(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 743 045**

(51) Int. CI.:

G06F 9/445 (2008.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **25.05.2005 PCT/BR2005/000094**

(87) Fecha y número de publicación internacional: **08.12.2005 WO05115074**

(96) Fecha de presentación y número de la solicitud europea: **25.05.2005 E 05746722 (7)**

(97) Fecha y número de publicación de la concesión europea: **01.05.2019 EP 1769342**

(54) Título: **Sistema para acceder a un terminal de PDV, método para descargar y actualizar aplicaciones y método para llevar a cabo una operación electrónica usando un sistema de ese tipo**

(30) Prioridad:

25.05.2004 US 574134 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
18.02.2020

(73) Titular/es:

MUXI TECNOLOGIA EM PAGAMENTOS S.A. (100.0%)
Rua do Carmo, n° 43 , 9° andar, Centro
20011-020 - Rio de Janeiro - RJ, BR

(72) Inventor/es:

SOARES PI FARIAS, ALEXANDRE

(74) Agente/Representante:

ISERN JARA, Jorge

DESCRIPCIÓN

Sistema para acceder a un terminal de PDV, método para descargar y actualizar aplicaciones y método para llevar a cabo una operación electrónica usando un sistema de ese tipo

Campo de la invención

La presente invención se refiere a un sistema para acceder a los dispositivos de PDV y de teclado de entrada de PIN usando un sistema de cliente conectado por medio de una red con al menos un servidor así como a un método para actualizar y descargar una aplicación y llevar a cabo una pluralidad de operaciones usando dicho sistema.

Descripción de la técnica anterior

Muchas empresas en el sector industrial de las operaciones electrónicas (seguros de salud, fidelidad, recarga prepago, tarjetas de regalo, etc.) están constantemente buscando soluciones para mejorar sus servicios y ofrecer otros nuevos, volviéndose más competitivas.

Antes de la aparición de la presente invención, los usuarios, también conocidos como compradores, han sido provistos con terminales tecnológicamente limitados.

Con respecto a la portabilidad, cada terminal de Punto de Venta, denominado en el presente documento terminal de PDV, comprendía solo su sistema operativo y soporte físico propio. Por consiguiente, era necesario reescribir y adaptar las aplicaciones para cada modelo de terminal, en un proceso laborioso y costoso. La compartición de aplicaciones también era insuficiente debido a que los terminales de PDV usados en la red solo eran capaces de enviar y recibir datos de transacción, sin aprovechar otras facilidades de red tales como servidores de aplicaciones. Asimismo, debido a que las aplicaciones estaban codificadas de forma rígida en los terminales de PDV (cliente pesado), era extremadamente difícil, tal vez imposible, usar diferentes aplicaciones en la misma máquina debido a restricciones de memoria y de desarrollo.

De esta forma, cada terminal de PDV solía actualizarse independientemente en un proceso no automatizado. En los últimos tiempos, solían aplicarse unos entornos heterogéneos y sistemas de mejora remota proporcionados por los fabricantes de terminales de PDV en los procesos de mejora. No obstante, tales terminales y sus sistemas de mejora respectivos no eran compatibles entre sí, requiriendo un conocimiento y un personal de mantenimiento específicos para cada sistema y cada fabricante, replicando de ese modo los esfuerzos y costes.

Otro problema existente en la técnica anterior era la necesidad de contar con unas habilidades avanzadas de programación así como con un conocimiento profundo del entorno operativo propio de cada fabricante de terminales en el mercado, con respecto a su especificidad. Esto daba como resultado un enorme coste en la capacitación de personal, desarrollo y mantenimiento de aplicaciones y, en consecuencia, limitaba el ámbito de las aplicaciones y servicios de PDV disponibles. El documento US 2002/046185 A1 (Villart Jean-Marc [FR] y col.) de 18 de abril de 2002 (18-04-2002) se refiere a la realización de transacciones de PDV (Punto de Venta) usando una red de comunicación inalámbrica.

Objetivo de la invención

A la vista del inconveniente indicado anteriormente, la invención proporciona un sistema para acceder a dispositivos de PDV, transformando las redes de PDV en redes de servicio y reduciendo los costes de implementación y gestión de tales redes de PDV.

Otro objeto de la presente invención es la provisión de un sistema que presenta más portabilidad para las aplicaciones en diferentes modelos de terminales de PDV así como permite la compartición de múltiples aplicaciones en el mismo dispositivo de PDV.

Un objeto adicional de la presente invención es la provisión de un sistema para acceder a dispositivos que posibilitan que los usuarios ejecuten una aplicación en un terminal de PDV fabricado por diferentes fabricantes sin necesidad alguna de personalización.

Un aspecto importante de la invención es que el sistema está diseñado para complementar el uso de terminales de PDV convencionales, no para competir con los mismos debido a que muchas empresas, como VeriFone, Ingenico, Lipman, Sagem, Axalto e Intellect entre otros, ya han desarrollado tal terminal de PDV hace mucho tiempo.

Sumario de la invención

De acuerdo con un aspecto de la invención, se proporciona un sistema para soportar aplicaciones Web en un terminal de PDV como se define en las reivindicaciones adjuntas.

Tales sistemas pueden posibilitar que cualquier terminal de PDV ejecute aplicaciones ubicadas en unos servidores de aplicaciones conectados con la red de PDV usando TCP/IP (Internet, Intranets, VPN, etc.) y páginas y secuencias de comandos de WML (Lenguaje de Marcado Inalámbrico), en un proceso similar a Internet.

La tecnología de sistema de cliente puede implementar un modelo de cliente ligero para la arquitectura de cliente/servidor. Al suponer que el cliente tiene un acceso directo al servidor, en donde sigue existiendo toda la lógica de negocio, esta suposición ofrece intrínsecamente dos grandes ventajas para el desarrollo y mantenimiento de soporte lógico. El desarrollo de aplicaciones tiene lugar, en su totalidad, en el lado de servidor en donde se encuentran disponibles muchas herramientas de alta productividad y una aplicación de lado de cliente se puede mejorar automáticamente inmediatamente después de una actualización de lado de servidor.

Norma de WAP, dirigida a los teléfonos celulares de primera generación, se refiere a las penalizaciones de comunicación entre el cliente y el servidor. Por lo tanto, el foro de WAP ha especificado algunos mecanismos de persistencia de información y de validación de datos que reducen drásticamente la cantidad de información intercambiada a través de la red durante una transición de páginas de WML (Lenguaje de Marcado Inalámbrico) sucesivas. Las solicitudes de página de pantalla sin cambios también se pueden evitar con mecanismos de almacenamiento en memoria caché de protocolos Web de HTTP o de WSP, aunque aún se sigue dependiendo de un proceso de actualización controlado por el cliente. Por lo tanto, junto con rutinas de validación de datos de WML Script, la navegación de WML casi se asemeja a un flujo de pantallas de aplicaciones de PDV.

WML es un lenguaje de marcado declarativo dirigido a la creación de pantallas de entrada de datos. Este ofrece facilidades de escritura de entradas básicas y una selección de opciones básica. Tiene un entorno variable volátil que tiene por objeto almacenar información entre pantallas del mismo flujo de transacciones. WML Script es un lenguaje de generación de secuencias de comandos minúsculo diseñado para una escritura de secuencias de comandos de validación de datos simple. Ni WML ni WML Script proporcionan almacenamiento de datos persistente alguno ni asume la presencia de dispositivo periférico particular alguno.

Por analogía, el concepto de sistema de cliente se trajo a los terminales de PDV. En particular, los navegadores de WML han mostrado que son la norma más próxima que ofrecería a un terminal de PDV los beneficios web y de navegación incluso aún careciendo de algunas características clave requeridas por las operaciones electrónicas, tales como impresión de recibos y almacenamiento de historial de transacciones, entre otros.

Los terminales de PDV solían tener periféricos, como un lector de tarjetas magnéticas, impresora térmica, un dispositivo de entrada de teclado de entrada de PIN, lectores de tarjetas inteligentes y otros dispositivos serie que se integran habitualmente a través de una interfaz RS-232 convencional tal como un lector de código de barras, un lector de tarjetas sin contacto Mifare y / o un lector de cheque. Además del acceso de dispositivos periféricos, las aplicaciones de PDV demandan algunas características no disponibles en las definiciones de WML convencional y de WML Script tales como almacén de registro de datos persistente, dar formato de mensaje de ISO-8583, criptografía de información clasificada y soporte de entrada de datos mejorado, autorización de transacciones de EMV y, para terminales de marcado, SDLC y soporte de redes heredadas de X.28.
Además, con el fin de concordar con los mecanismos de mejora de los servidores de transacción actuales se requería una directiva de actualización de almacenamiento en memoria caché controlada por el servidor.

Como se describirá, la tecnología de sistema de cliente permite el uso del terminal de PDV como una plataforma "de bajo coste" para servicios de valor añadido y de compartición de múltiples aplicaciones, convirtiendo una red de PDV en una red de servicio "de hecho".

Breve descripción de los dibujos

La figura 1 ilustra un terminal de PDV conectado con múltiples redes por medio de diferentes protocolos de comunicación.
La figura 2 ilustra la personalización individual de cada terminal de PDV que enseña el estado de la técnica.
La figura 3 ilustra cómo la invención posibilita la comunicación de cualquier terminal de PDV personalizado con un servidor de aplicaciones.
La figura 4 ilustra un diagrama de carga - descarga y un diagrama de operación electrónica que son ejecutados por la presente invención.
La figura 5 ilustra un diagrama con una secuencia de carga - descarga de aplicaciones de sistema de cliente de PDV.
La figura 6 ilustra un diagrama con una secuencia de operaciones Electrónicas de sistema de cliente de PDV.
La figura 7 ilustra las capas de protocolo y la arquitectura del sistema de cliente de PDV.

Descripción detallada de la invención

La presente invención consiste en un sistema para acceder a y conectar unos terminales de PDV 10 personalizados con cualquier red, usando un sistema de cliente 15 específico. Cuando se encuentra en funcionamiento, el sistema de cliente 15 asume el control de todos los recursos de los terminales de PDV 10 actuando como un sistema operativo.

Este posibilita que cualquier terminal de PDV 10 ejecute las aplicaciones 25 ubicadas en unos servidores de aplicaciones 30 conectados con la red de PDV usando TCP/IP (Internet, redes, VPN, etc.) y páginas y secuencias de comandos de WML (Lenguaje de Marcado Inalámbrico) en un proceso similar a Internet. El sistema de cliente 15 también posibilita que la misma aplicación 25 use múltiples protocolos de comunicación 20 cuando se conecta con distintas redes de acceso, posibilitando de ese modo el uso de TCP/IP, redes heredadas (SDLC y X.28, por ejemplo) y otras (GSM, GPRS, CDMA, Ethernet, Wi-Fi, Bluetooth, IR, etc.) como se muestra en la figura 1. Además, el sistema de cliente 15 también tiene una característica avanzada de almacenamiento en memoria caché con el fin de una velocidad creciente de intercambio de datos, permitiendo incluso unas operaciones sin conexión verdaderas. El sistema de cliente 15 sigue reconociendo y controlando la totalidad de los dispositivos de entrada y salida conectados con el PDV, tales como lectores de tarjetas magnéticas, lectores de tarjetas inteligentes, impresoras, teclados de entrada de PIN, lectores de código de barras, lectores de cheques, teclados, lector de tarjetas sin contacto Mifare y así sucesivamente.

El acceso a los servidores de aplicaciones 30 proporcionados por la presente invención posibilita traer a la red de PDV la totalidad de la flexibilidad y funcionalidades disponibles en Internet acerca de lo que concierne a los servicios y aplicaciones. Como resultado, reduce los costes de mantenimiento y añade flexibilidad al proceso de mejora de aplicaciones. Además, debido a la posibilidad de usar distintas redes físicas, el proceso de mejora de aplicaciones no afecta al tiempo promedio de intercambio de datos, que usan redes heredadas basándose en SDLC u otros protocolos asíncronos.

La figura 2 muestra cómo se mejoraban habitualmente los terminales de PDV 10. Antes de la presente invención, era necesario personalizar la aplicación, de tal modo que cada portabilidad se había de llevar a cabo para cada modelo de terminal de PDV. Tal operación solía ejecutarse individualmente, requiriendo un conocimiento y un personal de mantenimiento específicos para cada sistema de terminal de PDV.

Como se puede ver de la figura 3, el entorno operativo del sistema de cliente permite compartir una cantidad arbitraria de las aplicaciones 25 en solo un terminal de PDV 10, usando los servidores de aplicaciones 30 conectados a través de la red, posibilitando también mejorar tales aplicaciones de forma remota y automática y en tiempo real.

Realización preferida de la invención

Con el fin de solucionar los problemas descritos anteriormente, la presente invención define extensiones de sistema de cliente web que adaptan un navegador WAP convencional para usar los periféricos de PDV y lo extienden para llevar a cabo operaciones electrónicas.

La solución soluciona un problema existente con una tecnología nueva, aunando la fiabilidad y velocidad de ejecución de operaciones de ISO 8583 sobre protocolos heredados, tales como SDLC y X.28, con la capacidad de mantenimiento y velocidad de desarrollo de aplicaciones web.

Cuando se encuentra en funcionamiento, este sistema de cliente 15 asume el control de la totalidad de los recursos de terminal de PDV actuando como el propio sistema operativo. Este posibilita que cualquier terminal de PDV 10 ejecute unas aplicaciones web 25 descargadas desde los servidores de aplicaciones 30 conectados con la red de PDV usando HTTP sobre TCP/IP, opcionalmente asegurados por protocolo de SSL, en un proceso similar a Internet. El acceso de TCP/IP al servidor de aplicaciones 30 se puede realizar usando una conexión de marcado simple, o una red de banda ancha como GPRS, CDMA 1X, Ethernet II y WiFi (IEEE 802.11).

Las aplicaciones web de PDV, que consisten en páginas, secuencias de comandos y otros archivos de WML, llevan a cabo operaciones electrónicas de ISO-8583 con el servidor de transacción 35, pretendiendo ser una aplicación en C convencional. Tales operaciones fluyen tanto a través de conexiones de marcado de SDLC/X.28 como a través de redes de banda ancha de TCP/IP.

Este esquema se muestra en la figura 4. Observando tal figura, es posible detectar las dos situaciones distintas en el ciclo de vida del sistema de cliente de PDV 15. La primera es cuando este conecta con el servidor de aplicaciones 30 para borrar o actualizar la aplicación web y la segunda cuando esta aplicación 25 conecta con el servidor de transacción 35 para llevar a cabo una transacción electrónica.

La descarga de aplicación se puede detallar dentro de las siguientes etapas, de acuerdo con la figura 5:

1. El técnico escribe los ajustes de conexión de servidores de aplicaciones
2. El sistema de cliente de PDV 15 conecta con un Servidor de Acceso Remoto (RAS) 45 o se acopla a una red de banda ancha.
3. El sistema de cliente de PDV 15 recibe una dirección de Protocolo de Internet (IP).
4. El sistema de cliente de PDV 15 establece una sesión de Protocolo de Control de Transferencia (TCP) con el servidor de aplicaciones 30 y envía una solicitud de descarga/actualización de aplicación, habitualmente con una orden de HTTP (o HTTPs) de POST.
5. El servidor de aplicaciones 30 procesa la solicitud.

6. El servidor de aplicaciones 30 entra en contacto con el servidor de transacción 35, pretendiendo ser el terminal de PDV 10, con un mensaje de ISO-8583 sobre una conexión de TCP/IP.
7. El servidor de transacción 35 responde al servidor de aplicaciones 30 con los ajustes de configuración del sistema de cliente de PDV.
8. El servidor de aplicaciones 30 procesa los ajustes de aplicación y construye una versión personalizada de la Aplicación Web.
9. El servidor de aplicaciones 30 envía al sistema de cliente de PDV 15 las páginas, secuencias de comandos y archivos de datos de aplicación web.
10. El sistema de cliente de PDV 15 procesa los archivos recibidos.
11. El sistema de cliente de PDV 15 emite un recibo de la compleción de la descarga/actualización de aplicación al técnico
12. El sistema de cliente de PDV 15 inicia automáticamente la Aplicación Web.

Las etapas 6 y 7 anteriores son opcionales en la solución. En otro caso de uso, el servidor de aplicaciones puede simplemente descargar la totalidad de la aplicación sin la necesidad del proceso de personalización de aplicaciones o de acceder al servidor de transacción.

Además, las operaciones electrónicas llevan a cabo un proceso más simple que se puede detallar con las siguientes etapas, como se muestra en la figura 6:

1. El usuario de PDV escribe los datos de transacción;
2. El sistema de cliente de PDV 15 conecta con un Servidor de Acceso Remoto (RAS) 45 o se acopla a una red de banda ancha.
3. El sistema de cliente de PDV 15 recibe un acceso a red.
4. El sistema de cliente de PDV construye un mensaje de ISO-8583.
5. El sistema de cliente de PDV envía el mensaje de ISO-8583 al servidor de transacción 35.
6. El servidor de transacción 35 responde al sistema de cliente de PDV con un mensaje de ISO-8583.
7. El sistema de cliente de PDV 15 procesa el mensaje recibido.
8. El sistema de cliente de PDV 15 emite un recibo de transacción.

1. Arquitectura y capas de protocolo

Con el fin de definir mejor la invención, la arquitectura de sistema de cliente se detalla posteriormente y se divide en capas específicas como se ilustra en la figura 7.

1.1 - Protocolo de comunicación

Con respecto a los protocolos de comunicación, el sistema de cliente es capaz de ejecutar operaciones a través de uno de los siguientes protocolos: TCP/IP/PPP, SDLC, X.28, GSM, GPRS, CDMA, Ethernet, Wi-Fi, Bluetooth, IR, etc. La aplicación se encuentra a cargo de seleccionar qué protocolo se habrá de usar durante una operación.

1.2 - Protocolo de seguridad

La aplicación de cliente ligero intercambia datos con el servidor web a través del protocolo HTTPS, que se basa en la norma SSL 3.0. La capa de SSL se ejecuta sobre el protocolo TCP y proporciona soporte para que un cliente de HTTP establezca una conexión segura con cualquier servidor conforme.

El sistema de cliente soporta los siguientes conjuntos de cifrado: RSA + RC4 + MD5 y RSA + 3DES + MD5. Ambos de los mismos soportan la autenticación de certificado de servidor y la autenticación de certificado de cliente en el formato de PEM. El conjunto de cifrado con 3DES también soporta la autenticación de certificado de cliente en el formato de PKCS12.

1.3 - Protocolos de transacción

El sistema de cliente soporta operaciones de HTTP, de ISO-8583 y de XML. Las operaciones de HTTP se encuentran disponibles a través de una etiqueta de WML <go> (ir) y se soportan las órdenes tanto GET (obtener) como POST (publicar). Las operaciones de ISO-8583 y de XML son ofrecidas por las extensiones de WML Script. Todos los protocolos pueden coexistir en la misma aplicación y pueden conectar con distintos servidores de acuerdo con los ajustes de aplicación.

1.4 - Motor de tiempo de ejecución

Después de solicitar la interfaz de usuario, el sistema de cliente sigue aguardando que la interacción de usuario desencadene eventos contra el motor de tiempo de ejecución. Estos eventos se manejan de acuerdo con las definiciones de página actuales y pueden desencadenar un cierto procesamiento de WML Script local o invocar otra página que cargar. Cuando se carga una nueva página de WML, es analizada por el analizador de XML personalizado

para WML y se genera un árbol de sintaxis de DOM. El árbol se recorre entonces y las estructuras internas de tiempo de ejecución se establecen para responder apropiadamente a los siguientes eventos generados por el usuario. Tanto las páginas como las secuencias de comandos se pueden hallar en la memoria caché del sistema de cliente o se pueden solicitar a un servidor de HTTP(S).

1.5 - Gestor Multi-Aplicación (MAM)

El Gestor Multi-Aplicación del sistema de cliente permite que más de una aplicación resida y se ejecute en el mismo PDV. El sistema de cliente aísla las aplicaciones frente a otras aplicaciones que usan entornos de ejecución virtual diferentes. Los recursos (WML, secuencias de comandos, archivos de datos y variables) solo pueden ser creados, leídos o actualizados por la aplicación del propietario.

Las aplicaciones se pueden configurar en el terminal desde un PC, usando una comunicación serie, o se pueden coger de un servidor, a través de TCP/IP. La configuración básica se almacena en una tabla de aplicaciones (TA) en el PDV que almacena información acerca del conjunto actual de aplicaciones.

Funcionalidades principales de MAM:

- Carga Automática de Aplicaciones. Actualización o Borrado- La aplicación maestra, que tiene derechos privilegiados y habitualmente pertenece al propietario de PDV, puede solicitar una actualización de tabla de aplicaciones procedente del servidor, con el fin de añadir, borrar o actualizar aplicaciones en ese terminal. La misma operación se puede realizar para un técnico a través del menú de configuración. Véase la funcionalidad de Configuración Multi-Aplicación descrita posteriormente.
- Comunicación entre aplicaciones - el sistema de cliente permite la comunicación entre aplicaciones. La aplicación A puede iniciar la aplicación B, así como enviar parámetros de una aplicación a otra.
- Conmutación entre aplicaciones - El usuario puede conmutar de una aplicación a otra al presionar una tecla programable que llama al menú principal del sistema de cliente. Desde este, los usuarios pueden seleccionar otras aplicaciones.
- Validación de Tabla de Aplicaciones - Se usa una Clave, para verificar si la tabla de aplicaciones (TA) válida actual pertenece al PDV actual. Si la clave no coincide, es necesaria una nueva TA.
- Configuración Multi-Aplicación - Una interfaz para el operador de PDV. A través de la misma, es posible ejecutar funciones de configuración y de operación básicas, incluyendo borrar, añadir o actualizar aplicaciones.

1.6 - Biblioteca convencional de Protocolo de EMV

Con el fin de abstraer las capacidades de terminal de distintos fabricantes con respecto al protocolo de EMV, el sistema de cliente asume la existencia de una biblioteca de EMV. Esta biblioteca es, en realidad, una biblioteca de interfaces de teclado de entrada de PIN que encapsula la totalidad de las interacciones entre el sistema de cliente y el núcleo de EMV de PDV.

El sistema de cliente ofrece la Biblioteca de EMV a la aplicación de cliente ligero como una biblioteca de extensión de WML Script, denominada PINPadlib.

2. Extensiones de sistema de cliente

Los lenguajes tanto WML como WML Script se diseñaron originalmente para crear Sitios WEB que se pueden visualizar en navegadores WAP, que habitualmente se encuentran disponibles en los teléfonos celulares. Con el fin de satisfacer los requisitos de una aplicación de PDV convencional, la invención define algunas extensiones para los Lenguajes WML y WML Script.

Este conjunto de etiquetas, atributos y funciones de secuencia de comandos de WML permite, entre otras características, que las aplicaciones de PDV en ejecución sobre el sistema de cliente: 1) accedan a dispositivos habitualmente disponibles en terminales de PDV tales como lectores de tarjetas, impresora, lectores de código de barras, teclado de entrada de PIN, lectores de tarjetas sin contacto Mifare, lectores de cheques, etc...; 2) usen múltiples protocolos de comunicación cuando se conectan con distintas redes de acceso (SDLC, X.28, TCP/IP, PPP, GPRS, CDMA, WNB, etc...); 3) almacén de registro de datos persistente; 4) accedan al núcleo de EMV presenta en el sistema operativo de PDV; 5) den formato de mensaje de ISO-8583; 6) criptografía de información clasificada; etc...

La lista completa de extensiones así como su funcionalidad detallada se describe posteriormente.

2.1 - Extensiones de Etiqueta para el Lenguaje WML

Esta sección presenta las nuevas etiquetas, y sus atributos, proporcionadas a un navegador WAP con el fin de convertirlo en un sistema de cliente de PDV.

Etiqueta	Descripción	Atributos interpretados
pitido	Empieza a emitir un pitido en la frecuencia especificada hasta que se presiona una tecla.	tiempos de frecuencia - número de veces que ha de apagarse el pitido. sinc (verdadero | falso) - reproduce el pitido en tiempo de exhibición (verdadero) o en tiempo de ejecución (falso) cuando tiene lugar una redirección
formulario	Permite la entrada de datos en páginas que contienen las etiquetas <entrada>, posicionando automáticamente el cursor sobre el siguiente campo cuando se presiona la clave <intro>. Si el número de líneas entre los dos campos es mayor que el número de líneas en pantalla, el cursor se situará en la última línea en pantalla. Su función es similar a la <p> de la etiqueta, definiéndose en el mismo ámbito y aceptando los mismos atributos y contenido.	Ninguno
Teclado de entrada de PIN	Posibilita usar el teclado de entrada de PIN en páginas de WML. Se puede definir en el mismo ámbito que la <entrada> de la etiqueta. Exactamente igual que la <entrada> de la etiqueta, no puede tener contenido alguno. La explicación de los atributos está en pantalla	nombre - Nombre de la variable que recibe la entrada. titulo - Texto presentado en la pantalla del teclado de entrada de PIN longitudmax - longitud máxima de la cadena de entrada tipo (texto | oculto | contraseña) - texto y contraseña son similares al homónimo de entrada de etiqueta. Oculto, no repite lo escrito; se usa para confirmar los valores clavemaestra - Clave Codificada que acompaña al teclado de entrada de PIN (Proveedor) clavetrabajo - Clave Codificada proporcionada por el usuario (Cliente) cuenta - Número de Tarjeta (usada más frecuentemente para VISA)
Pre	Funciona de una forma similar a la de la etiqueta homónima de HTML. Muestra un texto con sus contenidos de la forma en la que se escribió. Tiene una función similar a la de la etiqueta <p>, definiéndose en el mismo ámbito y aceptando los mismos atributos y contenidos.	Ninguno
informe	Posibilita la impresión de informes. Los contenidos de la etiqueta se imprimen en lugar de mostrarse en la pantalla. Su función es similar a la <p> de la etiqueta, definiéndose en el mismo ámbito y aceptando los mismos atributos y contenidos	ALspiepagina - Número de caracteres de avance de línea en el final del informe altura (1 | 2) - altura de fuente anchura (1 | 2) - anchura de fuente
establecerEnv	Los atributos y el comportamiento son similares a <establecervar>. La diferencia es que esta establece variables en el entorno persistente de los navegadores	nombre - Nombre de la variable que recibe el valor valor - Valor que se va a asignar
Establecercampo	Los atributos y el comportamiento son similares a <campopublicación>. La diferencia es que esta establece un campo en una estructura Flexi-record que se enviará a otra aplicación a través del href de esquema de "vmac://" de la etiqueta <go>	nombre - Id de Flexi-record del campo que se va a establecer. valor - Valor que se va a asignar tipo - Tipo del campo (cadena | int | largo)

2.2 -Extensiones de Atributos para el Lenguaje WML

Esta sección presenta los nuevos atributos, valores y el nuevo comportamiento de los atributos convencionales, proporcionados a un navegador WAP con el fin de convertirlo en un sistema de cliente de PDV.

Etiqueta	Atributos	Descripción
	contextonuevo = (verdadero | falso | vars) autoimpresión = (verdadero | falso) marcado = (verdadero | falso)	Nuevo valor: el valor de "vars" inicializa únicamente las variables. Nuevo atributo: Imprime automáticamente los contenidos de tarjeta actuales. Nuevo atributo: Conecta automáticamente cuando se muestra la tarjeta.
Tarjeta	colgar = (verdadero | falso) persistido = (verdadero | falso | último | ninguno) al actualizar = hRef en evento de usuario = hRef al validar = hRef	Nuevo atributo: Desconecta automáticamente cuando se muestra la tarjeta. Nuevo atributo: Permite que se haga persistir una plataforma: verdadero - Creará una copia del archivo actual usando el nombre "persistido.wml". El archivo más antiguo que se hace persistir se renombrará a "último_persistido.wml" falso - No se emprenderá acción alguna. último - Renombrará "último_persistido.wml" a "persistido.wml". ninguno - Eliminará el archivo "persistido.wml" de tal modo que no se use de forma no deseable. Nuevo atributo: url al que ir cuando se desencadena el evento de actualización intrínseco Nuevo atributo: url al que ir cuando se desencadena cualquier evento de usuario (pasada de tarjeta, pulsación de tecla, lectura de código de barras). Habitualmente útil para las pantallas inactivas. Nuevo atributo: url al que ir después de cada confirmación de entrada en la tarjeta actual.
Go	Srclocal = hRef src = hRef delimitador usuario contraseña hRef = hRef	Nuevo atributo: Ruta al archivo local para la operación de obtención y de colocación. Nuevo atributo: Ruta al archivo remoto para la operación de obtención y de colocación. Nuevo atributo: Delimitador para la lista de campos de publicación. Nuevo atributo: Inicio de sesión de usuario para servidores que requieren una autorización de acceso. Nuevo atributo: Contraseña de usuario para servidores que requieren una autorización de acceso. Nuevo comportamiento: Enviar eventos de MAM de SO a otras aplicaciones se establece con unos URL como "osmam://logAppName: EventNo". Los argumentos se pueden pasar a través de una estructura de registro de TLV con la etiqueta <establecercampo> o a través de un formato de HTTP con sintaxis de obtención "?" o con etiqueta <campopublicación>.
Entrada	alinear = (izquierda | derecha) Dispositivo = (teclado numérico | teclado | código de barras | magnético) tipo = (texto | contraseña | divisa) alarma = (verdadero | falso) relleno formatocompleto = (verdadero | falso)	Nuevo atributo: El texto se alinea dentro del espacio de entrada. Nuevo atributo: Dispositivo de entrada de datos seleccionado. Son valores posibles: magnético - Lector de tarjetas magnéticas; código de barras - Lector de código de barras; teclado - teclado externo; teclado numérico - teclado convencional. Nuevo valor: El valor de divisa fuerza que el valor de entrada tenga céntimos (por ejemplo "0,00") Nuevo atributo: Si es verdadero, hace sonar un pitido cuando se ha alcanzado el tamaño de entrada. Nuevo atributo: Carácter con el que llenar el área de entrada con el fin de dejar que el usuario esté al tanto del tamaño de la entrada. Su presencia fuerza que se muestren las máscaras de entrada antes de la escritura y se suprime el símbolo de intercalación. Nuevo atributo: Requiere que se rellene la totalidad del campo de entrada.

(continuación)

Etiqueta	Atributos	Descripción
	literales = (verdadero | falso)	Nuevo atributo: Si es verdadero, suprime del valor de la variable los caracteres de formato.
Enevento	tipo =(... | al actualizar | en evento de usuario | al validar)	Nuevo valor: Nuevos eventos "intrínsecos" que se desencadenan de la misma forma que sus homólogos de atributo de <tarjeta>.
Temporizador	anular = (verdadero | falso)	Nuevo atributo: Indica si el temporizador se puede deshabilitar cuando se presiona una tecla

2.3 - Extensiones de Funciones de WML Script para las bibliotecas convencionales de WML Script

Esta sección presenta extensiones para las bibliotecas convencionales de WML Script, proporcionadas a un navegador WAP con el fin de convertirlo en un sistema de cliente de PDV. Las definiciones de tales extensiones de Funciones de secuencia de comandos se explican posteriormente de la tabla.

Biblioteca convencional de WML Script	Funciones extendidas de WML Script
FLOTANTE	aFlotante (valor)
CADENA	establecerPosElementos (Cadena, separador, nElementos)
	elementoEnPos (posElemento)
	formatoDivisa (cadenaDivisa)
	esNumerico (cadena)
	aMayusculas (cadena)
	aMinusculas (cadena)
	rellenarDerecha (cadena, relleno, lon)
	rellenarIzquierda (cadena, relleno, lon) cargarArchivo (url, nombreArchivo)
URL	cargarEnCache (url)
NavegadorWML	establecerEnv (Nombre, Valor)
	irExt (url, método, blsAtras)
	pitido (Frec, nVeces)
	colgar ( )
	marcado ( )
	Persistir ( )
	establecerCampoPost (nombre, valor)
	limpiarCacheConEtiqueta (etiqueta)
	obtenerPrimerNombreVar ( )
	obtenerNombreVarSiguiente ( )
	obtenerNombreVarPrev ( )
	pinApin ( )
	marcar (CadenaADonde)
	subida ( )
	añadirEnv (nombreVar, incremento)

(continuación)

Biblioteca convencional de WML Script	Funciones extendidas de WML Script
	adjuntarVar (nombreVar, valor)
	establecerEnvDeVars (nombreVar, listaVar)
	establecerVarsDeEnv (nombreVar)
	obtenerVarDeEnv (nombreVarEnv, nombreVar)
	borrarContexto (nombreAplic)
	estaConectado ( )
	instalarAplic (archivoDatosAplic)
	establecerVarsDeVars (nombreVar, listaVar)
	establecerVarsDeVars (nombreVar)
	obtenerVarDeVar (nombreVarwml, nombreVar)
	obtenerVarDeCad (wmlsCadenaOVar, nombreVar)
DIÁLOGOS	mostrar (mensaje, entradaPorDefecto)
	mostrarEstado (mensaje, entradaPorDefecto)

Funciones específicas creadas para el sistema de cliente de PDV:

aFlotante (valor) - Esta función devuelve la representación en coma flotante de un valor dado. Esta función ejecuta exactamente las mismas conversiones de tipos definidas para el lenguaje WML Script. Un *valor* no válido devuelve la cadena "no válido". Requerido para forzar una conversión de tipos cuando se usan variables con tipo.

establecerPosElementos (Cadena, separador, nElementos) - Esta función almacena *nElementos* de *Cadena* separados por un *separador.* En caso de que tenga éxito, devuelve un entero, que contiene la posición del último carácter del último elemento almacenado. Habitualmente, se usa antes de una llamada a un Cadena.elementoEnPos (posElemento) mostrado posteriormente.

elementoEnPos (posElemento) - Esta función devuelve el elemento de una cadena, que está en la posición *posElemento.* La cadena se ha de haber organizado ya con la aplicación de la función *establecerPosElementos (Cadena, separador, nElementos).* Si la cadena no esté organizada ya por la función *establecerPosElementos,* se usará la que se organizó previamente. En caso de error, devuelve *no válido.*

formatoDivisa (cadenaDivisa) - Esta función da formato a una Cadena en notación de divisa, separando los céntimos con una marca de coma o de punto, así como los millares y así sucesivamente. La función devuelve la cadena con formato o *no válido* en caso de que falle.

esNumerico (cadena) - Esta función devuelve verdadero si el argumento se puede convertir con éxito a un valor numérico, de lo contrario, devuelve falso.

aMayusculas (cadena) - Esta función devuelve un argumento de tipo cadena convertido a mayúsculas.

aMinusculas (cadena) - Esta función devuelve un argumento de tipo cadena convertido a minúsculas.

rellenarDerecha (cadena, relleno, lon) - Esta función devuelve un argumento de tipo cadena rellenado por la derecha con caracteres de relleno hasta la longitud *lon.*

rellenarIzquierda (cadena, relleno, lon) - Esta función devuelve un argumento de tipo cadena rellenado por la izquierda con caracteres de relleno hasta la longitud *lon.*

cargarArchivo (url, nombreArchivo) - Esta función recupera el contenido indicado por el *url absoluto y se guarda en un archivo denominado nombreArchivo.* Retira ya los encabezados de HTTP y devuelve el código de resultado de HTTP (200 quiere decir Correcto).

cargarEnCache (url) - Esta función recupera el contenido indicado por el *url* absoluto y lo almacena en la memoria caché del sistema de cliente para su uso futuro. Devuelve el código de resultado de HTTP (200 quiere decir Correcto).

establecerEnv (Nombre, Valor) - Esta es una extensión que recibe como parámetros el nombre de la variable de entorno de PDV y el valor que se va a atribuir a esta variable. Devuelve *verdadero* si tiene éxito en cambiar el *valor* de la variable o falso, si no es así. En caso de fallo (parámetros no válidos, por ejemplo), devuelve *no válido.*

irExt (url, método, blsAtras) - Esta extensión de la función NavegadorWML.go ( ) permite el reenvío o la devolución a un URL definido. La devolución a una página de WML requerirá la realización del evento "al introducir hacia atrás", en el caso de que se haya previsto. Recibe como parámetros el *url,* el *método* (POST o GET) y un valor lógico *blsAtras,* que indica la devolución al URL. Devuelve la cadena vacía en caso de que tenga éxito, de lo contrario, devuelve *no válido.*

pitido (Frec, nVeces) - Esta función emite un sonido en el dispositivo de salida en la frecuencia especificada por

Frec, y el número de veces especificado por *nVeces.* Devuelve la cadena vacía en caso de que tenga éxito, de lo contrario, devuelve *no válido.* colgar ( ) - Esta función desconecta el PDV de la línea de teléfono. Devuelve la cadena vacía.

marcado ( ) - Esta función inicia la conexión asíncrona de PDV con la línea de teléfono. Devuelve la cadena vacía.

persistir ( ) - Esta función genera una copia de la Secuencia de comandos actual con el nombre "i: persistido.wmlsc". En el caso de que ya exista este archivo, se renombrará a "i: último_persistido.wmlsc". Devuelve el número de bytes registrados en el archivo "i: persistido.wmlsc". Si la secuencia de comandos actual ya se ha eliminado a sí misma al borrar la memoria caché, entonces el resultado de la persistencia de llamadas está sin definir y puede dar como resultado un error de tiempo de ejecución.

establecerCampoPost (nombre, valor) - Esta función establece campos de publicación que se supone que se envían con la siguiente llamada a NavegadorWML.irExt (href, "POST", verdadero). Devuelve la cadena vacía si tiene éxito, de lo contrario devuelve no válido.

limpiarCacheConEtiqueta (etiqueta) - Esta función elimina de la memoria caché cada archivo que presenta un encabezado de HTTP "EtiquetaVW:" que comienza con *etiqueta.*

obtenerPrimerNombreVar ( ) - Esta función devuelve el nombre de la primera variable de entorno. Junto con obtenerNombreVarSiguiente ( ) y obtenerNombreVarPrev ( ), ofrece un iterador al entorno persistente.

obtenerNombreVarSiguiente ( ) - Esta función devuelve el nombre de la primera variable de entorno. Junto con obtenerPrimerNombreVar ( ) y obtenerNombreVarPrev ( ), ofrece un iterador al entorno persistente.

obtenerNombreVarPrev ( ) - Esta función devuelve el nombre de la primera variable de entorno. Junto con obtenerPrimerNombreVar ( ) y obtenerNombreVarSiguiente ( ), ofrece un iterador al entorno persistente.

pinApin ( ) - Esta función inicia un clon de sistema de cliente "inmediatamente consecutivo" controlado por el sistema operativo.

marcar (CadenaADonde) - Esta función establece los parámetros de marcado a partir de una configuración en un almacén de registros. Devuelve el estado de conexión.

obtenerEstadoCon ( ) - Esta función devuelve un entero cuyo valor es el estado de conexión.

subida ( ) - Esta función inicia la subida de clonación de aplicaciones "inmediatamente consecutiva".

añadirEnv (nombreVar, incremento) - Esta función recibe como parámetros el nombre de la variable de entorno de PDV y un valor entero que se va a añadir al valor actual de esta variable. Si la variable no existe, se crea con el valor recibido como parámetro. Devuelve verdadero si tiene éxito en cambiar el valor de la variable o falso, si no es así. En caso de fallo (parámetros no válidos, por ejemplo), devuelve no válido.

adjuntarVar (nombreVar, valor) - Esta función recibe como parámetros el nombre de la variable de entorno de PDV y un valor de cadena que se va a adjuntar al valor actual de esta variable. Si la variable no existe, se crea con el valor recibido como parámetro. Devuelve verdadero si tiene éxito en cambiar el valor de la variable o falso, si no es así. En caso de fallo (parámetros no válidos, por ejemplo), devuelve no válido.

establecerEnvDeVars (nombreVar, listaVar) - Esta función establece la variable de entorno nombreVar con las variables definidas en la lista de nombres de var separados por punto y coma listaVar. Devuelve la lon de la cadena establecida a nombreVar, que se debería haber guardado para su uso futuro con las funciones establecerVarsDeEnv ( ) u obtenerVarDeEnv ( ).

establecerVarsDeEnv (nombreVar) - Esta función establece el contexto de WML con las variables obtenidas del valor de la variable de entorno nombreVar. Devuelve el número de variables establecidas, que se debería haber guardado con la función establecerEnvDeVars ( ).

obtenerVarDeEnv (nombreVarEnv, nombreVar) - Esta función devuelve el valor de la variable denominada nombreVar a partir del valor de la variable de entorno nombreVarEnv. Si la variable no está definida por el valor, devuelve la cadena vacía.

borrarContexto (nombreAplic) - Esta función borra completamente los archivos de aplicación denominados nombreAplic. Esta solo puede ser llamada por la aplicación maestra de PDV.

estaConectado ( ) - Esta función sustituye a obtenerEstadoCon ( ) devuelve un valor Booleano que indica si la conexión está activa.

instalarAplic (archivoDatosAplic) - Esta función instala una aplicación desde un archivo de datos denominado archivoDatosAplic y devuelve un entero que indica el código de error. Si esta función se ejecuta bien, el terminal se reiniciará y el código devuelto no se someterá a prueba.

establecerVarsDeVars (nombreVar, listaVar) - Esta función establece la variable de WML nombreVar con las variables definidas en la lista de nombres de var separados por punto y coma listaVar. Devuelve la lon de la cadena establecida a nombreVar, que se debería haber guardado para su uso futuro con las funciones establecerVarsDeVars ( ) u obtenerVarDeVar ( ).

establecerVarsDeVars (nombreVar) - Esta función establece el contexto de WML con las variables obtenidas del valor de la variable de WML nombreVar. Devuelve el número de variables establecidas, que se debería haber guardado con la función establecerVarsDeVars ( ).

obtenerVarDeVar (nombreVarwml, nombreVar) - Esta función devuelve el valor de la variable denominada nombreVar a partir del valor de la variable de WML nombreVarwml. Si la variable no está definida por el valor, devuelve la cadena vacía.

obtenerVarDeCad (wmlsCadenaOVar, nombreVar) - Esta función devuelve el valor de la variable denominada nombreVar a partir del valor de una variable de WML Script o cadena wmlsCadenaOVar. Si la variable no está definida por el valor, devuelve la cadena vacía.

mostrar (mensaje, entradaPorDefecto) - Esta función muestra un mensaje en la pantalla de PDV y continúa la ejecución con independencia de la solicitud del usuario. En caso de que tenga éxito, devuelve la cadena vacía, de

lo contrario, devuelve *no válido.*
mostrarEstado (mensaje, entradaPorDefecto) - Esta función muestra un mensaje en la línea de Estado en la pantalla de PDV. En caso de que tenga éxito, devuelve la cadena vacía, de lo contrario, devuelve *no válido.*

2.4 - Bibliotecas y funciones de WML Script - Extensiones para las bibliotecas convencionales de WML Script

Esta sección presenta las bibliotecas de extensión de WML Script proporcionadas a un navegador WAP con el fin de convertirlo en un sistema de cliente de PDV.

Biblioteca extendida	Funciones extendidas
REGISTROALMACENES	abrirAlmacen (Nombre, CrearSiNecesario)
	CerrarAlmacen (IdAlmacen)
	borrarAlmacen (Nombre)
	obtenerNumRegistros (IdAlmacen)
	obtenerTamaño (IdAlmacen)
	añadirRegistro (IdAlmacen, Registro)
	borrarRegistro (IdAlmacen, IdRegistro)
	obtenerRegistro (IdAlmacen, IdRegistro)
	obtenerIdRegistroSiguiente (IdAlmacen, IdRegistro)
	obtenerTamañoRegistro (IdAlmacen, IdRegistro)
	establecerRegistro (IdAlmacen, IdRegistro, ValorRegistro)
	crearBaseDatos (NombreBase, NombreOrigen)
	hallarRegistro (BD, Clave, Pos, Separador, Ordenado)
	establecerClaveOrden (BD, Pos, Separador)
	hallarPrimerRegistro (IdAlmacen)
	hallarRegistroSiguiente (IdAlmacen)
	hallarRegistroRepetido (BD, Clave, Pos, Separador, Ordenado)
	defragAlmacen (NombreAlmacen)
	adjuntarultimoRegistro (IdAlmacen, cadena)
	añadirRegistroDeVars (IdAlmacen, listaVar)
	establecerVarsDeRegistro (IdAlmacen, IdRegistro)
	obtenerVarDeRegistro (IdAlmacen, IdRegistro, NombreVar)
	establecerVista (BD, Clave, Pos, Separador)
	borrarVista (BD)
	establecerVistaFiltro (BD, Clave, Pos, Separador)
CONSOLA	imprimir (cadena)
	imprimirLn (cadena)
CRIPTO	cifrar (Mensaje, ClaveSesion)
	descifrar (Mensaje, ClaveSesion)

(continuación)

Biblioteca extendida	Funciones extendidas
	cifrarADES (Mensaje, ClaveSesion)
	descifrarADES (Mensaje, ClaveSesion)
	oexclusivo (cadena1, cadena2)
	cifrarADESHexa (Mensaje, ClaveSesion)
	descifrarADESHexa (Mensaje, ClaveSesion)
	cifrarHexa (Mensaje, ClaveSesion)
	descifrarHexa (Mensaje, ClaveSesion)
Biblioteca extendida	Funciones extendidas
IMPRESORA	abrir ( )
	establecerModoAnchura (modo)
	establecerModoAltura (modo)
	imprimir (cadena)
	imprimirLn (cadena)
	cerrar ( )
	imprimirLGO (NombreArchivoLOGO)
TARJETA	comprobarIntervalos (PANactual, sufijoEtiquetaPrivada)
	comprobarIntervalosenAlmacen (PANactual, NombreAlmacen)
SISTEMA	tiempoActualSegs ( )
	horafechaASegundos (horafecha)
	segundosAhorafecha (segundos)
	fechaEsValida (horafecha)
	horafecha ( )
	ticsActuales ( )
ISO	carAHex (Cadena)
	hexACar (Cadena)
	leerEmpaquetador (Cadena)
	transacMensaje (Cadena anfitrión, Int puerto, Cadena lista de campos, Cadena canal, Cadena encabezado, Cadena finalizador, Cadena camposObligatorios)
	transacDeRegistro (Cadena anfitrión, Int puerto, Int idAlmacen, Int idRegistro, Cadena canal, Cadena encabezado, Cadena finalizador, Cadena camposObligatorios)
	transacDeRegistroAVar (Cadena anfitrión, Int puerto, Int idAlmacen, Int idRegistro, Cadena canal, Cadena encabezado, Cadena finalizador, Cadena camposObligatorios)
	hexAInt (Cadena hexa, BigEndian Booleano)
	intAHex (Int i, BigEndian Booleano)

Bibliotecas y funciones específicas creadas para el sistema de cliente de PDV:

2.4.1 - Biblioteca extendida de REGISTROALMACENES

El objeto de esta biblioteca es permitir la persistencia de información, almacenando los datos en archivos indexados.

Lista de funciones:

abrirAlmacen (Nombre, CrearSiNecesario) - Esta función recibe como parámetros el nombre del archivo que se van a crear y una variable booleana CrearSiNecesario, que apunta si el archivo se debería crear en el caso de que no exista. Al mismo tiempo de la creación del archivo, también se crea un archivo de índice respectivo. Devuelve un entero con el índice (id) del archivo abierto o no válido, en caso de fallo al abrir o crear el archivo.

cerrarAlmacen (IdAlmacen) - Esta función recibe como parámetro el índice (IdAlmacen) del archivo abierto, cerrando los archivos de datos y de índice. Devuelve un entero con el índice (IdAlmacen) del archivo cerrado. Si no tiene éxito en el cierre del archivo o si IdAlmacen no existe, devuelve no válido.

borrarAlmacen (Nombre) - Esta función recibe como parámetro una cadena, Nombre, que contiene el nombre del archivo abierto y borra este archivo de datos, así como su archivo de índice. Devuelve un entero mayor que o igual a cero si tiene éxito en borrar los archivos. Si no tiene éxito, devuelve *no válido.*

obtenerNumRegistros (IdAlmacen) - Esta función recibe como parámetro un entero, IdAlmacen, que contiene el índice del archivo abierto y obtiene el número de registros registrados. Devuelve un entero mayor que o igual a cero si tiene éxito en obtener el número de registros. De lo contrario, devuelve no válido.

obtenerTamaño (IdAlmacen) - Esta función recibe como parámetro un entero, IdAlmacen, que contiene el índice del archivo abierto y obtiene su tamaño en bytes. Si la operación tiene éxito devuelve un entero con el tamaño de archivo (número total de bytes), de lo contrario devuelve no válido.

añadirRegistro (IdAlmacen, Registro) - Esta función recibe como parámetro un entero, IdAlmacen, que contiene el índice del archivo abierto y una cadena, Registro, que contiene los datos que se van a incluir en el archivo. Si tiene éxito en la anexión, devuelve un entero con el índice del registro adjunto en el archivo de datos, de lo contrario, devuelve no válido.

borrarRegistro (IdAlmacen, IdRegistro) - Esta función recibe dos enteros como parámetros: IdAlmacen, que contiene el índice del archivo abierto e IdRegistro, que contiene el índice del registro que se va a excluir del archivo. Si tiene éxito en el borrado, devuelve un entero con el índice de ese registro borrado, de lo contrario, devuelve no válido.

obtenerRegistro (IdAlmacen, IdRegistro) - Esta función recibe dos enteros como parámetros: IdAlmacen, que contiene el índice del archivo abierto e IdRegistro, que contiene el índice del registro que se va a recuperar. Si tiene éxito en la operación de lectura, devuelve un entero con el índice de ese registro recuperado, de lo contrario, devuelve no válido.

obtenerIdRegistroSiguiente (IdAlmacen, IdRegistro) - Esta función recibe dos enteros como parámetros: IdAlmacen, que contiene el índice del archivo abierto e IdRegistro, que contiene el índice del registro previo que se va a recuperar. Si tiene éxito en hallar el índice del siguiente registro, devuelve el índice como un entero, de lo contrario, devuelve no válido.

obtenerTamañoRegistro (IdAlmacen, IdRegistro) - Esta función recibe dos enteros como parámetros: IdAlmacen, que contiene el índice del archivo abierto e IdRegistro, que contiene el índice del registro que se va a dimensionar. Si tiene éxito en hallar el índice de registro, devuelve un entero con su tamaño en bytes, de lo contrario, devuelve no válido.

establecerRegistro (IdAlmacen, IdRegistro, ValorRegistro) - Esta función recibe como parámetro los enteros IdAlmacen, que contiene el índice del archivo abierto e IdRegistro, que contiene el índice del registro que se va a cambiar y la cadena ValorRegistro, que contiene los nuevos valores que se van a registrar. Esta función es útil para actualizar información en un registro dado, siempre que no altere su tamaño. En caso de cambio con éxito, se devolverá el IdRegistro de ese registro, de lo contrario el valor devuelto será no válido. Si el tamaño del registro no es el mismo, devuelve no válido. En este caso, se ha de excluir y se debería registrar la información nueva en un registro nuevo.

crearBaseDatos (NombreBase, NombreOrigen) - Esta función crea un almacén de registros con el nombre NombreBase a partir de un archivo de origen NombreOrigen, ambos proporcionados como parámetros. La función devuelve un cero o un entero positivo si tiene éxito. Devuelve no válido si no puede crear el almacén de registros.

hallarRegistro (BD, Clave, Pos, Separador, Ordenado) - Esta función busca una cadena Tecla ubicada en la posición Pos en relación con el Separador. Si la BD de archivos esté organizada en relación con el campo deseado, el parámetro Ordenado puede ser igual a verdadero, permitiendo la realización de una búsqueda binaria en una BD.

establecerClaveOrden (BD, Pos, Separador) - Esta función define el orden de búsqueda en el archivo de BD de acuerdo con la posición Pos en relación con el Separador, para las funciones hallarPrimerRegistro ( ) y hallarRegistroSiguiente ( ).

hallarPrimerRegistro (IdAlmacen) - Esta función recibe como parámetro el manejador (IdAlmacen) del archivo abierto y devuelve su primer índice de registro. La función establecerClaveOrden ( ) se ha de haber llamado previamente. Devuelve un índice entero en relación con la posición del primer registro de acuerdo con el orden definido por la función establecerClaveOrden (BD, Pos, Separador), o no válido en caso de fallo (no puede abrir el archivo, por ejemplo).

hallarRegistroSiguiente (IdAlmacen) - Esta función recibe como parámetro el manejador (IdAlmacen) del archivo abierto y devuelve el siguiente índice de registro. La función hallarPrimerRegistro ( ) se ha de haber llamado previamente. Devuelve un índice entero en relación con la posición del siguiente registro de acuerdo con el orden definido por la función establecerClaveOrden (BD, Pos, Separador), o no válido en caso de fallo (no puede abrir el archivo, por ejemplo).

hallarRegistroRepetido (BD, Clave, Pos, Separador, Ordenado) - Esta función funciona como hallarRegistro ( ),

pero considera la aparición de registros con claves repetidas. Devuelve el Id del registro hallado.
defragAlmacen (NombreAlmacen) - Esta función elimina físicamente, del RegistroAlmacenes denominado NombreAlmacen, los registros borrados. El Almacén de registros no se ha de abrir cuando es llamado. Devuelve no válido si falla.
adjuntarultimoRegistro (IdAlmacen, cadena) - Esta función adjunta el argumento de tipo cadena al contenido del último registro físico, si no se borra. Se debería llamar justo después de añadirRegistro ( ). Devuelve el IdRegistro.
añadirRegistroDeVars (IdAlmacen, listaVar) - Esta función añade un registro al RegistroAlmacenes abierto identificado por IdAlmacen con las variables definidas en la lista de nombres de var separados por punto y coma listaVar. Devuelve el id del registro añadido, que se debería haber guardado para su uso futuro con las funciones establecerVarsDeRegistro ( ) u obtenerVarDeRegistro ( ).
establecerVarsDeRegistro (IdAlmacen, IdRegistro) - Esta función establece el contexto de WML con las variables obtenidas del registro identificado con IdRegistro del RegistroAlmacenes abierto identificado por IdAlmacen. Devuelve el id del registro añadido leído, que se debería haber guardado con la función añadirRegistroDeVars ( ).
obtenerVarDeRegistro (IdAlmacen, IdRegistro, NombreVar) - Esta función devuelve el valor de la variable denominada NombreVar a partir del registro identificado con IdRegistro del RegistroAlmacenes abierto identificado por IdAlmacen. Si la variable no está definida por el registro, devuelve la cadena vacía.
establecerVista (BD, Clave, Pos, Separador) - Esta función define un filtro de navegación para una BD de almacén de registros, que tiene una clave de valor en el campo en la posición Pos en relación con el Separador. Se supone que el almacén de registros está ordenado por el mismo campo. Las vistas creadas con esta función ofrecen una selección completamente transparente en el almacén de registros a las funciones de navegación básicas.
borrarVista (BD) - Esta función borra los registros seleccionados por la vista actualmente establecida para una BD de almacén de registros.
establecerVistaFiltro (BD, Clave, Pos, Separador) - Esta función define un filtro de navegación para una BD de almacén de registros, que tiene una clave de valor en el campo en la posición Pos en relación con el Separador. Se supone que el almacén de registros no está ordenado por campo alguno. Las vistas creadas con esta función ofrecen una selección completamente transparente en el almacén de registros a las funciones de navegación básicas, aunque los registros fuera de la vista serán omitidos secuencialmente cuando se llame a RegistroAlmacenes.obtenerIdRegistroSiguiente ( ).

2.4.2 Biblioteca extendida de CONSOLA

Esta Biblioteca contiene un conjunto de funciones usadas para la depuración de errores de aplicaciones.

Lista de funciones:

imprimir (cadena) - Esta función muestra el texto de la *cadena* de depuración de errores en la ventana de información del sistema de cliente. No se inserta carácter de nueva línea alguno después del texto. Devuelve 1 si tiene éxito en mostrar o, de lo contrario, *no válido.*
imprimirLn (cadena) - Esta función muestra el texto de la *cadena* de depuración de errores en la ventana de información del sistema de cliente. Inserta un carácter de nueva línea después del texto. Devuelve 1 si tiene éxito en mostrar, de lo contrario, devuelve *no válido.*

2.4.3 Biblioteca extendida de CRIPTO

Esta Biblioteca contiene un conjunto de funciones usadas para el cifrado y descifrado de información que se va a transmitir con más seguridad a través de la web.

Lista de funciones:

cifrar (Mensaje, ClaveSesion) - Esta función devuelve el parámetro de *Mensaje* cifrado con un algoritmo de DES. Si el tamaño de *mensaje* es mayor que 512 bytes o si el cifrado no tiene éxito, el mensaje devuelve *no válido.* ClaveSesion es la cadena de 16 bytes que se corresponde con una clave de 8 bytes codificada en hexadecimal. El valor devuelto es una cadena de ASCII codificada en hexadecimal con la longitud doble del argumento de Mensaje.
descifrar (Mensaje, ClaveSesion) - Esta función devuelve el parámetro de Mensaje descifrado con un algoritmo de DES. Si el tamaño de *Mensaje* es mayor que 1024 bytes o si el cifrado no tiene éxito, el mensaje devuelve *no válido.* ClaveSesion es la cadena de 16 bytes que se corresponde con una clave de 8 bytes codificada en hexadecimal. Se supone que el parámetro de *Mensaje* es una cadena codificada en hexadecimal de ASCII con la longitud doble del valor devuelto.
cifrarADES (Mensaje, ClaveSesion) - Esta función devuelve el parámetro de *Mensaje* cifrado con un algoritmo de DES. Antes del cifrado, adjunta como prefijo el *Mensaje* con la representación de BCD de su longitud. Si el tamaño de *Mensaje* es mayor que 512 bytes o si el cifrado no tiene éxito, el mensaje devuelve *no válido.* ClaveSesion es la cadena de 16 bytes que se corresponde con una clave de 8 bytes codificada en hexadecimal. El valor devuelto es una cadena de ASCII codificada en hexadecimal con la longitud doble del argumento de *Mensaje*.
descifrarADES (Mensaje, ClaveSesion) - Esta función devuelve el parámetro de Mensaje descifrado con un algoritmo de DES. Si el tamaño de *Mensaje* es mayor que 1024 bytes o si el cifrado no tiene éxito, el mensaje

devuelve *no válido.* Se supone que el parámetro de *Mensaje* es una cadena de ASCII codificada en hexadecimal con la longitud doble del valor devuelto más el tamaño de la longitud de la cadena.
oexclusivo (cadena1, cadena2) - Esta función recibe dos cadenas de ASCII codificadas en hexadecimal y devuelve la cadena de ASCII codificada en hexadecimal que se corresponde con oexclusivo entre sus valores binarios.
cifrarADESHexa (Mensaje. ClaveSesion) - Esta función devuelve el parámetro de *Mensaje* cifrado con un algoritmo de DES. Antes del cifrado, adjunta como prefijo el *Mensaje* con la representación de BCD de su longitud. Se supone que el argumento de *Mensaje* es una cadena de ASCII codificada en hexadecimal. Si la longitud de Mensaje es mayor que 1024 bytes o si el cifrado no tiene éxito, el mensaje devuelve *no válido.* ClaveSesion es la cadena de 16 bytes que se corresponde con una clave de 8 bytes codificada en hexadecimal. El valor devuelto es una cadena de ASCII codificada en hexadecimal con la misma longitud del argumento de *Mensaje*.
descifrarADESHexa (Mensaje, ClaveSesion) - Esta función devuelve el parámetro de Mensaje descifrado con un algoritmo de DES. Si el tamaño de *Mensaje* es mayor que 1024 bytes o si el cifrado no tiene éxito, el mensaje devuelve *no válido.* ClaveSesion es la cadena de 16 bytes que se corresponde con una clave de 8 bytes codificada en hexadecimal. Se supone que el parámetro de *Mensaje* es una cadena de ASCII codificada en hexadecimal con la misma longitud del valor devuelto más el tamaño de la longitud de la cadena.
cifrarHexa (Mensaje, ClaveSesion) - Igual que cifrar, pero el mensaje está en una representación hexadecimal. El valor devuelto es una cadena de ASCII codificada en hexadecimal con la longitud doble del argumento de *Mensaje*.
descifrarHexa (Mensaje, ClaveSesion) - Igual que cifrar, pero el mensaje está en una representación hexadecimal. El valor devuelto es el parámetro de Mensaje descifrado con un algoritmo de DES.

2.4.4 - Biblioteca de IMPRESORA

Esta biblioteca contiene un conjunto de funciones usadas para la impresión.

Lista de funciones:

abrir ( ) - Esta función abre la salida a la impresora. Si tiene éxito, devuelve el manejador obtenido, de lo contrario devuelve no válido.
establecerModoAnchura (modo) - Esta función configura la anchura para la fuente que se envía a la impresora, tomando como base el parámetro de modo. El modo ha de ser conforme con una conversión a entero o a coma flotante. Si tiene éxito devuelve verdadero, si no o el modo de parámetros es ilegal, devuelve no válido.
establecerModoAltura (modo) - Esta función configura la altura de la fuente que se envía a la impresora, tomando como base el modo de parámetros. El modo ha de ser conforme con una conversión a entero o a coma flotante. Si tiene éxito devuelve verdadero, si no o el modo de parámetros es ilegal, devuelve no válido.
imprimir (cadena) - Esta función imprime el contenido de la cadena. Devuelve verdadero si la impresión tiene éxito, de lo contrario, devuelve no válido.
imprimirLn (cadena) - Esta función imprime el contenido de la cadena y añade un carácter de nueva línea después de la impresión. Devuelve verdadero si la impresión tiene éxito, de lo contrario, devuelve no válido.
cerrar ( ) - Esta función cierra la salida a la impresora, devolviendo una cadena vacía ("").
imprimirLGO (NombreArchivoLOGO) - Esta función recibe como parámetro la ruta al archivo en donde está ubicado el mapa de bits, en formato de LGO, que se imprimirá. La función devuelve o bien un valor negativo en caso de que falle o bien "1" en caso de que tenga éxito en la impresión.

2.4.4 Biblioteca de TARJETAS

Esta biblioteca contiene un conjunto de funciones usadas para comprobación de intervalo BIN de tarjeta de financiación.

Lista de funciones:

comprobarIntervalos (PANactual, sufijoEtiquetaPrivada) - Esta función comprueba el intervalo del PAN de tarjeta dado frente a los leídos del entorno de PDV en las variables MAX*lblX* y MIN*lblX*, en donde lbl representa el sufijoEtiquetaPrivada dado y X representa el contador de intervalos para ese marcador. X comienza en 0. Si la PANactual concuerda con el intervalo X, entonces INF*lblX*. Si no concuerda con intervalo alguno, devuelve *no válido.* Si no se da sufijo de marcador privado alguno, supone el valor por defecto "PAN".
comprobarIntervalosenAlmacen (PANactual, NombreAlmacen) - Esta función comprueba el intervalo del PAN de tarjeta dado frente a los leídos de los registros del RegistroAlmacenes de NombreAlmacen. Se supone que cada registro tiene una cadena de tres elementos separados por punto y coma, por ejemplo "4400; 4999; info devuelta". El PAN se comprueba frente a los dos primeros elementos de cada registro hasta que encaja entre los mismos. La función devuelve el tercer elemento del primer registro que concuerda. Si no concuerda con el intervalo de cualquier registro, devuelve *no válido.*

2.4.5 Biblioteca de SISTEMA

Esta biblioteca contiene un conjunto de funciones usadas para acceder a las utilidades de tiempo del sistema.

Lista de funciones:

tiempoActualSegs ( ) - Esta función devuelve el tiempo de PDV actual en segundos desde 01/01/1970, 00:00:00.
horafechaASegundos (horafecha) - Esta función devuelve el número de segundos transcurridos desde 01/01/1970, 00:00:00 a la fecha y hora expresadas por *horafecha* con formato "AAAAMMDDHHMMSS".
segundosAhorafecha (segundos) - Esta función devuelve la fecha y hora que se corresponde con el número de segundos transcurridos desde 01/01/1970, 00:00:00 expresada en *segundos.* El valor devuelto tiene el formato "AAAAMMDDHHMMSS".
fechaEsValida (horafecha) - Esta función devuelve verdadero si el argumento de horafecha expresa una fecha/hora válida con formato "AAAAMMDDHHMMSS".
horafecha ( ) - Esta función devuelve la fecha y hora actual de PDV. El formato del valor devuelto es "AAAAMMDDHHMMSS".
ticsActuales ( ) - Esta función devuelve el número de milisegundos desde que se inició el PDV.

2.4.6 Biblioteca de ISO

Esta biblioteca contiene un conjunto de funciones para ocuparse de los mensajes de ISO-8583.

carAHex (Cadena) -Esta función recibe una Cadena como Entrada y devuelve otra Cadena con la representación HEXADECIMAL de la Cadena de entrada en ASCII.
hexACar (Cadena) - Esta función recibe una Cadena como Entrada que es una representación HEXADECIMAL de una Cadena (en ASCII) y devuelve la cadena representada. La Cadena de retorno siempre tendrá la mitad de tamaño que la Cadena de origen. Si la longitud de la Cadena de origen es impar, la función devolverá *no válido.*
leerEmpaquetador (Cadena) - Esta función actualiza el empaquetador que se usa para empaquetar el mensaje de ISO-8583. La Cadena de entrada es la RUTA del archivo con la definición de empaquetador.

El valor de devolución es el código de error. Si la devolución es cero, no tuvo lugar error alguno.

El formato del archivo que define el empaquetador es:
IDCAMPO = EMPAQUETADOR; LON<; RUTAARCHIVO EMPAQUETADORINTERNO>

Ejemplo:

```
# este es un comentario
0 = SIB_NUMERICO; 4
1 = SIB_MAPABITS; 16
2 = SIB_LLNUM; 19
3 = SIB_NUMERICO; 6
# 4 = SIB_NUMERICO; 12
34 = SIB_LLLIBINARIO; 999; f: subdef.pdf
```

Para reducir el tamaño del archivo y disminuir el tiempo de análisis, no se permiten caracteres de espacio en un archivo de definición de empaquetador. Este archivo es un archivo de texto ordinario (formato del texto).

Los campos de ISO tienen un empaquetador, que identifica el formato en el que se enviarán los datos en el mensaje de ISO. Un campo puede, opcionalmente, tener campos internos. Se puede definir un empaquetador interno para dejar que el sistema de cliente analice el campo. Para que esto ocurra, se puede especificar una ruta de archivo que contiene la definición de empaquetador interno.

Los empaquetadores posibles son: SIB_MAPABITS, SI_HEX, SIA_LLHEX, SIA_LLLHEX, SIB_LLHEX, SIB_LLLHEX, SI_BINARIO, SIA_LLBINARIO, SIA_LLLBINARIO, SIB_LLBINARIO, SIB_LLLBINARIO, SI_CAR, SIA_LCAR, SIA_LLCAR, SIA_LLLCAR, SIB_LLCAR, SIB_LLLCAR, SIA_NUMERICO, SIA_LLNUM, SIA_LLLNUM, SIB_NUMERICO, SIB_LLNUM, SIB_LLLNUM, I-FA_CANTIDAD, SIB_CANTIDAD, SIB_LLLIHEX, SIB_LLLIBINARIO, SIB_LLLICAR y SIB_LLLINUM.

Hay 6 tipos de empaquetador: Numérico, Cantidad, Carácter, Hexa, Binario y Mapa de bits.

Los caracteres L, LL o LLL indican el tamaño máximo de un campo. Si un campo es SIA_LLLNUM, quiere decir que su tamaño no puede ser mayor que 999.

El prefijo SIA o SIB indica que la porción numérica de un campo se almacena en ASCII o BCD.

La 'I' en algunos nombres de empaquetador (ejemplo: SIB_LLLINUM) indica que se identificará el campo. Esto es útil para definir los empaquetadores internos, en donde se deberían identificar los campos internos.

Numerico: Este empaquetador recibe una Cadena con una representación numérica (ejemplo: "12345").

Resultados:

SIA_NUMERICO: 12345	(número enviado como una Cadena)
SIA_LLNUM: 0512345	(el tamaño de campo en ASCII)
SIA_LLLNUM: 00512345	(el tamaño de campo en ASCII)
SIB_NUMERICO: x12 x34 x50	(número almacenado en BCD)
SIB_LLNUM: x05 x12 x34 x50	(tamaño de campo también en BCD)
SIB_LLLNUM: x00 x05 x12 x34 x50	(tamaño de campo también en BCD)
SIB_LLLINUM: x00 x05 x23 x12 x34 x50	(id es 23. También en BCD)

Cantidad: Este empaquetador es similar al numérico, excepto por el hecho de que va precedido por un carácter que indica la señal. La entrada para este empaquetador puede ser una Cadena como "D12050".

Resultados:

SIA_CANTIDAD: C12345	(número enviado como una Cadena)
SIB_CANTIDAD: x43 x12 x34 x50	(número enviado en BCD)

Carácter: Este empaquetador recibe una Cadena como "¡este texto es correcto! 12345". Resultados:

SIA_CAR: 12345	(campo enviado como una Cadena)
SIA_LCAR: 512345	(el tamaño de campo en ASCII)
SIA_LLCAR: 0512345	(el tamaño de campo en ASCII)
SIA_LLLCAR: 00512345	(el tamaño de campo en ASCII)
SIB_CAR: x31 x32 x33 x34 x35	(campo enviado como una Cadena)
SIB_LLCAR: x05 x31 x32 x33 x34 x35	(tamaño de campo en BCD)
SIB_LLLCAR: x00 x05 x31 x32 x33 x34 x35	(tamaño de campo también en BCD)
SIB_LLLICAR: x00 x05 x23 x31 x32 x33 x34 x35	(id es 23. También en BCD)

Hexa:

Este empaquetador recibe una Cadena que representa (en hexadecimal) un campo binario. La Cadena hexa se convierte en un campo binario y se adjunta en el mensaje.

Resultados:

SI_HEX: x31 x32 x33 x34 x35	(campo enviado en binario)
SIA_LLHEX: x30 x35 x31 x32 x33 x34 x35	(tamaño (05) en ASCII)
SIA_LLLHEX: x30 x30 x35 x31 x32 x33 x34 x35	(tamaño (005) en ASCII)
SIB_LLHEX: x05 x31 x32 x33 x34 x35	(tamaño (05) en BCD)
SIB_LLLHEX: x00 x05 x31 x32 x33 x34 x35	(tamaño (005) en BCD)

Binario:

Este empaquetador recibe un almacenamiento intermedio binario y crea un campo binario en el mensaje. Debido a que no hay tipo binario alguno para WML Script, este empaquetador se creó deliberadamente para encapsular campos internos. Los campos internos pueden ser numérico + carácter y el campo exterior se debería definir como binario. Cuando se empaquetan los campos internos, el resultado de almacenamiento intermedio binario (campo exterior) será empaquetado automáticamente como binario por el motor.

Resultados:

SI_BINARIO: x31 x32 x33 x34 x35	(campo enviado en binario)
SIA_LLBIN: x30 x35 x31 x32 x33 x34 x35	(tamaño (05) en ASCII)
SIA_LLLBIN: x30 x30 x35 x31 x32 x33 x34 x35	(tamaño (005) en ASCII)
SIB_LLBIN: x05 x31 x32 x33 x34 x35	(tamaño (05) en BCD)
SIB_LLLBIN: x00 x05 x31 x32 x33 x34 x35	(tamaño (005) en BCD)

Mapa de bits:

Este es un empaquetador especial que almacena un mapa de bits binario en el mensaje. Este mapa de bits se crea basándose en la existencia de los campos de mensaje. Este empaquetador se usa para el campo 1.

transacMensaje (Cadena anfitrión, Int puerto, Cadena lista de campos, Cadena canal. Cadena encabezado, Cadena finalizador, Cadena camposObligatorios) - Esta función envía y recibe un mensaje basándose en el empaquetador configurado y la lista de campos usada. La lista de campos es una Cadena que encapsulan los nombres de campo que se enviarán en el mensaje. Los testigos están separados por un carácter. Los campos se denominan mediante la regla f<ID de campo>. Si el nombre enviado es un nombre de subcampo, la regla es f<ID de campo>s<sub ID>.

Ejemplo de una lista de campos:

"f0; f2; f3; f28; f11; f34s0; f34s2s0; f34s2s2; f34s3; f34s5; f41; f70"
f34 está compuesto por f34s0, f34s2, f34s3 y f34s5, mientras que f34s2 está compuesto por f34s2s0 y f34s2s2. En el presente caso se hallaron tres niveles de campos internos.

El encabezado y el finalizador que se enviarán en el mensaje son cadenas hexadecimales que representan el encabezado y el finalizador binarios que se enviarán.

Ejemplo:

"0600030001" genera x06 x00 x03 x00 x01
El canal seleccionado refleja la forma en la que la longitud de mensaje se almacena en el mensaje. Los canales disponibles son: "ASCII", "RAW", "NAC" y "NCC".

Ejemplo (el tamaño de mensaje es 256):

ASCII: x30 x32 x35 x36 ("0256")
RAW: x00 x00 x01 x00 (0000-0000 0000-0000 0000-0001 0000-0000)
NAC: x01 x00 (0000-0001 0000-0000)
NCC: x02 x56 (BCD)
Los valores para los campos enviados en el parámetro de lista de campos, ha de almacenar valores como variables en memoria. Se los puede establecer con "NavegadorWML.establecervar (nombrecampo, valor);".

El mensaje se enviará al servidor y la secuencia de comandos bloqueará la espera de la respuesta. Si tienen lugar errores, se recuperará como el valor de devolución de la función, de lo contrario se devolverá cero.

El mensaje de respuesta es analizado por las reglas de empaquetador. Los campos y sub campos hallados también se guardarán en variables en memoria. La nomenclatura por convención para esas variables de respuesta es la misma para el envío, pero en lugar de una 'f' que precede al ID, el sufijo será una 'r'.

El argumento de camposObligatorios es una Cadena con formato como lista de campos es. Fuerza que se compruebe la respuesta. Si falta un campo obligatorio, tiene lugar un error y la transacción no se procesa.

Desde el momento en el que la función realiza la devolución y en lo sucesivo, la aplicación ha de recuperar las variables, verificarlas y hacer lo que se desea hacer. Tal vez imprimir un recibo.
transacDeRegistro (Cadena anfitrión, Int puerto, Int idAlmacen, Int idRegistro, Cadena canal, Cadena encabezado, Cadena finalizador, Cadena camposObligatorios) - Esta función envía y recibe un mensaje basándose en el empaquetador configurado y el conjunto de campos de ISO. Los campos se almacenan en un Registro de un RegistroAlmacenes. La identificación de RegistroAlmacenes se pasa como el parámetro idAlmacen y la identificación de Registro se pasa como parámetro registrado. Con el fin de construir un registro con los campos de ISO, se debería usar la función añadirRegistroDeVars ( ) de la biblioteca de RegistroAlmacenes.

Esta función ofrece una forma alternativa para establecer argumentos (en este caso, los campos de ISO) para una transacción de ISO.

El encabezado y el finalizador que se enviarán en el mensaje son cadenas hexadecimales que representan el encabezado y el finalizador binarios que se enviarán.

Ejemplo:

"0600030001" genera x06 x00 x03 x00 x01
El canal seleccionado refleja la forma en la que la longitud de mensaje se almacena en el mensaje. Los canales disponibles son: "ASCII", "RAW", "NAC" y "NCC".

Ejemplo (el tamaño de mensaje es 256):

ASCII: x30 x32 x35 x36 ("0256")
RAW: x00 x00 x01 x00 (0000-0000 0000-0000 0000-0001 0000-0000)
NAC: x01 x00 (0000-0001 0000-0000)
NCC: x02 x56 (BCD)
El argumento de camposObligatorios es una lista de campos que se van a comprobar si se encuentra presente en la respuesta.

Ejemplo de una lista de camposObligatorios:

"f0; f2; f3; f28; f11; f34s0; f34s2s0; f34s2s2; f34s3; f34s5; f41; f70"
f34 está compuesto por f34s0, f34s2, f34s3 y f34s5, mientras que f34s2 está compuesto por f34s2s0 y f34s2s2. En el presente caso se hallaron tres niveles de campos internos.

CamposObligatorios fuerza que se compruebe la respuesta. Si falta un campo obligatorio, tiene lugar un error y la transacción no se procesa.

Los campos de mensaje de respuesta se almacenan en un Registro nuevo del RegistroAlmacenes y su id es el valor de devolución de la función. Para restablecer los campos en variables, se debería llamar a la función EstablecerVarsDeRegistro ( ) de la biblioteca de RegistroAlmacenes. Las variables se establecen usando la misma regla de nomenclatura que usa transacMensaje ( ).

transacDeRegistroAVar (Cadena anfitrión, Int puerto, Int idAlmacen, Int idRegistro, Cadena canal, Cadena encabezado, Cadena finalizador, Cadena camposObligatorios) - Esta función envía y recibe un mensaje basándose en el empaquetador configurado y el conjunto de campos de ISO. Los campos se almacenan en un registro de un RegistroAlmacenes pero la devolución se establece a la Variable de WML denominada RESPISO. Si la longitud resultante supera la longitud máxima de las variables en WML (512), los bytes en exceso se dividen en unas variables de 500 bytes denominadas RESPISO*n*, n = 1... La Variable de WML NRESPISO también se establece con el número total de variables necesarias para almacenar el resultado, incluyendo RESPISO. La identificación de RegistroAlmacenes se pasa como el parámetro idAlmacen y la identificación de Registro se pasa como parámetro registrado. Con el fin de construir un registro con los campos de ISO, se debería usar la función añadirRegistroDeVars ( ) de la biblioteca de RegistroAlmacenes.

Esta función ofrece una forma alternativa para establecer argumentos (en este caso, los campos de ISO) para una transacción de ISO.

El encabezado y el finalizador que se enviarán en el mensaje son cadenas hexadecimales que representan el encabezado y el finalizador binarios que se enviarán.

Ejemplo:

"0600030001" genera x06 x00 x03 x00 x01
El canal seleccionado refleja la forma en la que la longitud de mensaje se almacena en el mensaje. Los canales disponibles son: "ASCII", "RAW", "NAC" y "NCC".

Ejemplo (el tamaño de mensaje es 256):

ASCII: x30 x32 x35 x36 ("0256")
RAW: x00 x00 x01 x00 (0000-0000 0000-0000 0000-0001 0000-0000)
NAC: x01 x00 (0000-0001 0000-0000)
NCC: x02 x56 (BCD)
El argumento de camposObligatorios es una lista de campos que se van a comprobar si se encuentra presente en la respuesta.

Ejemplo de una lista de camposObligatorios:

"f0; f2; f3; f28; f11; f34s0; f34s2s0; f34s2s2; f34s3; f34s5; f41; f70"
f34 está compuesto por f34s0, f34s2, f34s3 y f34s5, mientras que f34s2 está compuesto por f34s2s0 y f34s2s2. En el presente caso se hallaron tres niveles de campos internos.

CamposObligatorios fuerza que se compruebe la respuesta. Si falta un campo obligatorio, tiene lugar un error y la transacción no se procesa.

Los campos de mensaje de respuesta se almacenan en un Registro nuevo del RegistroAlmacenes y su id es el valor de devolución de la función. Para restablecer los campos en variables, se debería llamar a la función EstablecerVarsDeRegistro ( ) de la biblioteca de RegistroAlmacenes. Las variables se establecen usando la misma regla de nomenclatura que usa transacMensaje ( ).

hexAInt (Cadena hexa, BigEndian Booleano) - Esta función recibe una representación hexadecimal como Entero. El Entero en el motor de secuencias de comandos tiene una longitud de cuatro bytes, por lo que el parámetro hexa puede ser una cadena de 2, 4, 6 u 8 caracteres de largo, representando los bytes del Entero. Esta representación puede encontrarse como "Big Endian" o "Little Endian", y esto se notifica en el parámetro booleano BigEndian (si se está trabajando con una representación de tipo big endian, establecerla a verdadero). El valor de devolución es el Entero deseado.

intAHex (Int i, BigEndian Booleano) - Esta función convierte un entero en una representación hexadecimal de un Entero.

Adicionalmente, el sistema de cliente contiene una biblioteca denominada PinPad que contiene un conjunto de funciones usadas para acceder al dispositivo de teclado de entrada de PIN y el núcleo de EMV de PDV a través de la interfaz convencional de EMV.

La presente invención tiene por objeto su aplicación en una diversidad de áreas de negocio, tales como servicios / seguros de salud, fidelidad, banca remota, pago de facturas, financiación de recarga prepago, tarjetas de regalo, transferencias de dinero y verificación de edad, gestión de tiempo y mano de obra, entre otros.

La aplicación descrita en el presente documento es una realización de las posibilidades proporcionadas por la invención. Se pueden diseñar muchos modos de aplicación diferentes, usando diferentes medios de conexión, incluso dispositivos y redes. El alcance de la invención se define por las reivindicaciones.

REIVINDICACIONES

1. Sistema para soportar aplicaciones Web en un terminal de punto de venta, PDV, (10), comprendiendo el sistema:

un aparato de cliente (15) configurado para conectar el terminal de PDV (10) con una red usando una pluralidad de protocolos de comunicación y protocolos de transacción, protocolos de transacción que incluyen unas operaciones de HTTP, de ISO-8583 y de XML; incluyendo el aparato de cliente (15) al menos una extensión de WML y Script personalizada añadida a normas de WAP preexistentes para formar un entorno de desarrollo de aplicaciones y de tiempo de ejecución para desarrollar aplicaciones para su descarga y uso en el terminal de PDV, dispuesto el aparato de cliente (15) para:

descargar, ejecutar y actualizar aplicaciones a partir de unos servidores de aplicaciones (30) conectados con la red usando HTTP sobre TCP/IP;
posibilitar que una o más aplicaciones usen múltiples protocolos de comunicación cuando se conectan con distintas redes de acceso;
extender un entorno de tiempo de ejecución basado en web para llevar a cabo operaciones de ISO-8583 electrónicas usando dispositivos periféricos de PDV, configurado el entorno de tiempo de ejecución basado en web para soportar la persistencia de información y para evitar una interferencia multi-aplicación al soportar multi-aplicaciones llevadas a cabo independientemente en distintos entornos de ejecución virtual; y
reconocer, acceder a y controlar una pluralidad de dispositivos periféricos de PDV, comprendiendo los dispositivos periféricos de PDV unos dispositivos periféricos de entrada y salida,

en donde el sistema comprende adicionalmente:
unos servidores de transacción (35) configurados para que acceda a los mismos el sistema de cliente (15) para ejecutar operaciones electrónicas usando protocolos de transacción diferentes.

2. Sistema para soportar aplicaciones Web en el terminal de PDV de acuerdo con la reivindicación 1, en donde dichos protocolos de transacción diferentes son HTTP, ISO 8583 o XML.

3. Sistema para soportar aplicaciones Web en el terminal de PDV de acuerdo con la reivindicación 1, en donde dichos protocolos de comunicación (20) se pueden seleccionar de entre un grupo que consiste en TCP/IP, PPP, SDLC, X.28, GPRS, CDMA, CDMA 1X, Ethernet, GSM, Wi-Fi, Bluetooth e IR.

4. Sistema para soportar aplicaciones Web en el terminal de PDV de acuerdo con la reivindicación 1, en donde los dispositivos periféricos se pueden seleccionar de entre un grupo que consiste en lectores de tarjetas magnéticas, lectores de tarjetas inteligentes, impresoras, teclados de entrada de PIN, lectores de código de barras, lectores de cheques, teclados, lector de tarjetas sin contacto Mifare y pantalla táctil.

5. Sistema para soportar aplicaciones Web en el terminal de PDV de acuerdo con la reivindicación 1, en donde la extensión de WML y Script personalizada añadida a normas de WAP preexistentes comprende adicionalmente unos medios para ejecutar una operación electrónica con tarjetas con chip.

6. El sistema para soportar aplicaciones Web en el terminal de PDV de acuerdo con la reivindicación 1, en donde el sistema comprende adicionalmente un servidor de aplicaciones; y
en donde el servidor de aplicaciones está configurado para:

procesar una solicitud procedente del sistema de cliente,
entrar en contacto con al menos uno de los servidores de transacción para recuperar ajustes de configuración que describen el sistema de cliente, utilizar los ajustes de configuración para construir una versión personalizada de una aplicación para el sistema de cliente y enviar la versión personalizada de la aplicación para el sistema de cliente al sistema de cliente para su ejecución por el sistema de cliente.

7. El sistema para soportar aplicaciones Web en el terminal de PDV de acuerdo con la reivindicación 6 en donde los ajustes de configuración se recuperan del al menos uno de los servidores de transacción al enviar, el servidor de aplicaciones, un mensaje de ISO-8583 por medio de una conexión de TCP/IP al servidor de transacción.

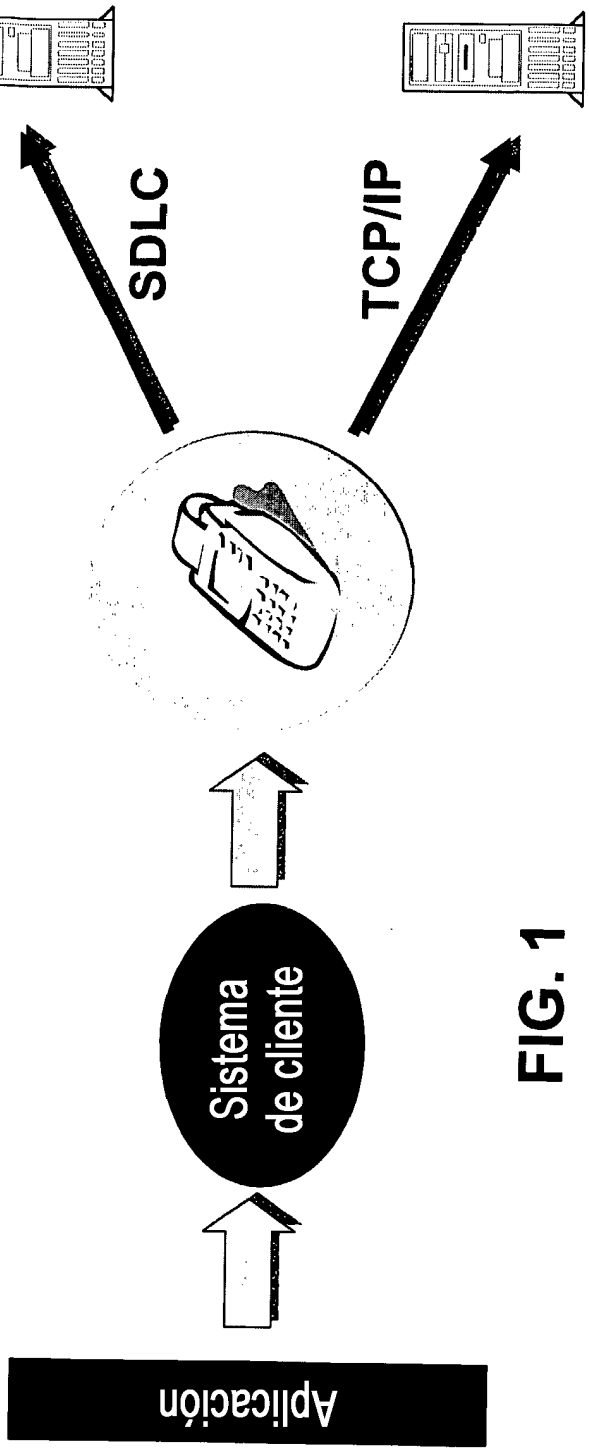

FIG. 1

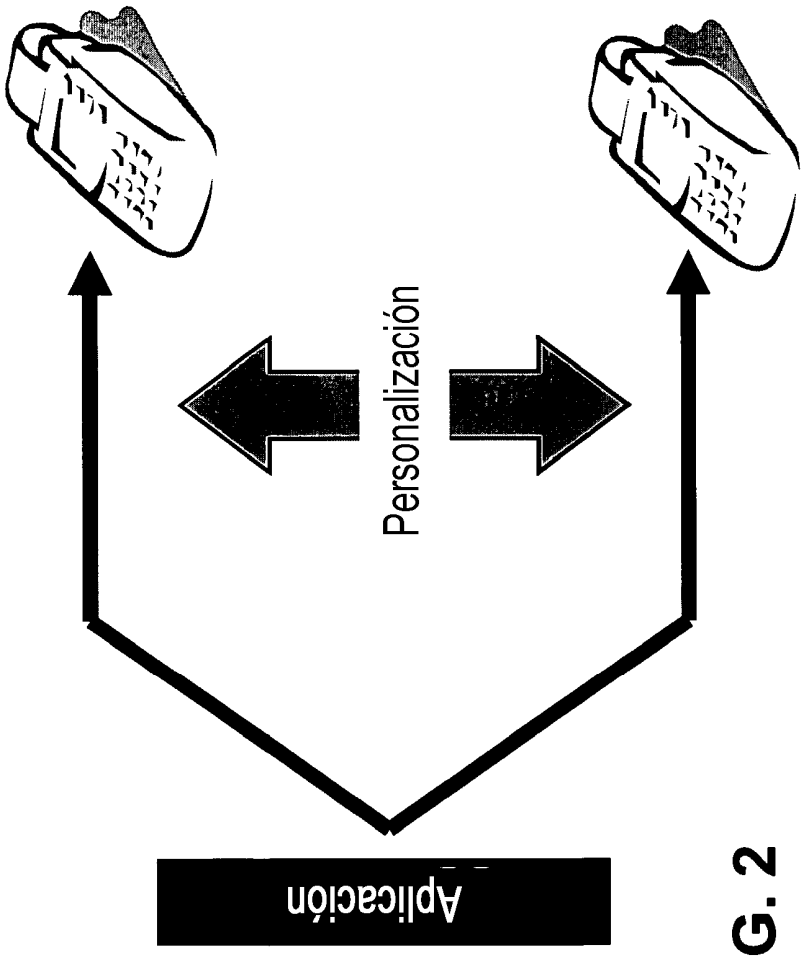

FIG. 2

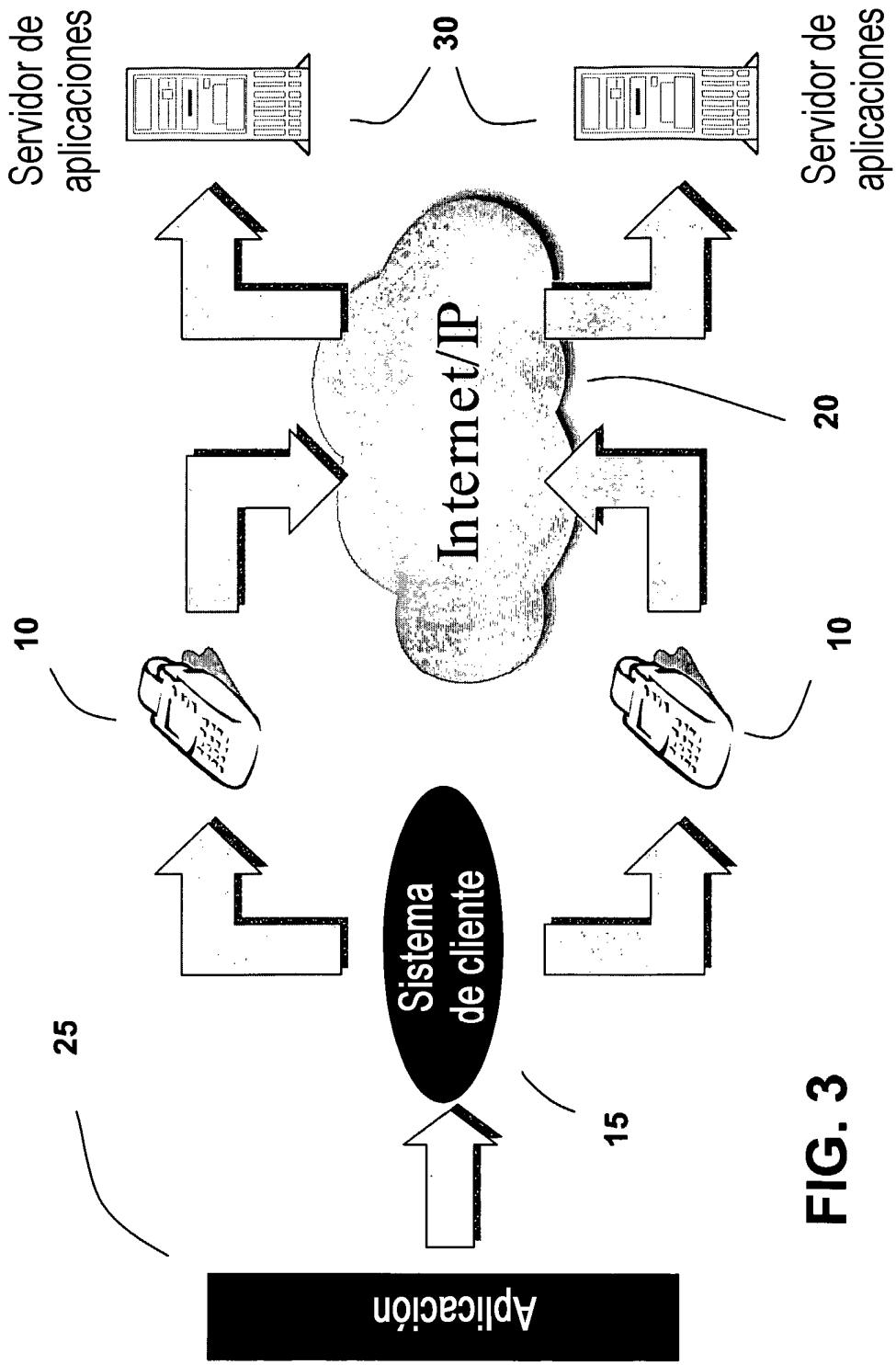
Servidor de aplicaciones
30
Servidor de aplicaciones
Internet/IP
20
10
10
Sistema de cliente
25
15
Aplicación

FIG. 3

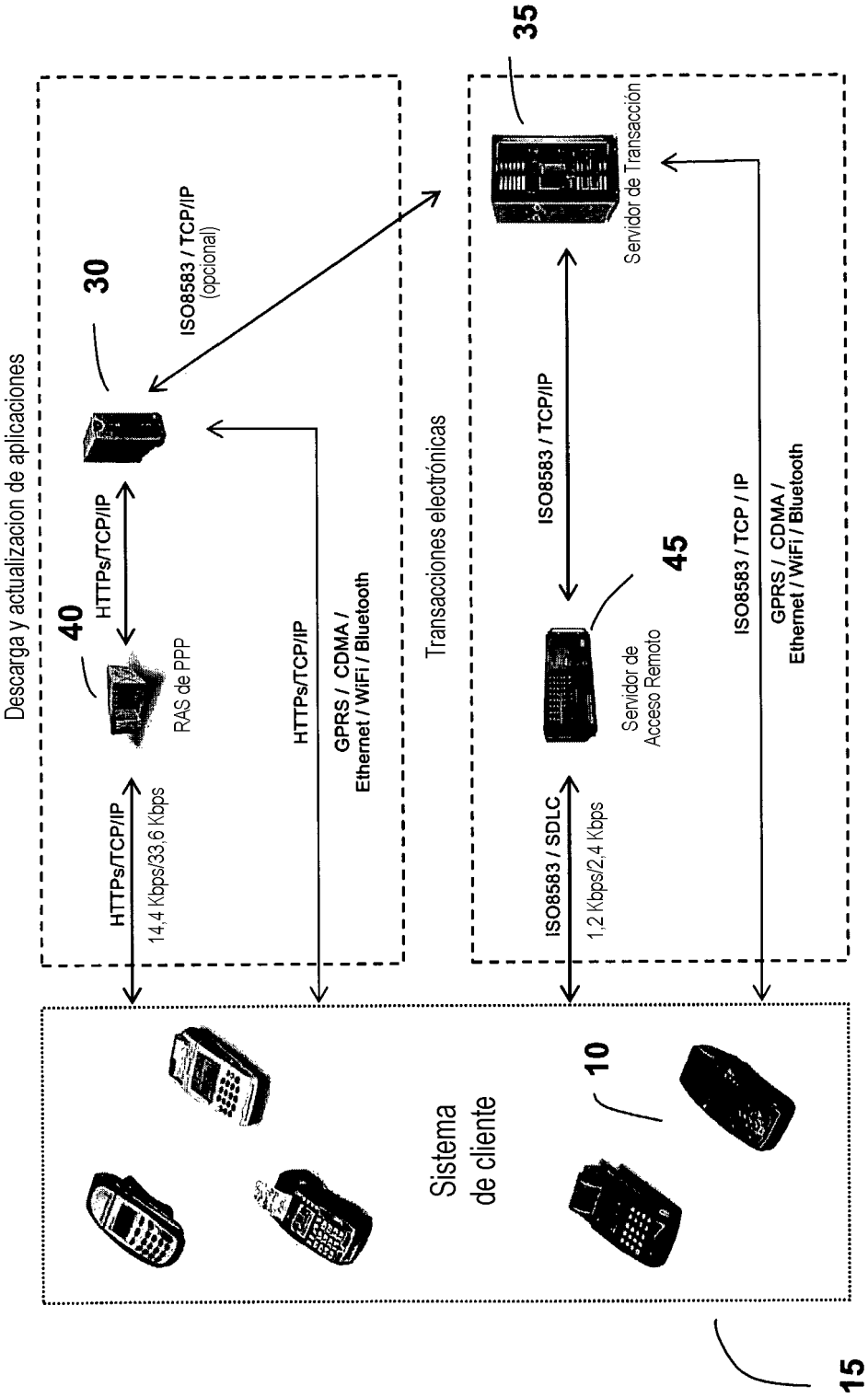
Descarga y actualizacion de aplicaciones
30
ISO8583 / TCP/IP
(opcional)
HTTPs/TCP/IP
40
RAS de PPP
HTTPs/TCP/IP
14,4 Kbps/33,6 Kbps
HTTPs/TCP/IP
GPRS / CDMA /
Ethernet / WiFi / Bluetooth
Transacciones electrónicas
35
Servidor de Transacción
ISO8583 / TCP/IP
45
Servidor de
Acceso Remoto
ISO8583 / SDLC
1,2 Kbps/2,4 Kbps
ISO8583 / TCP / IP
GPRS / CDMA /
Ethernet / WiFi / Bluetooth
Sistema
de cliente
10
15

FIG. 4

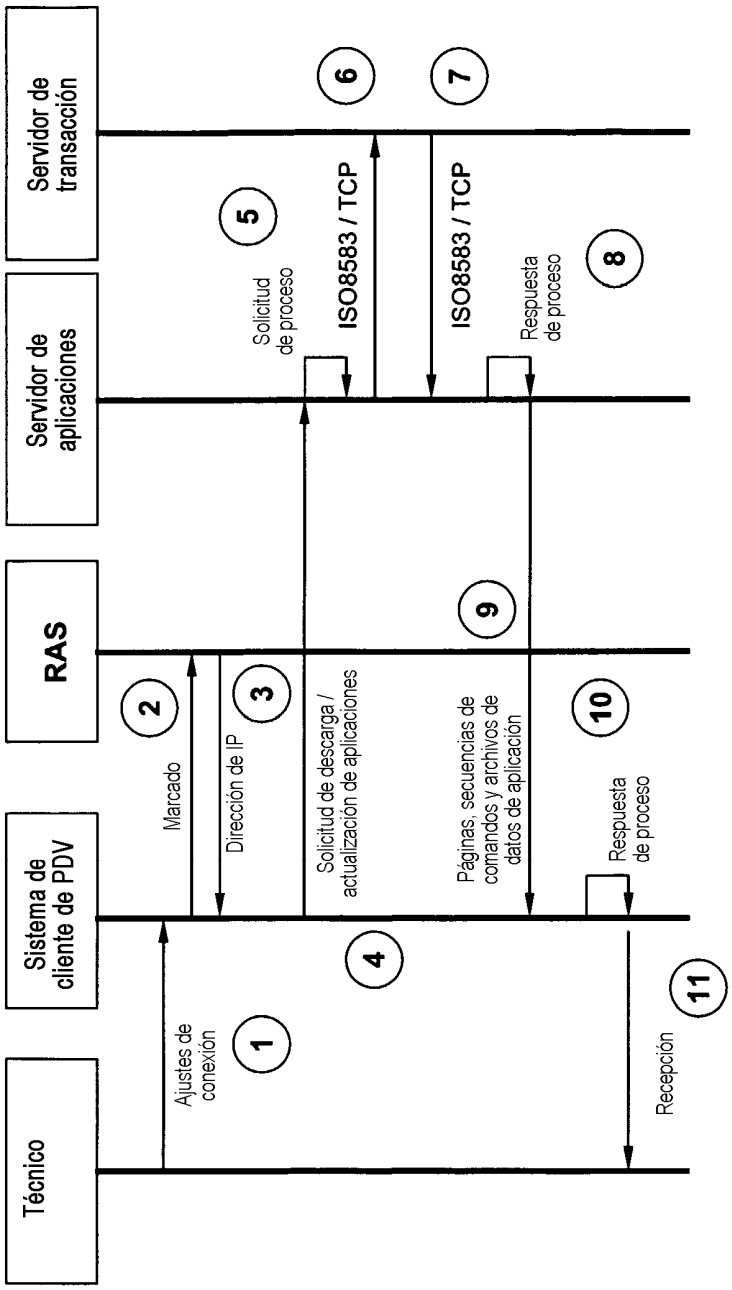
Técnico
Sistema de cliente de PDV
RAS
Servidor de aplicaciones
Servidor de transacción
Ajustes de conexión
1
Marcado
2
Dirección de IP
3
4
Solicitud de descarga / actualización de aplicaciones
Solicitud de proceso
5
ISO8583 / TCP
6
ISO8583 / TCP
7
Respuesta de proceso
8
9
Páginas, secuencias de comandos y archivos de datos de aplicación
Respuesta de proceso
10
Recepción
11

FIG. 5

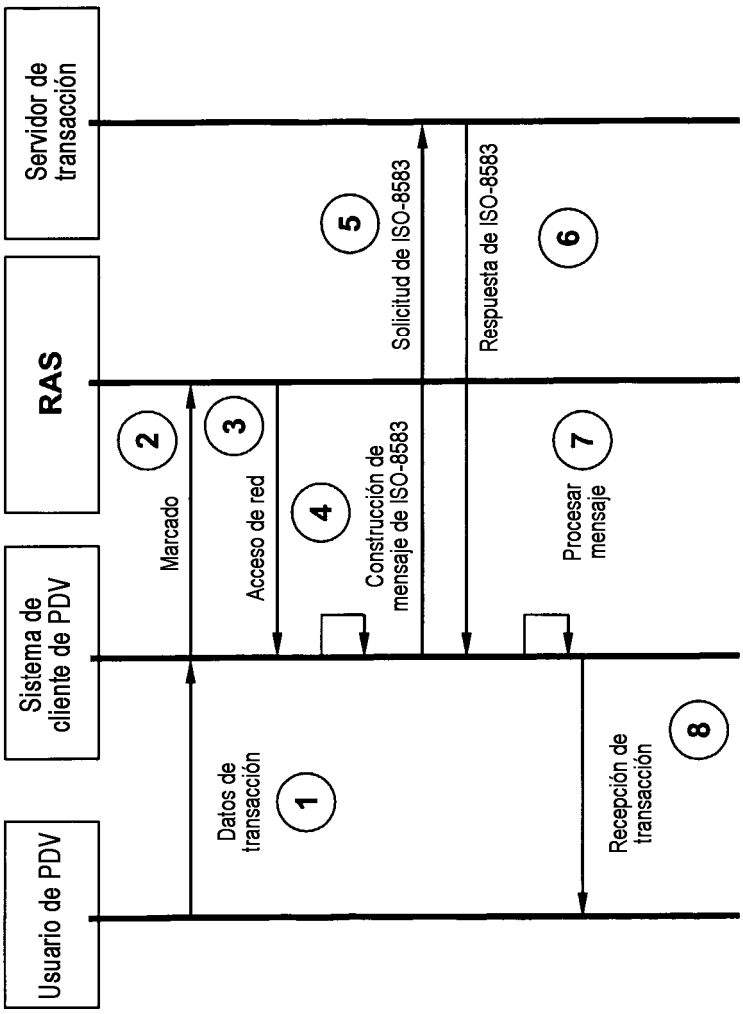
Usuario de PDV
Sistema de cliente de PDV
RAS
Servidor de transacción
Datos de transacción
1
Marcado
2
Acceso de red
3
Construcción de mensaje de ISO-8583
4
Solicitud de ISO-8583
5
Respuesta de ISO-8583
6
Procesar mensaje
7
Recepción de transacción
8

FIG. 6

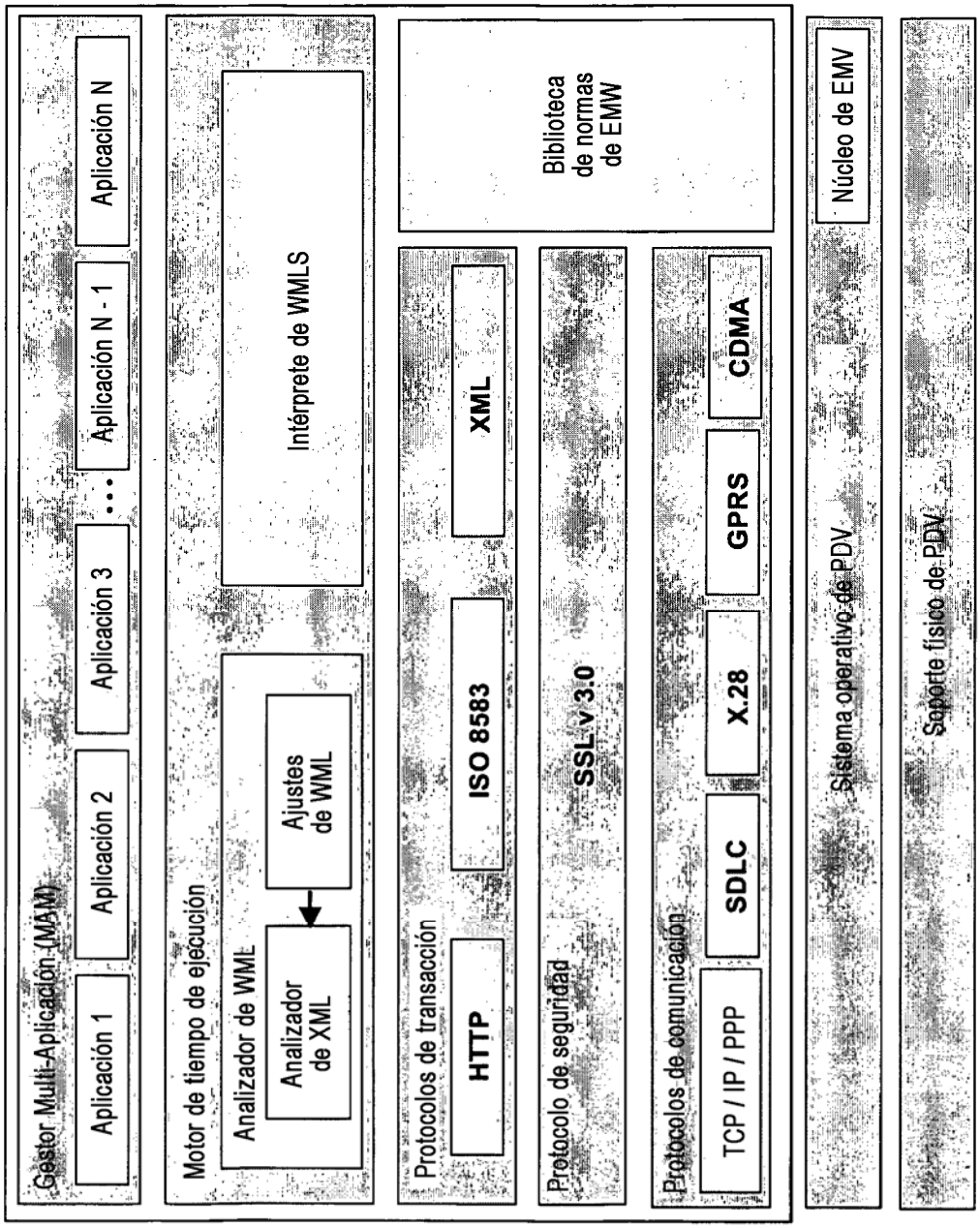
Gestor Multi-Aplicación (MAM)
Aplicación 1
Aplicación 2
Aplicación 3
...
Aplicación N - 1
Aplicación N
Motor de tiempo de ejecución
Analizador de WML
Analizador de XML
Ajustes de WML
Intérprete de WMLS
Protocolos de transacción
HTTP
ISO 8583
XML
Protocolo de seguridad
SSL v 3.0
Biblioteca de normas de EMW
Protocolos de comunicación
TCP / IP / PPP
SDLC
X.28
GPRS
CDMA
Sistema operativo de PDV
Núcleo de EMV
Soporte físico de PDV

FIG. 7