Industrial Integration made easy

ConnectorIO openhab bindings

ConnectorIO sp. z o.o.

Version 2.5.1-M1-20200504, 2020-05-04T00:07:03Z

Table of Contents

Colophon

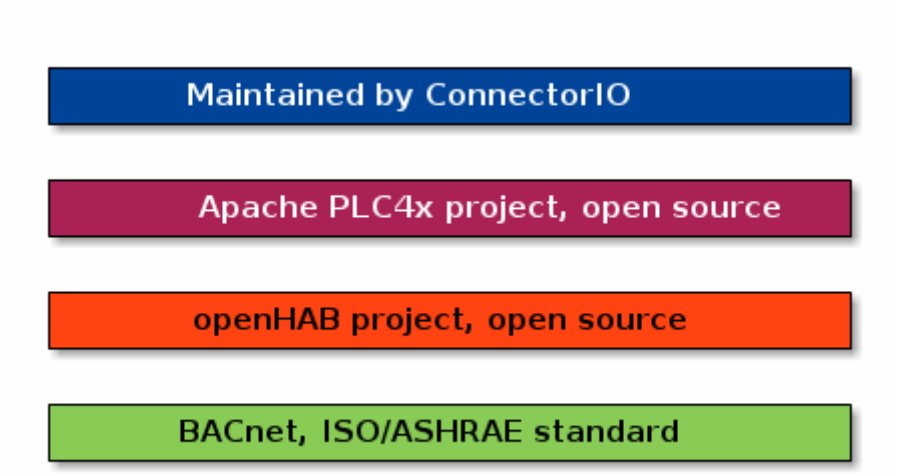
Published by ConnectorIO sp. z o.o.

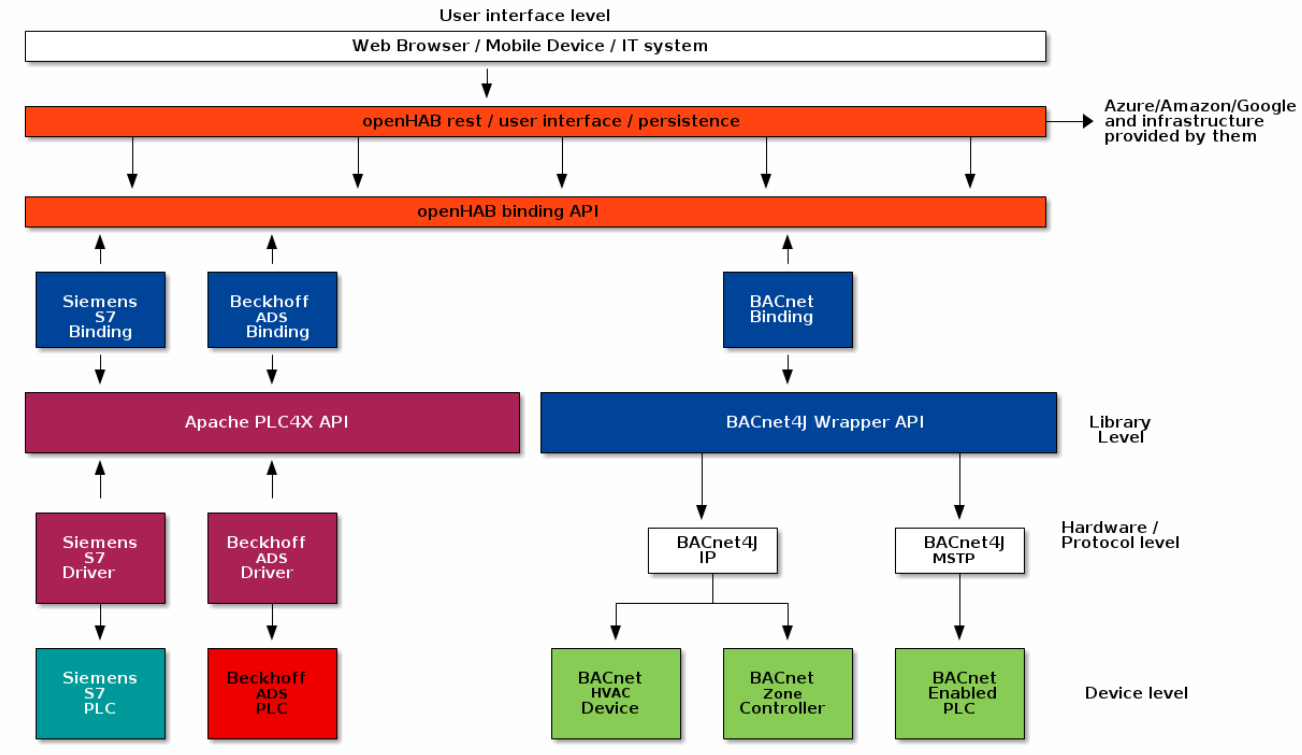
This document describes installation and configuration procedure of ConnectorIO bindings in openHAB. openHAB is popular home automation software, however its architecture is flexible and allows connectivity with various hardware. From our point of view the HAB part of that name is "Hardware Abstraction Bus", and it is used in such a way by us. Vanilla openHAB connect devices such as heating boilers, multimedia systems down to small sensors or smart meters. It has also support for Modbus, solar inverters, energy meters some PLC kinds and other serial based communications.

Chapter 1. Why?

ConnectorIO addons allow plugging typical industrial protocols and hardware to openHAB. You might be curious, why bother? Well, there are few reasons, but the primary one is really ability to visualize and store received data in a unified way. openHAB is quite compact when it is properly installed thus can be used as IoT gateway. With ConnectorIO bindings it will not become an IIoT gateway right away, however will gather connectivity abilities typical for these.

A below legend indicates with dark blue color components developed and solely maintained by ConnectorIO. Other colors indicate major dependencies used by us to provide desired functionality. To matter of fact, ConnectorIO bindings are third party addons for openHAB.





Chapter 2. Introduction

This documentation describes mainly how to configure industrial integrations supplied by ConnectorIO. In order to fully understand, "how" various assets are organized in openHAB, you need to learn some basic terms.

Main concepts which are used in openHAB are:

- bridges - usually represents a connection or complex element which can have things. Be aware, that bridge can be a child of another bridge too.
- things - representation of a given functionality, typically representing a piece of hardware or software function. Main responsibility of this element is to group operations of certain asset in one place.
- channels - each thing might have 0 and more channels which are inputs or outputs, generally speaking functions. A channel represents a single, atomic value or switch which can be controlled. One channel can have 0 or more linked items.
- items - items are elements which are created for visualisation and/or persistence reasons. Since one "thing" can have multiple "channels" the item allows selecting which channels of a device to track.
- links - it is a relation between item and channel. For each item there is one link.

Keep in mind above definitions because they will be referred in this documentation. Bindings provided by ConnectorIO give possibility to create bridges and things which do represent industrial hardware. First three elements are defined by shipped integration software, while later two are user responsibility.

If you still look for additional explanation please read more about concepts used in openHAB.

Chapter 3. Installation and upgrade

Each ConnectorIO binding is one archive. These archives have a `KAR` extension in their file name. If you need to use multiple protocols then you will need to repeat below procedure for each of them.

3.1. Step by step guide

First of all you need to install openHAB. Please follow official openHAB documentation and find instructions for your hardware platform.

Once your openHAB is up and running you can proceed to installation of bindings.

1. Download selected addons.
2. Locate your openHAB installation.
3. Find addons folder. With standard linux/openhabian installations it is `/usr/share/openhab2/addons`.
4. Copy KAR to this folder.
5. Launch browser.
6. Navigate to openHAB server user interface.
7. Pick paperUI from openHAB dashboard.
8. Navigate to Addons > Bindings.
9. Find integration you plan to install by typing protocol or brand name in search box.
10. Click install and wait for operation to complete.

3.2. Upgrade procedure

Because there is no guarantee which version of binding will start first you need to uninstall earlier version manually. The procedure for uninstalling is following:

1. Launch browser.
2. Navigate to openHAB server user interface.
3. Pick paperUI from openHAB dashboard.
4. Navigate to Addons > Bindings.
5. Find integration you plan to update.
6. Click uninstall.
7. Go to addons directory and remove selected `KAR` file.
8. Restart openHAB.

Do not worry about things which you created before. Their definitions and configurations will be kept in openHAB database. During upgrade, you will just loose ability to read their state. After successful installation of new version, they should automatically be back to "online" state.

Chapter 4. Bindings

4.1. BACnet

BACnet is shortcut of "Building Automation and Control Networks". As name suggests this protocol is very popular in building automation and control applications for HVAC but also other elements in infrastructure. BACnet is an open standard both in North America (ASHARE) and worldwide (ISO).

The protocol initially defined 23 standard object types, kinds of inputs/outputs and their role in automation systems. However this number grew up since 1990's.

The ConnectorIO BACnet binding allows to read values from the BACnet network as well as write (command) compatible devices from openHAB.

Currently, this binding uses library licensed under GPL license. Consider that if you plan to use related code commercially.

4.1.1. Supported hardware

All devices capable of communication BACnet.

4.1.2. Supported things and bridges

Thing	Type	Name	Description
ipv4	bridge	BACnet/IP bridge	The BACnet/IP bridge allows to connect devices supporting communication over BACnet/IP.
ip-device	bridge	BACnet/IP device	Device which is commendable over network connection.
mstp	bridge	BACnet/MSTP Bridge	The BACnet mstp bridge allows to connect devices communicating over serial interface (RS485).
mstp-device	bridge	BACnet/MSTP device	Device which is commendable over serial connection.

Following object types are supported:

- Analog input / output / value

- Binary input / output / value
- Multi state input / output / value
- Pulse converter

For test purposes below object types are available, however they are not officially supported.

- Schedule (not working)
- Character String
- Large analog
- Octet String
- Time
- Integer
- Positive Integer
- Date Time Pattern
- Time Pattern
- Date Pattern
- Accumulator

In order to start reading data from BACnet or commanding devices you need to create a valid connection (`mstp` or `ipv4`). Once done then you can start adding `mstp-device` or `ip-device` instances. Because BACnet objects have multiple properties they are modelled as separate things.

Currently binding supports only reading/writing of `Present Value` property.

Binding does not support Change of Value (COV) notifications yet.

4.1.3. Textual configuration

Below is example of a text configuration which you can upload to your openhab. It allows to use a regular version control system to track changes over time. The same can be done via user interface.

```
Bridge co7io-bacnet:ipv4:local "BACnet Network" [ localNetworkNumber=0,
localDeviceId=1010, localBindAddress=10.10.10.10, broadcastAddress="10.10.10.255" ] {

    Bridge ip-device "HVAC unit" [address="10.10.10.20", instance=1, network=0] {
        Thing analog-intput "Exhaust Temperature" [refreshInterval="500"] {
            channels:
                Type writeableNumber: ai1 "Temperature reading" [instance=1]
        }
        Thing analog-intput "Supply Temperature" [refreshInterval="500"] {
            channels:
                Type writeableNumber: ai2 "Temperature reading" [instance=2]
        }

    }
}
```

4.2. Beckhoff ADS

The "ADS" shortcut stands for "Automation Device Specification". It is often used with "AMS" which might be decoded as "Automation Message Specification". We do distinguish these terms cause they have different role. AMS is a routing and virtual networking layer and ADS is data exchange layer built on top of it. In order to communicate with ADS devices you must have AMS capabilities first.

Devices which are compatible with this protocol are usually Windows based PLCs of the same brand.

The ConnectorIO ADS binding allows to read values from the PLC as well as write it based on values set in openHAB.

4.2.1. Supported hardware

All devices capable of communication AMS/ADS.

4.2.2. Supported things and bridges

Thing	Type	Name	Description
ams	bridge	Beckhoff AMS/ADS Network	Defines AMS identifiers for communicating with other devices.
network	bridge	Beckhoff ADS Network Bridge	TCP/IP Connection to Beckhoff PLC
serial	bridge	Beckhoff ADS Serial Bridge	Serial port connection to Beckhoff PLC (note - not tested).

Thing	Type	Name	Description
ads	thing	Beckhoff ADS device	A PLC with ADS communication capabilities which can be polled for data.

In order to start reading data you need to create at least one `ams` bridge with `network` or `serial` connection. Once done then you can start adding `ads` devices. Be aware that there might be multiple `ads` devices for single connection allowing to group PLC inputs and outputs. If your PLC controls multiple motors then each of them can be created as a `ads` thing with separate state.

Because it is possible to configure polling (sampling) interval for each of above elements you can also group I/O by frequency of updates.

4.2.3. Textual configuration

Below is example of a text configuration which you can upload to your openhab. It allows to use a regular version control system to track changes over time. The same can be done via user interface.

```
Bridge co7io-plc4x-ads:ams:local "My connection" [ sourceAmsId="10.10.10.10.1.1",
sourceAmsPort="30000", ipAddress="10.10.10.10", broadcastAddress="10.10.10.255" ] {

    Bridge network [targetAmsId="10.10.10.20.1.1", targetAmsPort="851",
host="10.10.10.20"] {
        Thing ads "Inputs" [refreshInterval="500"] {
            channels:
                Type switch: Input001 "Input 001" [field="0x0000/0x0001:BOOL"]
                Type switch: Input002 "Input 002" [field="0x0000/0x0002:BOOL"]
                Type switch: Input003 "Input 003" [field="0x0000/0x0003:BOOL"]
                Type switch: Input004 "Input 004" [field="0x0000/0x0004:BOOL"]
                Type switch: Input005 "Input 005" [field="0x0000/0x0005:BOOL"]
        }

        Thing ads "Outputs" [refreshInterval="500"] {
            channels:
                Type switch: Output001 "Output 001" [field="0x0001/0x0001:BOOL"]
                Type switch: Output002 "Output 002" [field="0x0001/0x0002:BOOL"]
                Type switch: Output003 "Output 003" [field="0x0001/0x0003:BOOL"]
                Type switch: Output004 "Output 004" [field="0x0001/0x0004:BOOL"]
                Type switch: Output005 "Output 005" [field="0x0001/0x0005:BOOL"]
        }
    }
}
```

Create multiple things to manage inputs and outputs of your device in an organized way. All of them will reuse the same network or serial connection. Their role is to make navigation over complex installations easier.

4.3. Siemens S7

The "S7" shortcut stands for "Step 7".

Devices which are compatible with this protocol are usually Siemens PLCs. The ConnectorIO S7 binding allows to read values from the PLC as well as write it based on values set in openHAB.

4.3.1. Supported hardware

Models supported by this integration are S7-300, 400, 1200, 1500 and Logo which uses same protocol.

4.3.2. Supported things and bridges

Thing	Type	Name	Description
network	bridge	Siemens S7 TCP/IP Bridge	TCP/IP Connection to Siemens PLC
s7	thing	Siemens S7 device	A PLC with S7 communication capabilities which can be polled for data.

In order to start reading data you need to create at least one `network` bridge with `s7` device. Be aware that there might be multiple `s7` devices for a single connection allowing to group PLC inputs and outputs. If your PLC controls multiple motors then each of them can be created as a `s7` thing with separate state.

Because it is possible to configure polling (sampling) interval for each of above elements you can also group I/O by frequency of updates.

4.3.3. Textual configuration

Below is example of a text configuration which you can upload to your openhab. It allows to use a regular version control system to track changes over time. The same can be done via user interface.

```
Bridge co7io-plc4x-s7:network:local "S7-1200 vYXZ" [ host="10.10.10.10", rack=0,
slot=0 ] {

    Thing s7 "Inputs" [refreshInterval="500"] {
        channels:
            Type switch: Input001 "Input 001" [field="I0.1:BOOL"]
            Type switch: Input002 "Input 002" [field="I0.2:BOOL"]
            Type switch: Input003 "Input 003" [field="I0.3:BOOL"]
            Type switch: Input004 "Input 004" [field="I0.4:BOOL"]
            Type switch: Input005 "Input 005" [field="I0.5:BOOL"]
    }

    Thing s7 "Outputs" [refreshInterval="500"] {
        channels:
            Type switch: Output001 "Output 001" [field="Q0.1:BOOL"]
            Type switch: Output002 "Output 002" [field="Q0.2:BOOL"]
            Type switch: Output003 "Output 003" [field="Q0.3:BOOL"]
            Type switch: Output004 "Output 004" [field="Q0.3:BOOL"]
            Type switch: Output005 "Output 005" [field="Q0.5:BOOL"]
    }
}
```

Create multiple things to manage inputs and outputs of your device in an organized way. All of them will reuse the same network or serial connection. Their role is to make navigation over complex installations easier.

Chapter 5. Notes

5.1. Changelog

Below list contains summarized list of changes which been introduced over time. Above documentation describes latest state of integration. In case if you run older version of integration, please make sure you check earlier documentation.

Release 20200504

Released on **May 4th, 2020**. Compatible with openHAB 2.5.x.

- [Beckhoff] Support for discovery of network enabled devices.
- [Beckhoff] Fixes for doubled connections opened by binding.
- [Beckhoff] Reorganization of connection parameters, introduced `Beckhoff AMS/ADS Network` bridge.
- Fixes for possible errors while writing values to through Apache PLC4X API.

Release 20200102

Released on **January 2nd, 2020**. Compatible with openHAB 2.5.x.

- [Beckhoff] Fix for error in connection settings causing binding to fail.
- [Beckhoff, Siemens] Introduced support for numeric values.

Release 20191231

Released on **December 31th, 2019**. Compatible with openHAB 2.5.x.

- Initial release with possibility to connect a S7 or ADS device and read binary input/output.