Folletto Kiosk DLL Documentation

Version

v0.0.9

What's new:

- Added a function for the Kiosk client to publish error.
- Compiled with MBCS (Only for Jet Cafe Kiosk Client)

Overview

`FollettoKioskApi` is a C++ Dynamic Link Library designed to facilitate and enhance interactions between Kiosk systems and Robot Barista Booths. This library streamlines the process of data exchange and event handling, providing a efficient framework for kiosk-to-booth communication.

Key features of `FollettoKioskApi` include:

1. **Seamless Data Requesting:** It offers a variety of methods to efficiently request and retrieve data from Robot Barista Booths, ensuring timely and accurate communication between systems.
2. **Advanced Event Handling:** The library is equipped to manage and respond to a wide range of events originating from the booth, enabling kiosks to react dynamically to real-time scenarios.

Getting Started

Installation

1. Download provided `FollettoKioskApi.dll` and `FollettoKioskApi.lib`
2. Place the DLL in your project's directory.
3. Add `FollettoKioskApi.lib` to your linker in the project settings.

Dependencies

- Requires Microsoft Visual C++ Runtime.
- Compatible with C++17 standard and later.

DLL Compilation Details

- Character set:
  - Default: Unicode
  - Jet Cafe Kiosk Client: MBCS
- Architecture:
  - 32-bit
  - 64-bit

Core Functionality

> Initialize server

The `initializeServer` is a function provided by the DLL that initializes and starts the Http server inside the DLL.

Function Signature

```
ServerInitializationErrorCode initializeServer(int port=5000);
```

Parameters

- `port` (`int`): Http server port. Default: `5000`.

Return Value

- `ServerInitilizationErrorCode`: An enumeration value representing the outcome of the server initialization attempt.

Error Codes

```
enum class ServerInitializationErrorCode {
    Success = 0,
    InvalidPort,
    PortUnavailable,
    InternalError,
}
```

The `ServerInitializationErrorCode` enumeration includes the following values:

- `Success`: Server initialized successfully.
- `InvalidPort`: Port number is invalid or malformed.
- `PortUnavailable`: Provided port is unavailable or already in use.
- `InternalError`: Internal Error.

> Connect

The `connectToBooths` method is a function provided by the DLL that establishes a network connection to the main program of the barista robot booth with specified IP and Port. This function establishes connection to the booth and perform necessary actions (e.g. download menu images).

Function Signature

```
ConnectErrorCode connectToBooths(const IPAddressPort* ipAddresses, int count);
```

Parameters

- `ipAddresses` (`const IPAddressPort*`): A pointer to the first element in an array of `IPAddressPort` structures. Each structure contains an IP address as a null-terminated string and a port number. For current version, only the first item of the array will be considered.

  ```
  struct IPAddressPort {
   char ipAddress[16];
   unsigned short port;
  };
  ```

  - `ipAddress`: IP address of a booth.
  - `port`: Port of a booth.

- `count` (`int`): The number of elements in the `ipAddresses` array, indicating how many endpoints the function should attempt to connect to.

Return Value

- `ConnectErrorCode`: An enumeration value representing the outcome of the connection attempt. The value `Success` indicates a successful connection, while other values indicate specific types of errors.

Error Codes

```
enum ConnectErrorCode {
    Success = 0,
    InvalidIPAddress,
    InvalidPort,
    IPAddressNotAvailable,
    PortUnavailable,
    ConnectionTimeout,
    NetworkUnreachable,
    ConnectionRefused,
    ResourceLimitation,
    NetworkProtocolError,
    InternalError,
    SecurityRestrictions,
    HostNameResolutionFailure,
    SocketInitializationFailure,
};
```

The `ConnectErrorCode` enumeration includes the following values:

- `Success` (0): Connection established successfully.
- `InvalidIPAddress` (1): The provided IP address is invalid or malformed.
- `InvalidPort` (2): The provided port number is invalid.
- `IPAddressNotAvailable` (3): The specified IP address is not available.
- `PortUnavailable` (4): The specified port on the target IP address is not open or not listening for connections.
- `ConnectionTimeout` (5): The connection attempt timed out, possibly due to network latency or the server not responding.
- `NetworkUnreachable` (6): The network required to reach the specified IP is unreachable.
- `ConnectionRefused` (7): The connection was actively refused by the server.
- `ResourceLimitation` (8): System-level resource limitations prevent establishing a new connection (e.g., too many open sockets).
- `NetworkProtocolError` (9): Errors related to the underlying network protocol.
- `InternalError` (10): Internal Error.
- `SecurityRestrictions` (11): Connection blocked due to security policies, firewall restrictions, or lack of necessary permissions.
- `HostNameResolutionFailure` (12): Failure to resolve a hostname to an IP address.
- `SocketInitializationFailure` (13): Failure in initializing the network socket, possibly due to issues with the network stack or incorrect socket configuration.

Example Usage

Python3

```
import ctypes

class IPAddressPort(ctypes.Structure):
    _fields_ = [
        ("ipAddress", ctypes.c_char * 16),
        ("port", ctypes.c_ushort),
    ]

dll.connectToBooths.argtypes = [ctypes.POINTER(IPAddressPort), ctypes.c_int]

# Load the DLL
dll = ctypes.CDLL('/path/to/dll_file.dll')

# Define the argument and return types for the connect function
dll.connectToBooths.argtypes = [ctypes.POINTER(IPAddressPort), ctypes.c_int]
dll.connectToBooths.restype = ctypes.c_int  # ConnectErrorCode is returned as an
int

# Prepare the IPAddressPort array
ip_addresses = (IPAddressPort * 2)()  # Example with 2 IP addresses
ip_addresses[0].ipAddress = b"192.168.1.1"  # Byte string for the IP address
ip_addresses[0].port = 8080  # Port number as integer
ip_addresses[1].ipAddress = b"192.168.1.2"  # Second IP address
```

```
ip_addresses[1].port = 8080

# Call the connect function with the array
result_code = dll.connectToBooths(ip_addresses, 2)
if result_code == 0:
    print("Connection successful.")
else:
    print(f"Connection failed with error code: {result_code}")
```

C#

```
using System;
using System.Runtime.InteropServices;

[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi)]
public struct IPAddressPort
{
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 16)]
    public string ipAddress;
    public ushort port;
}

[DllImport("path_to_your_dll.dll", CallingConvention = CallingConvention.Cdecl)]
private static extern int connectToBooths([MarshalAs(UnmanagedType.LPArray,
SizeParamIndex = 1)] IPAddressPort[] ipAddresses, int count);
```

```
using System;
using System.Runtime.InteropServices;

class Program
{
    [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi)]
    public struct IPAddressPort
    {
        [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 16)]
        public string ipAddress;
        public ushort port;
    }

    [DllImport("path_to_your_dll.dll", CallingConvention =
CallingConvention.Cdecl)]
    private static extern int connectToBooths([MarshalAs(UnmanagedType.LPArray,
SizeParamIndex = 1)] IPAddressPort[] ipAddresses, int count);

    static void Main()
    {
        IPAddressPort[] ipAddresses = new IPAddressPort[]
        {
            new IPAddressPort { ipAddress = "192.168.1.1", port = 8080 },
            new IPAddressPort { ipAddress = "192.168.1.2", port = 8080 }
            // Add more IP addresses and ports as needed
        };
```

```
        int resultCode = connectToBooths(ipAddresses, ipAddresses.Length);
        if (resultCode == 0)
        {
            Console.WriteLine("Connection successful.");
        }
        else
        {
            Console.WriteLine($"Connection failed with error code:
{resultCode}");
        }
    }
}
```

Java

```
import com.sun.jna.*;

public class IPAddressPort extends Structure {
    public static class ByReference extends IPAddressPort implements
Structure.ByReference {}

    public String ipAddress;
    public short port;

    @Override
    protected List<String> getFieldOrder() {
        return Arrays.asList("ipAddress", "port");
    }
}
```

```
public interface MyLibrary extends Library {
    MyLibrary INSTANCE = (MyLibrary) Native.load("path_to_your_dll",
MyLibrary.class);
    int connectToBooths(IPAddressPort.ByReference[] ipAddresses, int count);
}
```

```
import com.sun.jna.Library;
import com.sun.jna.Native;
import com.sun.jna.Structure;

public class Main {
    public interface MyLibrary extends Library {
        MyLibrary INSTANCE = (MyLibrary) Native.load("path_to_your_dll",
MyLibrary.class);
        int connectToBooths(IPAddressPort.ByReference[] ipAddresses, int count);
    }

    public static void main(String[] args) {
        IPAddressPort.ByReference[] ipAddresses = (IPAddressPort.ByReference[])
new IPAddressPort.ByReference().toArray(2);

        ipAddresses[0].ipAddress = "192.168.1.1";
        ipAddresses[0].port = 8080;
        ipAddresses[1].ipAddress = "192.168.1.2";
```

```
        ipAddresses[1].port = 8080;

        int resultCode = MyLibrary.INSTANCE.connectToBooths(ipAddresses,
ipAddresses.length);
        if (resultCode == 0) {
            System.out.println("Connection successful.");
        } else {
            System.out.println("Connection failed with error code: " +
resultCode);
        }
    }
}
```

C++

```
#include <iostream>

struct IPAddressPort {
    char ipAddress[16];
    unsigned short port;
};
```

```
extern "C" int connectToBooths(const IPAddressPort* ipAddresses, int count);
```

```
#include <iostream>
#include <cstring>

struct IPAddressPort {
    char ipAddress[16];
    unsigned short port;
};

extern "C" int connectToBooths(const IPAddressPort* ipAddresses, int count);

int main() {
    IPAddressPort ipAddresses[] = {
        {"192.168.1.1", 8080},
        {"192.168.1.2", 8080}
        // Add more IP addresses and ports as needed
    };

    int count = sizeof(ipAddresses) / sizeof(IPAddressPort);
    int resultCode = connectToBooths(ipAddresses, count);

    if (resultCode == 0) {
        std::cout << "Connection successful." << std::endl;
    } else {
        std::cout << "Connection failed with error code: " << resultCode <<
std::endl;
    }

    return 0;
}
```

JavaScript (e.g. Electron App)

Install `ffi-napi`

```
npm install ffi-napi
```

Using DLL with `ffi-napi`

```
const IPAddressPort = StructType({
  ipAddress: 'char[16]',
  port: 'ushort'
});
```

```
const myDll = ffi.Library('path_to_your_dll', {
  'connectToBooths': ['int', [ref.refType(IPAddressPort), 'int']],
});
```

```
const ffi = require('ffi-napi');
const ref = require('ref-napi');
const ArrayType = require('ref-array-di')(ref);
const StructType = require('ref-struct-di')(ref);

const IPAddressPort = StructType({
  ipAddress: 'char[16]', // Adjust for IPv6 if needed
  port: 'ushort'
});

const IPAddressPortArray = ArrayType(IPAddressPort);

const myDll = ffi.Library('path_to_your_dll', {
  'connectToBooths': ['int', [ref.refType(IPAddressPort), 'int']],
});

// Create an array of IPAddressPort structures
let ipAddresses = new IPAddressPortArray(2);
ipAddresses[0].ipAddress = "192.168.1.1";
ipAddresses[0].port = 8080;
ipAddresses[1].ipAddress = "192.168.1.2";
ipAddresses[1].port = 8080;

// Convert array to pointer and call the function
const result = myDll.connectToBooths(ipAddresses.ref(), ipAddresses.length);

if (result === 0) {
  console.log("Connection successful.");
} else {
  console.log(`Connection failed with error code: ${result}`);
}
```

> Get DLL Version

The `getDllVersion` function retrieves the current DLL version.

Function Signature

```
const char* getDllVersion();
```

Parameters

This function does not require any input parameters.

Return Value

- `version` `char*`: DLL version.

  > The `version` is of form `MAJOR.MINOR.PATCH` where,
  >
  > `MAJOR` **version** indicates incompatible API changes. This number is incremented when changes are made that break backward compatibility with previous versions.
  >
  > `MINOR` **version** denotes the addition of functionality in a backward-compatible manner.
  >
  > `PATCH` **version** is for backward compatible bug fixes that do not add new feature or change existing ones.

> Get Booth Configuration

The `getBoothConfiguration` function retrieves the current booth configurations.

Function Signature

```
BoothConfig* getBoothConfig();
```

Parameters

This function does not require any input parameters.

Return Value

- `BoothConfig`: A structure containing the current configuration settings of the booth.

  > `BoothConfig` structure.
  >
  > - `qrRequired` (`Boolean`): `true` if QR code is required at beverage pickup, `false` otherwise. If set to `true`, Kiosk should print a QR code with QR data (see `Place Order` section of this document) on the receipt for the customer to use it for beverage pickup.
  > - `maxProductsInGroup` (`integer`): maximum number of products in a single order. E.g. when set to `3` the maximum number of products that can be ordered at once (in a single group) should be limited to 3.
  >   -> For example, if `maxProductsInGroup` is equal to 3:

        - **Accepted scenario:** a single order containing a list of 3 products.
        - **Not accepted:** a single order containing less than 1 product or more than 3 products.
    - `languages` (`AvailableLanguage`): Array of available language struct. See `AvailableLanguage` struct below.

        The structure of `AvailableLanguage` is as such:

        ```
        struct AvailableLanguage
        {
            char id[3];             // e.g. "en", "kr", "jp", "es"
            char display[128];      // e.g. "English", "한국어", "日本語",
        "Español"
        };
        ```

        - `id`: (`char*`): unique id of the language. This `id` will be used as a suffix for fields in Product and options. See Get Product Status.
        - `display` (`char*`): shows how this language will be displayed (UTF-8).
    - `languagesCount` (`int`): to keep track of the number of languages
    - `offlineMode` (`bool`): `true` if the booth is in offline mode, `false` otherwise
    - `fullyAutomaticMode` (`bool`): `true` if the booth is in fully automatic mode, `false` otherwise

Special Note

The DLL provides `freeBoothConfig` method to free the memory allocated. The `freeBoothConfig` function has the following function signature:

```
void freeBoothConfig(BoothConfig* config);
```

Example Usage

Python3

```
import ctypes
from ctypes import c_int, c_char, c_bool, POINTER, Structure

# Define the C structures in Python
class AvailableLanguage(Structure):
    _fields_ = [
        ("id", c_char * 50),
        ("display", c_char * 256),
    ]

class BoothConfig(Structure):
    _fields_ = [
        ("qrRequired", c_bool),
        ("maxProductsInGroup", c_int),
        ("languages", POINTER(AvailableLanguage)),
        ("languagesCount", c_int),
        ("offlineMode", c_bool),
```

```
        ("fullyAutomaticMode", c_bool),
    ]

# Load the DLL
follett_kiosk_dll = ctypes.CDLL('./FollettKiosk.dll')

# Specify the return types of the functions if they are not int
follett_kiosk_dll.getBoothConfig.restype = BoothConfig

# Call getBoothConfig
get_booth_config = follett_kiosk_dll.getBoothConfig
config = get_booth_config()

# Access the BoothConfig data
print(f"QR Required: {config.qrRequired}")
print(f"Max Products In Group: {config.maxProductsInGroup}")
print(f"Languages Count: {config.languagesCount}")

for i in range(config.languagesCount):
    language = config.languages[i]
    print(f"Language {i}: ID={language.id.decode('utf-8')}, Display=
{language.display.decode('utf-8')}")

# When done, free the allocated memory
free_booth_config = follett_kiosk_dll.freeBoothConfig
free_booth_config(ctypes.byref(config))
```

C#

```
using System;
using System.Runtime.InteropServices;

[StructLayout(LayoutKind.Sequential)]
public struct BoothConfig
{
    [MarshalAs(UnmanagedType.I1)]
    public bool qrRequired;
}

class Program
{
    [DllImport("your_dll.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern BoothConfig getBoothConfig();

    static void Main()
    {
        BoothConfig config = getBoothConfig();
        if (config.qrRequired)
        {
            // Handle QR code requirement
        }
        // Handle other configurations
    }
}
```

Java

```
import com.sun.jna.*;

public class Main {
    public static class BoothConfig extends Structure {
        public boolean qrRequired;
        // ... other fields ...
    }

    public interface MyLibrary extends Library {
        MyLibrary INSTANCE = Native.load("your_dll_name", MyLibrary.class);
        BoothConfig getBoothConfig();
    }

    public static void main(String[] args) {
        BoothConfig config = MyLibrary.INSTANCE.getBoothConfig();
        if (config.qrRequired) {
            System.out.println("QR code scanning is required.");
        }
        // Handle other configurations
    }
}
```

C++

```
int main() {
    BoothConfig config = getBoothConfig();
    if (config.qrRequired) {
        // Handle QR code requirement
    }
    // Handle other configurations
}
```

JavaScript (e.g. Electron App)

Install `ffi-napi`

```
npm install ffi-napi
```

Using DLL with `ffi-napi`

```
const ffi = require('ffi-napi');
const ref = require('ref-napi');
const Struct = require('ref-struct-di')(ref);

const BoothConfig = Struct({
    'qrRequired': 'bool',
    // ... other fields ...
});
const myDll = ffi.Library('path_to_your_dll', {
    'getBoothConfig': [BoothConfig, []]
});
```

```
const config = myDll.getBoothConfig();
if (config.qrRequired) {
    console.log("QR code scanning is required.");
}
// Handle other configurations
```

> Register Booth Configuration Update Callback

The `registerBoothConfigurationUpdateCallback` function is used to register booth configuration update callback function for notification when booth configuration update occurs.

Function Signature

```
void registerBoothConfigUpdateCallback(BoothConfigUpdateCallback callback);
```

Parameters

- **callback** (`BoothConfigUpdateCallback`): A pointer to the client's callback function. This function should conform to the `BoothConfigUpdateCallback` signature, which is a function pointer type with no parameters and no return value.

Callback Function Signature

```
int (*BoothConfigUpdateCallback)();
```

Callback Function Return Value

- `int` indicating configuration update success/fail

  > `0`: success.
  >
  > `1`: Fail.

Example Usage

Python3

```
import ctypes

# Define the callback type
BoothConfigUpdateCallback = ctypes.CFUNCTYPE(None)

# Load the DLL and define argument types
dll = ctypes.CDLL('/path/to/your_dll.dll')
dll.registerBoothConfigUpdateCallback.argtypes = [BoothConfigUpdateCallback]

# Python function to handle the callback
@ctypes.CFUNCTYPE(None)
def my_booth_config_update_handler():
    print("Booth configuration updated.")
```

```
# Register the callback
callback = BoothConfigUpdateCallback(my_booth_config_update_handler)
dll.registerBoothConfigUpdateCallback(callback)
```

C#

```
using System;
using System.Runtime.InteropServices;

class Program
{
    [UnmanagedFunctionPointer(CallingConvention.Cdecl)]
    private delegate void BoothConfigUpdateCallback();

    [DllImport("your_dll.dll", CallingConvention = CallingConvention.Cdecl)]
    private static extern void
registerBoothConfigUpdateCallback(BoothConfigUpdateCallback callback);

    static void MyBoothConfigUpdateHandler()
    {
        // Handle the configuration update
        Console.WriteLine("Booth configuration updated.");
    }

    static void Main()
    {
        BoothConfigUpdateCallback callbackDelegate = MyBoothConfigUpdateHandler;
        registerBoothConfigUpdateCallback(callbackDelegate);

        // Rest of your client code
    }
}
```

Java

```
import com.sun.jna.*;
import com.sun.jna.ptr.*;

public class Main {
    // Define a callback interface matching the C++ callback signature
    public interface BoothConfigUpdateCallback extends Callback {
        void invoke();
    }

    public interface MyLibrary extends Library {
        MyLibrary INSTANCE = Native.load("your_dll_name", MyLibrary.class);
        void registerBoothConfigUpdateCallback(BoothConfigUpdateCallback
callback);
    }

    public static void main(String[] args) {
        BoothConfigUpdateCallback callback = new BoothConfigUpdateCallback() {
```

```
            public void invoke() {
                System.out.println("Booth configuration updated.");
            }
        };

        MyLibrary.INSTANCE.registerBoothConfigUpdateCallback(callback);

        // ... rest of your code ...
    }
}
```

JavaScript

Install `ffi-napi`

```
npm install ffi-napi
```

Using DLL with `ffi-napi`

```
const ffi = require('ffi-napi');

const myDll = ffi.Library('path_to_your_dll', {
    'registerBoothConfigUpdateCallback': ['void', ['pointer']]
});

const callback = ffi.Callback('void', [], () => {
    console.log("Booth configuration updated.");
});

myDll.registerBoothConfigUpdateCallback(callback);

// Keep the callback reference to prevent it from being garbage-collected
process.on('exit', () => {
    callback
});
```

> Notify Kiosk-side error

The `notifyKioskError` function publishes the error occurred from the Kiosk.

Function Signature

```
int notifyKioskError(char* errorCodes[], int size);
```

Parameters

- `errorCodes`: an array of error code strings.
- `size`: the size of elements in the error array.

Return Value:

- `int` indicating success/fail:

  > `0`: Success
  >
  > `1`: Fail

> Get Number of Waiting Orders

The `getNumberOfWaitingOrders` function retrieves the current number of waiting orders.

Function Signature

```
int getNumberOfWaitingOrders();
```

Parameters

This function does not require any input parameters.

Return Value

- `int`: The function returns an integer value representing the number of waiting orders.

Example Usage

Python3

```
import ctypes

# Load the DLL
dll = ctypes.CDLL('/path/to/your_dll.dll')

# Define the return type of the function
dll.getNumberOfWaitingOrders.restype = ctypes.c_int

# Call the function
waiting_orders = dll.getNumberOfWaitingOrders()
print(f"Number of waiting orders: {waiting_orders}")
```

C#

```
using System;
using System.Runtime.InteropServices;

class Program
{
```

```
    [DllImport("path_to_your_dll.dll", CallingConvention =
CallingConvention.Cdecl)]
    private static extern int getNumberOfWaitingOrders();

    static void Main()
    {
        int waitingOrders = getNumberOfWaitingOrders();
        Console.WriteLine($"Number of waiting orders: {waitingOrders}");
    }
}
```

Java

```
import com.sun.jna.Library;
import com.sun.jna.Native;

public class Main {
    public interface MyLibrary extends Library {
        MyLibrary INSTANCE = (MyLibrary) Native.load("name_of_your_dll",
MyLibrary.class);
        int getNumberOfWaitingOrders();
    }

    public static void main(String[] args) {
        int waitingOrders = MyLibrary.INSTANCE.getNumberOfWaitingOrders();
        System.out.println("Number of waiting orders: " + waitingOrders);
    }
}
```

C++

```
// main.cpp
#include <iostream>

extern "C" int getNumberOfWaitingOrders();

int main() {
    int waitingOrders = getNumberOfWaitingOrders();
    std::cout << "Number of waiting orders: " << waitingOrders << std::endl;

    return 0;
}
```

JavaScript (e.g. Electron App)

Install `ffi-napi`

```
npm install ffi-napi
```

Using DLL with `ffi-napi`

```
const ffi = require('ffi-napi');

// Define the function signature
const myDll = ffi.Library('path_to_your_dll', {
  'getNumberOfWaitingOrders': ['int', []],
});

// Call the function
const numberOfWaitingOrders = myDll.getNumberOfWaitingOrders();
console.log("Number of waiting orders:", numberOfWaitingOrders);
```

> Get Estimated Waiting Time

The `getEstimatedWaitingTime` function retrieves the current estimated waiting time.

> ⚠️ **Warning:** The estimated waiting time is subject to change based on booth conditions and customer interactions, and therefore, it should not be considered an exact measure of the actual waiting period.

Function Signature

```
int getEstimatedWaitingTime();
```

Parameters

This function does not require any input parameters.

Return Value

- `int`: The function returns an integer value representing the estimated waiting time in seconds.

Example Usage

Python3

```
import ctypes

# Load the DLL
dll = ctypes.CDLL('/path/to/your_dll.dll')

# Define the return type of the function
dll.getEstimatedWaitingTime.restype = ctypes.c_int

# Call the function
estimated_waiting_time= dll.getEstimatedWaitingTime()
print(f"Estimated waiting time: {estimated_waiting_time}s")
```

C#

```
using System;
using System.Runtime.InteropServices;

class Program
{
    [DllImport("path_to_your_dll.dll", CallingConvention =
CallingConvention.Cdecl)]
    private static extern int getEstimatedWaitingTime();

    static void Main()
    {
        int estimatedWaitingTime = getEstimatedWaitingTime();
        Console.WriteLine($"Estimated waiting time: {estimatedWaitingTime}s");
    }
}
```

Java

```
import com.sun.jna.Library;
import com.sun.jna.Native;

public class Main {
    public interface MyLibrary extends Library {
        MyLibrary INSTANCE = (MyLibrary) Native.load("name_of_your_dll",
MyLibrary.class);
        int getEstimatedWaitingTime();
    }

    public static void main(String[] args) {
        int estimatedWaitingTime = MyLibrary.INSTANCE.getEstimatedWaitingTime();
        System.out.println("Estimated waiting time: " + estimatedWaitingTime +
"s");
    }
}
```

C++

```
// main.cpp
#include <iostream>

extern "C" int getEstimatedWaitingTime();

int main() {
    int estimatedWaitingTime = getEstimatedWaitingTime();
    std::cout << "Estimated waiting time: " << estimatedWaitingTime << std::endl;

    return 0;
}
```

JavaScript (e.g. Electron App)

Install `ffi-napi`

```
npm install ffi-napi
```

Using DLL with `ffi-napi`

```
const ffi = require('ffi-napi');

// Define the function signature
const myDll = ffi.Library('path_to_your_dll', {
  'getEstimatedWaitingTime': ['int', []],
});

// Call the function
const estimatedWaitingTime = myDll.getEstimatedWaitingTime();
console.log("Estimated waiting time:", estimatedWaitingTime);
```

> Get Product Status

The `getProductStatus` retrieves all menu details and its status. It is recommended for the Kiosk Client to continuously poll the product status for display. The recommended polling interval is 1 second.

Function Signature

```
const char* getProductStatusWrapped();
```

Parameters

This function does not require any input parameters.

Return Value

- `const char*`: The function returns a serialized JSON string. This string represents a list of products, with each product having its own set of attributes.
- **Data Format:**

  > The returned JSON string will be structured as follows:

  ```
  {
  "products": [
  {
       "menuId": String,
       "priceUnit": String,
       "description": String,
       "description_kr": String,
        // other language descriptions with suffix according to available
  language
       "alias": String,
       "alias_kr": String,
  ```

```
      // other alias for each available language
     "price": Double,
     "menuImage": String,
     "availability": Boolean,
    "categoryId": String,
     "categoryName": String,
     "categoryName_kr": String,
     // other category name for each available language
     "isTumblerEnabled": Boolean,
     "options": [
         {
             "optionCategoryId": String,
             "viewIndex": UnsignedInt,
             "optionId": String,
             "alias": String,
             "alias_kr": String,
             // other alias for each available language
             "optionCategoryName": String,
             "optionCategoryName_kr": String,
             // other option category name for each available language
             "price": Double,
             "isEnabled": Boolean,
             "extraInformation": []
                 {...}
             ]
         }
     ]
},
{...},
]
}
```

Here is an example of the product status data:

```
{
"products": [
{
     "menuId": "d82e0a96-63b5-41ef-a2f4-87e303bbdf90",
     "priceUnit": "KRW",
     "description": "This is Cafe Latte",
     "description_kr": "카페라떼입니다",
      // ... (other languages)
     "alias": "Cafe Latte",
     "alias_kr": "카페라떼",
      // ... (other languages)
     "price": 4000.0,
     "menuImage": "path_to_menu_image",
     "availability": true,
    "categoryId": "cec9aba0-4771-4bdd-909f-ca03f617c483",
     "categoryName": Coffee,
     "categoryName_kr": 커피류,
      // ... (other languages)
     "isTumblerEnabled": false,
     "options": [
         {
```

```
                "optionCategoryId": "53ed0dc0-33d2-42c6-ab3c-7126e0060479",
                "optionId": "313db0fa-9910-410b-ad90-99edeb754c50",
                "viewIndex": 0,
                "alias": "Add ice",
                "alias_kr": "얼음 추가",
                // ... (other languages)
                "optionCategoryName": "Ice",
                "optionCategoryName_kr": "얼음양 조절",
                // ... (other languages)
                "price": 1.99,
                "isEnabled": true,
                "extraInformation": [
                    {
                        "key": "cal",
                        "value": "89"
                    }
                ]
            }
        ]
    },
    {...},
    ]
    }
```

Data Field Description:

`products`: a list of all products retrieved from barista booth.

`menuId`: a unique menu identifier. Each menu has its own identifier.

`priceUnit`: the price unit set for this product.

`description`: menu description in English (default).

`description_kr`: menu description in Korean. **Note: you can get description in other languages using the `id` of the `AvailableLangue` struct from `getBoothConfig` (see Get Booth Config) as a suffix.** For example, if `es` (for Español) is one of the available language, you can add the id as a suffix like so, `description_es` to get the description in Spanish.

`alias`: menu name in English (default).

`alias_kr`: menu name in Korean.

`price`: price.

`menuImage`: path to the menu image.

`availability`: `true` if the menu is available for sale, `false` otherwise.

`categoryName`: category name in English.

`categoryName_kr`: category name in Korean.

`isTumblerEnabled`: `true` if user should be able to select takeout or tumbler for this product. If set to `true`, the Kiosk client should provide a option for the user to select either takeout or tumbler from the Kiosk display. See Tumbler Sequence diagram from Sequences section. Please note that the `isTumblerEnabled` field simply signifies that the product can be served in a tumbler; it does not imply that the product is exclusively available for service in a tumbler.

> `options`: available options from the menu:
>
> - `optionCategoryId`: the ID of the option category. Users should be able to choose only one option from this category.
> - `optionId`: option ID
> - `viewIndex`: view priority among the same option category. (e.g. if we have 2 options for "more ice" with `viewIndex` 1 and "less ice" with `viewIndex` 0, "less ice" option should appear before "more ice" option).
> - `alias`: option title in English (default)
> - `optionCategoryName`: option category name in English (default)
> - `alias_kr`: option title in Korean
> - `optionCategoryName_kr`: option category name in Korean
> - `price`: option price
> - `isEnabled`: `true` if the option is available. `false` otherwise.
> - `extraInformation`: extra information set by franchise owner.

> Place Order with Invoice

The `placeOrderWithInvoice` is a function that allows client applications to place an order with a list of products.

Function Signature

```
OrderResult placeOrderWithInvoice(const Invoice& invoice, Product* products,
const char numberOfProducts[10]);
```

Parameters

- Invoice structure which contains various order invoice data.

> The `Invoice` struct should include:
>
> - `orderId` (`char[50]`): A string representing a unique order identifier. (`UUID4`).
> - `orderLabel` (`char[50]`): A string representing a order label. This information will be used to display order label on the display of the barista booth so that the customer can identify their order from the pickup station. (e.g. "001", "002", "003", etc).
> - `qrData` (`char[999]`): A string representing a unique QR data. When QR pickup option is enabled from the booth, the Kiosk should print this QR code on the receipt and add this QR data to `qrData` field. The customer will be able to scan this printed QR code with the scanner for beverage pickup.
> - `invoiceNo` (`char[50]`): A string representing the invoice number.
> - `refNo` (`char[50]`): A string representing the reference number.

  - `purchase` (`char[20]`): A string representing the actual transaction amount for the order.
  - `authCode` (`char[24]`): A string representing the authentication code.
  - `acqRefData` (`char[200]`): A string representing the acquirer reference data.
  - `processData` (`char[999]`): A string representing the process data.
  - `recordNo` (`char[256]`): Token data.
  - `tranDeviceId` (`char[24]`): Transaction Device ID.
  - `DateTime` (`char[50]`): Date time of order registration (`YYYY-MM-DD HH:MM:SS.sss`).
- Product which contains an array of Product structure.

  The `Product` struct should include:

  - `menuId` (`char[50]`): A string representing a unique menu identifier.
  - `menuAliasCulture` (`char[50]`): A string representing the language/culture code that the customer selected when ordering. This should match one of the language IDs from the booth configuration (e.g., "en" for English, "kr" for Korean, "es" for Spanish). The booth uses this to display the order in the correct language and generate receipts in the appropriate language.
  - `options` (`char**`): A pointer to array of option IDs (`char*`).
  - `numberOfOptions` (`char[10]`): Number of options added for this product (as a string).
- `numberOfProducts` (`const char[10]`): String representing the number of products in order.

Return Value

- `OrderResult` struct

  `OrderResult` struct:

  ```
  struct OrderResult {
      OrderRegistrationErrorCode errorCode;
      char boothName[256];
  };
  ```

  - `OrderRegistrationErrorCode` Enum representing order registration result. (See Error Code below).
  - `boothName` to direct which booths the customer should go to (for multiple booths scenario). Do not need to consider for this version

Error Code

The following shows the `OrderRegistrationErrorCode` and description.

```
enum class OrderRegistrationErrorCode {
    Success = 0,
    OutOfStock,
    ExceedsMaximumAcceptableProductCount,
    Failed
};
```

- `Success`: represents successful order registration.
- `OutOfStock`: unsuccessful order placement because product is out of stock.
- `ExceedsMaximumAcceptableProductCount`: the received order contains more number of products than the maximum accepted product count in a single order (see `getBoothConfig` function).
- `Failed`: represents unsuccessful order registration due to internal error.

Note

- Tumbler option has a option ID as `uuid` of `d3b55e4a-cdc1-42d7-9983-22058474ec03`.

Example Usage

Python3

```
import ctypes

class Invoice(ctypes.Structure):
    _fields_ = [
        ("orderId", ctypes.c_char * 50),
        ("orderLabel", ctypes.c_char * 50),
        ("qrData", ctypes.c_char * 999),
        ("invoiceNo", ctypes.c_char * 50),
        ("refNo", ctypes.c_char * 50),
        ("purchase", ctypes.c_char * 20),
        ("authCode", ctypes.c_char * 24),
        ("acqRefData", ctypes.c_char * 200),
        ("processData", ctypes.c_char * 999),
        ("recordNo", ctypes.c_char * 256),
        ("tranDeviceId", ctypes.c_char * 24),
        ("DateTime", ctypes.c_char * 50),
    ]

class Product(ctypes.Structure):
    _fields_ = [
        ("menuId", ctypes.c_char * 50),
        ("menuAliasCulture", ctypes.c_char * 50),
        ("options", ctypes.POINTER(ctypes.c_char_p)),
        ("numberOfOptions", ctypes.c_char * 10)
    ]

class OrderResult(ctypes.Structure):
    _fields_ = [
        ("errorCode", ctypes.c_int),
        ("boothName", ctypes.c_char * 256),
    ]
```

```
# Define function signature
dll = ctypes.CDLL('/path/to/your_dll.dll')
dll.placeOrderWithInvoice.argtypes = [ctypes.POINTER(Invoice),
ctypes.POINTER(Product), ctypes.c_char * 10]
dll.placeOrderWithInvoice.restype = OrderResult

# Usage
invoice = Invoice()
invoice.orderId = b"a0f68b2d-04ff-4246-a002-2b6820153145"
invoice.orderLabel = b"001"
invoice.qrData = b"QR Data"
invoice.invoiceNo = b"INV001"
invoice.refNo = b"REF001"
invoice.purchase = b"100.50"
invoice.authCode = b"AUTH123"
invoice.acqRefData = b"ACQ123"
invoice.processData = b"PROCESS123"
invoice.recordNo = b"REC001"
invoice.tranDeviceId = b"DEV001"
invoice.DateTime = b"2024-01-01 10:00:00.000"

products = (Product * 2)()
products[0].menuId = b"53b053c3-442e-45db-ae6f-2c0e52db8228"
products[0].menuAliasCulture = b"en"
products[0].numberOfOptions = b"0"
products[1].menuId = b"another-menu-id"
products[1].menuAliasCulture = b"en"
products[1].numberOfOptions = b"0"

number_of_products = ctypes.create_string_buffer(b"2")
result = dll.placeOrderWithInvoice(ctypes.byref(invoice), products,
number_of_products)

if result.errorCode == 0:
    print(f"Order successful. Proceed to booth: {result.boothName.decode('utf-
8')}")
else:
    print(f"Order failed with error code: {result.errorCode}")
```

C#

```
[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi)]
public struct OrderResult
{
    public int errorCode;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
    public string boothName;
}

[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi)]
public struct Invoice
{
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 50)]
```

```
    public string orderId;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 50)]
    public string orderLabel;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 999)]
    public string qrData;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 50)]
    public string invoiceNo;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 50)]
    public string refNo;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 20)]
    public string purchase;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 24)]
    public string authCode;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 200)]
    public string acqRefData;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 999)]
    public string processData;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
    public string recordNo;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 24)]
    public string tranDeviceId;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 50)]
    public string DateTime;
}

[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi)]
public struct Product
{
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 50)]
    public string menuId;
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 50)]
    public string menuAliasCulture;
    public IntPtr options; // char**
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 10)]
    public string numberOfOptions;
}

class Program
{
    [DllImport("your_dll.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern OrderResult placeOrderWithInvoice(ref Invoice invoice,
[MarshalAs(UnmanagedType.LPArray)] Product[] products,
[MarshalAs(UnmanagedType.ByValTStr, SizeConst = 10)] string numberOfProducts);

    static void Main()
    {
        Invoice invoice = new Invoice
        {
            orderId = "a0f68b2d-04ff-4246-a002-2b6820153145",
            orderLabel = "001",
            qrData = "QR Data",
            invoiceNo = "INV001",
            refNo = "REF001",
            purchase = "100.50",
            authCode = "AUTH123",
            acqRefData = "ACQ123",
```

```
            processData = "PROCESS123",
            recordNo = "REC001",
            tranDeviceId = "DEV001",
            DateTime = "2024-01-01 10:00:00.000"
        };
        Product[] products = new Product[]
        {
            new Product { menuId = "53b053c3-442e-45db-ae6f-2c0e52db8228",
menuAliasCulture = "en", numberOfOptions = "0", options = IntPtr.Zero },
            new Product { menuId = "another-menu-id", menuAliasCulture = "en",
numberOfOptions = "0", options = IntPtr.Zero }
        };
        OrderResult result = placeOrderWithInvoice(ref invoice, products, "2");
        if (result.errorCode == 0)
        {
            Console.WriteLine($"Order successful. Proceed to booth:
{result.boothName}");
        }
        else
        {
            Console.WriteLine($"Order failed with error code:
{result.errorCode}");
        }
    }
}
```

Java

```
public class OrderResult extends Structure {
    public int errorCode;
    public byte[] boothName = new byte[256];

    @Override
    protected List<String> getFieldOrder() {
        return Arrays.asList("errorCode", "boothName");
    }
}

public interface MyLibrary extends Library {
    MyLibrary INSTANCE = Native.load("your_dll", MyLibrary.class);

    OrderResult placeOrderWithInvoice(Invoice invoice, Product[] products, byte[]
numberOfProducts);
}

public class Invoice extends Structure {
    public byte[] orderId = new byte[50];
    public byte[] orderLabel = new byte[50];
    public byte[] qrData = new byte[999];
    public byte[] invoiceNo = new byte[50];
    public byte[] refNo = new byte[50];
    public byte[] purchase = new byte[20];
    public byte[] authCode = new byte[24];
    public byte[] acqRefData = new byte[200];
```

```
    public byte[] processData = new byte[999];
    public byte[] recordNo = new byte[256];
    public byte[] tranDeviceId = new byte[24];
    public byte[] DateTime = new byte[50];

    @Override
    protected List<String> getFieldOrder() {
        return Arrays.asList("orderId", "orderLabel", "qrData", "invoiceNo",
"refNo",
                             "purchase", "authCode", "acqRefData", "processData",
                             "recordNo", "tranDeviceId", "DateTime");
    }
}

public class Product extends Structure {
    public byte[] menuId = new byte[50];
    public byte[] menuAliasCulture = new byte[50];
    public Pointer options; // char**
    public byte[] numberOfOptions = new byte[10];

    @Override
    protected List<String> getFieldOrder() {
        return Arrays.asList("menuId", "menuAliasCulture", "options",
"numberOfOptions");
    }
}

public class Main {
    public static void main(String[] args) {
        Invoice invoice = new Invoice();
        System.arraycopy("a0f68b2d-04ff-4246-a002-2b6820153145".getBytes(), 0,
invoice.orderId, 0, Math.min(invoice.orderId.length, "a0f68b2d-04ff-4246-a002-
2b6820153145".getBytes().length));
        System.arraycopy("001".getBytes(), 0, invoice.orderLabel, 0,
Math.min(invoice.orderLabel.length, "001".getBytes().length));
        System.arraycopy("QR Data".getBytes(), 0, invoice.qrData, 0,
Math.min(invoice.qrData.length, "QR Data".getBytes().length));
        // Initialize other invoice fields similarly...

        Product[] products = (Product[]) new Product().toArray(2);
        System.arraycopy("53b053c3-442e-45db-ae6f-2c0e52db8228".getBytes(), 0,
products[0].menuId, 0, Math.min(products[0].menuId.length, "53b053c3-442e-45db-
ae6f-2c0e52db8228".getBytes().length));
        System.arraycopy("en".getBytes(), 0, products[0].menuAliasCulture, 0,
Math.min(products[0].menuAliasCulture.length, "en".getBytes().length));
        System.arraycopy("0".getBytes(), 0, products[0].numberOfOptions, 0,
Math.min(products[0].numberOfOptions.length, "0".getBytes().length));
        products[0].options = Pointer.NULL;

        System.arraycopy("another-menu-id".getBytes(), 0, products[1].menuId, 0,
Math.min(products[1].menuId.length, "another-menu-id".getBytes().length));
        System.arraycopy("en".getBytes(), 0, products[1].menuAliasCulture, 0,
Math.min(products[1].menuAliasCulture.length, "en".getBytes().length));
        System.arraycopy("0".getBytes(), 0, products[1].numberOfOptions, 0,
Math.min(products[1].numberOfOptions.length, "0".getBytes().length));
        products[1].options = Pointer.NULL;
```

```
        byte[] numberOfProducts = new byte[10];
        System.arraycopy("2".getBytes(), 0, numberOfProducts, 0,
Math.min(numberOfProducts.length, "2".getBytes().length));

        OrderResult result = MyLibrary.INSTANCE.placeOrderWithInvoice(invoice,
products, numberOfProducts);

        if (result.errorCode == 0) {
            System.out.println("Order successful. Proceed to booth: " + new
String(result.boothName).trim());
        } else {
            System.out.println("Order failed with error code: " +
result.errorCode);
        }
    }
}
```

C++

```
#include <iostream>
#include <cstring>

struct Invoice {
    char orderId[50];
    char orderLabel[50];
    char qrData[999];
    char invoiceNo[50];
    char refNo[50];
    char purchase[20];
    char authCode[24];
    char acqRefData[200];
    char processData[999];
    char recordNo[256];
    char tranDeviceId[24];
    char DateTime[50];
};

struct Product {
    char menuId[50];
    char menuAliasCulture[50];
    char** options;
    char numberOfOptions[10];
};

enum class OrderRegistrationErrorCode {
    Success = 0,
    OutOfStock,
    ExceedsMaximumAcceptableProductCount,
    Failed
};

struct OrderResult {
    OrderRegistrationErrorCode errorCode;
```

```
    char boothName[256];
};

extern "C" OrderResult placeOrderWithInvoice(const Invoice& invoice, Product*
products, const char numberOfProducts[10]);

int main() {
    Invoice invoice{};
    strcpy(invoice.orderId, "a0f68b2d-04ff-4246-a002-2b6820153145");
    strcpy(invoice.orderLabel, "001");
    strcpy(invoice.qrData, "QR Data");
    strcpy(invoice.invoiceNo, "INV001");
    strcpy(invoice.refNo, "REF001");
    strcpy(invoice.purchase, "100.50");
    strcpy(invoice.authCode, "AUTH123");
    strcpy(invoice.acqRefData, "ACQ123");
    strcpy(invoice.processData, "PROCESS123");
    strcpy(invoice.recordNo, "REC001");
    strcpy(invoice.tranDeviceId, "DEV001");
    strcpy(invoice.DateTime, "2024-01-01 10:00:00.000");

    Product products[2]{};
    strcpy(products[0].menuId, "53b053c3-442e-45db-ae6f-2c0e52db8228");
    strcpy(products[0].menuAliasCulture, "en");
    strcpy(products[0].numberOfOptions, "0");
    products[0].options = nullptr;

    strcpy(products[1].menuId, "another-menu-id");
    strcpy(products[1].menuAliasCulture, "en");
    strcpy(products[1].numberOfOptions, "0");
    products[1].options = nullptr;

    char numberOfProducts[10] = "2";
    OrderResult result = placeOrderWithInvoice(invoice, products,
numberOfProducts);

    if (result.errorCode == OrderRegistrationErrorCode::Success) {
        std::cout << "Order successful. Proceed to booth: " << result.boothName
<< std::endl;
    } else {
        std::cout << "Order failed with error code: " << static_cast<int>
(result.errorCode) << std::endl;
    }

    return 0;
}
```

JavaScript (e.g. Electron App)

Install `ffi-napi`

```
npm install ffi-napi
```

Using DLL with `ffi-napi`

```
const ffi = require('ffi-napi');
const ref = require('ref-napi');
const Struct = require('ref-struct-di')(ref);

const OrderResult = Struct({
    'errorCode': 'int',
    'boothName': 'char[256]'
});

const myDll = ffi.Library('path_to_your_dll', {
    'placeOrderWithInvoice': [OrderResult, [Invoice, ProductArray,
ArrayType('char', 10)]]
});

const CharArray10 = ArrayType('char', 10);

const Invoice = Struct({
    'orderId': ArrayType('char', 50),
    'orderLabel': ArrayType('char', 50),
    'qrData': ArrayType('char', 999),
    'invoiceNo': ArrayType('char', 50),
    'refNo': ArrayType('char', 50),
    'purchase': ArrayType('char', 20),
    'authCode': ArrayType('char', 24),
    'acqRefData': ArrayType('char', 200),
    'processData': ArrayType('char', 999),
    'recordNo': ArrayType('char', 256),
    'tranDeviceId': ArrayType('char', 24),
    'DateTime': ArrayType('char', 50)
});

const Product = Struct({
    'menuId': ArrayType('char', 50),
    'menuAliasCulture': ArrayType('char', 50),
    'options': 'pointer', // char**
    'numberOfOptions': ArrayType('char', 10)
});

const invoice = new Invoice();
invoice.orderId = Buffer.from("a0f68b2d-04ff-4246-a002-2b6820153145\0");
invoice.orderLabel = Buffer.from("001\0");
invoice.qrData = Buffer.from("QR Data\0");
invoice.invoiceNo = Buffer.from("INV001\0");
invoice.refNo = Buffer.from("REF001\0");
invoice.purchase = Buffer.from("100.50\0");
invoice.authCode = Buffer.from("AUTH123\0");
invoice.acqRefData = Buffer.from("ACQ123\0");
invoice.processData = Buffer.from("PROCESS123\0");
invoice.recordNo = Buffer.from("REC001\0");
invoice.tranDeviceId = Buffer.from("DEV001\0");
invoice.DateTime = Buffer.from("2024-01-01 10:00:00.000\0");

const ProductArray = ArrayType(Product);
const products = new ProductArray(2);
products[0].menuId = Buffer.from("53b053c3-442e-45db-ae6f-2c0e52db8228\0");
products[0].menuAliasCulture = Buffer.from("en\0");
```

```
products[0].numberOfOptions = Buffer.from("0\0");
products[0].options = ref.NULL;

products[1].menuId = Buffer.from("another-menu-id\0");
products[1].menuAliasCulture = Buffer.from("en\0");
products[1].numberOfOptions = Buffer.from("0\0");
products[1].options = ref.NULL;

const numberOfProducts = Buffer.from("2\0");

const result = myDll.placeOrderWithInvoice(invoice.ref(), products,
numberOfProducts);

if (result.errorCode === 0) {
    console.log(`Order successful. Proceed to booth:
${ref.readCString(result.boothName, 0)}`);
} else {
    console.log(`Order failed with error code: ${result.errorCode}`);
}
```

> Register Order Cancellation Callback

The `registerOrderCancellationCallback` function allows client applications to register a callback function for order cancellation events. This function enables the DLL to notify the client whenever an order is cancelled, providing the order ID as a parameter to the callback.

Function Signature

```
void registerInvoiceOrderCancellationCallback(OrderCancellationCallback
callback);
```

Parameter

- **callback** (`OrderCancellationCallback`): A pointer to the client's callback function. This function should match the `OrderCancellationCallback` type.

Callback Function Signature

```
int (*OrderCancellationCallback)(const OrderCancellationInfo* info);
```

The `OrderCancellationInfo` structure passed to the callback includes the following fields:

- `orderId` (`char[50]`): A string representing the unique order identifier. (`UUID4`).
- `invoiceNo` (`char[50]`): A string representing the invoice number.
- `refNo` (`char[50]`): A string representing the reference number.
- `purchase` (`char[20]`): A string representing the amount to be refunded.
- `authCode` (`char[24]`): A string representing the authentication code.
- `acqRefData` (`char[200]`): A string representing the acquirer reference data.

- `processData` (`char[999]`): A string representing the process data.
- `recordNo` (`char[256]`): Token data.
- `tranDeviceId` (`char[24]`): Transaction Device ID.
- `DateTime` (`char[50]`): Date time of order registration (`YYYY-MM-DD HH:MM:SS.sss`).

Callback Function Return Value

- `int` value indicating order cancellation success/fail.

> `0`: Success.
>
> `1`: Fail.

Example Usage

Python3

```python
import ctypes

class OrderCancellationInfo(ctypes.Structure):
    _fields_ = [("orderId", ctypes.c_char * 50),
                ("refNo", ctypes.c_char * 50),
                ("invoiceNo", ctypes.c_char * 50),
                ("authCode", ctypes.c_char * 24)]
                # ... other fields ...

OrderCancellationCallback = ctypes.CFUNCTYPE(None,
ctypes.POINTER(OrderCancellationInfo))

dll = ctypes.CDLL('/path/to/your_dll.dll')
dll.registerInvoiceOrderCancellationCallback.argtypes =
[OrderCancellationCallback]

@OrderCancellationCallback
def my_callback(info):
    info = info.contents
    print(f"Order ID: {info.orderId.decode('utf-8')}")
    print(f"Ref No: {info.refNo.decode('utf-8')}")
    # ... handle other fields ...

dll.registerInvoiceOrderCancellationCallback(my_callback)
```

C#

```csharp
using System;
using System.Runtime.InteropServices;

[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Ansi)]
public struct OrderCancellationInfo
{
    [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 50)]
    public string orderId;
    [MarshalAs(UnmanagedType.R8)]
```

```
    public double purchase;
    // ... other fields like refNo, invoiceNo, etc., with appropriate attributes
...
}

// Delegate that matches the C++ callback function signature
[UnmanagedFunctionPointer(CallingConvention.Cdecl)]
public delegate void OrderCancellationCallback(ref OrderCancellationInfo info);

public class Program
{
    // Import the function from the DLL
    [DllImport("your_dll_name.dll", CallingConvention = CallingConvention.Cdecl)]
    public static extern void
registerInvoiceOrderCancellationCallback(OrderCancellationCallback callback);

    // Callback method
    private static int MyOrderCancellationCallback(ref OrderCancellationInfo
info)
    {
        Console.WriteLine($"Order ID: {info.orderId}");
        // ... handle other fields ...
    }

    static void Main(string[] args)
    {
        // Register the callback
        registerInvoiceOrderCancellationCallback(MyOrderCancellationCallback);

        // ... rest of your code ...
    }
}
```

Java

```
import com.sun.jna.*;
import com.sun.jna.ptr.*;

public class Main {
    public static class OrderCancellationInfo extends Structure {
        public static class ByReference extends OrderCancellationInfo implements
Structure.ByReference {}

        public byte[] orderId = new byte[50];
        public byte[] refNo = new byte[50];
        public byte[] invoiceNo = new byte[50];
        public byte[] authCode = new byte[24];
        // ... other fields ...
    }

    public interface MyLibrary extends Library {
        MyLibrary INSTANCE = Native.load("your_dll", MyLibrary.class);
        void registerInvoiceOrderCancellationCallback(Callback callback);
    }
```

```
    public static class OrderCancellationCallback implements Callback {
        public int invoke(OrderCancellationInfo.ByReference info) {
            String orderId = Native.toString(info.orderId);
            String refNo = Native.toString(info.refNo);
            // ... handle other fields ...
            System.out.println("Order ID: " + orderId);
            // ...
        }
    }

    public static void main(String[] args) {
        MyLibrary.INSTANCE.registerInvoiceOrderCancellationCallback(new
OrderCancellationCallback());
    }
}
```

JavaScript (e.g. Electron App)

Install `ffi-napi`

```
npm install ffi-napi
```

Using DLL with `ffi-napi`

```
const ffi = require('ffi-napi');
const ref = require('ref-napi');
const Struct = require('ref-struct-di')(ref);

const OrderCancellationInfo = Struct({
  'orderId': 'string',
  'refNo': 'string',
  'invoiceNo': 'string',
  'authCode': 'string'
  // ... other fields ...
});
const OrderCancellationInfoPtr = ref.refType(OrderCancellationInfo);
const OrderCancellationCallback = ffi.Callback('int', [OrderCancellationInfoPtr],
function(infoPtr) {
  let info = infoPtr.deref();
  console.log("Order ID:", info.orderId);
  // ... handle other fields ...
});

const myDll = ffi.Library('path_to_your_dll', {
  'registerInvoiceOrderCancellationCallback': ['int',
[OrderCancellationCallback]]
});

myDll.registerInvoiceOrderCancellationCallback(OrderCancellationCallback);
```

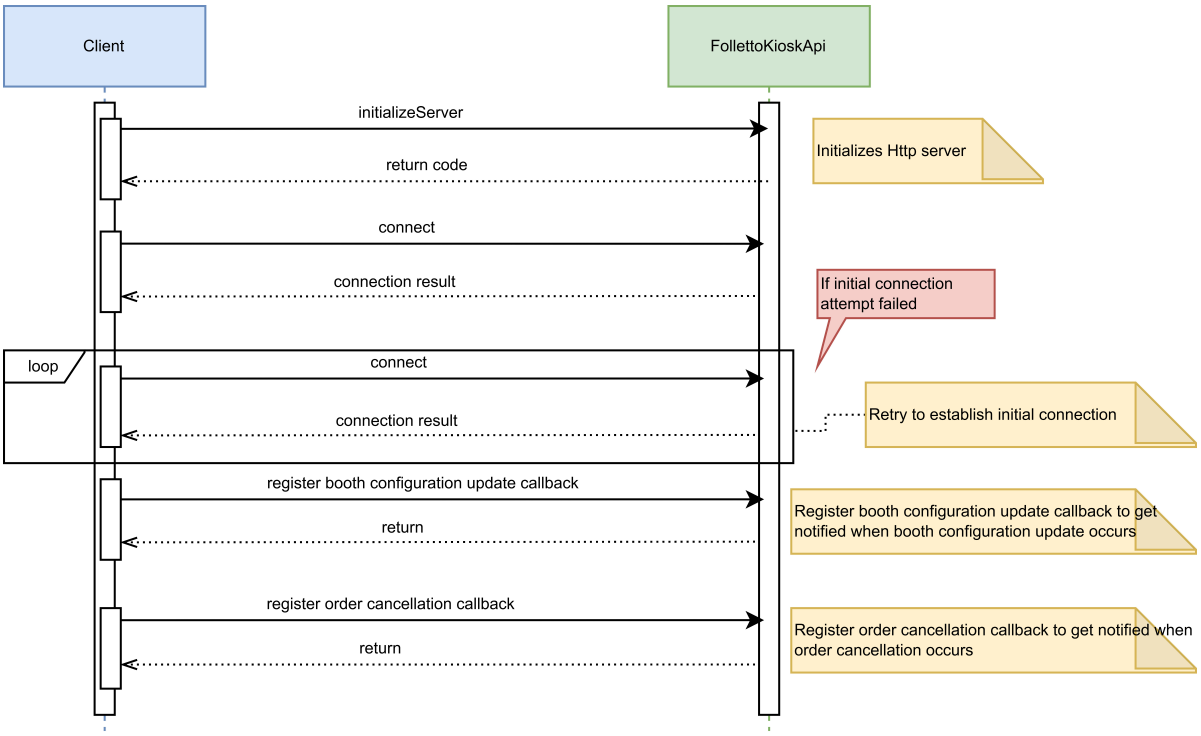
Sequences

> Initial Sequences

This section shows the initial interaction between DLL client and the `FollettKioskDllApi`.

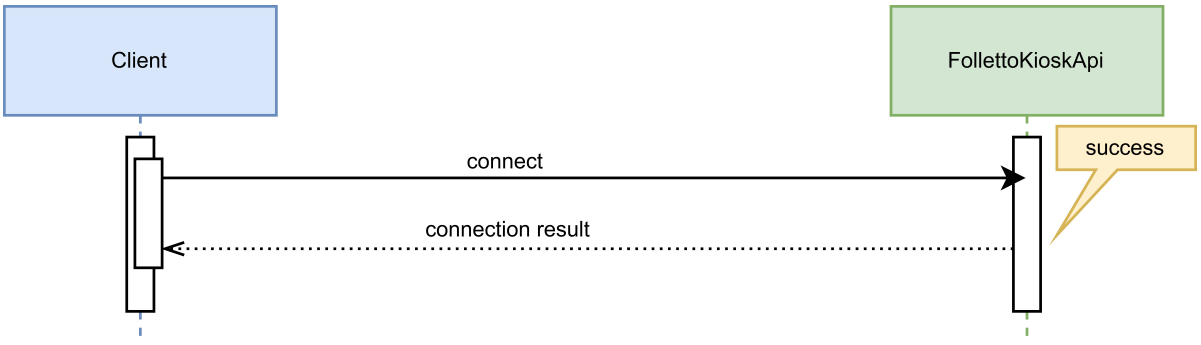
Initial Sequence Overview

This sequence diagram shows the initial interaction between client and DLL.



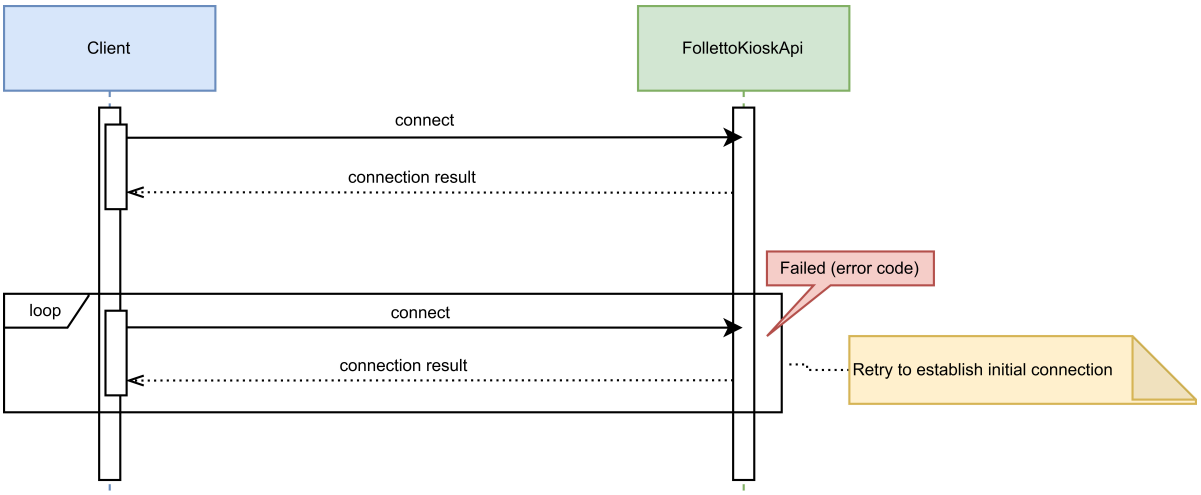
Initial Connection Sequence (Success)

This sequence diagram depicts the successful establishment of an initial connection.



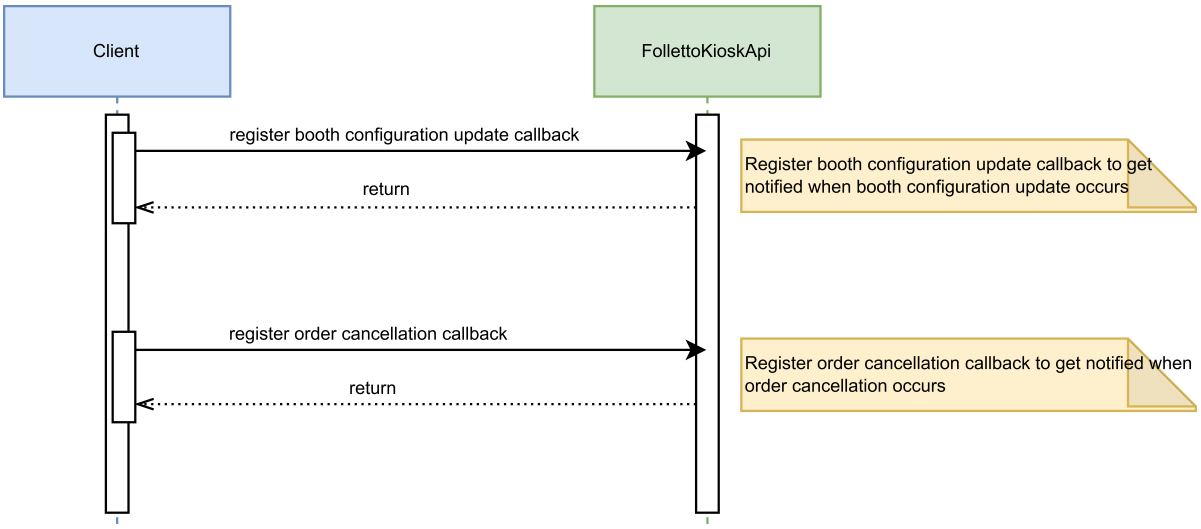
Initial Connection Sequence (Failure)

This sequence diagram depicts the unsuccessful establishment of an initial connection.



Initial Callback Registration

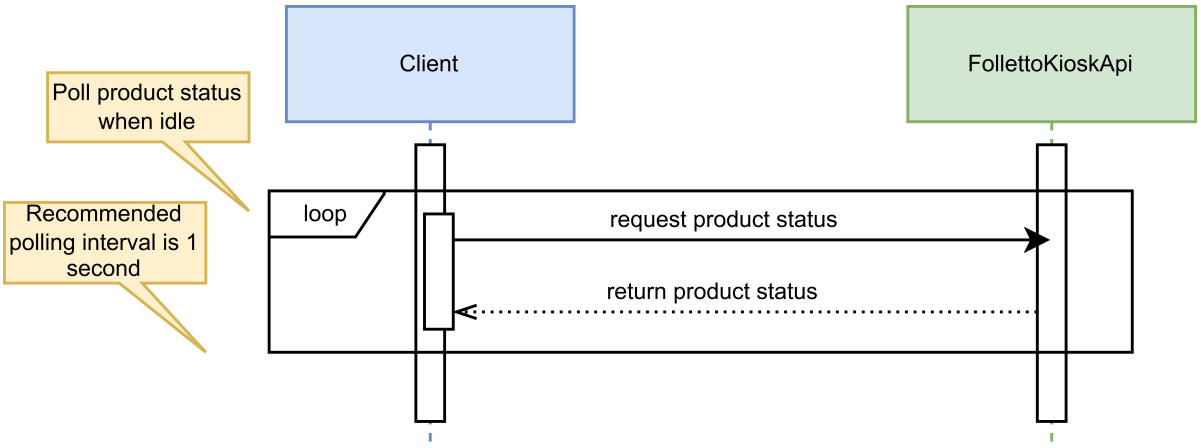
This sequence diagram depicts the callback registration sequence after successful connection.



> Product Status Polling Sequence

This section shows the sequence flow for product status polling after initialization. The Kiosk application should continuously poll product status to update available products.

> 💡 Note: The recommended polling interval is 1 second.

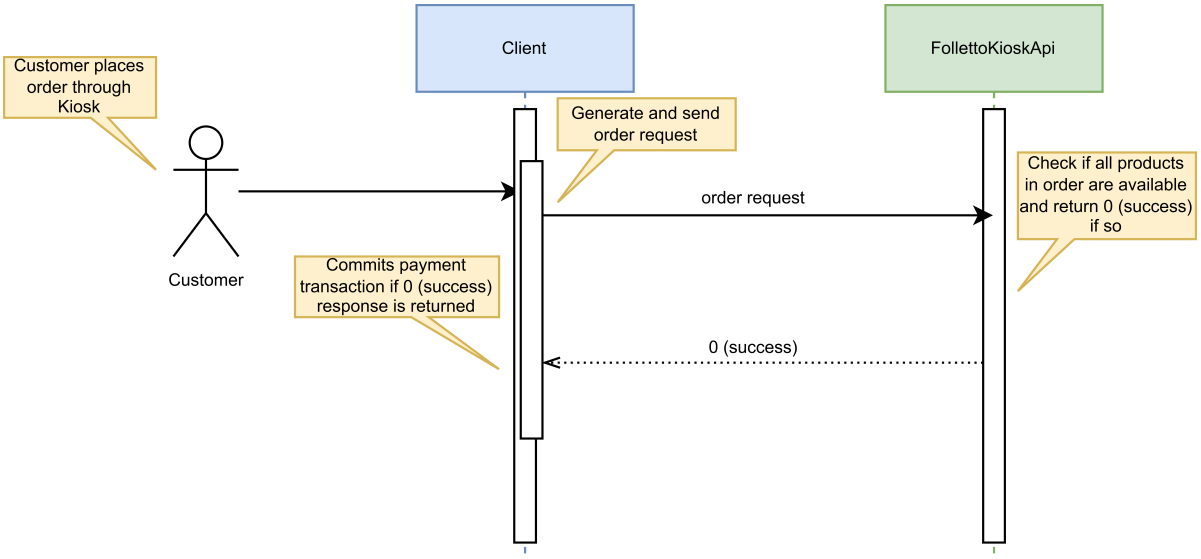


> Order Placement Sequence

This section shows the sequence flow for order placement.

Successful Order Placement

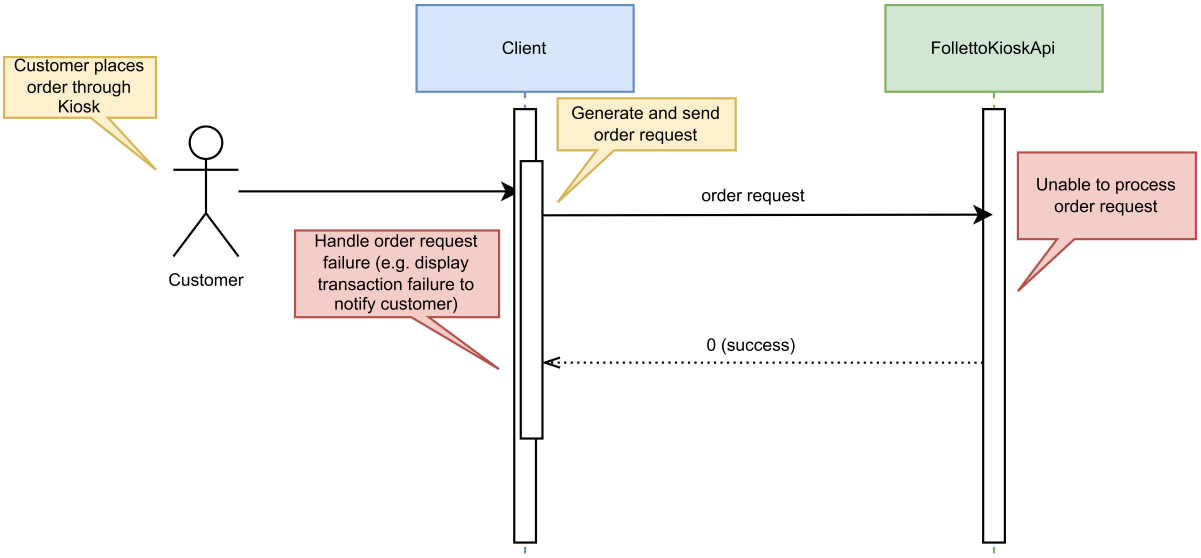
This sequence diagram shows the successful order placement sequence.



> 💡 Several points to note:
>
> - When the `FollettoKioskDllApi` receives the order request, it checks if all products in the order are currently available. If so, returns `0` (which represents `success` according to `placeOrderWithInvoice`).
> - It is **highly recommended** to perform the actual payment transaction only upon receiving this `success` response.

Unsuccessful Order Placement

This sequence diagram shows the unsuccessful order placement sequence.

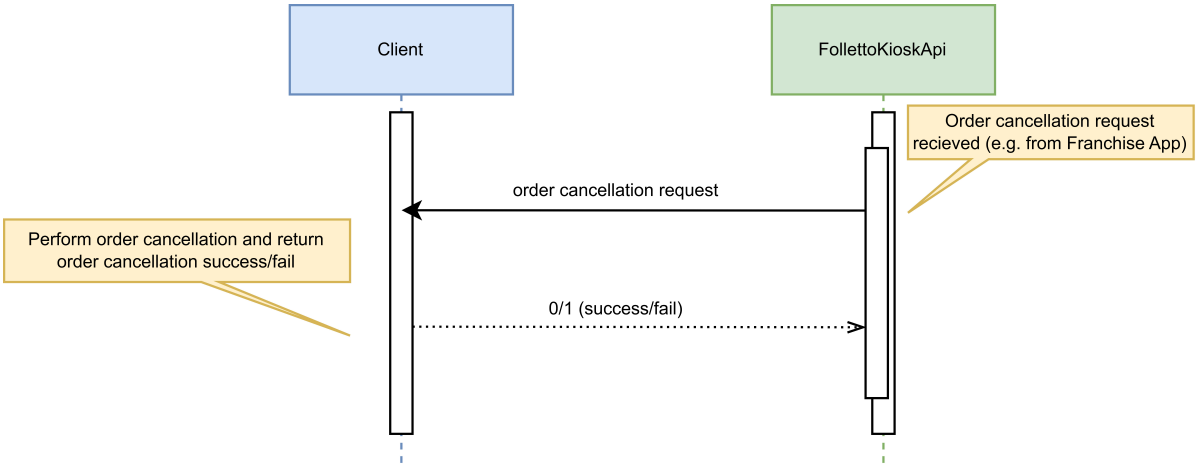


> 💡 Several points to note:
>
> - When the one or all of the products in the order request is not available, the DLL returns an error code according to `placeOrderWithInvoice`.

> - It is **highly recommended** to perform the actual payment only upon receiving the `success` response (not when other error code is received).

> Order Cancellation Sequence

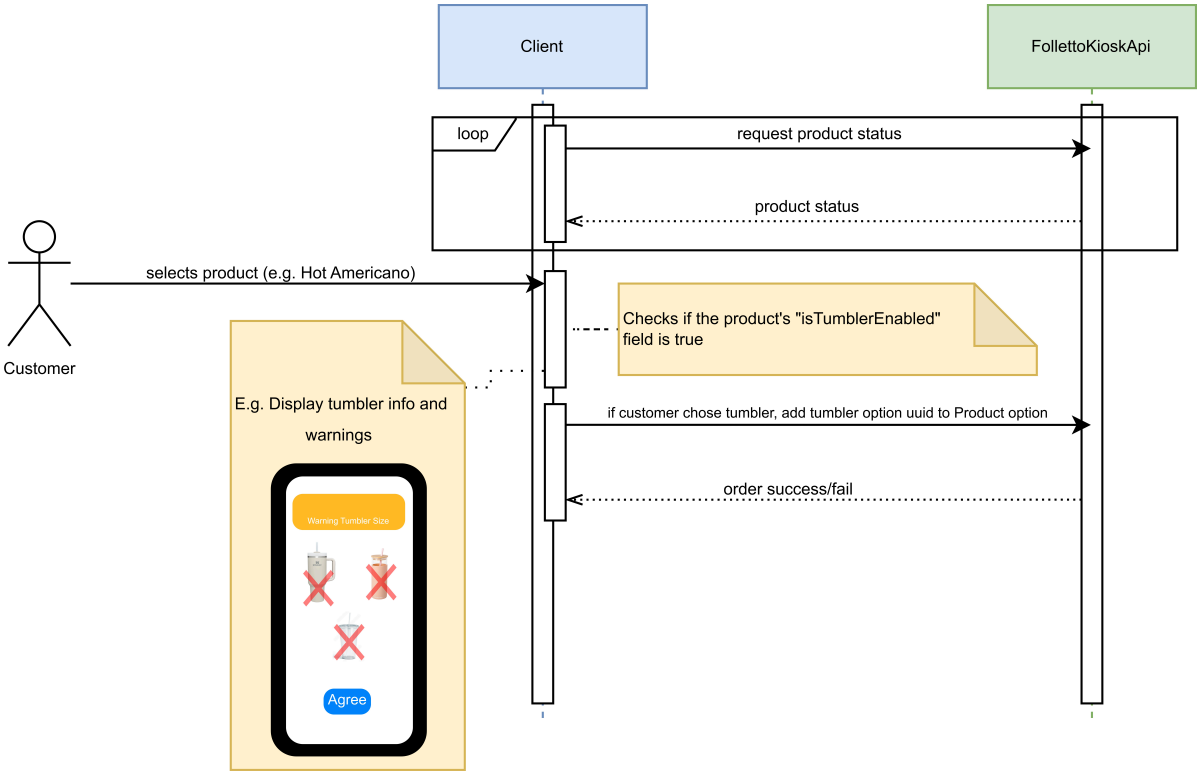
This section shows the sequence flow for order cancellation.



> - Order cancel request is received from external source (e.g. Franchise App).
> - The DLL sends order cancellation request to the client.
> - The client performs order cancellation and returns success or failure.

> Tumbler Sequence (Example)

This section shows the sequence flow for tumbler order.



In this example:

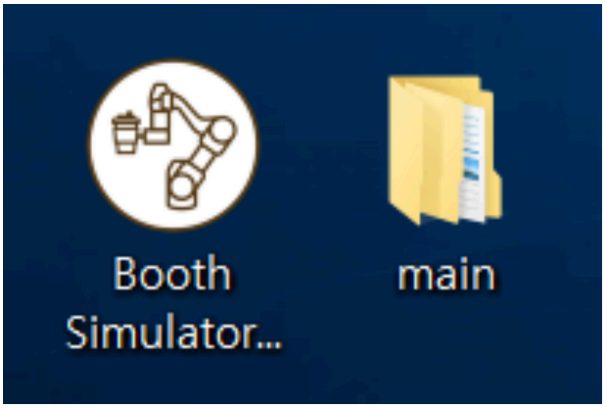
> - The Kiosk client continuously poll product status when idle.

> - When the customer uses the Kiosk to select a product which has `isTumblerEnabled` field set to `true`, the Kiosk displays various information about the allowed tumbler size or shape.
> - If the customer then selects tumbler option (instead of takeout), order should be placed with tumbler option where the tumbler option uuid should be included in `Product`'s `options` array. See tumbler option uuid here

Testing with Barista Booth Simulator Module

Installation

- Place `Booth Simulator Setup 1.0.0.exe` and `main` folder at `Desktop`.

> Note: `main` folder must be located at the Desktop directory as such:
>
> 
>

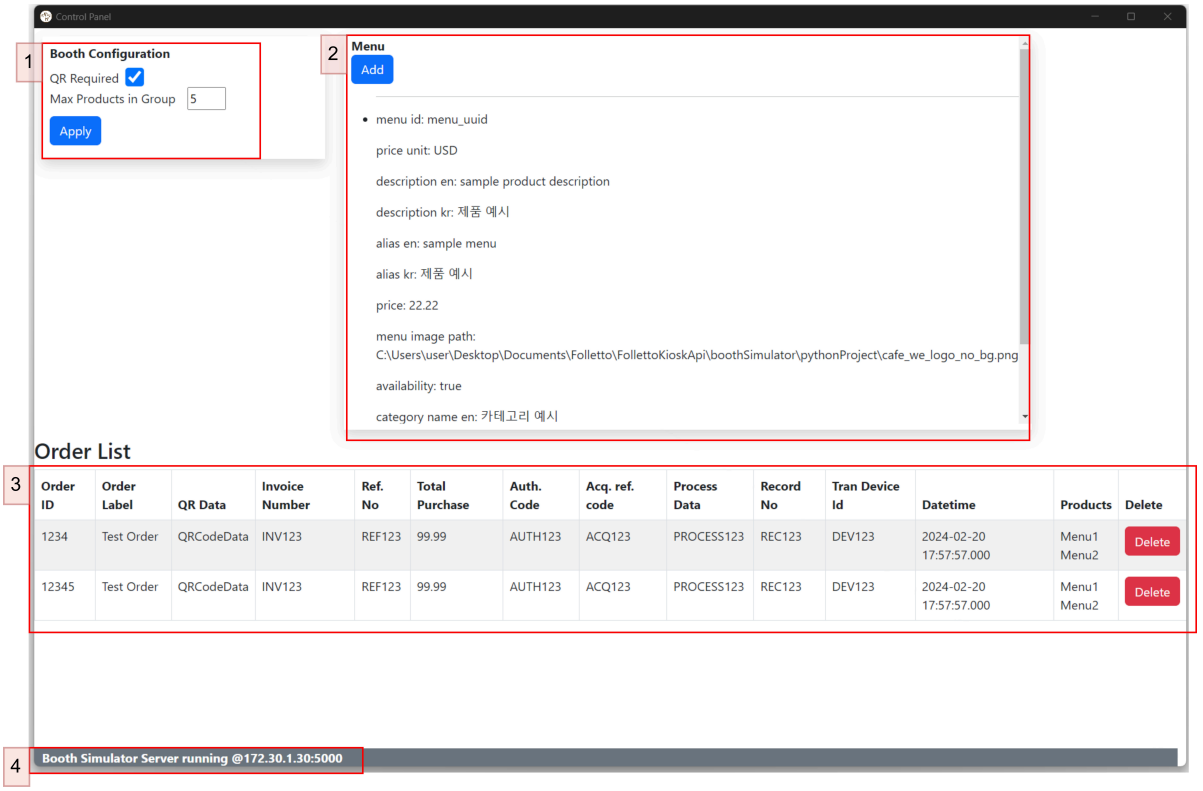
- Open `Booth Simulator Setup 1.0.0.exe` and install `Booth Simulator` program.

How to Use

Set Kiosk IP address and port for the booth to communicate with Kiosk DLL. In real scenario, this will be set in the booth settings page.

> 1: Kiosk IP and Port Inputs
>
> 2: Submit button to set the Kiosk IP and Port. When using the DLL, this port number should be same as the value passed to the parameter of `initializeServer` function.

The following shows the main page after setting the Kiosk IP and Port



> 1. Shows a section for booth configuration update testing. You can get the booth configuration using `getBoothConfig` function. Also, given that you successfully registered the booth configuration update callback using `registerBoothConfigUpdateCallback`, you can test the callback by clicking the "Apply" button.

2. Shows a menu section. You can get the product details by using `getProductStatusWrapped` function.
3. Shows the order list section. You can add new order using `placeOrderWithInvoice` function.

   Note: if you can't see the new order entry, press `ctrl + r` to refresh the page.
   Given that you registered order cancellation callback function using `registerInvoiceOrderCancellationCallback`, you can test the order cancellation by clicking "Delete" button.
4. Shows the IP and Port of the booth program. This information is required when using `connectToBooths` function (which takes IP and Port of the booths).

Appendix

Version Log

v0.0.8

What's new:

- Changed the return type of `getBoothConfig` to pointer.
- Bug fixed for option registration when placing order.

v0.0.7

What's new:

- Converted download logic to threads.

v0.0.6

What's new:

- Added image download functionality.

v0.0.5

What's new:

- Added `isTumblerEnabled` to product fields in `getProductStatus` function. Also added sequence diagrams to illustrate possible tumbler option interaction sequence between booth and the kiosk client.
- Added tumbler option `uuid` in Place Order with Invoice section.
- Added `viewIndex` to product option field for option rendering.
- Added available languages to `getBoothConfig` and language fields in `getProductStatus`. See Get Booth Configuration and Get Product Status.

Documentation Update:

- Added Sample Data section in the Appendix.

v0.0.4

What's new:

- Added new fields for options in `getProductStatusWrapped` and `placeOrderWithInvoice` function.

v0.0.3

What's new:

- Added `getDllVersion` to get DLL version.
- Converted data types to accommodate one kiosk, multiple booths scenario.

  > **Affected:**
  >
  > - `connect` **function:** changed input parameter to an array of struct to connect to multiple booths.
  > - `getBoothConfig` **function:** removed `boothId` field from the return value.
  > - `placeOrderWithInvoice` **function:** changed return value to struct.

v0.0.2

What's new:

- Added `categoryNameKr` and `categoryNameEn` in `getProductStatusWrapped` (Get Product Status) return value.
- Added `boothId` in `getBoothConfig` (Get Booth Configuration) return value.

Sample Data

This section shows some of the sample data (return data) from the booth (currently based on Jet Cafe robot barista booth).

Get DLL version

The following shows the sample output returned from the DLL using Get Version function.

```
0.0.5
```

Get Booth Configuration

The following shows the sample output returned from the Robot barista booth by using Get Booth Config function.

```
BoothConfig* -> POINTER to BoothConfig
```

where the returned sample data is as such:

```
BoothConfig.qrRequired -> true (or false if set to false)
BoothConfig.maxProductsInGroup -> 3 (currently set to 3 due to hardware
limitation)
BoothConfig.languages ->
    struct AvailableLanguage
    {
     char id[3];              // "en", "kr", "es"
     char display[128];       // "English", "한국어", "Español"
    };
```

Get Product Status Sample Data

The following shows the sample product data returned from Jet Cafe robot barista booth by using Get Product Status function.


```
{
    "products": [
        {
            "menuId": "fb96e8fc-4103-40aa-8a4a-98c1353d4c69",
            "priceUnit": "USD",
            "description": "Description in English",
            "description_kr": "한국어 설명",
            "description_es": "Descripción en español",
            "alias": "Hot Americano",
            "alias_kr": "핫 아메리카노",
            "alias_es": "Hot Americano(es)",
            "price": 3.10,
            "menuImage": "absolute_path_to_image",
            "availability": true,
            "categoryId": "cec9aba0-4771-4bdd-909f-ca03f617c483",
            "categoryName": "Coffee (Hot)",
            "categoryName_kr": "핫 커피류",
            "categoryName_es": "Coffee(es)",
            "isTumblerEnabled": true,
            "options": [
                {
                    "optionCategoryId": "8f0f6a90-3707-48af-9a33-caec64ecf843",
                    "viewIndex": 0,
                    "optionId": "d3b55e4a-cdc1-42d7-9983-22058474ec03",
                    "alias": "Use Tumbler",
                    "alias_kr": "텀블러 사용",
                    "alias_es": "Use Tumbler (ES)",
                    "optionCategoryName": "Takeout option",
```

```
                "optionCategoryName_kr": "테이크 아웃 옵션",
                "optionCategoryName_es": "Takeout option (ES)",
                "price": 0.00,
                "isEnabled": true,
                "extraInformation": []
            },
            {
                "optionCategoryId": "9e0ec2c7-1965-4bbb-92c8-795fe7f8cfab",
                "viewIndex": 0,
                "optionId": "cb513f50-0510-4ee6-a758-67c78572b7e3",
                "alias": "Light",
                "alias_kr": "연한맛",
                "alias_es": "Light (ES)",
                "optionCategoryName": "Strength",
                "optionCategoryName_kr": "커피 맛",
                "optionCategoryName_es": "Strength (ES)",
                "price": 0.00,
                "isEnabled": true,
                "extraInformation": []
            },
            {
                "optionCategoryId": "9e0ec2c7-1965-4bbb-92c8-795fe7f8cfab",
                "viewIndex": 1,
                "optionId": "723c6250-06c4-4d12-a283-625997c1784f",
                "alias": "Deep",
                "alias_kr": "진한맛",
                "alias_es": "Deep (ES)",
                "optionCategoryName": "Strength",
                "optionCategoryName_kr": "커피 맛",
                "optionCategoryName_es": "Strength (ES)",
                "price": 0.00,
                "isEnabled": true,
                "extraInformation": []
            }
        ]
    },
    {
        "menuId": "7c4d66ee-b966-44d8-9e5f-8245416c724e",
        "priceUnit": "USD",
        "description": "Description in English",
        "description_kr": "한국어 설명",
        "description_es": "Descripción en español",
        "alias": "Iced Americano",
        "alias_kr": "아이스 아메리카노",
        "alias_es": "Iced Americano(es)",
        "price": 3.10,
        "menuImage": "absolute_path_to_image",
        "availability": true,
        "categoryId": "cec9aba0-4771-4bdd-909f-ca03f617c483",
        "categoryName": "Coffee (Iced)",
        "categoryName_kr": "아이스 커피류",
        "categoryName_es": "Coffee(es)",
        "isTumblerEnabled": false,
        "options": [
            {
                "optionCategoryId": "9e0ec2c7-1965-4bbb-92c8-795fe7f8cfab",
```

```
                    "viewIndex": 0,
                    "optionId": "cb513f50-0510-4ee6-a758-67c78572b7e3",
                    "alias": "Light",
                    "alias_kr": "연한맛",
                    "alias_es": "Light (ES)",
                    "optionCategoryName": "Strength",
                    "optionCategoryName_kr": "커피 맛",
                    "optionCategoryName_es": "Strength (ES)",
                    "price": 0.00,
                    "isEnabled": true,
                    "extraInformation": []
                },
                {
                    "optionCategoryId": "9e0ec2c7-1965-4bbb-92c8-795fe7f8cfab",
                    "viewIndex": 1,
                    "optionId": "723c6250-06c4-4d12-a283-625997c1784f",
                    "alias": "Deep",
                    "alias_kr": "진한맛",
                    "alias_es": "Deep (ES)",
                    "optionCategoryName": "Strength",
                    "optionCategoryName_kr": "커피 맛",
                    "optionCategoryName_es": "Strength (ES)",
                    "price": 0.00,
                    "isEnabled": true,
                    "extraInformation": []
                }
            ]
        },
        {
            "menuId": "890696f8-5e09-4281-ae0e-54f089d032c7",
            "priceUnit": "USD",
            "description": "Description in English",
            "description_kr": "한국어 설명",
            "description_es": "Descripción en español",
            "alias": "Hot Cafe Latte",
            "alias_kr": "핫 카페라떼",
            "alias_es": "Hot Cafe Latte (es)",
            "price": 3.10,
            "menuImage": "absolute_path_to_image",
            "availability": true,
            "categoryId": "aac47eb7-65a5-46fa-b1a2-9f414c1ec754",
            "categoryName": "Coffee (Hot)",
            "categoryName_kr": "핫 커피류",
            "categoryName_es": "Coffee(es)",
            "isTumblerEnabled": true,
            "options": [
                {
                    "optionCategoryId": "8f0f6a90-3707-48af-9a33-caec64ecf843",
                    "viewIndex": 0,
                    "optionId": "d3b55e4a-cdc1-42d7-9983-22058474ec03",
                    "alias": "Use Tumbler",
                    "alias_kr": "텀블러 사용",
                    "alias_es": "Use Tumbler (ES)",
                    "optionCategoryName": "Takeout option",
                    "optionCategoryName_kr": "테이크 아웃 옵션",
                    "optionCategoryName_es": "Takeout option (ES)",
```

```
                "price": 0.00,
                "isEnabled": true,
                "extraInformation": []
            }
        ]
    },
    {
        "menuId": "dba481fd-9e0f-4436-b5a3-4a1c7aa4ba9b",
        "priceUnit": "USD",
        "description": "Description in English",
        "description_kr": "한국어 설명",
        "description_es": "Descripción en español",
        "alias": "Iced Cafe Latte",
        "alias_kr": "아이스 카페라떼",
        "alias_es": "Iced Cafe Latte(es)",
        "price": 3.10,
        "menuImage": "absolute_path_to_image",
        "availability": true,
        "categoryId": "aac47eb7-65a5-46fa-b1a2-9f414c1ec754",
        "categoryName": "Coffee (Iced)",
        "categoryName_kr": "아이스 커피류",
        "categoryName_es": "Coffee(es)",
        "isTumblerEnabled": false,
        "options": [
        ]
    },
    {
        "menuId": "3dfd7513-b60d-4c72-b38c-babd8af27ff6",
        "priceUnit": "USD",
        "description": "Description in English",
        "description_kr": "한국어 설명",
        "description_es": "Descripción en español",
        "alias": "Hot Cappuccino",
        "alias_kr": "핫 카푸치노",
        "alias_es": "Hot Cappuccino (es)",
        "price": 3.10,
        "menuImage": "absolute_path_to_image",
        "availability": true,
        "categoryId": "aac47eb7-65a5-46fa-b1a2-9f414c1ec754",
        "categoryName": "Coffee (Hot)",
        "categoryName_kr": "핫 커피류",
        "categoryName_es": "Coffee(es)",
        "isTumblerEnabled": true,
        "options": [
            {
                "optionCategoryId": "8f0f6a90-3707-48af-9a33-caec64ecf843",
                "viewIndex": 0,
                "optionId": "d3b55e4a-cdc1-42d7-9983-22058474ec03",
                "alias": "Use Tumbler",
                "alias_kr": "텀블러 사용",
                "alias_es": "Use Tumbler (ES)",
                "optionCategoryName": "Takeout option",
                "optionCategoryName_kr": "테이크 아웃 옵션",
                "optionCategoryName_es": "Takeout option (ES)",
                "price": 0.00,
                "isEnabled": true,
```

```
                "extraInformation": []
            }
        ]
    },
    {
        "menuId": "efb299a8-4aed-4130-89a2-38a0c3236817",
        "priceUnit": "USD",
        "description": "Description in English",
        "description_kr": "한국어 설명",
        "description_es": "Descripción en español",
        "alias": "Iced Cappuccino",
        "alias_kr": "아이스 카푸치노",
        "alias_es": "Iced Cappuccino(es)",
        "price": 3.10,
        "menuImage": "absolute_path_to_image",
        "availability": true,
        "categoryId": "aac47eb7-65a5-46fa-b1a2-9f414c1ec754",
        "categoryName": "Coffee (Iced)",
        "categoryName_kr": "아이스 커피류",
        "categoryName_es": "Coffee(es)",
        "isTumblerEnabled": false,
        "options": [
        ]
    },
    {
        "menuId": "890f6f57-51a6-415a-b2de-9e1a505dafa8",
        "priceUnit": "USD",
        "description": "Description in English",
        "description_kr": "한국어 설명",
        "description_es": "Descripción en español",
        "alias": "Hot Cafe Mocha",
        "alias_kr": "핫 카페 모카",
        "alias_es": "Hot Cafe Mocha (es)",
        "price": 3.10,
        "menuImage": "absolute_path_to_image",
        "availability": true,
        "categoryId": "aac47eb7-65a5-46fa-b1a2-9f414c1ec754",
        "categoryName": "Coffee (Hot)",
        "categoryName_kr": "핫 커피류",
        "categoryName_es": "Coffee(es)",
        "isTumblerEnabled": false,
        "options": [

        ]
    },
    {
        "menuId": "b9dca819-00f6-41b7-a9d7-098552ff01ec",
        "priceUnit": "USD",
        "description": "Description in English",
        "description_kr": "한국어 설명",
        "description_es": "Descripción en español",
        "alias": "Iced Cafe Mocha",
        "alias_kr": "아이스 카페모카",
        "alias_es": "Iced Cafe Mocha(es)",
        "price": 3.10,
        "menuImage": "absolute_path_to_image",
```

```
            "availability": true,
            "categoryId": "aac47eb7-65a5-46fa-b1a2-9f414c1ec754",
            "categoryName": "Coffee (Iced)",
            "categoryName_kr": "아이스 커피류",
            "categoryName_es": "Coffee(es)",
            "isTumblerEnabled": false,
            "options": [
            ]
        },
        {
            "menuId": "10c4ea1e-fc3d-4522-b318-72ea227036c1",
            "priceUnit": "USD",
            "description": "Description in English",
            "description_kr": "한국어 설명",
            "description_es": "Descripción en español",
            "alias": "Hot Vanila Latte",
            "alias_kr": "핫 바닐라 라떼",
            "alias_es": "Hot Vanila Latte (es)",
            "price": 3.10,
            "menuImage": "absolute_path_to_image",
            "availability": true,
            "categoryId": "aac47eb7-65a5-46fa-b1a2-9f414c1ec754",
            "categoryName": "Coffee (Hot)",
            "categoryName_kr": "핫 커피류",
            "categoryName_es": "Coffee(es)",
            "isTumblerEnabled": false,
            "options": [

            ]
        },
        {
            "menuId": "8de7c76b-4a85-4e6f-a48d-7c3c06cc5a05",
            "priceUnit": "USD",
            "description": "Description in English",
            "description_kr": "한국어 설명",
            "description_es": "Descripción en español",
            "alias": "Iced Vanila Latte",
            "alias_kr": "아이스 바닐라 라떼",
            "alias_es": "Iced Vanila Latte (es)",
            "price": 3.10,
            "menuImage": "absolute_path_to_image",
            "availability": true,
            "categoryId": "aac47eb7-65a5-46fa-b1a2-9f414c1ec754",
            "categoryName": "Coffee (Iced)",
            "categoryName_kr": "아이스 커피류",
            "categoryName_es": "Coffee(es)",
            "isTumblerEnabled": false,
            "options": [
            ]
        },
        {
            "menuId": "b7875d51-a9c8-4171-a789-15851a39774e",
            "priceUnit": "USD",
            "description": "Description in English",
            "description_kr": "한국어 설명",
            "description_es": "Descripción en español",
```

```
            "alias": "Hot Caramel Macchiato",
            "alias_kr": "핫 캬라멜 마끼야또",
            "alias_es": "Hot Caramel Macchiato (es)",
            "price": 3.10,
            "menuImage": "absolute_path_to_image",
            "availability": true,
            "categoryId": "aac47eb7-65a5-46fa-b1a2-9f414c1ec754",
            "categoryName": "Coffee (Hot)",
            "categoryName_kr": "핫 커피류",
            "categoryName_es": "Coffee(es)",
            "isTumblerEnabled": false,
            "options": [

            ]
        },
        {
            "menuId": "30cf3d4a-a35e-4a1a-918c-499c4d11c643",
            "priceUnit": "USD",
            "description": "Description in English",
            "description_kr": "한국어 설명",
            "description_es": "Descripción en español",
            "alias": "Iced Caramel Macchiato",
            "alias_kr": "아이스 캬라멜 마끼야또",
            "alias_es": "Iced Caramel Macchiato (es)",
            "price": 3.10,
            "menuImage": "absolute_path_to_image",
            "availability": true,
            "categoryId": "aac47eb7-65a5-46fa-b1a2-9f414c1ec754",
            "categoryName": "Coffee (Iced)",
            "categoryName_kr": "아이스 커피류",
            "categoryName_es": "Coffee(es)",
            "isTumblerEnabled": false,
            "options": [
            ]
        },
        {
            "menuId": "f89f6435-32c4-429d-ab6f-46c41e2cd6cd",
            "priceUnit": "USD",
            "description": "Description in English",
            "description_kr": "한국어 설명",
            "description_es": "Descripción en español",
            "alias": "Hot Chocolate Milk",
            "alias_kr": "핫 초코 우유",
            "alias_es": "Hot Chocolate Milk (es)",
            "price": 3.10,
            "menuImage": "absolute_path_to_image",
            "availability": true,
            "categoryId": "aac47eb7-65a5-46fa-b1a2-9f414c1ec754",
            "categoryName": "Non-Coffee (Hot)",
            "categoryName_kr": "핫 음료류",
            "categoryName_es": "Non-Coffee(es)",
            "isTumblerEnabled": false,
            "options": [

            ]
        },
```

```
        {
            "menuId": "b93862e8-d3de-4cd2-99ae-9178bfe470b9",
            "priceUnit": "USD",
            "description": "Description in English",
            "description_kr": "한국어 설명",
            "description_es": "Descripción en español",
            "alias": "Iced Chocolate Milk",
            "alias_kr": "아이스 초코 우유",
            "alias_es": "Iced Chocolate Milk (es)",
            "price": 3.10,
            "menuImage": "absolute_path_to_image",
            "availability": true,
            "categoryId": "aac47eb7-65a5-46fa-b1a2-9f414c1ec754",
            "categoryName": "Non-Coffee (Iced)",
            "categoryName_kr": "아이스 음료류",
            "categoryName_es": "Non-Coffee(es)",
            "isTumblerEnabled": false,
            "options": [
            ]
        },
        {
            "menuId": "0475408f-79d6-4581-a308-f1353e9528ee",
            "priceUnit": "USD",
            "description": "Description in English",
            "description_kr": "한국어 설명",
            "description_es": "Descripción en español",
            "alias": "Hot Caramel Milk",
            "alias_kr": "핫 캬라멜 우유",
            "alias_es": "Hot Caramel Milk (es)",
            "price": 3.10,
            "menuImage": "absolute_path_to_image",
            "availability": true,
            "categoryId": "aac47eb7-65a5-46fa-b1a2-9f414c1ec754",
            "categoryName": "Non-Coffee (Hot)",
            "categoryName_kr": "핫 음료류",
            "categoryName_es": "Non-Coffee(es)",
            "isTumblerEnabled": false,
            "options": [

            ]
        },
        {
            "menuId": "8e34796c-a4fe-4529-a77a-8be43f117030",
            "priceUnit": "USD",
            "description": "Description in English",
            "description_kr": "한국어 설명",
            "description_es": "Descripción en español",
            "alias": "Iced Caramel Milk",
            "alias_kr": "아이스 캬라멜 우유",
            "alias_es": "Iced Caramel Milk (es)",
            "price": 3.10,
            "menuImage": "absolute_path_to_image",
            "availability": true,
            "categoryId": "aac47eb7-65a5-46fa-b1a2-9f414c1ec754",
            "categoryName": "Non-Coffee (Iced)",
            "categoryName_kr": "아이스 음료류",
```

```
            "categoryName_es": "Non-Coffee(es)",
            "isTumblerEnabled": false,
            "options": [
            ]
        },
        {
            "menuId": "6d4c71a3-46c1-478e-9b9f-24a56a337530",
            "priceUnit": "USD",
            "description": "Description in English",
            "description_kr": "한국어 설명",
            "description_es": "Descripción en español",
            "alias": "Iced Grapefruit Ade",
            "alias_kr": "자몽 에이드",
            "alias_es": "Iced Grapefruit Ade (es)",
            "price": 3.10,
            "menuImage": "absolute_path_to_image",
            "availability": true,
            "categoryId": "52429859-858e-4e0c-8b7f-9432681fa726",
            "categoryName": "Ade",
            "categoryName_kr": "에이드류",
            "categoryName_es": "Ade(es)",
            "isTumblerEnabled": false,
            "options": [
            ]
        },
        {
            "menuId": "64f7a4fa-5229-4538-9a1d-de9fe43db73f",
            "priceUnit": "USD",
            "description": "Description in English",
            "description_kr": "한국어 설명",
            "description_es": "Descripción en español",
            "alias": "Lemonade",
            "alias_kr": "레몬 에이드",
            "alias_es": "Lemonade(es)",
            "price": 3.10,
            "menuImage": "absolute_path_to_image",
            "availability": true,
            "categoryId": "52429859-858e-4e0c-8b7f-9432681fa726",
            "categoryName": "Ade",
            "categoryName_kr": "에이드류",
            "categoryName_es": "Ade(es)",
            "isTumblerEnabled": false,
            "options": [
            ]
        },
        {
            "menuId": "64f7a4fa-5229-4538-9a1d-de9fe43db73f",
            "priceUnit": "USD",
            "description": "Description in English",
            "description_kr": "한국어 설명",
            "description_es": "Descripción en español",
            "alias": "Green Grape Ade",
            "alias_kr": "청포도 에이드",
            "alias_es": "Green Grape Ade(es)",
            "price": 3.10,
            "menuImage": "absolute_path_to_image",
```

```
            "availability": true,
            "categoryId": "52429859-858e-4e0c-8b7f-9432681fa726",
            "categoryName": "Ade",
            "categoryName_kr": "에이드류",
            "categoryName_es": "Ade(es)",
            "isTumblerEnabled": false,
            "options": [
            ]
        }
    ]
}
```