CASHKEEPER®

closed and secure cash handling

MANUAL DE INTEGRACIÓN

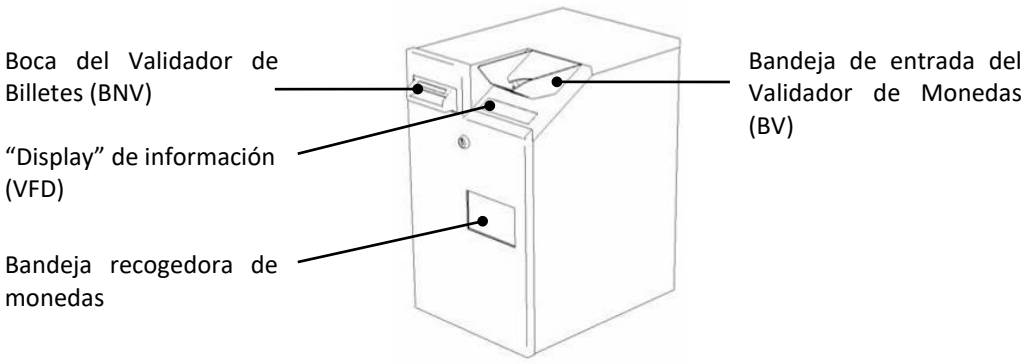
V.2

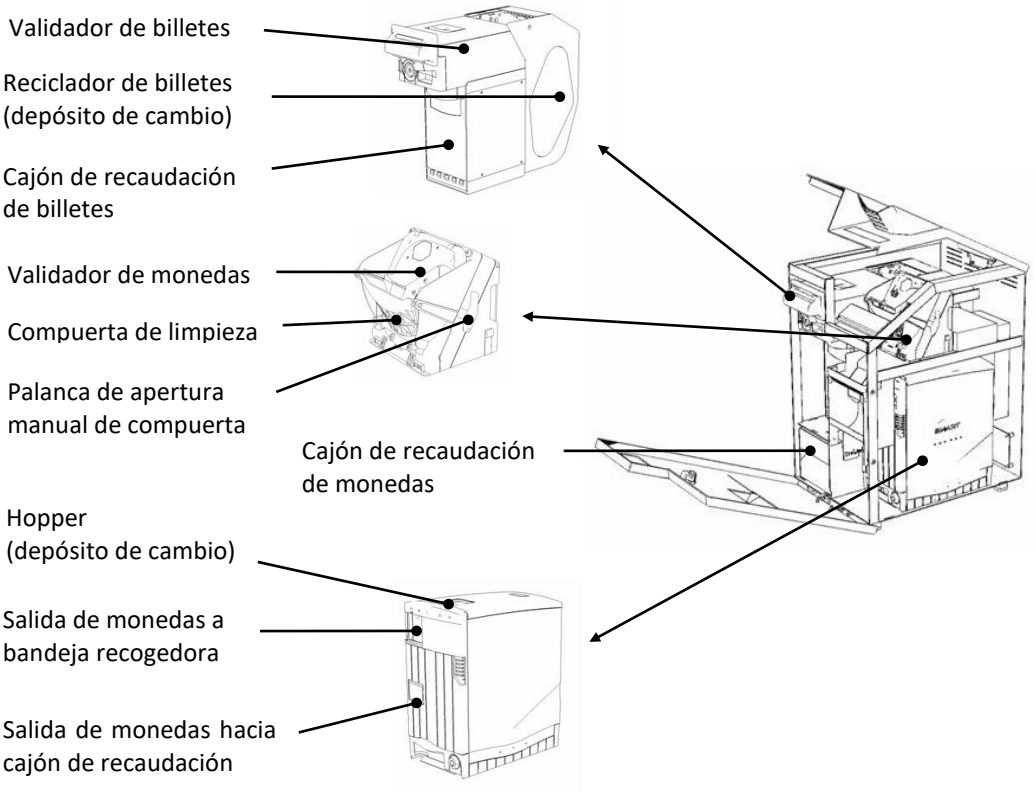
Fecha	Modificación	Autor
18/03/12	Versión preliminar	MIC
11/03/15	Añadido CriticalBehaviour	MIC
5/10/15	Actualizada la tabla de errores. Añadida función GetLastLevels. Añadida función SetDateTime. Añadida función GetNetworkParams. Añadida función SetNetworkParams. Añadida función EmptyCashBoxEx. Añadida 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
10/10/20	Añadido método de integración CKFast Añadido apartado 'Elección del método de integración" Actualizado apartado 'Descripción del CashKeeper' Eliminar apartado 'Concepto de Integración' Eliminar apartado 'Diagrama de integración' Redistribución general de cada apartado	JBC

ÍNDICE

1. DESCRIPCIÓN DE CASHKEEPER

- **Descripción de los elementos (modelo CK900/CK950)**





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).

3. METODOS DE INTEGRACIÓN CON CASHKEEPER

3.1. CKEasy

Se trata de un sistema fácil y rápido de integración, ya que incorpora su propia interfaz gráfica. Para ello, se debe trabajar con la librería *CKEasyV2.dll.* Esta librería incorpora todas las llamadas de tipo síncronas, lo que facilita el uso de las distintas funcionalidades, las cuales, son de muy alto nivel, del tipo cobra, paga, hacer caja... Así mismo, incorpora algunas llamadas sin soporte gráfico para permitir a la aplicación la obtención de datos de caja.

3.2. CKFast

Se trata del sistema más fácil y rápido de integración, ya que solamente se compone de dos comandos/instrucciones e incorpora su propia interfaz gráfica. Para ello es necesario trabajar con 'Sockets'. Tal método, requiere realizar una conexión hacia nuestro software gratuito *CKMonitor*, el cual trabaja como un gestor de colas, y este último, se encarga de comunicarse con el CashKeeper.

3.3. CKeeper

Se trata de un sistema que permite controlar CashKeeper en toda su plenitud a través de la librería *CKeeperV2.dll* (para Windows) o *CKeeper.jar* (para Java de Android). No incorpora interfaz gráfica, por lo que permite una completa integración con su aplicación. Ofrece la misma funcionalidad que el método *Raw Sockets*, pero al trabajar como un objeto, facilita la programación.

3.4. Raw Sockets

Se trata del sistema de integración de más bajo nivel y del que se requiere una mayor complejidad, aunque le permite ser transparente al S.O. utilizado. Eso significa que puede ser implementado en cualquier S.O., el único requerimiento es que el S.O. en el que se ejecuta su aplicación, disponga de TCP/IP (hoy en día todos). Para ello se utilizan 'Sockets'. Este método se comunica directamente con el CashKeeper, y se deben realizar los comandos/instrucciones precisas para cada una de las operaciones que se quieran efectuar.

4. ELECCIÓN DEL METODO DE INTEGRACIÓN

Presentados los diferentes métodos de integración, es momento de escoger el más adecuado. ¿Cómo saber cuál es el mejor método para mi software?

4.1. Escoger método con explicación

En primer lugar, debemos clasificar nuestros propios métodos de integración, y ver sus ventajas y desventajas al respeto y entre cada uno de ellos.

4.1.1. Métodos de alto nivel

Los métodos *CKEasy* y *CKFast* son de alto nivel, eso significa que incorporan interfaz gráfica. Como ventajas de esto, podemos garantizar que la integración a través de estos métodos se puede realizar y finalizar en un día. Además, podemos asegurar que su integración quedará 100% realizada y totalmente funcional para el cliente final. Eso significa, que su cliente no va a echar en falta ninguna funcionalidad a la hora de trabajar con el CashKeeper (eso si, en el caso de trabajar con *CKEasy*, deben implementarse todas las funcionalidades). Como desventaja de estos métodos es que ambos dependen, como mínimo, de disponer un terminal con el sistema operativo Windows.

¿Como saber cuál escoger, *CKEasy* o *CKFast*?

- **CKEasy:** Trabaja con la librería 'ActiveX' *CKEasyV2.dll*, lo cual facilita la programación ya que se trabaja como un "objeto" y su implementación queda muy bien integrada. En su defecto, es necesario que el software POS trabaje con Windows. Este método solo permite una conexión 1 a 1 (1 POS - 1 CashKeeper).

- **CKFast:** Trabaja con *Sockets*, lo cual permite que se pueda implementar en cualquier software POS. En su defecto, este método depende del software gratuito *CKMonitor*, el cual debe correr sobre un sistema operativo en Windows. La ventaja de trabajar con el software *CKMonitor*, el cual funciona como un gestor de colas, es que permite tener diferentes terminales que trabajen con un mismo CashKeeper (N POS - 1 CashKeeper).

4.1.2. Métodos de bajo nivel

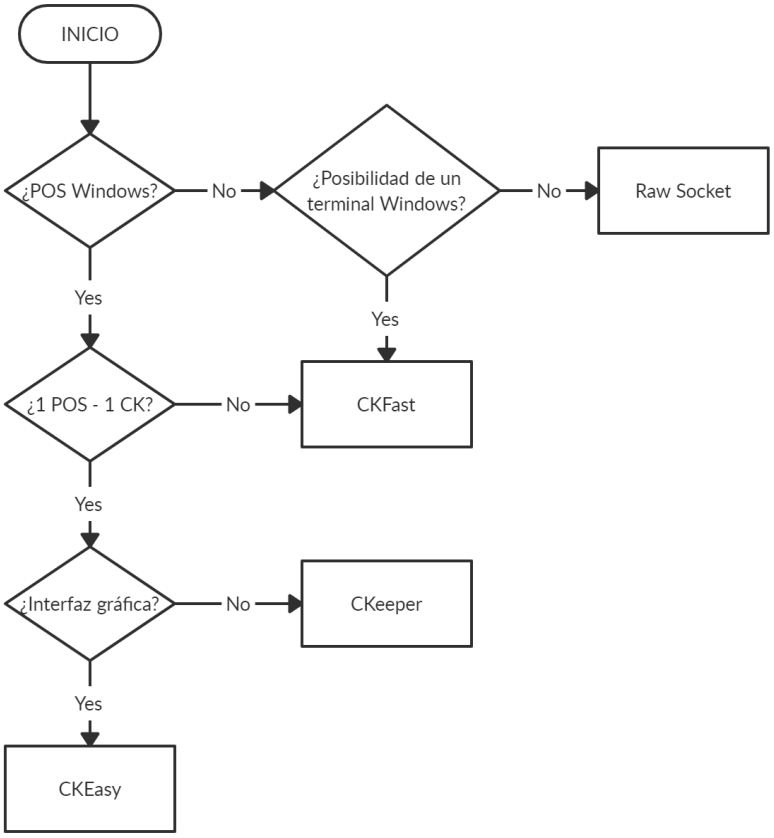
Los métodos *CKeeper* y *Raw Socket* son de bajo nivel, eso significa que no incorporan interfaz gráfica. Como ventaja de estos dos métodos, encontramos su máxima personalización a la hora de su integración. Todas las pantallas y visualización de datos quedan en manos del programador. Como desventajas encontramos: cierta dificultad a la hora de implementar, un gran coste de tiempo de integración (15 días mínimo), y una homologación requerida por nuestra parte (la cual es gratuita).

¿Como saber cuál escoger, *CKeeper* o *Raw Socket*?

- **CKeeper:** Trabaja con la librería 'ActiveX' *CKeeperV2.dll* (para Windows), o con la librería *CKeeper.jar* (para Java de Android), lo cual facilita la programación, ya que, en ambas, se trabajan como un "objeto". Para utilizar este método, es necesario que el software POS trabaje con Windows (usar *CKeeperV2.dll*) o Android (usar *CKeeper.jar*). Este método solo permite una conexión 1 a 1 (1 POS - 1 CashKeeper).

- **Raw Socket:** Trabaja con *Sockets*, lo cual permite que se pueda implementar en cualquier software POS con cualquier sistema operativo. Este método solo permite una conexión 1 a 1 (1 POS - 1 CashKeeper).

4.2. Escoger método con diagrama de flujo



5. UTILIZAR CKEasy

Para utilizar *CKEasy*, lo primero que debe hacer es importar la librería *CKEasyV2.dll* en su proyecto. Una vez importada, cree un "objeto" 'ActiveX' del tipo *EasyCashKeeper*. Puesto que el proceso de conexión requiere unos segundos, se recomienda conectar al CashKeeper (**5.1. Connect**) al iniciar la aplicación (aunque el integrador puede elegir cuando conectar) y desconectar (**5.2. Disconnect**) al salir. Una vez efectuado la conexión, 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 (Ojo con los "timers").

NOTA: Se recomienda consultar el ejemplo de integración 'CKEasyV2Test' que se encuentra en el archivo comprimido 'SDK.zip'.

5.1. Conexión
Hace falta informar de la propiedad *IP* con la IP del CashKeeper o la del PC donde se encuentra conectado la máquina (máquinas USB) y llamar al método ***Connect***.

5.2. Desconexión
Para la desconexión del CashKeeper, se debe llamar a la función ***Disconnect***.

5.3. Cobro
Para realizar un cobro, se debe llamar a la función ***Charge***, devolviendo una vez finalizado: el importe introducido, el cambio devuelto y si se ha producido algún problema.

5.4. Pago
Para realizar un pago, se debe llamar a la función ***Pay***, la cual devolverá el resultado del pago (correcto/ incorrecto) así como el importe pagado.

5.5. 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.

5.6. Añadir cambio
Para añadir cambio, se debe llamar a la función ***AddChange.*** Esta abrirá una pantalla con la carencia/insuficiencia, según la configuración, de las distintas denominaciones y permitiendo la introducción de monedas y billetes. La pantalla incluye dos botones para finalizar la operación, y dependiendo de cual se pulse, el sistema actuará de forma distinta:

- 'Aceptar': La máquina se quedará con el dinero y la función devolverá el importe introducido. Esto modifica el fondo de caja (hay que generar el asiento necesario).

- ‘Cancelar’: La máquina devolverá el dinero y la función devolverá importe ‘0’. La devolución se realiza pagando la menor cantidad posible de monedas y billetes. En este caso no se modifica el fondo de caja.

Ejemplo: Si se introducen 20 monedas de 1€ nos devolverá un billete de 20€.

5.7. Dar cambio

La función ***Change*** abre una pantalla que permite introducir dinero y elegir en que forma queremos que nos lo devuelva. Devuelve el importe introducido y el valor pagado.

NOTA: Recordar que en una instalación con CashKeeper, no hay dinero en mano para dar cambio para la máquina de tabaco, cochecitos, etc.

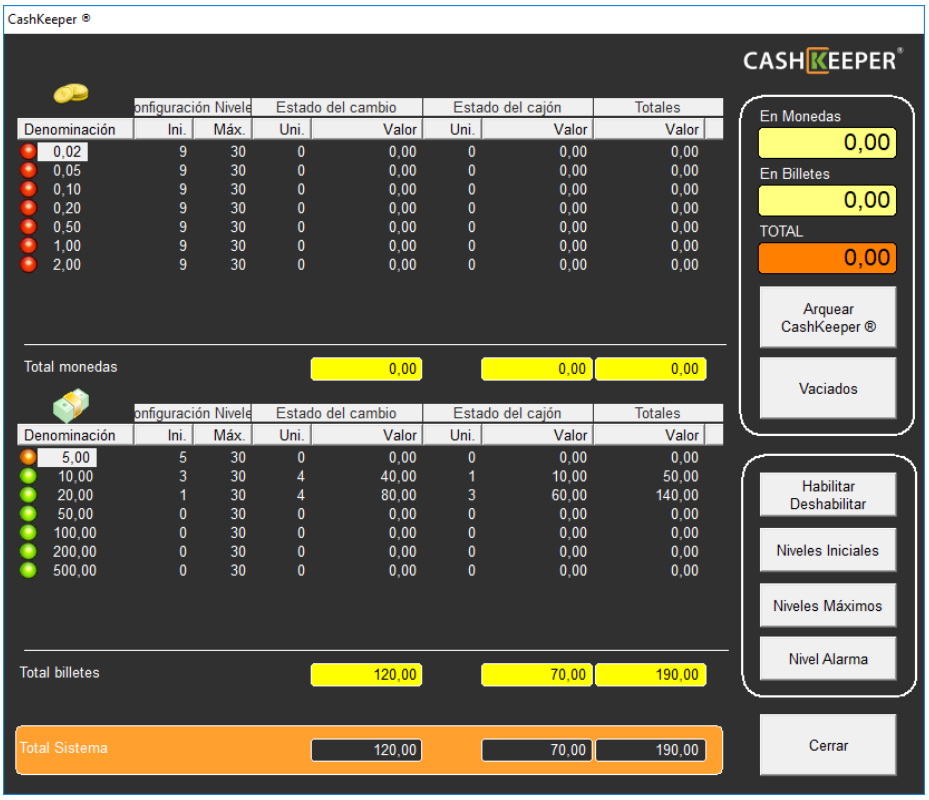
5.8. Parámetros

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

NOTA: La definición de cada uno de los parámetros configurables se encuentra explicado en el recuadro amarillo de la parte inferior de la captura de pantalla cuando el cursor selecciona los campos modificables.

5.9. 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 interfaz gráfica, dependiendo del tipo de modo que se le pase a la función, para el caso de que el software POS ya disponga de una pantalla propia.

5.10. Funciones sin interfaz 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.

5.11. 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 decimales a mostrar en los importes.

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

*Ejemplo: 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)*

5.12. 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.

6. UTILIZAR CKFast

La implementación del método *CKFast* consiste en abrir una conexión 'Socket' hacia nuestro software gratuito *CKMonitor*, y este, es quien se va a encargar de comunicarse con el CashKeeper. Aquí puede ver un pequeño diagrama explicativo de cómo se realiza la comunicación.

El funcionamiento es muy simple, se abre una conexión 'Socket', y se envía el comando de ***CrearTiquet***. A raíz de esto, nuestro software *CKMonitor*, que previamente ha sido configurado y se encuentra ejecutándose (puede estar minimizado), se despliega automáticamente para así poder tramitar/gestionar el "tiquet" (cobro) enviado.

Se debe instalar un *CKMonitor* por cada uno de los CashKeeper instalados. En el caso de tener más de un terminal que va a trabajar con un mismo CashKeeper, deben considerar cual va a ser el PC principal/central que va a contener el gestor de colas *CKMonitor*. No es necesario tener un terminal de uso exclusivo para el *CKMonitor*, pero si se debe tener en cuenta en que, si un mismo terminal contiene el *CKMonitor* y a la vez, hace de punto de venta, en el momento que se reciba un nuevo "tiquet", el *CKMonitor* se va a desplegar en pantalla completa, obstruyendo la visibilidad del operador que estaba utilizando ese terminal.

NOTA: Se recomienda leerse el "ES - Manual técnico CKMonitor".

NOTA: Se recomienda consultar el ejemplo de integración "CKFastTest" que se encuentra en el archivo comprimido 'SDK.zip'.

6.1. Consideraciones previas

CKMonitor permite tener N conexiones abiertas. Eso significa que no importa el número de terminales que tengan una conexión 'Socket' abierta y establecida con él. De esta forma cada terminal puede ir enviado sus datos ("tiquets") sin ningún problema.

Queda a la elección del programador si efectuar una única conexión en toda la sesión, o realizar una conexión/desconexión para cada comando.

NOTA: CashKeeper recomienda abrir y cerrar la conexión de un 'Socket' por cada operación (comando/instrucción) realizada/enviada. Hemos detectado que en conexiones donde interviene VPN's y/o internet, estas se cortan debido a la inactividad.

6.2. Puesta en marcha

El primer paso a seguir es abrir una conexión 'Socket'. Tal conexión se efectúa con una IP y un Puerto.

- Parámetros de entrada:
  - *IP*: Host/IP del PC donde se encuentra el *CKMonitor* instalado y ejecutándose.
  - *Puerto*: Puerto de comunicación. Por defecto es "16501". Se puede modificar des de la configuración del *CKMonitor*.

Una vez efectuado la conexión, envíe los comandos/instrucciones según se desee.

6.3. Formato de mensajes

Solamente existen dos comandos disponibles a enviar a través de *Socket* i deberán tener el siguiente formato:

Deben empezar por el carácter de comando a realizar, "C" (ASCII 67) en caso de ***CrearTiquet*** o "S" (ASCII 83) en caso de consultar el ***EstadoTiquet***. En caso de tener parámetros, estos irán precedidos por la barra vertical "|" (ASCII 124).

Cada vez que enviemos un comando, vamos a obtener siempre una única respuesta.

6.3.1. CrearTiquet

Este comando es el que utilizaremos para crear el tiquet. Debe empezar por "C" (ASCII 67), seguido de un número de tiquet único identificatorio (si es preciso, introducir fecha y hora), y acabar siguiendo el formato siguiente:

```
C|NumTiquet|Importe|ID2|IDCaja|Terminal|Vendedor|FormaPago
```

- Parámetros de entrada obligatorios:
  - *NumTiquet*: [VARCHAR(50)] - Número identificatorio único de tiquet.
  - *Importe:* [Int32] - Importe del tiquet a cobrar en céntimos.
- Parámetros de entrada no obligatorios:
  - *ID2*: [VARCHAR(50)] - Identificador secundario.
  - *IDCaja*: [Int32] - Número de punto de venta.
  - *Terminal*: [VARCHAR(50)] - Terminal punto de venta.
  - *Vendedor*: [VARCHAR(50)] - Código de vendedor.
  - *FormaPago*: [VARCHAR(50)] - Identificado de forma de pago.

NOTA: Los parámetros opcionales (no obligatorios) son posicionales, por lo que, si se quiere implementar uno de ellos, hay que añadir tantos separadores "|" como sea necesario. En el caso de no querer implementar ninguno, no es necesario implementar ningún separador "|" extra.

Ejemplo: Crear un tiquet con el parámetro 'vendedor = pepe':

```
C|NumTiquet|Importe|||Pepe
```

Las respuestas posibles son las siguientes:

- El tiquet se ha añadido correctamente en *CKMonitor*
  OK|Added
- El tiquet contiene una ID no valida
  NOK|Invalid ID
- La ID del tiquet ya se ha creado anteriormente
  NOK|Already exists
- El tiquet no se ha creado/añadido correctamente en *CKMonitor*
  NOK|Failed

6.3.2. EstadoTiquet

Este comando es el que utilizaremos para consultar el estado del tiquet que previamente se ha creado. Debe empezar por "S" (ASCII 83) i seguir el formato siguiente:

S|NumTiquet

- *NumTiquet*: Número identificatorio único de tiquet.

Las respuestas posibles son las siguientes:

- El tiquet se ha cobrado correctamente
  OK|OK
- El tiquet está pendiente de ser cobrado
  OK|Pending
- El tiquet se ha cancelado des de *CKMonitor*
  OK|Cancel
- El tiquet consultado no existe
  NOK|Not found

6.3.3. Otras respuestas

En caso de enviar un comando que no se corresponda a ninguno de los anteriores recibiremos la siguiente respuesta:

NOK|Unknown command

7. UTILIZAR CKeeper

A continuación, se detallan los procedimientos a realizar para operar con *CKeeper.dll* (Windows). Para operar con *CKeeper.jar* (Java de Android), seguir los mismos procedimientos que se muestran a continuación, pero con la lectura previa del apartado **7.16. Diferencias al implementar método *CKeeper* en Android**.

ATENCIÓN: Estas librerías no es "thread-safe", por lo que no se pueden realizar llamadas durante la ejecución de otra llamada (Ojo con los "timers").

NOTA: Se recomienda consultar el ejemplo de integración "CKV2Demo" que se encuentra en el archivo comprimido 'SDK.zip' para la integración en Windows.

7.1. 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 está o no está 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.

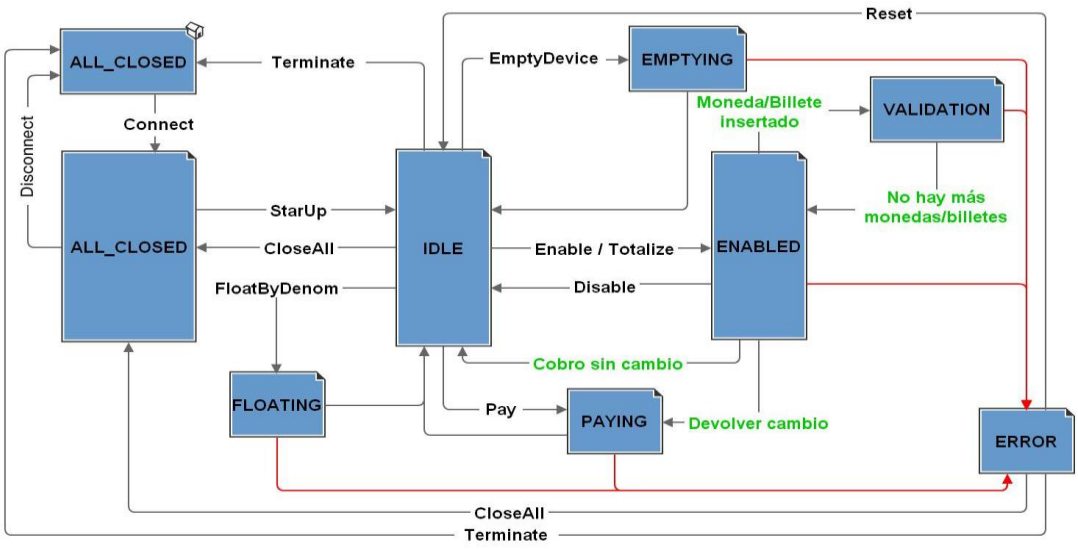
- FLOATING (0x04): El sistema está enviando monedas y/o billetes hacia los cajones de recaudación. Durante este estado, el sistema desencadenará el evento *ValueToCashBox* informando del total del importe enviado a cajón/es.

- 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.

7.2. 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.

7.3. Consideraciones previas

7.3.1. 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.

7.3.2. 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 (depende de la divisa) y sin decimales. La función ***GetCCDetails*** nos indicará el multiplicador y el número de decimales a mostrar.

*Ejemplo: La función GetCCDetails (para euro) nos devolverá 100 como multiplicador y 2 como decimales a mostrar. Si deseamos hacer una operación de 4,5€, el importe a comunicar a la máquina es 450 (4,5 * 100).*

7.3.3. Respuesta de funciones

Todas las funciones de CashKeeper devuelven un valor tipo 'boolean' como resultado de esta. 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.*

7.3.4. Pagos en CashKeeper

CashKeeper está programado para realizar los pagos en el menor número de monedas y/o billetes posibles. De todas formas, en algún caso, podrían no utilizarse las denominaciones óptimas teóricas para reducir los tiempos de pago.

Ejemplo: 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).

7.3.5. 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 de este.

7.3.6. Gestión de Eventos

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

7.3.7. Características

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

7.4. 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 máquina mediante el siguiente método:

```
Connect (HostIP as string, HostPort as byte, OfficePort as byte,
ConnectionType as byte, SecuritySeed as Int32)
```

- Parámetros de entrada:
  - *HostIP*: nombre de red o dirección IP de la máquina.
  - *HostPort*: puerto de escucha de la máquina para las conexiones "Host".
  - *OfficePort*: puerto de escucha de la máquina para las conexiones "BackOffice".

- *ConnectionType*: tipo de conexión que se está realizando.
  - 0 = Host
  - 1 = BackOffice
- *SecuritySeed*: Código de seguridad para asegurar las conexiones con la máquina (de valor mayor a 100000 e inferior a 999999).

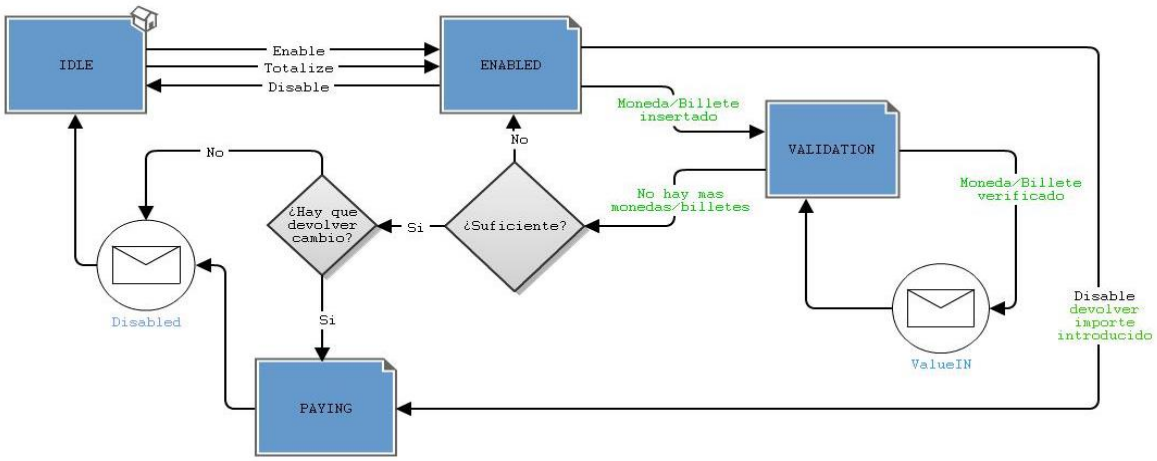
Si el resultado de la función ***Connect*** es satisfactorio, la función devolverá verdadero. En caso contrario, devolverá falso y se deberá consultar las propiedades *ErrorCode* y *ErrorDescription* para conocer la causa del fallo.

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

7.5. Cobros

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

La transición de estados para realizar un cobro sería el 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 está en estado VALIDATION (se está 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 número 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.

7.5.1. 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***.

7.6. Realizar pagos - (proceso asíncrono)

7.6.1. Pago de un importe

[Funciones implicadas: ***Pay***]

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* informándonos 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.

7.6.2. 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.

7.7. Operaciones de vaciado

7.7.1. Vaciado de cambio - (proceso asíncrono)

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

7.7.1.1. 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.

7.7.1.2. 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.

NOTA: En algunos modelos de CashKeeper, el vaciado tiene preferencia respecto a la validación, eso significa que, si una moneda no ha sido reconocida, se manda a cajón de recaudación sin contabilizar. Esto puede provocar un descuadre entre el importe antes de vaciar y el importe de vaciado.

7.7.2. 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***.

7.8. 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.

7.9. 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.

7.9.1. Efectivo disponible para cambio

[Funciones implicadas: ***GetCurrentLevel***]

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

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

7.9.2. 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.

7.10. 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 más. La aplicación puede enterarse de este hecho llamando la función ***GetBrokenCents***.

Ejemplo: En EUROS, CashKeeper, en alguno de nuestros modelos, 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.

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 más se reduce el número de monedas, si efectivamente se reduce, devolverá un céntimo de más 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.

7.11. 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.

7.12. Otras funciones

7.12.1. 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.

7.12.2. 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.

7.12.3. 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 ejecución (hasta 20min.). 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.

7.13. Parametrización del sistema

7.13.1. Inhibición de denominaciones

[Funciones implicadas: ***SetInhibitState, GetInhibitState***]

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

7.13.2. Protección de cambio

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

7.13.2.1. Avisos de nivel bajo

El sistema dispone de un evento *LowLevel* que nos informara cuando alguna denominación está 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 (aun estando por encima de dicho nivel) si alguna denominación está 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.

7.13.2.2. 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 está 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* está 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.

*NOTA: Ver anexo **8.4.1. Diagrama 'Note Level Protection'***

7.13.3. 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 a poder pagar en un intervalo de tiempo (en minutos) determinado.

Ejemplo: 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.

7.13.4. Otros parámetros y propiedades

7.13.4.1. DisableAutoText (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” 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***.

7.13.4.2. LightTime (propiedad de ‘Aspecto’)

La propiedad *LightTime* nos permitirá especificar el tiempo (en milisegundos) 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).

7.13.4.3. 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).

7.13.4.4. CancelLowLevel (propiedad de ‘Comportamiento’)

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

7.13.4.5. 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.

7.13.4.6. ConfigID (propiedad de 'Identificación')

La propiedad *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.

7.13.4.7. LogDisable

La propiedad *LogDisable*, habilita o deshabilita el registro de monedas y billetes entrantes. Actualmente función obsoleta.

7.13.4.8. CoinCashBoxDetect

La propiedad *CoinCashBoxDetect*, 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 0 = FALSE.

7.13.5. 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.

7.14. 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 ***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***.

7.15. Manejo de más de un dispositivo simultáneo

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.

7.16. Diferencias al implementar método CKeeper en Android

Para implementar la librería *CKeeper.jar* para Java de Android es necesario conocer las diferencias que podemos encontrar respecto la librería 'Active X' de Windows, o, mejor dicho, las diferencias que supone trabajar con objetos y funciones en Java.

La principal diferencia que encontramos en trabajar de una forma u otra, son los parámetros de salida. Todas las funciones que tienen parámetros por referencia (valores de salida), devuelven un objeto de tipo *CkResult.* Dicha clase, solo contiene el atributo *Response*. Este atributo es solo de lectura e informa de los parámetros de salida en la última función ejecutada.

Los parámetros se obtienen a través del método 'GET', del atributo *Response*, y el nombre del parámetro:

```
.get("nombre_parametro")
```

El nombre del parámetro va relacionado a la función ejecutada, siendo así, un valor predefinido por la librería *CKeeper.dll*.

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.jar* para Java de Android.

Ejemplo:

- *Función en CKeeper.dll (lenguaje VB6)*

```
Function GetBrokenCents(ByRef value as Long, ByVal
resetValue as Boolean) as Boolean
```

- *Función en CKeeper.jar (lenguaje Java)*

```
Public CkResult GetBrokenCents(boolean resetValue) {}

CkResult ckRes = socket.GetBrokenCents(false) ;

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

8. UTILIZAR Raw Sockets

A continuación, se detallan los procedimientos a realizar para operar con CashKeeper con el método *Raw Sockets* (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 **7. Utilizar CKeeper**. Este se distingue básicamente por algunos cambios en propiedades solo disponibles en el método *CKeeper* y algunos métodos solo disponibles en el método *Raw Sockets* (método directo).

*NOTA: En el anexo **9.2. Listado funciones y propiedades bajo nivel** encontrarás la lista de funciones, propiedades y eventos con su correspondiente código.*

8.2. Formato de mensajes

8.2.1. Envios

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 comando. 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#

*Ejemplo: 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 que los envíos. Empiezan por el carácter "$" (ASCII 36), seguido del código de comando y, siempre como mínimo un parámetro de "0" o "1" en función de si se ha producido un error o no. En caso de haberse producido un error incluirá dos parámetros más que se corresponderán al código de error y descripción del error. Todos estos separados por la barra vertical "|" (ASCII 124) y terminando por el carácter "#" (ASCII 35).

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

- Enviamos:
  - **$14#**
- Recibimos:
  - Respuesta sin error
    - **$14|1#**

- Respuesta con error
  - **$14|0|37|Dispositivo no disponible#**

*Ejemplo: Secuencia de comando y respuesta para solicitud de la cantidad disponible **(19) GetCurrentLevel** en cambio de las denominaciones de 1.00€ y 2.00 €*

- Enviamos:
  - **$19|100,200#**
- Recibimos:
  - **$19|1|30,25#**

  NOTA: 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:

Ejemplo: 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 abrir un Socket hacia el CashKeeper. 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|*númeroProporcionado*#**

Ahora, con el *númeroProporcionado*, deberemos calcular la clave, sumándole el valor del parámetro *SecuritySeed* (el mismo que habremos configurado en la aplicación del servicio).

Ejemplo: SecuritySeed = 140806 y NúmeroProporcionado = 1620168189
La clave de acceso = SecurtiySeed + NúmeroProporcionado = 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 *SecuritySeed* 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. 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.

2. Negociar la clave de acceso. Por defecto el parámetro *SecuritySeed* (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 *(106) StateChange*.

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 ***(48) Terminate***. Esto cerrará 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 cual sea el estado de éste.

NOTA: Aún y devolver un error del comando ***(7) CloseAll****, 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.*

9. ANEXOS

9.1. Listado funciones y propiedades alto nivel (*CKEasy.dll*)

9.1.1. Funciones

- **`AddChange (PaidInValue As Int32) as Boolean`**

  Abre una pantalla que permite añadir cambio a CashKeeper.
  - Parámetros de salida:
    - *PaidInValue*: contendrá el importe añadido a cambio.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: 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) as Boolean`**

  Permite gestionar los cajones del CashKeeper dependiendo del modo.
  - 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: ha terminado correctamente.

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

- **`Configuration () as Boolean`**

  Abre una pantalla que permite configurar distintos parámetros de CashKeeper.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **`Connect () as Boolean`**

  Inicia la conexión con el CSSI
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **`Change (PaidInValue As Int32, PaidOutValue As Int32) as Boolean`**

  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: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **`Charge (ValueInCents As Int32, PaidInValue As Int32, PaidOutValue As Int32) as Boolean`**

  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: ha terminado correctamente.
    - Falso: 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`*`) as Boolean`**

  Función sin pantalla para vaciar los cajones de recaudación.
  - Parámetros de entrada:
    - *Device*: Cajón a vaciar (valores posibles)
      - 0 = CKD_Coins
      - 1 = CKD_Notes
  - Parámetros de salida:
    - *Value*: Importe que contenía el/los cajón/es vaciado/s antes de vaciarlo/s.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **`GetAmounts (`*`Denoms As String, Quantities As String, Value As Int32`*`) as Boolean`**

  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: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **`GetLastIn () as String`**

  Función sin pantalla para obtener el detalle de como se ha introducido el último importe.
  - Devuelve:

    Lista de las denominaciones ordenadas según se han ido introduciendo. En caso de superar las 50 unidades devuelve el importe introducido.

- **GetLevels (*LevelType As CKL_Levels, Denoms As String, Levels As String*) as Boolean**

  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.
  - Parámetros de entrada:
    - *LevelType*: valores posibles:
      0 = CKL_Initial
      1 = CKL_Max
      2 = CKL_Current
      3 = CKL_CashBox
      4 = CKL_LevelStatus
  - 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, este contendrá los códigos de los semáforos (0 = verde, 1 = naranja, 2 = rojo).
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

  ATENCIÓN: En caso de que algún dispositivo esté en error, el contenido de dicho dispositivo puede no mostrarse.

- **Pay (*ValueInCents As Int32, PaidOutValue As Int32*) as Boolean**

  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: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **PaySpecific (*ValueInCents As Int32, PaidOutValue As Int32*) as Boolean**

  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: ha terminado correctamente.
  - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

9.1.2. Propiedades

- **`AmountFactor`**

  Contiene el factor de conversión entre la unidad de la divisa y los importes a enviar y recibir con CashKeeper.
  - Ejemplo:
    - En euros, el factor de conversión es 100, ya que los importes son en céntimos.
    - En la moneda chilena el factor de conversión actualmente es 1.

- **`BackColor`**

  Indica el color de fondo de los formularios.

- **`ErrorCode`**

  Contiene el número (código) de error que se ha producido en el CashKeeper.

- **`ErrorDescription`**

  Contiene la descripción del error que se ha producido en el CashKeeper.

- **`InverseColor`**

  Indica el color de las letras, se calcula automáticamente al cambiar la propiedad *BackColor*.

- **`IP`**

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

- **`OnErrorDiscard`**

  Puede suceder que durante una operación de cobro, se produzca un problema (tipo corte eléctrico) y quede un saldo “pendiente”. Si esta propiedad está desactivada (FALSE), al reiniciar la máquina y realizar el primer cobro, tendrá en cuenta el saldo “pendiente” de la operación anterior. En caso contrario, la propiedad este activada (TRUE), el saldo “pendiente” no se tendrá en cuenta.

- **`ShowDecimals`**

  Contiene el número de decimales a mostrar en los importes.
  - Ejemplo:
    - En euros, los decimales a mostrar es 2.
    - En la moneda chilena no utilizan decimales, así pues, el número de decimales a mostrar es 0.

9.2. Listado funciones y propiedades bajo nivel (*CKeeper.dll/Raw Socket*)

9.2.1. Funciones

NOTA: Todos los valores relativos a valores de denominación o importes se deben enviar y se devuelven en función del factor de conversión obtenido en la función ***GetCCDetails****.*

- **`(0) AbortTimer ()`**

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

- **`(3) AcceptPending ()`**

  Después de desencadenarse alguno de los eventos ***ProtectedValueNote***, ***MaxCoinsWarning***, 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.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

  NOTA: Solo valido en algunos modelos de billeteros concretos.

- **`(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'.
  - Parámetro de entrada:
    - *Operation*: Número de la operación (de 0 a 9)
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **`(6) CleanBulk (`*`Complete as Boolean`*`) as Boolean`**

  Desencadena un ciclo de limpieza del dispositivo de entrada de monedas.
  - Parámetros de entrada:
    - *Complete*: tipo de limpieza
      - 0 = FALSE: ciclo de limpieza corto
      - 1 = TRUE: ciclo de limpieza completo

- Devuelve:
  - Verdadero: ha terminado correctamente.
  - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(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.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(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 lo siguiente:
  - Parámetros de salida:
    - LowLevelActive: Indica si existe alguna denominación por debajo del nivel de alarma.
      - 0 = FALSE: No hay denominaciones en estado de alarma por nivel bajo.
      - 1 = TRUE: Existen denominaciones en estado de alarma por nivel bajo. Consultar niveles a través de la función ***GetCurrentLevel***.
    - HopperFull: Indica si se ha rebasado la capacidad de reciclador.
      - 0 = FALSE: Nivel del reciclador de monedas correcto.
      - 1 = TRUE: Reciclador de monedas lleno. Debe ser vaciado almenos parcialmente.
    - CoinCashBoxState: Nivel de llenado de cajón de recaudación monedas.
      - LCCBS_OK: Nivel del cajón de recaudación de monedas correcto.
      - LCCBS_AlmostFull: Nivel del cajón de recaudación de monedas superior al 90%.
      - LCCBS_Full: Nivel del cajón de recaudación de monedas excedido. Debe ser vaciado.
    - NoteCashBoxState: Nivel de llenado de cajón de recaudación billetes.
      - LCCBS_OK: Nivel del cajón de recaudación de billetes correcto.
      - LCCBS_AlmostFull: Nivel del cajón de recaudación de billetes superior al 90%.
      - LCCBS_Full: Nivel del cajón de recaudación de billetes excedido. Debe ser vaciado.

      NOTA: Se puede activar el LCCBS_Full sin antes haberse activado el LCCBS_AmostFull.

- Devuelve:
  - Verdadero: ha terminado correctamente.
  - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(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, así como la compatibilidad con el dispositivo especificado. Cabe remarcar, que el "path" es relativo a la ubicación del CSSI.
  - Parámetros de entrada:
    - *FullPathFile*: Path relativo del fichero done se chequea el firmware i dataset.
    - *Device_ID*: Tipo de dispositivo seleccionado
      - 0 = LCS_CoinHopper
      - 1 = LCS_NotePayout
  - Parámetros de salida:
    - *FirmwareVersion*: Versión del firmware chequeado
    - *DataSet*: Versión del dataset chequeado
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(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.
  - Parámetros de salida:
    - *PayBack*: Si queremos que devuelva el importe introducido
      - 0 = FALSE: No devuelve el importe introducido.
      - 1 = TRUE: Devuelve el importe introducido.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(52) DiscardOperation (*Operation as Byte*) as Boolean**

  Descarta la operación indicada, dejando los valores pendientes de dicha operación a cero.
  - Parámetros de entrada:
    - *Operation*: Operación a descartar.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

ATENCIÓN: Esta función no provoca la devolución del importe introducido.

- **(68) DiscardPayOperation () as Boolean**

  Descarta la operación de pago interrumpida.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

  ATENCIÓN: Esta función no provoca la devolución de ningún importe.

- **Disconnect()**

  Desconecta la conexión con CashKeeper.

- **(10) Display (*Line1 as String, Line2 as String, UseBigFont as Boolean*) as Boolean**

  Envía un mensaje a la pantalla del dispositivo. Cada una de las líneas no puede contener más de 20 caracteres. En caso de activar UseBigFont (texto de altura doble) el texto contenido en Line2 no se considerará.
  - Parámetros de entrada:
    - *Line1*: Texto a mostrar.
    - *Line2*: Texto a mostrar.
    - *UseBigFont*: Activación del texto de altura doble.
      - 0 = FALSE: Texto de altura doble desactivado.
      - 1 = TRUE: Texto de altura doble activado.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

  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 especificado. Se debe utilizar cada vez que se vacían los cajones de recaptación.
  - Parámetros de entrada:
    - *Device_ID*: Cajón a resetear.
      - 1 = LCCB_Coins
      - 2 = LCCB_Notes
      - 5 = LCCB_All
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(11) EmptyCashBoxEx (*CashBox As LC_CashBoxes*) As Boolean**

  Extensión de la función ***EmptyCashBox*** que permite vaciar los cajones de recaudación de billetes individualmente en las máquinas de más de un billetero.
  - Parámetros de entrada:
    - *CashBox*: Cajón a vaciar.
      - 1 = CashBox_1
      - 2 = CashBox_2
      - 3 = CashBox_3
      - 4 = CashBox_4
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(12) EmptyDevice (*Device_ID as LC_CashBox*) as Boolean**

  Envía el contenido del reciclador al cajón de recaudación del dispositivo especificado.
  - Parámetros de entrada:
    - *Device_ID*: Cajón a vaciar.
      - 1 = LCCB_Coins
      - 2 = LCCB_Notes
      - 5 = LCCB_All
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(13) EmptyDeviceSpecific (*Denoms as String, NumberToKeep as String*) as Boolean**

  Envía hacia el cajón de recaudación la cantidad que corresponda de cada una de las denominaciones especificadas para dejar en los recicladores (cambio) los valores correspondientes especificados.
  - Parámetros de entrada:
    - *Denoms*: Valor de la/s denominación/es a consultar (en caso de ser varias, se debe indicar separado por comas).
    - *NumberToKeep*: Nivel/es de la/s denominación/es a mantener en la máquina (en caso de ser varias, indicar los valores separados por comas en el mismo orden que se han introducido en *Denoms*)
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

  NOTA: Toda denominación no presente en la lista Denoms, será completamente vaciada.

- **(14) Enable () as Boolean**

  Habilita el dispositivo para dejarlo en estado de recepción de efectivo.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(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.
  - Parámetros de entrada:
    - *Value*: Valor de la denominación a modificar.
    - *Level*: Nivel de la denominación
    - *AddToCurrentLevel*:
      - 0 = FALSE: Sustituye al nivel que contiene (reemplaza).
      - 1 = TRUE: Añade al nivel que contiene (suma).
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

  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

** Ver anexo de propiedades*

- **(16) GetBrokenCents (*Value as INT32, ResetValue as Boolean*) as Boolean**

  Almacena 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 y los almacena en un variable (contador).
  - Parámetros de entrada:
    - *Value*: Almacena la cantidad de céntimos de más que se han devuelto
    - *ResetValue*: Reseteo del contador.
      - 0 = FALSE: No resetea el contador.
      - 1 = TRUE: Resetear el contador.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(17) GetCashBoxLevel (*CountryCode As String, Values As String, Levels As String*) As Boolean**

  Devuelve el nivel contenido en el cajón de recaudación.
  - Parámetros de salida:
    - *CountryCode*: Moneda a la que hacen referencia los valores (EUR, GBP, USD, etc.).
    - *Values*: Valor de la/s denominación/es a consultar (en caso de ser varias, se debe indicar separado por comas).
    - *Levels*: Nivel/es de la/s denominación/es pedida/s (en caso de ser varias, devuelve los valores separados por comas en el mismo orden que se han consultado).
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(77) GetCCChange (*CountryCode As String, Change As Int32*) As Boolean**

  Devuelve el importe cambio de la divisa especificada sobre la moneda principal.
  - Parámetros de entrada:
    - *ContryCode*: Divisa a consultar
  - Parámetros de salida:
    - *Change*: Importde de cambio
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(86) GetCCDetails (*CC As String, Multiplier As int32, Decimals As Int32*) As Boolean**

  Devuelve el factor de conversión (multiplicador) y el número de decimales a mostrar. Permite hacer que nuestra aplicación se adapte a la moneda de otros países.
  - Parámetros de entrada:
    - *CC*: Divisa a consultar.
  - Parámetros de salida:
    - *Multiplier*: Multiplicador de la divisa.
    - *Deciamles*: Decimales de la divisa.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.
  - Ejemplos:
    - En euros, el multiplicador es 100, ya que los importes son en céntimos, el número de decimales a mostrar es 2.
    - En la 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 divisa separado por monedas y billetes.
  - Parámetros de entrada:
    - *CountryCode*: Divisa a consultar.
  - Parámetros de salida:
    - *Coins*: Lista de valores de las monedas (separados por comas).
    - *Deciamles*: Lista de valores de los billtes (separados por comas).
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(24) GetCounters (*CoinCounter as Int32, NoteCounter as Int32*) as Boolean**

  Devuelve el valor de los contadores absolutos (número total de unidades de entrada) de monedas y billetes desde la fabricación del dispositivo.
  - Parámetros de salida:
    - *CoinCounter*: Valor de los contadores absolutos de monedas.
    - *Deciamles*: Valor de los contadores absolutos de billetes.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(38) GetCountryCodes (*MainCC As String, OtherCC As String*) As Boolean**

  Devuelve el código de la divisa principal (EUR, GBP, MXN, USD, etc.) y la lista de codigos de las divisas soportadas.
  - Parámetros de salida:
    - *MainCC*: Código de la divisa principal.
    - *OtherCC*: Lista de códigos de divisas soportadas (separadas por comas).
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(19) GetCurrentLevel (*Values as String, Levels as String*) as Boolean**

  Devuelve la cantidad contenida en cambio de la/s denominación/es especificada/s.
  - Parámetros de entrada:
    - *Values*: Valor de la/s denominación/es a consultar (en caso de ser varias, se debe indicar separado por comas).
  - Parámetros de salida:
    - *Levels*: Nivel/es de la/s denominación/es pedida/s (en caso de ser varias, devuelve los valores separados por comas en el mismo orden que se han consultado).
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

  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. Solo valido para máquinas USB.
  - Parámetros de salida:
    - *DeviceList*: Lista de identificadores de CashKeeper.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(54) GetFirmwareVersion (*Device as LC_Logical_Devices, FirmwareVersion as string, Dataset as string*) as Boolean**

  Devuelve el firmware y el dataset actuales del dispositivo especificado.
  - Parámetros de entrada:
    - *Device*: Dispositivo a consultar.
      - 1 = LCLD_CoinsValidator
      - 2 = LCLD_CoinDispenser
      - 3 = LCLD_NoteRecycler
  - Parámetros de salida:
    - *FirmwareVersion*: Firmware del dispositivo.
    - *Dataset*: DataSet del dispositivo.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(21) GetInhibitState (*CountryCode As String, Values As String, Inhibits As String*) As Boolean**

  Recupera el estado de inhibición (permiso de aceptación) de las denominaciones y divisa especificadas.
  - Parámetros de entrada:
    - *CountryCode*: Divisa a consultar.
    - *Values*: Valor de la/s denominación/es a consultar (en caso de ser varias, se debe indicar separado por comas).
  - Parámetros de salida:
    - *Inhibit*: Estado/s de aceptación de la/s denominación/es pedida/s (en caso de ser varias, devuelve los valores separados por comas en el mismo orden que se han consultado).
      - 0: La denominación se acepta.
      - 1: La denominación se acepta, pero no se paga
      - 2: La denominación se rechaza
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(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.
  - Parámetros de entrada:
    - *Detail*: Modo de consulta
      - TRUE: Devuelve la lista ordenada de denominaciones utilizadas en el pago. En caso de ser superior a 50 denominaciones, devuelve el valor acumulado y modifica el parámetro *Detail*.
      - FALSE: Devuelve el importe total en cada divisa utilizada en el pago.
    - *Operation*: Número de operación a consultar (255 = operación activa)
  - Parámetros de salida:
    - *CountryCodes*: Divisa de cada denominación introducida o de cada total (dependiedo del modo seleccionado en *Detail*).
    - *Detail*: Modo de consulta (puede haber sido modificado).
    - *AmountsOrDenom*: Muestra de datos según el modo *Detail* elegido:
      - *Detail* = TRUE: Lista de las denominaciones introducidas en el último cobro
      - *Detail* = FALSE: Importe introducido por cada una de las divisas implicadas en el último cobro
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(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***.
  - Parámetros de entrada:
    - *ConfigID*: Número identificativo de la máquina (Solo útil en máquinas USB).
  - Parámetros de salida:
    - *Denoms*: Valor de las denominaciónes (valores separados por comas).
    - *Qtys*: Niveles de las denominaciónes (valores separados por comas).
    - *CCs*: Valor de la divisa (EUR, GBP…).
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(8) GetLowLevelNotes (*Values As String, Levels As String*) As Boolean**

  Devuelve la lista de niveles mínimos de la lista de denominaciones especificada.
  - Parámetros de entrada:
    - *Values*: Valor de la/s denominación/es a consultar (en caso de ser varias, se debe indicar separado por comas).
  - Parámetros de salida:
    - *Levels*: Nivel/es de la/s denominación/es pedida/s (en caso de ser varias, devuelve los valores separados por comas en el mismo orden que se han consultado).
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(23) GetMaxLevel (*Values as String, Levels as String*) as Boolean**

  Devuelve la lista de niveles máximos de la lista de denominaciones especificada.
  - Parámetros de entrada:
    - *Values*: Valor de la/s denominación/es a consultar (en caso de ser varias, se debe indicar separado por comas).
  - Parámetros de salida:
    - *Levels*: Nivel/es de la/s denominación/es pedida/s (en caso de ser varias, devuelve los valores separados por comas en el mismo orden que se han consultado).
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(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.
  - Parámetros de salida:
    - *HostName*: Nombre de la máquina.
    - *DHCPEnabled*: Si el DHCP está habilitado.
      - 0 = FALSE: No está habilitado.
      - 1 = TRUE: Está habilitado.
    - *Gateway*: Puerta de enlace.
    - *DNS1*: Servidor DNS preferido.
    - *DNS2*: Servidor DNS alternativo.

- *MasterPort*: Puerto de control.
- *OfficePort*: Puerto “office”.
- *Seed*: Contraseña de seguridad.

- Devuelve:
  - Verdadero: ha terminado correctamente.
  - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(63) GetProperties (*Property as LC_Properties, Value as Int32*) as Boolean**

  Devuelve un identificador único para el dispositivo CashKeeper.

  - Parámetros de entrada:
    - *Property*: Codigo de la propiedad
      - 0 = P_BCMaxCoins
      - 1 = P_BCMinValue
      - 2 = P_CancelLowLevel
      - 3 = P_CancelValueEvents
      - 5 = P_CoinCashBoxDetect
      - 6 = P_DisableAutoText
      - 7 = P_LightTime
      - 8 = P_LogDisable
      - 9 = P_MaxCoins
      - 10 = P_MaxPayout
      - 11 = P_MinFastIn
      - 12 = P_NLPAutoProtect
      - 13 = P_NLPPercentValue
      - 14 = P_NLPStartValue
      - 15 = P_NoteLevelProtection
      - 16 = P_PayoutInterval
      - 17 = P_RejectIfClosed
      - 18 = P_CoinsLowLevel
      - 19 = P_DeviceType
      - 20 = P_BarcodeLenght
      - 21 = P_CriticalBehavior
  - Parámetros de salida:
    - *Value*: Valor de la propiedad.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

  *NOTA: Ver apartado **9.2.2. Propiedades** para la definición.*

- **(74) GetUnikeID (*ID as string*) as Boolean**

  Devuelve un identificador único para el dispositivo CashKeeper.

  - Parámetros de salida:
    - *ID*: Identificador único.

- Devuelve:
  - Verdadero: ha terminado correctamente.
  - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

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 (pago) por el valor especificado (la repartición de denominaciones en monedas y billetes se realiza de forma automática). Dependiendo del modo especificado iniciará el proceso de pago o comprobará que pueda realizar el pago deseado.
  - Parámetros de entrada:
    - *Value*: Valor a pagar.
    - *TestOnly*: Modo de pago
      - 0 = FALSE: Inicia el proceso de pago.
      - 1 = TRUE: Realiza un testeo de si es posible pagar dicha cantidad.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(30) PaySpecific (*Denoms as String, NumberOf as String, TestOnly as Boolean*) as Boolean**

  Desencadena un proceso de salida de efectivo (pago) con las denominaciones y cantidades especificadas. Dependiendo del modo especificado iniciará el proceso de pago o comprobará que pueda realizar el pago deseado.
  - Parámetros de entrada:
    - *Value*: Valor de la/s denominación/es a pagar (en caso de ser varias, se debe indicar separado por comas).
    - *NumberOf*: Cantidad de la/s denominación/es a pagar (en caso de ser varias, indicar los valores separados por comas en el mismo orden que se han introducido en *Value*).
    - *TestOnly*: Modo de pago
      - 0 = FALSE: Incia el proceso de pago.
      - 1 = TRUE: Realiza un testeo de si es posible pagar dicha cantidad.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(87) PaySpecificEx (*Denoms as String, NumberOf as String, TestOnly as Boolean*) as Boolean**

  Desencadena un proceso de salida de efectivo (pago) con las denominaciones y cantidades especificadas. Dependiendo del modo especificado iniciará el proceso de pago o comprobará que pueda realizar el pago deseado.
  - Parámetros de entrada:
    - *Value*: Valor de la/s denominación/es a pagar (en caso de ser varias, se debe indicar separado por comas).
    - *NumberOf*: Cantidad de la/s denominación/es a pagar (en caso de ser varias, indicar los valores separados por comas en el mismo orden que se han introducido en *Value*).
    - *TestOnly*: Modo de pago
      - 0 = FALSE: Incia el proceso de pago.
      - 1 = TRUE: Realiza un testeo de si es posible pagar dicha cantidad.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

  NOTA: 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* se deberá responder a dichos eventos con ***AcceptPending*** para aceptar o con ***RejectPending*** para rechazarlo.

- **(33) Reset () as Boolean**

  Resetea el dispositivo y lo reinicializa. Función obsoleta.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(75) SetCCChange (*CountryCode As String, Change As Int32*) As Boolean**

  Establece el cambio de la divisa especificada respecto de la divisa principal.
  - Parámetros de entrada:
    - *CountryCode*: Divisa no principal.
    - *Change*: Importe de cambio.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

*NOTA: La fórmula que se realiza para calcular el importe durante 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. 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 especificando el código y el texto. Ver **9.7. Relación de códigos y textos del display en modo automático** para consultar la tabla de códigos.
  - Parámetros de entrada:
    - *Code*: Código del texto a modificar.
    - *NewText*: Nuevo texto a mostrar en el "display".
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.
  - Ejemplo:
    - Originalmente el código 17 de texto de "display" contiene el texto "A PAGAR:", si este texto lo quisiéramos modificar por "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 y divisa especificadas.
  - Parámetros de entrada:
    - *CountryCode*: Divisa a assignar.
    - *Values*: Valor de la/s denominación/es a asignar (en caso de ser varias, se debe indicar separado por comas).
    - *Inhibit*: Estado de aceptación de la/s denominación/es a asignar (en caso de ser varias, indicar los valores separados por comas en el mismo orden que se han introducido en *Values*).
      - 0: La denominación se acepta.
      - 1: La denominación se acepta, pero no se paga (solo válido para monedas).
      - 2: La denominación se rechaza.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(70) SetLanguage (*Idioma As Idiomas*) As Boolean**

  Establece el idioma por defecto que utilizará CashKeeper para comunicarse vía "display" y mensajes de texto.
  - Parámetros de entrada:
    - *Idioma*: Idioma a asignar.
      - 1 = ESP (Español).
      - 2 = ENG (Ingles).
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(69) SetLogPath (*NewPath as String*) as Boolean**

  Establece el "path" de almacenamiento de los archivos de .LOG a la ruta especificada de forma permanente. Este "path" debe ser relativo al equipo donde se está ejecutando el CSSI. Únicamente para máquinas USB.
  - Parámetros de entrada:
    - *NewPath*: "path" relativo donde almacenar los archivos .LOG.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(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".
  - Parámetros de entrada:
    - *Values*: Valor/es de la/s denominación/es a asignar (en caso de ser varias, se debe indicar separado por comas).
    - *Levels*: Nivel/es de la/s denominación/es a asignar (en caso de ser varias, indicar los valores separados por comas en el mismo orden que se han introducido en *Values*).
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(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.
  - Parámetros de entrada:
    - *Values*: Valor/es de la/s denominación/es a asignar (en caso de ser varias, se debe indicar separado por comas).

  - *Levels*: Nivel/es de la/s denominación/es a asignar (en caso de ser varias, indicar los valores separados por comas en el mismo orden que se han introducido en *Values*).
- Devuelve:
  - Verdadero: ha terminado correctamente.
  - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

NOTA: El modelo CK1000 el nivel máximo de los billetes este forzado a 26.

- **(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 los valores de la configuración de red.
  - Parámetros de entrada:
    - *HostName*: Nombre de la máquina.
    - *DHCPEnabled*: Si el DHCP está habilitado.
      - 0 = FALSE: No está habilitado.
      - 1 = TRUE: Está habilitado.
    - *Gateway*: Puerta de enlace.
    - *DNS1*: Servidor DNS preferido.
    - *DNS2*: Servidor DNS alternativo
    - *MasterPort*: Puerto de control
    - *OfficePort*: Puerto “office”
    - *Seed*: Contraseña de seguridad
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(63) SetProperties (*Property as LC_Properties, Value as Int32*) as Boolean**

  Devuelve un identificador único para el dispositivo CashKeeper.
  - Parámetros de entrada:
    - *Property*: Codigo de la propiedad
      - 0 = P_BCMaxCoins
      - 1 = P_BCMinValue
      - 2 = P_CancelLowLevel
      - 3 = P_CancelValueEvents
      - 5 = P_CoinCashBoxDetect
      - 6 = P_DisableAutoText
      - 7 = P_LightTime
      - 8 = P_LogDisable
      - 9 = P_MaxCoins
      - 10 = P_MaxPayout
      - 11 = P_MinFastIn

        - 12 = P_NLPAutoProtect
        - 13 = P_NLPPercentValue
        - 14 = P_NLPStartValue
        - 15 = P_NoteLevelProtection
        - 16 = P_PayoutInterval
        - 17 = P_RejectIfClosed
        - 18 = P_CoinsLowLevel
        - 19 = P_DeviceType
        - 20 = P_BarcodeLenght
        - 21 = P_CriticalBehavior
    - *Value*: Valor de la propiedad.
- Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

*NOTA: Ver apartado **9.2.2. Propiedades** para la definición.*

- **(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.
  - Parámetros de salida:
    - *Configuration*: Identificador de la configuración que queremos iniciar, en caso de no existir, se crea.
    - *Device*: Parámetro opcional (solo se usa en las máquinas USB) 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.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

  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 valor objetivo establecido en la función ***Totalize*** de la operación especificada.

- **(48) Terminate**

  Este método es equivalente a ejecutar ***CloseAll*** y ***Disconnect***. En las máquinas USB si el CSSI está 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.
  - Parámetros de entrada:
    - *Total_Value*: Importe total de una operación.
    - *AutoClose*: Vuelta de cambio
      - 0 = FALSE: No se cierra la operación.
      - 1 = TRUE: Si hay importe suficiente, devuelve el cambio de forma automática.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

  NOTA: 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.
  - Parámetros de entrada:
    - *FullPathFile*: Ubicación (“path”) del archivo de actualización.
    - *Device_ID*: Tipo de dispositipo a actualizar
      - 1 = LCSD_CoinHopper
      - 2 = LCSD_NoteFloat
      - 3 = LCSD_NotePayout
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(43) ValueIN (*Value as Int32, Operation as Byte*) as Boolean**

  Devuelve el valor total introducido en la operación especificada.
  - Parámetros de entrada:
    - *Operation*: Número de operación a consultar (255 = operación activa).
  - Parámetros de salida:
    - *Value*: Valor total introducido en la operación especificada.
  - Devuelve:
    - Verdadero: ha terminado correctamente.

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

- **(44) ValueOUT (*Value as Int32*) as Boolean**

  Devuelve el valor total pagado hasta el momento.
  - Parámetros de salida:
    - *Value*: Valor total pagado hasta el momento.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

- **(45) ValueToCashBox (*Value as Int32*) as Boolean**

  Devuelve el valor total enviado a cajón de recaudación hasta el momento.
  - Parámetros de salida:
    - *Value*: Valor total enviado a cajón hasta el momento.
  - Devuelve:
    - Verdadero: ha terminado correctamente.
    - Falso: se ha producido algún error, consulte las propiedades *ErrorCode* y *ErrorDescription* para más información.

9.2.2. Propiedades

- **`(20) BarCodeLength as Int32`**

  Especifica/devuelve 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. Función obsoleta.

- **`(0) BCMaxCoins as Int32`**

  Especifica el número de monedas mínimo para permitir el ***BrokenCent*** extendido.

- **`(1) BCMinValue as Int32`**

  Especifica el importe mínimo para permitir el ***BrokenCent*** extendido.

- **`(2) CancelLowLevel as Int32`**

  Permite desactivar los eventos de niveles bajo mínimos. No recomendado.
  - Valores permitidos:
    - 0 = FALSE
    - 1 = TRUE

- **`(3) CancelValueEvents as Int32`**

  Permite desactivar los eventos de valor (*ValueIN, ValueOUT, CashBoxValue*).
  - Valores permitidos:
    - 0 = FALSE
    - 1 = TRUE

- **`(5) CoinCashBoxDetect as Int32`**

  Especifica si se controla que el Cajón de recaudación de monedas este colocado.
  - Valores permitidos:
    - 0 = FALSE
    - 1 = TRUE

- **`(18) CoinsLowLevel as Int32`**

  Especifica el número de monedas mínimo debajo del cual se empezará a avisar de que estamos en niveles bajos (evento *LowLevel*). Valor único para todas las denominaciones de monedas.

- **`(4) ConfigID as Int32`**

  Propiedad de solo lectura que devuelve el identificador de configuración activo (el que se ha utilizado en la función ***StartUp***). Solo útil en máquinas USB.

- **(*) ConnectionType as Int32**

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

- **(21) CriticalBehavior as Int32**

  Propiedad que controla como se procederá en caso de nivel crítico de moneda. Una moneda entra en situación de crítico cuando está en franca desventaja respecto de las demás.

  Ejemplo: de todas las monedas tenemos 200 unidades, pero de una solo tenemos 30).

  - Valores permitidos:
    - 0 = Se enviarán 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 realizarán 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 as Int32**

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

- **(6) DisableAutoText as Int32**

  Indica si el “display” del CashKeeper funciona de forma autónoma o no.
  - Valores permitidos:
    - 0 = FALSE
    - 1 = TRUE

- **(*) ErrorCode as Int32**

  Propiedad de solo lectura que informa del código de error de la última función ejecutada.

- **(*) ErrorDescription as String**

  Propiedad de solo lectura que informa de la descripción del error producido en la última función ejecutada.

- **(*) HostIP as String**

  Propiedad de solo lectura que devuelve la IP informada en el comando ***Connect***.

- **(*) HostPort as uInt16**

  Propiedad de solo lectura que devuelve el puerto Host indicado en el comando ***Connect***.

- **(7) LightTime as Int32**

  Indica el tiempo que el led de salida de monedas estará abierto cuando se produzca una salida de moneda sea cual sea la causa. Esta en milisegundos.

- **(8) LogDisable as Int32**

  Indica si se desactiva o no los logs para posterior diagnóstico. Función obsoleta.
  - Valores permitidos:
    - 0 = FALSE
    - 1 = TRUE

  NOTA: Muy recomendable de tener siempre en FALSE.

- **(9) MaxCoins as Int32**

  Indica a partir de cuantas monedas al realizar un pago, el sistema nos avisara de que vamos a sobre pasar el límite indicado.

- **(10) MaxPayout as Int32**

  Indica el 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 as Int32**

  Indica a partir de que numero de billetes se considera un “pago largo” y se enviaran los billetes a cajón de recaudación para acelerar el cobro.

  NOTA: Solo efectivo en los algunos modelos de billeteros.

- **(12) NLPAutoProtect as Int32**

  Indica si la protección de cambio en billetes decide de forma autónoma o pregunta al usuario.
  - Valores permitidos:
    - 0 = Pregunta al usuario
    - 1 = Automático

- **(13) NLPPercentValue as Int32**

  Indica el porcentaje de cambio en billetes a proteger.

- **(14) NLPStartValue as Int32**
  Indica a partir de que valor de billete actuará la protección de cambio.

- **(15) NoteLevelProtection as Int32**
  Indica si la protección de cambio en billetes esta activa.
  - Valores permitidos:
    - 0 = FALSE
    - 1 = TRUE

- **(*) OfficePort as uInt16**
  Indica el puerto de conexión para conexiones tipo OFFICE.

- **(16) PayoutInterval as Int32**
  Indica el periodo en protección de pagos en minutos.

- **(17) RejectIfClosed as Int32**
  Indica si en estado de reposo de CashKeeper, la puerta de introducción de monedas estará en posición de rechazo de monedas.
  - Valores permitidos:
    - 0 = FALSE
    - 1 = TRUE

  NOTA: Solo efectivo en los algunos modelos de monederos.

- **(*) SecuritySeed as Int32**
  Numero de seguridad para la conexión debe de tener 6 dígitos.

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

(*) Solo disponibles en método *CKeeper*

9.2.3. 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 ***RejectPending*** devuelve el "billete" que contiene el código de barras (apto para carnets, etc.). Función obsoleta.

- **(115) BarcodeStored (*Code as String*)**

  Se desencadena cuando un "billete" que contiene un código de barras se ha almacenado en cajón de recaudación correctamente. Función obsoleta.

- **(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 asíncrona.
  - Parámetros de salida:
    - *ErrorCode*: Codigo de error ('0' si no hay ningún error).
    - *ErrorDescription:* Descripción del error.
    - *CurrentValue:* Importe entrado/pagado.
    - *TargetValue:* Valor objetivo establecido.
    - *State:* Tipo de operación.
    - *Operation:* Número de 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*.**
  - Parámetros de salida:
    - *ValuesInfo:* Denominaciones que responden al evento.
    - *LevelsInfo:* 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*. Se debe responder con las funciones ***RejectPending*** o ***AcceptPending*** antes de 60 segundos.
  - Parámetros de salida:
    - *ValuesInfo:* Denominaciones que responden al evento.
    - *LevelsInfo:* Niveles actuales de cada una de las denominaciones informadas en *ValuesInfo*.

- **(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).
  - Parámetros de salida:
    - *NotePresent*: Informa del estado del billete
      - 0 = FALSE: Billete retirado.
      - 1 = TRUE: Billete presente.

  NOTA: Solo efectivo en los algunos modelos de billeteros.

- **(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.
  - Parámetros de salida:
    - *State:* Estado del dispositivo en el momento de la interrupción.
    - *Value:* Valor (de entrada o salida) al que se había llegado hasta el momento de la interrupción.
    - *Target:* Valor objetivo (de entrada o salida) al que se debía llegar en el momento de la interrupción.
    - *Operation:* Número de operación que no se pudo completar.
  - Ejemplo:
    - 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.

- **(105) ProtectedValueNote (*Value as Int32, PayoutNeeds as Int32, TotalChange as Int32, Operation As Byte*)**

  Se desencadena si la propiedad *NLPAutoProtect* está en falso. Cuando se desencadena ***ProtectedValueNote.*** Se debe responder con las funciones ***AcceptPending*** o ***RejectPending*** antes de 60 segundos.
  - Parámetros de salida:
    - *Value*: Denominación del billete.
    - *PayoutNeeds*: Importe de la operación.
    - *TotalChange*: Cambio disponible en billetes.
    - *Operation:* Número de operación que no se pudo completar.

  *NOTA: Ver apartado **7.13.2.2 Note Level Protection.***

- **(106) StateChange (*State as Byte, OldState as Byte*)**

  Se desencadena cada vez que el estado del sistema cambia.
  - Parámetros de salida:
    - *State:* Nuevo estado

- *OldState:* 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.
  - Parámetros de salida:
    - *CurrentValue*: Importe introducido hasta el momento.
    - *Target*: Importe total de la operación.
    - *Operation:* Número de operación que no se pudo completar.

- **(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.
  - Parámetros de salida:
    - *CurrentValue*: Importe pagado hasta el momento.
    - *Target*: Importe total a pagar.
    - *Operation:* Número de operación que no se pudo completar.

- **(109) ValueToCashBox (*CurrentValue as Int32, Target as Int32, Operation As Byte*)**

  Se desencadena en las acciones de salida de efectivo hacia el cajón de recaudación y la propiedad *CancelValueEvents* contiene falso.
  - Parámetros de salida:
    - *CurrentValue*: Importe enviado a cajón de recaudación hasta el momento.
    - *Target*: Importe total a enviar a cajón de recaudación.
    - *Operation:* Número de operación que no se pudo completar.

- **(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*.
  - Parámetros de salida:
    - *Code:* Codigo de “warning”.
    - *Description:* Descripción del “warning”.

9.3. Descripción de constantes

- **CKL_Levels**
  - 0 = CKL_Initial
  - 1 = CKL_Max
  - 2 = CKL_Current
  - 3 = CKL_CashBox

- **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_CashBoxes**
  - 1 = CashBox_1
  - 2 = CashBox_2
  - 3 = CashBox_3
  - 4 = CashBox_4

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

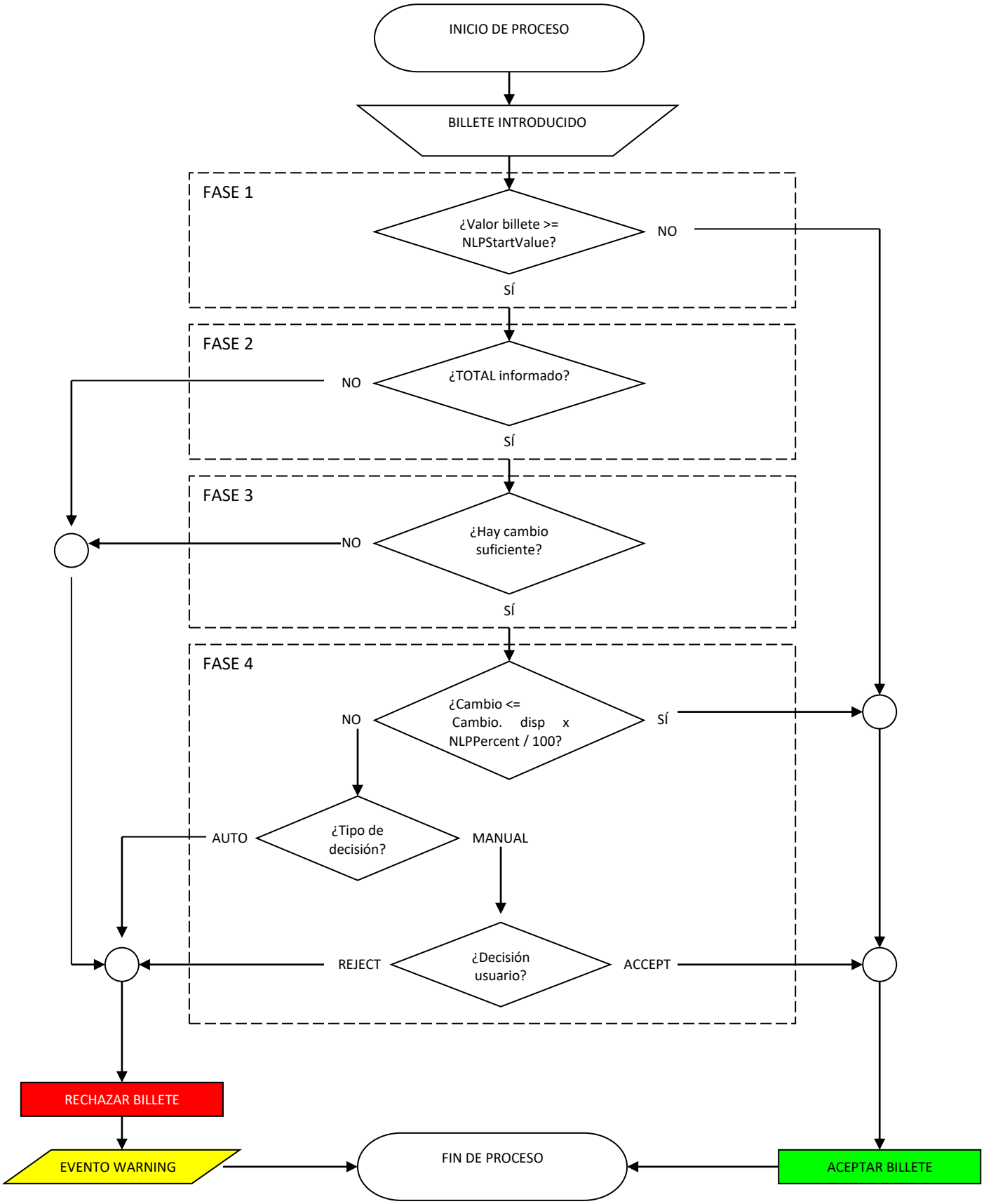
- **LC_Properties**
  - 0 = P_BCMaxCoins
  - 1 = P_BCMinValue
  - 2 = P_CancelLowLevel
  - 3 = P_CancelValueEvents
  - 5 = P_CoinCashBoxDetect
  - 6 = P_DisableAutoText
  - 7 = P_LightTime
  - 8 = P_LogDisable
  - 9 = P_MaxCoins
  - 10 = P_MaxPayout
  - 11 = P_MinFastIn
  - 12 = P_NLPAutoProtect
  - 13 = P_NLPPercentValue
  - 14 = P_NLPStartValue
  - 15 = P_NoteLevelProtection
  - 16 = P_PayoutInterval
  - 17 = P_RejectIfClosed
  - 18 = P_CoinsLowLevel
  - 19 = P_DeviceType
  - 20 = P_BarcodeLenght
  - 21 = P_CriticalBehavior

- **CBC_Modes**
  - 0 = Normal
  - 1 = CBC_Disable_Config
  - 4 = CBC_Disable_Actions
  - 5 = Combinación de modo *CBC_Disable_Config* y *CBC_Disable_Actions*
  - 7 = CBC_Blind
  - 15 = CBC_Blind_By_Value

- **Idiomas**
  - 1 = Español
  - 2 = English

9.4. Imágenes

9.4.1. Diagrama 'Note Level Protection'



9.5. Tablas de errores y warnings

9.5.1. Tabla 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 denominaciones 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)

9.5.2. Tabla 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

9.6. Listado de funciones por ejecución SÍNCRONA o ASÍNCRONA.

9.6.1. 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	Com.	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*

9.6.2. 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	**Com.**	**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

9.7. 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