IOP Conf. Series: Journal of Physics: Conf. Series **870** (2017) 012006 doi:10.1088/1742-6596/870/1/012006

Dynamic mapping of EDDL device descriptions to OPC UA

Kofi Atta Nsiah, Manuel Schappacher, and Axel Sikora

Institute of Reliable Embedded Systems and Communication Electronics (ivESK), Offenburg University of Applied Sciences, 77652 Offenburg, Germany

kofi.nsiah@hs-offenburg.de

Abstract. OPC UA (Open Platform Communications Unified Architecture) is already a well-known concept used widely in the automation industry. In the area of factory automation, OPC UA models the underlying field devices such as sensors and actuators in an OPC UA server to allow connecting OPC UA clients to access device-specific information via a standardized information model. One of the requirements of the OPC UA server to represent field device data using its information model is to have advanced knowledge about the properties of the field devices in the form of device descriptions. The international standard IEC 61804 specifies EDDL (Electronic Device Description Language) as a generic language for describing the properties of field devices. In this paper, the authors describe a possibility to dynamically map and integrate field device descriptions based on EDDL into OPCUA.

1. Introduction

In today's automation industry, the increasing number and types of field devices from different manufacturers require integration into industrial control systems. As a result, the provision of a standardized access to these field devices allows fairly easy device management such as configuration, diagnosis and maintenance. So far, major device description and integration technologies are EDDL (Electronic Device Description Language) [1-3], FDT (Field Device Tool) [1, 2, 4] and FDI (Field Device Integration) [1, 2, 5]. These device integration technologies provide different mechanisms to solve the provision of standardized access to field device problem.

In OPC UA terms, this standardized access is provided by the information model of the server-side of OPC UA's client server architecture. An OPC UA server exposes its data via its information model. In order to represent the data of field devices, the OPC UA server has to have advanced knowledge about the field devices in the form of device descriptions. EDDL, FDT/DTM and FDI technologies define device descriptions and their integration into OPC UA. The device descriptions can either be used to create new information models before the OPC UA server is started or can be used to alter the information model of the already running OPC UA server. In this paper, the authors investigate the dynamic integration of device descriptions by altering the information model of the OPC UA server at run-time.

This paper is organized as follows: Chapter 2 provides a brief overview of the OPC UA technology in terms of its system architecture, information model and comparisons of the existing open source implementations of the technology. Chapter 3 presents the EDDL technology and provides a brief overview of other competing solutions. Finally, chapter 4 provides an insight into the implementation of the dynamic mapping of device descriptions based on the EDDL technology to OPC UA by the authors.

2. OPC UA

OPC UA, the successor to the legacy OPC technology, was developed to address the inherent technical defects associated with the classic OPC technology. Compared to the OPC technology, OPC UA provides a consistent and integrated address space that allows a single OPC UA server to integrate data, historical data, alarms and events [6]. Additionally, OPC UA defines a service model that provides a set of services that allows the data exposed in the address space of the OPC UA server to be accessed by connected OPC UA clients. The contents of the address space are described by well-defined information models.

2.1 OPC UA system architecture

The OPC UA system architecture models OPC UA clients and servers as interacting partners. The OPC UA server provides data by exposing its information model while the OPC UA client consumes the data provided by the information model of the OPC UA server. OPC UA services define the APIs (Application Programming Interfaces) that allow the OPC UA clients to interact with the OPC UA servers. Generally, a typical OPC UA client or server consists of three software layers briefly explained as follows:

- OPC UA Communication stack – implements the different data encoding mechanisms, message security and transport protocols defined by the OPC UA technology.
- OPC UA client or server SDK – implements common OPC UA functionalities including OPC UA services. The SDK is not a mandatory feature, however if implemented, reduces the development efforts of the client or server application.
- Client or Server application – uses the OPC UA communication stack and SDK to send and receive OPC UA messages. Specific functionalities can be implemented in this layer.

2.2 OPC UA information model

The OPC UA information model is based on the OPC UA Meta model [7]. The OPC UA information model provides not only the pure data but also exposes the semantic of the provided data. OPC UA information modelling is always done on the server-side however, they can be accessed and modified by connected OPC UA clients.

The data provided by the information model of OPC UA servers are represented as a set of nodes described by attributes and interconnected by references [8]. This data exposed by the OPC UA server is based on the eight base nodes and their attributes as shown in Figure 1.

Each OPC UA node has a unique purpose. For example, the Variable node is used to model the value of a field device while the DataType node models simple and structured data types of the Variable node's value. Depending on the purpose of an OPC UA node, they can have different sets of attributes. However, there are some attributes that are common to all OPC UA nodes.

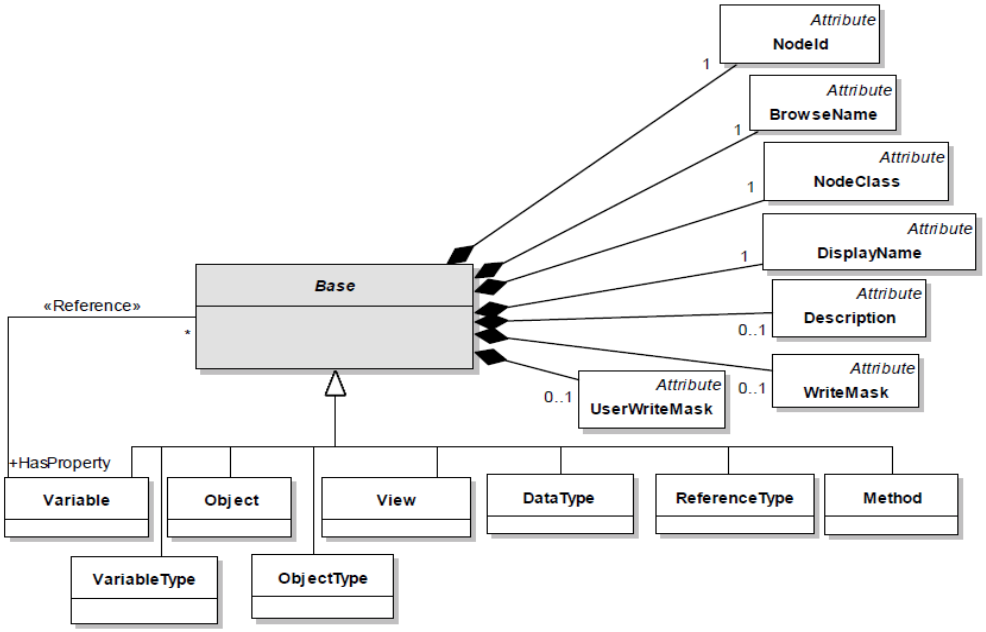

Figure 1. OPC UA Meta Model [7]

3. Device Descriptions

3.1. EDDL

EDDL (Electronic Device Description Language) is a structured text-based device description technology for describing the information that is accessible in digital devices. EDDL is an open technology with international standard status and can be applied to several field devices. The format of the described information is similar to an XML, HTML web page or SGML document. EDDL describes these digital devices in terms of device parameters and their dependencies, device functions, visual representations as well as interactions with control systems [9]. EDD (Electronic Device Description) is the resulting file deployed by device manufacturers that use the EDDL technology to describe their devices. A sample EDD describing the parameters contained in a NIKI current and temperature sensor is as shown in Figure 2.

```
/**
 * Sample EDD showing a sample description of
 * parameters  contained in a current or temperature
 * sensor.
 */

MANUFACTURER      66,
DEVICE_TYPE       0x070E,
DEVICE_REVISION   1,
DD_REVISION       1

/**
 * Describes the NIKI Current Sensor.
 */
VARIABLE       NIKI_Current_Sensor
{
     LABEL        NIKI_Current_Sensor;
     HELP         measures_actual_current;
     CLASS        CONTAINED & DYNAMIC;
     TYPE         FLOAT;
     HANDLING   READ;
}

/**
 * Describes the NIKI Temperature sensor.
 */
VARIABLE       NIKI_Temperature_Sensor
{
     LABEL        NIKI_Temperature_Sensor;
     HELP         measures_actual_temperature;
     CLASS        CONTAINED;
     TYPE         FLOAT;
     {
          DEFAULT_VALUE   0.0;
     }
     HANDLING  READ & WRITE;
}
```

Figure 2. EDD for NIKI Current and Temperature Sensor.

3.2. EDDL as a Data Model

The general motivation of this sub-chapter is to describe the EDDL as a data model. The EDDL technology provides a set of scalable language elements that are used in describing the field devices. EDDL language elements consist of identification elements, basic construction elements and special elements [9]. The identification elements (e.g. MANUFACTURER, DEVICE_TYPE) specify identification information that uniquely identifies the device description of a specific device type from a device manufacturer.

The basic construction elements support device descriptions in terms of device properties and related device functionalities. These elements define a set of associated attributes that refine their definition. Finally, special elements are EDDL extensions that support additional features. [9]. Table 1 presents some commonly used EDDL basic construction elements with their respective descriptions.

Table 1. EDDL language elements and their descriptions.

EDDL element	Description
VARIABLE	Describes parameters contained in a device.
METHOD	Defines executable subroutines.
COMMAND	Describes the structure and the addressing of variables in the device.
BLOCK	Describes a field device that is organized in logical blocks.
MENU	Organizes EDDL elements into a hierarchical structure.
LIKE	Used to create a new instance of an existing item (e.g. VARIABLE)
GRAPH	Used to display data from a device.

EDDL technology consists of EDD file and a host PC that hosts an EDDL interpreter. EDD file contains the descriptions of the devices as supplied by the device manufacturers. EDDs supplied in the form of ASCII text files are imported into the host PC, interpreted by the EDDL interpreter and executed.

3.3. EDDL vs FDT/DTM

EDDL and FDT/DTM (Field Device Tool/ Device Type Manager) technologies are two most widely used device description technologies. EDDL technology comprises two components – EDD file, which contains the device descriptions and a host application that reads and executes the EDD file. On the other hand, the FDT/DTM technology consist of two software components – DTM, a software component that contains device descriptions or descriptions of the communication component and FDT Frame Application that provides a common runtime environment for all DTMs [4]. Compared to the FDT/DTM technology, EDDL technology is operating system independent and portable. Recently, the FDI (Field Device Integration) technology has been developed to combine the advantages of both competing solutions in a single solution. The core component of the FDI technology is the FDI Package which contains the device descriptions and other optional components. The FDI device descriptions are based on the EDDL technology [5].

4. Implementation

4.1. Introduction

This section presents the OPC UA open source solutions [10] that were compared by the authors to finally select one for our implementation, presents the EDDL parser implementation and the EDDL to OPC UA mapping. This section also describes shortly the test setup to test whether the authors' approach of dynamically mapping devices based on the EDDL technology to OPC UA is feasible.

4.2 OPC UA open source solutions

Open source solutions of the OPC UA technology developed by the OPC Foundation, other companies, research institutes and academia as well as commercial solutions of the technology are available. In this paper, the authors compare the existing twelve open source implementations [10] based on some selected parameters. These parameters include the open source solution's support for the different OPC UA communication mappings, security profiles, transport profiles, OPC UA services, license types, community updates, programming languages used, operating system and hardware targets as well as tools required to enhance usability of the solutions. A snapshot of the parameters supported by three of the open source solutions, UA.Net [11], open62541 [12], AsNeG [13-14] are as shown in Figure 3.

Comparing the selected parameters of all the open source solutions, the AsNeG (Automation Service Next Generation) solution was preferred by the authors and selected as our reference OPC UA server implementation. AsNeG is open, free and is developed in C++.

Parameter		UA-.NET	open62541	AsNeG
Data encoding	OPC UA Binary	✓	✓	✓
	OPC UA XML	✓	×	×
Message security protocols	UA Secure Conversation	✓	✓	✓
Transport protocols	UA-TCP	✓	✓	✓
	HTTPS	✓	×	×
Security Profiles	None	✓	✓	✓
	Basic128Rsa15	✓	×	✓
	Basic256	✓	×	✓
	Basic256Sha256	✓	×	✓
Transport Profiles	UA-TCP UA-SC UA Binary	✓	✓	✓
	HTTPS UA Binary	✓	×	×
	HTTPS UA XML	✓	×	×
Discovery Service Set	FindServers	✓	✓	✓
	GetEndpoints	✓	✓	✓
	RegisterServer	✓	×	✓
SecureChannel Service Set	OpenSecureChannel	✓	✓	✓
	CloseSecureChannel	✓	✓	✓
Session Service Set	CreateSession	✓	✓	✓
	ActivateSession	✓	✓	✓
	CloseSession	✓	✓	✓
	Cancel	✓	×	✓

Figure 3. OPC UA open source solutions

4.3. EDDL Parser

EDD files can be distributed by device manufacturers either in text or binary format depending on what the host system requires. Binary EDD file is normally tokenized to a compressed binary to prevent tampering of the EDD file. These compressed files are relatively smaller, uploaded by the host system and decoded by DD services. On the other hand, text-based EDD files are interpreted by EDDI (Electronic Device Description Interpreter) in the host system where these EDD files are uploaded. In this paper, only text-based EDD files are considered.

The main motivation for implementing the EDDL parser is to extract the EDDL language elements and map them to OPC UA nodes. Several tools and libraries such as GNU Bison – YACC compatible parser generator, Boost Spirit library and Flex are possible solutions to implement the EDDL parser. The EDDL parser implemented by the authors has the following characteristics:

- Based on the Boost Spirit library
- Parses all EDDL language elements (e.g. language elements for identifying a specific EDD, describing data and communication components of field devices).
- Skips all whitespaces, pre-processor directives and all comments in the EDD file
- Parser can be extended to parse additional EDDL language elements that would be required in future.

4.4. EDDL to OPC UA mapping

The FDI technical specification part 5 defines the mapping of EDDL and other FDI package information to the FDI information model and its underlying OPC UA information models [15]. Based on these defined mappings, the authors have designed APIs to automatically map the parsed EDDL language elements or constructs to their corresponding OPC UA nodes or attributes. Even though these mappings are well-defined, not all EDDL language elements have their corresponding mappings. For example, while the EDDL VARIABLE language element maps to a Variable node in OPC UA, the EDDL COMMAND language element has no mapping. The reason is based on the purpose of the EDDL language element. OPC UA clients access Variable nodes in the OPC UA server to read or write sensor values but do not need to access the COMMAND element. As explained earlier, the EDDL COMMAND language element supports the mapping of the EDDL file communication elements to the underlying communication system.

The simple logic flow of how the EDDL to OPC UA mapping is achieved is as shown in Figure 4.

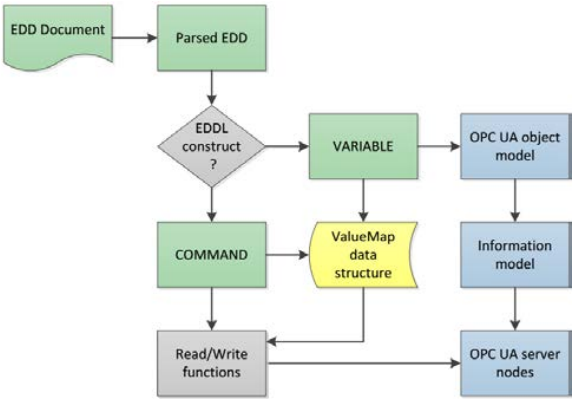

Figure 4. Flowchart showing EDDL to OPC UA mapping

In Figure 6, an EDDL file (EDD Document) with the VARIABLE and COMMAND EDDL language elements is parsed by our own EDDL parser. According to the FDI specification [15], the VARIABLE language element is mapped to create corresponding OPC UA variable nodes in the OPC UA server. The COMMAND language element with no defined mapping however allows the OPC UA server to have knowledge about how data can be read or written to the actual field devices or sensors. In between these operations, a complex data structure is implemented to store all relevant information.

4.5 Test setup

The objective of our test setup is to provide a platform to test the dynamic integration of device descriptions based on the EDDL technology into OPC UA. The test setup consists of an embedded platform, which hosts the server-side of the OPC UA implementation and a Windows-based PC running the client-side of the OPC UA implementation. In our set up, the OPC UA server open source implementation of the AsNeG solution is selected, fine-tuned and run on the embedded platform. On the other hand, UaExpert developed by Unified Automation is selected as our OPC UA client application and run on the Windows PC.

4.5.1 Embedded platform. The BBB (BeagleBone Black) [16] is used as the embedded platform to run the OPC UA server implementation from AsNeG. The BBB runs Debian OS, has a 512MB DDR3 RAM, 4GB 8-bit eMMC on-board flash storage and connects to the host PC via its USB host and Ethernet interfaces [16].

The tools required to build and run this implementation on the BBB are as follows:

- Boost version 1.54
- Gcc 4.9
- Cmake
- Openssl
- OPC UA server configuration file (.xml)
- EDDL files (.ddl)
- EDDL configuration files (.xml)

The build and run process of the OPC UA server implementation have been automated by bash scripts. To start the OPC UA server, a configuration file which defines information models, security profiles, transport profiles is required.

At run-time, the OPC UA server is able to load several EDDL configuration files. Each EDDL configuration file contain information about the EDDL file that shall be loaded by the OPC UA server and predefined ids and attributes of the OPC UA nodes that would be created by the OPC UA server. For example, the EDDL configuration file for the NIKI Current Sensor as shown in Figure 5 defines the corresponding EDD description of the NIKI Current Sensor in Figure 2. The OPC UA server therefore loads several EDDL configuration files, extracts the specified EDDL files, invokes the EDDL parser, maps the parsed EDDL languages elements to the corresponding OPC UA nodes as defined in [15] and finally creates the corresponding OPC UA nodes in the address space of the OPC UA server.

```
<?xml version="1.0" encoding="utf-8"?>
<EddlModel xmlns="http://eddl/EddlModel.xsd">
   <EddlPath File="@CONF_DIR@/Nodes/
     Current-Sensor.ddl" />
   <ObjectNode Id="998" BrowseName="NIKI
      Current Sensor" DisplayName="NIKI Current
     Sensor" Description="Current Sensor" />
   <VariableNodes FirstId="1200" />
</EddlModel>
```

Figure 5. EDDL Configuration file for NIKI Current Sensor.

4.5.2 Windows PC. A standard Windows PC is used to host the UaExpert application. UaExpert is a full-featured OPC UA client implementation that supports several OPC UA functionalities such as data access, alarms and conditions and historical access. In our test setup, the UaExpert is used to connect and visualize the OPC UA nodes created on the OPC UA server.

4.6 Test results

Figure 6 shows a snapshot of the UaExpert's visualized OPC UA nodes that were created dynamically by the OPC UA server running on the BBB after loading and parsing sample NIKI Current and Temperature Sensor EDDL configuration files and their corresponding EDD files.

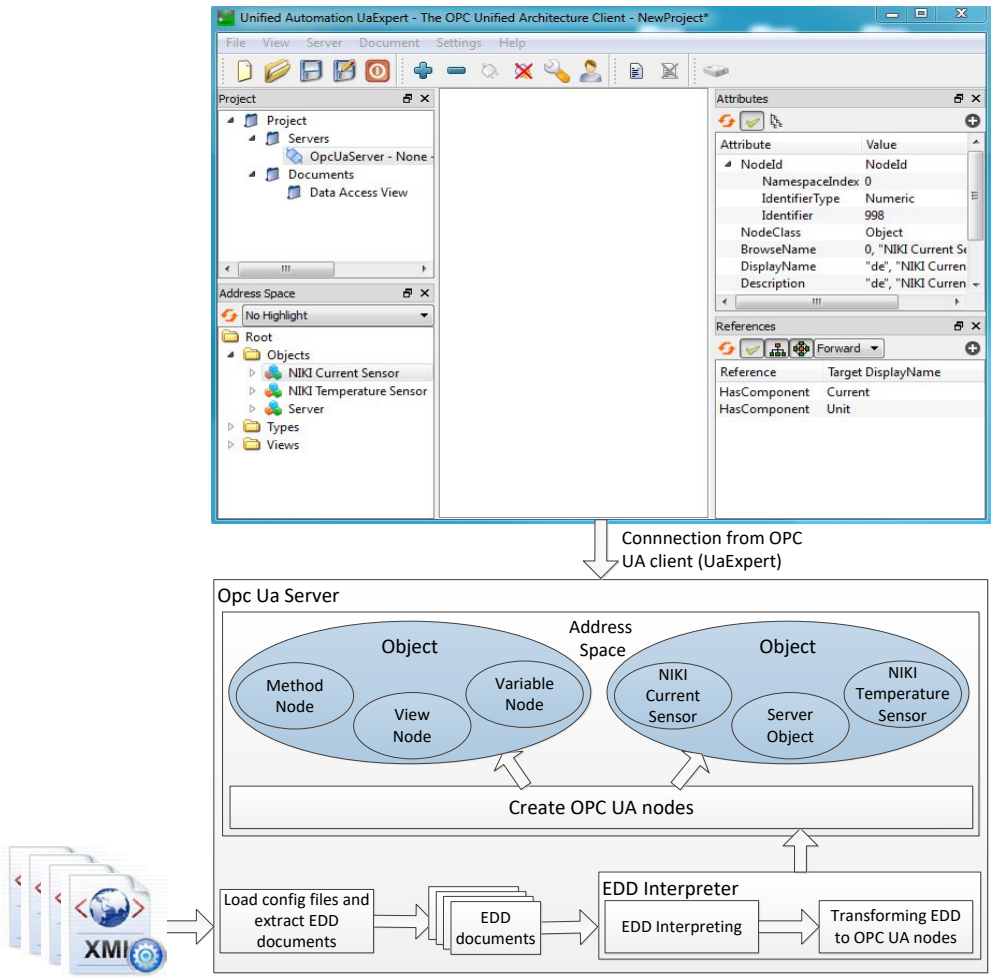

Figure 6. Snapshot showing UaExpert visualized nodes from the OPC UA server.

5. Outlook

We have presented in this paper our approach to dynamically map device descriptions based on EDDL to OPC UA. So far, our test results are based on Profibus, FOUDATION and HART devices. Currently, work is ongoing to integrate the open source LWM2M server implementation in the OPC UA server to allow the OPC UA server to access or get notified of data changes in the corresponding client, for example, a temperature sensor hosting a LWM2M client. Furthermore, we are working to support more devices based on protocols such as Modbus.

References

[1] M. Gunzert. Compatibility and interoperability in field device integration – A view on EDDL, FDT and FDI, (2015). In IEEE Conference Publications pp. 941-946G. Lütjering, J.C. Williams, *Titanium*, Springer, 2003.
[2] S. Runde, G. Wolf, M. Braun. EDDL and Semantic Web – from Field Device Integration (FDI) to Future Device Management (FDM), (2013). In IEEE Conference Publications pp. 1-8.
[3] www.eddl.org.
[4] www.fdtgroup.org.
[5] http://www.fdi-cooperation.com/technology.html.
[6] OPC Foundation, OPC Unified Architecture Specification Part 1: Overview and Concepts, Release 1.03 (2015)
[7] OPC Foundation, OPC Unified Architecture Specification Part 3: Address Space Model, Release 1.03 (2015)
[8] OPC Foundation, OPC Unified Architecture Specification Part 5: Information Model, Release 1.03 (2015)
[9] PROFIBUS Nutzerorganisation e.V., EDDL Specification for Profibus Device Description and Device Integration, (2005).
[10] https://github.com/open62541/open62541/wiki/List-of-Open-Source-OPC-UA-Implementations.
[11] https://github.com/OPCFoundation/UA-.NET
[12] http://open62541.org/
[13] http://wiki.asneg.de/wiki/index.php/Produkte/OpcUaStack
[14] https://81.169.197.52:8443/tree/OpcUaStack.git/master/src
[15] FDI Cooperation, FDI Technical Specification Part 5: Information Model (2015)
[16] https://cdn.sparkfun.com/datasheets/Dev/Beagle/BBB_SRM_C.pd

Acknowledgement

The work of this paper was partially supported within the project NIKI4.0 “NIcht disruptives Kit für die Evalution von Industrie 4.0” funded by Baden-Württemberg Stiftung GmbH. The authors are grateful for this support.