devolver el dinero introducido y volver al estado IDLE.

Una vez esta en estado VALIDATION (se esta validando el dinero introducido), se dispara el evento **ValueIN** informándonos del importe validado hasta ese momento junto con el importe a cobrar, en caso de haber sido informado, y el numero de operación en curso.

Una vez finalizado el cobro o bien cancelado el cobro, CashKeeper envía un evento **Disabled** indicando si todo ha ido bien y de los importes implicados en el proceso o el error que se ha producido.

6.5.1. Trabajar con distintas operaciones

[Funciones implicadas: AlternateOperation ValueIN, DiscardOperation]
CashKeeper es capaz de gestionar hasta 10 operaciones de cobro de forma simultánea. Esto significa que, con efectivo introducido, éste se puede 'aparcar' y gestionar otro cobro (hasta 10). Cuando se habilita el dispositivo por primera vez, la operación activa es la operación número 0 (cero). Para poder cambiar y aparcar la operación activa, se debe invocar el método **AlternateOperation**, donde le indicaremos la operación a activar.

Al llamar a este método, el valor de la operación actual queda almacenado y se activa la nueva operación, dejándola como operación actual.

Para consultar en cualquier momento si hay alguna operación pendiente, se puede invocar la función **ValueIN** (¡ATENCIÓN!, no confundir con el evento ValueIN), que nos devolverá el importe actual introducido en la operación especificada. Si no se especifica el número de operación, la función nos devolverá el importe introducido de la operación actual.

Hay ocasiones que nos puede interesar "olvidar" una operación con la consiguiente pérdida del importe introducido, esto se puede conseguir con la función **DiscardOperation**.

6.6. Realizar pagos *(proceso asíncrono)*

[Funciones implicadas: Pay]

6.6.1. Pago de un importe

Se deberá usar la función **Pay** para poder realizar pagos de importes con CashKeeper. CashKeeper se debe encontrar en estado IDLE para poder iniciar un pago o un test de pago.

Si CashKeeper es capaz de realizar el pago, la función devolverá verdadero y si '*TestOnly*' se ha establecido a falso se iniciará el pago, cambiando el estado del dispositivo a PAYING. A medida que vayan saliendo monedas y/o billetes, se desencadenará el evento **ValueOUT** informando del importe pagado hasta el momento. De la misma manera, cada vez que aparezca o desaparezca un billete en la boca de salida se desencadenará el evento **NoteHeldInBezel** *informandonos de la presencia o ausencia de billetes en la boca. Una* vez se haya completado el pago, se desencadenará el evento Disabled, indicando la finalización del proceso y dejando el estado de CashKeeper a IDLE.

6.6.2. Pagos con denominaciones específicas

[Funciones implicadas: PaySpecific]
Para realizar pagos con denominaciones específicas se deberá usar la función **PaySpecific**.
Las condiciones y el proceso que sigue un pago específico son las mismas que un pago estándar.

6.7. Operaciones de vaciado

6.7.1. Vaciado de cambio *(proceso asíncrono)*

Existen dos modalidades de vaciado del cambio hacia los cajones de recaudación.

6.7.1.1. Vaciado Parcial

[Funciones implicadas: EmptyDeviceSpecific]

La modalidad más comúnmente usada será la de vaciar los sobrantes de cambio. Esto es, enviar hacia los cajones de recaudación las monedas y/o billetes cuya cantidad actual en cambio esté por encima de los niveles deseados. Muy usado para restablecer el cambio inicial.

Para llevar a cabo este proceso, se deberá utilizar la función **EmptyDeviceSpecific**, durante el proceso, CashKeeper nos mantendrá informados a través del evento ValueToCashBox.

NOTA: Si alguna de las denominaciones presentes en cambio no se informa, CashKeeper enviará todas las unidades de dicha denominación hacia el cajón de recaudación. Si alguna de las cantidades especificadas de alguna de las denominaciones es superior al nivel actual disponible en cambio, se dejará al nivel disponible.

6.7.1.2. Vaciado total

[Funciones implicadas: EmptyDevice]

Para realizar un vaciado total de los importes de cambio hacia los cajones de recaudación, se deberá utilizar la función **EmptyDevice** indicando que dispositivo se quiere vaciar.

Una vez se haya iniciado el proceso de vaciado (parcial o total), el dispositivo pasará a estado FLOATING (0x04) o EMPTYING (0x05) respectivamente e irá informando a través del evento ValueToCashBox del importe enviado a cajón hasta el momento.

Una vez haya finalizado el vaciado (parcial o total) se desencadenará el evento Disabled indicando la finalización del proceso y dejando el dispositivo en estado IDLE.

6.7.2. Vaciado de cajones *(proceso síncrono)*

[Funciones implicadas: EmptyCashBox]

Aún y disponer de un sistema de detección de extracción de los cajones de recaudación, el vaciado de cajones, al ser una tarea completamente manual y al poderse realizar con el dispositivo apagado, se deberá informar a CashKeeper que dicha tarea se ha realizado. Para ello se deberá utilizar la función **EmptyCashBox**.

6.8. Llenado de cambio

Para poder llenar el dispositivo de cambio de manera efectiva, será necesario hacer uso de la función **ActivateRefillMode**. La llamada a esta función provocará que en la próxima función **Enable** (un Totalize cuando CashKeeper está en estado IDLE implica un enable a todos los efectos) y únicamente en el siguiente, todos los billetes y monedas introducidos se dispongan en los almacenes de cambio. En caso de no poder enviarlos a los almacenes de cambio, éstos serán devueltos. Hay dos excepciones referentes a los billetes, la primera es que el billete este en tan mal estado, que CashKeeper sea incapaz de devolverlo y la segunda, se produce cuando el reciclador de billetes está lleno.

6.9. Recoger la información del efectivo del sistema

Para poder usar todo el potencial de CashKeeper, se deberán conocer los importes que éste contiene tanto en los dispositivos de cambio como en los cajones de recaudación.

6.9.1. Efectivo disponible para cambio

[Funciones implicadas: GetCurrentLevel]

Para conocer los niveles disponibles para cambio, se debe usar la función **GetCurrentLevel**.

La información de niveles de CashKeeper se da en UNIDADES de la denominación, no en importes (25 monedas de 2 céntimos, no 50 céntimos).

6.9.2. Efectivo disponible en cajones de recaudación

[Funciones implicadas: GetCashBoxLevel]

Para conocer los niveles contenidos en los cajones de recaudación, se deberá recurrir a la función **GetCashBoxLevel**, hay que recordar, que como CashKeeper puede trabajar con distintas monedas, le es imposible valorar el cajón de recaudación, por lo que devuelve la cantidad, la denominación y la moneda del contenido del cajón de recaudación.

6.10. El 'quebranto' de moneda

[Funciones implicadas: GetBrokenCents]

En algunas divisas, CashKeeper no puede manejar todos los importes/denominaciones. Aún y la limitación del sistema de contener dichas denominaciones CashKeeper puede pagar prácticamente todos los importes excepto alguno MUY concreto, en esos casos, lo que hace es pagar un céntimo de mas. La aplicación puede enterarse de este hecho llamando la función **GetBrokenCents**.

Por ejemplo, en EUROS, CashKeeper no puede manejar la denominación de un céntimo. Aún y la limitación del sistema de contener monedas de 1 céntimo, existen solo dos importes que son completamente imposibles de ser pagados por CashKeeper: 0,01 € y 0,03 €.

Hay también otra posibilidad, existe dos propiedades *BCMaxCoins* y *BCMinValue* que permiten reducir el número de monedas en cambio a costa de "regalar" un céntimo de vez en cuando.

Por ejemplo: En una carnicería, los importes suelen ser altos a la vez que muy dispersos (no redondeados). Esto se traduce en un importante consumo de moneda pequeña.
Si establecemos BCMinValue = 1000 y BCMaxCoins = 4. Esto significa que en cualquier cobro de un importe igual o superior a 10 Euros en que la devolución del cambio implique más de 4 monedas, el sistema mira si devolviendo un céntimo de mas se reduce le número de monedas, si efectivamente se reduce, devolverá un céntimo de mas y establecerá el quebranto.
Supongamos una operación de 10,01 € que pagamos con un billete de 20,00€ y una moneda de dos céntimos con el objetivo de que el sistema nos devolviera un billete de 10,00€ y un céntimo.
En una situación estándar, el sistema devolvería el siguiente cambio: 1 x 5€, 2 x 2€, 1 x 0.50€, 2 x 0.20€, 1 x 0.05€, 3 x 0.02€ (que seria 10.01€)
Con la configuración descrita el sistema devolverá el siguiente cambio: 1 x 10€, 1 x 0.02€ y apuntará 1 céntimo de quebranto.

6.11. Trabajar con más de una divisa

CashKeeper permite mezclar en cobros billetes de distintas divisas (EUR, GBP, USD, etc.). El cambio siempre se devuelve en la divisa principal. Los pasos a seguir para utilizar esta funcionalidad son los siguientes:

- Disponer de una unidad preparada para trabajar con distintas denominaciones.
- "Habilitar" la divisa, para ello debemos suministrar el cambio respecto de la divisa principal mediante la función **SetCCChange.** Este valor, no se mantiene en la configuración, por lo es necesario informarlo cada vez que iniciamos CashKeeper.
- Habilitar las distintas denominaciones de dicha divisa.

6.12. Códigos de barras

CashKeeper permite leer códigos de barras en el modelo CK900/v, estos deben ser "Interleaved 2 of 5" y estar centrados en el papel (Consultar anexo 10 para mas detalles). Para habilitar la lectura, solo hay que informar la propiedad BarCodeLength con el número de caracteres del código.
Una vez CashKeeper lee un código de barras, dispara el evento BarCodeRead informándonos del código y debemos contestar al evento con la función RejectPending en caso de querer devolver el ticket o con la función AcceptPending en el caso de querer que envíe el ticket a cajón de recaudación. En este último caso, CashKeeper envía un último evento BarCodeStored indicándonos que se ha almacenado correctamente.

6.13. Otras funciones

6.13.1. Limpieza del validador de monedas

La función **CleanBulk** realiza una limpieza del validador de monedas. En función del parámetro complete, el proceso se puede demorar hasta 7 segundos.

6.13.2. Modificar los contadores de monedas en cambio

Mediante la función **ForceCoinLevel** se puede modificar los contadores de monedas que hay en cambio. Se utiliza en situaciones de substitución del reciclador de monedas.

6.13.3. Actualizar Firmware de los dispositivos

Mediante la función **CheckSmartFirmwareFile** podemos verificar si el archivo de actualización del que disponemos es compatible con el dispositivo que queremos actualizar, así mismo, nos indica también la versión.

Mediante la función **GetFirmWareVersion** podemos obtener las versiones que tenemos actualmente instaladas en los dispositivos

La función **UpdateSmartFirmware** realiza la actualización propiamente dicha.

ATENCIÓN: La actualización del firmware de un componente es una funcionalidad crítica además de larga de ejecución. Asegúrese de la estabilidad del sistema antes de proceder a actualizar un firmware ya que en caso de fallada (caída de la alimentación, desconexión, etc.) puede provocar que el componente quede totalmente INSERVIBLE e IRRECUPERABLE.

6.14. Cerrar el sistema

[Funciones implicadas: CloseAll, Disconnect, Terminate]

Podemos distinguir dos situaciones:

Cuando el CSSI se encuentra en el mismo ordenador en el que se esta ejecutando la aplicación se suele cerrar el CSSI y para ello se ejecuta la función **Terminate**.

Cuando se trata de distintos ordenadores, se suele dejar el CSSI abierto (ya que no se puede arrancar CSSI en un ordenador remoto), para ello se libera el dispositivo con la función **CloseAll** y a continuación se libera la conexión con la función **Disconnect**.

6.15. Parametrización del sistema

6.15.1. Inhibición de denominaciones

[Funciones implicadas: SetInhibitState, GetInhibitState

CashKeeper permite definir dentro de las denominaciones reconocidas, cuales serán aceptadas, cuales se aceptaran pero no se usaran en pagos y cuales no se aceptaran. Para ello, disponemos de las funciones **GetInhibitState** para obtener las inhibiciones activas y **SetInhibitState** para establecer la inhibición activa.

6.15.2. Protección de cambio

CashKeeper dispone de dos sistemas para protegernos de posibles faltas de cambio.

6.15.2.1. Avisos de nivel bajo

El sistema dispone de un evento **LowLevel** que nos informara cuando alguna denominación esta por debajo de los niveles recomendados. Se puede distinguir entre monedas y billetes.

Las monedas tienen un parámetro común para todas ellas (es importante intentar mantener un nivel equilibrado de monedas) que es la propiedad **CoinsLowLevel**, también puede dispararse (aún estando por encima de dicho nivel) si alguna denominación esta en franca desventaja respecto de las otras.

Los billetes se controlan exclusivamente por la función **SetLowLevelNotes**, en ella se especifica el nivel por cada denominación.

6.15.2.2. Note Level Protection

El Note Level Protection es un complejo sistema de protección de los billetes disponibles para cambio. El sistema trata de controlar el cambio a devolver en operaciones realizadas con billetes de valor alto.

¿Cómo funciona? Este sistema de protección se divide en 4 fases:

1. Reconocimiento del valor de billete
   Si el valor del billete introducido en una operación está por debajo del valor informado en la propiedad NLPStartValue, la operación proseguirá normalmente. En caso contrario se procede a la siguiente fase.

2. Validación del Total de la operación
   Si el valor total de la operación no ha sido informado (a través de la función 'Totalize(…)'), el billete se devuelve. Si el total SÍ ha sido informado, se procede a la siguiente fase.

3. Cambio suficiente
   Se determina el cambio teórico que se debería devolver en caso de terminar la operación en este momento. Si el cambio necesario esta por debajo del límite establecido con la propiedad **NLPPercentValue** se continúa el proceso, en caso contrario se procede a la siguiente fase.

4. Cambio a entregar fuera de límites establecidos. Si la propiedad **NLPAutoProtect** esta en verdadero, se rechaza el billete, en caso contrario, se genera el evento **ProtectedValueNote**. Al desencadenarse este evento, el billete queda retenido en el sistema de forma temporal hasta recibir una respuesta indicando que hacer con el billete. La respuesta puede ser '**AcceptPending**' para aceptar el billete o '**RejectPending**' si se desea rechazar.

6.15.3. Protección 'antiatraco'

CashKeeper dispone de un sistema de protección para evitar salidas masivas de efectivo por vía de pago.

Mediante las propiedades ***MaxPayout*** y ***PayoutInterval*** se podrá establecer un importe máximo de pago (*MaxPayout*) a poder pagar en un intervalo de tiempo (en minutos) determinado (*PayoutInterval*).

P.e. si establecemos MaxPayout = 10000 y PayoutInterval = 20, significa que durante 20 minutos puedo realizar tantos pagos como sean necesarios mientras la suma total de ellos no supere los 100 Euros. Una vez se haya llegado a los 100 Euros o el importe del próximo pago haga superar los 100 Euros, el sistema rechazara el pago con el código de error 4.

6.15.4. Otros parámetros y propiedades

6.15.4.1. DisableAutoText (Int32) *(propiedad de 'Aspecto')*

La propiedad '*DisableAutoText*' nos servirá para poder controlar el contenido del display o, dejar que el sistema controle de forma autónoma dichos contenidos.

(NOTA: los caracteres disponibles del display VFD serán 20 por línea cuando se utilice el tamaño de letra estándar y 20 si se utiliza el tamaño de letra doble)

Para tener acceso a especificar el contenido del display se utilizará la función **Display**.

6.15.4.2. LightTime (Int32) *(propiedad de 'Aspecto' y de 'Bloqueo')*

La propiedad *LighTime*, tiene una doble funcionalidad. Por un lado, nos permitirá especificar (en milisegundos) el tiempo que deseamos que el cajetín de recogida de monedas esté encendido en el momento de realizar un pago (sea vía cambio, rechazo de moneda o pago directo). Por otro lado, el tiempo especificado, será el mismo que CashKeeper estará 'bloqueado' después de realizar una salida de efectivo y antes de desencadenar el evento *Disabled*. Si *LighTime* se establece a cero, solamente se podrá provocar el evento *Disabled* de forma manual a través de la función *AbortTimer*. El valor por defecto es 1000 (1 segundo).

6.15.4.3. RejectIfClosed *(propiedad de 'Comportamiento')*

La propiedad *RejectIfClosed* determinará si la compuerta de rechazo del validador de monedas se deja en posición abierta, en situación de dispositivo deshabilitado (IDLE 0x01) o fuera de servicio (ALL_CLOSED 0x00).

6.15.4.4. CancelLowLevel *(propiedad de 'Comportamiento')*

Esta propiedad, determinará si el sistema, desencadena los eventos de aviso de nivel bajo de monedas y billetes.

6.15.4.5. CancelValueEvents *(propiedad de 'Comportamiento')*

Al igual que la anterior, CancelValueEvents determinará si el sistema desencadena los eventos ValueIN, ValueOUT y ValueToCashBox al recibir ingresos, realizar pagos o enviar importe a los cajones de recaudación respectivamente.

6.15.4.6. ConfigID *(propiedad de 'Identificación')*

ConfigID nos devuelve la configuración activa (la informada en el comando **Startup)**. Puede utilizarse para distinguir entre dispositivos cuando queremos controlar más de un dispositivo de forma simultánea.

6.15.4.7. LogDisable

Para habilitar o deshabilitar el registro de monedas y billetes entrantes.

- No implentada -

6.15.4.8. CoinCashBoxDetect

Esta propiedad nos permitirá habilitar o deshabilitar la detección de presencia del cajón de recaudación de monedas antes de realizar operaciones con el dispositivo. Valor por defecto 'Falso'=0.

6.15.5. Cambiar el idioma

Es posible cambiar el idioma con el que CashKeeper® se comunica con el usuario. Existen dos idiomas predefinidos que marcarán en qué lenguaje se enviarán las descripciones de los errores y avisos, así como el texto mostrado en el display de CashKeeper® cuando éste está configurado en automático.

Los dos idiomas predefinidos son el español (ESP) y el inglés (ENG). Por defecto el sistema está configurado en español. Para modificarlo se deberá usar la función **SetLanguage** donde se especificará el código el idioma deseado.

Aún y así, el sistema dispone de una funcionalidad para modificar a voluntad, el texto mostrado en el display de CashKeeper® cuando éste está configurado en modo automático. Para ello se deberá hacer uso de la

función **SetDisplayText**, donde se le deberá indicar el código del mensaje a modificar y el nuevo texto. Esta modificación no es permanente, por lo que cada vez que se inicialice el sistema se deberá volver a especificar.

6.16. Resolución de errores

Algunos de los errores proporcionados por CashKeeper, son errores críticos que limitan o finalizan el uso del sistema. En estos casos, se recomienda hacer uso de las funciones **Reset**, **CloseAll/StartUP** para poder solucionar el problema y restablecer el servicio ya que dichos errores, implican la desconexión de alguno de los dispositivos (validador de monedas, billetes, etc.) y consecuentemente la pérdida de comunicación con el Host.

Para evitar conflictos en las comunicaciones con el Host, antes de proceder a la desconexión de cualquier dispositivo se deben cerrar los puertos de comunicación a través de las funciones **CloseAll** y, una vez restituido el dispositivo y la alimentación, hacer uso de **StartUp**.

Por otra parte, algunos errores (identificados específicamente) requerirán el envío del comando ***Reset*** una vez se haya solucionado el problema.

6.17. Manejo de más de un dispositivo simultáneo

Es posible gestionar más de un dispositivo CashKeeper® de manera simultánea y conectados al mismo ordenador.
El primer paso, seria identificar cuantos y cuales son, Con esa finalidad existe la función **GetDevices** que devuelve una lista de los dispositivos (separados por coma) conectados al mismo tiempo que muestra en el display de cada uno de ellos su identificador correspondiente.
Una vez identificado el dispositivo a conectar, podremos conectarnos a él con la función **StartUp** especificando el identificador de dispositivo y operaremos con normalidad.
Cada CashKeeper con el que se quiera trabajar simultáneamente, necesita su propia instancia del objeto CKeeper así como su propio juego de puertos TCP/IP de conexión.

7. Diferencia integración CKeeper para Android y CKeeper de windows

7.1. Parámetros de salida de las funciones

La diferencia principal de utilizar la librería 'CKeeper para Android' con la librería CKeeper para windows, son los parámetros de salida.

Todas las funciones que tenían parámetros por referencia (Valores de salida), ahora devuelven un objeto de tipo 'CkResult'. Dicha clase, solo contiene el atributo **Response**. Atributo de solo lectura que informa de los parámetros de salida en la última función ejecutada.

Se obtienen los parámetros a través del método 'get' del atributo Response y el nombre del parámetro ('.get("nombre parámetro")').

El nombre parámetro va relacionado a la función ejecutada, siendo así un valor predefinido por la librería 'ckeeper.dll' explicada anteriormente.

El nombre del valor pasado por referencia de una función de la librería 'ckeeper.dll', coincide con el nombre del parámetro del método '.get' de la librería 'CKeeper para Android'.

Ejemplo:

Función en ckeeper.dll:

```
Function GetBrokenCents(ByRef Value As Long, ByVal resetValue As Boolean)
        As Boolean
```

En Android:

```
Public CkResult getBrokenCents(boolean resetValue) { ...}
ckSocket = ckConfiguration.getSocket();
CkResult ckRes = ckSocket.getBrokenCents(false);

Value val = ckRes.get("Value");
```

8. UTILIZAR *CashKeeper®con Método Directo*

A continuación se detallan los procedimientos a realizar para operar con ***CashKeeper®*** con el método directo:

8.1. Consideraciones previas

La operativa en el método directo es casi idéntica al método CKeeper, por lo que en caso de duda, consulte el apartado 6. Se distingue básicamente por algunos cambios en propiedades solo disponibles en el método Keeper y algunos métodos solo disponibles en el método directo.
En los anexos 3, 5 y 6 encontrarás la lista de funciones, propiedades y eventos con su correspondiente código

8.2. Formato de mensajes

8.2.1. Comandos

La estructura de los comandos enviados al servicio (CCSI) de CashKeeper® deberán tener el siguiente formato:

Empiezan por el carácter '$' ascii 36.
Seguido del código de mensaje.
En caso de tener parámetros, estos irán precedidos por la barra vertical '|' ascii 124
Terminan por el carácter '#' ascii 35.

Ejemplo
Enviar el comando **Terminate**.
$48#

Enviar un **Disable** con el parámetro payback a 1
$9|1#

8.2.2. Respuestas

La estructura de las respuestas a los comandos enviados al servicio tiene el mismo formato

Ejemplos:
La secuencia de comando y respuesta para habilitar el dispositivo (comando **Enable** 14) seria:

Enviamos:
← *$14#*
Recibimos:
… con respuesta afirmativa
→ *$14|1#*
…con respuesta negativa
→*$14| 0|37| Dispositivo no disponible#*

(con respuestas negativas, los parámetros que siguen a la respuesta son el código de error y la descripción del error)

Secuencia de comando y respuesta para solicitud de la cantidad disponible (**GetCurrentLevel** 19) en cambio de las denominaciones de 1.00€ y 2.00 €
Enviamos:
←*$19|100,200#*
Recibimos:
→*$19|1|30,25#*

(indica que tenemos 30 monedas de 1€ y 25 de 2€).

8.2.3. Eventos

La estructura de los eventos es propia para cada uno de los eventos, aún y siguiendo el formato estándar establecido para todas las comunicaciones:

El siguiente ejemplo de evento STATECHANGE nos indica que estamos en estado **ENABLED**
→*$106|2#*

8.3. Puesta en marcha *(proceso síncrono)*

Para la puesta en marcha del sistema, primero se deberá iniciar la aplicación del servicio (CKeeper.exe) y abrir un canal de comunicación con el servicio (CSSI). Para ello antes se deberá haber decidido si el tipo de comunicación es para tener un control completo del dispositivo (conexión tipo HOST) o, solamente, para funciones de consulta y configuración (conexión tipo OFFICE). La diferencia radica en el puerto al que deberemos apuntar, ya que hay un puerto específico de conexión para cada uno de los tipos. Una vez establecida la comunicación se deberá de negociar la clave de acceso al servicio.

8.3.1. Negociar la clave de acceso al servicio

(NOTA: este es el único procedimiento que no sigue el estándar de comunicación COMANDO → RESPUESTA COMANDO)
Para negociar la clave de acceso al servicio, una vez establecida la comunicación con el servicio, se deberá enviar el comando 47 **(StartCNT)** para indicar al servicio el inicio de la negociación:
←*$47#*

A su vez, el servicio responderá con el comando 57 **(CNT)** al que asociará un parámetro adicional, que será la base de cálculo para la clave de acceso:
→*$57|número#*

Ahora, con el número que nos ha proporcionado el servicio, deberemos calcular la clave, sumándole el valor del parámetro 'Security Seed' (el mismo que habremos configurado en la aplicación del servicio). P.e. 140806.
Clave de acceso = 140806 + número (supongamos que número = 1620168189)
Clave de acceso = 1620308995

Ahora enviaremos la clave de acceso resultante al servicio, con **el comando 57 (CNT)**:
←*$57|**1620308995#***

Si el valor de la clave concuerda con el que el servicio ha calculado (conociendo previamente el valor de 'Security Seed' o 'Semilla de Seguridad'), el servicio contestará con el comando 58 **(CNT_OK)**: y el número de versión de comunicaciones.
→*$58|5#*

En caso contrario el servicio se limitará a cerrar la comunicación.

Resumen de pasos para el inicio de comunicación con el servicio (CCSI):

1. Ejecutar aplicación del servicio (CKeeper.exe)
2. Abrir canal de comunicación por sockets (protocolo TCP/IP) sobre el puerto dedicado a conexiones tipo HOST o sobre el puerto dedicado a conexiones tipo OFFICE. Por defecto, los puertos para conexiones tipo HOST es el **8001**, para conexiones tipo OFFICE el puerto **8002**.
3. Negociar la clave de acceso. Por defecto el parámetro 'Security Seed' (semilla de seguridad) del servicio está establecida en **100001**.

Una vez establecida la comunicación y negociada la clave de acceso al servicio, podremos iniciar la comunicación con el dispositivo CashKeeper®. Para ello se deberá llamar el comando 39 **(START_UP)**:
←*$39|0#*

Si la respuesta es satisfactoria, el estado de CashKeeper pasará de ALL_CLOSED (0x00) a IDLE (0x01). Dicho cambio de estado se nos comunicará mediante la transmisión del evento STATE_CHANGE (código 106).

8.4. Cerrar el sistema

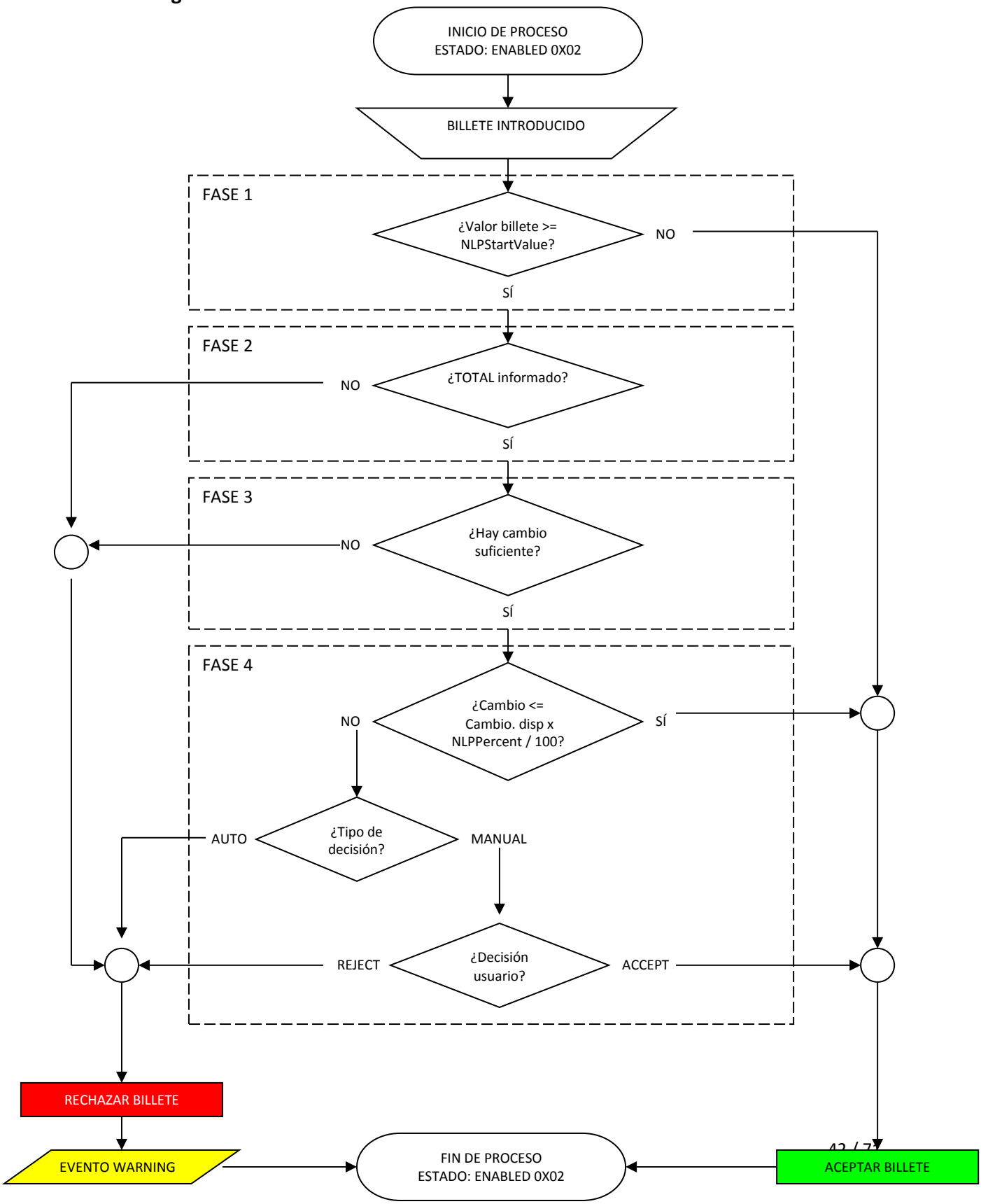
[Comandos implicados: 7 – CloseAll, 48 - Terminate,
Para cerrar las comunicaciones con CashKeeper y cerrar el servicio, se deberá utilizar el comando **Terminate** (48). Esto cerrará los puertos de comunicaciones con CashKeeper dejándolo en estado ALL_CLOSED (0x00) y seguidamente cerrará el CSSI.
Para cerrar las comunicaciones con CashKeeper y mantener el servicio en marcha para una posterior conexión, se deberá utilizar el comando **CloseAll (7)**. Esto cerrará solo los puertos de comunicaciones con CashKeeper dejándolo en estado ALL_CLOSED (0x00) .

ATENCIÓN: El servicio está programado para que, en caso de una pérdida de comunicación con el cliente (socket), cerrará las comunicaciones con el dispositivo CashKeeper® sea cuál sea el estado de éste.

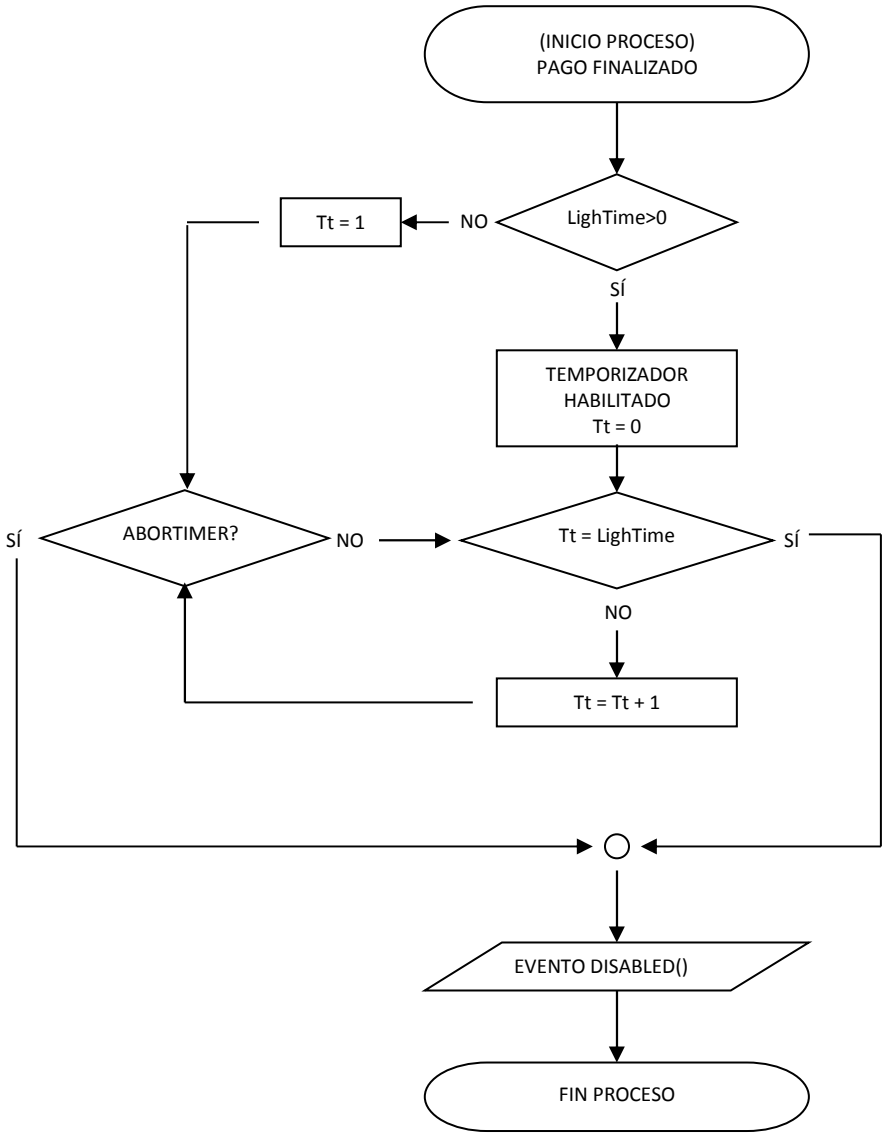
NOTA: Aún y devolver un error del comando CloseAll (7), el servicio procurará por todos los medios cerrar la comunicación con el dispositivo y, por tanto, el cambio de estado a ALL_CLOSED (0x00) siempre será efectivo.

ANEXO 0. Imágenes

- **Figura 1**



- **Figura 2**


(INICIO PROCESO)
PAGO FINALIZADO
LighTime>0
NO
Tt = 1
SÍ
TEMPORIZADOR
HABILITADO
Tt = 0
Tt = LighTime
SÍ
NO
ABORTIMER?
SÍ
NO
Tt = Tt + 1
EVENTO DISABLED()
FIN PROCESO

ANEXO 1. Descripción de constantes

LC_DeviceType:

- *0 = LCDT_900*
- *1 = LCDT_1000*

LC_Smart_Devices:

- *0 = LCS_CoinHopper*
- *1 = LCS_NotePayout*

LC_CashBox_State:

- *0 = LCCBS_OK*
- *1 = LCCBS_Almost_Full*
- *2 = LCCBS_Full*

LC_Smart_Devices:

- *1 = LCSD_CoinHopper*
- *2 = LCSD_NoteFloat*
- *3 = LCSD_NotePayout*

LC_Logical_Devices:

- *1= LCLD_CoinValidator*
- *2 = LCLD_CoinDispenser*
- *3 = LCLD_NoteRecycler*

LC_CashBox:

- *1= LCCB_Coins*
- *2 = LCCB_Notes*
- *5 = LCCB_All*

LC_ConnTypes:

- *0 = HostMachine*
- *1 = BackOffice*

Idiomas:

- *1 = Español*
- *2 = English*

ANEXO 2. Tablas de errores y warnings

ERRORES

Código	Mensaje
1	Comunicación no iniciada
2	Dispositivo ocupado. No se puede procesar el comando
3	No es posible habilitar el dispositivo
4	Bloqueo de seguridad (MaxPayout excedido)
5	Intento de fraude
6	Dispositivo fuera de servicio
7	Moneda atascada en dispositivo de salida
8	Billete atascado en zona segura
9	Billete atascado en zona no segura
10	Billete atascado
11	Dispositivo/s no conectado/s
12	Error al cargar datos de configuración
13	Error en datos de salida
14	Error de comunicación sockets
15	Error al abrir puertos COM
16	Error al cerrar puertos COM
17	Error al resetear dispositivos
18	Detectado reset del dispositivo inesperado
19	Error de configuración de puertos
20	Error al inicializar dispositivos
21	Límite de capacidad del cajón de monedas excedido
22	Cajón de recaudación de billetes lleno
23	Pago rechazado por el operador
24	El valor debe estar comprendido entre 100.000 y 9.999.999
25	Error al habilitar dispositivos
26	No se encuentra CKeeper. Imposible iniciar comunicación con CashKeeper
27	Tiempo excedido para encontrar las monedas necesarias para efectuar el pago. Reintente el pago
28	Error al verificar el valor del billete al efectuar el pago. Reintente el pago.
29	Error de comunicación recuperado
30	Error de comunicación con dispositivo
31	Error interno del dispositivo
32	No hay importe suficiente para realizar el pago
33	Cajón de recaudación de billetes quitado
34	Valor de Device_ID no soportado
35	Límite de tiempo excedido para la ejecución del comando
36	El dispositivo no está habilitado. No se puede procesar el comando
37	Dispositivo no disponible. No se puede procesar el comando
38	Modo no soportado
39	Valor no soportado
40	Valores no soportados: número de denominacions y valores distintos
41	El dispositivo ya está procesando este comando

42	Se produjo un error inesperado al ejecutar el comando
43	Drivers del dispositivo incorrectos
44	Error de lectura EEPROM
45	No hay registros de log de entrada de monedas y billetes
46	Propiedad de solo lectura
47	La configuración no existe. Debe especificar un device para configurar
48	El dispositivo especificado no existe
49	El dispositivo ya está inicializado
50	Error genérico del sistema
51	No se soporta este código de país
52	El canal de salida de billetes del reciclador está abierto
53	Ruta invalida o faltan privilegios
54	Se ha quitado el cajón de recaudación de monedas
55	Error de calibración del reciclador de monedas
56	Uso de caracteres reservados (# | $) en envío de datos a DISPLAY
57	Código de país no soportado
58	Billete en salida de cambio pendiente de ser retirado
59	Reciclador de monedas por encima de su capacidad, realice pagos o bien vacíe parcial o totalmente.
60	La tapa superior del reciclador de monedas está abierta
61	No se puede determinar el tipo de dispositivo
62	La compuerta del validador de monedas, no puede cerrarse
63	Fichero de firmware corrompido.
64	Fichero de firmware incorrecto para este dispositivo.
70	Tiempo de pago excedido
90	Error de comunicación con CashKeeper (CSSI)
92	Límite de tiempo excedido para recibir respuesta de CashKeeper (CSSI)

WARNINGS

Código	Mensaje
1	Posible atasco en el cajón de recaudación de billetes
2	Se ha aceptado una moneda desconocida
3	La última operación de recaudación no finalizó correctamente
4	Detectado error en validador de monedas. Existe la posibilidad de monedas no contabilizadas
5	Intento de fraude recuperable
6	Billete pagado al iniciar el dispositivo
8	Billete rechazado por activación de protección de cambio: Límite de cambio superado
9	Billete en salida de cambio pendiente de ser retirado
10	Billete rechazado por activación de protección de cambio: Total no informado
11	Billete rechazado por activación de protección de cambio: No hay cambio suficiente
12	El reciclador de monedas está lleno

13	El cajón de recaudación de monedas está lleno
14	El cajón de recaudación de monedas está al 90% de su capacidad
15	La tapa superior del reciclador de monedas está abierta
16	La compuerta del validador de monedas está obstruida y no puede cerrarse
17	Atasco de monedas en zona de validación o el sensor necesita limpieza
18	Sensor de monedas bloqueado o sucio
19	Disco del validador de monedas atascado
20	Se enviarán a cajón mas billetes de los pedidos
21	Reciclador de billetes no operativo, NO se pagarán billetes
22	El cajón de recaudación de billetes está lleno
23	El cajón de recaudación de billetes está al 90% de su capacidad
24	Vaciando el reciclador de billetes para poder usar la unidad
25	El cajón de recaudación de billetes ha sido retirado
26	El cajón de recaudación de monedas ha sido retirado
32	Error al guardar importes de cajón
33	Error al devolver cambio automático : [razón]
34	Reciclador de billetes lleno, los siguientes billetes irán a cajón
35	Error inesperado en lectura de billete
36	Pago rechazado por el operador
50	Se ha detectado una posible anomalía. Verifique el contenido del cajón de recaudación de billetes.
55	Error de calibración del reciclador de monedas
108	Billete atascado en zona segura
109	Billete atascado en zona no segura
110	Billete atascado
111	Lector de billetes no funciona
112	Validador de monedas no operativo

ANEXO 3. Listado de funciones. Definición

NOTA: Todos los valores relativos a valores de denominación o importes se deben enviar y se devuelven en céntimos.

- **(0) AbortTimer**
  Envía una orden para cancelar el tiempo de 'espera' establecido para la recogida de cambio (propiedad *LighTime*).

- **(3) AcceptPending**
  Después de desencadenarse alguno de los eventos *ProtectedValueNote*, *MaxCoinsWarning*, *BarCodeRead* se deberá responder a dicho evento con *AcceptPending* para aceptar o con *RejectPending* para rechazarlo.

- **(67) ActivateRefillMode() as boolean**
  Este método habilita el modo de 'llenado de cambio' únicamente durante el próximo 'Enable()' (sea directo o indirecto vía 'Totalize'). Este modo provocará que TODOS los billetes entrantes se envíen al almacén de cambio o sean devueltos.

- **(4) AlternateOperation (*Operation as Byte*) as boolean**
  Permite alternar la operación actual con otra, dejando aparcados los valores introducidos de la operación vigente. El número de operación activo cuando se inicia el sistema es '0'.

- **(6) CleanBulk (*Complete as* boolean) as boolean**
  Desencadena un ciclo de limpieza del dispositivo de entrada de monedas.

  Complete:
  - falso (ciclo de limpieza corto)
  - verdadero (ciclo de limpieza completo)

- **(7) CloseAll ( ) as boolean**
  Cierra los canales de comunicación con el dispositivo. Aunque la consecución del cierre dé como resultado un error, se considerará, a todos los niveles, que los canales han sido cerrados.

- **(5) CheckLevels (LowLevelActive As boolean, HopperFull As boolean, CoinCashBoxState As LC_CashBox_State, NoteCashBoxState As LC_CashBox_State) as boolean**
  Realiza un chequeo general de los niveles de monedas y billetes en cambio y de los niveles de cajones, devolviendo los siguientes valores sobre cada uno de los parámetros:

Parámetro	Valor	Significado
LowLevelActive	falso	No hay denominaciones en estado de alarma por nivel bajo
LowLevelActive	Verdadero	Existen denominaciones en estado de alarma por nivel bajo. Consultar niveles a través de la función **GetCurrentLevel**
HopperFull	Falso	Nivel del reciclador de monedas OK
HopperFull	Verdadero	Reciclador de monedas lleno. Debe ser vaciado almenos parcialmente.
CoinCashBoxState	LCCBS_OK	Nivel del cajón de recaudación de monedas OK
CoinCashBoxState	LCCBS_AlmostFull	Nivel del cajón de recaudación de monedas superior al 90%
CoinCashBoxState	LCCBS_Full	Nivel del cajón de recaudación de monedas excedido. Debe ser vaciado.
NoteCashBoxState	LCCBS_OK	Nivel del cajón de recaudación de billetes OK
NoteCashBoxState	LCCBS_AlmostFull	Nivel del cajón de recaudación de billetes superior al 90%
NoteCashBoxState	LCCBS_Full	Nivel del cajón de recaudación de billetes excedido. Debe ser vaciado.

- **(55) CheckSmartFirmwareFile (FullPathFile as string, Device_ID as LC_Smart_Devices, FirmwareVersion as string, DataSet as string) as Boolean**
  Chequea la versión de firmware y de dataset contenidos en el fichero especificado en '*FullPathFile*', así como la compatibilidad con el '*Device_ID*' especificado. Cabe remarcar, que el path es relativo a la ubicación del CSSI.

- **(9) Disable(PayBack As Boolean) As Boolean**
  Deshabilita el dispositivo dejándolo en un estado de reposo (no apto para la entrada de efectivo) devolviendo, en su caso, el importe introducido hasta el momento.

Parámetro	Valor	Significado
PayBack	Falso	No devuelve el importe introducido.
PayBack	Verdadero	Devuelve el importe introducido.

- **(52) DiscardOperation(Operation as byte) as Boolean**
  Descarta la operación indicada, dejando los valores pendientes de dicha operación a 0 (cero). ¡ATENCIÓN! Esta función no provoca la devolución del importe introducido.

- **(68) DiscardPayOperation() as Boolean**
  Descarta la operación de pago interrumpida. ¡ATENCIÓN! Esta función no provoca la devolución de ningún importe.

- **Disconnect()**
  Desconecta la conexión con CashKeeper. Si el estado es diferente a ALL_CLOSED, el CSSI ejecutará un **CloseAll** automáticamente.

- **(10) Display(*Line1 as String, Line2 as String, UseBigFont*) as Boolean**
  Envía un mensaje al display VFD del dispositivo. Cada una de las líneas no puede contener más de 20 caracteres. En caso de que UseBigFont (texto de altura doble) se establezca a verdadero el texto contenido en Line2 no se considerará.
  NOTA: Los caracteres '#', '|' y '$' son reservados y su uso está restringido

- **(11) EmptyCashBox(*Device_ID as LC_CashBox*) as Boolean**
  Resetea (=0) el contador del valor almacenado en el cajón de recaptación del dispositivo identificado en *Device_ID*. Se debe utilizar cada vez que se vacían los cajones de recaptación.
- **(11) EmptyCashBoxEx(CashBox As LC_CashBoxes) As Boolean**
  Extensión de la función (11) EmptyCashBox, que permite vaciar los cajones de recaudación de billetes individualmente. (Solo funcional en la CK1000)

- **(12) EmptyDevice(*Device_ID as LC_ CashBox*) as Boolean**
  Vacía completamente el cambio hacía el cajón de recaptación del dispositivo identificado en *Device_ID*.

- **(13) EmptyDeviceSpecific(*Denoms as String, NumberToKeep as String*) as boolean**
  Envía hacia el cajón la cantidad que corresponda de cada una de las denominaciones especificadas en la lista '*Denoms*' (separadas por comas) para dejar en cambio los valores correspondientes especificados en la lista '*NumberToKeep*' (separadas por comas respectivamente).

- **(14) Enable() as Boolean**
  Habilita el dispositivo para dejarlo en estado de recepción de efectivo.

- **(15) ForceCoinLevel(Value as Integer, Level as Integer, AddToCurrentLevel As Boolean) as Boolean**
  El dispositivo de almacenamiento y entrega de monedas, en casos excepcionales puede ser cargado masivamente por vía superior (sin usar el dispositivo de entrada de monedas). Solo en estos casos, será necesaria la utilización de este método, para informar al dispositivo de la cantidad de cada una de las denominaciones introducidas.

  ATENCIÓN: Un uso irresponsable de esta función puede conllevar problemas en el recuento y pago de monedas.

- **(66) GetAllProperties**
  Devuelve todas las propiedades en el siguiente orden:
  BCMAXCOINS
  P_BCMINVALUE
  CANCELLOWLEVEL
  CANCELVALUEEVENTS
  CONFIGID
  COINCASHBOXDETECT
  DISABLEAUTOTEXT
  LIGHTTIME
  LOGDISABLE
  MAXCOINS
  MAXPAYOUT
  MINFASTIN
  NLPAUTOPROTECT
  NLPPERCENTVALUE
  NLPSTARTVALUE
  NOTELEVELPROTECTION
  PAYOUTINTERVAL
  REJECTIFCLOSED
  COINSLOWLEVEL
  DEVICETYPE
  BARCODELENGTH
  CRITICALBEHAVIOR

- **(16) GetBrokenCents(*Value as INT32, ResetValue as Boolean)* as boolean**
  Almacena en la variable *Value* la cantidad de céntimos de más que se han devuelto en cambio debido a los redondeos al alza de los valores de 1 y 3 céntimos. El parámetro ResetValue, indica si se quiere resetear el contador.

- **(17) GetCashBoxLevel(CountryCode As String, Values As String, Levels As String) As Boolean**
  Devuelve sobre la variable Levels, el nivel contenido en el cajón de recaudación la/s denominación/es especificada/s en Values. TODOS los parámetros son exclusivamente de retorno.

CountryCode: Moneda a la que haran referencia los valores (EUR, GBP, USD, etc.).
Values: Valor de la denominación, se pueden especificar varias separadas por coma.
Levels: Cantidad de la denominación pedida, en caso de ser varias, devuelve una lista de cantidades separadas por coma en el mismo orden que las pedidas.

- **(77) GetCCChange(CountryCode As String, Change As Int32) As Boolean**
  Devuelve sobre la variable *Change*, el importe cambio de la moneda *CountryCode* sobre la moneda principal.

- **(86) GetCCDetails(CC As String, Multiplier As int32, Decimals As int32) As Boolean**
  Devuelve el multiplicador y el número de decimales a mostrar. Permite hacer que nuestra aplicación se adapte a la moneda de otros países. Por ejemplo:
  Euro, el multiplicador es 100, ya que los importes son en céntimos, el número de decimales a mostrar es 2.
  Moneda Chilena, el multiplicador es 1, ya que no utilizan decimales, el número de decimales a mostrar es 0.

- **(78) GetCCDenominations(CountryCode As String, Coins As String, Notes As String) As Boolean**
  Devuelve la lista de valores de las distintas denominaciones de una moneda, depositando en Coins la lista de valores de las monedas separadas por coma y en *Notes* la lista de valores de las denominaciones en billetes separadas por coma.

- **(24) GetCounters(*CoinCounter as Int32, NoteCounter as Int32*) as Boolean**
  Devuelve sobre las variables *CoinCounter* y *NoteCounter*, el valor de los contadores absolutos (número total de unidades de entrada) de monedas y billetes desde la fabricación del dispositivo.

- **(38) GetCountryCodes(MainCC As String, OtherCC As String) As Boolean**
  Devuelve en la variable *MainCC* el código de moneda principal (EUR, GBP, MXN, USD, etc.) y en variable ***OtherCC*** la lista de las demás monedas soportadas separadas por coma.

- **(19) GetCurrentLevel(Values as String, Levels as String) as Boolean**
  Devuelve sobre la variable *Levels*, la cantidad contenida en cambio de la/s denominación/es especificada/s en la variable Values separadas por coma.
  Atención: En caso de que algún dispositivo esté en error, el contenido de dicho dispositivo puede no mostrarse.

- **(72) GetDevices(DeviceList As String) As Boolean**
  Devuelve la lista de identificadores de CashKeeper conectados en el ordenador donde reside el CSSI.

- **(54) GetFirmwareVersion(Device as LC_Logical_Devices, FirmwareVersion as string, Dataset as string) as Boolean**
  Devuelve sobre las variables *FirmwareVersion* y *Dataset* la versión de firmware y el dataset actuales del dispositivo especificado en Device.

- **(21) GetInhibitState(CountryCode As String, Values As String, Inhibits As String) As Boolean**
  Devuelve sobre la variable *Inhibit*, una lista de estados de aceptación o inhibición de la lista de denominaciones especificadas en *Values* de la moneda *CountryCode*.

  Inhibit = 0 : la denominación se acepta.
  Inhibit = 1 : la denominación se acepta pero no se paga.
  Inhibit = 2 : la denominación se rechaza.

- **(76) GetLastIN(CountryCodes As String, AmountsOrDenom As String, Detail As Boolean, Operation As Byte) As Boolean**
  Obtiene la forma de pago utilizada en el último cobro realizado devolviendo la lista en *AmountsOrDenom,* el parámetro *Detail* se utiliza para seleccionar el modo siendo verdadero detallado y falso acumulado.
  Acumulado: Devuelve el importe total en cada divisa utilizada en el pago.
  Detallado: Devuelve la lista de denominaciones utilizadas en el pago, en caso de ser superior a 50 denominaciones, devuelve el valor acumulado y modifica el parámetro *Detail*.

- **(79) GetLastLevels(ConfigID As Byte, Denoms As String, Qtys As String, CCs As String) As Boolean**
  Obtiene el ultimo nivel en cambio de todas las denominaciones. Se utiliza para saber el último nivel conocido de cambio en caso de error en la función startup.

- **(8) GetLowLevelNotes(Values As String, Levels As String) As Boolean**
  Devuelve sobre el parámetro *levels* la lista de niveles mínimos (separados por coma) de la lista de denominaciones proporcionada en el parámetro *Values*.

- **(23) GetMaxLevel(Values as String, Levels as String) as Boolean**
  Devuelve sobre el parámetro *levels* la lista de niveles máximos (separados por coma) de la lista de denominaciones proporcionada en el parámetro *Values*.

- **(82) GetNetworkParams(HostName As String, DHCPEnabled As Boolean, IP As String, Gateway As String, Mask As String, DNS1 As String, DNS2 As String, MasterPort As int32, OfficePort As int32, Seed As int32) As Boolean**
  Devuelve los valores de la configuración de red.

- **(74) GetUnikeID (ID as string) as Boolean**
  Devuelve sobre el parámetro *ID* un identificador único para el dispositivo CashKeeper®.

  NOTA: Este identificador puede variar en función de cambios en el hardware

- **(29) Pay(Value as Int32, TestOnly as Boolean) as Boolean**
  Desencadena un proceso de salida de efectivo por el valor especificado en *Value*. La repartición de denominaciones en monedas y billetes se realiza de forma automática.

  Si la variable *TestOnly* se establece a verdadero solo se realizará un testeo de si es posible pagar dicha cantidad.

- **(30) PaySpecific(Denoms as String, NumberOf as String, TestOnly as Boolean) as Boolean**
  Desencadena un proceso de salida de efectivo con las denominaciones especificadas en 'Denoms' (separadas por comas) en las cantidades correspondientes de cada una de ellas especificadas en 'NumberOf' (separadas por comas respectivamente).

  Si la variable TestOnly se establece a verdadero solo se realizará un testeo de si es posible pagar dichas cantidades.

- **(87) PaySpecificEx(Denoms as String, NumberOf as String, TestOnly as Boolean) as Boolean**
  Desencadena un proceso de salida de efectivo con las denominaciones especificadas en 'Denoms' (separadas por comas) en las cantidades correspondientes de cada una de ellas especificadas en 'NumberOf' (separadas por comas respectivamente).

  Si la variable TestOnly se establece a verdadero solo se realizará un testeo de si es posible pagar dichas cantidades.
  Esta función se diferencia de la anterior en que se tiene en cuenta las propiedades MaxPayout y PayoutInterval.

- **(32) RejectPending()**
  Después de desencadenarse alguno de los eventos *ProtectedValueNote*, *MaxCoinsWarning*, *BarCodeRead* se deberá responder a dicho evento con *AcceptPending* para aceptar o con *RejectPending* para rechazarlo.

- **(33) Reset( ) as Boolean**
  Resetea el dispositivo y lo reinicializa.

- **(26) ResetCounters(Device_ID as LC_CashBox) as Boolean**
  Resetea el contador absoluto de monedas o billetes.

- **(75) SetCCChange(CountryCode As String, Change As Int32) As Boolean**
  Establece el cambio de la divisa especificada en *CountryCode* respecto de la divisa principal. La formula para calcular el cobro ((*Change* * Valor billete) / 1000) sin decimales.

- **(81) SetDateTime(Year As Integer, Month As Integer, Day As Integer, Hour As Integer, Minute As Integer, Second As Integer) As Boolean**
  Establece la fecha y la hora en los dispositivos equipados con una tarjeta SmartCK (modelos CK900e y CK1000). En el modelo USB, esta función devuelve verdadero y se ignora.

- **(71) SetDisplayText(Code as byte, NewText as string) as Boolean**
  Modifica el texto de display automático identificado con la variable *Code* por el texto enviado en *NewText*.

  P.E.
  Originalmente el código 17 de texto de display contiene el texto 'A PAGAR:', si este texto lo quisiéramos modificar por, p.e., 'TOTAL :', deberíamos invocar la función de la siguiente manera…

  SetDisplayText (17, 'TOTAL :')

- **(36) SetInhibitState(CountryCode As String, Values As String, Inhibits As String) As Boolean**
  Establece el estado de inhibición (permiso de aceptación) de las denominaciones especificadas en *CountryCode* / *Values*.

  Inhibit = 0 : la denominación se acepta
  Inhibit = 1 : la denominación se acepta, pero no se paga.
  Inhibit = 2 : la denominación se rechaza.

- **(70) SetLanguage(Idioma As Idiomas) As Boolean**
  Establece el idioma por defecto que utilizará CashKeeper® para comunicarse vía display y mensajes de texto. Los idiomas disponibles son español (ESP) e inglés (ENG).

- **(69) SetLogPath(NewPath as String) as Boolean**
  Establece el path de almacenamiento de los archivos de LOG a la ruta indicada de forma permanente. Este path debe ser relativo al equipo donde se está ejecutando el CSSI.

- **(27) SetLowLevelNotes(Values As String, Levels As String) As Boolean**
  Asigna el nivel mínimo por cada denominación de billete a partir del cual, el sistema empezara a avisar de nivel bajo.
  El parámetro *Values* se informará con uno o varios valores separados por coma de las denominaciones a las que queremos informar y el parámetro *Levels* contendrá tantos valores separados por coma como valores tenga el parámetro *Values*.

- **(35) SetMaxLevel(Values As String, Levels As String) As Boolean**
  Asigna el nivel máximo por cada denominación de moneda/billete a partir del cual, el sistema empezara a enviar a cajón de recaudación por exceso de cambio. En el modelo CK1000, El nivel máximo de los billetes esta forzado a 26.
  El parámetro *Values* se informará con uno o varios valores separados por coma de las denominaciones a las que queremos informar y el parámetro *Levels* contendrá tantos valores separados por coma como valores tenga el parámetro *Values*.

- **(83) SetNetworkParams(HostName As String, DHCPEnabled As Boolean, IP As String, Gateway As String, Mask As String, DNS1 As String, DNS2 As String, MasterPort As int32, OfficePort As int32, Seed As int32) As Boolean**
  Assigna la configuración de red.

- **(39) StartUp(Configuration As Byte, Device as Int32) As Boolean**
  Es el primer método al que se debe llamar una vez iniciada la conexión. Abre los puertos de comunicación, carga datos de configuración e inicializa el dispositivo.
  Configuration: Identificador de la configuración que queremos iniciar, en caso de no existir, se crea.
  Device: Parámetro opcional, que indica que dispositivo queremos iniciar con esa configuración. Solo es obligatorio si el Ordenador que ejecuta el CSSI tiene más de un CashKeeper conectado físicamente.
  Como resultado en el método directo, devuelve lo mismo que el comando **GetAllProperties**, ya que lo más común al iniciar, es averiguar la configuración que se ha cargado.

- **(51) TargetValue(Value As Long, Operation As Byte) As Boolean**
  Obtiene el Target de la operación en pedida.

- **(48) Terminate**
  Este método es equivalente a ejecutar **CloseAll** y **Disconnect**, con la particularidad de que si el CSSI esta en la misma máquina que la aplicación, cierra el CSSI.

- **(42) Totalize(Total_Value as Int32, AutoClose as Boolean) as Boolean**
  Establece el total de una operación para informar al cliente el valor que debe hacer efectivo en la operación actual.

  Si se establece el parámetro AutoClose a verdadero, al detectar que hay importe suficiente, se devuelve el cambio de forma automática. Si se invoca en estado IDLE (0x01), el dispositivo se habilita automáticamente.

- **(56) UpdateSmartFirmware(FullPathFile as string, Device_ID as LC_Smart_Devices) as Boolean**
  Verifica y actualiza el firmware del dispositivo especificado con el archivo de actualización indicado.

- **(43) ValueIN(Value as Int32, Operation as Byte) as Boolean**
  Devuelve el valor total introducido en la operación especificada en el parámetro *Value*. La operación 255 es equivalente a la operación activa actual.

- **(44) ValueOUT(Value as Int32) as Boolean**
  Devuelve el valor total pagado hasta el momento en el parámetro *Value*.

- **(45) ValueToCashBox(Value as Int32]) as Boolean**
  Devuelve el valor total enviado a cajón hasta el momento en el parámetro Value.

ANEXO 4. Listado de funciones por ejecución SÍNCRONA o ASÍNCRONA.

Listado de funciones SÍNCRONAS

Las funciones síncronas son aquellas que el resultado devuelto implica la finalización de la tarea de la función, por lo que no habrá que esperar el desencadenamiento de ningún evento para dar por finalizado dicho proceso.

Función	Comando	Estados en los que es posible ejecutarla
AbortTimer	0	0x01, 0x02, 0x07
AcceptPending	3	Al recibir un evento *ProtectedValueNote, MaxCoinsWarning, BarCodeRead*.
ActivateRefillMode	67	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
AlternateOperation	4	0x01, 0x02
CheckLevels	5	0x01
CheckSmartFirmwareFile	55	0x01
CleanBulk	6	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
CloseAll	7	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
CNT*	57	0x00
CNTOK*	58	0x00
DISCARD_PAY_OPERATION*	68	0x01
DiscardOperation	52	0x01
Display	10	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
EmptyCashBox	11	0x01
Enable	14	0x01, 0x02
ForceCoinLevel	15	0x01
GET_PROPERTY*	63	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
GET_PROTOCOL_VERSION*	62	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
GETALLPROPERTY	66	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
GetBrokenCents	16	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
GetCashBoxLevel	17	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
GetCCChange	77	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
GetCCDenoms	78	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
GetCounters	24	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
GetCountryCodes	38	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
GetCurrentLevel	19	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
GetDevices	72	0x00
GetFirmwareVersion	54	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
GetInhibitState	21	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
GetLastIN	76	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
GetLowLevelNotes	8	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
GetMaxLevel	23	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
GetUnikeId	74	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
RejectPending	32	Al recibir un evento *ProtectedValueNote, MaxCoinsWarning, BarCodeRead*.

Reset	33	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
ResetCounters	26	0x01
SET_PROPERTY*	60	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
SetCCChange	75	0x01
SetDisplayText	71	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
SetInhibitState	36	0x01
SetLanguage	70	0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
SetLogPath	69	-todos- (0x00 a 0x07)
SetLowLevelNotes	27	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
SetMaxLevel	35	0x01
SetRefillMode	67	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
START_CNT*	47	0x00
StartUp	39	0x00
State	40	-todos- (0x00 a 0x07)
TargetValue	51	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
UpdateSmartFirmware	56	0x01
ValueIN	43	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
ValueOUT	44	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07
ValueToCashBox	45	0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07

*Solo disponibles en método directo

Listado de funciones ASÍNCRONAS

Las funciones asíncronas son aquellas que el resultado devuelto NO implica la finalización de la tarea de la función. El resultado devuelto solo implica la disponibilidad del sistema para iniciar dicha tarea, así como la comprobación de que los parámetros especificados en la función son sintácticamente correctos.

Para conocer la finalización de estas funciones, siempre se deberá esperar el desencadenamiento del evento Disabled(…).

Función	**Comando**	**Estados en los que es posible ejecutarla**	**Significado de respuesta correcta**
Disable	9	0x02	Se procede a deshabilitar el sistema (poner sistema en estado IDLE 0x01)
EmptyDevice	12	0x01	Es posible realizar el vaciado
EmptyDeviceSpecific	13	0x01	Es posible realizar el vaciado
Pay	29	0x01	Es posible realizar el pago
PaySpecific	30	0x01	Es posible realizar el pago
Totalize	42	0x01, 0x02, 0x07	Es posible indicar el total de la operación

ANEXO 5. Listado de propiedades

- **(20) BarCodeLength**
  Especifica el tamaño del código de barras que podrá leer el CK900. Debe de ser par y entre 6 y 24 dígitos, en caso de no querer activarlo, se le puede asignar longitud cero.

- **(0) BCMaxCoins**
  Especifica el número de monedas mínimo para permitir el BrokenCent extendido.

- **(1) BCMinValue**
  Especifica el importe mínimo para permitir el BrokenCent extendido.

- **(2) CancelLowLevel**
  Permite desactivar los eventos de niveles bajo mínimos (No recomendado).

- **(3) CancelValueEvents**
  Permite desactivar los eventos de valor (ValueIN, ValueOUT, CashBoxValue).

- **(5) CoinCashBoxDetect**
  Especifica si se controla que el cajon de recaudación de monedas este colocado.

- **(18) CoinsLowLevel**
  Especifica el número de monedas mínimo debajo del cual se empezará a avisar de que estamos en niveles bajos (evento LowLevel).

- **(4) ConfigID**
  Propiedad de solo lectura que devuelve el identificador de configuración activo (el que se ha utilizado en la función StarUp).

- **(*) ConnectionType**
  Propiedad de solo lectura que devuelve el de conexión activa (el que se ha utilizado en la función Connect).

- **(21) CriticalBehavior**
  Propiedad que controla como se procederá en caso de nivel crítico de moneda. Una moneda entra en situación de crítico cuando esta en franca desventaja respecto de las demás. Por Ejemplo, de todas las monedas tenemos 200 unidades pero de una solo tenemos 30.
  0 = Se enviaran a cajón de recaudación las monedas con niveles superiores a 40 hasta que desaparezca el nivel crítico. Este envío se realiza poco a poco, por lo que si se repone la moneda con nivel crítico, se detendrá el proceso.
  1 = Los pagos en monedas, se realizaran según se encuentren. Esto forzará los pagos con más monedas de las necesarias, equilibrando los niveles.
  2 = Ignorar. No hace nada, esta opción permite configuraciones extremas, no obstante, puede aumentar el tiempo de pago, así como el número de errores.

- **(*) CurrentOperation**
  Propiedad de solo lectura que devuelve la operación activa (por defecto es la cero, pero se puede cambiar con la función AlternateOperation).

- **(19) DeviceType**
  Propiedad de solo lectura que devuelve el tipo de dispositivo al que se ha conectado.
  Valores posibles:
  - 0 - CK900
  - 1 - LCDT_1000

- **(6) DisableAutoText**
  Indica si el display del CashKeeper funciona de forma autónoma o no.

- **(*) ErrorCode**
  Propiedad de solo lectura que informa del código de error de la última función ejecutada.

- **(*) ErrorDescription**
  Propiedad de solo lectura que informa de la descripción del error producido en la última función ejecutada.

- **(*) HostIP**
  Propiedad de solo lectura que devuelve la IP informada en el comando Connect.

- **(*) HostPort**
  Propiedad de solo lectura que devuelve el puerto Host indicado en el comando Connect.

- **(7) LightTime**
  Indica el tiempo que el led de salida de monedas estará abierto cuando se produzca una salida de moneda sea cual sea la causa. Ademas, permite controlar el retraso en que se dispara el evento Disabled respecto de la finalización real.
  Esta en milisegundos.

- **(8) LogDisable**
  Indica si se desactiva o no los log para posterior diagnostico. (Parcialmente implementado, MUY recomendable de tener siempre en falso).

- **(9) MaxCoins**
  Indica a partir de cuantas monedas al realizar un pago, el sistema nos avisara indicándoos que vamos a usar mas monedas.

- **(10) MaxPayout**
  Indica la importe máximo autorizado en pagos durante un intervalo de tiempo definido en la propiedad PayoutInterval. No importa si se ha realizado uno, dos o mil pagos durante ese intervalo de tiempo.
  ATENCIÓN!!!. No se contabilizan como pagos las devoluciones de cambio y eso incluye los Totalize de importes negativos.

- **(11) MinFastIn**
  Solo efectivo en el CK900. Indica el numero de billetes que en un solo cobreodisplay del CashKeeper funciona de forma autónoma o no.

- **(12) NLPAutoProtect**
  Indica si la protección de cambio en billetes decide de forma autónoma o pregunta al usuario.

- **(13) NLPPercentValue**
  Indica el porcentaje de cambio en billetes a proteger.

- **(14) NLPStartValue**
  Indica a partir de que valor de billete actuará la protección de cambio.

- **(15) NoteLevelProtection**
  Indica si la protección de cambio en billetes esta activa.

- **(*) OfficePort**
  Indica el puerto de conexión para conexiones tipo OFFICE.

- **(16) PayoutInterval**
  Indica el periodo en protección de pagos.

- **(17) RejectIfClosed**
  Indica si en estado de reposo de CashKeeper, la puerta de introducción de monedas estará en posición de rechazo de monedas.

- **(*) SecuritySeed**
  Numero de seguridad para la conexión debe de tener 6 dígitos.

- **(*) State**
  Indica el estado de CashKeeper.

*Solo disponibles en método CKeeper

ANEXO 6. Listado de eventos

- **(114) BarcodeRead**(Code as String)
  Se desencadena cada vez que el validador de billetes lee un código de barras. Este evento debe de responderse. La función AcceptPending envía el "billete" que contiene el código de barras a cajón de recaudación (vales descuento, vales regalo, etc.). La función RecjectPending devuelve el "billete" que contiene el código de barras (apto para carnets, etc.).

- **(115) BarcodeStored**(Code as String)
  Se desencadena cuando un "billete" que contiene un código de barras se ha almacenado en cajón de recaudación correctamnte.

- **(100) Disabled** (ErrorCode As Int32, ErrorDescription As String, CurrentValue As Int32, TargetValue As Int32, State As Byte, Operation As Byte)
  Se desencadena al terminar cualquier operación asincrona. En caso de Error, el parámetro ErrorCode será distinto de cero, ErrorDesciption contendrá la descripción del error, CurrentValue indicara el importe entrado/pagado, TargetValue indica el valor objetivo, State del tipo de operación y Operation de la operación en curso.

- **(102) LowLevel** (ValuesInfo as String, LevelsInfo as String)
  Se desencadena al final de una operación de salida de efectivo (vía cambio o vía cajón) si alguna de las denominaciones se encuentra por debajo de *CoinsLowLevel* en el caso de las monedas y en el caso de billetes de su valor de LowLevel configurado con la función **SetLowLevelNotes.**

  ValuesInfo(string): Cadena que contiene las denominaciones que responden al evento.
  LevelsInfo(Int32): Cadena que contiene los niveles actuales de cada una de las denominaciones informadas en 'ValuesInfo'.

- **(103) MaxCoinsWarning** (ValuesInfo as String, NumberInfo as String)
  Se desencadena cada vez que la cantidad de monedas a utilizar para un pago (por vía directa o por devolución de cambio) supera el máximo establecido por la propiedad *MaxCoins*.

- **(113) NoteHeldInBezel** (NotePresent as Boolean)
  Se desencadena cada vez que aparece un billete en la boca del validador (sea por pago de billetes o por rechazo en entrada)

  NotePresent (boolean): Informa del estado del billete
  True: billete presente
  False: billete retirado

- **(104) ProcessInterrupted** (State as Byte, Value as Int32, Target as Int32, Operation as Byte)
  Se desencadena al iniciar CashKeeper si este se había cerrado en medio de algún proceso. Imagina que en medio de una transacción se va la luz. Al iniciar nos indicara si en ese momento estaba realizando un cobro, que importe habían introducido, que importe querían cobrar y la operación. En el caso de pago, que cantidad queríamos pagar, y la cantidad pagada.

  State (byte): Informa del estado del dispositivo en el momento de la interrupción.
  Value (Int32): Informa del valor (de entrada o salida) al que se había llegado hasta el momento de la interrupción.
  Target (Int32): Informa del valor objetivo (de entrada o salida) al que se debía llegar en el momento de la interrupción.
  Operation(Byte): Informa del numero de operación que no se pudo completar.

- **(105) ProtectedValueNote** (Value as Int32, PayoutNeeds as Int32, TotalChange as Int32, Operation As Byte)
  Se desencadena si … (ver punto 6.15.2.2)

  Cuando se desencadena *ProtectedValueNote*, todos los sistemas de entrada de efectivo quedan paralizados a la espera de la respuesta del usuario mediante los métodos *AcceptPending* o *RejectPending*.

- **(106) StateChange** (State as Byte, OldState as Byte)
  Se desencadena cada vez que el estado del sistema cambia. Indicándonos el nuevo estado y el estado anterior.

- **(107) ValueIN** (CurrentValue as Int32, Target as Int32, Operation As Byte)
  Se desencadena cada vez que se inserta un billete o una moneda en el sistema y la propiedad *CancelValueEvents* contiene falso.

- **(108) ValueOUT** (CurrentValue as Int32, Target as Int32, Operation As Byte)
  Se desencadena en las acciones de salida de efectivo (vía cambio o devolución) y la propiedad *CancelValueEvents* contiene falso.

- **(109) ValueToCashBox** (CurrentValue as Int32, Target as Int32, Operation As Byte)
  Se desencadena en las acciones de salida de efectivo hacia el cajón de recaptación y la propiedad *CancelValueEvents* contiene falso.

- **(110) Warning** (Code as Int32, Description as String)
  Se desencadena cuando aparecen situaciones no críticas pero que requieren del conocimiento del usuario. Después de liberar el evento, el dispositivo intentará volver al estado anterior al evento de 'Warning'.

ANEXO 7. Configuración manual de parámetros de conexión al servicio (CSSI)

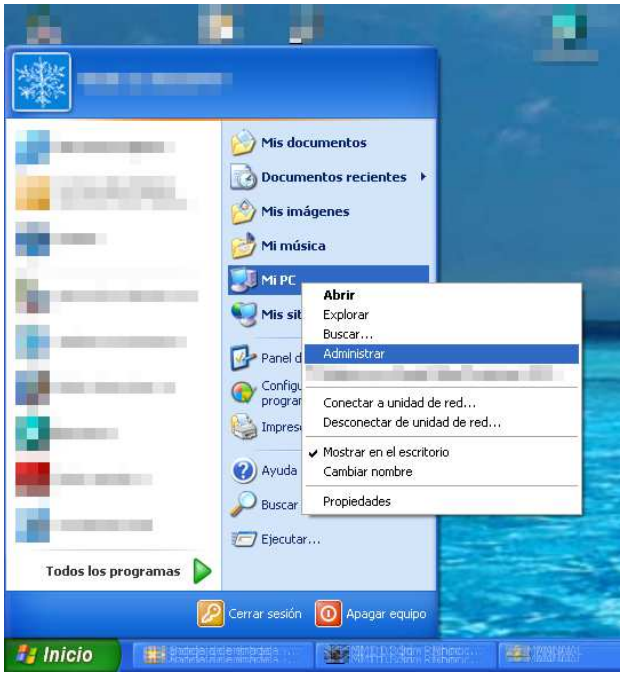
Cuando el aplicativo se ejecuta en el mismo ordenador que tiene conectado físicamente CashKeeper, la configuración de los parámetros de conexión a al CSSI se realiza de forma automática en el momento de invocar al método '***Connect(…)***'. Si el CSSI (así como la conexión física a CashKeeper) se encuentra en una máquina dedicada, la configuración de los parámetros de conexión se debe realizar a mano siguiendo unos sencillos pasos:

Crear un acceso directo al CSSI (CKeeper.exe) ubicado por defecto en el directorio de sistema ("c:\windows\system32" o "C:\windows\syswow64") pasándole como parámetro el puerto Host, puerto Office, Seed. Quedando aproximadamente de la siguiente forma
"C:\windows\system32\CKeeper.exe 8001,8002,100001"

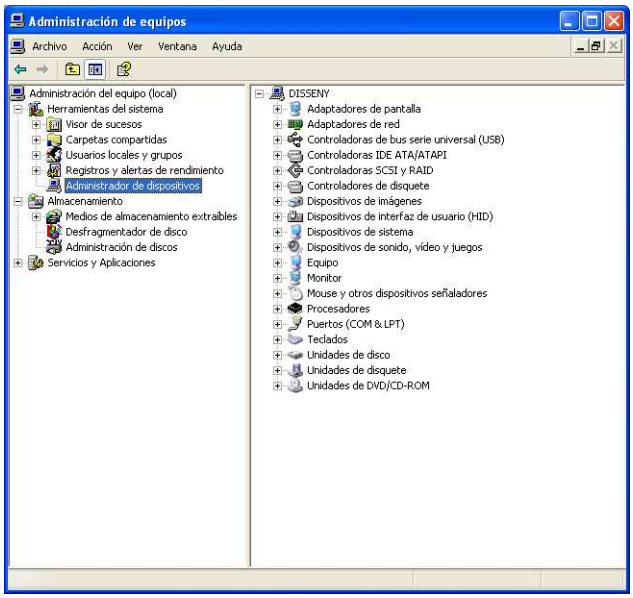
ANEXO 8. Configuración de la gestión de energía del puerto USB (S.O. WINDOWS®)

Para asegurar que la comunicación con CashKeeper está siempre disponible, en los PC / TPV / Server que estén conectados al dispositivo se deberá desactivar la opción de desconexión automática de la alimentación del puerto USB que los sistemas operativos WINDOWS® tienen configurada por defecto.

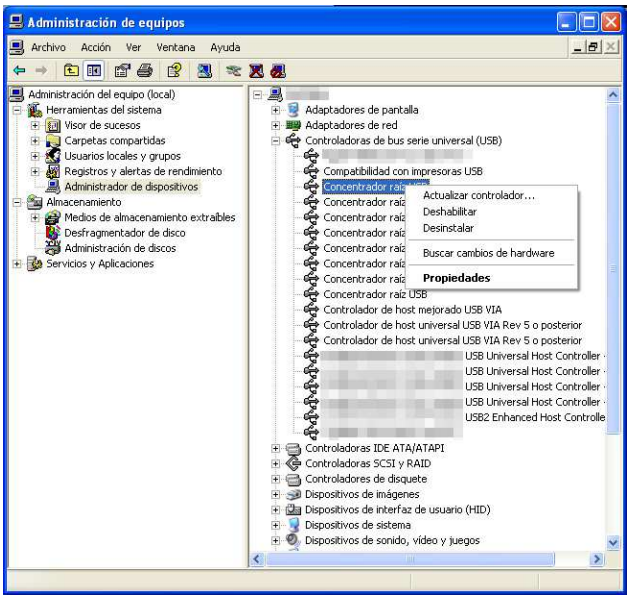
Para realizar esto se deberá acceder al 'Administrador de dispositivos' (MiPC, botón derecho, Administrar)



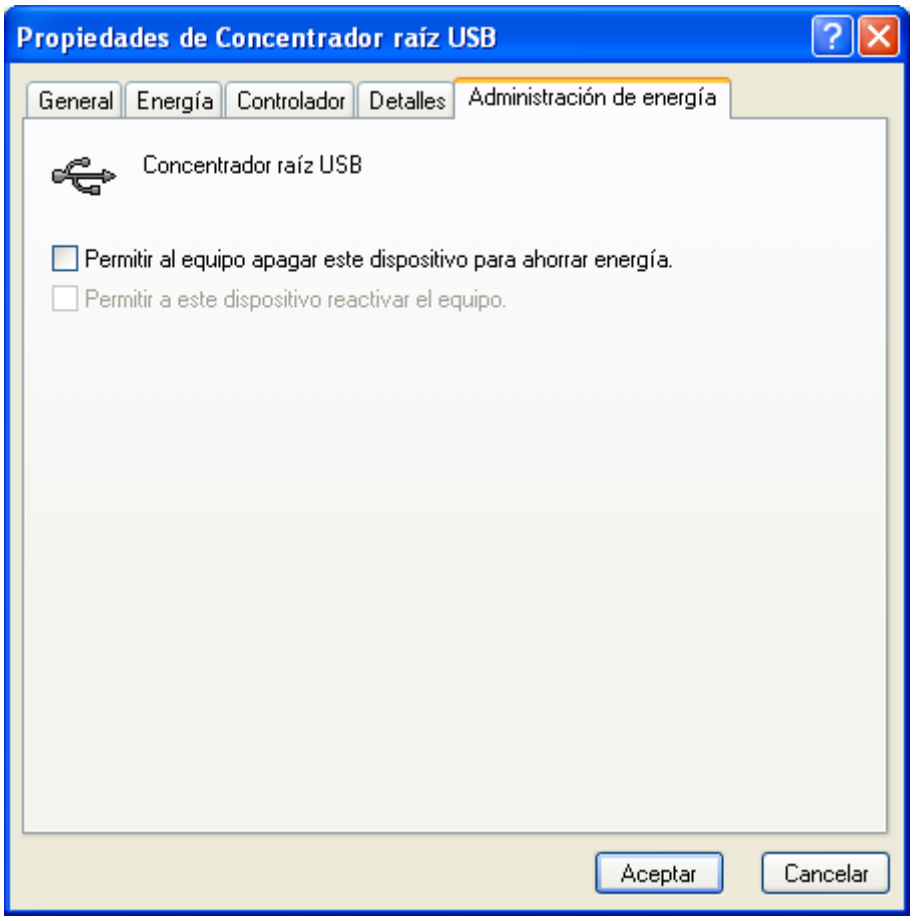
Se abrirá el 'Administrador de Equipos' y, ahí seleccionaremos el 'Administrador de dispositivos'.



Ya con el 'Administrador de dispositivos' activo, deberemos seleccionar los concentradores raíz USB de nuestro sistema, botón derecho, 'Propiedades'.



Finalmente, en la ventana de propiedades, seleccionar la pestaña de 'Administración de energía' y DESMARCAR la opción 'Permitir al equipo apagar este dispositivo para ahorrar energía'.


Propiedades de Concentrador raíz USB
General
Energía
Controlador
Detalles
Administración de energía
Concentrador raíz USB
Permitir al equipo apagar este dispositivo para ahorrar energía.
Permitir a este dispositivo reactivar el equipo.
Aceptar
Cancelar

ANEXO 9. Relación de códigos y textos del display en modo automático

Código	Texto ESP	Text ENG
1	CASHKEEPER *(protegido)*	CASHKEEPER *(protected)*
2	Iniciando Sistema	System startup
3	Configurando Sistema	Configurating
4	INTRODUCIR IMPORTE	INSERT COINS/NOTES
5	ERROR EN EL SISTEMA	SYSTEM ERROR
6	--------------------	--------------------
7	Actualizando sistema	Updating system
8	FUERA DE SERVICIO	OUT OF SERVICE
9	Limpiando validador	Cleaning validator
10	...Espere...	...Wait...
11	- NO DISPONIBLE -	- NOT AVAILABLE -
12	Comprobando estado	Checking state
13	Cerrando sistema....	Closing system.....
14	Cerrando puertos....	Closing ports.....
15	....OCURRIó UN ERROR	..... ERROR
16	(20 espacios)	(20 blanks)
17	A PAGAR:	TOTAL:
18	PAGADO :	PAID :
19	IMPORTE INTRODUCIDO:	TOTAL PAID:
20	Terminal ocupado	Device blocked
21	por el operador	by the operator
22	¡ATENCIÓN!	¡WARNING!
23	-BILLETE ATASCADO-	-NOTE JAMMED-
25	-TERMINAL BLOQUEADO-	-DEVICE BLOCKED-
26	A DEVOLVER:	TO DISPENSE:
27	DEVUELTO :	DISPENSED :
28	- RECOGER IMPORTE -	- YOUR CHANGE -
29	Euro	Euro
30	Terminal detenido	Device Temporary
31	Temporalmente	Stopped
32	Terminal activo	Active Terminal

ANEXO 10. Formato de los tiques con código de barras

Codificación	Interleave 2 of 5 (ITF)
Ancho de barras	Mínimo: 0.5mm Máximo: 0.6mm
Ratio de contraste	2:1
Número de caracteres	Mínimo: 6 Máximo: 24

Dimensiones

Orientación	Vertical
A	Mínimo: 65mm Máximo: 82mm
B	Mínimo: 120mm Máximo: 156mm
C	Mínimo: 10mm
D	Mínimo: 35mm
E	Mínimo: 10mm