API WebSocket Quickpaymini
Versión 1.0.0

Manual de Integración

APS®, Aplicaciones y Soluciones de Pago
Marzo 2026

Bitácora de cambios

Versión	Fecha	Descripción de Cambios
1.0.0	Marzo 11, 2026	Se libera primera versión del documento.

Manual de Integración para el Servidor Quickpaymini

Introducción

Este manual está diseñado para programadores que desean integrar sistemas con el equipo Quickpaymini, un cajero automático que utiliza una computadora con Windows 10 Profesional. El equipo está configurado para recibir y procesar instrucciones a través de una red Wi-Fi generada por un router inalámbrico conectado a la computadora del cajero. Además, también es posible realizar la integración en una red local (LAN) dentro de la misma red Wi-Fi.

Configuración del Sistema

- **Nombre del equipo**: Servidor Quickpaymini
- **Sistema Operativo**: Windows 10 Profesional
- **Conexión de Red**:
    - **Red Wi-Fi Directa**: El servidor está conectado a un router que genera una red Wi-Fi con nombre quickpayminiXXX donde XXX es un número de serie único. La contraseña predeterminada para esta red es “abcdefgh” (sin las comillas).
    - **Red Local (LAN)**: Además de la conexión Wi-Fi directa, también es posible realizar la integración en una red local (LAN) dentro de la misma red Wi-Fi, permitiendo a otros dispositivos en la red enviar instrucciones al servidor Quickpaymini.
- **Puerto de Comunicación**: El servidor Quickpaymini está configurado para escuchar en el puerto 8080.

Acceso a la Webapp

El servidor Quickpaymini aloja una aplicación web (webapp) que permite controlar el equipo y todas sus funciones. Esta webapp está contenida en un archivo index.html, el cual puedes encontrar en el directorio “C:\inetpub\wwwroot” del servidor.

- **Archivo Principal**: index.html
- **Propósito**: Este archivo contiene todas las funciones necesarias para realizar operaciones con el equipo Quickpaymini.
- **Acceso al Archivo**: Los integradores pueden acceder al archivo index.html en cualquier momento para consultar o aclarar dudas durante la implementación.

Integración con Websockets

El servidor Quickpaymini utiliza Websockets para la comunicación en tiempo real entre la webapp y el motor que controla las funciones del cajero.

- **Aplicación del Motor**: Quickpaymini_Engine.exe
- **Función**: Esta aplicación se ejecuta en el servidor y se encarga de procesar todas las instrucciones enviadas a través de los Websockets.
- **Estructura de Comunicación**: La webapp envía y recibe datos mediante Websockets, los cuales son gestionados por Quickpaymini_Engine.exe para ejecutar las operaciones solicitadas.

Funcionalidades Clave de la Webapp

La webapp cuenta con diversas funcionalidades para manejar las operaciones del cajero. Algunas de las funciones clave incluyen:

- **Conexión y Re-conexión**: La webapp está diseñada para intentar reconectarse automáticamente al servidor si la conexión se pierde.
- **Manejo de Transacciones**: Se incluye un sistema de manejo de transacciones que permite realizar pagos, consultas de saldo, y más.
- **Sonidos y Notificaciones**: La webapp puede reproducir sonidos específicos para distintas acciones (e.g., al hacer clic en un botón, cuando una transacción es exitosa, etc.).
- **Visualización de Estado**: La interfaz de usuario muestra el estado actual del servidor, incluyendo el número de conexiones activas y si se está imprimiendo o procesando una transacción.

Consideraciones para la Implementación

1. **Acceso y Control**: Es importante asegurarse de que el servidor y la webapp estén correctamente configurados para permitir la comunicación sin problemas.
2. **Uso del Archivo index.html**: Este archivo no solo actúa como la interfaz de usuario, sino también como una referencia de las funciones disponibles para la integración.
3. **Pruebas y Validaciones**: Se recomienda realizar pruebas exhaustivas utilizando el index.html y la webapp para garantizar que todas las integraciones funcionen como se espera antes de su implementación en un entorno de producción.
4. **Conexión en Red Local**: Si prefieres realizar la integración en una red local (LAN), asegúrate de que todos los dispositivos involucrados estén conectados a la misma red Wi-Fi y que puedan comunicarse con el servidor Quickpaymini a través del puerto 8080.

Paso 1: Establecimiento de Conexión WebSocket

El primer paso para integrar tu aplicación con el servidor Quickpaymini es establecer una conexión WebSocket. A continuación, se presenta un ejemplo detallado utilizando el código tal como está en el archivo index.html proporcionado, explicando cómo funciona cada parte del código y cómo puedes adaptarlo para tus necesidades.

Ejemplo: Conexión WebSocket en JavaScript utilizando index.html

Código Ejemplo (Extraído y Adaptado del index.html):

```
// Definición de variables globales
var socket = null;
var isConnected = false;
var reconnectInterval = null;
var hostComputer = "";

// Función para conectar al WebSocket
function connectWebSocket() {
  if (socket !== null) {
    socket.close();
    socket = null;
  }

  // Determinar el host del servidor
  if (hostComputer == window.location.hostname) {
    hostComputer = "192.168.1.135";  // Cambia a la IP del servidor si es necesario
  } else {
    hostComputer = window.location.hostname;
  }

  console.log("Host Computer: " + hostComputer);

  // Crear una nueva conexión WebSocket
  socket = new WebSocket("ws://" + hostComputer + ":8080/");

  // Evento cuando la conexión es abierta
  socket.addEventListener("open", event => {
    isConnected = true;
    clearInterval(reconnectInterval);
    console.log("Conexión establecida con el servidor Quickpaymini.");
  });

  // Evento cuando se recibe un mensaje del servidor
  socket.addEventListener("message", event => {
    const message = event.data;
    console.log("Mensaje recibido:", message);
```

```
    // Procesar el mensaje recibido (ejemplo)
    const jsonMessage = JSON.parse(message);
    const comando = jsonMessage.comando;

    switch (comando) {
      case "bvresetfinish":
        console.log("Comando bvresetfinish recibido y procesado.");
        // Agregar aquí la lógica para manejar el comando
        break;
      // Otros casos para diferentes comandos
    }
  });

  // Evento cuando la conexión es cerrada
  socket.addEventListener("close", event => {
    if (isConnected) {
      isConnected = false;
      console.log("Conexión cerrada. Intentando reconectar en 3 segundos...");
      reconnectInterval = setInterval(connectWebSocket, 3000);
    }
  });

  // Evento cuando ocurre un error en la conexión
  socket.addEventListener("error", event => {
    if (isConnected) {
      isConnected = false;
      console.error("Error en la conexión WebSocket:", event);
      reconnectInterval = setInterval(connectWebSocket, 3000);
    }
  });
}


// Iniciar la conexión WebSocket

connectWebSocket();
```

Explicación Detallada del Código:

1. **Variables Globales**:
    - socket: Es la variable que mantiene la instancia de WebSocket.
    - isConnected: Indica si la conexión con el servidor está activa.
    - reconnectInterval: Se utiliza para manejar los intervalos de reconexión.
    - hostComputer: Define el host del servidor al que se conectará.
2. **Función connectWebSocket()**:
    - **Reinicialización del Socket**: Si ya existe una conexión previa, se cierra y se reinicializa para evitar múltiples conexiones.

- **Determinación del Host**: Utiliza window.location.hostname para determinar si la aplicación está corriendo en el mismo dominio que el servidor. Si no es así, se utiliza una IP fija, como 192.168.1.135, que puedes personalizar según tu entorno.
- **Creación de Conexión**: La conexión WebSocket se establece con el servidor en ws://hostComputer:8080/.

3. **Manejo de Eventos**:
   - **open**: Este evento se dispara cuando la conexión se establece correctamente. Aquí se marca isConnected como true y se limpia cualquier intento de reconexión anterior.
   - **message**: Este evento se dispara cada vez que el servidor envía un mensaje. Se convierte el mensaje a un objeto JSON y se procesa según el comando recibido.
   - **close**: Este evento se dispara cuando la conexión se cierra. Si isConnected es true, se intentará reconectar cada 3 segundos.
   - **error**: Este evento se maneja cuando ocurre un error en la conexión. Similar al evento close, se inicia un proceso de reconexión automática.
4. **Iniciar la Conexión**:
   - Se llama a la función connectWebSocket() para establecer la conexión cuando la aplicación se carga.

Adaptaciones y Usos:

- **Usar window.location.hostname**: Este método permite que la conexión sea más dinámica, especialmente útil si el servidor WebSocket y la aplicación web comparten el mismo dominio. Esto es particularmente beneficioso cuando la IP del servidor no es fija.
- **Re-conexión Automática**: El código está preparado para intentar reconectar automáticamente en caso de que la conexión se pierda, lo que es crucial en entornos de producción donde la estabilidad de la conexión no siempre puede garantizarse.
- **Manejo de Comandos**: Dentro del evento message, el código está preparado para manejar múltiples tipos de comandos. Esto es útil si tu aplicación necesita procesar diferentes tipos de mensajes del servidor.

Paso 2: Descripción de los Comandos WebSocket

En este paso, se describen todos los comandos que pueden ser enviados a través del WebSocket al servidor Quickpaymini. Estos comandos son esenciales para interactuar con las distintas funciones del servidor. A continuación, se presenta un resumen de cada comando, junto con sus parámetros y el propósito de cada uno, todos ellos pueden ser encontrados en el archivo index.html.

---

paycash

Descripción: Este comando inicia un pago en efectivo.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "paycash"
- due: El monto que se debe pagar en efectivo.

Ejemplo:

```
var f = new Date();
var c = "paycash";
var d = "10.00";   // Monto a pagar
var data = {
  fecha: f,
  comando: c,
  due: d
};
socket.send(JSON.stringify(data));
```

---

paycreditcard

Descripción: Este comando inicia un pago con tarjeta de crédito.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "paycreditcard"
- dueCC: El monto que se debe pagar con tarjeta de crédito.

Ejemplo:

```
var f = new Date();
var c = "paycreditcard";
var d = "10.00";            // Monto a pagar
var data = {
  fecha: f,
```

```
  comando: c,
  dueCC: d
};
socket.send(JSON.stringify(data));
```

---

opendoor

Descripción: Este comando abre la puerta del cajero para mantenimiento o recolección de efectivo.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "opendoor"

Ejemplo:

```
var f = new Date();
var c = "opendoor";
var data = {
  fecha: f,
  comando: c
};
socket.send(JSON.stringify(data));
```

---

cashrefill

Descripción: Este comando inicia el proceso de dotación de efectivo en el sistema.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "cashrefill"

Ejemplo:

```
var f = new Date();
var c = "cashrefill";
var data = {
  fecha: f,
  comando: c
};
socket.send(JSON.stringify(data));
```

---

cashcollect

Descripción: Este comando recolecta efectivo de una fuente específica, como monedas o billetes.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.

- comando: "cashcollect"
- source: La fuente de recolección, que puede ser "coin" o "bill".

Ejemplo:

```
var f = new Date();
var c = "cashcollect";
var s = "coin"; // O puede ser "bill"
var data = {
  fecha: f,
  comando: c,
  source: s
};
socket.send(JSON.stringify(data));
```

---

resetbalance

Descripción: Este comando realiza el corte de caja del sistema, típicamente usado para iniciar un nuevo ciclo de transacciones.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "resetbalance"

Ejemplo:

```
var f = new Date();
var c = "resetbalance";
var data = {
  fecha: f,
  comando: c
};
socket.send(JSON.stringify(data));
```

---

shutdown

Descripción: Este comando apaga o reinicia el servidor.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "shutdown"
- param: Puede ser "r" para reiniciar o "s" para apagar.

Ejemplo:

```
var f = new Date();
var c = "shutdown";
var d = "r"; // Reiniciar
var data = {
```

```
  fecha: f,
  comando: c,
  param: d
};
socket.send(JSON.stringify(data));
```

---

cancel

Descripción: Este comando cancela la operación actual. El valor de cancelación puede variar según la operación en curso.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "cancel"
- cancelvalue: El valor de cancelación, que puede representar diferentes niveles o etapas de cancelación.

Ejemplo:

```
var f = new Date();
var c = "cancel";
var cv = "1";      // Valor 1 cancela regresando cambio, valor 2 cancela sin regresar cambio.
var data = {
  fecha: f,
  comando: c,
  cancelvalue: cv
};
socket.send(JSON.stringify(data));
```

Paso 3: Recepción de Respuestas del Servidor

En este paso, se explicará cómo la aplicación JavaScript que se conecta al servidor Quickpaymini debe manejar las respuestas que recibe del servidor a través del WebSocket. Esta sección se centrará en cómo se configura el evento que escucha los mensajes entrantes y cómo se procesan estos mensajes en función del contenido recibido.

Manejo de Respuestas con socket.addEventListener("message", event => { ...)

El manejo de las respuestas del servidor se realiza mediante la escucha de eventos message en el objeto WebSocket. Este evento se dispara cada vez que el servidor envía un mensaje al cliente, y es aquí donde se procesan las respuestas basadas en el contenido de los mensajes.

Configuración Básica:

```
socket.addEventListener("message", event => {
  // Parsear el mensaje recibido del servidor
  const message = JSON.parse(event.data);

  // Identificar el comando dentro del mensaje recibido
  const comando = message.comando;

  // Procesar el mensaje según el comando
  switch (comando) {
    case "exampleCommand":
      // Aquí se maneja la lógica específica para el comando "exampleCommand"
      console.log("Se recibió el comando 'exampleCommand' con los datos:", message);
      break;

    // Incluir otros casos para diferentes comandos
    default:
      console.warn("Comando no reconocido:", comando);
      break;
  }
});
```

Explicación:

1. **Escucha del Evento message:**
    - socket.addEventListener("message", event => { ... }) se utiliza para registrar un manejador de eventos que se ejecutará cada vez que el servidor envíe un mensaje al cliente.
    - El event contiene la información del mensaje enviado por el servidor.
2. **Parseo del Mensaje:**
    - El mensaje recibido por el servidor suele estar en formato JSON, por lo que se parsea usando JSON.parse(event.data) para convertirlo en un objeto JavaScript.

- Este objeto contendrá varios campos, uno de los cuales es el comando, que identifica la naturaleza de la respuesta.

3. **Identificación del Comando:**
    - El campo comando dentro del mensaje se extrae para determinar qué acción tomar en función de la respuesta recibida.
    - Dependiendo del valor de comando, se ejecutará una lógica diferente.
4. **Manejo del Comando:**
    - Se utiliza un bloque switch para manejar diferentes valores de comando. Por ejemplo, si el comando es "exampleCommand", se ejecutará un bloque de código específico para ese comando.
    - Si se recibe un comando que no está reconocido, se maneja en el bloque default, donde se puede registrar una advertencia o un mensaje de error.

Caso Genérico:

En la implementación genérica, como en el ejemplo anterior, solo se muestra cómo se puede manejar un comando específico (exampleCommand). Sin embargo, en una aplicación completa, este bloque switch incluirá todos los comandos posibles que se esperan recibir del servidor, cada uno con su lógica de manejo particular. Los detalles de la lógica para cada comando se explorarán más adelante.

Paso 4: Manejo de Respuestas del Servidor

En este paso, se detallarán las respuestas que puede enviar el servidor Quickpaymini a través del WebSocket y cómo la aplicación cliente debe manejar estas respuestas. Cada respuesta corresponde a un comando específico que el servidor envía como mensaje JSON en función de las acciones o eventos que ocurren en el servidor. A continuación, se describen los comandos identificados en el código fuente de la aplicación C# y los parámetros asociados con cada uno.

---

access

Descripción: Este comando se envía para indicar si el acceso al menú administrativo ha sido permitido o denegado.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "access"
- validNIP: Indica si el NIP ingresado es válido (true o false).
- token: El token asociado con la sesión de acceso.

---

productdeleted

Descripción: Este comando se envía cuando un producto ha sido eliminado exitosamente de la base de datos.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "productdeleted"

---

searchresults

Descripción: Este comando se envía con los resultados de una búsqueda de productos en la base de datos.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "searchresults"
- datos: Un array de objetos que contiene los detalles de los productos encontrados (código, descripción, unidad de medida, precio, imagen).

---

newproductcode

Descripción: Este comando se envía con un nuevo código de producto generado por el sistema.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "newproductcode"
- codigoproducto: El nuevo código de producto generado.

---

newproductsaved

Descripción: Este comando se envía cuando un nuevo producto ha sido guardado exitosamente en la base de datos.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "newproductsaved"

---

bvresetfinish

Descripción: Este comando se envía para indicar que el proceso de reinicio del billetero electrónico ha finalizado.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "bvresetfinish"

---

productinfo

Descripción: Este comando se envía con la información de un producto escaneado o seleccionado.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "productinfo"
- codigoproducto: El código del producto.
- descripcion: La descripción del producto.
- precio: El precio del producto.
- udm: La unidad de medida del producto.

---

checkbrresult

Descripción: Este comando se envía para informar sobre el estado de los niveles del reciclador de billetes.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "checkbrresult"
- brempty: Indica si el reciclador de billetes está vacío (true o false).
- maxqtybr: Un array que contiene los niveles máximos de billetes en el reciclador.

---

cashcollectfinish

Descripción: Este comando se envía cuando ha finalizado el proceso de recolección de efectivo.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "cashcollectfinish"

---

reportready

Descripción: Este comando se envía para indicar que un reporte solicitado ha sido generado y está listo.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "reportready"

---

screensettings

Descripción: Este comando se envía con los parámetros de configuración de la pantalla.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "screensettings"
- showcta: Indica si se debe mostrar el mensaje de llamada a la acción ("1" para mostrar, "0" para ocultar).
- showeyes: Indica si se debe mostrar la animación de ojos ("1" para mostrar, "0" para ocultar).
- showtime: Indica si se debe mostrar el reloj ("1" para mostrar, "0" para ocultar).
- ctamessage: El mensaje de llamada a la acción que se muestra en la pantalla.

---

status

Descripción: Este comando se envía para proporcionar el estado actual del servidor Quickpaymini, incluyendo el estado de las transacciones, crédito disponible, y otros detalles operativos.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "status"
- conexiones: El número de conexiones activas.
- cobrando: Indica si el servidor está procesando un cobro (true o false).
- bvError: Información sobre errores en el billetero electrónico (si los hay).
- caError: Información sobre errores en el monedero (si los hay).
- tarjeta: Indica si el pago con tarjeta está disponible (true o false).
- Status: El estado actual traducido al español.
- referenceCC: El número de referencia de la transacción actual.
- Credit: El monto de crédito disponible.
- Due: El monto debido en la transacción actual.
- changeDispensed: El cambio entregado.

- tubesSW: La cantidad de monedas en los tubos.
- coinCashboxAmount: El monto de monedas en la caja.
- coinCashboxQty: La cantidad de monedas en la caja.
- billsCashboxAmount: El monto de billetes en la caja.
- billsCashboxQty: La cantidad de billetes en la caja.
- changeBar: Información sobre el cambio disponible.
- inventory: Información detallada sobre el inventario de monedas y billetes (si se realiza un arqueo).

---

sockets

Descripción: Este comando se envía para proporcionar información sobre las conexiones WebSocket activas y el estado de las transacciones.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "sockets"
- conexiones: El número de conexiones activas.
- cobrando: Indica si el servidor está procesando un cobro (true o false).
- Due: El monto debido en la transacción actual.
- Credit: El monto de crédito disponible.
- changeBar: Información sobre el cambio disponible.
- tarjeta: Indica si el pago con tarjeta está disponible (true o false).
- formaPago: La forma de pago seleccionada para la transacción actual.

---

updateCredit

Descripción: Este comando se envía para actualizar el monto de crédito disponible durante una transacción.

Parámetros:

- fecha: La fecha y hora en la que se envía el comando.
- comando: "updateCredit"
- Due: El monto debido en la transacción actual.
- Credit: El monto de crédito disponible.
- Status: El estado actual traducido al español.
- changeBar: Información sobre el cambio disponible.

APS Aplicaciones y Soluciones de Pago
Ramón Corona No. 289-C
Colonia Santa Anita
Tlaquepaque, Jalisco
C.P. 45600, México

Teléfono Oficina: +52 (33) 1871 6119
Correo Electrónico: **ventas@apsmx.com**
Página en Internet: **www.apsmx.com**