iOneTech

Getnet

Integración POS Getnet con Puntos de Venta

Manual de Integración Paso a Paso Para Javascript

INTRODUCCIÓN

El siguiente documento contempla una guía paso a paso para realizar la integración POS Integrado Getnet con cualquier software de caja front end en Javascript.

REQUERIMIENTOS MINIMOS

Se asume que integrado de la integración sea un profesional con experiencia en el desarrollo Javascript que pueda entender conceptos simples que permitan la integración de una librería desde la cual se podrán utilizar métodos específicos que permitan la comunicación con el POS Integrado Getnet; de manera que su sistema de caja pueda solicitar pagos con tarjetas de crédito y/o débito, esperar que el usuario final realice la transacción y recibir la respuesta y que sepa trabajar con métodos asíncronos.

Lo que se requiere para iniciar el proceso de integración sería lo siguiente:

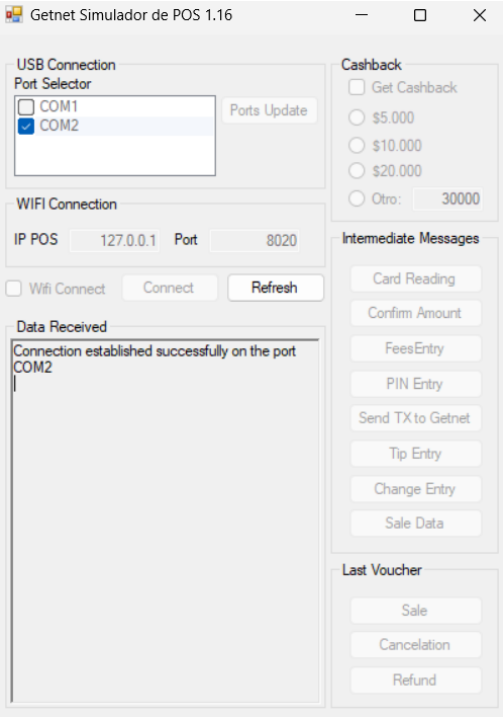
- Windows 10 o superior.
- Microsoft Visual Studio Code
- Extensión de visual studio code: "*Live Server Ritwick Dey*"
- Librería getnet
- Getnet Simulador de POS
- Dispositivo POS Getnet físico (no obligatorio)
- Cable usb tipo C (para el dispositivo POS Getnet físico)

PRUEBA DE SIMULADOR DE CAJA JAVASCRIPT

Para realizar la prueba con el simulador de caja javascript primeramente se debe tener iniciado el simulador de POS o en su defecto tener conectado un POS físico.

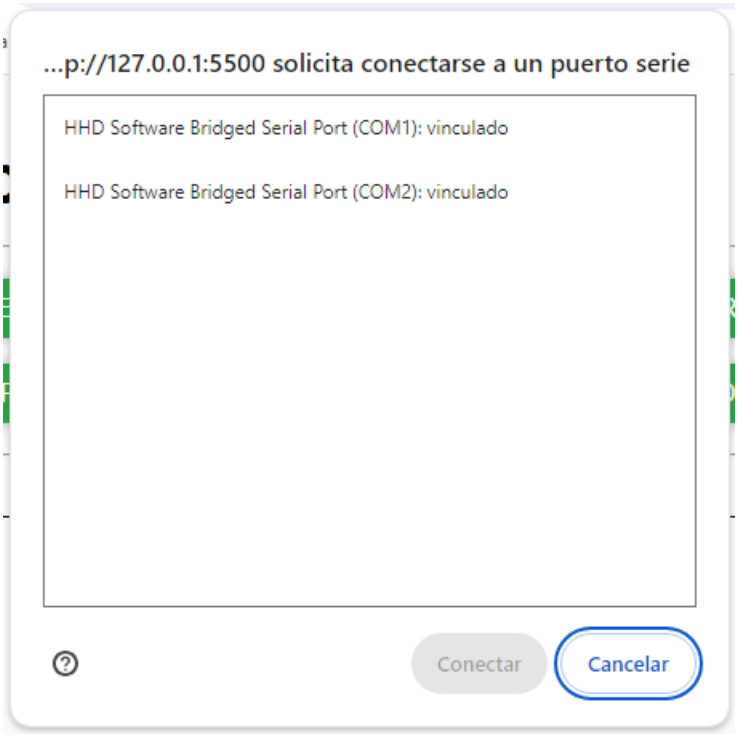
INICIAR SIMULADORES DE POS Y CAJA

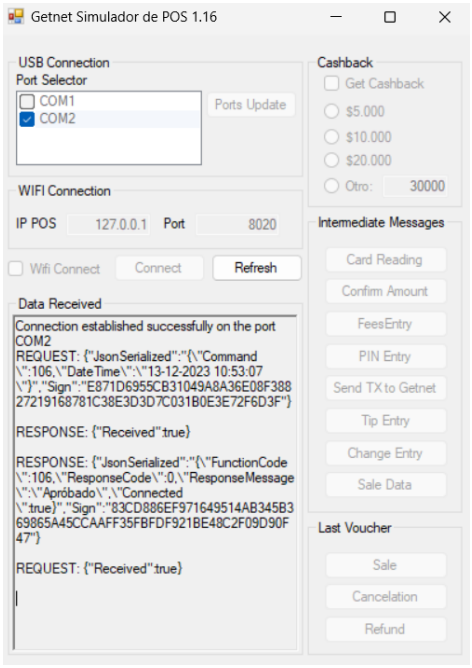
Iniciar el agente POS y establecer los puertos, luego abrir el simulador de POS y conectarlo en el puerto correspondiente, iniciar el servidor del simulador de caja de javascriptt (utilizando *"go live"* de la extensión de visual studio code "Live Server"*)*.



PROBAR FUNCIONAMIENTOS DE COMANDOS

Presionar el botón Poll para enviar comando de conexión, al presionar el botón se abrirá automáticamente un dialogo del navegador para seleccionar el puerto en el que se desea comunicar (Este paso es obligatorio de los navegadores).


...p://127.0.0.1:5500 solicita conectarse a un puerto serie
HHD Software Bridged Serial Port (COM1): vinculado
HHD Software Bridged Serial Port (COM2): vinculado
Conectar
Cancelar


Getnet Simulador de POS 1.16
USB Connection
Port Selector
COM1
COM2
Ports Update
WIFI Connection
IP POS
127.0.0.1
Port
8020
Wifi Connect
Connect
Refresh
Data Received
Connection established successfully on the port COM2
REQUEST: {"JsonSerialized":"{\"Command\":106,\"DateTime\":\"13-12-2023 10:53:07\"}","Sign":"E871D6955CB31049A8A36E08F38827219168781C38E3D3D7C031B0E3E72F6D3F"}
RESPONSE: {"Received":true}
RESPONSE: {"JsonSerialized":"{\"FunctionCode\":106,\"ResponseCode\":0,\"ResponseMessage\":\"Apróbado\",\"Connected\":true}","Sign":"83CD886EF971649514AB345B369865A45CCAAFF35FBFDF921BE48C2F09D90F47"}
REQUEST: {"Received":true}
Cashback
Get Cashback
$5.000
$10.000
$20.000
Otro:
30000
Intermediate Messages
Card Reading
Confirm Amount
FeesEntry
PIN Entry
Send TX to Getnet
Tip Entry
Change Entry
Sale Data
Last Voucher
Sale
Cancelation
Refund

PROCESO DE INTEGRACION PARA PROYECTOS JAVASCRIPT

Para el proceso de integración se entrega el proyecto de Simulador de Caja de manera que se puede revisar en su totalidad y así tener un ejemplo claro de cómo se puede modificar la aplicación de caja para incorporar la integración con getnet.

SERIAL COMMMUNICACION SERVER

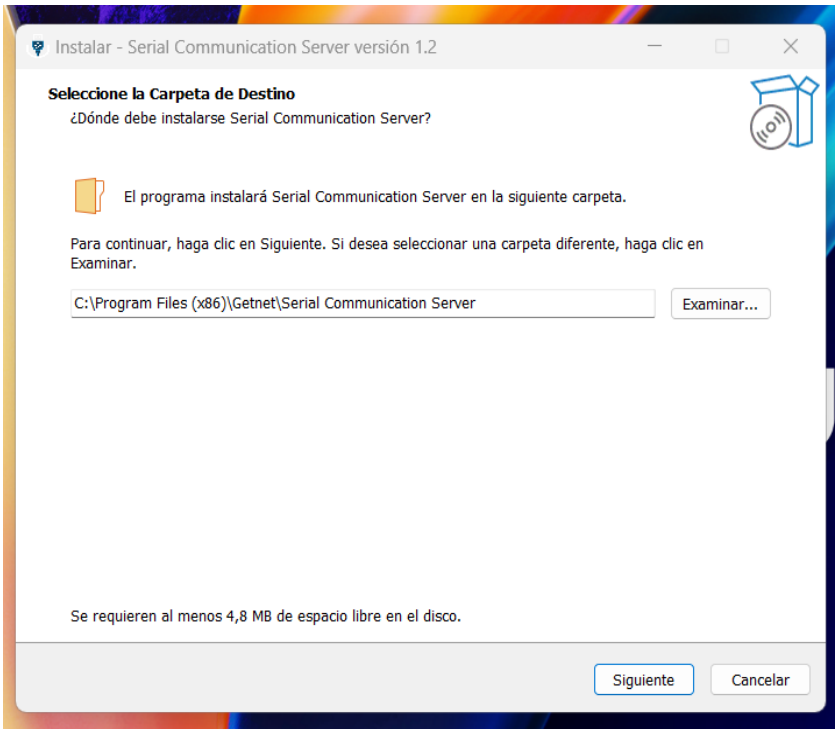
El software es un programa que se conecta a la librería de JavaScript vía websocket y sirve de intermediario de comunicación serial entre la librería y el POS Getnet, evitando así el dialogo de selección de puertos en el navegador.

PROCEDIMIENTO DE INSTALACIÓN

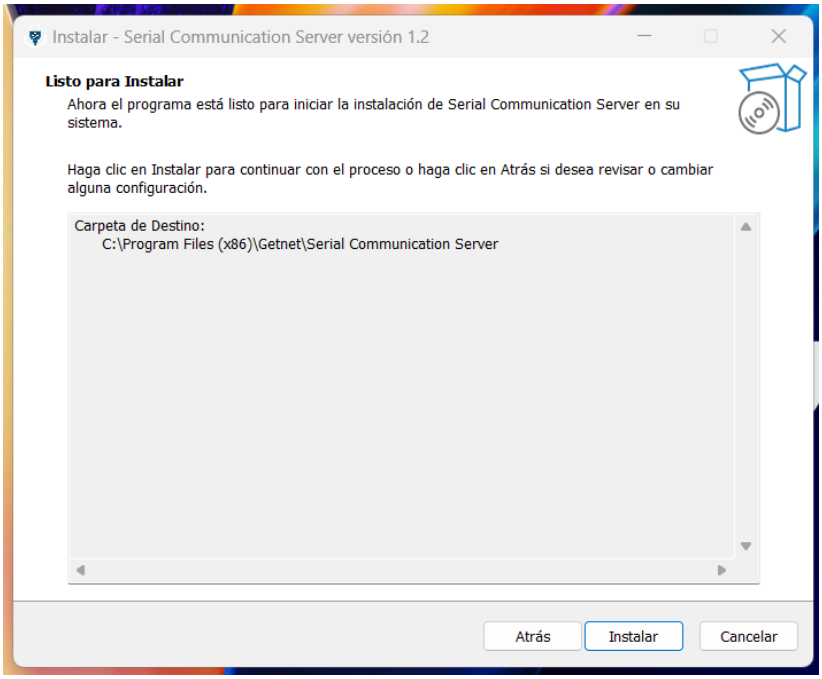
1. Ejecutar el instalador

2. En este paso, por defecto el programa se instalará en la carpeta

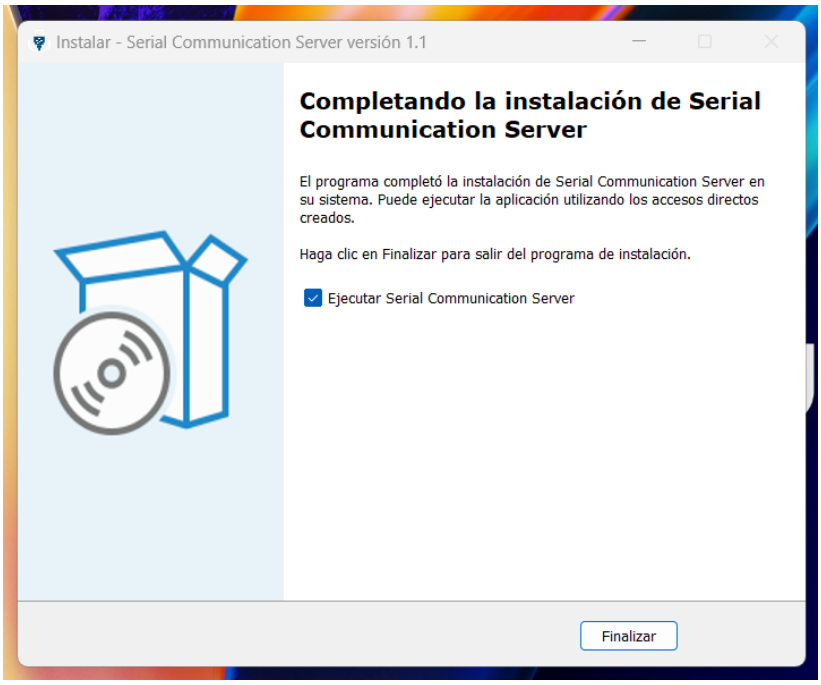
   C:\Program Files (x86)\Getnet\Serial Communication Server (Recomendado)
   Presionar Siguiente.



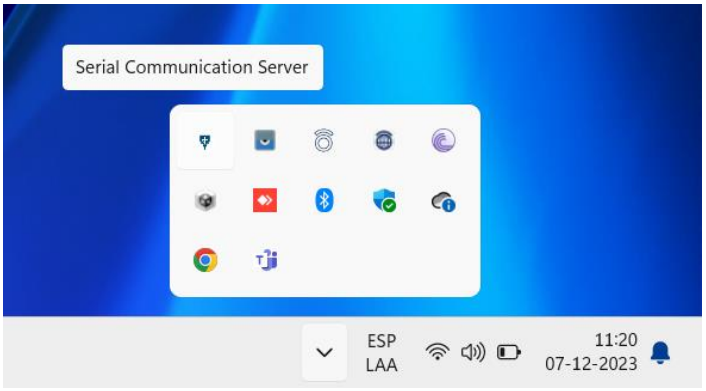
3. En este paso muestra un resumen y un mensaje de “listo para instalar”, solo se debe presionar “Instalar”



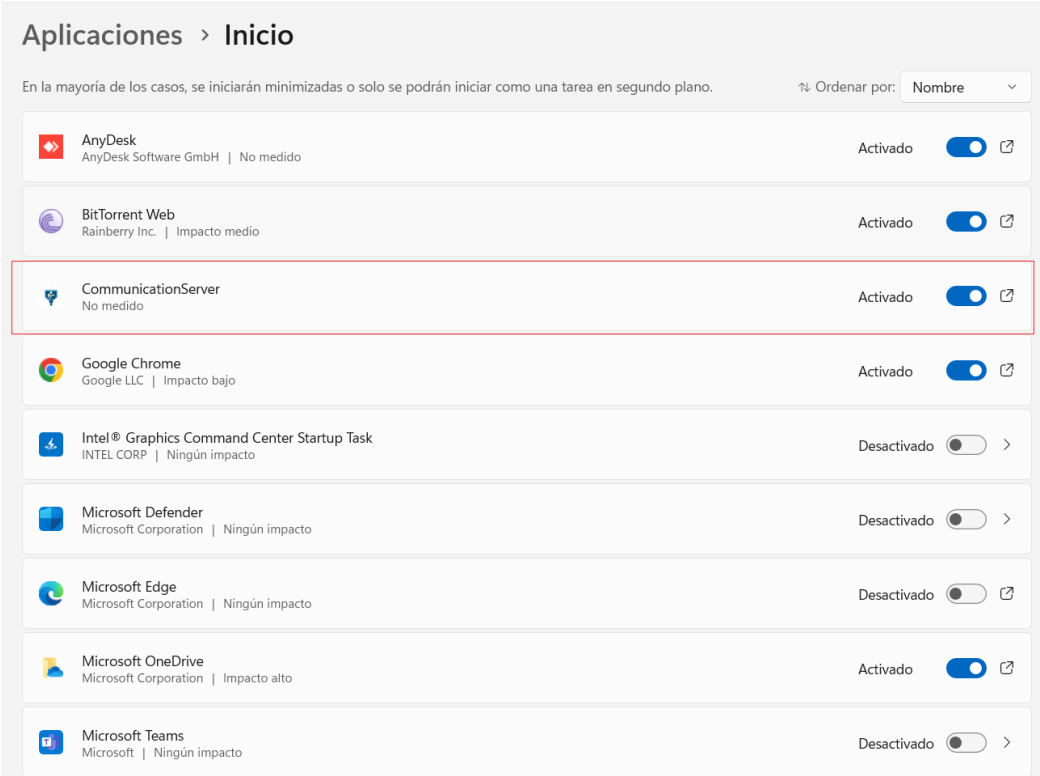
4. Finalmente, el mensaje de instalación exitosa. Recomendado dejar la casilla “Ejecutar Serial Communication Server” seleccionada para iniciar automáticamente el programa.



5. Luego revisar si el programa fue iniciado correctamente, en la sección oculta de las aplicaciones en la barra de tareas de Windows, es el icono de software primero a la izquierda en la imagen, al posicionarse con el cursor encima del icono debería desplegarse el nombre “Serial Communication Server”.



6. Finalmente debemos asegurarnos de que el programa esté en los programas de inicio de Windows. Para ello debemos ir a configuración de Windows, luego a aplicaciones, presionar inicio (aplicaciones que se inician automáticamente sin iniciar sesión) y ver que esté registrado el programa ahí.



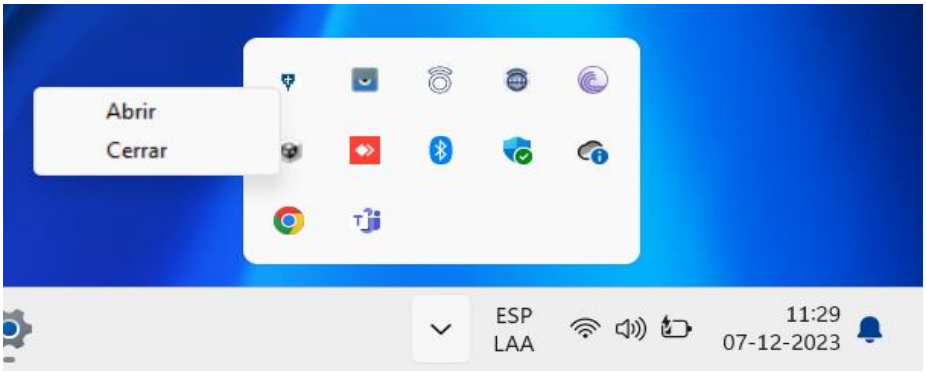
En caso de que no esté debemos activarlo, así el programa se iniciará automáticamente al iniciar Windows.

FUNCIONAMIENTO SERIAL COMMUNICATION SERVER

El software es un programa que se conecta a la librería de JavaScript vía websocket y sirve de intermediario de comunicación serial entre la librería y el POS Getnet. El programa está diseñado para funcionar en segundo plano oculto, ninguna configuración es necesaria, solo que esté instalado y andando. Dado que se conectará automáticamente al POS al enviar comandos desde el frontend.

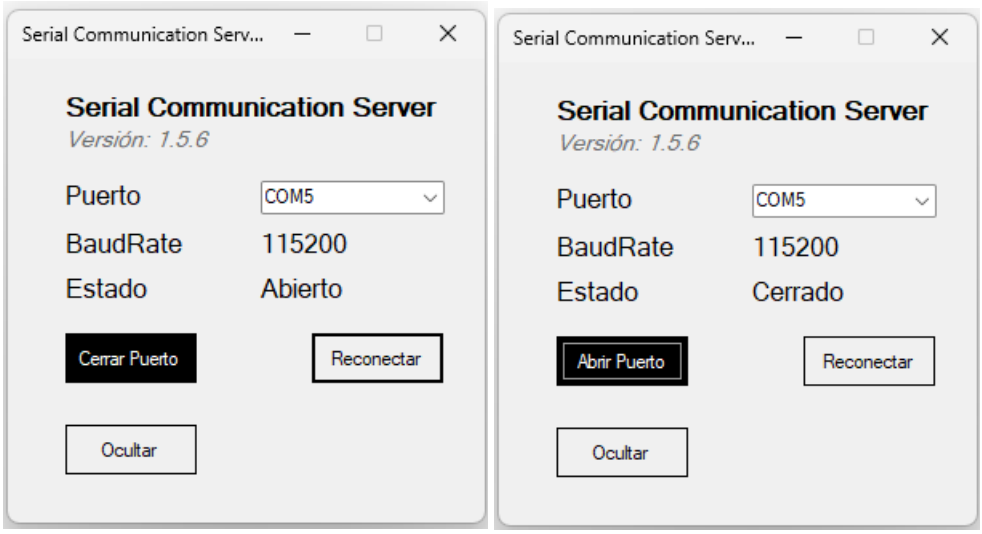
A continuación las funciones del programa.

1. Al hacer click derecho se desplegará una pequeña venta donde hay dos opciones “Abrir” y “Cerrar”.



2. Al presionar “Abrir”, se abrirá la ventana.
3. La ventana contiene los siguientes datos:

a. Puerto: puerto COM seleccionado, al desplegar el listado puede seleccionar otro puerto manualmente (no recomendado)
b. BaudRate: Velocidad de transmisión de los datos (por defecto siempre será 115200)
c. Estado: Estado del puerto COM, puede estar abierto o cerrado
d. El botón “Cerrar Puerto” | “Abrir Puerto”: este botón cerrará el puerto o abrirá dependiendo del estado



e. El botón Reconectar: Este botón rebuscará el POS en los puertos COM y al encontrarlo lo re asignará al puerto seleccionado.
f. Botón Ocultar: tanto al presionar este botón, minimizar la ventana o cerrarla desde la ventana el programa seguirá funcionando ocultado en la barra de tareas.
g. La única forma de cerrar el programa es presionando “cerrar” en el dialogo mencionado en el punto 1.

LIBRERÍA

El primer paso es agregar el archivo getner.min.js en la raíz (puede ser agregada en cualquier carpeta, en caso de usar un framework como react js no ingresar el archivo dentro de la carpeta node_modules porque no es un módulo de node).

AGREGAR LA LIBRERÍA EN LA SECCIÓN IMPORT

Paga agregar la librería se debe importar al inicio del documento javascript

```
import Getnet from "./getnet/getnet.min.js";
```

ESTABLCER METODO DE COMUNICACIÓN SERIAL

Comunicación vía Navegador

La comunicación vía navegador está por defecto entonces no es requerido ningún método para establecerlo, solo asegurarse que el método que establece la comunicación vía Serial Communication Server no esté establecido

Comunicación vía software Serial Communicación Server

Para establecer que la comunicación serial se realizará a través de software Serial Communication Server se debe agregar la siguiente línea de código:

```
Getnet.establecerWebSerialCommunication();
```

ESTABLECER CALLBACKS DE COMUNICACIÓN

```
Getnet.SetCallback(mostrarResponse);
Getnet.SetLogCallback(logCallbackBackend);
Getnet.SetTimeErrorCallback(timeOutCallback);
```

Explicación:

- Getnet: un objeto que contiene todo lo que se necesita de los modulos, las funciones y los métodos callbacks.
- SetCallback: El método SetCallback establece el método que recibirá la respuesta del POS.
- SetLogCallback: Este método establece el sistema para guardar el log de lo que se envía y recibe, obtiene siempre lo que se envía y recibe en formato de texto.
- SetTimeErrorCallback: Este método establece una función que será llamada en caso de que el timeout de respuesta del POS se complete, ahí se establece que notificación o handler se usará en caso de que se cumpla el timeout (El timeout puede completarse en caso de que no haya comunicación con el POS).

La librería maneja el control de los puertos automáticamente al enviar cualquier comando, no es necesario “abrir puerto” o “cerrarlo” todo lo maneja de manera automatizada la librería.

ENVIAR SOLICITUDES AL POS

El envío de comandos al POS Getnet es tan simple como llamar a un método y la librería se encarga de todo el procesamiento de los datos para construir el JSON, firmar los datos y enviar el mensaje.

Polling

```
try {
    Getnet.Poll();
} catch (error) {
    callback(error.message);
}
```

Sale

```
try {

    Getnet.Sale(1000, 1, true, Getnet.POSCommands.SaleType.Sale, true, 1, );
} catch (error) {

    callback(error.message);
```

```
}
```

Last Voucher

```
try {
    Getnet.LastVoucher(true);
} catch (error) {
    callback(error.message);
}
```

Cancelation

```
try {
    Getnet.Refund(1, true);
} catch (error) {
    callback(error.message);
}
```

Close

```
try {
    Getnet.Close(true);
} catch (error) {
    callback(error.message);
}
```

Totals

```
try {
    Getnet.Totals(true);
} catch (error) {
    callback(error.message);
}
```

Details

```
try {
    Getnet.Details(true);
} catch (error) {
```

```
    callback(error.message);
}
```

Normal Mode

```
try {
    Getnet.SetNormalMode();
} catch (error) {
    callback(error.message);
}
```

Refund

```
try {
    Getnet.Return(1, 100, true);
} catch (error) {
    callback(error.message);
}
```

Duplicate

```
try {
    Getnet.DuplicateOthers(1, true);
} catch (error) {
    callback(error.message);
}
```

Seller Sales

```
try {
    Getnet.SalesBySeller(1, true);
} catch (error) {
    callback(error.message);
}
```

Tip Report

```
try {
    Getnet.TipReport(1, true);
```

```
} catch (error) {
    callback(error.message);
}
```

Default Sale

```
try {
    Getnet.DefaultSaleType(POSCommands.SaleType.Sale);
} catch (error) {
    callback(error.message);
}
```

Params

```
try {
    Getnet.ParameterReport(true);
} catch (error) {
    callback(error.message);
}
```

Sim Report

```
try {
    Getnet.SimReport(true);
} catch (error) {
    callback(error.message);
}
```

Cancel Sale

```
try {
    Getnet.CancelSale();
} catch (error) {
    callback(error.message);
}
```

NOTA: Todos los métodos expuestos aquí tienen una sobrecarga, lo que significa que se pueden enviar en su forma más simple (como se exponen aquí) o bien agregarle parámetros como el PrintOnPos o Seconds (para controlar el timeout).

CASTEO DE OBJETOS RESPONSE

En la función callback que se define en `Getnet`(la cual puede ser cambiada en cualquier momento) sirve para recibir los datos y hacer algo con ello.

Para poder saber el tipo de respuesta que recibe callback se puede comprobar que tipo es y luego hacer lo que se requiera con ello.

Obtener el tipo de objeto

Primeramente se obtiene el tipo de objeto, con ello luego podemos hacer la consulta de qué tipo de respuesta que se obtuvo y así poder hacer el casteo.

Received

Esta respuesta "{ Received: true }" es la respuesta de conexión del POS, por ende puede agregar una lógica o no.

```
function callback(datos) {
    if (datos.Received)
        return;
    // Más Código
```

Obtener el objeto y revisar el tipo de respuesta

Luego se obtiene el JSON y se necesita deserializar para poder revisar sus datos.

```
function callback(datos) {
    if (datos.Received)
        return;
    const respuesta = JSON.parse(datos.JsonSerialized);
    switch (respuesta.FunctionCode) {
      // Más Código
```

Polling

```
case Getnet.POSCommands.Function.Poll: {

    // Código

    break;

}
```

Sale, Last Voucher, Duplicate

```
case Getnet.POSCommands.Function.Sale: {

    // Código

    break;
```

```
}
```

Cancelation

```
case Getnet.POSCommands.Function.Refund: {

    // Código

    break;

}
```

Close

```
case Getnet.POSCommands.Function.Close: {

    // Código

    break;

}
```

Totals

```
case Getnet.POSCommands.Function.Totals: {

    // Código

    break;

}
```

Details

```
case Getnet.POSCommands.Function.Details: {

    // Código

    break;

}
```

Normal Mode

```
case Getnet.POSCommands.Function.SetNormalMode: {

    // Código

    break;

}
```

Refund

```
case Getnet.POSCommands.Function.Return: {

    // Código

    break;
```

```
}
```

Seller Sales

```
case Getnet.POSCommands.Function.SalesBySeller: {

    // Código

    break;

}
```

Tip Report

```
case Getnet.POSCommands.Function.TipReport: {

    // Código

    break;

}
```

Default Sale

```
case Getnet.POSCommands.Function.DefaultSaleType: {

    // Código

    break;

}
```

Params

```
case Getnet.POSCommands.Function.ParameterReport: {

    // Código

    break;

}
```

Sim Report

```
case Getnet.POSCommands.Function.SimReport: {

    // Código

    break;

}
```

Cancel Sale

```
case Getnet.POSCommands.Function.CancelSale: {

    // Código

    break;
```

```
}
```

Ejemplo

Aquí proponemos una manera de hacerlo. Se parsea la respuesta y se establece un switch para poder ejecutar el comando que se requiera.

NOTA: No es necesario leer el resultado del puerto serial y deserializarlo para procesarlo de forma manual, la librería ya hace esto.

```
function callback(datos) {
    if (datos.Received)
        return;
    // Más Código
    const respuesta = JSON.parse(datos.JsonSerialized);
    switch (respuesta.FunctionCode) {
        case Getnet.POSCommands.Function.Poll: {
            // Código
            break;
        }
        case Getnet.POSCommands.Function.Sale: {
            // Código
            break;
        }
        case Getnet.POSCommands.Function.Refund: {
            // Código
            break;
        }
        case Getnet.POSCommands.Function.Close: {
            // Código
            break;
        }
        case Getnet.POSCommands.Function.Totals: {
            // Código
            break;
        }
        case Getnet.POSCommands.Function.Details: {
            // Código
            break;
        }
        case Getnet.POSCommands.Function.SetNormalMode: {
```

```
            // Código
            break;
        }
        case Getnet.POSCommands.Function.Return: {
            // Código
            break;
        }
        case Getnet.POSCommands.Function.SalesBySeller: {
            // Código
            break;
        }
        case Getnet.POSCommands.Function.TipReport: {
            // Código
            break;
        }
        case Getnet.POSCommands.Function.DefaultSaleType: {
            // Código
            break;
        }
        case Getnet.POSCommands.Function.ParameterReport: {
            // Código
            break;
        }
        case Getnet.POSCommands.Function.SimReport: {
            // Código
            break;
        }
        case Getnet.POSCommands.Function.CancelSale: {
            // Código
            break;
        }
        default: {
            // Código
            break;
        }
    }
}
```