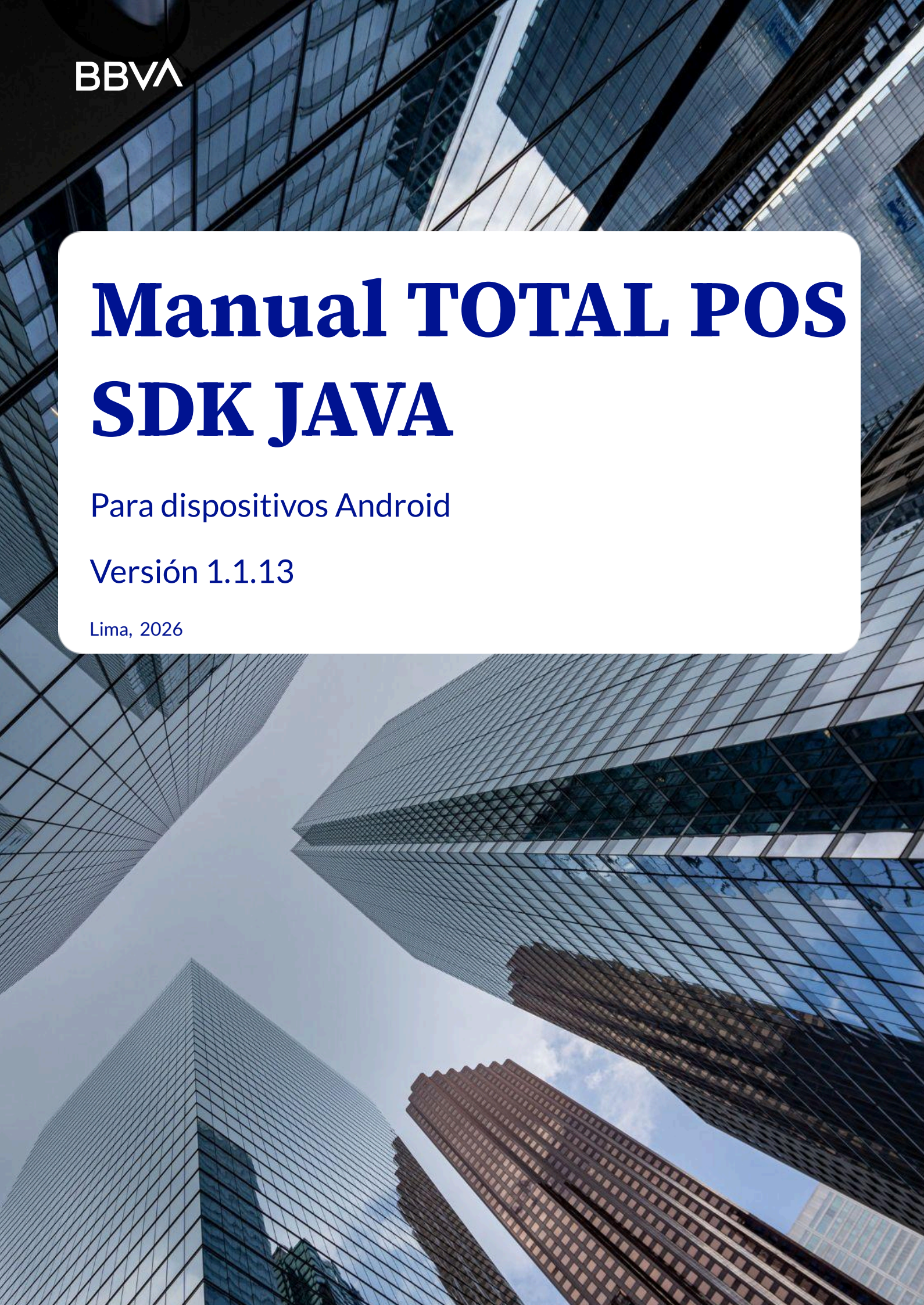


Manual TOTAL POS SDK JAVA

Para dispositivos Android

Versión 1.1.13

Lima, 2026

BBVA

Tabla de contenido

Objetivo del Documento

Describir a detalle las funciones que integran la plataforma transaccional de TOTAL POS SDK para su implementación por parte del comercio en sus puntos de venta, para el servicio de autorización de tarjetas bancarias.

Definiciones

TOTAL POS SDK: Es la aplicación diseñada para que sea centralizada, escalable, segura y orientada a la alta disponibilidad a través de la división de procesos en API's utilizando la tecnología REST, estándares de tecnología (Como mensajería JSON) y buenas prácticas de programación e implementación basados en las normas de PA-DSS y PCI.

Punto de venta: Controla todos los dispositivos para realizar una venta como son la interfaz con el operador, la báscula, la torreta, el lector de barras, impresión del pagaré, etc. Además, se encarga de interactuar con el sistema TOTAL POS SDK para la realización de operaciones financieras con tarjeta.

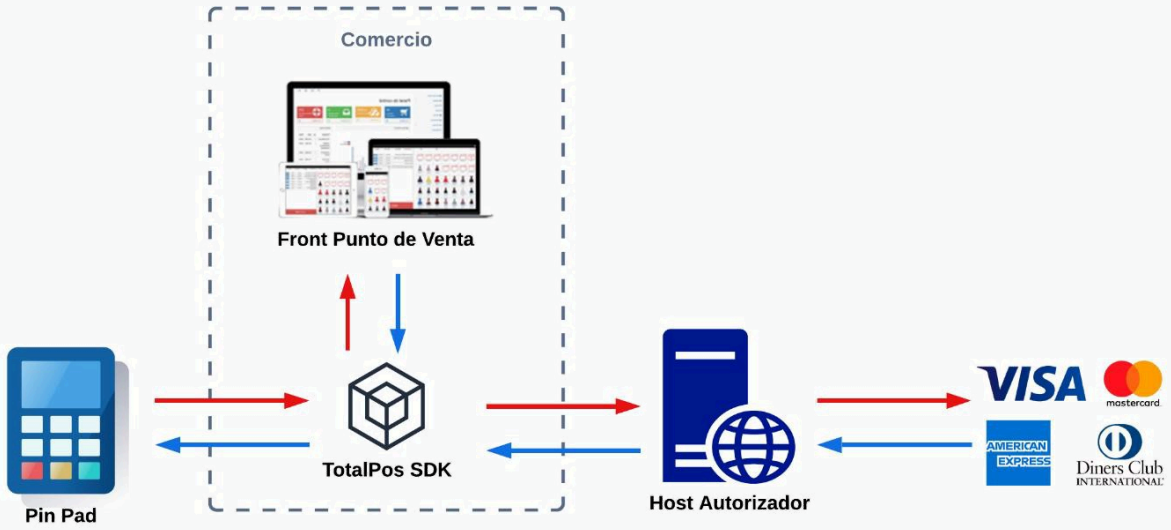
Afiliación: Es proporcionada por BBVA, está ligada con las cuentas en donde le deposita al comercio, además de ser el identificador asignado a los comercios.

Retail: Es el término utilizado para hacer referencia al conjunto de transacciones estándar de ventas, anulación de venta, venta QR y promociones.

Pago rápido: Esquema de pagos rápidos permite a los comercios emitir boletas sin firma para transacciones de compra que califiquen para el servicio.

Carga de Llaves: Es una operación que se debe realizar para que el dispositivo Pin Pad pueda cifrar los datos que obtiene al leer las tarjetas, es de carácter obligatoria. De esta forma se protege la integridad de los datos, si no se realiza la carga de llaves, el Pin Pad no funcionará.

Diagrama de interacción



1. **TOTAL POS SDK** al estar integrado con el punto de venta del comercio es quien manejará el envío y la recepción de la mensajería (JSON) con el host autorizador, al igual que maneja la mensajería con el Pin Pad para el uso de tarjetas de crédito y débito.
2. La conexión del punto de venta es por internet en donde la transacción viajará cifrada. Esta conexión puede ser de internet local o empresarial.
3. El Host Autorizador es quien recibe los requerimientos (transacción) para enviar estos datos a las marcas y solicitar una autorización. Una vez obtenida una respuesta de las marcas se devolverá la respuesta correspondiente al punto de venta.
4. Las marcas son las encargadas de dar una autorización a la transacción correspondiente o declinarla según sea el caso.

Fases para liberación

Para poder liberar a producción, es necesario pasar por las siguientes etapas:

- **Pruebas Unitarias e Integrales**
  - Se requiere que el medio principal de comunicación esté instalado y configurado.
  - Se deben programar ventanas de prueba con BBVA.
  - Tiempo máximo de 10 días en sesiones de 2 horas diarias.
- **Certificación**
  - En conferencia se procesa la matriz de pruebas proporcionada por BBVA.
  - Se deben programar ventanas de prueba.

- Tiempo máximo de dos días en sesiones de 2 horas diarias.
- En caso de ser exitosas las pruebas se programará la entrada en producción.

➢ **Salida a producción**

- Entrega por parte de E-Global el usuario y contraseña para el acceso al portal de generación de credenciales de autenticación.
- Concluida la etapa del PILOTO se procederá a la expansión de sucursales o terminales, según sea el caso.
- Entrega de matriz de escalamiento para la atención de reportes a problemas.

Tiempo promedio para salir a producción desde el inicio del desarrollo: 1 mes (El tiempo puede variar dependiendo de diferentes circunstancias)

Transacciones Soportadas

Funcionalidad Retail

- Venta
- Venta con cuotas
- Anulación de venta
- Venta QR
- Reversos
- Carga de llaves
- Cierre de Turno
- Confirmación de Preautorización
- Anulación de Preautorización
- Anulación de Venta QR
- Preautorización
- Reporte Firma
- Reporte Totalizado
- Reporte Detallado

Requerimientos de Hardware y Software

➢ Sistema operativo

- Windows 11 o superior (64-bit)
- Mac OS X Tiger (10.4) o superior (32/64-bit Intel y Apple Silicon)
- Todas las distribuciones Linux (32/64-bit x86 y ARM)
- Solaris 10 o superior (32/64-bit x86 y SPARC)

➢ CPU Pentium recomendable i3 o superior.

➢ Java (JDK)

- Java 8 (32/64-bit)

    - Java 11 (64-bit)
    - Java 17 (64-bit)
    - Java 21 (64-bit)
- ➢ Aplicación **TOTAL POS SDK.**
- ➢ Configuración de direccionamientos.
- ➢ Desarrollo del comercio para interactuar con la aplicación **TOTAL POS SDK.**
- ➢ Wi-Fi, puerto serial o USB para conexión del Pin Pad.
- ➢ Tener permisos a nivel comunicación para alcanzar al Router o acceso al internet corporativo.
- ➢ Tener permisos de escritura y lectura sobre la ruta raíz del proyecto.
- ➢ Contar con impresora para la emisión de los pagarés, incluyendo los drivers del proveedor (para Pin Pad seriales).

Estos requerimientos se deben considerar para cualquier equipo que el comercio asigne como punto de venta

Dispositivo Pin Pad Android

Un Pin Pad o dispositivo de entrada de Pin es un dispositivo electrónico cuya finalidad es obtener el número de identificación personal (PIN) de la tarjeta habiente. En la actualidad también puede ser usado para obtener la firma electrónica de algunas tarjetas de crédito, así como el código de seguridad.

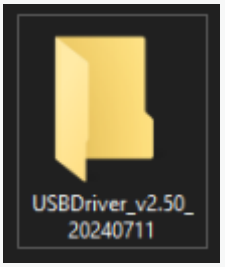
Antes de iniciar la instalación de algún dispositivo Android proporcionado por BBVA, se debe de tener en consideración lo siguiente:

- Utilice únicamente dispositivos procedentes de BBVA.
- No use controladores de dispositivos que no hayan sido proporcionados por BBVA.
- No se garantiza el funcionamiento adecuado con controladores genéricos alternativos.
- Al inicio de la instalación, NO conecte el dispositivo al equipo de cómputo, solo conéctelo al finalizar la ejecución del instalador

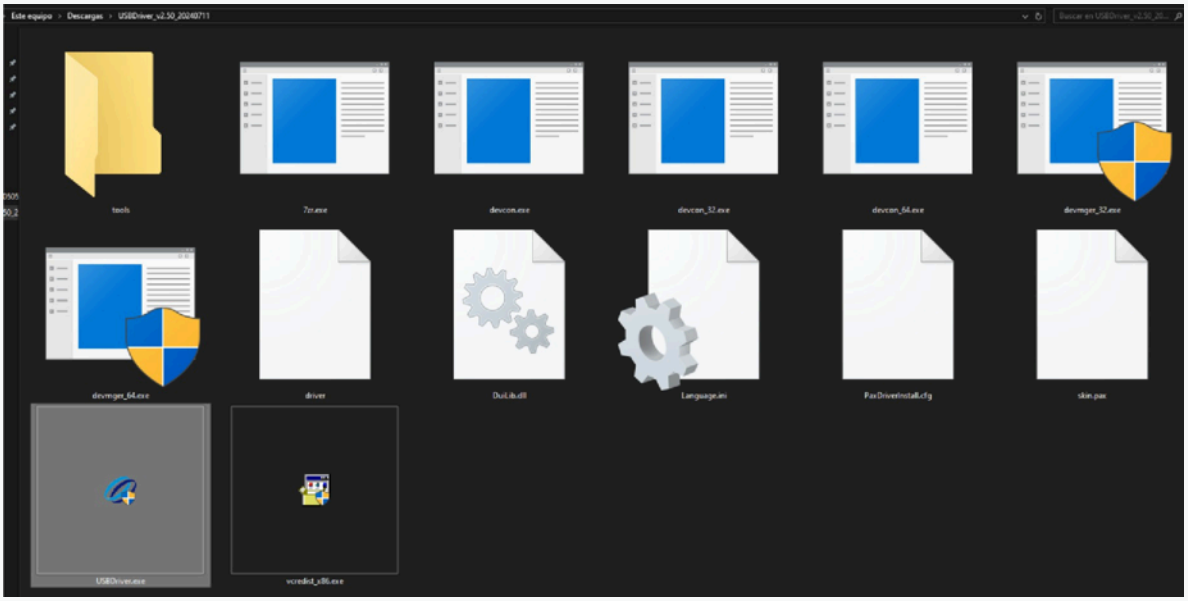
Instalación de drivers

Driver: Software esencial que permite que el sistema operativo (como Windows o macOS) se comunique correctamente con un componente de hardware específico (como una impresora, terminal pos o un dispositivo USB).

Para iniciar, es fundamental contar con el software USBDriver, ya que este programa será la herramienta principal que guiará y permitirá la correcta aplicación de este manual.

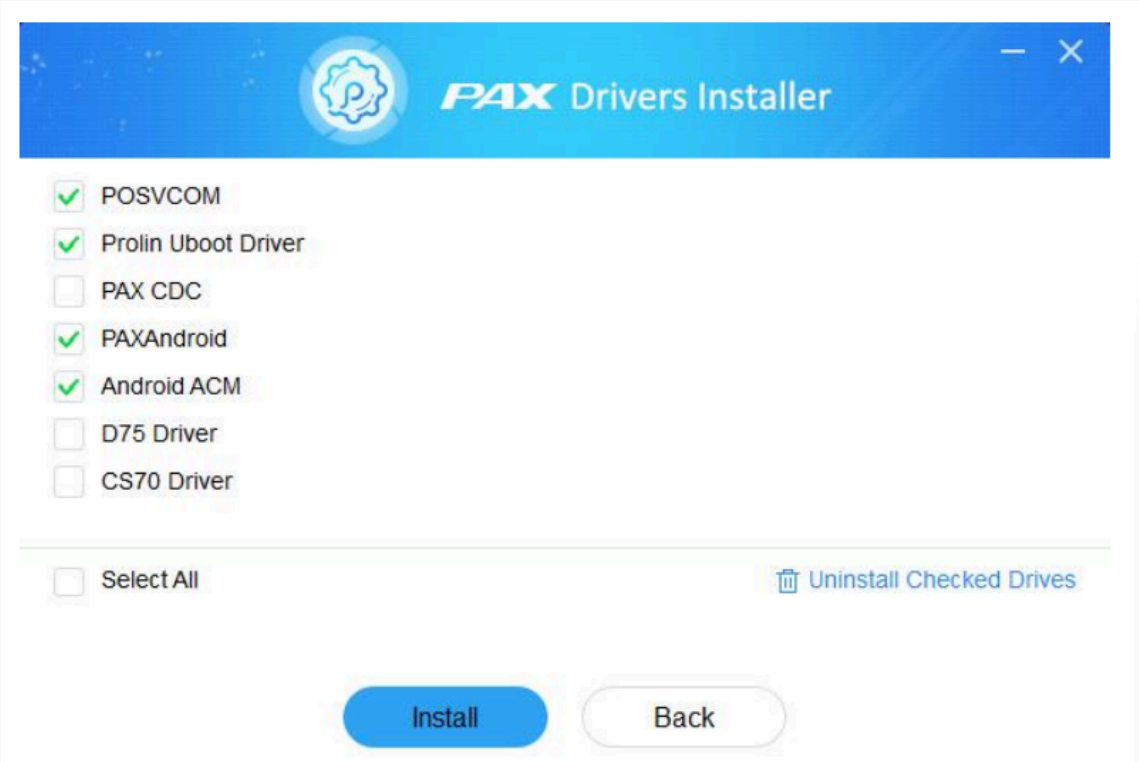


Localiza y ejecuta el archivo USBDriver.exe (doble clic) dentro de la carpeta del software. Cuando se requieran, otorga los permisos de administrador. Esto asegurará la instalación de los componentes esenciales y el reconocimiento de los dispositivos.



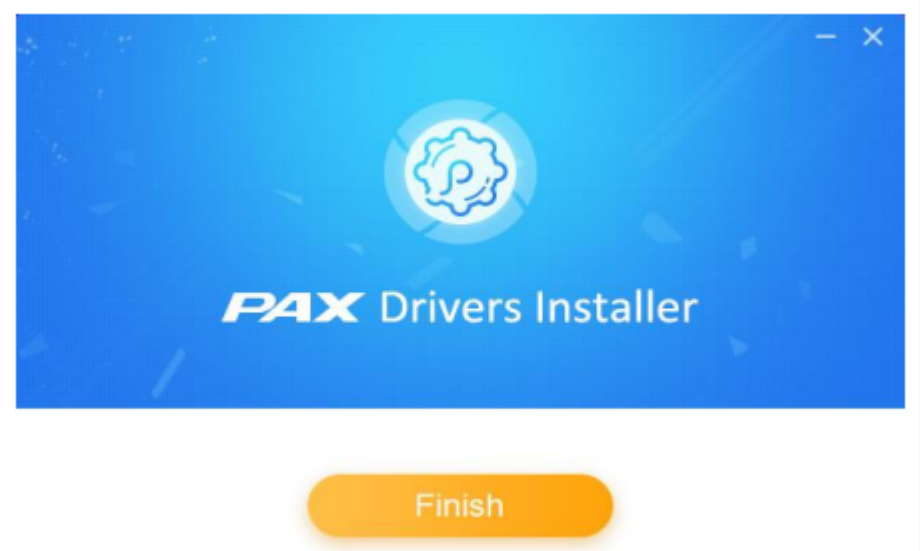
Una vez abierto el PAX Drivers Installer, visualizarás dos opciones de instalación, representadas por dos botones. El primero, de color naranja y etiquetado como "One-Click Install", realizará una instalación estándar de los drivers esenciales para la compatibilidad con los terminales PAX. El segundo botón, llamado "Custom", te brinda la posibilidad de llevar a cabo una instalación avanzada, donde podrás elegir individualmente los drivers que se instalarán en el sistema.

Al optar por la instalación "Custom", se mostrará una lista de drivers con selecciones predeterminadas, las cuales puedes ajustar. En la parte inferior, el botón "Install" ejecuta la instalación de los drivers elegidos, mientras que el otro botón te devuelve a la pantalla principal. A la derecha, el botón "Uninstall Checked Drivers" realiza la desinstalación de los drivers que hayas marcado en la lista y que se encuentren instalados en el equipo.



Al presionar "Install" en cualquier modo de instalación, el software comenzará a instalar los elementos necesarios. Una barra de progreso en la parte inferior indicará el avance. Simplemente espera a que termine.

Cuando la barra de progreso indique que la instalación está completa, un botón naranja con el texto "Finish" se hará visible. Haz clic en él para dar por concluido el proceso y cerrar la ventana del software de instalación.

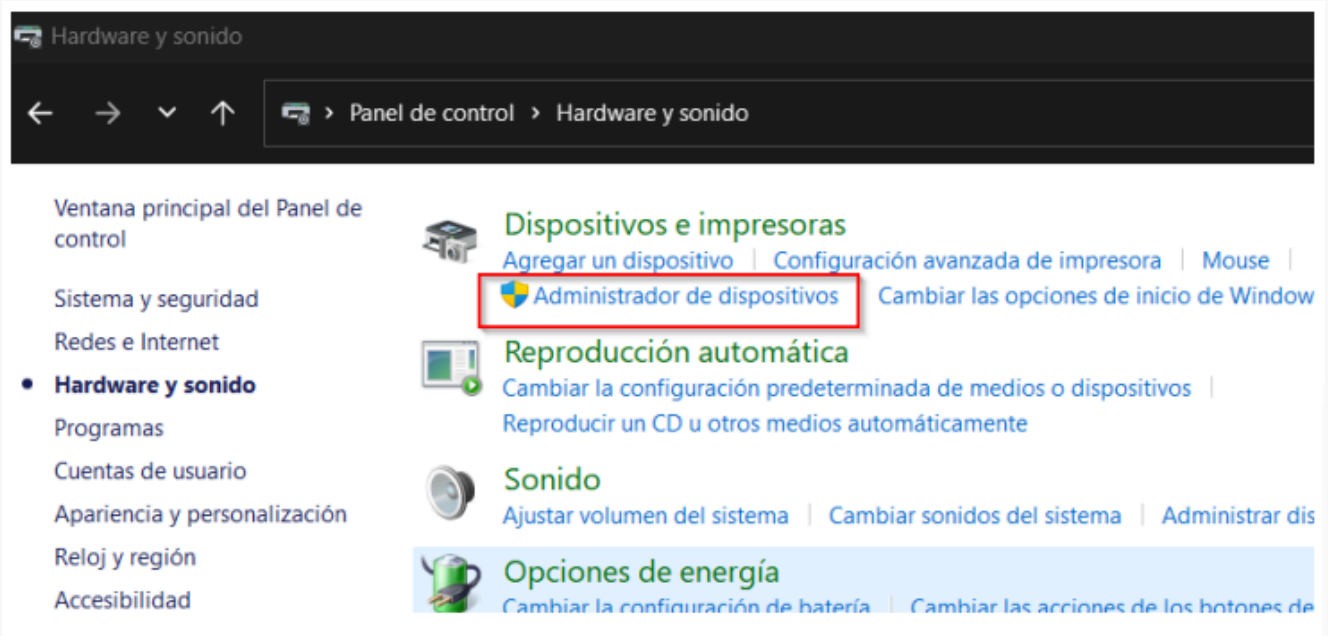


Configuración de dispositivo Android Serial (COM)

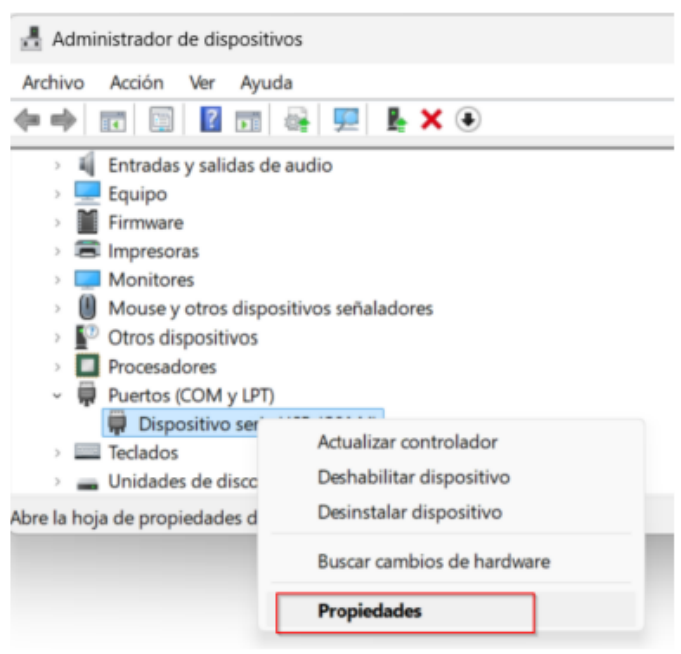
PAX A35

Una vez concluida la instalación conecte el dispositivo Pin Pad a un puerto USB disponible, es necesario que valide el puerto COM asignado por el sistema operativo en el Administrador de dispositivos. Esta consideración es debido a que en algunas circunstancias el puerto COM mostrado en la pantalla principal del administrador de dispositivos no muestra el correcto puerto COM asignado. Verifique el puerto de la siguiente manera:

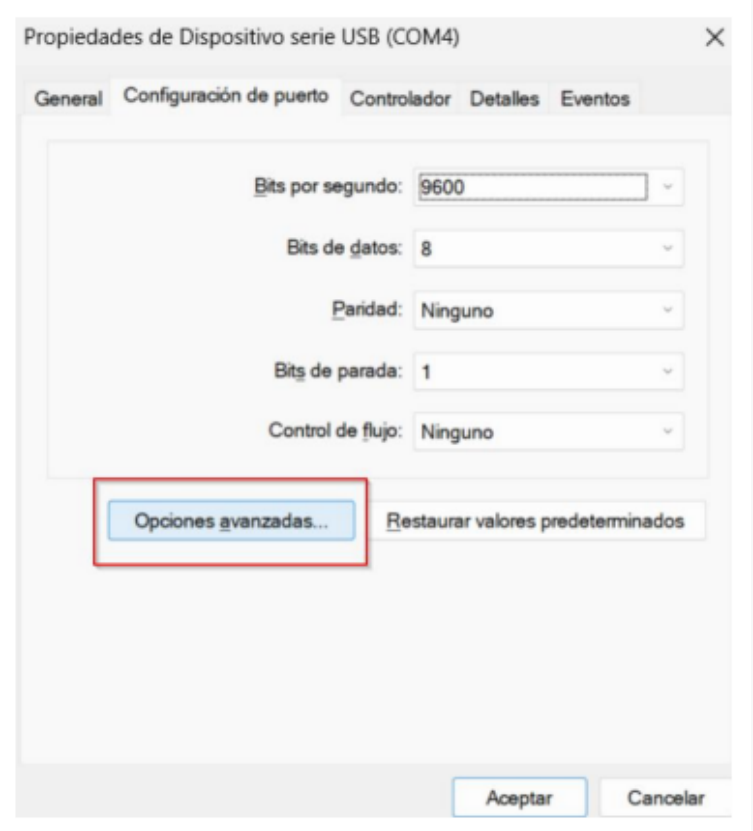
Desde el panel de control seleccione la opción de Administrador de dispositivos (en Windows)



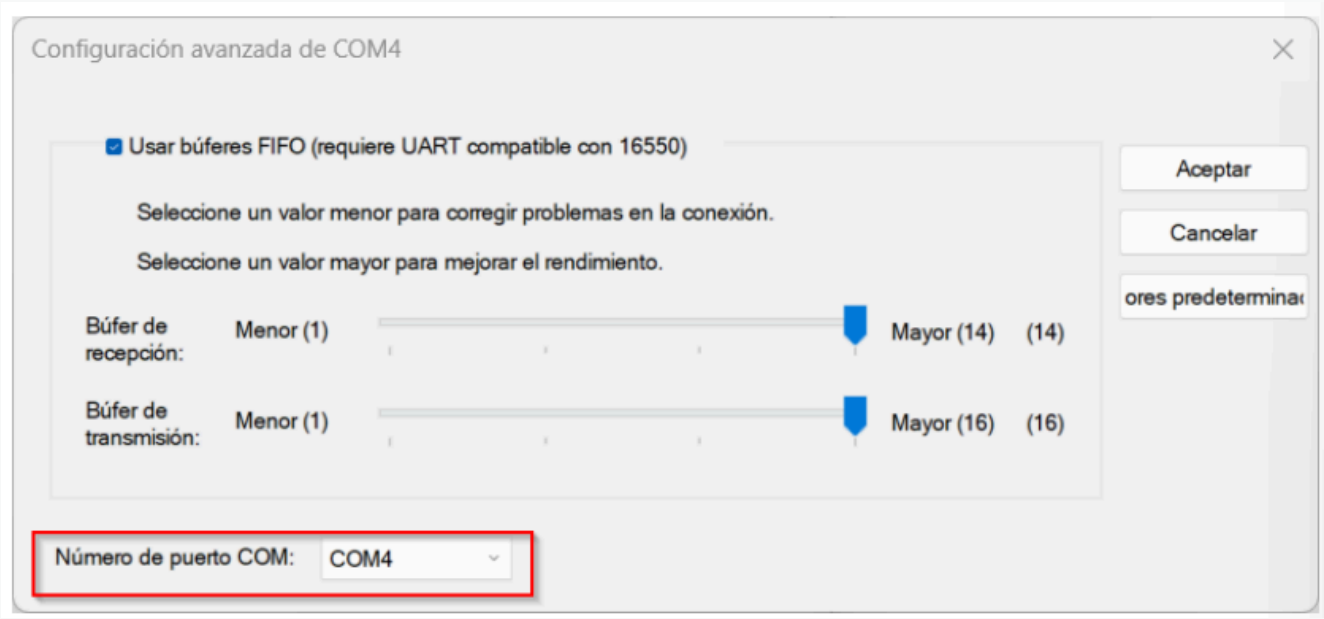
En el administrador de dispositivos señale con el botón secundario del mouse el dispositivo Pin Pad recién conectado, en la opción "puertos COM" y seleccione la opción propiedades del menú emergente.



Seleccione la pestaña configuración de puerto y opciones avanzadas...



En la ventana Configuración avanzada de COM vemos el puerto real que el sistema operativo esta asignando al dispositivo Pin Pad.



Su personal de TI (Sistemas) puede modificar estos valores de acuerdo con las necesidades o disponibilidad de los puertos de sus equipos. Lo más importante es que debe tener conocimiento del puerto real interno ya que este dato será necesario en la configuración del componente SDK.net [Conectividad por puerto COM].

Configuración de dispositivo Android Inalambrico (Wifi)

PAX A35

Existen algunos Pin Pad's que pueden operar mediante conectividad WIFI, es decir, se comunicarán con el desarrollo del comercio a través de su red inalámbrica en lugar del cable USB.

Nota 1: Solo para los dispositivos que entregue BBVA que soporten la conectividad Wifi deberán ser configurados mediante los pasos citados más adelante, en caso de que la conectividad sea mediante USB no podrá conectar los dispositivos mediante Wifi.

Nota 2: Debe conectar los dispositivos a la misma red que se encuentren conectados los puntos de venta o garantizar que se puede realizar una comunicación entre la caja y el Pin Pad a través de la red del comercio.

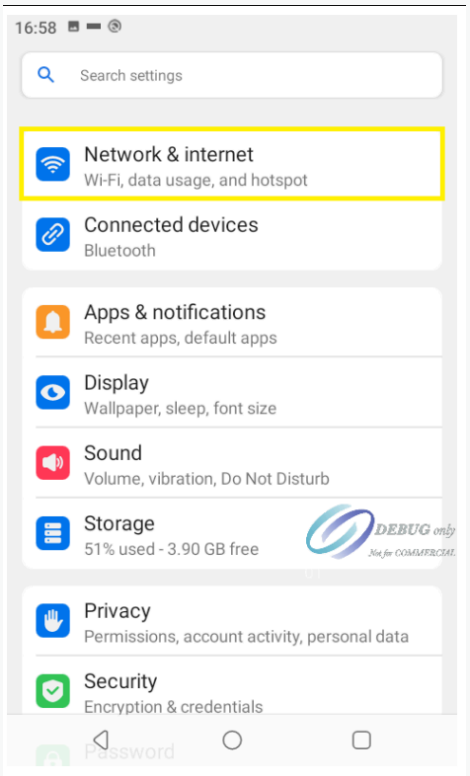
Nota 3: Sólo debe asociar un Punto de Venta a un Pin Pad inalámbrico, de lo contrario no se podrá garantizar el correcto funcionamiento del dispositivo lector de tarjetas.

Pin Pad Pax A35

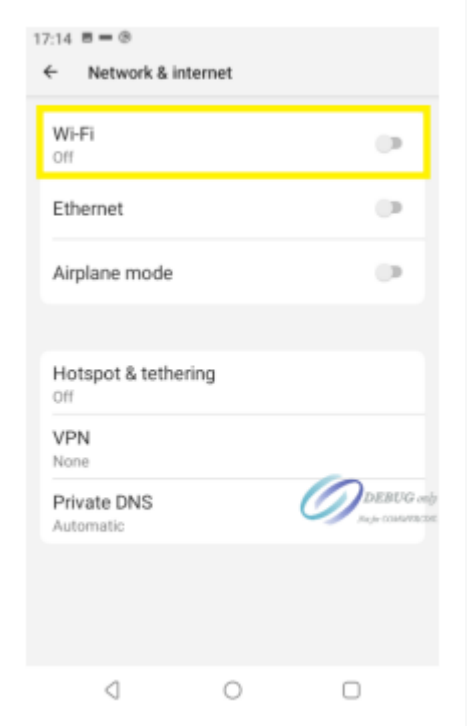
1. Para esta marca el modelo que se utiliza para transaccionar con la aplicación es el A35, para iniciar el pinpad conecte el dispositivo al puerto USB de su PC.
2. Una vez que termine de cargar observará la pantalla principal, en ella dar clic en el icono del engrane.

3. Introducir la clave (9876 o pax9876@@) y presionar el botón verde en el dispositivo.

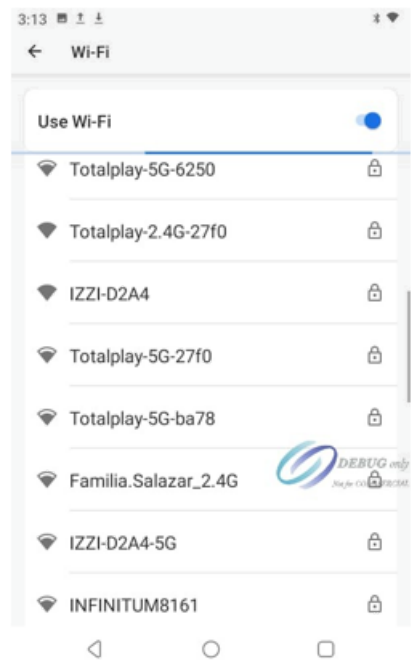
4. Seleccionar la opción "Network & Internet"



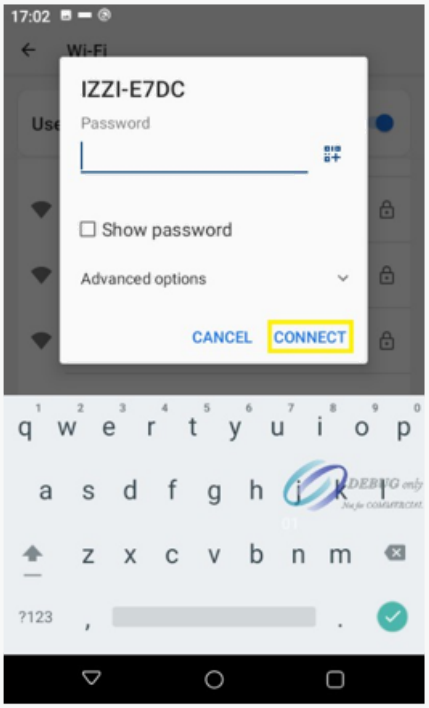
5. Seleccionar la opción "Wi-Fi" (en caso de que no esté habilitada dar clic en el botón switch del que se encuentra del lado derecho).



6. Seleccionar la red a la que se desea conectar.



7. Ingresar los datos de conexión y presionar la opción “CONNECT”



8. Regresar a la pantalla principal e iniciar la aplicación dando clic en el icono con la imagen de BBVA.

9. Si se realizó correctamente la configuración, observará, en la esquina inferior derecha, la IP y puerto asignados al dispositivo, mismos que deberá indicar en la configuración del SDK.

Una vez conectado el Pin Pad a la red del comercio y haber obtenido la IP asignada a éste, deberá introducir los siguientes parámetros en el objeto Configuracion que se describe más adelante:

- **PinPadConexion:** Deberá introducir el valor de la IP asignada para indicar que se utilizará un Pin Pad Wifi.
- **PinPadPuerto:** Generalmente es el 5000 pero puede variar si el puerto ya está siendo utilizado por el dispositivo.

Implementación

Objeto Configuración e inicialización

Es necesario realizar una configuración e inicialización del SDK previo al envío de transacciones, preferentemente al iniciar el punto de venta. Para ello se instancia un objeto de la clase Configuracion y se brindan los parámetros necesarios para la inicialización los cuales se enlistan a continuación:

Parámetros

- Logs: Nos indica si se tiene acceso a la descarga de un archivo de texto que permite conocer la comunicación y las acciones realizadas entre el Pin Pad y el Host autorizador.
- PinPadConexion: Es el puerto de conexión con el dispositivo Pin Pad:
  - Ejemplo: COM3, para dispositivos seriales
  - Ejemplo: 192.168.1.1 para dispositivos Wifi.
- PinPadPuerto: Indica el puerto en que el dispositivo Pin Pad escuchará los comandos (cuando la conexión sea por Wifi, si la conexión es serial se ignorará este valor).
- PinpadTimeout: Es el tiempo en segundos que el Pin Pad espera para realizar la lectura de una tarjeta antes de declinar la transacción.
- PinpadAndroid: Indica si el dispositivo Pin Pad cuenta con el sistema operativo Android.
- RutaBase: Indica la ruta donde se crearán los archivos que necesita el SDK para su operación. Por default la propiedad tiene el valor de vacío ("") y los archivos se generarán en la misma ruta del desarrollo que implemente el componente.
- HostUrl: Es la dirección URL del Host Autorizador donde se conectará TotalPOS SDK.
- HostTimeout: Es el tiempo máximo en segundos que se espera para recibir la respuesta de una transacción por parte del Host Autorizador. Por default se recomiendan 40 segundos.
- UrlProxy: Url del servidor proxy que se utilizará para la comunicación.
- PuertoProxy: Puerto del servidor proxy que se utilizará para la comunicación.
- Afiliacion: Este es el número afiliación default utilizado para enviar transacciones.
- MonedaAfiliacion: Moneda de la afiliación.
- ldAplicacion: Client-id para realizar la autenticación con el Host Autorizador.
- ClaveSecreta: Client-secret para realizar la autenticación con el Host Autorizador.
- NumeroTerminal: Identificador numérico de la terminal. En caso de que no se requiera la funcionalidad de cierre de turno, se debe asignar el valor 1 por defecto.

Una vez establecidos los parámetros se asigna el objeto configuración a la instancia de Interfaz de la siguiente forma:

Interfaz.getInstance().setConfiguracion(configuración)

Finalmente se inicializa el SDK mediante la siguiente instrucción:

Interfaz.getInstance().inicializar()

Es importante que se inicialice el SDK debido a que se inician los procesos en donde se recibe información acerca de la carga de llaves y dispara los reversos.

Ejemplo de configuración e inicialización en Windows (Pinpad USB)

```java
try {
    Configuracion configuracion = new Configuracion();
    configuracion.setLogs(false);

    configuracion.setPinpadAndroid(true);
    configuracion.setPinpadConexion("COM4");
    configuracion.setPinpadTimeOut("40");
    configuracion.setPinpadMensaje("BBVA");

    configuracion.setHostUrl("http://localhost:8080");
    configuracion.setHostTimeOut("30");

    configuracion.setAfiliacion("79791285");
    configuracion.setMonedaAfiliacion(MONEDA.SOLES);
    configuracion.setNumeroTerminal("001");

    configuracion.setIdAplicacion("8c678668c897df37c2ebe29d03cc");
    configuracion.setClaveSecreta("5921-e11c-4107-9813-4a9e72c0");

// Configuración opcional de Proxy
    configuracion.setUrlProxy("192.0.0.103");
    configuracion.setPuertoProxy("779");

    Interfaz.getInstance().setConfiguracion(configuracion);
    Interfaz.getInstance().inicializar();
} catch (PinPadKeysException ex) {
    System.out.println("Requiere carga de llaves");
} catch (PeticionException ex) {
    System.out.println("Error al iniciar SDK");
}
```

Ejemplo de configuración e inicialización en Linux y Mac (Pinpad USB)

```
try {
    Configuracion configuracion = new Configuracion();

    configuracion.setLogs(false);

    configuracion.setPinpadAndroid(true);
    configuracion.setPinpadConexion("/dev/tty...");
    configuracion.setPinpadTimeOut("40");
    configuracion.setPinpadMensaje("BBVA");

    configuracion.setHostUrl("http://localhost:8080");
    configuracion.setHostTimeOut("30");

    configuracion.setAfiliacion("79791285");
    configuracion.setMonedaAfiliacion(MONEDA.SOLES);
    configuracion.setNumeroTerminal("001");

    configuracion.setIdAplicacion("8c678668c897df37c2ebe29d03cc");
    configuracion.setClaveSecreta("5921-e11c-4107-9813-4a9e72c0");

// Configuración opcional de Proxy
    configuracion.setUrlProxy("192.0.0.103");
    configuracion.setPuertoProxy("779");

    Interfaz.getInstance().setConfiguracion(configuracion);
    Interfaz.getInstance().inicializar();
} catch (PinPadKeysException ex) {
    System.out.println("Requiere carga de llaves");
} catch (PeticionException ex) {
    System.out.println("Error al iniciar SDK");
}
```

Por default tendrán valor false el siguiente método, en caso de no asignar un valor:

- setLogs

Ejemplo de configuración e inicialización de Pinpad Wi-Fi

```java
try {
    Configuracion configuracion = new Configuracion();
    configuracion.setLogs(false);

    configuracion.setPinpadAndroid(true);
    configuracion.setPinpadConexion("192.168.68.125");
    configuracion.setPinpadPuerto("5000");
    configuracion.setPinpadTimeOut("40");
    configuracion.setPinpadMensaje("BBVA");

    configuracion.setHostUrl("http://localhost:8080");
    configuracion.setHostTimeOut("30");

    configuracion.setAfiliacion("79791285");
    configuracion.setMonedaAfiliacion(MONEDA.SOLES);
    configuracion.setNumeroTerminal("001");

    configuracion.setIdAplicacion("8c678668c897df37c2ebe29d03cc");
    configuracion.setClaveSecreta("5921-e11c-4107-9813-4a9e72c0");

// Configuración opcional de Proxy
    configuracion.setUrlProxy("192.0.0.103");
    configuracion.setPuertoProxy("779");

    Interfaz.getInstance().setConfiguracion(configuracion);
    Interfaz.getInstance().inicializar();
} catch (PinPadKeysException ex) {
    System.out.println("Requiere carga de llaves");
} catch (PeticionException ex) {
    System.out.println("Error al iniciar SDK");
}
```

Operaciones Retail

A continuación, se enlistan los métodos necesarios para realizar las transacciones de la funcionalidad Retail, los cuales están definidos dentro de una clase especial del tipo enum. Los tipos enum sirven para restringir el contenido de una variable a una serie de valores predefinidos.

- **OPERACION:** Objeto de tipo enum que contendrá las variables definidas relacionadas a los tipos de operación, los cuales se describen a continuación.

  - ✔ VENTA: Es la petición que realiza el comercio al host solicitando la autorización por el importe de la mercancía o servicio solicitando la tarjeta para su autorización.
  - ✔ VENTA_QR: Es la petición que realiza el comercio al host solicitando la autorización por el importe de la mercancía o servicio solicitando la lectura de un código QR para su autorización.
  - ✔ ANULACION_VENTA_QR: Es la petición que realiza el comercio al host autorizador para que se cancele el movimiento QR y se le restituya el importe.
  - ✔ ANULACION_ VENTA_TARJETA: Es la petición que realiza el comercio al host autorizador para que se cancele el movimiento y se le restituya el importe. A diferencia de la transacción anterior se solicita la tarjeta para realizar la operación.
  - ✔ CARGA_LLAVES: Esta operación es obligatoria ya que es de configuración para el Pin Pad, se debe realizar cada vez que el Pin Pad es nuevo y antes de enviar cualquier transacción, tiene tres modalidades las cuales se explican en el apartado de carga de llaves.

Los parámetros establecidos en cada operación son manejados de forma interna, de modo que ya están preestablecidos por el host autorizador.

Parámetros de operación

- **PARAMETRO_OPERACION:** Objeto de tipo enum que contendrá las variables definidas de manera interna con relación a los parámetros que tendrán las operaciones, dependiendo también el tipo de acción.

- ✔ IMPORTE
- ✔ PROPINA
- ✔ REFERENCIA_FINANCIERA

- **Importe:** Monto total de la venta.
- **Propina:** Importe de propina para una venta.
- **Referencia financiera:** Identificador único para una transacción realizada.

Venta

```
Map<PARAMETRO_OPERACION, Object> parametros = new HashMap<>();
parametros.put(PARAMETRO_OPERACION.IMPORTE, "100.00");
parametros.put(PARAMETRO_OPERACION.PROPINA, "10.00"); // Opcional

try {
    Peticion p = new Peticion();
(1) p.addOperationListener((OPERACION oldOpe, OPERACION newOpe) -> {
        System.out.println("Notify Change Operation " + oldOpe + " - " + newOpe);
    });
    p.setOperador("OPE001");
    p.setFecha(new SimpleDateFormat("yyyyMMddHHmmss").format(new Date()));

    p.setOperacion(OPERACION.VENTA, parametros);

    Respuesta response = p.autorizar();
    for (String key : response.keySet()) {
        System.out.println(key + " - " + response.get(key));
    }
} catch (PinPadKeysException ex) {
    System.out.println("Requiere carga de llaves");
} catch (PeticionException ex) {
    System.out.println("Error en operación");
}
```

(1) El método `addOperationListener` permite agregar un listener para saber cuándo la operativa cambia a `OPERACION.VENTA_QR` después de seleccionar la opción desde la pantalla del PinPad.

Venta QR

```java
Map<PARAMETRO_OPERACION, Object> parametros = new HashMap<>();
parametros.put(PARAMETRO_OPERACION.IMPORTE, "53.00");
parametros.put(PARAMETRO_OPERACION.PROPINA, "0.00"); // Opcional

try {
    Peticion p = new Peticion();
    p.setOperador("OPE001");
    p.setFecha(new SimpleDateFormat("yyyyMMddHHmmss").format(new Date()));

    p.setOperacion(OPERACION.VENTA_QR, parametros);

    Respuesta response = p.autorizar();
    for (String key : response.keySet()) {
        System.out.println(key + " - " + response.get(key));
    }
} catch (PinPadKeysException ex) {
    System.out.println("Requiere carga de llaves");
} catch (PeticionException ex) {
    System.out.println("Error en operación");
}
```

NOTA: La llamada al SDK para esperar la confirmación del pago QR no debe bloquear el hilo de la interfaz de usuario (UI Thread), por lo que se requiere estrictamente el uso de procesos asíncronos en Java (como Task).

Flujo de Cancelación y Expiración en Pagos con QR

Para garantizar una experiencia de usuario fluida, el sistema debe contemplar dos escenarios de interrupción de la venta:

- **Expiración de Tiempo en Pantalla:** El proceso de espera en el POS cuenta con una vigencia de 120 segundos, que inicia en el momento en que el código se despliega en pantalla. Si el pago no se confirma en este intervalo, la sesión de espera en la terminal finalizará y retornará un código de respuesta "99" indicando *"ESTADO INDETERMINADO, CONSULTAR ESTADO"*.

Para estos casos, donde el estado de la transacción es indeterminado, es mandatorio que la aplicación del comercio implemente una Consulta de Transacción para verificar el estado final de la venta antes de intentar un nuevo cobro. Esto evita duplicar la transacción.

- **Cancelación Manual (Interfaz de Usuario):** Es responsabilidad del comercio implementar un botón de *"Cancelar"* visible en la interfaz del punto de venta (POS) mientras se muestra el código QR. Si el cliente decide cambiar su método de pago o desistir de la compra, el operador debe poder activar esta función. Al hacerlo, el sistema debe invocar el método correspondiente del SDK para liberar la transacción y permitir que el POS regrese a su estado inicial de forma inmediata.

Ejemplo de Implementación

```
public void cancel() {
   p.finalizarOperacionQR();
}
```

Venta PSI

```
Map<PARAMETRO_OPERACION, Object> parametros = new HashMap<>();
parametros.put(PARAMETRO_OPERACION.IMPORTE, "120.20");

try {
    Peticion p = new Peticion();
    p.setOperador("OPE001");
    p.setFecha(new SimpleDateFormat("yyyyMMddHHmmss").format(new Date()));

    p.setOperacion(OPERACION.VENTA, parametros);

    Tarjeta tarjeta = p.leerTarjeta();

(1) if (tarjeta.getProducto().equals("C") && tarjeta.getMaxCuotasPsi() > 0) {
        p.setPromocionMeses(PROMOCION.MESES_SIN_INTERESES, 12);
    }

    Respuesta response = p.autorizar();
    for (String key : response.keySet()) {
        System.out.println(key + " - " + response.get(key));
    }
} catch (PinPadKeysException ex) {
    System.out.println("Requiere carga de llaves");
} catch (PeticionException ex) {
    System.out.println("Error en operación");
}
```

(1) Si el método `getMaxCuotasPsi` es mayor a cero significa que se le puede ofrecer a tarjetahabiente la promoción de pagos sin intereses.

Venta con cuotas

```
Map<PARAMETRO_OPERACION, Object> parametros = new HashMap<>();
parametros.put(PARAMETRO_OPERACION.IMPORTE, "120.20");

try {
    Peticion p = new Peticion();
    p.setOperador("OPE001");
    p.setFecha(new SimpleDateFormat("yyyyMMddHHmmss").format(new Date()));

    p.setOperacion(OPERACION.VENTA, parametros);

    Tarjeta tarjeta = p.leerTarjeta();

(1) if (tarjeta.isCuotasPci()) {
        p.setPromocionMeses(PROMOCION.MESES_CON_INTERESES, 12);
    }

    Respuesta response = p.autorizar();
    for (String key : response.keySet()) {
        System.out.println(key + " - " + response.get(key));
    }
} catch (PinPadKeysException ex) {
    System.out.println("Requiere carga de llaves");
} catch (PeticionException ex) {
    System.out.println("Error en operación");
}
```

(1) Al ofrecer cuotas o PSI es necesario validar que la tarjeta sea de crédito.

Anulación de Venta QR

```java
Map<PARAMETRO_OPERACION, Object> parametros = new HashMap<>();
parametros.put(PARAMETRO_OPERACION.IMPORTE, "53.00");
parametros.put(PARAMETRO_OPERACION.REFERENCIA_FINANCIERA,  "123456789012");

try {
    Peticion p = new Peticion();
    p.setOperador("OPE001");
    p.setFecha(new SimpleDateFormat("yyyyMMddHHmmss").format(new Date()));

    p.setOperacion(OPERACION.ANULACION_VENTA_QR, parametros);

    Respuesta response = p.autorizar();
    for (String key : response.keySet()) {
        System.out.println(key + " - " + response.get(key));
    }
} catch (PeticionException ex) {
    System.out.println("Error en operación");
}
```

Anulación de Venta

```
Map<PARAMETRO_OPERACION, Object> parametros = new HashMap<>();
parametros.put(PARAMETRO_OPERACION.IMPORTE, "100");
parametros.put(PARAMETRO_OPERACION.REFERENCIA_FINANCIERA,  "123456789012");

try {
    Peticion p = new Peticion();
    p.setOperador("OPE001");
    p.setFecha(new SimpleDateFormat("yyyyMMddHHmmss").format(new Date()));

    p.setOperacion(OPERACION.ANULACION_VENTA_TARJETA, parametros);

    Respuesta response = p.autorizar();
    for (String key : response.keySet()) {
        System.out.println(key + " - " + response.get(key));
    }
} catch (PinPadKeysException ex) {
    System.out.println("Requiere carga de llaves");
} catch (PeticionException ex) {
    System.out.println("Error en operación");
}
```

Consulta de transacción

```
try {
    Peticion p = new Peticion();
    p.consultarTransaccion("f3b29b92-25b2-4527-9b5d-d99ed19b138f");

    Respuesta response = p.autorizar();
    for (String key : response.keySet()) {
        System.out.println(key + " - " + response.get(key));
    }
} catch (PeticionException ex) {
    System.out.println("Error en operación");
}
```

Operaciones Pre-autorización

A continuación, se enlistan los métodos necesarios para realizar las transacciones de la funcionalidad Pre-autorización, los cuales están definidos dentro de una clase especial del tipo enum. Los tipos enum sirven para restringir el contenido de una variable a una serie de valores predefinidos.

- **OPERACIÓN:** Objeto de tipo enum que contendrá las variables definidas relacionadas a los tipos de operación, los cuales se describen a continuación:
  - ✔ **PREAUTORIZACIÓN**: Es la petición que realiza el comercio al host para apartar una cantidad del saldo disponible de su cliente para asegurar que la tarjeta cuenta con lo necesario para cubrir con los productos o servicios que desea adquirir. Esta operación es utilizada para garantizar las reservaciones; no es una operación contable, no se hace el cobro, solo se retiene la cantidad solicitada.
  - ✔ **CONSULTA:** Es la petición que realiza el comercio al host autorizador para que obtenga el listado de pre-autorizaciones pendientes de confirmación.
  - ✔ **CONFIRMACIÓN**: Es la petición que realiza el comercio al host autorizador para cerrar o "liberar" la pre-autorización, ya sea enviando el mismo importe, importe menor o mayor al de la pre-autorización.
  - ✔ **ANULACIÓN**: Es la petición que realiza el comercio al host autorizador para que se cancele el movimiento y se le restituya el importe.

Los parámetros establecidos en cada operación son manejados de forma interna, de modo que ya están preestablecidos por el host autorizador.

Parámetros de operación

- **PARAMETRO_OPERACION:** Objeto de tipo enum que contendrá las variables definidas de manera interna con relación a los parámetros que tendrán las operaciones, dependiendo también el tipo de acción.
  - IMPORTE
  - FOLIO
  - CONFIRMACION_IMPORTE
  - REFERENCIA_FINANCIERA

- **Importe:** Importe total de la pre-autorización.
- **Folio:** Identificador Dato alfanumérico de siete posiciones con el cual se asociará la tarjeta del tarjetahabiente mientras dure la retención. Por lo general se utiliza para el número de reservación.
- **Confirmación de importe:** Importe total por el cual se hará la confirmación de Pre-autorización.
- **Referencia financiera:** Identificador único para una transacción realizada.

Pre-autorización

```
Map<PARAMETRO_OPERACION, Object> parametros = new HashMap<>();
    parametros.put(PARAMETRO_OPERACION.IMPORTE, "90.99");
    parametros.put(PARAMETRO_OPERACION.FOLIO, "001");

    try {
        Peticion p = new Peticion();
        p.setOperador("OPE001");
        p.setFecha(new SimpleDateFormat("yyyyMMddHHmmss").format(new Date()));
        p.setOperacion(OPERACION.PREAUTORIZACION, parametros);

        Respuesta response = p.autorizar();

        for (String key : response.keySet()) {
            System.out.println(key + " - " + response.get(key));
        }
    } catch (PinPadKeysException ex) {
        System.out.println("Requiere carga de llaves");
    } catch (PeticionException ex) {
        System.out.println("Error en operación");
    }
```

Consulta de Pre-autorizaciones

```
try {
        Peticion p = new Peticion();
        RespuestaPreautorizaciones response =
      p.consultarPreautorizaciones("20251029000000", "20251029235959");

        for (Preautorizacion item : response.getPreautorizacion()) {
            System.out.println(item.toString());
        }
    } catch (PinPadKeysException ex) {
        System.out.println("Requiere carga de llaves");
    } catch (PeticionException ex) {
        System.out.println("Error en operación");
    }
```

Confirmación pre-autorización

```
Map<PARAMETRO_OPERACION, Object> parametros = new HashMap<>();
    parametros.put(PARAMETRO_OPERACION.IMPORTE, "90.99");
    parametros.put(PARAMETRO_OPERACION.CONFIRMACION_IMPORTE, "90.99");
    parametros.put(PARAMETRO_OPERACION.FOLIO, "A01");

    try {
        Peticion p = new Peticion();
        p.setOperador("OPE001");
        p.setFecha(new SimpleDateFormat("yyyyMMddHHmmss").format(new Date()));
        p.setOperacion(OPERACION.CONFIRMACION_PREAUTORIZACION, parametros);

        Respuesta response = p.autorizar();

        for (String key : response.keySet()) {
            System.out.println(key + " - " + response.get(key));
        }
    } catch (PinPadKeysException ex) {
        System.out.println("Requiere carga de llaves");
    } catch (PeticionException ex) {
        System.out.println("Error en operación");
    }
```

Anulación de Pre-autorización

```
Map<PARAMETRO_OPERACION, Object> parametros = new HashMap<>();
    parametros.put(PARAMETRO_OPERACION.IMPORTE, "90.99");
    parametros.put(PARAMETRO_OPERACION.REFERENCIA_FINANCIERA, "REF001");
    try {
        Peticion p = new Peticion();
        p.setOperador("CAJA01");
        p.setFecha(new SimpleDateFormat("yyyyMMddHHmmss").format(new Date()));
        p.setOperacion(OPERACION.ANULACION_PREAUTORIZACION, parametros);
        Respuesta response = p.autorizar();
        for (String key : response.keySet()) {
            System.out.println(key + " - " + response.get(key));
        }
    } catch (PeticionException e) {
        System.out.println("Error en operación");
    }
```

Cierre de turno

Permite el comercio finalizar el turno activo de una terminal enviando la solicitud correspondiente al host autorizador y obteniendo la información de las operaciones realizadas durante el turno Esta operación es utilizada para cerrar el turno actual y consultar los reportes asociados.

Las operaciones de cierre de turno no requieren parámetros adicionales enviados por el comercio ya que la información necesaria es obtenida de forma interna por el SDK y el autorizador.

Una vez realizado el cierre de turno se genera el reporte el cual permite consultar la información del turno cerrado.

Este reporte permite obtener un resumen general del cierre de turno incluyendo totales y datos de las operaciones realizadas para generar este reportes necesario proporcionar el identificador del turno (idTurno)

- **Operación:** Objeto de tipo en enum que contiene las variables definidas para los distintos tipos de operación. Requiere el siguiente parámetro
  - ✔ **idTurno:** identificador del turno a consultar datos alfanumérico de entre 1 y cuatro caracteres

Ejemplo de implementación de cierre de turno

```
try {
            Peticion p = new Peticion();
            p.setOperador("OPE001");
            p.setFecha(new SimpleDateFormat("yyyyMMddHHmmss").format(new Date()));

            p.setOperacion(OPERACION.CIERRE_TURNO, null);

            Respuesta response = p.autorizar();

            for (String key : response.keySet()) {
                System.out.println(key + " - " + response.get(key));
            }

        } catch (PinPadKeysException ex) {
            System.out.println("testException: " + ex.getMessage());
        } catch (PeticionException ex) {
            System.out.println("testException: " + ex.getMessage());
        }
```

Carga de llaves

Existen 3 escenarios a considerar para generar la carga de llaves los cuales se detallan a continuación, se deben desarrollar los tres escenarios.

1. **Carga de Llave Manual:** Esta transacción es para cargar la llave que el Pin Pad utiliza para cifrar la información sensible de las tarjetas y sin la cual el sistema no puede hacer cobros con tarjetas, esta carga de llave será generada por el comercio mediante un mensaje que más adelante se detalla al igual que su flujo. Esta operación sólo se deberá realizar cuando se configura el equipo por primera vez y cuando la interfaz lo indique.

2. **Carga de Llaves al inicializar:** Esta operación se realiza cuando el host autorizador determina que es necesario mandar una carga de llaves automática la próxima vez que se inicialice el SDK de TotalPos. Por lo que se almacena una bandera de forma interna y cuando se manda a llamar el método inicializar de la instancia Interfaz de forma automática se realiza la carga de llaves.

3. **Carga de Llaves en Automático:** Esta operación la realiza la interfaz en automático cuando el host autorizador indica que hay que realizar carga de llaves, la interfaz regresa la respuesta de la transacción que solicitó para que el punto de venta imprima el voucher y de forma simultánea la interfaz también está realizando la carga de llaves, el resultado de esta operación se refleja en la pantalla del Pin Pad.

Carga de llaves manual

```
try {
    Peticion p = new Peticion();
    p.setOperador("OPE001");
    p.setFecha(new SimpleDateFormat("yyyyMMddHHmmss").format(new Date()));

    p.setOperacion(OPERACION.CARGA_LLAVES, null);

    Respuesta response = p.autorizar();
    for (String key : response.keySet()) {
        System.out.println(key + " - " + response.get(key));
    }
} catch (PinPadKeysException ex) {
    System.out.println("Requiere carga de llaves");
} catch (PeticionException ex) {
    System.out.println("Error en operación");
}
```

Reversos

Cuando se envía una transacción y no se logra recuperar la respuesta del servidor, la interfaz genera Un reverso y lo guarda en una base de datos local de forma cifrada Dicho reverso es enviado al servidor para no generar un doble cargo por parte de la interfaz de forma automática. Esto se logra únicamente llamando el método de inicialización de total POS SDK

```
Interfaz.getInstance().inicializar();
```

Con esta instrucción se permite disparar los reversos tal y como se especifica en el apartado “Objeto Configuración”.

Reportes

Es necesario realizar una configuración e inicialización del SDK previo a la generación de los reportes preferentemente al iniciar el punto de venta. Para ello se instancia un objeto de la clase ***“Configuración”*** y se brindan los parámetros necesarios para la inicialización (ver apartado ***“Objeto Configuración e Inicialización”***).

Reporte Firma Física

Para generar el reporte de Música se deberán proporcionar los siguientes parámetros a la función:

- ✔ **Moneda:** Indica la moneda de la cual se desea obtener el reporte (Soles o Dólares)

```
try {
        RespuestaReporteFirma response =
Reporteria.generarReporteFirma(MONEDA.SOLES);

        for (String key : response.keySet()) {
            System.out.println(key + " - " + response.get(key));
        }
    } catch (PeticionException ex) {
        System.out.println("Error en operación");
    }
```

Reporte Firma Totalizado

Para generar el reporte de Música se deberán proporcionar los siguientes parámetros a la función:

- ✔ **Moneda:** Indica la moneda de la cual se desea obtener el reporte (Soles o Dólares)

```
try {
        RespuestaReporteTotalizado response =
Reporteria.generarReporteTotalizado(MONEDA.SOLES);

        for (String key : response.keySet()) {
            System.out.println(key + " - " + response.get(key));
        }
    } catch (PeticionException ex) {
        System.out.println("Error en operación");
    }
```

Reporte Firma Detallado

Para generar el reporte de Música se deberán proporcionar los siguientes parámetros a la función:

- ✔ **Moneda:** Indica la moneda de la cual se desea obtener el reporte (Soles o Dólares)

```
try {
        RespuestaReporteDetallado response =
Reporteria.generarReporteDetallado(MONEDA.SOLES);

        for (String key : response.keySet()) {
            System.out.println(key + " - " + response.get(key));
        }
    } catch (PeticionException ex) {
        System.out.println("Error en operación");
    }
```

Reporte de Cierre de Turno

Para generar el reporte de cierre de turno se deberán hacer las funciones:

Ejemplo de Reporte Operaciones

```
try {
    RespuestaReporteOperaciones response =
Reporteria.generarReporteCierreOperaciones("123",”12345678”);

            for (String key : response.keySet()) {
                System.out.println(key + " - " + response.get(key));
            }
        } catch (PeticionException ex) {
            System.out.println("testException: " + ex.getMessage());
        }
```

Ejemplo de Reporte Resumen

```
try {
            RespuestaReporteResumen response =
Reporteria.generarReporteCierreResumen("123",”12345678”);

            for (String key : response.keySet()) {
                System.out.println(key + " - " + response.get(key));
            }
        } catch (PeticionException ex) {
            System.out.println("testException: " + ex.getMessage());
        }
```

Ejemplo de Reimprimir Reporte Operaciones

Este reporte detalla cada una de las transacciones realizadas durante el último turno cerrado

```
try {
    RespuestaReporteOperaciones response =
Reporteria.reimprimirReporteCierreOperaciones(”12345678”);

            for (String key : response.keySet()) {
                System.out.println(key + " - " + response.get(key));
            }
        } catch (PeticionException ex) {
            System.out.println("testException: " + ex.getMessage());
        }
```

Ejemplo de Reimprimir Reporte Resumen

Este reporte genera un consolidado totalizado de las operaciones del último turno cerrado

```
try {
            RespuestaReporteResumen response =
Reporteria.reimprimirReporteCierreResumen(”12345678”);

            for (String key : response.keySet()) {
                System.out.println(key + " - " + response.get(key));
            }
        } catch (PeticionException ex) {
            System.out.println("testException: " + ex.getMessage());
        }
```

Respuesta

Una vez que una transacción fue ejecutada, vamos a poder obtener la información necesaria para realizar la impresión de los vouchers a través de los siguientes métodos que se proporcionan, los cuales están dentro de la clase Respuesta.

Los datos contenidos en la respuesta varían dependiendo de la transacción realizada. Si la respuesta no contiene alguna propiedad el valor por default devuelto será nu/1.

Para obtener el valor de la propiedad es necesario utilizar el método get seguido del nombre de la propiedad (Para más información ver la descripción de la clase Respuesta).

PROPIEDAD	VOUCHER	DESCRIPCION
leyenda	Mandatorio	Respuesta del emisor.
codigoRespuesta		Regresa el código de autorización (2 dígitos) proporcionado por el banco emisor. Ver Tabla 2 (Códigos de respuesta)
idTransaccion		Identificador único generado para la transacción realizada.
fechaHora		Fecha y hora de la transacción.
operador		Operador que envía la operación.
firma	Mandatorio	Indica si la firma es autógrafa, electrónica o pago rápido.
afiliacion		Este número lo proporciona BBVA y está formado por 8 números.
afiliacionAmex		Uso futuro
moneda	Mandatorio	Moneda con la que se realiza la operación.
direccion		Dirección del comercio.
razonSocial		Razón social del comercio.
numeroTarjeta	Mandatorio	Contiene el número de la tarjeta
modoLectura		Contiene los valores: • 01 cuando la tarjeta fue digitada • 03 QR • 05 para tarjetas insertadas (Chip) • 07 para tarjetas contactless (Chip) • 80 cuando fue por error en chip (full back) • 90 cuando se deslizó la tarjeta (banda) • 91 para tarjetas contactless (banda)
tarjetahabiente		Nombre del tarjetahabiente.
productoTarjeta		Indica si la tarjeta es de crédito o débito.
emisorTarjeta		Indica el emisor de la tarjeta.
criptogramaTarjeta		Esta propiedad contiene información cuando la transacción se realizó con una tarjeta chip
idAplicacionTarjeta		Esta propiedad debe de estar presente en el ticket cuando la transacción se realizó con una tarjeta con chip.
aplicacionTarjeta		Etiqueta asociada a la tarjeta
codigoPromocion		Código del tipo de promoción, verificar en tabla correspondiente.
cuotas		Número de meses para dividir el cargo.

financiamiento		Numero de meses para comenzar el pago.
montoCuota		Monto devuelto por el emisor que refleja el monto de cada cuota
mensajes		Mensajes de texto devueltos por el emisor en una transacción de cuotas
importe	Mandatorio	Indica el importe de la transacción
autorizacion	Mandatorio	Regresa el número de autorización (6 dígitos) proporcionado por el banco emisor.
codigoTransaccion		Código de transacción del requerimiento, identifica el tipo de operación que será procesada.
nombreTransaccion		Indica el tipo de transacción que se realizó.
propina		Indica la propina de la transacción
secuenciaTransaccion		Indica el número de transacción
referenciaFinanciera		Identificador de la transacción utilizado para realizar una anulación de la venta.
serieTerminal		Número de serie del equipo que envía la operación.
idBilletera		Indica el Id de la billetera con la que se realizó un pago con QR.
Billetera		Nombre de la billetera con la que se realizó un pago con QR.
Folio		Identificador para hacer una confirmación o una anulación de pre-autorización.
idTurno		Número de turno para el cual se generará el resumen
numeroTerminal		Número de la terminal de una afiliación o comercio registrado.

Catálogos

Moneda

Catálogo de códigos para identificar el tipo de moneda de la afiliación.

Constante
DOLARES
SOLES

Firma

Catálogo de códigos para identificar el tipo de firma.

Constante
AUTOGRAFA
ELECTRONICA
SIN_FIRMA
SIN_LEYENDA

Códigos de Respuesta

CÓDIGO	LEYENDA	DESCRIPCIÓN
00	Aprobada XXXXXX	Tarjeta autorizada por el emisor
01	Llame al adquiriente	Consulte al emisor de la tarjeta
03	Terminal inválida	Este mensaje se genera por que la afiliación no existe o no están bien definidos sus atributos.
04	Retenga y llame	Mensaje que indica que la tarjeta está reportada como extraviada
05	Declinada	El emisor declina la solicitud de autorización por algún problema en la tarjeta.
10	Aprobación Parcial XXXXXXX	Aprobación parcial
12	Transacción inválida	Transacción inválida (Fallback) / PostAutorización sin una Pre-Autorización.
13	Cantidad inválida	Importe de una transacción que no cumple con lo esperado.
14	Tarjeta inválida	Tarjeta inválida
19	Reintente	Mensaje generado por problemas de comunicación
30	Error en formato	Error de formato
33	Tarjeta expirada	Tarjeta vencida o expirada
40	Servicio no disponible	Función no soportada
41	Retenga y llame	Recoger tarjeta
43	Retenga y llame	Recoger tarjeta
45	Promoción no permitida	La afiliación no está permitida para realizar operaciones de promociones.
47	Transacción no realizada acuda a sucursal	Transacción no realizada por haber excedido su límite permitido.
48	Ingrese cód. Seg.	Se requiere que sea ingresado el código de seguridad o cvv2 que aparece en la parte posterior de las tarjetas
49	Cod.de.seg.erroneo	Se ingresó un código de seguridad o cvv2 erróneo.
50	Ha superado el núm. de txns. rechazadas	Cuando se excedió el número de intentos en la parametría establecida para operaciones.
51	Fondos insuficientes	No cuenta con el disponible suficiente para amparar la venta.
53	Cuenta inexistente	Cuenta inexistente
55	Pin incorrecto	Nip incorrecto
58	Operacion invalida	La afiliación no tiene definidos los atributos para permitir este tipo de operación.
62	Tarjeta No Permitida	Tarjeta No Permitida.
65	Retiro cash excedido	Intentos De Retiros excedido.
71	Debe inicializar PIN PAD	El host determina que el dispositivo debe cargar llaves por primera vez o renovarlas
72	Reintente – Llamar a Soporte	Problema inicializando Llaves
73	Error en el Sistema – Llamar a Soporte	La caja o aplicación del comercio, alteraron o truncaron los datos cifrados, entregados por el dispositivo
75	PIN Incorrecto	Número de intentos de NIP excedidos
76	Cuenta bloqueada	Cuenta bloqueada

Descripción de Clases

Clase Tarjeta

RETORNO	MÉTODO	DESCRIPCIÓN
String	getPan()	Número de tarjeta.
String	getMaxCuotasPsi()	Número máximo de cuotas sin intereses que se puede ofrecer al tarjetahabiente
String	getProducto()	Si se tiene registrado retorna el producto de la tarjeta
int	isCuotasPci()	Indicador para poder ofrecer cuotas con intereses al tarjetahabiente.

Clase Interfaz

RETORNO	MÉTODO	DESCRIPCIÓN
Configuración	getConfiguracion()	Instancia de configuración.
void	setConfiguracion(Configuracion configuracion)	
static Interfaz	getInstance()	Obtiene instancia de Interfaz.
Info	getPinPadInfo()	Recupera la versión, modelo, número de serie y actualización del Pin Pad.
void	inicializar()	Inicializa el componente SDK.

Clase Configuración

RETORNO	MÉTODO	DESCRIPCIÓN
String	getBasePath()	Ruta base de archivos.
void	setBasePath (String basePath)	
String	getClaveSecreta()	Número secreto del comercio que autoriza realizar transacciones.
void	setClaveSecreta(String claveSecreta)	
String	getAfiliacion()	Indica la afiliación del comercio
void	setAfiliacion(String afiliacion)	

MONEDA void	getMonedaAfiliacion() setMonedaAfiliacion(MONEDA moneda)	Indica la moneda de la afiliación del comercio
String void	getHostTimeOut() setHostTimeOut(String hostTimeOut)	Tiempo que el componente espera una respuesta del Host.
String void	getHostUrl() setHostUrl(String hostUrl)	Dirección IP o URL a la que el componente enviará las transacciones.
String void	getIdAplicacion() setIdAplicacion(String idAplicacion)	Identificador único del comercio que autoriza realizar transacciones.
String void	getPinpadConexion() setPinpadConexion(String pinpadConexion)	Indica el tipo de conexión del PinPad.
String void	getPinpadMensaje() setPinpadMensaje(String pinpadMensaje)	Mensaje que se muestra en la pantalla del PinPad.
String void	getPinpadTimeOut() setPinpadTimeOut(String pinpadTimeOut)	Equivale al tiempo en segundos que el PinPad espera una respuesta.
String void	getUrlProxy () setUrlProxy (String urlProxy)	URL del servidor proxy que se desea utilizar.
String void	getPuertoProxy () setPuertoProxy (String puertoProxy)	Puerto del servidor proxy que se desea utilizar.
boolean void	getPinpadAndroid () setPinpadAndroid (boolean pinpadAndroid)	Indica si el dispositivo Pin Pad cuenta con el sistema operativo Android.
boolean void	isLogs() setLogs(boolean logs)	Indica si es necesario generar logs o no.
String void	getPinpadPuerto() setPinpadPuerto(String urlProxy)	Indica el puerto en que el dispositivo Pin Pad escuchará los comandos (cuando la conexión sea por Wifi, si la conexión es serial se ignorará este valor).
String void	getNumeroTerminal() setNumeroTerminal(String terminal)	Identificador numérico de la terminal. En caso de que no se requiera la funcionalidad de cierre de turno, se debe asignar el valor 1 por defecto.

Clase Petición

RETORNO	MÉTODO	DESCRIPCIÓN
Respuesta	autorizar()	Solicita la autorización de la operación.
Respuesta	consultarTransaccion(String idTransaccion)	Consulta el estatus de una transaccion.
void	finalizarOperacionQR()	Método para dejar de mostrar el QR en el dispositivo.
void	finalizarLecturaTarjeta(String codigoRespuesta, String numeroAutorizacion, String datosAutenticacion, READING_MODE modoLectura)	Método para finalizar una transacción en curso con bines de excepción.
Tarjeta	leerTarjeta()	Lee tarjeta cargo del PinPad.
void	setFecha(String fechaLocal)	Fecha local en la que se realiza la operación, con un formato de año, mes, día, hora, minutos y segundos. (AAAAMMDDHHMMSS)
void	setOperacion(OPERACION codigoOperacion, Map<PARAMETRO_OPERACION,Object> params)	Agrega la operación a realizar.
void	setOperador(String operador)	Identifica operador que realiza la operación.
void	setPromocionMeses(PROMOCION codigo, int cuotas)	Indica el plan para diferir el pago.
void	sincronizacionFinal()	Cancelar transacción y finalizar comunicación con pinpad.
void	abortarTransaccion()	Aborta la transacción en curso, evitando que se envíe al host.
void	addOperationListener(OperationListener listener)	Asigna un listener para eventos de la clase *Peticion*.
RespuestaPreautorizaciones	consultarPreautorizaciones(String fechaInicial, String fechaFinal)	Consulta las pre-autorizaciones realizadas en un rango de fechas.
void	finalizarOperacionQR()	Finaliza la operación de venta con QR, deteniendo la visualización en el dispositivo.

Clase Respuesta

RETORNO	MÉTODO	DESCRIPCIÓN
String	getAfiliacion()	Este número lo proporciona BBVA y está formado por 8 números.
String	getAfiliacionAmex()	Uso futuro
String	getAplicacionTarjeta()	Esta propiedad debe de estar presente en el ticket cuando la transacción se realizó con una tarjeta con chip.

String	getAutorizacion()	Regresa el número de autorización (6 dígitos) proporcionado por el emisor.
OPERACION	getCodigoOperacion()	Código de transacción del requerimiento, identifica el tipo de operación que será procesada.
PROMOCION	getCodigoPromocion()	Código del tipo de promoción, verificar en tabla correspondiente.
String	getCodigoRespuesta()	Regresa el código de respuesta (2 dígitos) proporcionado por el emisor
String	getCriptogramaTarjeta()	Esta propiedad contiene información cuando la transacción se realizó con una tarjeta chip.
String	getDireccion()	Dirección del comercio.
String	getEmisorTarjeta()	Banco emisor de la tarjeta.
String	getFechaHora()	Fecha y hora de la transacción.
String	getFinanciamiento()	Número de meses para financiar el cargo.
FIRMA	getFirma()	Indica si la firma es autógrafa, electrónica o pago rápido.
String	getIdAplicacionTarjeta()	Id de aplicación de la tarjeta
String	getIdTransaccion()	Id de la transacción.
String	getImporte()	Monto por el cual se realizará la solicitud.
String	getPropina()	Propina de la transacción
String	getLeyenda()	Respuesta del emisor.
String	getModoLectura()	Contiene los valores de 05 para tarjetas insertadas (Chip), 90 cuando se deslizó la tarjeta (banda), 80 cuando fue por error en chip (full back).
MONEDA	getMoneda()	Moneda de la solicitud.
String	getMontoCuota()	Monto devuelto por el emisor que refleja el monto de cada cuota
String	getNombreTransaccion()	Indica el nombre de la transacción realizada.
String	getNumeroTarjeta()	Contiene el número de la tarjeta.
String	getOperador()	Operador que envía la operación.
String	getCuota()	Número de meses a dividir el cargo.
String	getProductoTarjeta()	Esta propiedad debe de estar presente en el ticket cuando la transacción se realizó con una tarjeta con chip.
String	getPropina()	Solo se utiliza para operaciones en donde se ingresa propina su formato es como el del campo importe.

String	getRazonSocial()	Razón social del comercio.
String	getReferenciaFinanciera()	Durante el proceso de una operación de anulación, es necesario que éste dato concuerde con el enviado durante el proceso de venta, en caso de no coincidir con el de la venta la operación de cancelación y de post propina será denegada.
String	getSecuenciaTransaccion()	Durante el proceso de una operación de anulación, es necesario que éste dato concuerde con el enviado durante el proceso de venta.
String	getSerieTerminal()	Número de serie del Pinpad.
String	getTarjetahabiente()	Contiene el nombre del tarjetahabiente.
String	getIdBilletera()	Indica el Id de la billetera con la que ser realizó un pago con QR.
String	getBilletera()	Nombre de la billetera con la que se realizó un pago con QR.
String	getFolio()	Identificador para hacer una confirmación o una anulación de pre-autorización.
String	getIdTurno()	Identificador único generado para la transacción realizada.
String	getNumeroTerminal()	Contiene el número de Terminal

Clase PeticionException

La excepción es generada cuando genera cualquier error dentro del SDK.

RETORNO	MÉTODO	DESCRIPCIÓN
String	getMessage()	Recupera el mensaje de error.

Clase PinPadKeysException

La excepción es generada cuando el PinPad requiere una carga de llaves.

RETORNO	MÉTODO Y DESCRIPCIÓN
String	getMessage() Recupera el mensaje de error generado.

Clase Reporteria

Retorno	Método	DESCRIPCIÓN
RespuestaReporteDetallado	generarReporteDetallado(MONEDA moneda)	Obtiene el reporte detallado
RespuestaReporteFirma	generarReporteFirma(MONEDA moneda)	Obtiene el reporte de firma física.
RespuestaReporteTotalizado	generarReporteTotalizado(MONEDA moneda)	Obtiene el reporte totalizado.
RespuestaReporteResumen	generarReporteCierreResumen(String idTurno, String afiliacion)	Obtiene el reporte Resumen
RespuestaReporteOperaciones	generarReporteCierreOperaciones(String idTurno, String afiliacion)	Obtiene el reporte Operaciones
RespuestaReporteResumen	reimprimirReporteCierreResumen(String afiliacion)	Este reporte genera un consolidado totalizado de las operaciones del último turno cerrado.
RespuestaReporteOperaciones	reimprimirReporteCierreOperaciones(String afiliacion)	Este reporte genera un consolidado totalizado de las operaciones del último turno cerrado.

Clase RespuestaReporteFirma

Tipo de Dato	Propiedad	DESCRIPCIÓN
string	getOperaciones()	Número total de transacciones que requirieron firma autógrafa
string	getTotalOperaciones()	Importe totalizado de las transacciones que requirieron firma autógrafa
string	getOperacionesVisa()	Número total de transacciones pagadas con tarjeta Visa y que requirieron firma autógrafa
string	getTotalOperacionesVisa()	Importe totalizado de las transacciones pagadas con tarjeta Visa y que requirieron firma autógrafa
string	getOperacionesMc()	Número total de transacciones pagadas con tarjeta Master Card y que requirieron firma autógrafa
string	getTotalOperacionesMc()	Importe totalizado de las transacciones pagadas con tarjeta Master Card y que requirieron firma autógrafa
string	getOperacionesAmex()	Número total de transacciones pagadas con tarjeta Amex y que requirieron firma autógrafa
string	getTotalOperacionesAmex()	Importe totalizado de las transacciones pagadas con tarjeta Amex y que requirieron firma autógrafa
string	getOperacionesDiners()	Número total de transacciones pagadas con tarjeta Diners y que requirieron firma autógrafa
string	getTotalOperacionesDiners()	Importe totalizado de las transacciones pagadas con tarjeta Diners y que requirieron firma autógrafa

Clase RespuestaReporteTotalizado

Tipo de Dato	Propiedad	DESCRIPCIÓN
string	getOperacionesConf()	Número total de transacciones confirmadas
string	getTotalOperacionesConf()	Importe totalizado de las transacciones confirmadas
string	getVenta()	Número total de ventas
string	getTotalVenta()	Importe totalizado de las ventas
string	getPreautorizacionesConf()	Número total de pre-autorizaciones confirmadas
string	getTotalPreautorizacionesConf()	Importe totalizado de las preautorizaciones confirmadas
string	getAnulacion()	Número total anulaciones
string	getTotalAnulaciones()	Importe totalizado de las anulaciones
string	getPuntos()	Número total de transacciones con puntos
string	getTotalPuntos()	Importe totalizado de las transacciones con puntos
string	getTotalConfVisa()	Importe totalizado de las transacciones confirmadas realizadas con tarjeta Visa
string	getTotalConfMc()	Importe totalizado de las transacciones confirmadas realizadas con tarjeta Master Card
string	getTotalConfAmex()	Importe totalizado de las transacciones confirmadas realizadas con tarjeta Amex
string	getTotalConfDiners()	Importe totalizado de las transacciones confirmadas realizadas con tarjeta Diners
string	getPreautorizacionesPend()	Número total de pre-autorizaciones pendientes
string	getTotalPreautorizacionesPend()	Importe totalizado de pre-autorizaciones pendientes
string	getTotalPendVisa()	Importe totalizado de las operaciones pendientes realizadas con tarjeta Visa
string	getTotalPendMc()	Importe totalizado de las operaciones pendientes realizadas con tarjeta Master Card
string	getTotalPendAmex()	Importe totalizado de las operaciones pendientes realizadas con tarjeta Amex
string	getTotalPendDiners()	Importe totalizado de las operaciones pendientes realizadas con tarjeta Diners
string	getVentasQr()	Número total de ventas QR
string	getTotalVentasQr()	Importe totalizado de las ventas QR
string	getAnulacionesQr()	Número total de anulaciones QR
string	getTotalAnulacionesQr()	Importe totalizado de anulaciones QR
string	getTotalQrVisa()	Importe totalizado de las operaciones QR realizadas con tarjeta Visa
string	getTotalQrMc()	Importe totalizado de las operaciones QR realizadas con tarjeta Master Card
string	getTotalQrAmex()	Importe totalizado de las operaciones QR realizadas con tarjeta Amex
string	getTotalQrDiners()	Importe totalizado de las operaciones QR realizadas con tarjeta Diners

Clase RespuestaReporteDetallado

Tipo de Dato	Propiedad	DESCRIPCIÓN
string	**getVenta()**	Número total de ventas
string	getTotalVenta()	Importe totalizado de las ventas
string	getAnulacion()	Número total anulaciones
string	getTotalAnulacion()	Importe totalizado de las anulaciones
string	getPuntos()	Número total de transacciones con puntos
string	getTotalPuntos()	Importe totalizado de las transacciones con puntos
string	getPreautorizacionConf()	Número total de pre-autorizaciones confirmadas
string	getTotalPreautorizacionConf()	Importe totalizado de las preautorizaciones confirmadas
string	getOperacionesConf()	Número total de transacciones confirmadas
string	getTotalConf()	Importe totalizado de las transacciones confirmadas
string	getDebitoConf()	Número de transacciones confirmadas realizadas con tarjeta de débito
string	getTotalDebitoConf()	Importe totalizado de las transacciones confirmadas realizadas con tarjeta de débito
string	getCreditoConf()	Número de transacciones confirmadas realizadas con tarjeta de crédito
string	getTotalCreditoConf()	Importe totalizado de las transacciones confirmadas realizadas con tarjeta de crédito
List<Txn>	getDetalleConf()	Lista con el detalle de transacciones confirmadas
string	getPreautorizacionPend()	Número total de pre-autorizaciones pendientes
string	getTotalPreautorizacionPend()	Importe totalizado de las preautorizaciones pendientes
string	getDebitoPend()	Número de transacciones pendientes realizadas con tarjeta de débito
string	getTotalDebitoPend()	Importe totalizado de las transacciones pendientes realizadas con tarjeta de débito
string	getCreditoPend()	Número de transacciones pendientes realizadas con tarjeta de crédito
string	getTotalCreditoPend()	Importe totalizado de las transacciones pendientes realizadas con tarjeta de crédito
List<Txn>	getDetallePend()	Lista con el detalle de transacciones pendientes
string	getOperacionQr()	Número total de operaciones QR
string	getTotalQr()	Importe totalizado de operaciones QR
string	getAnulacionesQR()	Número total de anulaciones QR

string	getTotalAnulacionesQR()	Importe totalizado de las anulaciones QR realizadas
string	getDebitoQr()	Número de transacciones QR realizadas con tarjeta de débito
string	getTotalDebitoQr()	Importe totalizado de las transacciones QR realizadas con tarjeta de débito
string	getBilleteraBBVA()	Número de transacciones QR realizadas con billetera BBVA
string	getTotalBilleteraBBVA()	Importe totalizado de las transacciones QR realizadas con billetera BBVA
string	getBilleteraYape()	Número de transacciones QR realizadas con billetera Yape
string	getTotalBilleteraYape()	Importe totalizado de las transacciones QR realizadas con billetera Yape
List<Txn>	getDetalleQR()	Lista con el detalle de transacciones QR

Clase RespuestaReporteResumen

Tipo de Dato	Propiedad	DESCRIPCIÓN
string	**getFechaInicio()**	*Fecha de inicio del reporte.*
string	getFechaCierre()	*Fecha de cierre del reporte.*
string	getTipoOperaciones()	*Número de operaciones por tipo.*
string	getTipoTotal()	*Total por tipo.*
string	getTipoVentasTotal()	*Total de ventas por tipo.*
string	getTipoVentasOperaciones()	*Número de operaciones de ventas por tipo.*
string	getTipoVentasDccTotal()	*Total de ventas DCC por tipo.*
string	getTipoVentasDccOperaciones()	*Número de operaciones de ventas DCC por tipo.*
string	getTipoPropinasTotal()	*Total de propinas por tipo.*
string	getTipoPropinasOperaciones()	*Número de operaciones de propinas por tipo.*
string	getTipoPagoConPuntosTotal()	*Total de pagos con puntos por tipo.*
string	getTipoPagoConPuntosOperaciones()	*Número de operaciones de pagos con puntos por tipo.*
string	getTipoAnulacionesTotal()	*Total de anulaciones por tipo.*
string	getTipoAnulacionesOperaciones()	*Número de operaciones de anulaciones por tipo.*

List<Txn>	getTipoConfirmacionesDccTotal()	*Total de confirmaciones DCC por tipo.*
string	getTipoConfirmacionesDccOperaciones()	*Número de operaciones de confirmaciones DCC por tipo.*
string	getTipoConfirmacionesTotal()	*Total de confirmaciones por tipo.*
string	getTipoConfirmacionesOperaciones()	*Número de operaciones de confirmaciones por tipo.*
string	getMarcasVisaTotal()	*Total de Visa por marca.*
string	getMarcasVisaOperaciones()	*Número de operaciones de Visa por marca.*
string	getMarcasMcTotal()	*Total de Mastercard por marca.*
List<Txn>	getMarcasMcOperaciones()	*Número de operaciones de Mastercard por marca.*
string	getMarcasAmexTotal()	*Total de Amex por marca.*
string	getMarcasAmexOperaciones()	*Número de operaciones de Amex por marca.*
string	getMarcasDinersTotal()	*Total de Diners por marca.*
string	getMarcasDinersOperaciones()	*Número de operaciones de Diners por marca.*
string	getProductoTotal()	*Total por producto.*
string	getProductoOperaciones()	*Número de operaciones por producto.*
string	getProductoDebitoTotal()	*Total de débito por producto.*
string	getProductoDebitoOperaciones()	*Número de operaciones de débito por producto.*
string	getProductoCreditoTotal()	*Total de crédito por producto.*
string	getProductoCreditoOperaciones()	*Número de operaciones de crédito por producto.*
List<Txn>	getDetalleQR()	Lista con el detalle de transacciones QR
string	getOperacionQr()	*Número total de operaciones QR.*
string	getTotalQr()	*Importe totalizado de operaciones QR.*
string	getAnulacionesQr()	*Número total de anulaciones QR.*
string	getTotalAnulacionesQr()	*Importe totalizado de las anulaciones QR.*

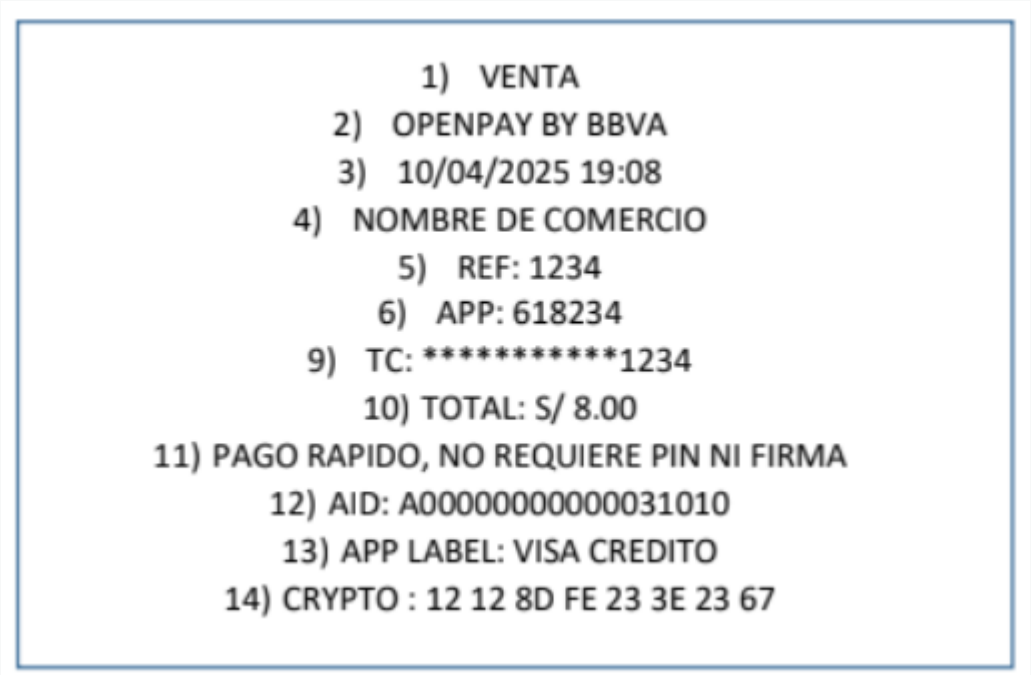
Clase Respuesta ReporteOperaciones

Tipo de Dato	Propiedad	DESCRIPCIÓN
string	**getFechaInicio()**	*Fecha de inicio del reporte.*
string	getFechaCierre()	*Fecha de cierre del reporte.*
string	getConfOperaciones()	*Número de operaciones confirmadas.*
string	getConfTotal()	*Total de operaciones confirmadas.*
string	getConfVentasTotal()	*Total de ventas confirmadas.*
string	getConfVentasOperaciones()	*Número de operaciones de ventas confirmadas.*
string	getConfVentasDccTotal()	*Total de ventas DCC confirmadas.*
string	getConfVentasDccOperaciones()	*Número de operaciones de ventas DCC confirmadas.*
string	getConfPagoPuntosTotal()	*Total de pagos con puntos confirmados.*
string	getConfPagoPuntosOperaciones()	*Número de operaciones de pagos con puntos confirmados.*
string	getConfAnulacionesTotal()	*Total de anulaciones confirmadas.*
string	getConfAnulacionesOperaciones()	*Número de operaciones de anulaciones confirmadas.*
string	getConfConfirmacionesTotal()	*Total de confirmaciones.*
string	getConfConfirmacionesOperaciones()	*Número de operaciones de confirmaciones.*
List<Txn>	getConfConfirmacionesDccTotal()	*Total de confirmaciones DCC.*
string	getConfConfirmacionesDccOperaciones()	*Número de operaciones de confirmaciones DCC.*
string	getConfProductoTotal()	*Total por producto confirmado.*
string	getConfProductoOperaciones()	*Número de operaciones por producto confirmado.*
string	getConfProductoCreditoTotal()	*Total de crédito por producto confirmado.*
string	getConfProductoCreditoOperaciones()	*Número de operaciones de crédito por producto confirmado.*
string	getConfProductoDebitoTotal()	*Total de débito por producto confirmado.*
string	getConfProductoDebitoOperaciones()	*Número de operaciones de débito por producto confirmado.*
List<TransaccionOperaciones>	getConfTxns()	*Lista de transacciones confirmadas.*
string	getPendOperaciones()	*Número de operaciones pendientes.*
string	getPendTotal()	*Total de operaciones pendientes.*

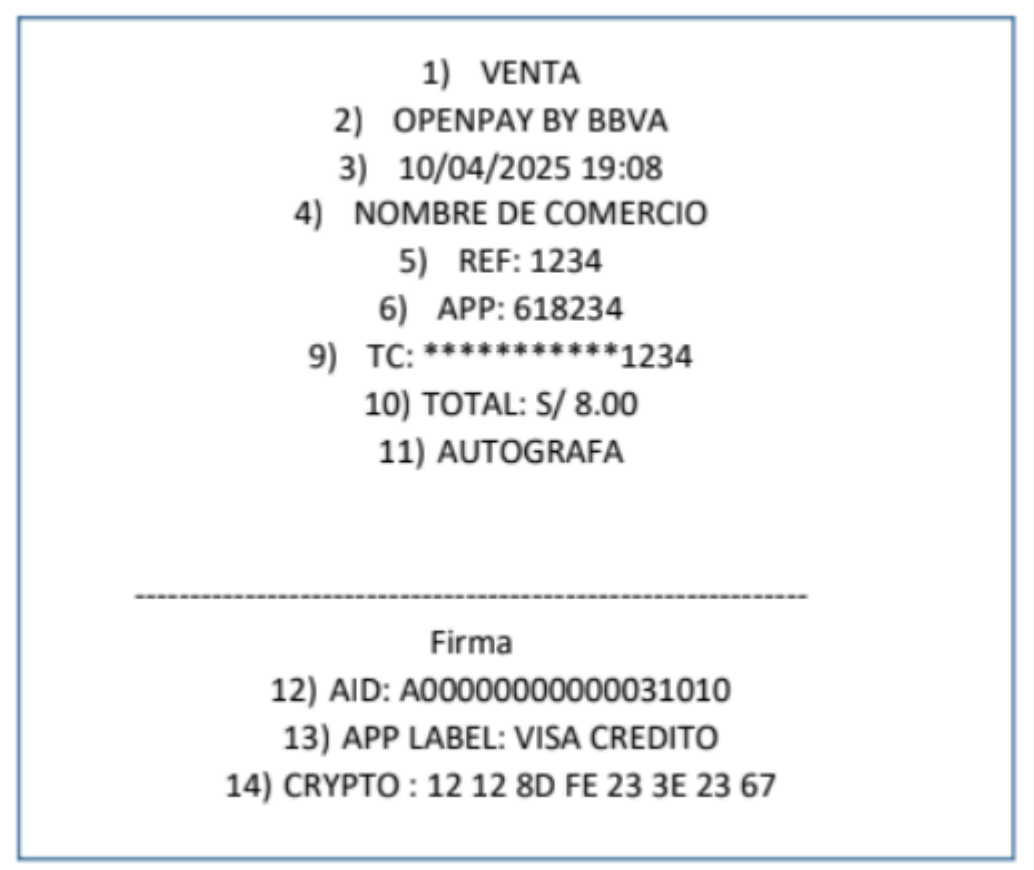
List<TransaccionOperaciones>	getPendTxns()	*Lista de transacciones pendientes.*
string	getOperacionConfQr()	*Número total de operaciones QR.*
string	getTotalConfQr()	*Importe totalizado de operaciones QR.*
string	getAnulacionesConfQr()	*Número total de anulaciones QR.*
string	getTotalAnulacionesConfQr()	*Importe totalizado de las anulaciones QR.*
List<TransaccionDetalle>	getDetalleConfQr()	*Lista con el detalle de transacciones QR.*
string	getOperacionPendQr()	*Número total de operaciones QR.*
string	getTotalPendQr()	*Importe totalizado de operaciones QR.*
string	getAnulacionesPendQr()	*Número total de anulaciones QR.*
List<TransaccionDetalle>	getTotalAnulacionesPendQr()	*Importe totalizado de las anulaciones QR.*
String	getDetallePendQr()	*Lista con el detalle de transacciones QR.*

ANEXO. Ejemplos Diagramación de Vouchers

1) PAGO CON TARJETA FISICA PIN



2) PAGO CON TARJETA FISICA CON FIRMA

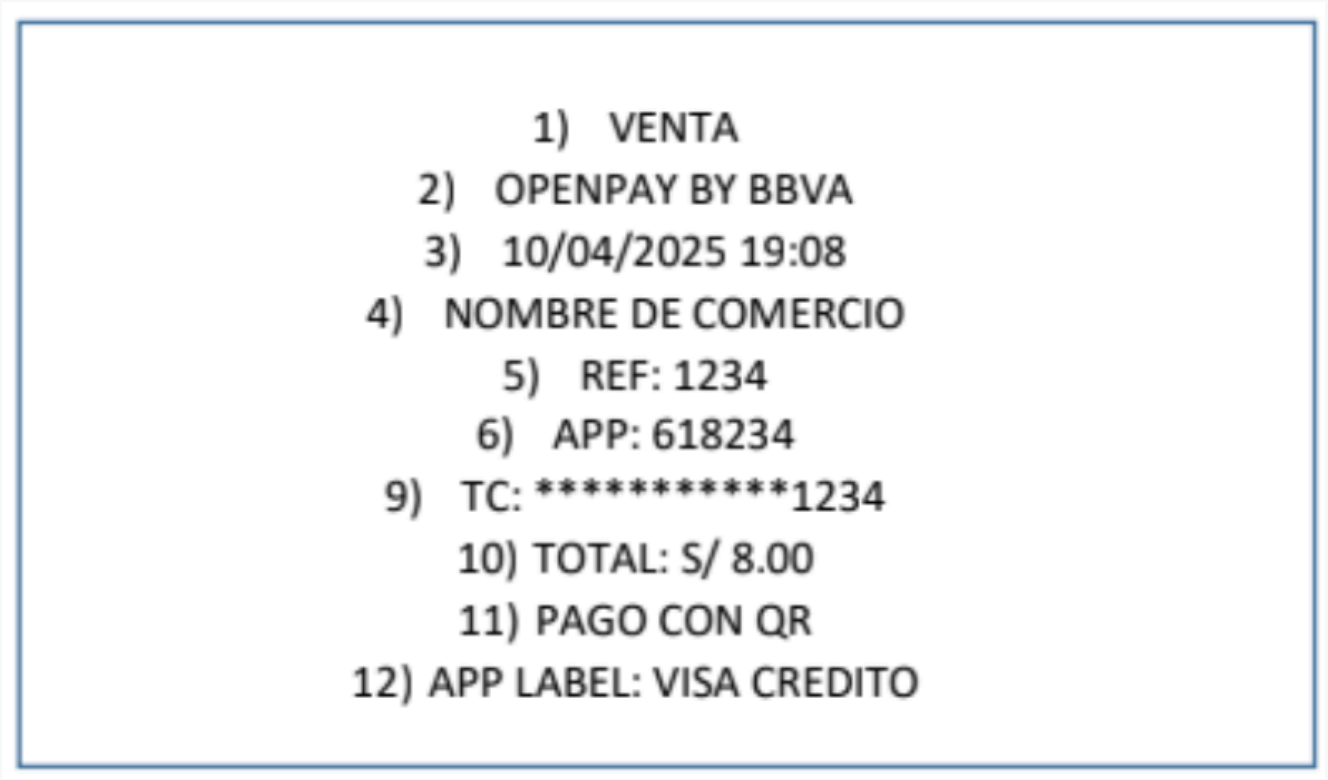

1) VENTA
2) OPENPAY BY BBVA
3) 10/04/2025 19:08
4) NOMBRE DE COMERCIO
5) REF: 1234
6) APP: 618234
9) TC: **********1234
10) TOTAL: S/ 8.00
11) AUTOGRAFA

Firma
12) AID: A00000000000031010
13) APP LABEL: VISA CREDITO
14) CRYPTO : 12 12 8D FE 23 3E 23 67

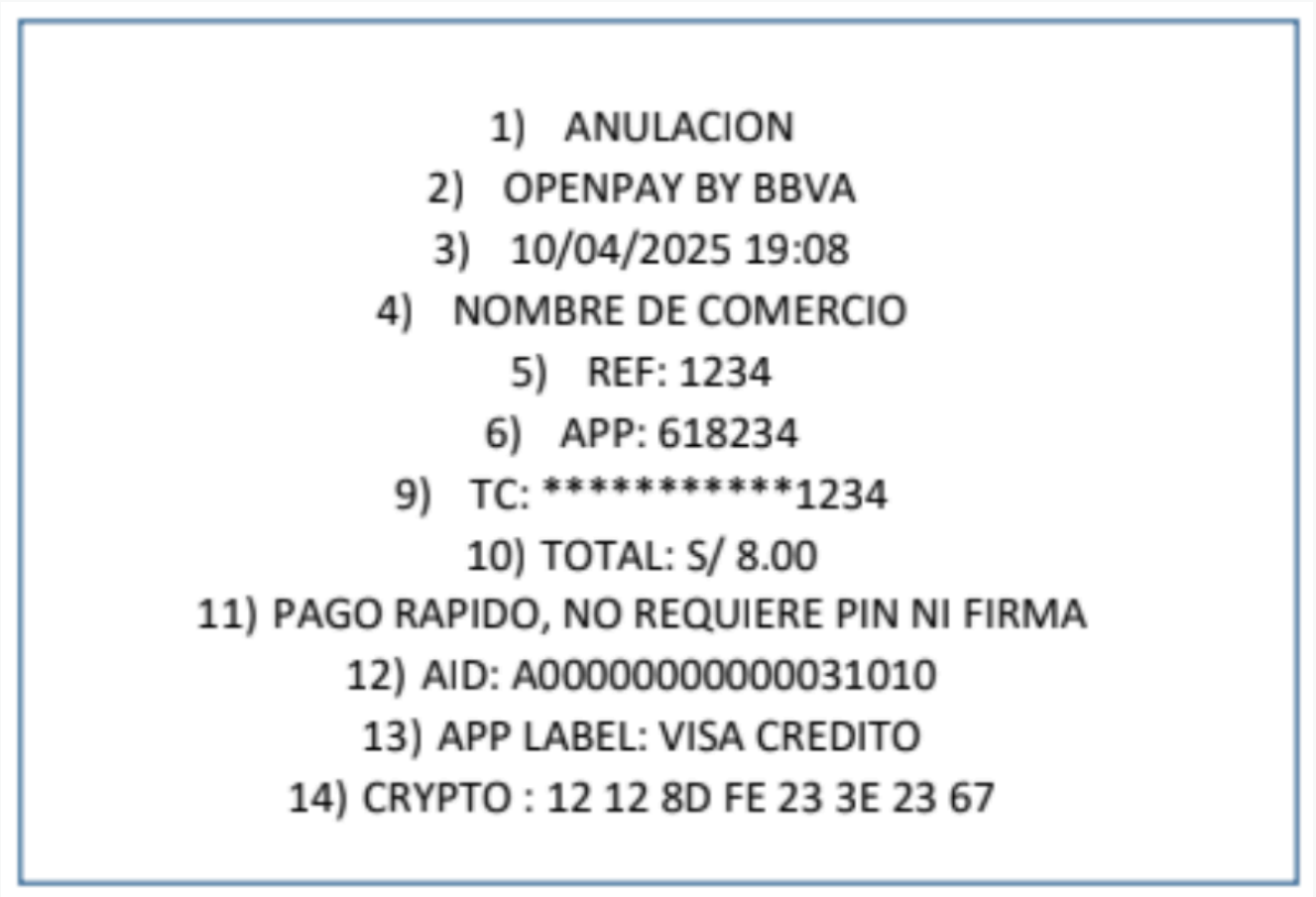
3) PAGO CON TARJETA FISICA VENTA CON CUOTAS

1) VENTA
2) OPENPAY BY BBVA
3) 10/04/2025 19:08
4) NOMBRE DE COMERCIO
5) REF: 1234
6) APP: 618234
7) CUOTAS: 2
8) CUOTA APROX: 12.34
9) TC: **********1234
10) TOTAL: S/ 8.00
11) PAGO RAPIDO, NO REQUIERE PIN NI FIRMA
12) AID: A0000000000031010
13) APP LABEL: VISA CREDITO
14) CRYPTO : 12 12 8D FE 23 3E 23 67

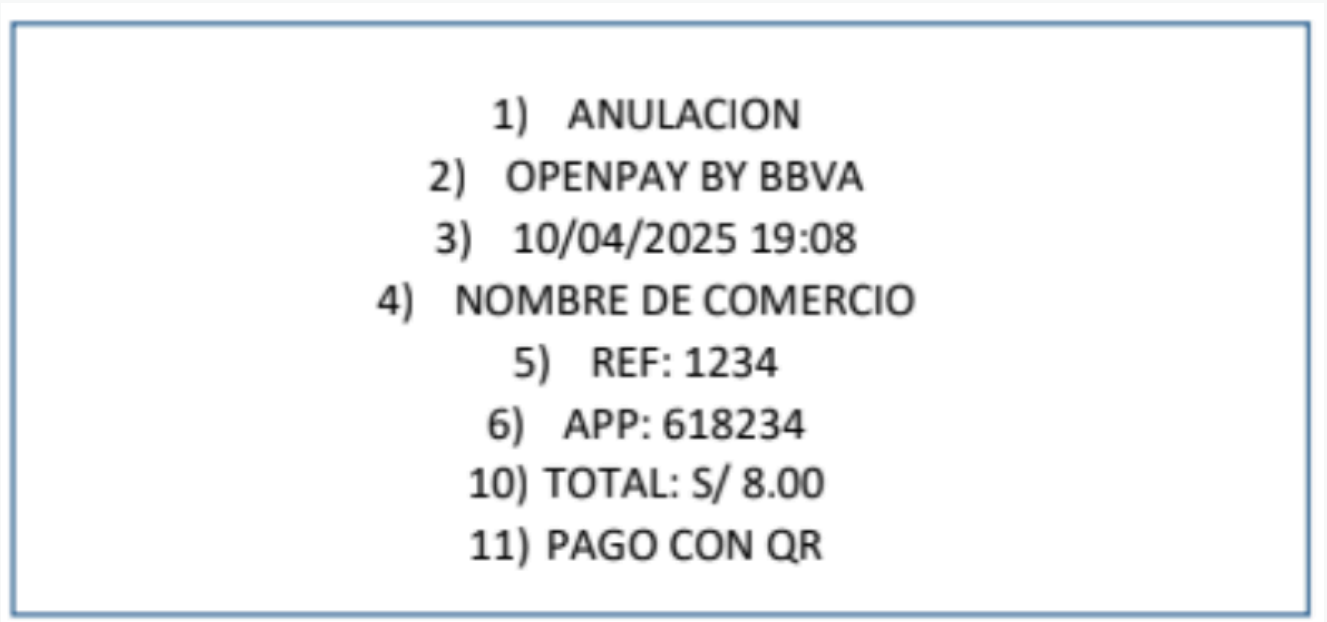
4) PAGO CON QR


1) VENTA
2) OPENPAY BY BBVA
3) 10/04/2025 19:08
4) NOMBRE DE COMERCIO
5) REF: 1234
6) APP: 618234
9) TC: **********1234
10) TOTAL: S/ 8.00
11) PAGO CON QR
12) APP LABEL: VISA CREDITO

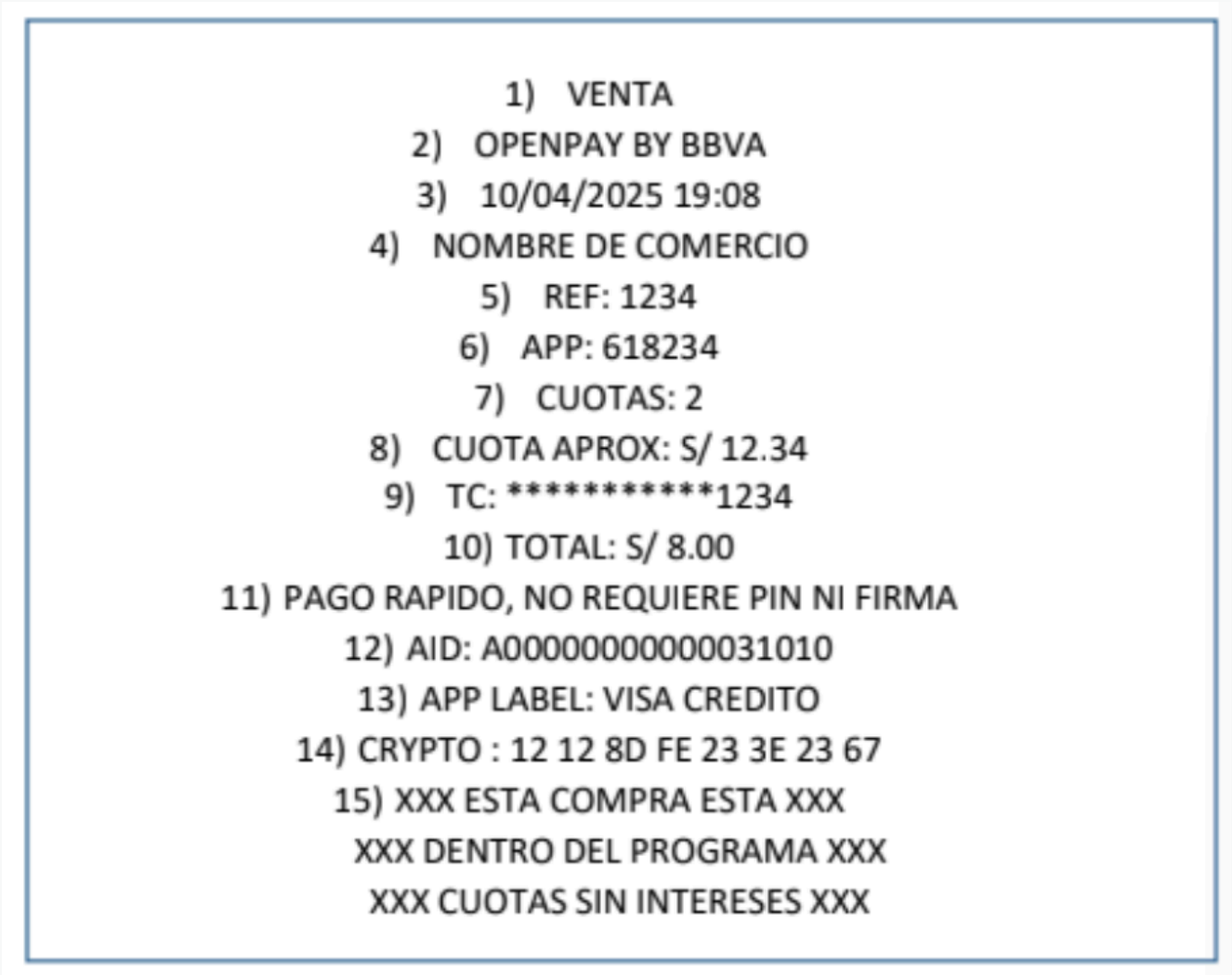
5) ANULACIÓN PAGO CON TARJETA FISICA



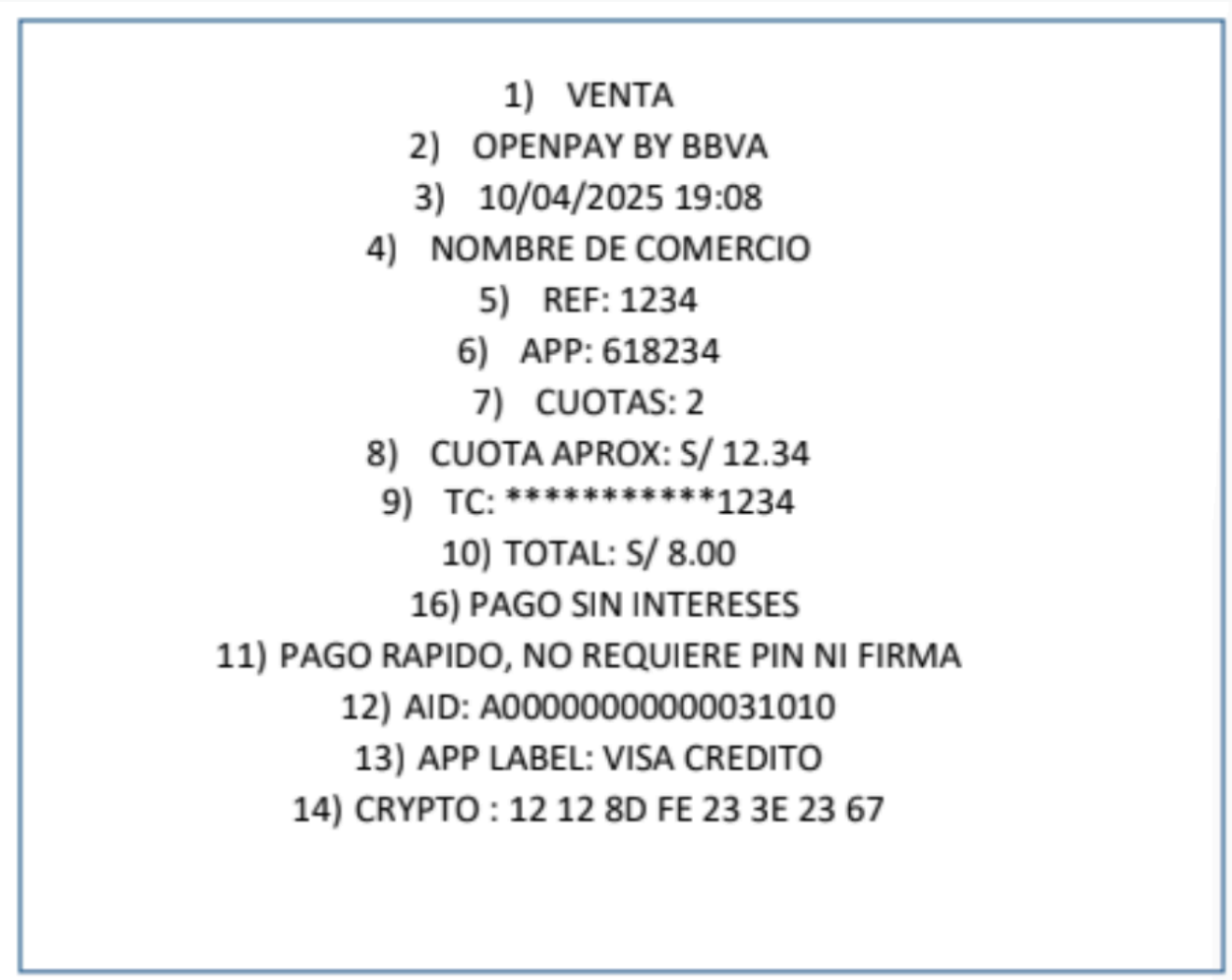
6) ANULACION PAGO CON QR


1) ANULACION
2) OPENPAY BY BBVA
3) 10/04/2025 19:08
4) NOMBRE DE COMERCIO
5) REF: 1234
6) APP: 618234
10) TOTAL: S/ 8.00
11) PAGO CON QR

7) VOUCHER CON CAMPAÑA CSI DINERS


1) VENTA
2) OPENPAY BY BBVA
3) 10/04/2025 19:08
4) NOMBRE DE COMERCIO
5) REF: 1234
6) APP: 618234
7) CUOTAS: 2
8) CUOTA APROX: S/ 12.34
9) TC: **********1234
10) TOTAL: S/ 8.00
11) PAGO RAPIDO, NO REQUIERE PIN NI FIRMA
12) AID: A00000000000031010
13) APP LABEL: VISA CREDITO
14) CRYPTO : 12 12 8D FE 23 3E 23 67
15) XXX ESTA COMPRA ESTA XXX
XXX DENTRO DEL PROGRAMA XXX
XXX CUOTAS SIN INTERESES XXX

8) VOUCHER CON CAMPAÑA PSI BBVA



Obtención de datos

1. Tipo de operación si es VENTA o ANULACION este valor es fijo
2. Nombre del adquirente OPENPAY BY BBVA este valor es fijo
3. Fecha y hora de la transacción se puede obtener del campo “fechaHora” de la respuesta de la transacción, formato DD/MM/AAAA HH:MM
4. Nombre del Comercio se puede obtener del campo “razonSocial” de la transacción
5. Se debe poner solo los ultimo 4 dígitos del número que viene en el campo “referenciaFinanciera”
6. Se debe mostrar el código de autorización que se obtiene del campo “autorización”
7. Se debe mostrar el número de cuotas que se obtiene del campo “cuotas”
8. Se debe mostrar el cálculo de cuota aproximada se obtiene del campo “montoCuota”

9. Se debe mostrar los últimos 4 dígitos de la tarjeta que se usó para el pago se obtiene del campo “numeroTarjeta”
10. Se debe mostrar el monto de la venta, se obtiene del campo uniendo el campo “moneda” + “importe”
11. Aquí se debe agregar el método de captura según la siguiente lógica:
    a. Si en el campo firma viene:

        i. ELECTRÓNICA – PIN VERIFICADO
        ii. SIN FIRMA – PAGO RAPIDO, NO REQUIERE PIN NI FIRMA
        iii. AUTÓGRAFA – Se debe agregar el campo de firma en el voucher para que el cliente firme de manera presencial
        iv. Si se paga con QR en el campo “modoLectura” venta el código “03” – PAGO CON QR
12. Se debe mostrar el número que viene en el campo “IdAplicacionTarjeta”
13. Colocar la marca de la tarjeta si es Visa/Mastercard/Amex o Diners, esto vienen en el campo “aplicación tarjeta”
14. Colocar el número que se puede extraer del campo “CriptogramaTarjeta”
15. Colocar en el caso que venga poblado el campo “mensajes”, por lo general viene poblado cuando el Comercio está afiliado a las campañas CSI de Diners = XXX ESTA COMPRA ESTA XXX XXX DENTRO DEL PROGRAMA XXXX XXX CUOTAS SIN INTERESES XXXX
16. Se debe mostrar el mensaje “PAGO SIN INTERESES”, solo si el campo “codigoPromocion” venga poblado con el mensaje “PROMOCION.MESES_SIN_INTERESES”, caso contrario no se debe mostrar ningún mensaje. Esto solo aplica si el Comercio está afiliado al programa de PAGOS SIN INTERESES con BBVA