Protocols, data models for command of metadata, and data representation

---

OPC-UA protocol S/W and microservices

Description

In these days, various data transmission protocols are used in the factory system according to the vendors and machines. To make efficient communication scheme from the data srouce to the data service for smart factory solution, data protocol shoud be integrated one way. OPC-UA(Open Platform Communication United Architecture) is one of the well-known data tranmission protocol for industrial technology. It can be the most desirable way with very high possibility, we expect. The OPC-UA protocol could be used entire smart factory data pipeline (industrial things - edge device - cloud service), so the microservices that used OPC-UA protocol could be located in the device service layer and the export service layer either. Therefore, the code repositories are separated into protocol S/W part and microservice part. The protocol S/W is provided with Java and C (based on open62541, until Feb. 2018) version. For the Java version protocol S/W, it is based on the OPC-UA milo open source. The external open sources are not included in the repository, when you try to build protocol S/W, the milo open source will be downloaded in your build environment. And the microservices are provided as Device Service (based on Device Service SDK of EdgeX) and Export Service (planned). Also when you try to build the OPC-UA device service, the OPC-UA protocol S/W is automatically downloaded.

Requirements

for OPC-UA protocol S/W

- OPC-UA client could establish connection with the OPC-UA server. (connect procedure from OPC-UA specification)
- OPC-UA client could browse data node on the OPC-UA server. (browse procedure from OPC-UA Service specification)
- OPC-UA client could access data node on the OPC-UA server. (attribute service procedure from OPC-UA Service specification)
- OPC-UA client could monitor data node on the OPC-UA server. (pub/sub procedure from OPC-UA Service specification)

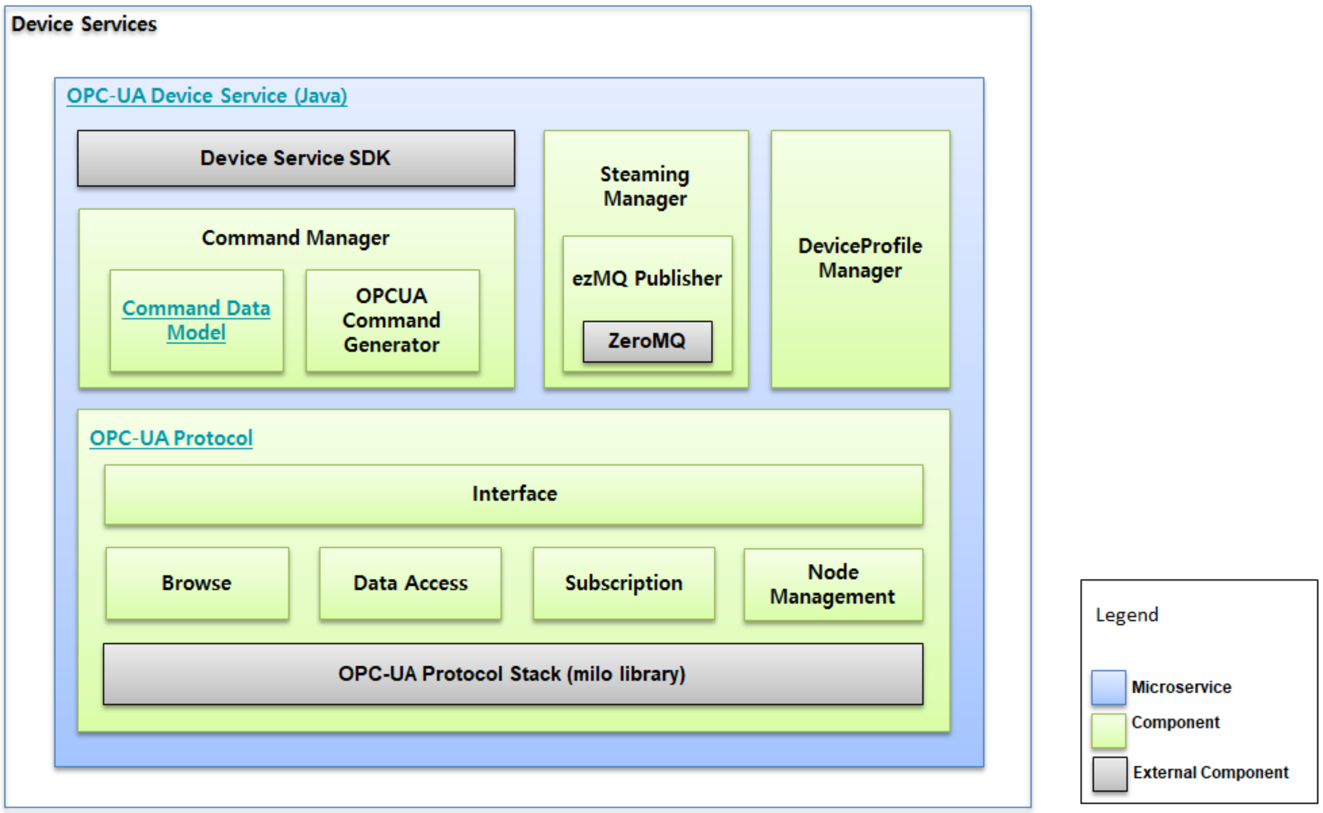
for OPC-UA device service

- System(OPC-UA device service) could process command requests from/to the OPC-UA protocol devices.
- System could publish data stream to the other microservices.
- System could update metadata with the OPC-UA protocol device attributes.

Design

The attached documents describe the high level design consideration.

- OPC-UA protocol S/W & device service



Code repository

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

https://github.com/edgexfoundry-holding/protocol-opcua-java

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

Current status

- OPC-UA protocol stack library has been provided in C and Java
    - Verified the CTT test using OPC-UA protocol S/W written in Java.
    - Verified the interoperability with PROSYS OPC-UA S/W.
    - Verified the interoperability with EdgeX core components (Metadata , Command, Coredata, Registry&Config)
- OPC-UA Device Service in Java from Smart Factory Project has been provided as an experimental version to EdgeX core working groups, and being developed from Device and Device SDK Working Group as one of the functionalities of the EdgeX regular release.
    - More information is available from Device and Device SDK Working Group Wiki.

Command Data Model

Command Data Model is for command of metadata. It will get you a easy expression of your request or response data which has a lot of categories and attributes. And Also, it's good at expressing a complex data which has depths. because Attribute Category can contain another Attribute. It provides some mandatory categories in below.

- version - version of the datamodel (default : edge-1.0)
- dataTitle - protocol or data title
- edgeElements - element unit like command of metadata.
- elementTitle - each element of edgeElements list can be expressed as operation of metadata.
- edgeAttributes - attribute list
- name - attribute name
- dataType - data type
- value - actual value of the data

Code repository

https://github.com/edgexfoundry-holding/datamodel-command-go https://github.com/edgexfoundry-holding/datamodel-command-java

Current status

- Command Data Model in Go and Java (which is command of metadata) are available.

Data Representation for AutomationML

Data Representation is for offering the schema the collected data in manufacturing domains in a standardized way. Currently Smart Factory Project has provided AutomationML, which is one of the global standard data representation in manufacturing industries. Per language, Data Representation with AutomationML is a library, which provides the way to present raw data (key/value based) to AutomationML standard format. C++, Go, Java, and Node.js libraries of Data Representation for AutomationML are bindings written over the C library of Data Representation for AutomationML.

- Transform raw data to AutomationML data in XML.
- Serialization / Deserialization AutomationML data using Protobuf.

Code repository

https://github.com/edgexfoundry-holding/datamodel-aml-c

https://github.com/edgexfoundry-holding/datamodel-aml-cpp

https://github.com/edgexfoundry-holding/datamodel-aml-go

https://github.com/edgexfoundry-holding/datamodel-aml-java

https://github.com/edgexfoundry-holding/datamodel-aml-node

Current status

- Data Representation for AutomationML are available in C, C++, Go, Java, and Node.js.