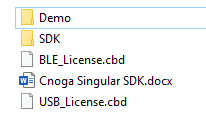
Cnoga Singular SDK – How To

# SDK folder structure:



1. SDK - All the SDK library files resides in this folder.
2. Demo – Console app for demonstration purpose.

## Demo Compilation:

1. Use ***Visual Studio 2017***.
2. Use **x86** configuration.
3. Copy all the files located within the SDK folder into the x86 ***Debug/Release*** compiled folders and complete with missing (not referenced, but required in runtime) Microsoft DLLs.
4. Copy the "**license.cbd**" file into the x86 compiled folders (i.e.: *Debug*/*Release*) – should resides next to the **.exe** file.

# Libraries:

## ***CNoga.Singular.SDK* library**:

### Namespace: *CNoga.Singular.SDK*

1. **IDeviceDetectorManager** (Interface):
   1. Detection of Cnoga **MTX** devices (**BG**/**HG** derivatives included).
   2. Connection to a single Cnoga MTX device.
   3. Disconnection from Cnoga MTX device.
   4. Subscription to events:
      1. Device Arrived
      2. Device Removed
      3. Device Disconnected
      4. Devices Detection Finished
2. **DeviceDetectorManager<T>** (Base **abstract** **generic** class) – Implements the Interface above.
3. **BleDeviceDetectorManager** (Concrete class - Inherits the base class above) **–** Manages all the **BLE** communication with the device during its initial **detection phase.**
4. **UsbDeviceDetectorManager** (Concrete class - Inherits the base class above) **–** Manages all the **USB** communication with the device during its initial **detection phase.**

## **CNoga.FirmwareUpgrader.Types library**:

### Namespace: CNoga.FirmwareUpgrader.Types.SDK\_APIs

1. **IDetectableDevice** (Interface) - Represents the initially detected Cnoga MTX device (both **USB** and **BLE** communications).
2. **IOPSWDevice** (Interface) - Represents the connected (operational) Cnoga MTX device (both **USB** and **BLE** communications).

Responsibilities: Communication with the Cnoga MTX device and data streaming of measurements results.

* 1. Allows sending these following requests:

1. **Start Measurement**
2. **Stop Measurement**
3. **Get Battery Status**
   1. Allows subscription to the following events:
   2. **Measurement Started**
   3. **Measurement Stopped**
   4. **Measurement Arrived**

# SDK bring up:

## License:

1. License file name: **license.cbd**, located in SDK.
2. Way to load the SDK for **USB** communication: **DeviceDetectorManager<UsbDeviceDetectorManager>.GetManagerInstance(licesnePath)**.
3. Way to load the SDK for **BLE** communication: **DeviceDetectorManager<BleDeviceDetectorManager>.GetManagerInstance(licesnePath)**.
4. Both of the above returns the Interface - **IDeviceDetectorManager**.

## IDeviceDetectorManager Properties:

1. **LoggerFactory –** An option for the consumer to utilize a built-in Microsoft logger with enhanced logging capabilities.
2. **Devices** – Returns a generic read-only collection of type ***ReadOnlyCollection<IDetectableDevice>***. This collection encompasses the currently detected devices.
3. **IsDetecting** – Returns Boolean value. Indication whether the detection process is still running or not.

## IDeviceDetectorManager Methods:

1. **StartDetection** – Commencing the detection process for searching and detecting devices.
2. **StopDetection** – Termination of the detection process.
3. **OpenDevice** –Receives an argument of **IDetectableDevice** instance. Returns an **IOPSWDevice** instance.

Through this method the device becomes from a detectable device into an operational device and a communication with the device is achieved. Only one detected device can become operational at any given time.

1. **CloseDevice** - Receives an argument of **IOPSWDevice** instance. Through this method the device ceased to be operational anymore and no more communication with the device is possible.

## IDeviceDetectorManager Events:

1. **DevicesDetectionFinished** – Invoked after the device detection phase is finished (for the **BLE** communication - operation takes **30 seconds** to complete; for the **USB** communication – the operation has **no duration**).
2. **DeviceArrived** – Invoked after device was detected (during detection).

Provides ***IDetectableDevice*** object.

1. **DeviceRemoved** – Invoked after device was removed (during detection).

Provides ***IDetectableDevice*** object.

1. **DeviceDisconnected** – Invoked after device was disconnected (during its work).

Provides ***IOPSWDevice*** object.

## IOPSWDevice Properties:

1. **IsOpen** – Returns Boolean value. Indication whether the device is currently in an operational connected state (ready for a transmission with its host or not).
2. **DeviceInformation** – Returns **IDeviceInformation** interface instance. It encompasses all the information data concerning the device itself.
3. **DeviceCapabilities** – Returns **DeviceCapabilities** class instance. It encompasses all the device's measurements capabilities (i.e.: Which bio-parameters it supports).

* Note: All the properties above are read-only (Getters).

## IOPSWDevice Methods:

1. **StartMeasurement** - Commanding the device to be in a measurable state (Ready to stream measurements data).
2. **StopMeasurement** - Commanding the device to return to its default non measurable state (Cuts off the measurements data stream).
3. **GetBatteryStatus** - Retrieving device's battery status (i.e.: Its charging level).

## IOPSWDevice Events:

1. **MeasurementStarted** – Invoked after measurement starts. This method is now obsolete.
2. **MeasurementStopped** – Invoked after measurement stops. This method is now obsolete.
3. **MeasurementsStatusChanged –** Replacement for the obsolete methods above.
4. **MeasurementArrived** – Invoked when measurement results arrive.

Provides a ***MeasurementsResults*** object.