Device Services - existing and work underway

This page contains a list of device services that currently exist, are under development, or are considered archived (potentially serving as example code at best).

Current Device Services

Random Number Generator (Go): Written with the Go SDK and demonstrates the simplest of device services - generating a random number (ints of different size) on a scheduled interval and sends the number as an Event/Reading to Core Data. It also allows the random number range to be set through a request via the Command micro service (demonstrating commands).

https://github.com/edgexfoundry/device-random

Modbus (Go): Written with the Go SDK. Provides generic connectivity to read and write Modbus input status, coils, input registers and holding registers. This implementation works for both TCP and RTU connections.

https://github.com/edgexfoundry/device-modbus-go

MQTT (Go): Written with the Go SDK. Provides the connectivity to listen to and publish to the specific MQTT topics.

https://github.com/edgexfoundry/device-mqtt-go

SNMP (Go): Written with the Go SDK and used to provide data ingestion and command/control of a Dell N-series PoE switch. Future development plans include abstraction for use with other switch types.

https://github.com/edgexfoundry/device-snmp-go

Grove (C): Written with the C SDK. A device service to communicate with the Grove PI sensors.

https://github.com/edgexfoundry/device-grove-c

Virtual Device (Go): Written with the Go SDK. This device service simulates different kinds of devices with different data types to generate Events and Readings to the Core Data micro service, and allows sending commands through the Command micro service.

https://github.com/edgexfoundry/device-virtual-go

Camera (Go): Written with the Go SDK and used to provide access to still images and RTP video streams from ONVIF compliant IP cameras. Also provides access to metadata on Axis cameras.

https://github.com/edgexfoundry-holding/device-camera-go

Rest (Go): Written with the Go SDK. This device service provides easy way for 3rd party applications, such as Point of Sale, CV Analytics, etc., to push data into EdgeX via the REST protocol. The current implementation is meant for one-way communication into EdgeX via async readings.

https://github.com/edgexfoundry/device-rest-go

BACnet (C): Wirtten with the C SDK. A device service to communicate with BACnet sensors/devices.

https://github.com/edgexfoundry/device-bacnet-c

Developed & In EdgeX Holding (approval needed by TSC to make part of EdgeX)

SNMP (Patlite) (Go): Written with the Go SDK and used to provide data ingestion and command/control of an SNMP driven Patlite signal tower, but could easily be copied and modified to support other SNMP supported devices.

https://github.com/edgexfoundry-holding/device-snmp-patlite-go

In Development

Contact the party associated with the development to get more background, status and potentially early code drops.

OPC UA (C): being developed by IoTech using the C SDK. A device service to communicate with OPC UA sensors/devices.

https://github.com/edgexfoundry-holding/device-opcua-c

Bluetooth (C): being developed by IoTech using the C SDK. A device service to communicate with BLE devices/sensors.

https://github.com/edgexfoundry-holding/device-bluetooth-c

GPS (C): being developed by IoTech and written in Go. A device service to communicate with GPS devices/sensors.

https://github.com/edgexfoundry-holding/device-gps

Archived (device services no longer supported or potentially functioning)

MQTT (Java): https://github.com/edgexfoundry/device-mqtt

Modbus (Java): https://github.com/edgexfoundry/device-modbus

SNMP (Java): https://github.com/edgexfoundry/device-snmp

Bluetooth (Java): https://github.com/edgexfoundry/device-bluetooth

Fischertecknic (Java): https://github.com/edgexfoundry/device-fischertechnik

Bacnet (Java): https://github.com/edgexfoundry/device-bacnet

Commercially available

from IOTech Systems

Commercial connectors (Current):

- File Exporter in C
- BLE in C
- Zigbee in C
- GPS in C
- CAN in C
- MEMS in C
- EtherCat in C
- Profinet in C
- There are additional commercial MQTT (Go), Modbus (Go), OPC-UA (in C) and BACnet (in C) that have added features over the community editions

Commercial connectors (Future):

- EtherNet/IP in C (summer 2020)
- CanOpen in C (summer 2020)
- OPC-UA Pub/Sub in C (summer 2020)
- ONVIF in C (summer 2020)