# Towards a web-of-things approach for OPC UA field device discovery in the industrial IoT

Quang-Duy Nguyen, Saadia Dhouib, Jean-Pierre Chanet, Patrick Bellot

# Towards a Web-of-Things Approach for OPC UA Field Device Discovery in the Industrial IoT


Quang-Duy NGUYEN*, Saadia DHOUIB*, Jean-Pierre CHANET†, and Patrick BELLOT‡

*Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France.

†Université Clermont Auvergne, INRAE, UR TSCF, 63178 Aubière, France

‡LTCI, Télécom Paris, Institut Polytechnique de Paris, 91120, Palaiseau, France

Email: quang-duy.nguyen@cea.fr, saadia.dhouib@cea.fr, jean-pierre.chanet@inrae.fr, patrick.bellot@telecom-paris.fr



***Abstract*—In the OPC UA standard, the OPC UA information model is the key to semantic interoperability. It refers to an organized structure representing resources in the form of OPC UA nodes residing in an OPC UA address space. Before runtime, an OPC UA server requires an OPC UA information model already filled with OPC UA nodes corresponding to the available resources of its field devices. It is tricky to the Industrial Internet of Things, a scenario where random field devices can join a system at any time. Indeed, the OPC UA server needs to dynamically discover such devices and update its OPC UA information model with new relevant OPC UA nodes. Regarding the above challenge, this paper introduces a Web-of-Things (WoT) approach that enables field devices to register with an OPC UA server to join the system without manual configuration. The approach relies on WoT Discovery and WoT Thing Description.**



***Index Terms*—Industry, Internet of Things, OPC UA, Web of Things, Thing Description, Discovery**


## I. Introduction

The fourth industrial revolution has been an inevitable technological event of this 21st century. It is about using the most advanced technologies in emerging fields, such as the Internet of Things (IoT), to improve industrial systems' automation, self-monitoring, and communication. The IoT is an ideal scenario in which heterogeneous computing devices, from resource-contained embedded systems to high-performance infrastructure, communicate and exchange data through Internet connections. Thus, it is an interoperability high-demanded environment. The Industrial Internet of Things (IIoT) is a specification of the IoT: computing devices in the IIoT belong to industrial systems. In this sense, the industrial systems in the IIoT must consider interoperability as a fundamental quality besides other conventional ones in the industry, such as robustness, reliability, and security.

Regarding interoperability, the Open Platform Communications Unified Architecture (OPC UA) is one of the best candidates. OPC UA is a widely-accepted industrial standard that receives support from many credible universities and industrial companies[1]. The newest version of OPC UA, v1.05, consists of mainly 22 specifications that provide conventions to implement industrial systems with reliability, security, and interoperability [1]. OPC UA provides transport protocols, data formats, and security policies for technical and syntactic interoperability. The semantic interoperability of OPC UA relies on the OPC UA information model. It is an organized structure to represent the resources of an OPC UA system in the form of OPC UA nodes residing in an OPC UA address space [2], [3]. In this paper, the term "OPC UA system" implies the concept of an OPC UA server and the field devices under its control. The reason to adopt this term instead of an industry-familiar term, such as "workstation," is that the concept is not only used in industrial applications but also used in other domains, for example, in agriculture [4]. Figure 2 shows an example of OPC UA systems. External devices can browse the OPC UA address space, query data, and call methods on the OPC UA system using OPC UA services [5].

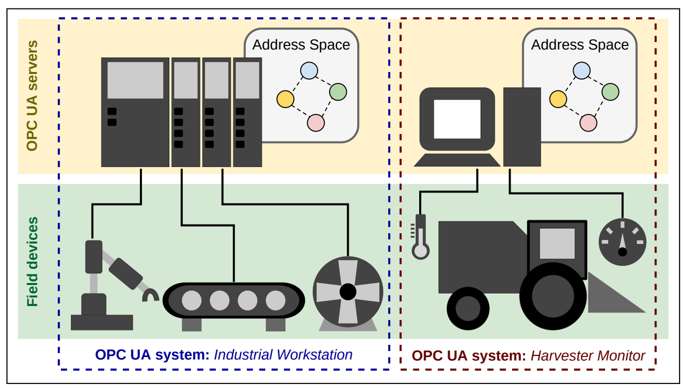


Fig. 1. Example of OPC UA systems

Often, the OPC UA information model of a system is formed statically before runtime [6], [7]. It means system developers manually determine field devices' shared resources, generate relevant OPC UA nodes, and add them to the OPC UA address space before launching the OPC UA server. However, this approach cannot satisfy the IIoT, where field devices can randomly join the OPC UA system at any time. The OPC UA server needs to discover new field devices and dynamically update its information model with new OPC UA nodes corresponding to their shared resources. This paper proposes a Web-of-Things (WoT) approach based on WoT Thing Description and WoT Discovery. It is called WoT4UA-dm as the short form of the WoT for OPC UA discovery mechanism. WoT4UA-dm enables field devices to join a running OPC UA system without manual configuration.

[1] https://opcfoundation.org/members/

The organization of this paper is as follows. Section II presents the background of our contribution: the WoT, WoT Thing Description, and WoT Discovery. Section III presents other discovery approaches in OPC UA. Section IV introduces the main contribution of this research: WoT4UA-dm. The next section focuses on an experiment to demonstrate this discovery mechanism. Section VI discusses the challenges ahead to improve WoT4UA-dm. Finally, a brief conclusion sums up this paper and presents our future works.

## II. Background

The background technologies of this contribution are the WoT, WoT Thing Description, and WoT Discovery. WoT[2] is a set of standards to improve the flexibility and interoperability in the IoT using Web technologies and paradigms. While the IoT shows its heterogeneous side with devices, networks, and formats, the WoT standards are tools to unify and enable things to speak in the same language. A thing can be a physical or virtual entity. Under the development of a World Wide Web Consortium (W3C) working group, WoT gains significant agreement and support from the community and progressively becomes an official standard contributing to the IoT. The principal works of WoT group contribute to six task forces: WoT architecture, WoT Thing Description, WoT Discovery, WoT Security, WoT Scripting API, and WoT Uses Cases.

This research focuses on WoT Thing Description and WoT Discovery. WoT Thing Description is a vocabulary that allows describing the metadata and interfaces of a thing. The vocabulary contains a collection of basic terms defined by its W3C working group. The details of each thing can also be specified using linked data or other vocabularies, such as the SAREF[3] and OM[4] ontologies. By default, a WoT Thing Description instance uses JSON encoding. Thus, WoT Thing Description can expose a thing's properties, capabilities, and resources understandable by both humans and machines.

WoT Discovery defines two mechanisms to manage WoT Thing Description instances: directory and self-description. The former mechanism is about storing description instances in a directory. Then, it enables to find a description instance by browsing the directory. The latter one is to let things manage their self-descriptions. Things will expose their self-descriptions to needers. To find things and directories to retrieve the description instances, WoT Discovery proposes five other mechanisms. Among the above mechanisms, our contribution adopts the WoT Discovery self-description.

## III. Related Work

OPC foundation proposes three discovery mechanisms in specification 12: Local Discovery Server, Local Discovery Server with Multicast Extension (LDS-ME), and Global Discovery Server [8]. Their goal is to make OPC UA servers available to other OPC UA clients and servers. Unlike our contribution, which focuses on discovering field devices on one OPC UA system, they address inter-OPC-UA systems.

Besides the official specification proposed by the OPC foundation, some other works contribute to the discovery in OPC UA. They are usually mending with the Plug and Produce (PnP) concept. PnP is an ability that a new field device joins an industrial system "without any configuration changes on the system, other devices, or any additional registration to the database" [9]. An industrial system with PnP needs to succeed in four phases: discovery, configuration, production, and reconfiguration [10]. The two first phases assure that the system detects and configure a new device. The third one is about the collaboration of the new device with the existing resources to accomplish tasks. The fourth one demands devices the adaptability, that is, to be reconfigured based on context changes. The last two phases are out of this paper's scope.

Another point to note is that self-description is not an original mechanism from WoT but is also used by other works [11]. The difference between them is the language to encode descriptions. AutomationML is a highly-used candidate for this job [12], [13].

## IV. OPC UA Field Devices Discovery Mechanism Using Web of Things Technologies

This section presents our main contribution: WoT4UA discovery mechanism for field devices to join a running OPC UA system. Of which, field devices host their self-descriptions as in WoT Discovery self-description. A description is encoded using the materials of WoT Thing Description. WoT4UA-dm includes four steps. The first step is to install one description encoded in JSON format to each field device. Second, the device reads its description and may transform it into a format agreed with the OPC UA server beforehand. Then, the device creates a registration message containing the device description and sends the message to the OPC UA server. Third, the OPC UA server extracts the necessary information from the received message. Based on the information, it creates a new OPC UA node in the OPC UA address space. Finally, the OPC UA server sends configuration messages to the device. Figure 2 illustrates the WoT4UA-dm.

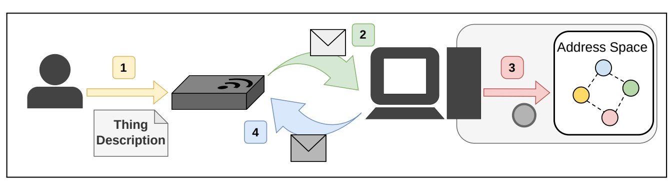


Fig. 2. Four steps of WoT4UA-dm

The WoT4UA-dm stays abstract: it is open for detailed designs and implementations. In the first step, system developers can write the description of a device manually or use a tool. A description must contain at least the information necessary for the OPC UA server to create an OPC UA node, such as the device's name, data source's name, and the data type. It should also contain information about the device's security, transport, and capabilities.

[2]https://www.w3.org/WoT/
[3]https://ontology.tno.nl/saref/
[4]https://github.com/HajoRijgersberg/OM

In the second step, the description information may be serialized into another format before being put into a registration message. Otherwise, the simplest solution is to use JSON format, so no serialization is needed. There are numerous possibilities to transfer the sensor device's registration message to the OPC UA server. For example, when they are in a Local Area Network (LAN), the sensor device can diffuse the registration message to a multicast address.

In the third step, system developers have two solutions to update the OPC UA address space with new OPC UA nodes. The first solution is to call a method or service predefined in the OPC UA server to demand to update its address space at runtime. The second solution is to use an operating program with the permissions to update the OPC UA information model and build/stop/start the OPC UA server. In this sense, the update is not at the OPC UA server's runtime.

Finally, the OPC UA server may send configuration messages back to field devices. Depending on system design, a configuration message may include configuration information or may work as an acknowledgment message.

## V. Proof-of-Concept Experiment

This proof-of-concept experiment consists of four components. Two Raspberry Pi[5] 4 play as two sensor devices. Each sensor device has one sensor which measures the temperature of its microprocessor. Respectively, they are called *thermometer_01* and *thermometer_02*. One Raspberry Pi 3B+ works as an OPC UA server. The OPC UA server is built based on the open62541[6] library. The three Raspberry Pi are in a local network, and they communicate using the OPC UA PubSub brokerless pattern, transport profile PubSub UDP UADP[7]. The two sensor devices work as two publishers, and the OPC UA server is also a subscriber. Globally, they form the data provider's side. The fourth component is a Linux laptop connecting to the OPC UA server via the Internet. It runs the OPC UA client UaExpert[8] and represents the data consumer's side. Figure 3 illustrates the use case's physical architecture.

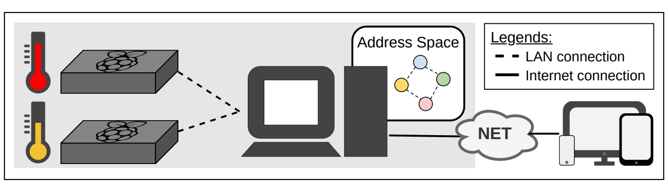


Fig. 3. Physical architecture of the proof-of-concept use case

The operation of the OPC UA system has two phases: discovery and data exchange. The discovery phase follows WoT4UA-dm. In detail, system developers manually write two minimal descriptions, then install them into two publishers. Table I shows the *thermometer_01*'s description. Each publisher sends registration messages to the multicast IPv4 address 224.0.0.1. A registration message has the description of a sensor device as its payload. After receiving a registration message, the OPC UA server extracts the necessary information to create a new OPC UA node. As this proof-of-concept scenario is minimal, the OPC UA server uses only the device's title, properties, and data type; and ignores the other information. It also assigns a *namespace* and a *node id* to the new node. Next, the server sends a configuration message with the *publisher id*, *writer group id*, and *dataset writer id* to the corresponding publisher. Such information is necessary for OPC UA PubSub brokerless communication in the data exchange phase [14].

TABLE I
The description of Thermometer_01

```
{
    "@context": "https://www.w3.org/2019/wot/td/v1",
    "@type": "saref:TemperatureSensor",
    "id": "localsea:dev:thermometer01",
    "title": "thermometer_01",
    "properties": {
        "temperature": {
            "type": "double",
            "unit": "om:degree_Celsius"
        }
    }
}
```

In the data exchange phase, publishers, with *publisher id*, *writer group id*, and *dataset writer id*, can form UADP network messages. They send each message to the multicast IPv4 address 224.0.0.2 every 10 seconds. The OPC UA server listens to this address, receives the messages, and updates the values of the corresponding OPC UA nodes. The Linux laptop with UaExpert accesses the OPC UA server to monitor the operation of the OPC UA system. Figure 4 is a screenshot of UaExpert. The Address Space window shows the new OPC UA nodes that the OPC UA server dynamically has added to its address space using WoT4UA in the discovery phase. The Data Access View window shows the new updated value of two OPC UA nodes relevant to the temperature of the sensor devices. The Attributes window shows the detailed information related to the OPC UA node with identification "ns=2,i=6002".

## VI. Discussion

WoT4UA-dm is realizable and potential; however, the following issues need further discussion.

First, the vocabulary of the current version of WoT Thing Description is unable to cover OPC UA standards and industrial devices. One example is that the float data type of OPC UA has no corresponding data type in the WoT Thing Description. Another example is that the concept of WoT Thing Description's security schemes is different from OPC UA's security policies. It is necessary to specify a new vocabulary for OPC UA and industrial devices to extend WoT Thing Description. Even better, if both W3C and the OPC foundation accept the vocabulary as a companion specification.

Second, since WoT4UA-dm is for the IIoT, the vocabulary must support resource-constrained embedded devices. It means

[5]https://www.raspberrypi.com/
[6]https://open62541.org/
[7]http://opcfoundation.org/UA-Profile/Transport/pubsub-udp-uadp
[8]https://www.unified-automation.com/downloads/opc-ua-clients.html

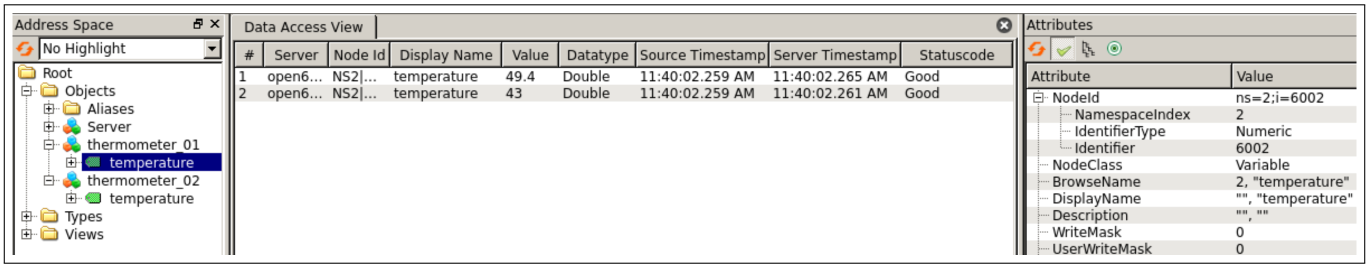


Fig. 4. Screenshot of UaExpert after receiving UADP messages from publishers

that the vocabulary must be packed and precise so that the descriptions of such devices take a small amount in the memory. As a result, registration messages will be smaller and have less impact on the network traffic.

Third, manually writing descriptions for devices is a painful job, for example, when a system contains a significant number of devices, as in the IIoT. It is necessary to have a user-friendly user interface tool that automatically generates descriptions and installs them on corresponding devices.

Fourth, the WoT4UA-dm demands the OPC UA server be also a configurator to exchange registration and configuration messages with field devices. It is tricky for the existing OPC UA server available in the market to add this feature. One possible solution is to have a separate configurator that occupies only the configuration feature. It communicates to the OPC UA server with OPC UA services.

While the first and second issues are mandatory, the two last ones aim to support the community deploying and integrating the mechanism easier.

## VII. Conclusion and Future Works

To sum up, this work-in-progress paper presents a discovery mechanism for OPC UA systems in the IIoT, called the WoT4UA-dm. The discovery mechanism adopts the idea of the WoT Discovery self-description exploration mechanism, that is, to encourage field devices to expose themselves using their self-description. The description follows the format and the vocabulary defined in WoT's Thing Description. Then, the OPC UA server dynamically updates its address space with new OPC UA nodes corresponding to received descriptions. The proof-of-concept experiment in this paper is a minimal scenario to show that the discovery mechanism is realizable.

Many future research and experiments remain before this discovery mechanism can turn into practice. Our short-term plan addresses the first three issues presented in Section VI. The first work is to form a vocabulary, such as a new ontology for OPC UA standards and industrial devices. The vocabulary follows the concept of WoT Thing Description. The second work is to develop a practical use case that is complex enough to highlight the advantages of WoT4UA-dm in the IIoT. The development of the vocabulary and the use case can occur parallel using a hybrid engineering methodology as in [15]. Also, we will develop a plugin for Eclipse Papyrus[9] to have a user-friendly tool for generating descriptions.

## References


[1] OPC Foundation, "OPC Unified Architecture - Part 1: Overview and Concepts," OPC Foundation, Industry Standard Specification OPC 10000-1, 2017.

[2] OPC Foundation, "OPC Unified Architecture - Part 3: Address Space Model," OPC Foundation, Industry Standard Specification OPC 10000-3, 2017.

[3] OPC Foundation, "OPC Unified Architecture - Part 5: Information Model," OPC Foundation, Industry Standard Specification OPC 10000-5, 2017.

[4] T. Oksanen, R. Linkolehto, and I. Seilonen, "Adapting an industrial automation protocol to remote monitoring of mobile agricultural machinery: a combine harvester with IoT," *IFAC-PapersOnLine*, vol. 49, no. 16, pp. 127–131, 2016.

[5] OPC Foundation, "OPC Unified Architecture - Part 4: Services," OPC Foundation, Industry Standard Specification OPC 10000-4, 2017.

[6] S. Rohjans, K. Piech, and S. Lehnhoff, "UML-based modeling of OPC UA address spaces for power systems," in *2013 IEEE International Workshop on Inteligent Energy Systems (IWIES)*. Vienna, Austria: IEEE, November 2013, pp. 209–214.

[7] F. Pauker, T. Frühwirth, B. Kittl, and W. Kastner, "A Systematic Approach to OPC UA Information Model Design," *Procedia CIRP*, vol. 57, pp. 321–326, 2016.

[8] OPC Foundation, "OPC Unified Architecture - Part 12: Discovery and Global Services," OPC Foundation, Industry Standard Specification OPC 10000-12, 2018.

[9] T. Arai, Y. Aiyama, Y. Maeda, M. Sugi, and J. Ota, "Agile Assembly System by Plug and Produce," *CIRP Annals*, vol. 49, no. 1, pp. 1–4, January 2000.

[10] S. Profanter, K. Dorofeev, A. Zoitl, and A. Knoll, "OPC UA for plug & produce: Automatic device discovery using LDS-ME," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, September 2017, pp. 1–8.

[11] L. Dürkop, J. Imtiaz, H. Trsek, L. Wisniewski, and J. Jasperneite, "Using OPC-UA for the autoconfiguration of real-time Ethernet systems," in *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, July 2013, pp. 248–253.

[12] K. Dorofeev, C.-H. Cheng, M. Guedes, P. Ferreira, S. Profanter, and A. Zoitl, "Device adapter concept towards enabling plug & produce production environments," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, September 2017, pp. 1–8.

[13] X. Ye, J. Jiang, C. Lee, N. Kim, M. Yu, and S. H. Hong, "Toward the Plug-and-Produce Capability for Industry 4.0: An Asset Administration Shell Approach," *IEEE Industrial Electronics Magazine*, vol. 14, no. 4, pp. 146–157, December 2020.

[14] OPC Foundation, "OPC Unified Architecture - Part 14: PubSub," OPC Foundation, Industry Standard Specification OPC 10000-14, 2018.

[15] Q.-D. Nguyen, C. Roussey, M. Poveda-Villalón, C. d. Vaulx, and J.-P. Chanet, "Development Experience of a Context-Aware System for Smart Irrigation Using CASO and IRRIG Ontologies," *Applied Sciences*, vol. 10, no. 5, p. 1803, March 2020.


[9] https://www.eclipse.org/papyrus/